Recommendation vs Sentiment Analysis: A Text-Driven Latent Factor Model for Rating Prediction with Cold-Start Awareness

Kaisong Song[1], Wei Gao[2*], Shi Feng[1], Daling Wang[1], Kam-Fai Wong[3†], Chengqi Zhang[4†]
[1]Northeastern University, Shenyang, China
[2]Victoria University of Wellington, New Zealand
[3]The Chinese University of Hong Kong, Hong Kong
[4]University of Technology Sydney, Australia
kaisongsong@gmail.com wei.gao@vuw.ac.nz {fengshi,wangdaling}@cse.neu.edu.cn

Abstract

Review rating prediction is an important research topic. The problem was approached from either the perspective of recommender systems (RS) or that of sentiment analysis (SA). Recent SA research using deep neural networks (DNNs) has realized the importance of user and product interaction for better interpreting the sentiment of reviews. However, the complexity of DNN models in terms of the scale of parameters is very high, and the performance is not always satisfying especially when user-product interaction is sparse. In this paper, we propose a simple, extensible RS-based model, called Text-driven Latent Factor Model (TLFM), to capture the semantics of reviews, user preferences and product characteristics by jointly optimizing two components, a user-specific LFM and a product-specific LFM, each of which decomposes text into a specific low-dimension representation. Furthermore, we address the cold-start issue by developing a novel Pairwise Rating Comparison strategy (PRC), which utilizes the difference between ratings on common user/product as supplementary information to calibrate parameter estimation. Experiments conducted on IMDB and Yelp datasets validate the advantage of our approach over state-of-the-art baseline methods.

1 Introduction

Review rating prediction is a fundamental problem in the field of sentiment analysis and opinion mining. Detecting users' sentiment polarity or intensity (e.g., 1-5 stars in Yelp and 1-10 stars in IMDB) about all kinds of products from vast amount of reviews has recently drawn close attention from research communities due to its importance to wide range of applications, such as product recommendation, product quality tracking and public opinion mining.

The problem has been approached from two points of views in the literature, i.e., Recommender Systems (RS) and Sentiment Analysis (SA). The RS-based approach typically adopts matrix factorization techniques, which is also known as collaborative filtering, by modeling the inner product of user and product factors [Salakhutdinov and Mnih, 2007]. In this approach, most of its variants focus on capturing user-product interactions while ignoring the valuable content of reviews [Koren, 2008] or just use it as auxiliary information to enhance the representations of users and products [Zhang *et al.*, 2014b]. The SA-based approach has largely regarded the problem as a multi-class classification task focusing on text content following Pang et al. [2005]. Under this direction, most studies rely on handcrafted features and lexicons for effective learning performance, which however is biased and labor intensive. More recently, people using Deep Neural Networks (DNNs) to learn discriminative representations from text has realized the importance of incorporating user/product information [Tang *et al.*, 2015a; 2015b]. However, DNNs typically have much more parameters to estimate, which lead to high model complexity, exhaustive parameter tuning and heavy reliance on word embeddings. Besides, DNNs easily over-fit limited training data and cannot be generalized well when the user-product interactions are sparse in common case.

Compared with DNNs, RS-based methods such as Latent Factor Models (LFM) are simple, easy to train and has achieved promising results on review rating prediction [Jin *et al.*, 2016; Zhang *et al.*, 2014b]. However, built from the user-item rating matrix, they encounter challenges when review content has to be considered. For example, different users providing similar reviews on a movie might rate it differently, or they might give it the same rating while writing very different reviews, depending on how strict/lenient they are, or how they like to convey their opinions. This presents the influence of user-specific preference on rating. Similarly, different products given similar review opinions might receive different ratings or receive the same rating but with different reviews, depending on the specific properties of the products. Therefore, it is difficult to know what factors beyond the global user-product interactions really matter to the rating.

Another challenge is the existence of "cold-start" users and products commonly encountered in social media environment, which can decrease the prediction performance dramatically due to the lack of enough reviews and ratings for training [Zhang *et al.*, 2014a; Xu *et al.*, 2015]. Existing works [Li

*Work done when this author was affiliated with Qatar Computing Research Institute

†kfwong@se.cuhk.edu.hk {MoE Key Laboratory on High Confidence Software Technologies, China}, chengqi.zhang@uts.edu.au

et al., 2011; Tang *et al.*, 2015a; 2015b] cannot learn the representations well for inactive users and unpopular products, and fail to deal with newly entered users or products that do not contribute any review.

In this paper, we propose a simple, extensible Text-driven Latent Factor Model (TLFM) for review rating prediction. We model user- and product-specific LFM components separately and then jointly optimize the two components in a unified framework. Thereafter, TLFM will be further optimized by a novel Pairwise Rating Comparison (PRC) strategy to alleviate the cold-start problem in product reviews, which was not considered in SA-based methods. The intuitive idea is to enrich the rating information by using the difference between the expected result and ground truth in a rating comparison as supplementary information to calibrate parameter estimation. The main contributions of our paper are three-fold:

- We present a simple, extensible review rating prediction model to capture semantics of review text, user preferences and product characteristics based on a variant of latent factor model.
- We propose a novel cold-start optimization strategy to calibrate the parameter estimation for cold-start users and products which suffer from insufficient ratings. In addition, two mechanisms, AvgUI and UnkUI, are presented for the unseen users and products.
- Our comparative results on three public datasets show that TLFM outperforms state-of-the-art methods especially SA-based approach for review rating prediction.

