Recommender System Incorporating User Personality Profile through Analysis of Written Reviews

Peter Potash
Department of Computer Science
University of Massachusetts Lowell
Lowell, Massachusetts
ppotash@cs.uml.edu

Anna Rumshisky
Department of Computer Science
University of Massachusetts Lowell
Lowell, Massachusetts
arum@cs.uml.edu

ABSTRACT

In this work we directly incorporate user personality profiles into the task of matrix factorization for predicting user ratings. Unlike previous work using personality in recommender systems, we use only the presence of written reviews by users. Other work that incorporates text directly into the recommendation framework focuses primarily on insights into products/categories, potentially disregarding important traits about the reviewers themselves. By using the reviews to determine the users' personalities directly, we can acquire key insights into understanding a user's taste. Our ability to create the personality profile is based on a supervised model trained on the MyPersonality dataset. Leveraging a set of linguistics features, we are able to create a predictive model for all Big 5 personality dimensions and apply it to the task of predicting personality dimensions for users in a different dataset. We use Kernelized Probabilistic Matrix Factorization to integrate the personality profile of the users as side-information. Lastly, we show the empirical effectiveness of using the MyPersonality dataset for predicting user ratings. Our results show that combining the personality model's raw linguistic features with the predicted personality scores provides the best performance. Furthermore, the personality scores alone outperform a dimensionality reduction of the linguistics features.

CCS Concepts

•**Human-centered computing** → **Collaborative filtering;** *Empirical studies in collaborative and social computing;* Social networks;

Keywords

Human-Centered Computing; Collaborative Filtering; Recommender Systems; Social Networks

1. INTRODUCTION

Recent work [20, 2, 1] has shown the effectiveness of incorporating user reviews into the matrix factorization framework. Unfortunately, the information derived from the reviews is primarily used to understand items/item categories, as opposed to users. Given that it is the users who provide the reviews, we believe that there could be important information about the reviewers lost in these methodologies. Even if the methodologies were modified slightly to glean insight into the users themselves, the representations learned by these methodologies still require manual inspection to fully understand their meaning. Alternatively, when it comes to understanding users, personality can be an important concept to leverage – the intersection of personality and linguistics dates back decades [8, 33, 14]. Given that personality is a well-researched topic, it is an interpretable aspect to attempt to derive from written reviews. Furthermore, we believe it can be effective side-information that can be used to produce more accurate predictions.

More specifically, we will use the MyPersonality dataset [18] to build a predictive model to attain the Big 5 Personality traits [13] for reviewers (users). The dataset provides status updates from Facebook users along with users' personality scores that are based on the users taking separate psychological tests. Thus, the personality scores in this dataset are grounded in proven psychological research. We will then take advantage of the Kernelized Probabilistic Matrix Factorization (KPMF) framework to incorporate the personality scores as side-information.

To further motivate the idea of personality profile as an added signal for user rating prediction, take as an example the following excerpts from two different movie reviews for the film 'Inception'. Both of the reviewers rated the movie 10 out of 10, but observe how each user begins his/her review. One reviewer writes:

> "My sister has been bothering me to see this movie for more than two months, and I am really glad that she did, because this movie was excellent, E-X-C-E-L-L-E-N-T, EXCELLENT!"

Whereas the other reviewer notes:

> "So far, Christopher Nolan has not disappointed me as a director, and 'Inception' is another good one."

While the two users have given the same numerical rating to the movie, we can obtain deeper insight into the users them-

EMPIRE 2016, September 16, 2016, Boston, MA, USA.

selves by examining what they wrote. The first reviewer appears to be a more casual moviegoer, seeing movies people recommend, and finding pleasure in them. The second reviewer, in contrast, appears to be more of a movie aficionado. The reviewer immediately identifies who the director is, and indicates that he/she is familiar with the director's work. Such an analysis can indicate that their ratings for other items could diverge substantially.

