Review Quality Aware Collaborative Filtering

Sindhu Raghavan
The University of Texas at Austin
sindhu@cs.utexas.edu

Suriya Gunasekar*
The University of Texas at Austin
suriya@utexas.edu

Joydeep Ghosh
The University of Texas at Austin
ghosh@ece.utexas.edu

ABSTRACT

Probabilistic matrix factorization (PMF) and other popular approaches to collaborative filtering assume that the ratings given by users for products are genuine, and hence they give equal importance to all available ratings. However, this is not always true due to several reasons including the presence of opinion spam in product reviews. In this paper, the possibility of performing collaborative filtering while attaching weights or quality scores to the ratings is explored. The quality scores, which are determined from the corresponding review data are used to "up–weight" or "down–weight" the importance given to the individual rating while performing collaborative filtering, thereby improving the accuracy of the predictions. First, the measure used to capture the quality of the ratings is described. Different approaches for estimating the quality score based on the available review information are examined. Subsequently, a mathematical formulation to incorporate quality scores as weights for the ratings in the basic PMF framework is derived. Experimental evaluation on two product categories of a benchmark data set from Amazon.com demonstrates the efficacy of our approach.

Categories and Subject Descriptors

I.2.8 [**Information Systems**]: Database Applications - Data Mining; I.2.7 [**Computing Methodologies**]: Natural Language Processing - Text Analysis

General Terms

Algorithms, Experimentation

Keywords

Recommender Systems, Probabilistic Matrix Factorization, Collaborative Filtering, Review Quality

*The first and the second authors have contributed equally to the paper.

RecSys'12, September 9–13, 2012, Dublin, Ireland, UK.

1. INTRODUCTION

Collaborative filtering (CF), popularly used in recommendation systems, involves the task of predicting the missing scores or ratings in a user-item matrix by collecting preference information from similar users and/or items. The underlying assumption of the CF approach is that those who have agreed in the past, tend to agree again in the future. Such systems are widely deployed in various domains including movie/music recommendation (Pandora[1], Netflix[2]) and product recommendation by several online retailers like Amazon.com[3] and eBay.com[4].

Probabilistic matrix factorization (PMF) [25], which is one of the popular approaches to collaborative filtering infers latent factors of both users and items and estimates ratings based on the interaction of user and item factors. PMF assumes that the ratings given by users for products are genuine, hence it gives equal importance to all available ratings. However, this assumption does not always hold due to several reasons. Often, users unhappy with a seller tend to give a poor rating for the product, which does not necessarily reflect on the quality of the product. Other times, some sellers might deliberately give superior ratings to promote their products, or they might give unjust poor ratings to competitors' products. The presence of such spurious ratings could impact the performance of the underlying collaborative filtering technique [19].

In this paper, the possibility of performing collaborative filtering while attaching quality scores to the ratings is explored. Most online retailers allow users to provide reviews of products in natural language text. Further, some websites also allow users to provide feedback on how useful the reviews have been. We believe that it is possible to assess the quality of a review/rating using this additional information. The quality score can then be used to "up–weight" or "down–weight" the importance given to the individual rating in the user-item matrix. As a result, ratings with a lower quality score will have lower impact on the predicted scores, thereby improving the accuracy of the predictions.

Our approach to collaborative filtering using quality scores consists of two stages. The first stage involves estimating the quality scores for individual ratings in the data set. In order to quantify the quality of ratings, we use the "review helpfulness" score defined by Kim et al. [13], which uses the feedback information provided by the users for individual re-

[1]http://www.pandora.com
[2]https://www.netflix.com
[3]http://www.amazon.com
[4]http://www.ebay.com

views. This measure provides a reasonable indication of the quality when the amount of feedback is high. However, for more recent reviews that have less feedback, this score might not necessarily capture the true quality of the rating. For such reviews, the quality score is estimated using a regression model trained on reviews that have sufficient feedback. A variety of features extracted from the review text as well as user and review metadata information are used to train the regression model. In the second stage, the quality scores estimated from the previous stage are used as weights for the ratings in the probabilistic matrix factorization framework.

The novelty of our approach lies in the integration of quality scores based on product reviews with collaborative filtering to improve the performance of recommender systems. On the one hand, there is a large body of work on the analysis of online product reviews, especially in the area of assessing the helpfulness of online reviews [13, 7, 16] as well detecting opinion spam [11, 15, 28, 22]. On the other hand, there is also a fair amount of work in the area of collaborative filtering for recommender systems [14, 30, 27] using various approaches including PMF [25] and its Bayesian variant [26]. To the best of our knowledge, this is the first paper that tries to combine online product review helpfulness with collaborative filtering to improve the overall performance of recommender systems. The efficacy of our approach is demonstrated on two product categories from a benchmark data set from Amazon.com.