2 Related Work

Sentiment analysis on subjective documents such as tweets and product reviews has been widely studied [Pang and Lee, 2005; Feng *et al.*, 2011; Jiang *et al.*, 2011]. Most of these work ignored the crucial characteristics of users and products which make significant influences on sentiments. Tang et al. [2015b] presented a neural network model that took user information into account, and later they further considered product information and developed a User Product Neural Network model (UPNN) which achieved state-of-the-art performance on rating prediction. However, the number of parameters for each user/product preference matrix in their method was large, which made representation learning inefficient and hard to tune. Besides, learning was ineffective when the user-product interactions were sparse.

LFM, a kind of collaborative filtering method, has drawn great attention in recent years for its promising recommendation performance in Netflix Prize competition [Koren, 2008; Bell and Koren, 2007]. Compared to Neural Networks, LFM is easy to extend and learn, which also has many variants, such as comments rank [Agarwal *et al.*, 2011] and tweet recommendation [Song *et al.*, 2014]. Recent advances are shifting towards utilizing collaborative filtering methods to make sentiment prediction. Song et al. [2015] and Wu et al. [2016] proposed personalized sentiment classification models by considering both microblog users and their social relations. Li et al. [2011] developed a user-product-word tensor factorization model for review rating prediction. However, these studies did not consider cold-start problem, which is very common on review sites and usually influences system performance. Although some LFM methods have considered text, they mostly use the content of reviews as auxiliary information, such as enhancing the interpretation of LFM by extracting user and product features from reviews [Zhang *et al.*, 2014b] or using DNN-based text representation to guide factors estimation [Kim *et al.*, 2016].

Cold start is a challenging issue. Zhang et al. [2014a] proposed a context-aware semi-supervised co-training algorithm for tackling cold start in product recommendation. However, context (e.g., user occupation and movie genre) is not always available and the resulting models are hard to generalize. Xu et al. [2015] utilized active users/popular products with high occurrence frequency to enhance the representation of cold-start users/products. But it is tricky to assign an appropriate threshold to identify cold-start users/products. In this work, we propose a pairwise rating comparison optimization strategy without need to identify cold-start users/products, based on our intuition that the ratings of all users/products are useful to provide clues in comparison. Meanwhile, the factors for all the users and products can be calibrated, which can further improve the performance.

3 Preliminaries

In this section, we first introduce some frequently used notations, and then provide an overview of the Latent Factor Model, which paves way for proposing our TLFM method.

3.1 Notations

For a typical online review website such as Yelp or IMDB, we would have a set of users $\mathcal{U}$ writing reviews on a set of products $\mathcal{I}$. We use y_{ui} to denote a rating user $u \in \mathcal{U}$ gives on a product $i \in \mathcal{I}$, which is typically associated with a textual review r_{ui} on i written by u. The ratings can be integers ranging from 1 (star) indicating no interest to 5 or 10 (star) indicating a strong interest. Given the training set $\mathcal{T} = \{y_{ui}|r_{ui}, u, i\}$ where y_{ui} is known, we want to learn a function f_Θ which predicts the most likely rating $\widehat{y}_{ui}$ for each review in the test set $\mathcal{T}' = \{y_{ui}|r_{ui}, u, i\}$ where y_{ui} is unknown. We can formulate our review rating prediction task as below:

$$f_\Theta(r_{ui}, u, i) \longrightarrow \widehat{y}_{ui} \tag{1}$$

where function f_Θ indicates a predictor with parameters Θ that can be learned by minimizing the differences $|y_{ui} - \widehat{y}_{ui}|$ of the estimated rating $\widehat{y}_{ui}$ and the real rating y_{ui} over $\mathcal{T}$.

3.2 Latent Factor Model (LFM)

Latent factor model (LFM) is a kind of model-based collaborative filtering method, which is designed for improving the prediction accuracy of the Netflix movie recommendation systems [Bell and Koren, 2007; Koren, 2008]. LFM is trained only based on observed ratings, which can be defined as:

$$\widehat{y}_{ui} = \mu + b_u + b_i + q_i^{\mathrm{T}} p_u \tag{2}$$

where the observed rating is factorized into four components: global average rating μ, user bias b_u, product bias b_i and user-product interaction $q_i^{\mathrm{T}} p_u$ that captures user u's personalized preference on product i. The user-factors vector $p_u \in \mathbb{R}^K$ and product-factors vector $q_i \in \mathbb{R}^K$ usually have a low-dimensional representation in the same factor space and

$K \ll \min\{|\mathcal{U}|, |\mathcal{I}|\}$. The LFM only considers user and product information without considering text content which actually dominates sentiment expression in subjective text.

