Learning Distributed Representations from Reviews for Collaborative Filtering

Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, Aaron Courville
Département d'Informatique et de Recherche Opérationelle
Université de Montréal
{amjad.almahairi, kyle.kastner, kyunghyun.cho, aaron.courville}@umontreal.ca

ABSTRACT

Recent work has shown that collaborative filter-based recommender systems can be improved by incorporating side information, such as natural language reviews, as a way of regularizing the derived product representations. Motivated by the success of this approach, we introduce two different models of reviews and study their effect on collaborative filtering performance. While the previous state-of-the-art approach is based on a latent Dirichlet allocation (LDA) model of reviews, the models we explore are neural network based: a bag-of-words product-of-experts model and a recurrent neural network. We demonstrate that the increased flexibility offered by the product-of-experts model allowed it to achieve state-of-the-art performance on the Amazon review dataset, outperforming the LDA-based approach. However, interestingly, the greater modeling power offered by the recurrent neural network appears to undermine the model's ability to act as a regularizer of the product representations.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering

Keywords

Recommender Systems; Neural Networks; Deep Learning

1. INTRODUCTION

Recommendation systems are a crucial component of many e-commerce enterprises, providing businesses with metrics to direct consumers to items they may find appealing. A general goal of these systems is to predict a user's preference for a certain product, often represented as an integer-valued rating, e.g., between 1 (unsatisfied) and 5 (satisfied).

In order to predict the user's preference for a product, it is often beneficial to consider as many sources of information as possible, including the preference of the user for other products, the preferences of other users, as well as any side information such as characteristics of each user and product. A data-driven approach based on this idea is called *collaborative filtering*.

Collaborative filtering has been successfully used for recommendation systems (see, e.g., [17]). A typical approach to using collaborative filtering for recommendation systems is to consider all the observed ratings given by a set of users to a set of products as elements in a matrix, where the row and column of this matrix correspond to users and products, respectively. As the observed ratings is typically only a small subset of the possible ratings (all users rating all products), this matrix is sparse. The goal of collaborative filtering is to fill in the missing values of this matrix: to predict, for each user, the rating of products the user has not rated. In this setting, collaborative filtering is usually cast as a problem of matrix factorization with missing values [10, 16, 18]. The sparse matrix is factorized into a product of two matrices of lower rank representing a user matrix and a product matrix. Once these matrices are estimated, a missing observation can be trivially reconstructed by taking a dot product of a corresponding user vector (or representation) and a product vector (or representation).

In this formulation of collaborative filtering, an important issue of data sparsity arises. For instance, the dataset provided as a part of the Netflix Challenge[1] had only 100,480,507 observed ratings out of more than 8 billion possible ratings[2] (user / product pairs) meaning that 99% of the values were missing. This data sparsity easily leads to naive matrix factorization overfitting the training set of observed ratings [10].

In this paper, we are interested in regularizing the collaborative filtering matrix factorization using an additional source of information: reviews written by users in natural language. Recent work has shown that better rating prediction can be obtained by incorporating this kind of text-based side information [13, 12, 1]. Motivated by these recent successes, here we explore alternative approaches to exploiting this side information. Specifically, we study how different models of reviews can impact the performance of the regularization.

We introduce two approaches to modeling reviews and compare these to the current state-of-the-art LDA-based approaches [13, 12]. Both models have previously been studied as neural-network-based document models. One is based on the Bag-of-Words Paragraph Vector [11]. This model is similar to the existing LDA-based model, but, as we argue, it offers a more flexible natural language model. The other is

[1] `http://www.netflixprize.com/`
[2] 480,189 users and 17,770 movies

RecSys'15, September 16–20, 2015, Vienna, Austria.
© 2015 ACM. ISBN 978-1-4503-3692-5/15/09 ...$15.00.
DOI: http://dx.doi.org/10.1145/2792838.2800192.

a recurrent neural network (RNN) based approach. RNNs have recently become very popular models of natural language for a wide array of tasks [11]. Here we will find that despite the considerable additional modelling power brought by the RNN, it does not offer better performance when used as a regularizer in this context.

The proposed approaches are empirically evaluated on the Amazon Reviews Dataset [13]. We observe that the proposed bag-of-words language model outperforms the existing approach based on latent Dirichlet allocation (LDA, [4]). We also confirm the use of an RNN language model does not lead to improved performance. Overall, our experiments demonstrate that, in this particular application where we rely on the document model to regularize the collaborative filtering matrix factorization, controlling the model flexibility is very important.