The rest of the paper is organized as follows. Related work in the area of opinion spam detection and collaborative filtering is reviewed in Section 2. In Section 3, the mathematical formulation of our model and the two stage approach to collaborative filtering using quality scores are described. The experimental methodology used to evaluate the performance of our approach is described in Section 4. Finally, the results of our experiments are discussed in Section 5.

2. RELATED WORK

Kim et al. [13] proposed a quantitative measure based on the review feedback information to assess the helpfulness of reviews. They trained a regression model using various features extracted from the review text and predict the helpfulness score for new reviews. O'Mahony and Smyth [21] modeled the same problem as a classification task. Rather than predicting a score for helpfulness, they trained a classifier using reputation, content, social, and sentiment based features derived from user and item metadata to classify a review as helpful or unhelpful. Danescu-Niculescu-Mizil et al. [7] studied the correlation of different aspects of review metadata with review helpfulness. The results of their study showed that there is a strong correlation between the signed deviation of the review rating to the average rating of the product. Ghose and Ipeirotis [8] model the helpfulness of reviews as a function of the user subjectivity in the reviews. The user subjectivity is in turn predicted using a classifier trained on reviews that are subjective and objective.

There are several approaches proposed in the literature for opinion spam detection [11, 15, 22, 28]. Jindal and Liu [11] trained a classifier based on user, item, and review metadata to identify different categories of spam in online reviews. Liu et al. [15] proposed a method to detect low quality reviews in order to improve the quality of opinion summarization. They used expensive human annotation for the task of estimating the ground truth. Ott et al. [22] proposed approaches to detect fictitious and imaginative opinions that have been deliberately written to sound authentic. Deceptive spam are not easily noticeable by human readers and hence they cannot be identified by user helpfulness votes, which we consider in our work. Ott et al. acquired 400 samples of spam reviews using Amazon Mechanical Turk[5] and 400 reliable genuine reviews from the Trip Advisor website[6] and trained a classifier using n-gram text features and other linguistic metadata.

There are other approaches in literature that identify spam reviews from the perspective of recommender systems [19, 20, 29]. O'Mahony et al. [20] examined the robustness of various collaborative recommendation techniques in the face of malicious attacks. They derived theoretical results on recommendation accuracy and stability in the presence of malicious agents. Mobasher et. al [19] analyzed various new attack models and their impact on recommendation algorithms through extensive simulation-based evaluation. In a more recent work, Wu et al. [29] proposed a semi–supervised learning algorithm to identify spam reviews/ shillings using user metadata. The spam reviews are then removed from the training set while performing collaborative filtering. However, none of these approaches provide a robust methodology to improve the performance of the recommendation systems in the presence of opinion spam.

The current approaches to recommendation systems are usually classified as content-based [2, 3], collaborative [24, 25, 26], and hybrid recommendations [18, 2, 1]. Content based approaches predict the recommendations from user and item profiles derived from characteristic features of users and items, such us demographic data and product descriptions. An alternative approach to recommendation that is heavily used when rich user and item information is not available is collaborative filtering (CF). CF makes use of past user preferences to make predictions for the future. CF algorithms try to identify similarities between users and items to predict user preferences. There are several memory based and model based approaches to collaborative filtering for recommender systems [27, 14, 25]. The most successful methods for CF are the latent factor models based on probabilistic matrix factorization (PMF) [25, 26]. The PMF model is described in more detail in Section 3. Yifan, Koren, and Volinsky [10] proposed an approach that uses implicit feedback about users likes and dislikes(as produced in signed networks), to assign weights for the raw ratings obtained.

3. MODEL

The two–stage model for performing collaborative filtering with quality scores is proposed here. In *Stage 1*, the quality scores of ratings using the review and user data are estimated. In *Stage 2*, these quality scores are used as weights assigned to ratings and *weighted probabilistic matrix factorization* is performed on the ratings to predict new recommendations.

The following notation is used in the rest of the paper. There are n users and m items in the system. The users are indexed by $i \in \{1, 2, \ldots n\}$ and items by $j \in \{1, 2, \ldots, m\}$. The user-item rating matrix is represented as $Y \in \mathbb{R}^{n \times m}$, where y_{ij} represents the rating given by the ith user to jth item. Given the sparsely populated matrix Y, the task of