4 Text-driven LFM for Review Rating

Existing RS-based approaches using LFM focus on modeling user-product interactions, which is not text-driven even though text content might be considered as an auxiliary resource in some of the variants [Zhang *et al.*, 2014b; Kim *et al.*, 2016]. In this section, we propose text-driven LFMs which learn the semantic factors of words directly from the review text. We first present a baseline model, a user-specific model (ULFM) and a product-specific model (PLFM), and then compose them into a fully-configured text-driven model.

4.1 Baseline Model

The baseline model computes rate scoring from general perspective without considering the specific influences from different users and products on review texts. We first present a baseline model which embodies general properties of words and takes the first-order form as follow:

$$\widehat{y}_{ui} = \mu + \sum_{w \in \mathcal{W}(r_{ui})} \alpha_w b_w \tag{3}$$

where μ is global average rating, $\mathcal{W}(r_{ui})$ is the set of words in review r_{ui}, b_w is the word bias and $\alpha_w = \frac{1}{|\mathcal{W}(r_{ui})|}$ is the normalization term. Therefore, sentiment intensity can be calculated by accumulating the effects of word biases.

4.2 User Latent Factor Model (ULFM)

Review text often reflects user's specific individuality due to their frequently embedded language habit, personal character, opinion bias and so on. Therefore, we propose a User-based Latent Factor Model (ULFM) which can be represented as a linear combination of a baseline component and a user-specific component as follow:

$$\widehat{y}_{ui} = \underbrace{\mu + \sum_{w \in \mathcal{W}(r_{ui})} \alpha_w b_w}_{\text{baseline component}} + \underbrace{b_u + \Big(\sum_{w \in \mathcal{W}(r_{ui})} \alpha_w v_w^{\mathrm{T}} \Big) p_u}_{\text{user-specific component}} \tag{4}$$

where the baseline component captures common sentiment knowledge and the user-specific component explicitly captures user influences. We mainly consider user-sentiment consistency and user-text consistency which were first discussed in DNN-based sentiment model [Tang *et al.*, 2015a].

User-sentiment consistency. A user has rating preferences which are independent of the rated products. For example, a critical user tends to give lower ratings, but a lenient user favors giving higher ratings, which is captured by user bias b_u. If global average rating $\mu = 3$, the user with $b_u = +1$ is more lenient than the user with $b_u = -1$, since $3+1 > 3-1$.

User-text consistency. A user has sentiment specific word preferences. For example, a critical user will use the word "*good*" to express a much more satisfying attitude than a lenient user, which can be captured by the user-word interaction. Specifically, we decompose review r_{ui} into word semantic level by representing each word $w \in \mathcal{W}(r_{ui})$ as a K-dimensional factor vector v_w. Then, user-word interaction $v_w^{\mathrm{T}} p_u$ can capture user u's preference on word w. Finally, user-text consistency can be captured by averaging all user-word interactions $v_w^{\mathrm{T}} p_u$ ($w \in \mathcal{W}(r_{ui})$).

4.3 Product Latent Factor Model (PLFM)

Based on the model symmetry, we can easily derive a Product Latent Factor Model (PLFM) by explicitly considering product-specific information and common sentiment knowledge. This PLFM focuses on capturing sentiment expression on specific target (i.e., event or product), which is actually the task of target-dependent sentiment analysis [Jiang *et al.*, 2011; Vo and Zhang, 2015]. Our PLFM can be formulated as a linear combination of a baseline component and a product-specific component as below:

$$\widehat{y}_{ui} = \underbrace{\mu + \sum_{w \in \mathcal{W}(r_{ui})} \alpha_w b_w}_{\text{baseline component}} + \underbrace{b_i + \Big(\sum_{w \in \mathcal{W}(r_{ui})} \alpha_w v_w^{\mathrm{T}} \Big) q_i}_{\text{product-specific component}} \tag{5}$$

where the product-specific component explicitly captures product-sentiment consistency and product-text consistency:

Product-sentiment consistency. A product has rating preferences determined by its characteristics. That is, a high-quality product tends to receive high ratings, but a low-quality one might receive low ratings, which is captured by bias b_i. For example, if global average rating $\mu = 3$, the product with bias $b_i = +2$ has a better quality than the product with bias $b_i = -2$, since $3 + 2 > 3 - 2$.

Product-text consistency. A product has specific preferences on word selection. For example, people use "*comedy*" and "*family*" to evaluate the movie of "*Mr. Bean's Holiday*", but use "*action*", "*fantasy*" and "*adventure*" for "*The Hobbit*". This product-text consistency can be captured by averaging all product-word interactions $v_w^{\mathrm{T}} q_i$ ($w \in \mathcal{W}(r_{ui})$).

Tang et al. [2015a] examined three public benchmark datasets (i.e., IMDB, Yelp 2014 and Yelp 2013) and verified the general existence of these consistencies.