We also make methodological contributions in studying the effect of the train / test splits used in experimentation. Previous works on this subject (e.g. [13], [12] and [1]), do not clearly identify how the data was split into train and test sets. Here we empirically demonstrate the importance of doing so. We show that for a given fixed split, conclusions regarding the relative performance of competing approaches do generalize to other splits, but comparing absolute performance across difference splits is highly problematic.

2. MATRIX FACTORIZATION FOR COLLABORATIVE FILTERING

Let us assume that we are given a set $R = \{r_{u,i}\}_{(u,i)\in O_R}$ of observed ratings, where $r_{u,i} \in \{1, 2, \cdots, 5\}$ is the rating given by the user u to the product i. Collaborative filtering aims at building a model that is able to predict the rating of an unobserved user-product pair, i.e., $r_{u,i}$ where $(u, i) \notin O_R$.

In collaborative filtering based on matrix factorization, we estimate each user-product rating as

$$r_{u,i} \approx \hat{r}_{u,i} = \mu + \beta_u + \beta_i + \boldsymbol{\gamma}_u^\top \boldsymbol{\gamma}_i, \qquad (1)$$

where μ, β_i and β_u are a global bias, a user-specific bias for the user u and a product-specific bias for the product i, respectively. The vectors $\boldsymbol{\gamma}_u$ and $\boldsymbol{\gamma}_i$ are the latent factors of the user u and the product i respectively.

We estimate all the parameters in the r.h.s of Eq. (1) by minimizing the mean-squared error between the predicted ratings and the observed, true ratings:

$$C_R(\boldsymbol{\theta}) = \frac{1}{|O_R|} \sum_{(u,i)\in O_R} \left(\hat{r}_{u,i} - r_{u,i}\right)^2, \qquad (2)$$

where $\boldsymbol{\theta} = \left\{\mu, \{\beta_u\}_{u=1}^N, \{\beta_i\}_{i=1}^M, \{\gamma_u\}_{u=1}^N, \{\boldsymbol{\gamma}_i\}_{i=1}^M\right\}$.

Once the parameters $\boldsymbol{\theta}$ are estimated by minimizing C_R, it is straightforward to predict the rating of an unobserved user-product pair (u, i) using Eq. (1).

2.1 Taming the Curse of Data Sparsity

It has been observed earlier, for instance in [10], that this matrix factorization approach easily overfits the observed ratings, leading to poor generalization performance on the held-out set, or unseen user-product pairs. This issue is especially serious in the case of recommendation systems, as it is highly likely that each user purchases/watches only a fraction of all the available products. For instance, in the Amazon Reviews Dataset more than 99.999% of ratings, or elements in the rating matrix are missing.

The issue of overfitting is often addressed by adding a regularization term Ω to the cost C_R in Eq. (2). One of the most widely used regularization term is a simple weight decay

$$\Omega(\boldsymbol{\theta}) = \sum_{\theta\in\boldsymbol{\theta}} \|\theta\|^2 .$$

Hence parameters are estimated by minimizing $C_R(\boldsymbol{\theta}) + \lambda\Omega(\boldsymbol{\theta})$, where λ is a regularization coefficient.

Another approach is to interpret matrix factorization in a probabilistic framework [10, 16, 18]. In this approach, all the parameters such as the user and product representations are considered as latent random variables on which the distribution of the rating, an observed random variable, is conditioned. This probabilistic matrix factorization can automatically regularize itself by maintaining the confidence of the estimates/predictions based on the observations.

On the other hand, we can improve generalization, hence reduce overfitting, by simultaneously estimating the parameters $\boldsymbol{\theta}$ of the matrix factorization to perform well on another related task [5]. With a model predicting a rating by a user on a product, we can consider letting the model also try to account for the product review given by the user. This seems like a useful side task as users often write reviews that justify their ratings and describe features that affected their opinions.

In this paper, we explore this approach of exploiting extra tasks to improve generalization performance of collaborative filtering based on matrix factorization.

3. REGULARIZING WITH EXTRA DATA

3.1 Reviews as Extra Data

In many e-commerce systems, each rating is often accompanied with a user's review of the product. As mentioned earlier, it is natural to expect that the accompanying review is used by the user to justify her/his rating or opinion, which suggests the possibility of improving the generalization performance of the prediction model for ratings [13]. As an illustrating example, a user, who wrote "this is a great adventure movie that children and adults alike would love!" for the movie "Free Willy", is likely to give a higher rating to this movie.[3]

In this section, we will propose two approaches to utilizing this type of review data for improving the generalization performance of a rating prediction model based on matrix factorization.