The rest of this paper is organized as follows. Section 2 provides an overview of the related work on matrix factorization, as well as at the intersection of recommender systems and natural language processing (NLP). Section 3 describes the KPMF methodology. In Section 4, we explain how the predictive model for the Big 5 personality traits was built, as well as how it is incorporated as the side-information format for KPMF. Section 5 describes our experimental design for predicting user ratings that incorporate personality. Finally, in Sectons 6 and 7, we present and discuss our results, as well as future research directions based on this work.

2. BACKGROUND

In this section, we will give a brief review of the history of recommender systems using matrix factorization over the course of the past decade, as well as then discuss examples of previous work where NLP methods have been used to create recommender systems.

2.1 Matrix Factorization Systems

The Netflix Challenge that commenced in 2006 marked a seminal event in the field of recommender systems. As [3] notes, The state-of-the art system that Netflix was using, Cinematch, was based on a nearest-neighbor technique. The system used an extension of Pearson's correlation, which the system produced by analyzing the ratings for each movie. The system then uses these correlation values to create neighborhoods for the movies. Finally, the system uses these correlations in multi-variate regression to produce the final rating prediction.

The team that ultimately took home the million dollar prize, however, relied on a fundamentally different technique: latent factors via matrix factorization [17]. Rather than calculating neighborhoods for items and/or users, matrix factorization models users and items as latent vectors. Stacking these vectors into two separate matrices, one for users and one for items, produces the latent matrices that represent users and items. The models predict ratings simply by taking the dot-product of the latent vectors of the user and item for which it is desired, or simply multiplying the two matrices to predict all ratings.

During the course of the Netflix Challenge, researchers developed probabilistic extensions of standard matrix factorization [26, 27] that could adapt well to large, sparse matrices that are generally representative of rating matrices. These models assume a generative process of probability distributions for the latent user/item vectors, as well as the ratings themselves. Our technique for rating prediction follows the methodology of KPMF, detailed by [36]. KPMF builds upon a probabilistic framework and we will explain the model in full detail in Section 3.

2.2 Recommender Systems and NLP

Various researchers have already completed NLP-related tasks in the overall goal of constructing an effective recommender system. [28] combines topic modeling on plot summaries with probabilistic matrix factorization to predict user ratings for movies. Their paper proposes an expanded generative process for rating prediction that can incorporate the models of Correlated Topic Modeling [5] and Latent Dirichlet Association [6]. In similar fashion, [35] combines topic modeling on the text of scientific article with probabilistic matrix factorization in the effort of recommending relevant articles/papers to researchers. In an example of a non-matrix factorization approach, [29] uses sentiment analysis on movie reviews for movie recommendations. Here, the researchers use a recommendation technique more akin to nearest-neighbors by defining a similarity measure among users and items based on how users rate items and how items are rated. Once the similarity is measured, the researchers use the result of the sentiment analysis to produce their final recommendations. In [10], the authors mine users' written reviews to understand both generalized and context-specific user preferences. These two aspects are then combined into a linear regression-based recommendation system. [11] provides a thorough presentation of the intersection between NLP and recommender systems.

In recent years, researchers have established methodologies that integrate the content of text reviews directly into the matrix factorization framework. In [20, 2], the authors fuse together topic modeling with matrix factorization, allowing models to learn representations of users and items, as well as topical distributions related to items and categories. More recently, in [1], the authors add the modeling of distributed language representations to the matrix factorization framework. This allows the authors to learn individual word representations as well as a general language model for the categories in their dataset.

The work that closely resembles ours is that of [25]. In their work, the authors create a personality-based recommender algorithm for recommending relevant online reviews. The authors train their personality model on a corpus of stream-of-consciousness essays, that include an accompanying personality score for each writer [24]. The authors, unfortunately, do not detail what accuracy their personality model scores on a supervised cross-validation of the dataset. Our own efforts to create a classification model from the same data using similar features produced an accuracy below 60%, which we do not deem accurate enough for use in further applications. Once the authors predicted the users' personalities, they clustered the results together in order to provide recommendations for users. While the approach is relevant, the authors are unable to test their recommendations against a gold-standard. Furthermore, in the effort of generating recommendations, matrix factorization has shown to be more accurate than nearest-neighbor approaches.