4.4 The Unified Text-driven LFM (TLFM)

Here, we combine ULFM and PLFM smoothly under a unified RS-based framework, which can be formulated as below:

$$\widehat{y}_{ui} = g\Big(base + \Big(\sum_{w \in \mathcal{W}(r_{ui})} \alpha_w v_w^{\mathrm{T}} \Big) \Big(p_u + q_i \Big) \Big) \tag{6}$$

where $base = \mu + b_u + b_i + \sum_{w \in \mathcal{W}(r_{ui})} \alpha_w b_w$ and function $g(x) = 1 + \frac{\mathcal{C}-1}{1+e^{-x}}$ makes outputs within the range of valid rating values $[1, \mathcal{C}]$ by reformulating the sigmoid function, say, if $x \to +\infty$ (or $x \to -\infty$), $g(x) = \mathcal{C}$ (or 1).

Finally, we obtain our objective function that is to minimize the sum of squared errors with a regularization term:

$$\begin{aligned} \mathcal{L} = \sum_{y_{ui} \in \mathcal{T}} \big(y_{ui} - \widehat{y}_{ui} \big)^2 + \lambda \big(b_u^2 + b_i^2 + \\ \sum_{w \in \mathcal{W}(r_{ui})} b_w^2 + \|p_u\|^2 + \|q_i\|^2 + \sum_{w \in \mathcal{W}(r_{ui})} \|v_w\|^2 \big) \end{aligned} \tag{7}$$

where the first term strives to fit the given ratings, and the last term avoids over-fitting by penalizing the magnitudes of the parameters $\Theta = \big\{ \{p_u\}, \{q_i\}, \{v_w\}, \{b_u\}, \{b_i\}, \{b_w\} \big\}$ in the model, weight λ controls the strength of regularization. With the estimated $\Theta = \arg\min_\Theta \mathcal{L}$, we can predict the rating for each testing instance in the test set $\mathcal{T}'$ using formula 6.

5 Optimization for Cold Start

In this section, we first clarify cold-start problem in review rating task, then present a Pairwise Rating Comparison strategy (PRC) for alleviating the problem, and finally optimize TLFM with the PRC strategy.

5.1 Cold-Start Problem Preliminaries

To improve the prediction quality for the unseen (or inactive) users and unseen (or rarely rated) products is challenging. The lack of ratings from these users/products may weaken the parameter estimation with respect to the latent factors. Technically, this is referred to cold-start problem [Zhang *et al.*, 2014a; Xu *et al.*, 2015].

For dealing with unseen users or unseen products, inspired by [Tang *et al.*, 2015a] we adopt two solutions called AvgUI and UnkUI. The AvgUI averages over all the observed p_u and b_u or q_i and b_i as the representations of new users/products. The UnkUI method learns a shared "unknown" representation for new users/products by randomly drawing ξ reviews as their alternative training instances.

For inactive users and rarely rated products, exiting works [Zhang *et al.*, 2014a; Lin *et al.*, 2013] resort to context information such as attributes (e.g., movie genre or user occupation), implicit feedbacks (e.g., clicks), social relation, etc. Such kind of context is not always available and the resulting models are hard to generalize. With the data at hand, we attempt to improve parameter estimation via a Pairwise Rating Comparison strategy (PRC) which is detailed below.

5.2 Pairwise Rating Comparison Strategy (PRC)

The basic idea of our strategy is to minimize the difference between actual difference and expected difference via rating comparisons to calibrate the factors of users/products. Since there exists no definite boundary between cold-start users/products and active users/popular products, we try to calibrate factors for all the users and products because cold-start users/products can also contribute ratings. Specifically, let r_{ui} and r_{vi} be any two reviews given by different users (i.e., u and v) on the same product i, we can build a pair of rating comparisons: actual difference $y_{ui} - y_{vi}$ and expected difference $\widehat{y}_{ui} - y_{vi}$. The intuitive idea is to make expected difference approximate actual difference as close as possible over the whole training set $\mathcal{T}$. We ignore regularization term temporarily to sharpen the focus and obtain the objective function which balances generalization and specificity by combining the two terms in a linear way as below:

$$\begin{aligned}\mathcal{O} &= \frac{1}{\mathcal{N}_{\mathcal{I}}}\sum_{y_{ui}}\sum_{y_{vi}}\Big(\underbrace{(y_{ui}-y_{vi})}_{\textbf{actual}}-\underbrace{(\widehat{y}_{ui}-y_{vi})}_{\textbf{expected}}\Big)^2+\frac{\mathcal{L}}{\mathcal{N}_{\mathcal{I}}}\\ &= \frac{1}{\mathcal{N}_{\mathcal{I}}}\sum_{y_{ui}}\sum_{y_{vi}}\big(y_{ui}-\widehat{y}_{ui}\big)^2+\frac{1}{\mathcal{N}_{\mathcal{I}}}\sum_{y_{ui}}\big(y_{ui}-\widehat{y}_{ui}\big)^2 \qquad (8)\\ &= \sum_{y_{ui}}\frac{|\mathcal{T}_i|}{\mathcal{N}_{\mathcal{I}}}\big(y_{ui}-\widehat{y}_{ui}\big)^2\end{aligned}$$

where $|\mathcal{T}_i|$ is the number of reviews received by the product i and $\mathcal{N}_{\mathcal{I}}$ is a normalization term. As we can see, this objective function can be viewed as a weighted version of formula 7.