3.2 Natural Language Review Modeling

More technically, let us suppose that the set R of ratings (see Sec. 2) is accompanied with a set D of reviews $D = \{d_{u,i}\}_{(u,i)\in O_D}$. Each review $d_{u,i} = \left(w_{u,i}^{(1)}, \cdots, w_{u,i}^{(n_{u,i})}\right)$ is a piece of natural language text written by a user u about an item i, which we represent as a sequence of words.

Following the multitask learning framework [5], we build a model that jointly predicts the rating given by a user u to a product i and models the review written by the user u

[3] This is an actual sample from the Amazon Reviews datatset.

on the product i. The model has two components; matrix factorization in Eq. (1) and review modeling, which shares some of the parameters $\boldsymbol{\theta}$ from the rating prediction model.

Here, we follow the approach from [13] by modeling the conditional probability of each review given the corresponding product $\boldsymbol{\gamma}_i$:

$$p\left(d_{u,i} = \left(w_{u,i}^{(1)}, \cdots, w_{u,i}^{(n_{u,i})}\right) \mid \boldsymbol{\gamma}_i, \boldsymbol{\theta}_D\right), \tag{3}$$

where $\boldsymbol{\theta}_D$ is a set of parameters for this review model.

We estimate the parameters of this review model ($\boldsymbol{\theta}_D$ and $\boldsymbol{\gamma}_i$'s) by minimizing the negative log-likelihood:

$$\underset{\boldsymbol{\theta}_D, \{\boldsymbol{\gamma}_i\}_{i=1}^M}{\arg\min} \; C_D(\boldsymbol{\theta}_D, \{\boldsymbol{\gamma}_i\}_{i=1}^M),$$

where

$$C_D(\boldsymbol{\theta}_D, \{\boldsymbol{\gamma}_i\}_{i=1}^M) = \tag{4}$$
$$-\frac{1}{|O_D|} \sum_{(u,i)\in O_D} \log p\left(d_{u,i} = \left(w_{u,i}^{(1)}, \cdots, w_{u,i}^{(n_{u,i})}\right) \mid \boldsymbol{\gamma}_i\right). \tag{5}$$

We jointly optimize the rating prediction model in Eq. (1) and the review model in Eq. (3) by minimizing the convex combination of C_R in Eq. (2) and C_D in Eq. (4):

$$\underset{\boldsymbol{\theta}, \boldsymbol{\theta}_D}{\arg\min} \, \alpha \, C_R(\boldsymbol{\theta}) + (1-\alpha) C_D(\boldsymbol{\theta}_D, \{\boldsymbol{\gamma}_i\}_{i=1}^M), \tag{6}$$

where the coefficient α is a hyperparmeter.

3.2.1 BoWLF: Distributed Bag-of-Word

The first model we propose to use is a distributed bag-of-words prediction. In this case, we represent each review as a bag of words, meaning

$$d_{u,i} = \left(w_{u,i}^{(1)}, \cdots, w_{u,i}^{(n_{u,i})}\right) \approx \left\{w_{u,i}^{(1)}, \cdots, w_{u,i}^{(n_{u,i})}\right\}. \tag{7}$$

This leads to

$$p(d_{u,i} \mid \boldsymbol{\gamma}_i) = \prod_{t=1}^{n_{u,i}} p(w_{u,i}^{(t)} \mid \boldsymbol{\gamma}_i).$$

We model $p(w_{u,i}^{(t)} \mid \boldsymbol{\gamma}_i)$ as an affine transformation of the product representation $\boldsymbol{\gamma}_i$ followed by, so-called softmax, normalization:

$$p(w_{u,i}^{(t)} = j \mid \boldsymbol{\gamma}_i) = \frac{\exp\{y_j\}}{\sum_{l=1}^{|V|} \exp\{y_l\}}, \tag{8}$$

where

$$\mathbf{y} = \mathbf{W}\boldsymbol{\gamma}_i + \mathbf{b}$$

and V, $\mathbf{W}$ and $\mathbf{b}$ are the vocabulary, a weight matrix and a bias vector. The parameters $\boldsymbol{\theta}_D$ of this review model include $\mathbf{W}$ and $\mathbf{b}$.

When we use this distributed bag-of-words together with matrix factorization for predicting ratings, and we call this joint model the *bag-of-words regularized latent factor model* (BoWLF).