[5]https://www.mturk.com
[6]http://www.tripadvisor.com

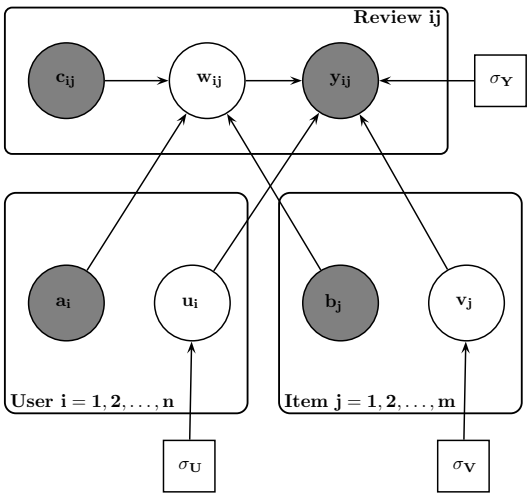

Figure 1: Graphical model for the two stage approach to recommender systems

CF is to estimate the missing entries of Y. To perform CF, the probabilistic matrix factorization approach is used, in which the rating matrix Y, is approximated as a product of two low rank matrices, $U \in \mathbb{R}^{n \times d}$ and $V^T \in \mathbb{R}^{m \times d}$, which represent the user and item latent factors respectively. The latent factor vector of user i is denoted as $\mathbf{u_i}$ and that of item j as $\mathbf{v_j}$. Independent Gaussian priors are used for user and item latent factors. With each rating, y_{ij}, a quality score, w_{ij} is associated, which is estimated from the corresponding review. The user and item meta data and review based features, which are used in the estimation of quality scores are represented by $\mathbf{a_i}$, $\mathbf{b_j}$ and $\mathbf{c_{ij}}$ respectively. The standard deviations associated with the user and item latent factors are given by σ_U and σ_V respectively. Finally, The standard deviation associated with the model for ratings is given by σ_Y. The graphical model for our approach is given in Figure 1.

3.1 Stage 1: Quality Score Estimation

Stage 1 involves estimating the quality score for individual ratings. The quality score for a rating is reflective of the authenticity of the rating. Most websites like Amazon and Yelp allow users to indicate if reviews were helpful or not. The amount of positive feedback obtained by a product review is indicative of the authenticity of the corresponding rating. As a result, any measure that uses the feedback information is useful for measuring the quality of the rating. Kim et al. [13] have proposed the measure below to quantify helpfulness for online product reviews based on the feedback information, which we use as the quality score.

$$\text{helpfulness} = \frac{\text{Number of helpful votes}}{\text{Total number of votes}} \quad (1)$$

The quality score computed as described above is a fair indication of quality if the amount of feedback is sufficiently high. However, for more recent reviews that have low feedback, this score might not necessarily capture the true quality of the rating. For such reviews, the quality score is estimated using a regression model trained on those reviews that have sufficient feedback.

First, a regression model is trained using features extracted from those reviews/ratings that have sufficient feedback. The quality score computed using the formula described above is used as the response variable in the regression model. For a review/rating that has low user feedback, the quality score is predicted using the trained regression model. Several features were examined for training the regression model:

- **Text based features**
  In most online websites, ratings are accompanied by reviews written by the users in natural language text. Several different features were extracted from the review text for training the regression model:

  1. *Unigram counts or bag-of-words features*
     Kim et al. [13] have demonstrated that the unigram counts or bag-of-words based features have been very useful for predicting the helpfulness score. In our approach also, bag-of-words features were extracted from the review text to train a regression model.
  2. *Features from topic modeling*
     Each review can be seen as being composed of words from several different latent topics. For instance, words like *amazing*, *awful*, *terrible*, etc. mainly convey a user's subjective opinion about the product. On the other hand, words that describe technical features of the product convey a more objective opinion about the product. Typically, reviews that are less genuine, and hence less useful are dominated by words from the former topic, while the more useful reviews have a larger proportion of words from the latter topic. The overall quality of the review could be influenced by the distribution of such latent topics. One approach to discovering these *latent topics* in natural language text involves using techniques from topic modeling like Latent Dirichlet Allocation (LDA) [5]. In our approach, LDA was applied to discover latent topics in the review text. The latent topic probabilities were then used as features to train the regression model.

- **Metadata based features**
  Along with the natural language text, most reviews are associated with various other information about the user as well as the product, which were used as features to train the regression model. Specifically, *the average rating given to the user*, which indicates how useful his reviews have been, *duration for which the review has been around*, *deviation of the rating from the mean rating of the product*, *length of the title of the review*, and *length of the review text* were used as features in our approach. Note that item based metadata information was not used as features in the regression model since item related features do not necessarily impact the quality of the review.

- **Text and metadata based features**
  In our third model, features extracted from both natural language text and metadata were used to train a regression model.

3.2 Stage 2: Collaborative Filtering

In Stage 2, the quality scores estimated in Stage 1 are used to build a recommendation system based on collaborative filtering. Among the methods used for collaborative

filtering, latent factor models have been shown to give the best performance in most scenarios. Thus for our analysis, we adapt the probabilistic matrix factorization framework [25] to incorporate quality scores. This method can be trivially extended to other matrix factorization based models.