2.3 Recommender Systems with Personality

Aside from [25], several other researchers have integrated personality profiles into recommender systems. For example, [31] and [22] both use user personality profiles in the process of generating recommendations. However, the important difference between our work and the work of these researchers is that their methodology requires the explicit completion of personality tests by users. The researchers then derive personality scores directly from these tests. Such requirements make it inconceivable to use these systems in

a large-scale, applied nature. Our work is unique in the fact that we derive personality scores purely from an analysis of the users' written reviews. We require no further action from users aside from allowing them to express their opinion through ratings and reviews. Because of this, we contend that our methodology has the potential for large-scale application.

3. MATRIX FACTORIZATION

As we have previously mentioned, we use the technique of KPMF to incorporate the information that we generate by analyzing a given user's written reviews. What we generate from the analysis is a personality profile for a given user. We conjecture that by including this information of user personality in our model, we can ultimately produce more accurate movie ratings. We acknowledge that the choice of KPMF to incorporate side-information into the matrix factorization framework is somewhat arbitrary, and the work of [7, 15] could potentially be used instead.

3.1 KPMF

For the purpose of this paper we will explain the specifics of KPMF. To understand probabilistic matrix factorization in general and how KPMF is unique in this area, we encourage the reader to refer to the previously cited papers. In KPMF, we assume that the dimensions for the latent vectors representing items and users are drawn from a Gaussian Process (GP). Although in this GP we assume a zero mean function, it is the formulation of the covariance function that allows us to integrate side-information into our model. This covariance function – or covariance matrix in our application – dictates a 'similarity' across the the users and/or items. Our notation will follow the notation the original authors provided. Here is the notation we will use:

R — $N \times M$ data matrix
U — $N \times D$ latent matrix for rows of R
V — $M \times D$ latent matrix for columns of R
K_U — $N \times N$ covariance matrix for rows
K_V — $M \times M$ covariance matrix for columns
S_U — $N \times N$ inverse of K_U
S_V — $M \times M$ inverse of K_V
A — number of non-missing entries in R
$\delta_{n,m}$ — indicator variable for rating $R_{n,m}$

The generative process for KPMF is as follows (refer to Figure 1 for plate diagram):

1. Generate $U_{:,d} \sim GP(\mathbf{0}, K_U)$ for $d \in \{1,...,D\}$
2. Generate $V_{:,d} \sim GP(\mathbf{0}, K_V)$ for $d \in \{1,...,D\}$
3. For each non-missing entry $R_{n,m}$, generate $R_{n,m} \sim \mathcal{N}(U_{n,:} V^T_{m,:}, \sigma)$, where σ is constant

The likelihood of the data matrix R given U and V over the observed entries is:

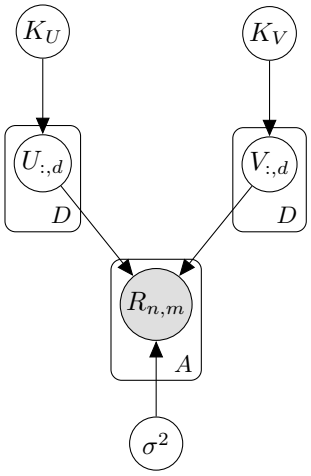

Figure 1: The generative process for KPMF.

$$p(R|U,V,\sigma^2) = \prod_{n=1}^{N} \prod_{m=1}^{M} [\mathcal{N}(R_{n,m}|U_{n,:} V^T_{m,:}, \sigma^2)]^{\delta_{n,m}} \quad (1)$$