3.2.2 LMLF: Recurrent Neural Network

The second model of reviews we propose to use is a recurrent neural network (RNN) language model (LM) [14]. Unlike the distributed bag-of-words model, this RNN-LM does not make any assumption on how each review is represented, but takes a sequence of words as it is, preserving the order of the words.

In this case, we model the probability over a review which is a variable-length sequence of words by rewriting the probability as

$$\begin{aligned} &p(d_{u,i} = (w_{u,i}^{(1)}, \cdots, w_{u,i}^{(n_{u,i})}) \mid \boldsymbol{\gamma}_i) \\ &= p\left(w_{u,i}^{(1)} \mid \boldsymbol{\gamma}_i\right) \prod_{t=2}^{n_{u,i}} p\left(w_{u,i}^{(t)} \mid w_{u,i}^{(1)}, \cdots, w_{u,i}^{(t-1)}, \boldsymbol{\gamma}_i\right), \end{aligned}$$

We approximate each conditional distribution with

$$p\left(w_{u,i}^{(t)} = j \mid w_{u,i}^{(<t)}, \boldsymbol{\gamma}_i\right) = \frac{\exp\left\{y_j^{(t)}\right\}}{\sum_{l=1}^{|V|} \exp\left\{y_l^{(t)}\right\}},$$

where

$$\mathbf{y}^{(t)} = \mathbf{W}\mathbf{h}^{(t)} + \mathbf{b}$$

and

$$\mathbf{h}^{(t)} = \phi\left(\mathbf{h}^{(t-1)}, w_{u,i}^{(t-1)}, \boldsymbol{\gamma}_i\right).$$

There are a number of choices available for implementing the recurrent function ϕ. Here, we use a long short-term memory (LSTM, [9]) which has recently been applied successfully to natural language-related tasks [7].

In the case of the LSTM, the recurrent function ϕ returns, in addition to its hidden state $\mathbf{h}^{(t)}$, the memory cell $\mathbf{c}^{(t)}$ such that

$$\left[\mathbf{h}^{(t)}; \mathbf{c}^{(t)}\right] = \phi\left(\mathbf{h}^{(t-1)}, \mathbf{c}^{(t-1)}, w_{u,i}^{(t-1)}, \boldsymbol{\gamma}_i\right),$$

where

$$\begin{aligned} \mathbf{h}^{(t)} &= \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}) \\ \mathbf{c}^{(t)} &= \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}. \end{aligned}$$

The output $\mathbf{o}$, forget $\mathbf{f}$ and input $\mathbf{i}$ gates are computed by

$$\begin{aligned} \begin{bmatrix} \mathbf{o}^{(t)} \\ \mathbf{f}^{(t)} \\ \mathbf{i}^{(t)} \end{bmatrix} = \sigma(&\mathbf{V}_g \mathbf{E}\left[w_{u,i}^{(t-1)}\right] + \mathbf{W}_g \mathbf{h}^{(t-1)} + \\ &\mathbf{U}_g \mathbf{c}^{(t-1)} + \mathbf{A}_g \boldsymbol{\gamma}_i + \mathbf{b}_g), \end{aligned} \tag{9}$$

and the new memory content $\tilde{\mathbf{c}}^{(t)}$ by

$$\begin{aligned} \tilde{\mathbf{c}}^{(t)} = \tanh(&\mathbf{V}_c \mathbf{E}\left[w_{u,i}^{(t-1)}\right] + \mathbf{W}_c \mathbf{h}^{(t-1)} + \\ &\mathbf{U}_c \mathbf{c}^{(t-1)} + \mathbf{A}_c \boldsymbol{\gamma}_i + \mathbf{b}_c), \end{aligned} \tag{10}$$

where $\mathbf{E}$, $\mathbf{V}_g$, $\mathbf{W}_g$, $\mathbf{U}_g$, $\mathbf{b}_g$, $\mathbf{V}_c$, $\mathbf{W}_c$, $\mathbf{U}_c$, $\mathbf{b}_c$, $\mathbf{A}_g$ and $\mathbf{A}_c$ are the parameters of the RNN-LM. Note that $\mathbf{E}[w]$ denotes a row indexing by the word index w of the matrix $\mathbf{E}$.

Similarly to the BoWLF, we call the joint model of matrix factorization and this RNN-LM the *language model regularized latent factor model* (LMLF).

3.3 Related Work: LDA-based Approach

Similar approaches of modeling reviews to regularize matrix factorization have recently been proposed, however, with different review models such as LDA [13, 12] and non-negative matrix factorization [1]. Here, we describe "Hidden Factors as Topics" (HFT) recently proposed in [13], and discuss it with respect to the proposed approaches.