Similarly, we can also define actual difference $y_{ui} - y_{uj}$ and expected difference $\widehat{y}_{ui} - y_{uj}$ with the same user u and different products, and obtain another objective function:

$$\mathcal{Q} = \sum_{y_{ui}}\frac{|\mathcal{T}_u|}{\mathcal{N}_{\mathcal{U}}}\big(y_{ui}-\widehat{y}_{ui}\big)^2 \qquad (9)$$

where $|\mathcal{T}_u|$ is the number of reviews written by the user u and $\mathcal{N}_{\mathcal{U}}$ is a normalization term. Intuitively, $|\mathcal{T}_u|$ and $|\mathcal{T}_i|$ reflect the popularity of user u and product i, respectively [Li *et al.*, 2011]. In this work, we denote $\frac{|\mathcal{T}_u|}{\mathcal{N}_{\mathcal{U}}}$ and $\frac{|\mathcal{T}_i|}{\mathcal{N}_{\mathcal{I}}}$ as the relative importance of training instance from user and product perspectives, respectively, and define the constant $\mathcal{N}_{\mathcal{U}} = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{U}}|\mathcal{T}_u|$ and constant $\mathcal{N}_{\mathcal{I}} = \frac{1}{|\mathcal{I}|}\sum_{i\in\mathcal{I}}|\mathcal{T}_i|$.

5.3 PRC Optimization for TLFM

Based on a smooth linear combination of formulas 8 and 9, we rewrite formula 7 to obtain the final objective function:

$$\begin{aligned}\mathcal{G} = \sum_{y_{ui}\in\mathcal{T}}\big(\sigma(\tfrac{|\mathcal{T}_u|}{\mathcal{N}_{\mathcal{U}}})+\sigma(\tfrac{|\mathcal{T}_i|}{\mathcal{N}_{\mathcal{I}}})\big)\big(y_{ui}-\widehat{y}_{ui}\big)^2+\lambda\big(b_u^2+\\ b_i^2+\sum_{w\in\mathcal{W}(r_{ui})}b_w^2+\|p_u\|^2+\|q_i\|^2+\sum_{w\in\mathcal{W}(r_{ui})}\|v_w\|^2\big)\end{aligned} \qquad (10)$$

where $\mathcal{C}_{ui} = \sigma(\frac{|\mathcal{T}_u|}{\mathcal{N}_{\mathcal{U}}}) + \sigma(\frac{|\mathcal{T}_i|}{\mathcal{N}_{\mathcal{I}}})$ can be considered as the weight of the observed rating y_{ui} and $\sigma(x) = \frac{1}{1+e^{-x}}$ is sigmoid function for normalizing relative importance into $(0.5, 1)$, so we can easily derive $\mathcal{C}_{ui} \in (1, 2)$. Parameters Θ can be learned by stochastic gradient descent [Bottou, 2003], which searches for minimum by updating the related parameters in the negative direction of the gradient as below:

$$\begin{aligned}\frac{\partial\mathcal{G}}{\partial b_w} &= \lambda b_w - \hat{e}_{ui}\alpha_w\hat{g}\ ,\ \frac{\partial\mathcal{G}}{\partial v_w} = \lambda v_w - \hat{e}_{ui}\alpha_w(p_u+q_i)\hat{g}\\ \frac{\partial\mathcal{G}}{\partial b_u} &= \lambda b_u - \hat{e}_{ui}\hat{g}\quad ,\quad \frac{\partial\mathcal{G}}{\partial p_u} = \lambda p_u - \hat{e}_{ui}(\sum_w\alpha_w v_w)\hat{g}\\ \frac{\partial\mathcal{G}}{\partial b_i} &= \lambda b_i - \hat{e}_{ui}\hat{g}\quad ,\quad \frac{\partial\mathcal{G}}{\partial q_i} = \lambda q_i - \hat{e}_{ui}(\sum_w\alpha_w v_w)\hat{g}\end{aligned} \qquad (11)$$

where $\hat{e}_{ui} = \mathcal{C}_{ui}(y_{ui} - \hat{y}_{ui})$ is the weighted difference between real rating and prediction, and $\hat{g} = \frac{(\widehat{y}_{ui}-1)(\mathcal{C}-\widehat{y}_{ui})}{\mathcal{C}-1}$ is derived from $g(x)'$. Intuitively, the inputs with popular users and products will have higher weights (i.e., higher $\mathcal{C}_{ui}$) since popular users and products usually have better estimation on their factors. Our model will learn more confidently from the important observations with higher weights, which can lead to better performance and alleviate the cold-start issue.

6 Experiments and Results

6.1 Settings

We experiment on three public datasets[1]: IMDB, Yelp 2013 and Yelp 2014. The datasets are tokenized by Stanford CoreNLP [Manning *et al.*, 2014] and split into training, development and testing sets with exactly the same 80/10/10 split as [Tang *et al.*, 2015a]. The statistics is presented in Table 1.

We evaluate the performance of predictions by Root Mean Squared Error: $RMSE = \sqrt{\sum_{y_{ui}\in\mathcal{T}'}(y_{ui}-\widehat{y}_{ui})^2/|\mathcal{T}'|}$.