Where the prior probabilities over U and V are:

$$p(U|K_U) = \prod_{d=1}^{D} GP(U_{:,d}|\mathbf{0}, K_U) \quad (2)$$

$$p(V|K_V) = \prod_{d=1}^{D} GP(V_{:,d}|\mathbf{0}, K_V) \quad (3)$$

Combining (1) with (2) and (3), the log-posterior over U and V becomes:

$$\begin{aligned} &log\ p(U,V|R,\sigma^2,K_U,K_V) \\ &= -\tfrac{1}{2\sigma^2} \sum_{n=1}^{N} \sum_{m=1}^{M} \delta_{n,m}(R_{n,m} - U_{n,:} V^T_{m,:})^2 \\ &- \tfrac{1}{2} \sum_{d=1}^{D} U^T_{:,d} S_U U_{:,d} - \tfrac{1}{2} \sum_{d=1}^{D} V^T_{:,d} S_V V_{:,d} \\ &- Alog\sigma^2 - \tfrac{D}{2}(log|K_U| + log|K_U|) + C \end{aligned} \quad (4)$$

Where $|K|$ is the determinant of K and C is a constant that does not depend on U and V.

3.2 Learning KPMF

To learn the matrices U and V we can apply a MAP estimate to (4). The result is optimizing the following objective function:

$$\begin{aligned} E = &\tfrac{1}{2\sigma^2} \sum_{n=1}^{N} \sum_{m=1}^{M} \delta_{n,m}(R_{n,m} - U_{n,:} V^T_{m,:})^2 \\ &+ \tfrac{1}{2} \sum_{d=1}^{D} U^T_{:,d} S_U U_{:,d} + \tfrac{1}{2} \sum_{d=1}^{D} V^T_{:,d} S_V V_{:,d} \end{aligned} \quad (5)$$

[36] provides implementations of both gradient descent and stochastic gradient descent to minimize E. For our experiments we used regular gradient descent, as gradient descent achieved the highest accuracy in the original work and our rating matrix is a manageable size. We will note that in the authors' work, the accuracy of stochastic gradient descent was less than that of regular gradient descent by only

a small margin and its speed was hundreds of times faster.

The partial derivatives for our objective function are the following:

$$\frac{\partial E}{\partial U_{n,d}} = -\frac{1}{\sigma^2} \sum_{m=1}^{M} (R_{n,m} - U_{n,:} V_{m,:}^T) V_{m,d} + \frac{1}{2} \boldsymbol{e}_{(n)}^T S_U U_{:,d} \quad (6)$$

$$\frac{\partial E}{\partial V_{m,d}} = -\frac{1}{\sigma^2} \sum_{n=1}^{N} (R_{n,m} - U_{n,:} V_{m,:}^T) U_{n,d} + \frac{1}{2} \boldsymbol{e}_{(m)}^T S_V V_{:,d} \quad (7)$$

where $\boldsymbol{e}_{(n)}$ represents an N-dimensional vector of all zeros except for the n^{th} index, which is one.

The update equations for U and V are as follows:

$$U_{n,d}^{t+1} = U_{n,d}^t - \eta(\frac{\partial E}{\partial U_{n,d}}) \quad (8)$$

$$V_{m,d}^{t+1} = V_{m,d}^t - \eta(\frac{\partial E}{\partial V_{m,d}}) \quad (9)$$

where η is the learning rate of the algorithm.

This completes our detailing of KPMF. In the next section we describe our approach for creating the covariance matrix for the users, K_U.

4. CREATING PERSONALITY PROFILES

Since we are using KPMF as our recommendation model, any vector representation of the written reviews (for a given user, across all users) would suffice to create K_U. However, it is best to generate covariance across a numeric representation that we can interpret. Since personality scores have a long history of analysis, which we will detail in this section, personality profiles are an optimal representation for K_U. In this section we cover two topics: first, how we create the personality profile for a given user. Second, how we use this personality profile to generate the user covariance matrix.