The HFT model is based on latent Dirichlet allocation (LDA, [4]), and similarly to the distributed bag-of-word model in Sec. 3.2.1, considers each review as a bag of words (see Eq. (7).) Thus, we start by describing how LDA models a review $d_{u,i}$.

LDA is a generative model of a review/document. It starts by sampling a so-called topic proportion $\boldsymbol{\tau}$ from a Dirichlet distribution. $\boldsymbol{\tau}$ is used as a parameter to a multinomial topic distribution from which a topic is sampled. The sampled topic defines a probability distribution over the words in a vocabulary. In other words, given a topic proportion, the LDA models a review with a mixture of multinomial distributions.

Instead of sampling the topic proportion from the top-level Dirichlet distribution in LDA, HFT replaces it with

$$\boldsymbol{\tau} = \frac{1}{\left\|\exp\left\{\kappa \boldsymbol{\gamma}_i\right\}\right\|_1} \exp\left\{\kappa \boldsymbol{\gamma}_i\right\},$$

where κ is a free parameter estimated along with all the other parameters of the model. In this case, the probability over a single review $d_{u,i}$ given a product $\boldsymbol{\gamma}_i$ becomes

$$\begin{aligned} p(d_{u,i} \mid \boldsymbol{\gamma}_i) &= \prod_{t=1}^{n_{u,i}} \sum_{k=1}^{\dim(\boldsymbol{\gamma}_i)} p(w_{u,i}^{(t)} \mid z_k = 1) p(z_k = 1 \mid \boldsymbol{\gamma}_i) \\ &= \prod_{t=1}^{n_{u,i}} \sum_{k=1}^{\dim(\boldsymbol{\gamma}_i)} \tau_k p(w_{u,i}^{(t)} \mid z_k = 1) \end{aligned} \tag{11}$$

where z_k is an indicator variable of the k-th topic out of $\dim(\boldsymbol{\gamma}_i)$, and τ_k is the k-th element of $\boldsymbol{\tau}$. The conditional probability over words given a topic is modeled with a stochastic matrix $\mathbf{W}^* = \left[w_{j,k}^*\right]_{|V| \times \dim(\boldsymbol{\gamma}_i)}$ (each column sums to 1). The conditional probability over words given a product $\boldsymbol{\gamma}_i$ can be written as

$$p(w_{u,i}^{(t)} = j \mid \boldsymbol{\gamma}_i) = \sum_{k=1}^{\dim(\boldsymbol{\gamma}_i)} w_{j,k}^* \frac{\exp\left\{\kappa \gamma_{i,k}\right\}}{\left\|\exp\left\{\kappa \boldsymbol{\gamma}_i\right\}\right\|_1}. \tag{12}$$

The matrix $\mathbf{W}^*$ is often parametrized by $w_{j,k}^* = \frac{\exp\left\{q_{j,k}\right\}}{\sum_l \exp\left\{q_{l,k}\right\}}$, where $\mathbf{Q} = [q_{j,k}]$ is an unconstrained matrix of the same size as $\mathbf{W}^*$. In practice, a bias term is added to the formulation above to handle frequent words.

3.4 Comparing HFT and BoWLF

From Eq. (8) and Eq. (12), we can see that the HFT and the proposed BoWLF (see Sec. 3.2.1) are closely related. Most importantly, both of them consider a review as a bag of words and parametrize the conditional probability of a word given a product representation with a single affine transformation (weight matrix plus offset vector).

The main difference is in how the product representation and the weight matrix interact to form a point on the $|V|$-dimensional simplex. In the case of HFT, both the product representation $\boldsymbol{\gamma}_i$ and the projection matrix $\mathbf{W}^*$ are separately stochastic (i.e. each $\boldsymbol{\gamma}_i$ and each column of $\mathbf{W}^*$ are interpretable as a probability distribution), while the BoWLF projects the result of the matrix-vector product $\mathbf{W}\boldsymbol{\gamma}_i$ onto the probability simplex.

This can be understood as the difference between a mixture of experts and a product of experts [8]. On a per word basis, the BoWLF in Eq. (8) can be re-written as a (conditional) product of experts by

$$p(w = j \mid \boldsymbol{\gamma}_i) = \frac{1}{Z(\boldsymbol{\gamma}_i)} \prod_{k=1}^{\dim(\boldsymbol{\gamma}_i)} \exp\left\{w_{j,k}\gamma_{i,k} + b_j\right\},$$

where $w_{j,k}$ and b_j are the element at the j-th row and k-th column of $\mathbf{W}$ and the j-th element of $\mathbf{b}$, respectively. On the other hand, an inspection of Eq. (11) reveals that, on a per word basis, the HFT model is clearly a mixture model, with the topics playing the role of the mixture components.