[1] http://ir.hit.edu.cn/~dytang/paper/acl2015/dataset.7z

Dataset	scale	#users	#items	#reviews	Length (Avg)
IMDB	1∼10	1,310	1,635	84,919	394.6 (word)
Yelp14	1∼5	4,818	4,194	231,163	196.9 (word)
Yelp13	1∼5	1,631	1,633	78,966	189.3 (word)

Table 1: Statistics of experimental datasets we used.

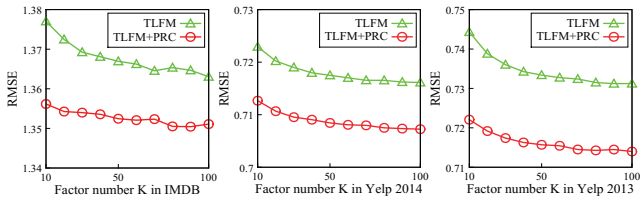


Figure 1: Settings of factor number K on our datasets.

We optimize the factor number K via validation on the development set by performing a grid search on all values of $10 * x$ with $x \in \{1, ..., 10\}$ and display results in Figure 1. We see that the performance becomes stable when $x = 10$, so we set $K = 100$. We set $\lambda = 1e - 4$, and the number of iterations as 50 according to our observation on the best RMSE scores on the development set. All the parameters are initialized randomly from a uniform distribution on the interval $(0, 0.1)$. Our experiments are conducted on a commodity Windows 7 PC with Core i7-6700 CPU and 16GB RAM.

6.2 Comparison of different methods

We compare TLFM with some traditional and advanced baselines as shown in Table 2, and separate results into three groups: (1) text-independent methods; (2) text-based methods; (3) the methods using text, user and product information.

(1) **Majority** considers the rating value which is the majority in training set as the predicted rating of each review in test set; **LFM** is the matrix factorization method trained on user-product rating matrix (see formula 2).

(2) **Ngram** is a support vector machine (SVM) classifier trained on unigram, bigram and trigram features [Fan *et al.*, 2008]; **TextFeature** is the feature engineering method using extracted text features including word n-grams, sentiment lexicon features, and negation features for training a SVM classifier [Kalchbrenner *et al.*, 2014]; **AvgWordvec** is obtained by training a SVM classifier with input features that are averaged word embeddings in a review [Mikolov *et al.*, 2013]; **SSWE** averages Sentiment-Specific Word Embeddings as review text representation, and then trains a SVM classifier; **Paragraph Vector** is a SVM classifier trained on paragraph representations of documents [Le and Mikolov, 2014]; **RNTN+Recurrent** learns sentence representations with Recursive Neural Tensor Network [Socher *et al.*, 2013] which are then fed into a recurrent neural network, where the hidden vectors are averaged for classification.

(3) **JMARS** is a probabilistic model based on collaborative filtering and topic modeling, which considers user and aspects of a review [Diao *et al.*, 2014]; **TFM** [Li *et al.*, 2011] is a linear model that combines the entries estimated from a user-product-word tensor factorization model; **UPNN** is a neural network model which modifies word embeddings in the input layer with user/product preference matrix, and then concatenates user/product vector with generated review representation via softmax layer [Tang *et al.*, 2015a]; **TLFM** is our proposed text-driven LFM, and **TLFM+PRC** additionally considers cold start with PRC.

Method	IMDB	Yelp 2014	Yelp 2013
Majority°	2.495	1.097	1.060
LFM	**1.953**	**0.998**	**0.981**
Trigram°	1.783	0.804	0.814
TextFeature°	1.793	**0.800**	0.845
AvgWordvec+SVM°	1.985	0.893	0.898
SSWE+SVM°	1.973	0.851	0.849
Paragraph Vector°	1.814	0.802	0.832
RNTN+Recurrent°	**1.764**	0.821	**0.804**
JMARS°	1.773	0.999	0.985
TFM	1.598	0.835	0.836
UPNN°	1.602	0.764	0.784
TLFM	1.358	0.720	0.735
TLFM+PRC	**1.352**	**0.712**	**0.716**

Table 2: Comparison among different methods. The results with superscript ∘ are reported in [Tang *et al.*, 2015a]. The best results (lower is better) in each group are highlighted.

In Group 1, **LFM** outperforms **Majority** by using user and product interactions, but it still cannot compete with text-based methods. In Group 2, SVM classifiers with **Ngrams** or handcrafted text features outperform **AvgWordvec**, **SSWE** and **Paragraph Vector** in most cases that are trained on word embeddings. **RNTN+Reccurent** performs the best by modeling document with semantic composition. In Group 3, **UPNN** and **TLFM** outperform **JMARS** and **TFM** by explicitly modeling user preferences and product characteristics, which implies considering the four consistencies is effective. Our **TLFM** is better than **UPNN** since the latter needs to learn much more parameters regarding preference representations in different network layers, which makes it difficult for parameter tuning and achieving comparable performance on our sparse rating matrices. Also, **UPNN** heavily relies on word embeddings which may be not well trained on the limited datasets. This is highly encouraging, indicating the simplicity and effectiveness of our method. Besides, **PRC** strategy achieves further improvements, which suggests that cold-start problem can be alleviated.