4.1 MyPersonality

In 2013, [9] held a workshop on computational personality recognition. For this workshop, the organizers released a subset of the data collected by the MyPersonality project [18]. The dataset for the workshop consists of the Facebook activity for 250 users, roughly 10,000 status updates from all users. Along with the status updates, the dataset includes information about the users' social networks. For each user, the dataset includes a personality score as well as a binary classification as to whether the user exhibits a given personality trait. The personality scores/classifications for each user have five dimensions, one for each trait in the Big 5 personality model. The five traits in the model are openness, conscientiousness, extraversion, agreeableness, and neuroticism. Analysis of lexicon and personality has a long-standing tradition [8, 33, 13], and it is [14] who brought the current model to prominence.

The approaches to the dataset in the workshop are varied. [32] focus on predicting a single personality trait, conscientiousness. The authors exploit an analysis of event-based verbs in the status updates to produce features for their model. [34] create an ensemble model for predicting personality traits. In their base model, the authors use most frequent trigrams as features. The authors then use the prediction of the baseline model to generate their final predictions. [12] and [19] have similar approaches: using a general textual analysis combined with social network attributes to create features for their predictive models. However, Markoviki et al. report a higher precision/recall for their model, so we will use their approach to feature selection as the guide for our model for personality prediction.

4.2 Personality Model

In their paper, Markoviki et al. detail a fined-grained feature selection for each personality trait, including social network features. Since, for our recommendation experiment, we will not have social network information, we do not include these features in our model. While most authors who used the MyPersonality data sought to create a classification model for personality prediction, we will predict personality score. We believe having a continuous output from our model will make for a better translation into user covariance. Based on an analysis of correlation between features and personality traits in Markoviki et al., we use the following features in our personality model (and we encourage a review of the original work for a thorough discussion of the effectiveness of these features):

Punctuation Count: We count the frequency of the following punctuation marks in a user's status updates: . ? ! - , <> / ; : [ ] { } ( ) & ' " ?

POS Count: We count the frequency of verbs and adjectives appearing in a user's status updates. We used the POS tagger available in NLTK [4].

Affin Count: We count the frequency of words appearing in a user's status updates that have an emotional valence score between -5 and 5 [21].

"To" Count: We count the number of times the word "to" appears in a user's status updates.

General Inquirer Tags: We process the text using the General Inquirer (GI) tool [30]. This tool has 182 categories for tagging words in a text. We use the frequency of these tags for our feature set.

While Markovikij et al. produced their best results when using a different subset of the GI tags for each personality trait, as well as Affin words only of a particular score, we did not find that this fine-grained breakdown produced the best results for our own experiments. Instead, we use the same feature space for all the personality traits, which included all GI tags and all words with any recorded Affin score. Lastly, all count features are normalized by the total word count (for a given user), and punctuation count is normalized by the total character count.

The personality scores are in a continuous range from 1 to 5 for users in the MyPersonality dataset . Thus, linear regression is a natural choice to train our model. We use the Ridge Regression algorithm available from scikit-learn [23]. Ridge Regression implements standard linear regression with a regularization parameter. The optimization task

is:

$$\min_w \|Xw - y\|_2{}^2 + \alpha\|w\|_2{}^2 \quad (10)$$

Where w is the weight vector, X is the data matrix, y is the vector of scores and α is the regularization parameter. The algorithm in scikit-learn performs automatic cross-validation on the regularization parameter by allowing us to define a list of α's for the input. While the feature space for each personality trait is the same, we train a different model for each trait. To be clear, we are not testing the personality of a single status update, but rather of a given user, which is the amalgamation of his/her status updates.