As argued in [8], a product of experts can more easily model a *peaky* distribution, especially, in a high-dimensional space.[4] The reviews of each product tend to contain a small common subset of the whole vocabulary, while those subsets vastly differ from each other depending on the product. In other words, the conditional distribution of words given a product puts most of its probability mass on only a few product-specific words, while leaving most other words with nearly zero probabilities. Product of experts are naturally better suited to modeling peaky distributions rather than mixture models.

A more concrete way of understanding the difference between HFT and BoWLF may be to consider how the product representation and the weight matrix interact. In the case of the BoWLF, this is a simple matrix-vector product with no restrictions on the weight matrix. This means that both the product representation elements as well as the elements of the weight matrix are free to assume negative values. Thus, it is possible that an element of the product representation could exercise a strong influence *suppressing* the expression of a given set of words. Alternatively, with HFT model, as the model interprets the elements of the product representation as mixture components, these elements have no mechanism of suppressing probability mass assigned to words by the other elements of the product representation.

We suggest that this difference allows the BoWLF to better model reviews compared to the HFT, or any other LDA-based model by offering a mechanism for negative correlations between words to be explicitly expressed by elements of the product representation. By offering a more flexible and natural model of reviews, the BoWLF model can improve the rating prediction generalization performance. As we will see in Sec. 4, our experimental results support this proposition.

The proposed LMLF takes one step further by modeling each review with a chain of products of experts taking into account the order of words. This may seem an obvious benefit at the first sight. However, it is not clear whether the order of the words is specific to a product or is simply a feature of language itself. In the latter case, we expect that LMLF will model reviews very well, but may not improve rating prediction.

4. EXPERIMENTS

4.1 Dataset

We evaluate the proposed approaches on the Amazon Reviews dataset [13].[5] There are approximate 35 million ratings and accompanying reviews from 6,643,669 users and

[4] Note that the size of a usual vocabulary of reviews is on the order of thousands.

[5] https://snap.stanford.edu/data/web-Amazon.html

2,441,053 products. The products are divided into 28 categories such as music and books. The reviews are on average 110 words long. We refer the reader to [12] for more detailed statistics.

4.2 Experimental Setup

Data Preparation.

We closely follow the procedure from [13] and [12], where the evaluation is done per category. We randomly select 80% of ratings, up to two million samples, as a training set, and split the rest evenly into validation and test sets, for each category. We preprocess reviews only by tokenizing them using a script from Moses[6], after which we build a vocabulary of 5000 most frequent words.

Evaluation Criteria.

We use mean squared error (MSE) of the rating prediction to evaluate each approach. For assessing the performance on review modeling, we use the average negative log-likelihood.

Baseline.

We compare the two proposed approaches, BoWLF (see Sec. 3.2.1) and LMLF (see Sec. 3.2.2), against three baseline methods; matrix factorization with L_2 regularization (MF, see Eqs. (1)–(2)), the HFT model from [13] (see Sec. 3.3) and the RMR model from [12]. In the case of HFT, we report the performance both by evaluating the model ourselves[7] and by reporting the results from [13] directly. For RMR, we only report the results from [12].

Hyper-parameters.

Both user $\boldsymbol{\gamma}_u$ and product $\boldsymbol{\gamma}_i$ vectors in Eq. (1) are five dimensional for all the experiments in this section. This choice was made mainly to make the results comparable to the previously reported ones in [13] and [12].

We initialize all the user and product representations by sampling each element from a zero-mean Gaussian distribution with its standard deviation set to 0.01. The biases, μ, β_u and β_i are all initialized to 0. All the parameters in BoWLF and LMLF are initialized similarly except for the recurrent weights of the RNN-LM in LMLF which were initialized to be orthogonal.

Training Procedure.

When training MF, BoWLF and LMLF, we use minibatch RMSProp with the learning rate, momentum coefficient and the size of minibatch set to 0.01, 0.9 and 128, respectively. We trained each model at most 200 epochs, while monitoring the validation performance. For HFT, we follow [13] which uses the Expectation Maximization algorithm together with L-BFGS. In all cases, we early-stop each training run based on the validation set performance.

In the preliminary experiments, we found the choice of α in Eq. (6), which balances matrix factorization and review modeling, to be important. We searched for the α that maximizes the validation performance, in the range of $[0.1, 0.01]$.