The PMF model aims at inferring latent factors of users and items from the available ratings. The missing ratings are estimated based on the interaction of user and item latent factors. These factors represent various hidden dimensions of users' tastes and preferences. The $n \times m$ matrix of ratings, Y, is approximated as $Y = UV^T$, where $U \in \mathbb{R}^{n \times d}$ and $V \in \mathbb{R}^{m \times d}$. The priors for U and V are assigned as follows:

$$P(U|\sigma_U^2) = \prod_{i=1}^{n} P(\mathbf{u_i}|\sigma_U^2), \quad P(V|\sigma_V^2) = \prod_{j=1}^{m} P(\mathbf{v_j}|\sigma_V^2) \quad (2)$$

where,

$$\begin{aligned} P(\mathbf{u_i}|\sigma_U^2) &= \mathcal{N}(\mathbf{u_i}|0, \sigma_U^2 \mathbf{I}) \\ P(\mathbf{v_j}|\sigma_V^2) &= \mathcal{N}(\mathbf{v_j}|0, \sigma_V^2 \mathbf{I}) \end{aligned} \quad (3)$$

$\mathcal{N}(x|\mu, \sigma^2)$ represents the Gaussian distribution with mean μ and variance σ^2 evaluated at x. For vector valued variables, $\mathcal{N}(\mathbf{x}|\mu, \Sigma)$ represents the multivariate Gaussian distribution with mean μ and variance Σ evaluated at $\mathbf{x}$. Also, σ_U^2 and σ_V^2 are the variances associated with user and item latent factors respectively.

In a traditional PMF setting, the rating matrix Y is modeled as:

$$P(Y|U, V) = \prod_{i=1}^{n} \prod_{j=1}^{m} \left[\mathcal{N}(y_{ij}|\mathbf{u_i}^T \mathbf{v_j}, \sigma_Y^2)\right]^{I_{ij}} \quad (4)$$

where I_{ij} is the indicator variable indicating if the rating y_{ij} is available and σ_Y^2 is the variance associated with the model for the ratings.

We now modify the existing collaborative filtering framework to weight the ratings using the quality scores estimated in Stage 1. The quality score is modeled as a factor which inversely affects the variance of the prediction from the mean of the factor model. The intuition is that higher quality ratings are given a prior with lower deviations from the model and thus their deviations from the model mean are more heavily penalized. On the other hand, low quality scores are allowed larger deviations from the model mean. We keep the same priors for U and V, as in Equation 3. Our new prior on Y is given by:

$$P(Y|U, V) = \prod_{i=1}^{n} \prod_{j=1}^{m} \left[\mathcal{N}\left(y_{ij}|\mathbf{u_i}^T \mathbf{v_j}, \frac{\sigma_Y^2}{w_{ij}}\right)\right]^{I_{ij}} \quad (5)$$

where I_{ij} is the indicator variable indicating if the rating y_{ij} is available.

Maximizing log posterior of observed Y in this model leads to the minimization objective given below, which follows an intuitive interpretation of minimizing the weighted squared error of the observed ratings, regularized appropriately:

$$L(\theta) = \sum_{i,j} I_{ij}[w_{ij}(y_{ij} - \mathbf{u_i}^T \mathbf{v_j})^2] + \lambda_1 \|U\|_F^2 + \lambda_2 \|V\|_F^2 \quad (6)$$

where $\lambda_1 = \sigma_Y^2/\sigma_U^2$ and $\lambda_2 = \sigma_Y^2/\sigma_V^2$ are the regularization parameters and $\|\|_F$ is the Frobenius norm of a matrix. In our experiments we take $\lambda_1 = \lambda_2 = \lambda$. The above objective is a differentiable function in u_i and v_j and the maximization can be performed using the Stochastic Gradient Descent (SGD) algorithm.

	Books	**Audio CDs**
Total Users	772674	46491
Total Items	493991	29477
No. Ratings in Training	1677892	75518
No. Ratings in Validation	41081	902
No. Ratings in Test	105285	2683

Table 1: Various statistics about the data sets used in experimental evaluation.

4. EXPERIMENTS

4.1 Data set

Amazon.com is an online retailer that sells products like books, movies, furniture etc. Amazon widely employs recommender systems to recommend products to users based on the user's purchase and rating history. In our work, we use the open source data set provided by Jindal and Liu [11]. The categories of *Books* and *Audio CDs* were used for experimental evaluation as they have a reasonable number of users, products, and reviews. The other categories had a very small number of reviews, and hence were not used in our experiments. Each data set consists of three types of information:

1. Review information consisting of user ID, product ID, time stamp, text and title of the review text, rating, and feedback statistics in the form of helpful and unhelpful votes by users.

2. User information consisting of explicit user metadata such as name, location, and a few derived statistics based on the number of reviews written by the user, rank of user, etc.

3. Product information consisting of product specific details like category, brand, price, description, etc.

4.2 Data Preprocessing

Our first preprocessing task was to eliminate multiple reviews for a single user-item pair. This could be due to errors in data transmission on the internet due to which the same review might have appeared multiple times, thereby resulting in duplicate reviews. Other times, it could just be that the user's opinion of a product might have changed over time. In the event that multiple such reviews were present, the latest one was retained and the rest were discarded. Secondly, the entire set of ratings was split into training, validation, and test sets for the recommendation system based on time. The reviews in validation appeared after the ones in training and the reviews in test were posted later than those in validation. Finally, to avoid cold start scenarios, those entries from validation and test set for which either the user or the item was not seen in the training set were removed. Table 1 gives details about the two data sets used in experimental evaluation.