6.3 Effect of the four consistencies

We study the effects of consistency consideration on rating prediction, including user-sentiment consistency (**us**), user-text consistency (**ut**), product-sentiment consistency (**ps**) and product-text consistency (**pt**). We compare different configurations of UPNN and our TLFM+PRC in Table 3.

It is clear that the partial configurations cannot compete with full models indicating all the consistencies need to be considered. In our method, we find that **ut+pt**>**us+ps**, implying that **ut** and **pt** are more important than **us** and **ps**, because text provides better sentiment clues. In addition, **us+ut**>**ps+pt** indicates that user preferences are more impactful than product traits. TLFM outperforms UPNN under

Dataset	Method	us+ut	ut+pt	ps+pt	ps+us	full
IMDB	UPNN	1.712	1.622	1.743	1.607	1.602
	Ours	1.409	1.408	1.521	1.675	1.352
Yelp14	UPNN	0.776	0.808	0.778	0.789	0.764
	Ours	0.740	0.722	0.745	0.788	0.712
Yelp13	UPNN	0.802	0.823	0.828	0.802	0.784
	Ours	0.762	0.732	0.781	0.802	0.716

Table 3: Comparison of different consistency configurations.

Method	IMDB	Yelp 2014	Yelp 2013
AvgUI	1.394	0.719	0.723
UnkUI	**1.385**	**0.718**	**0.720**

Table 4: Comparison between AvgUI and UnkUI methods

most configurations, which verifies its effectiveness on capturing the effects of users and products.

6.4 Effect of unseen users and products

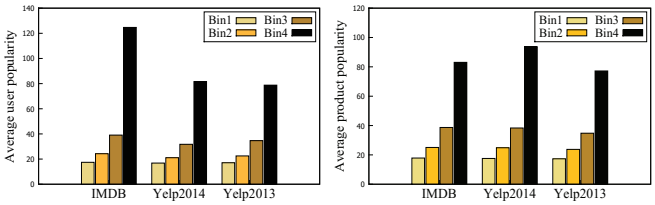
We also present the prediction results of our methods (i.e., **UnkUI** and **AvgUI**) for the unseen users or products. We first randomly select 10% users and products from test set, replace their names with unseen names so as to obtain a cold-start test set. We set ξ as 200 for **UnkUI**. As shown in Table 4, both **AvgUI** and **UnkUI** perform much better than UPNN (see Table 2), which verifies the effectiveness of our methods.

6.5 Effect of user and product popularity

We further study the influence of user/product popularity. User or product popularity is determined by the number of times a user rates or a product is rated. We partition users/products into four equal-sized bins according to their popularity in training set as shown in Figure 2 where the popularity from low to high is given from bin 1 to 4. We then group the test instances corresponding to these bins, and obtain the performance in Table 5. We find that the bins with more popular (active) users/products usually have better results since user/product factors are estimated more accurately. **TLFM+PRC** improves **TLFM** substantially over all the bins, which indicates the effectiveness of our **PRC** strategy on alleviating cold-start problem.

6.6 Complexity analysis and running time

Let $|\mathcal{T}|$ be the size of training set, L be the average number of words in each training instance, K be the factor number and N be the number of iterations. The time complexity of TLFM+PRC mainly lies in updating parameters Θ. Its overall time complexity is $\mathcal{O}\big(N|\mathcal{T}|(K+1)(L+2)\big)$. UPNN is based on a Convolutional Neural Network with three filters (see [Tang *et al.*, 2015a]). Its complexity mainly lies in producing the convolutional features. Let d_U be the output length of the multiplicative composition between user preference matrix and word embedding, and d_P be the output length of the multiplicative composition between product preference matrix and word embedding, X be the dimension of word embeddings and S be the output length of each filter. In this work, we have $X = 2K$, so the overall time complexity of



(a) Grouped users (b) Grouped products

Figure 2: Statistics of grouped users and products by popularity (Bins 1-4 are arranged from low to high popularity).

Dataset	Method	Bin_u1	Bin_u2	Bin_u3	Bin_u4
IMDB	TLFM	1.499	1.522	1.386	1.283
	+PRC	1.497	1.503	1.385	1.277
Yelp14	TLFM	0.767	0.751	0.727	0.695
	+PRC	0.759	0.739	0.721	0.686
Yelp13	TLFM	0.767	0.756	0.754	0.711
	+PRC	0.748	0.734	0.732	0.694
-	-	Bin_i1	Bin_i2	Bin_i3	Bin_i4
IMDB	TLFM	1.368	1.401	1.332	1.355
	+PRC	1.356	1.400	1.323	1.350
Yelp14	TLFM	0.759	0.750	0.714	0.706
	+PRC	0.755	0.741	0.706	0.696
Yelp13	TLFM	0.772	0.765	0.735	0.717
	+PRC	0.758	0.750	0.715	0.695

Table 5: Cold-start performance of users and products.

UPNN is $\mathcal{O}\big(N|\mathcal{T}|(X+6S)(d_U+d_P)L\big)$. Obviously, UPNN has much higher time complexity, which results from much more parameters regrading preference representations.