To test the utility of our models, we divide the set of Facebook users into a 80%/20% training/test split. Also, we normalize the matrices we use in our models by, for each feature dimension, subtracting the mean and dividing by the standard deviation. We randomly shuffle the set of users and record the root-mean-square error (RMSE) of the resulting trained model on the held-out test set. That is, given a predicted personality score for user i, $\hat{y_i}$, and the true personality score y_i, we calculate the RMSE of all users in the test set. Table 1 shows the accuracy of our model averaged across 5 different times shuffling the dataset. This model is compared to a baseline, which is the average user rating for personality scores in the training set. When creating the models that we will apply to predicting personality traits from movie reviews, we included all the Facebook users when training the models.

Personality Trait	Model	Baseline
Extraversion	0.785	0.833
Neuroticism	0.738	0.767
Agreeableness	0.635	0.661
Conscientiousness	0.767	0.799
Openness	0.529	0.563

Table 1: RMSE for personality model trained on Facebook statuses, as well as baseline model.

4.3 User Covariance Matrix

Once we have trained the personality models on the Facebook data we apply it to the movie reviews written by a given user to determine his/her personality profile. We preprocess the movie reviews just as we did for the Facebook data to create the same feature space. The result is a 5-dimensional vector, which we will denote p_i, for user i. For users i and j, we calculate entry i, j of K_U as follows:

$$K_{U_{i,j}} = \frac{CS(p_i,p_j)-\alpha}{\beta-\alpha} * \gamma \quad (11)$$

Where $CS(x, y)$ denotes the cosine similarity between vectors x, y, calculated as follows:

$$CS(x,y) = \frac{xy^T}{\|x\|\|y\|} \quad (12)$$

α and β are minimum and maximum values from our computed cosine similarities, across all possible user pairs:

$$\alpha = \min_{i,j} CS(p_i, p_j) \quad (13)$$

$$\beta = \max_{i,j} CS(p_i, p_j) \quad (14)$$

γ controls the ceiling of the normalization: $K_{U_{i,j}} \in [0, \gamma]$. We set $\gamma = 0.4$. To compute cosine similarity we use the cosine similarity method provided in scikit-learn. Note β will always be 1, as $CS(p_i, p_i) = 1$.

This, however, is not the final covariance matrix we will use in our recommender system. Since all the personality scores are in the range [1,5], the cosine similarity between personality vectors p_i and p_j is very close to one. To accentuate the differences in personality profile, we create a regularized covariance matrix, $\overline{K_U}$, as follows:

$$\overline{K_U} = {K_U}^n \quad (15)$$

Where n is a hyperparameter we hand-tune. The proper value of n can greatly influence the accuracy of the model. We take $\overline{K_U}$ as the covariance matrix in our experiment when we use personality profiles to produce the user covariance matrix, but we still refer to it as K_U to avoid confusion.

5. EXPERIMENTAL DESIGN

Our goal is to integrate the information contained in the reviews written by a user into a recommender system, and in particular, investigate whether user personality, as reflected in the text generated by that user, would allow us to improve the accuracy of predicted ratings. We crawled IMDB to collect a dataset of scores and written reviews for multiple IMDB users. Our dataset consists of 2,087 users and 3,500 movies. Each user has rated/reviewed as little as 4 movies and as many as 210, with 54 being the average number of ratings/reviews for the users. The total rating matrix is 1.55% dense, which reflects the typical sparsity of this type of dataset [16].

We randomly split the ratings by each user into training, evaluation, and test sets, each comprising 3/5, 1/5 and 1/5 of the data, respectively. We randomly shuffle the full set of ratings to produce five different training/evaluation/test splits, and report the results averaged over five runs. We use the ratings from these sets to create the appropriate matrices in our methodology. The training matrix is equivalent to R in our notation.

In all the experiments, we use a diagonal item covariance matrix, K_V. Thus, in our model, we are not assuming any covariance across items. Following the results of Zhou et al. we let $D = 10$ and $\sigma = 0.4$. We use gradient descent to learn the latent matrices U and V. We use the proportional change in RMSE on our evaluation matrix as the stopping criteria for gradient descent. Once the algorithm converges, we calculate the RMSE on our test matrix. When calculating RMSE, we only do so for non-zero entries, i.e. $\delta_{n,m} = 1$.