4.3 Regression

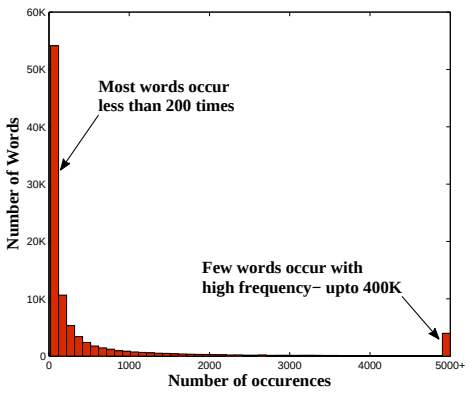

Figure 2: Histogram of words in the reviews in the Books data set.

Regression models were trained for predicting quality scores using three different sets of features – text, metadata, text and metadata. For the Books data set, all those reviews in the training set that had feedback from more than 50 users were used for training the regression model. Since the training set for Audio CDs had reviews with fewer users providing feedback, reviews that had feedback from more than 20 users were used for training the regression model. The methodology used to extract different types of features for training the regression model is described below:

- **Extracting Bag-of-Words features**
  The SRI Language Modeling Toolkit[7] was used to extract unigram counts for bag-of-words features. Commonly occurring stop words were removed while extracting bag-of-words features. Unigram counts were normalized to obtain term frequency values, which were used as features to the regression model. There were around 760,000 unique words in total in the Books data set and around 128,000 unique words in the Audio CDs data set. Following the Zipf's law, a large number of these occurred very rarely in the entire corpus. Figure 2 gives the histogram of words with frequency for the Books data set. The histogram of words from reviews in the Audio CDs data set looked similar. Since a majority of the words occurred very rarely in the corpus and also since quite a few of them were not even valid English words, words that occurred less than 10 times in our corpus were eliminated from the analysis. This reduced the number of features by almost 10 times.

- **Extracting features using LDA**
  David Blei's LDA implementation[8] was used to identify latent topics in the review text. With the reduced set of words, LDA was scalable on the large number of reviews we dealt with. LDA was run with 5, 10, and 50 latent topics and the latent topic probabilities were used as features to train the regression model. The resulting models are called LDA_5, LDA_{10}, and LDA_{50} respectively.

[7]http://www.speech.sri.com/projects/srilm
[8]http://www.cs.princeton.edu/ blei/lda-c/

- **Extracting metadata features**
  Metadata-based features were extracted as described in Section 3.1. *All* metadata features were scaled to a value between 0 and 1 by the transformation suggested in [9], $\tilde{f} = \frac{\log(f+1)}{1+\log(f+1)}$, where f is the original value of the feature.

Regression models were trained using the techniques described below:

- **Logistic Regression**
  Since the quality scores lie in the range 0 to 1, logistic regression was suitable as the predicted values from the logistic regression model lie between 0 and 1. Logistic regression requires the response variable in the training set to be either 0 or 1. An alternate approach to specifying the response variable in logistic regression involves specifying the number of successes and total trials for a given example during training. In our context, this corresponds to specifying the number of users that found the review useful and the total number of users that provided feedback for a given review. Logistic regression models were trained by specifying the response variable in terms of the number of users that provided useful feedbacks and total number of users that provided feedback using features extracted from LDA and metadata.

- **Support Vector Regression**
  Support Vector Regression (SVR) [12] was used for bag-of-words features extracted from natural language text since it can handle any number of features and feature vectors of arbitrary size. One issue with SVR is that it does not guarantee to predict a value between 0 and 1 for the test example. To overcome this, quality scores of the reviews were mapped to real values in $\mathbb{R}$ using the inverse logistic function. The mapped scores were used as response variables to train the SVR model. The predicted value for a test sample was then passed through the logistic function to get the quality score. The linear kernel was used since it performed the best in our preliminary experiments. Note that SVR was used only when feature vectors were not of fixed size, like in bag-of-words, since the implementations of logistic regression and LASSO used in our experiments were inefficient for large feature vectors due to the lack of support for sparse vector representation.

- **LASSO Regression**
  For features extracted from metadata and LDA, a regression model was trained using lasso regression since it helps identify more useful features from the entire set. Like in SVR, quality scores of the reviews were mapped to real values in $\mathbb{R}$ using the inverse logistic function, and the predicted scores were later mapped back to get a value between 0 and 1. The python interface for LASSO in Sklearn package [23] was used in our experimental evaluation.