Meanwhile, we also compare the running time between TLFM+PRC and UPNN on the training sets. We run the code of UPNN with suggested settings in [Tang *et al.*, 2015a] on the same experimental environment as TLFM+PRC. We then have the results: UPNN (IMDB: 441.84 min, Yelp14: 450.64 min, Yelp13: 158.65 min) and TLFM+PRC (IMDB: 7.21 min, Yelp14: 9.67 min, Yelp13: 3.12 min). We can find that UPNN is much more time-consuming than TLFM+PRC for training, which indicates the high efficiency of our method.

7 Conclusion

We proposed a simple, extensible text-driven latent factor model for review rating prediction, which is focused on capturing interactions between review content and user preferences/product characteristics. We also addressed two cases of "cold-start" by developing AvgUI and UnkUI for the unseen users/products and presenting a PRC strategy to calibrate the factors for the inactive users and rarely rated products. The experimental results on IMDB and Yelp datasets show that our method outperforms state-of-the-art baselines.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61370074, 61402091) and partially supported by the UGC, HK, under the GRF initiative (#14232816) and ITF, HK, (#ITP00416PL).

References

[Agarwal *et al.*, 2011] Deepak Agarwal, Bee-Chung Chen, and Bo Pang. Personalized recommendation of user comments via factor models. In *EMNLP*, pages 571–582, 2011.

[Bell and Koren, 2007] Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explorations*, 9(2):75–79, 2007.

[Bottou, 2003] Léon Bottou. Stochastic learning. In *Adv. Lectures Mach. Learn.*, pages 146–168, 2003.

[Diao *et al.*, 2014] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *SIGKDD*, pages 193–202, 2014.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[Feng *et al.*, 2011] Shi Feng, Daling Wang, Ge Yu, Wei Gao, and Kam-Fai Wong. Extracting common emotions from blogs based on fine-grained sentiment clustering. *Knowl. Inf. Syst.*, 27(2):281–302, 2011.

[Jiang *et al.*, 2011] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *ACL*, pages 151–160, 2011.

[Jin *et al.*, 2016] Zhipeng Jin, Qiudan Li, Daniel Dajun Zeng, YongCheng Zhan, Ruoran Liu, Lei Wang, and Hongyuan Ma. Jointly modeling review content and aspect ratings for review rating prediction. In *SIGIR*, pages 893–896, 2016.

[Kalchbrenner *et al.*, 2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665, 2014.

[Kim *et al.*, 2016] Dong Hyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*, pages 233–240, 2016.

[Koren, 2008] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *ICDM*, pages 426–434, 2008.

[Le and Mikolov, 2014] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[Li *et al.*, 2011] Fangtao Li, Nathan Nan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, and Xiaoyan Zhu. Incorporating reviewer and product information for review rating prediction. In *IJCAI*, pages 1820–1825, 2011.

[Lin *et al.*, 2013] Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *SIGIR*, pages 283–292, 2013.

[Manning *et al.*, 2014] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL*, pages 55–60, 2014.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124, 2005.

[Salakhutdinov and Mnih, 2007] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[Socher *et al.*, 2013] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng, and Chris Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642, 2013.

[Song *et al.*, 2014] Kaisong Song, Daling Wang, Shi Feng, Yifei Zhang, Wen Qu, and Ge Yu. CTROF: A collaborative tweet ranking framework for online personalized recommendation. In *PAKDD*, pages 1–12, 2014.

[Song *et al.*, 2015] Kaisong Song, Shi Feng, Wei Gao, Daling Wang, Ge Yu, and Kam-Fai Wong. Personalized sentiment classification based on latent individuality of microblog users. In *IJCAI*, pages 2277–2283, 2015.

[Tang *et al.*, 2015a] Duyu Tang, Bing Qin, and Ting Liu. Learning semantic representations of users and products for document level sentiment classification. In *ACL*, pages 1014–1023, 2015.

[Tang *et al.*, 2015b] Duyu Tang, Bing Qin, Ting Liu, and Yuekui Yang. User modeling with neural network for review rating prediction. In *IJCAI*, pages 1340–1346, 2015.

[Vo and Zhang, 2015] Duy-Tin Vo and Yue Zhang. Target-dependent twitter sentiment classification with rich automatic features. In *IJCAI*, pages 1347–1353, 2015.

[Wu and Huang, 2016] Fangzhao Wu and Yongfeng Huang. Personalized microblog sentiment classification via multitask learning. In *AAAI*, pages 3059–3065, 2016.

[Xu *et al.*, 2015] Jingwei Xu, Yuan Yao, Hanghang Tong, XianPing Tao, and Jian Lu. Ice-breaking: Mitigating cold-start recommendation problem by rating comparison. In *IJCAI*, pages 3981–3987, 2015.

[Zhang *et al.*, 2014a] Mi Zhang, Jie Tang, Xuchen Zhang, and Xiangyang Xue. Addressing cold start in recommender systems: a semi-supervised co-training algorithm. In *SIGIR*, pages 73–82, 2014.

[Zhang *et al.*, 2014b] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, pages 83–92, 2014.