6. RESULTS

For each run, we train five different models and calculate their RMSE on a held-out test set: (1) KPMF with K_U calculated according to user personality profile, (2) KPMF with K_U calculated using a user's text-generated feature space for (10) as our p vector in equation (11), (3) KPMF with K_U

as a diagonal matrix (no similarity across users), (4) matrix factorization (MF) without trying to optimize U and V according to an objective function, and (5) KPMF with a PCA-reduction of the text-based feature space as p. Aside from providing a tangible vector representation of user reviews, the Big 5 personality model also acts as a guided dimensionality reduction of the textual feature space we use to generate personality scores. Therefore, we have compared the 5-dimensional output of our personality model to the result of using PCA to compute a reduction of the text-based feature space to 5 dimensions. We used the PCA implementation from scikit-learn. The RMSE values averaged over five runs for each model are shown in Table 2. For the purposes of RMSE calculation, the rating values in our data, which were originally 1-10, have been normalized to fall in the interval $[0.1, 1]$.

Model	RMSE
KPMF with Personality	0.2006
KPMF with Personality Model Features	0.1980
KPMF Personality *and* Model Features	**0.1901**
KPMF with Diagonal Matrix	0.2122
KPMF with PCA Feature Reduction	0.2087
MF	0.2262

Table 2: RMSE predicting user ratings.

7. DISCUSSION

As we expected, the KPMF models performed better than the non-optimized MF model, lowering the RMSE by 16.0%, 12.5%, 11.3%, 7.7% and 6.2% respectively. Comparing the KPMF models together, the personality model improves upon the diagonal model by 5.5%, however we see that a more accurate model is achieved by applying the textual personality features directly, and the most effective model uses a combination of the textual features and the predicted personality scores. It is important to note the percent difference along with RMSE, especially when the baseline metric performs well. When comparing the two models of 'dimensionality reduction', the personality model performs better than the PCA-model. This would dictate that the personality scores do capture a stronger signal of user similarity, as opposed to an arbitrary reduction of the raw text features. The the personality scores on their own do not perform as well as the raw textual features. We will discuss shortly a major added benefit for using personality scores, aside from testing accuracy.

One immediate question that arises is whether a more accurate personality predictive model actually does correlate to a more accurate KPMF model when using the personality profile. While our personality predictive model scores reasonably well, it is inconsistent across the personality traits. Future work can have a renewed focus on the MyPersonality data now that the recommendation framework has a solid foundation. Furthermore, as we have previously stated, representing users as personality profiles provides a gateway to a number of interesting analyses relating personality to product recommendation. For example, in our current recommendation model, each personality trait is given equal weight when we use the personality model to generate the covariance matrix. However, it is interesting to imagine a model where each personality trait should be weighted differently. For example, similarity in user conscientiousness might be more important than similarity in user agreeableness when determining overall similarity in user preference. We can create a new variable Q, a 5-by-5 diagonal matrix where each entry $Q_{i,i}$ is the weight for a given personality trait. If we stack the personality vectors to form a $M \times 5$ matrix P, the covariance matrix K_U becomes:

$$K_U = PQP^T \tag{16}$$

We can learn the diagonal entries of Q along with U and V in our model. The final values of Q would provide a novel outcome as to how important each personality trait is for predicting movie ratings. We leave this approach for future work.

8. REFERENCES

[1] A. Almahairi, K. Kastner, K. Cho, and A. Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154. ACM, 2015.

[2] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, pages 2–8, 2014.

[3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[4] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[5] D. Blei and J. Lafferty. Correlated topic models. *Advances in neural information processing systems*, 18:147, 2006.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[7] G. Bouchard, D. Yin, and S. Guo. Convex collective matrix factorization. In *AISTATS*, volume 13, pages 144–152, 2013.