4.4 Collaborative Filtering

For the baseline estimate, the default implementation of PMF in Graphlab [17], which uses alternating least squares method to perform factorization was used. The number of latent factors was set to $D = 40$. Grid search was performed to tune the regularization parameter λ using the validation

Model	Books	Audio CDs
LR-metadata	0.24	0.25
LR-metadata+LDA_5	0.23	0.24
LR-metadata+LDA_{10}	0.23	0.23
LR-metadata+LDA_{50}	-	0.24
LASSO-metadata	0.30	0.25
LASSO-metadata+LDA_5	0.26	0.25
LASSO-metadata+LDA_{10}	0.26	0.23
LASSO-metadata+LDA_{50}	-	0.26
SVR-bag-of-words	0.27	0.35
SVR-metadata+bag-of-words	0.24	0.27

Table 2: RMSE for 10-fold cross validation on the training set for different regression models in Stage 1. "LR", "LASSO", and "SVR" refer to the models in which quality scores are predicted using logistic regression, LASSO regression, and support vector regression respectively.

data set over the range of 0.01 to 0.8. Note that the weighted PMF approach defined in Section 3.2 has the same objective as the one formulated in [10], though the latter derives the model for a system that has implicit feedback about users likes and dislikes for certain items. The implementation of the latter objective in Graphlab was used to build our new recommendation system. The running time of this implementation is the same as that of the default implementation of PMF in Graphlab.

For the first baseline model, which we refer to as "vanilla PMF", probabilistic matrix factorization was performed by setting weights for all reviews to 1. For the second baseline, which we refer to as "second baseline", quality scores were estimated from the feedback votes for those reviews that had sufficient feedback (50 or more for Books and 20 or more for Audio CDs). For the remaining reviews, the weights were set to the average of the scores that were estimated using feedback information in the previous step and probabilistic matrix factorization was performed. Initial experiments with these two models showed marginal improvement in the performance of latter over former. This observation supported the hypothesis that incorporating quality scores might improve the performance of the recommender system.

5. RESULTS AND DISCUSSION

5.1 Quality Score Estimation

10-fold cross validation was performed on the training set and root mean square error (RMSE) was computed to measure the performance of regression models. Table 2 shows the cross validation RMSE scores for different regression models. Note that due to computational complexity, LDA_{50} did not run till completion on the Books data set. Hence, results for LDA_{50} on the Books data set are not reported. Logistic regression models trained on both LDA and metadata based features perform the best on both data sets. In general, the performance of LASSO and logistic regression models trained only on LDA features is inferior to that of the models trained on both metadata and LDA features. Hence, we do not report results for LASSO and logistic regression models trained only on LDA features. The performance of logistic regression models trained only on metadata features is only slightly inferior to that of the best performing models, thereby indicating that the improvements obtained by adding LDA features are not be substantial. For a given set of features, logistic regression generally performs better than LASSO regression. SVR models trained on bag-of-words features generally perform poorly when compared to the other models. Here again, the performance of the SVR model trained only on bag-of-words features is substantially inferior to that of the model trained on both metadata and bag-of-words features. These results indicate that there might be a stronger signal in metadata based features for predicting the quality of reviews when compared to the current set of text features considered.

5.2 Collaborative Filtering

Table 3 shows the RMSE for different models on the test set for both Books and Audio CDs. On the Books data set, all models including the second baseline outperform the vanilla PMF, while on Audio CDs data set, a majority of the models outperform the vanilla PMF. Lack of sufficient data in the Audio CDs data set could possibly be the reason for inferior performance of some models. Due to lack of reviews with sufficient feedback, less reliable reviews were used to train the regression models in Stage 1. As a result, the accuracy of the scores predicted by Stage 1 could be inferior, thereby impacting the overall quality of the predictions on the Audio CDs data set.

Logistic regression model trained on metadata features is the best performing model on the Books data set and it results in an improvement of 0.0355 (2.49%) over vanilla PMF and 0.0344 (2.41%) over the second baseline. However, on the Audio CDs data set, SVR trained on metadata and bag-words features is the best performing model, with a performance improvement of 0.0175 (1.27%) over vanilla PMF and 0.0128 (.93%) over the second baseline. In general, models trained only on metadata based features perform better than those trained on both LDA and metadata based features indicating strong signals from the metadata features used – time stamp, length of review text, length of review title, rank of the user and deviation of the rating from the mean rating of the product. Even though models trained with text based features are outperformed by the metadata based models, they still show significant improvement over the baseline models on both data sets, which warrants further investigation into linguistic feature engineering on review text. Overall, our results indicate that incorporating quality scores as weights for ratings in collaborative filtering improves the performance of recommender systems.