We used a CPU cluster of 16 nodes each with 8 cores and $8 - 16$ GB of memory to run experiments on BoWLF, MF, and HFT. For LMLF, we used a cluster of K20 GPUs where we had up to 50 GPUs available.

4.3 Rating Prediction Results

We list results of the experiments in Table 1 for the 28 categories in terms of MSE with the standard error of mean shown in parentheses. From this table, we can see that except for a single category of "Jewelry", the proposed BoWLF outperforms all the other models with an improvement of 20.29% over MF and 5.64% over HFT across all categories.[8] In general, we note better performance of BoWLF and LMLF models over other methods especially as the size of the dataset grows, which is evident from Figs. 1 and 2.

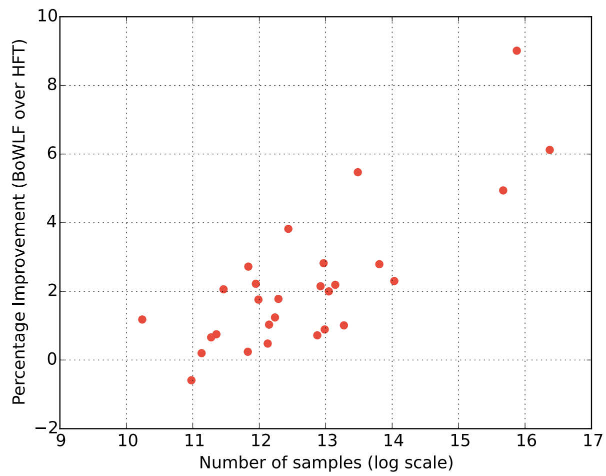

Figure 1: Scatterplot showing performance improvement over the number of samples. We see a performance improvement of BoWLF over HFT as dataset size increases.

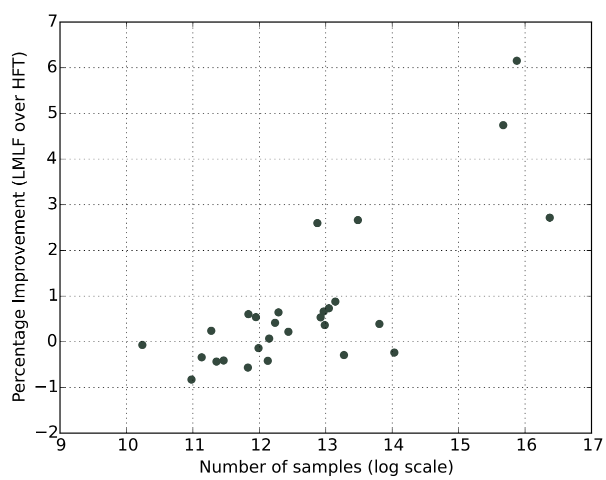

Figure 2: Scatterplot showing performance improvement over the number of samples. We see a modest performance improvement of LMLF over HFT as dataset size increases.

[6] `https://github.com/moses-smt/mosesdecoder/`

[7] The code was kindly provided by the authors of [13].

[8] Due to the use of different splits, the results by HFT reported in [13] and RMR in [12] are not directly comparable.