[8] R. B. Cattell. Personality and motivation structure and measurement. 1957.

[9] F. Celli, F. Pianesi, D. Stillwell, and M. Kosinski. Workshop on computational personality recognition: Shared task. In *Seventh International AAAI Conference on Weblogs and Social Media*, 2013.

[10] G. Chen and L. Chen. Augmenting service recommender systems by incorporating contextual opinions from user reviews. *User Modeling and User-Adapted Interaction*, 25(3):295–329, 2015.

[11] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, 2015.

[12] G. Farnadi, S. Zoghbi, M.-F. Moens, and M. De Cock. Recognising personality traits using facebook status updates. In *Proceedings of the workshop on computational personality recognition (WCPR13) at the 7th international AAAI conference on weblogs and social media (ICWSM13)*, 2013.

[13] L. R. Goldberg. Language and individual differences: The search for universals in personality lexicons.

Review of personality and social psychology, 2(1):141–165, 1981.

[14] L. R. Goldberg. The development of markers for the big-five factor structure. *Psychological assessment*, 4(1):26, 1992.

[15] S. Gunasekar, M. Yamada, D. Yin, and Y. Chang. Consistent collective matrix completion under joint low rank structure. In *AISTATS*, 2015.

[16] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[17] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[18] M. Kosinski, D. Stillwell, and T. Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

[19] D. Markovikj, S. Gievska, M. Kosinski, and D. Stillwell. Mining facebook data for predictive personality modeling. In *Proceedings of the 7th international AAAI conference on Weblogs and Social Media (ICWSM 2013)*, 2013.

[20] J. McAuley and J. Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[21] F. Å. Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*, 2011.

[22] M. A. S. N. Nunes. *Recommender systems based on personality traits*. PhD thesis, Université Montpellier II-Sciences et Techniques du Languedoc, 2008.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] J. W. Pennebaker and L. A. King. Linguistic styles: language use as an individual difference. *Journal of personality and social psychology*, 77(6):1296, 1999.

[25] A. Roshchina, J. Cardiff, and P. Rosso. A comparative evaluation of personality estimation algorithms for the twin recommender system. In *Proceedings of the 3rd international workshop on Search and mining user-generated contents*, pages 11–18. ACM, 2011.

[26] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, volume 1, pages 2–1, 2007.

[27] 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, 2008.

[28] H. Shan and A. Banerjee. Generalized probabilistic matrix factorizations for collaborative filtering. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1025–1030. IEEE, 2010.

[29] V. K. Singh, M. Mukherjee, and G. K. Mehta. Combining collaborative filtering and sentiment classification for improved movie recommendations. In *Multi-disciplinary Trends in Artificial Intelligence*, pages 38–50. Springer, 2011.

[30] P. J. Stone, D. C. Dunphy, and M. S. Smith. The general inquirer: A computer approach to content analysis. 1966.

[31] M. Tkalcic, M. Kunaver, J. Tasic, and A. Košir. Personality based user similarity measure for a collaborative recommender system. In *Proceedings of the 5th Workshop on Emotion in Human-Computer Interaction-Real world challenges*, pages 30–37, 2009.

[32] M. T. Tomlinson, D. Hinote, and D. B. Bracewell. Predicting conscientiousness through semantic analysis of facebook posts. *Proceedings of WCPR*, 2013.

[33] E. C. Tupes and R. E. Christal. Recurrent personality factors based on trait ratings. Technical report, DTIC Document, 1961.

[34] B. Verhoeven, W. Daelemans, and T. De Smedt. Ensemble methods for personality recognition. In *Proc of Workshop on Computational Personality Recognition, AAAI Press, Melon Park, CA*, pages 35–38, 2013.

[35] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[36] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, volume 12, pages 403–414. SIAM, 2012.