In our last experiment, the impact of using ratings with low quality scores in training the PMF was studied. Our hypothesis was that ratings with poor quality scores could possibly affect the predictions adversely, and hence eliminating them during training might further improve the performance of recommender systems. Figures 3 shows the distribution of quality scores from LR-metadata, which is the best performing model on the Books data set. While most of ratings have reasonably high quality scores, a small number of them have fairly poor quality scores. Analysis of the distribution of quality scores from SVR-metadata+bag-of-words, the best performing model on the Audio CDs data set yielded similar results. In our experiment, all ratings with a quality score less than 0.4 were eliminated and PMF was performed with the remaining ratings using the best performing model on both data sets. The results from these experiments, which we call "Best-Model-Low-Quality-Scores-

Model	Books	Audio CDs
Vanilla PMF	1.4230 ($\lambda = 0.35$)	1.3739 ($\lambda = 0.30$)
Second Baseline	1.4219 ($\lambda = 0.35$)	1.3692 ($\lambda = 0.20$)
LR-metadata	**1.3875** ($\lambda = 0.25$)	1.3664 ($\lambda = 0.25$)
LR-metadata+LDA_5	1.3972 ($\lambda = 0.20$)	1.3740 ($\lambda = 0.25$)
LR-metadata+LDA_{10}	1.3966 ($\lambda = 0.25$)	1.3779 ($\lambda = 0.25$)
LR-metadata+LDA_{50}	-	1.3731 ($\lambda = 0.25$)
LASSO-metadata	1.3910 ($\lambda = 0.30$)	1.3662 ($\lambda = 0.30$)
LASSO-metadata+LDA_5	1.3952 ($\lambda = 0.30$)	1.3634 ($\lambda = 0.30$)
LASSO-metadata+LDA_{10}	1.3958 ($\lambda = 0.30$)	1.3745 ($\lambda = 0.20$)
LASSO-metadata+LDA_{50}	-	1.3680 ($\lambda = 0.30$)
SVR-metadata+bag-of-words	1.4135 ($\lambda = 0.30$)	**1.3564** ($\lambda = 0.30$)
SVR-bag-of-words	1.4219 ($\lambda = 0.30$)	1.3740 ($\lambda = 0.30$)
Best-Model-Low-Quality-Scores-Dropped	1.3945 ($\lambda = 0.25$)	**1.3389** ($\lambda = 0.30$)

Table 3: Test RMSE for Books and Audio CDs data sets in Stage 2. "LR", "LASSO", and "SVR" refer to the models in which quality scores are predicted using logistic regression, LASSO regression, and support vector regression respectively.

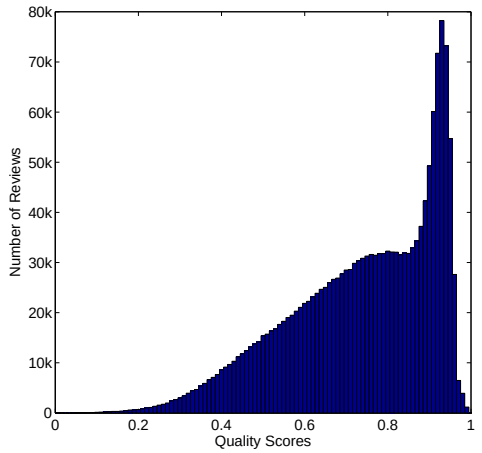

Figure 3: Distribution of quality scores from LR-metadata, the best performing model on the Books data set.

Dropped" are shown in Table 3. Eliminating low quality scores improved the results on the Audio CDs data set considerably, thereby supporting our original hypothesis. However, the performance on the Books data set dropped marginally.

5.3 Quality Indicators

Regression models were analyzed to identify features that impacted the quality of the rating. First, coefficients learned from LASSO and logistic regression on metadata features were examined on the two data sets. Both the regression models had learned similar coefficients for individual metadata features. On both data sets, the review length had the highest positive coefficient, while the deviation of the ratings from the mean rating had the highest negative coefficient from both models. These observations can be intuitively explained as longer reviews are indicators of a thorough analysis of the product by the reviewer and hence the reviewer's rating is highly reliable. On the other hand, a reviewer giving a rating that is highly deviant from the mean rating is likely to be a spammer with a malicious intention of either boosting or degrading a product popularity and hence the negative correlation with the quality. The other features like time stamp and review title length were found to be not very influential in estimating the quality as both regression models assigned low or near-zero coefficients to these features.

Next, regression coefficients learned using LASSO and logistic regression on the topics induced by LDA were examined on the Books data set. Logistic regression assigned more or less the same weights for all topics induced by LDA. However, LASSO regression was able to assign different weights to different topics induced by LDA. There were two topics induced by LDA_5 that had negative coefficients. Some of the words from the former topic included *information*, *good*, *great*, *excellent*, *guide*, *books*, while the words from the latter topic included *history*, *book*, *war*, *world*, *people*, *american*. The remaining topics had low or near zero coefficients indicating that they did not play a significant role in determining the quality of the rating. While the words in the former topic indicate strong opinions which could be used to mask the real quality of the products, the words in the latter topic mostly describe different categories of books, which might not necessarily describe the quality of the product. In general, we found that LDA was more inclined to clustering thematic topics together rather than topics that were indicative of quality. This inability to distinguish thematic words from quality indicators is possibly one of the reasons for the modest performance of LDA-based features in our experiments. Our analysis of topics induced by LDA_{10} yielded similar results on the Books data set. Further analysis of words induced by LDA on the Audio CDs data set did not yield any interesting observations. Overall, these results emphasize the need for extraction of more sophisticated features from the review text.

In summary, incorporating quality scores or weights to ratings improves the performance of collaborative filtering in recommender systems. Our experiments with different types of features extracted from both review metadata and text indicate that some of the metadata-based features are highly indicative of the quality of the rating. Further, our experiments with text-based features also demonstrate promise, but also indicate the need for extraction of more sophisticated features from review text. Overall, we find that our two stage approach to collaborative filtering is a robust

method that is capable of overcoming the negative effects caused by spurious reviews and ratings in recommender systems.

6. FUTURE WORK

Future work includes incorporating several additional features including bigrams and semantic features as described in Kim et al. [13] for learning the regression model to predict quality scores. We would also like to explore measures proposed by Ghose and Ipeirotis [8] for assessing the quality of reviews. The other direction of future work involves exploring feature reduction techniques like PCA to reduce the number of bag-of-words features, which we believe could help improve the performance of the regression model. In our current experiments, LDA could not induce latent word distributions that were reflective of the quality of reviews. To help improve the performance, supervised approaches like supervised LDA [4] can be explored in future. Further, to overcome the lack of sufficient reviews in data sets like the Audio CDs data set, transfer learning approaches [6] can be incorporated for the estimation of quality scores in our framework. Finally, experimental evaluation of our approach on other data sets like the Yelp academic data set and other product categories available in the Amazon data set will also be considered in the future.

7. CONCLUSION

In this paper, a two-stage approach to collaborative filtering that incorporates weights or quality scores for ratings is proposed. Several approaches to estimating quality scores using product reviews associated with the ratings are examined. Experimental evaluation of our approach on two product categories from a large benchmark data set from Amazon.com demonstrates that the proposed two-stage approach performs better than the vanilla PMF that assigns equal importance to the ratings. To the best of our knowledge, this is the first paper that has combined assessing review helpfulness with collaborative filtering to improve the overall performance of recommender systems.

Acknowledgements

We would like to thank Raymond J. Mooney, Sreangsu Acharyya, and Gautam Muralidhar for their valuable feedback on the paper.

8. REFERENCES

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

[2] M. Balabanović and Y. Shoham. Fab: content-based, collaborative recommendation. *Communnucation ACM*, 1997.

[3] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. *AAAI*, 1998.

[4] D. Blei and J. McAuliffe. Supervised topic models. *NIPS*, 2008.

[5] D. M. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[6] J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. *ACL*, 2007.

[7] C. Danescu-Niculescu-Mizil, G. Kossinets, J. Kleinberg, and L. Lee. How opinions are received by online communities: a case study on amazon.com helpfulness votes. *WWW*, 2009.

[8] A. Ghose and P. G. Ipeirotis. Designing novel review ranking systems: predicting the usefulness and impact of reviews. *ICEC*, 2007.

[9] C.W. Hsu, C.C. Chang, and C.J Lin. A practical guide to svm classification. Technical report, National Taiwan University, 2003.

[10] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. *ICDM*, 2008.

[11] N. Jindal and B. Liu. Opinion spam and analysis. *WSDM*, 2008.

[12] T. Joachims. Making large-scale support vector machine learning practical. *Advances in kernel methods*, 1999.

[13] S. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. *EMNLP*, 2006.

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

[15] J. Liu, Y. Cao, C. Y. Lin, Y. Huang, and M. Zhou. Low-quality product review detection in opinion summarization. *EMNLP-CoNLL*, 2007.

[16] Y. Liu, X. Huang, A. An, and X. Yu. Modeling and predicting the helpfulness of online reviews. *ICDM*, 2008.

[17] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. *CoRR*, 2010.

[18] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. *AAAI*, 2002.

[19] B. Mobasher, R. D. Burke, R. Bhaumik, and C. Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technologies*, 2007.

[20] M. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technologies*, 2004.

[21] M. P. O'Mahony and B. Smyth. Learning to recommend helpful hotel reviews. *RecSys*, 2009.

[22] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. *ACL-HLT*, 2011.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *JMLR*, 2011.

[24] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. *CSCW*, 1994.

[25] R. Salakhutdinov. Probabilistic matrix factorization. *NIPS*, 2008.

[26] R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. *ICML*, 2008.

[27] X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in Artificial Intelligence*, 2009.

[28] G. Wu, D. Greene, B. Smyth, and P. Cunningham. Distortion as a validation criterion in the identification of suspicious reviews. *SOMA*, 2010.

[29] Z. Wu, J. Cao, B. Mao, and Y. Wang. Semi-sad: applying semi-supervised learning to shilling attack detection. *RecSys*, 2011.

[30] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. *AAIM*, 2008.