Dataset	Dataset Size	(a) MF	(b) HFT	(c) BoWLF	(d) LMLF	BoWLF improvement over (a)	over (b)	HFT*	RMR**
Arts	27K	1.434 (0.04)	1.425 (0.04)	**1.413** (0.04)	1.426 (0.04)	2.15%	1.18%	1.388	1.371
Jewelry	58K	1.227 (0.04)	**1.208** (0.03)	1.214 (0.03)	1.218 (0.03)	1.24%	-0.59%	1.178	1.160
Watches	68K	1.511 (0.03)	1.468 (0.03)	**1.466** (0.03)	1.473 (0.03)	4.52%	0.20%	1.486	1.458
Cell Phones	78K	2.133 (0.03)	2.082 (0.02)	**2.076** (0.02)	2.077 (0.02)	5.76%	0.66%	N/A	2.085
Musical Inst.	85K	1.426 (0.02)	1.382 (0.02)	**1.375** (0.02)	1.388 (0.02)	5.12%	0.75%	1.396	1.374
Software	95K	2.241 (0.02)	2.194 (0.02)	**2.174** (0.02)	2.203 (0.02)	6.70%	2.06%	2.197	2.173
Industrial	137K	0.360 (0.01)	0.354 (0.01)	**0.352** (0.01)	0.356 (0.01)	0.76%	0.24%	0.357	0.362
Office Products	138K	1.662 (0.02)	1.656 (0.02)	**1.629** (0.02)	1.646 (0.02)	3.32%	2.72%	1.680	1.638
Gourmet Foods	154K	1.517 (0.02)	1.486 (0.02)	**1.464** (0.02)	1.478 (0.02)	5.36%	2.22%	1.431	1.465
Automotive	188K	1.460 (0.01)	1.429 (0.01)	**1.419** (0.01)	1.428 (0.01)	4.17%	1.03%	1.428	1.403
Kindle Store	160K	1.496 (0.01)	1.435 (0.01)	**1.418** (0.01)	1.437 (0.01)	7.83%	1.76%	N/A	1.412
Baby	184K	1.492 (0.01)	1.437 (0.01)	**1.432** (0.01)	1.443 (0.01)	5.95%	0.48%	1.442	N/A
Patio	206K	1.725 (0.01)	1.687 (0.01)	**1.674** (0.01)	1.680 (0.01)	5.10%	1.24%	N/A	1.669
Pet Supplies	217K	1.583 (0.01)	1.554 (0.01)	**1.536** (0.01)	1.544 (0.01)	4.74%	1.78%	1.582	1.562
Beauty	252K	1.378 (0.01)	1.373 (0.01)	**1.335** (0.01)	1.370 (0.01)	4.33%	3.82%	1.347	1.334
Shoes	389K	0.226 (0.00)	0.231 (0.00)	**0.224** (0.00)	0.225 (0.00)	0.23%	0.72%	0.226	0.251
Tools & Home	409K	1.535 (0.01)	1.498 (0.01)	**1.477** (0.01)	1.490 (0.01)	5.78%	2.15%	1.499	1.491
Health	428K	1.535 (0.01)	1.509 (0.01)	**1.481** (0.01)	1.499 (0.01)	5.35%	2.82%	1.528	1.512
Toys & Games	435K	1.411 (0.01)	1.372 (0.01)	**1.363** (0.01)	1.367 (0.01)	4.71%	0.89%	1.366	1.372
Video Games	463K	1.566 (0.01)	1.501 (0.01)	**1.481** (0.01)	1.490 (0.01)	8.47%	2.00%	1.511	1.510
Sports	510K	1.144 (0.01)	1.137 (0.01)	**1.115** (0.01)	1.127 (0.01)	2.94%	2.19%	1.136	1.129
Clothing	581K	0.339 (0.00)	0.343 (0.00)	**0.333** (0.00)	0.344 (0.00)	0.60%	1.01%	0.327	0.336
Amazon Video	717K	1.317 (0.01)	1.239 (0.01)	**1.184** (0.01)	1.206 (0.01)	13.33%	5.47%	N/A	1.270
Home	991K	1.587 (0.00)	1.541 (0.00)	**1.513** (0.00)	1.535 (0.01)	7.41%	2.79%	1.527	1.501
Electronics	1.2M	1.754 (0.00)	1.694 (0.00)	**1.671** (0.00)	1.698 (0.00)	8.29%	2.30%	1.724	1.722
Music	6.3M	1.112 (0.00)	0.970 (0.00)	**0.920** (0.00)	0.924 (0.00)	19.15%	4.94%	0.969	0.959
Movies & Tv	7.8M	1.379 (0.00)	1.089 (0.00)	**0.999** (0.00)	1.022 (0.00)	37.95%	9.01%	1.119	1.120
Books	12.8M	1.272 (0.00)	1.141 (0.00)	**1.080** (0.00)	1.110 (0.00)	19.21%	6.12%	1.135	1.113
All categories	35.3M	1.289	1.143	1.086	1.107	20.29%	5.64%		

Table 1: Prediction Mean Squared Error results on test data. Standard error of mean in parenthesis. Dimensionality of latent factors $\dim(\gamma_i) = 5$ for all models. Best results for each dataset in bold. HFT* and RMR represent original paper results over different data splits [13, 12].**

Interestingly, BoWLF always outperforms LMLF. These results indicate that the complex language model, which the LMLF learns using an LSTM network, does not seem to improve over a simple bag-of-word representation, which the BoWLF learns, in terms of the learned product representations.

This can be understood from how the product representation, which is used *linearly* by the rating prediction model, is handled by each model. The word distribution modeled by the BoWLF depends linearly on the product representation, which requires the product-related structure underlying reviews be encoded linearly as well. On the other hand, LMLF *nonlinearly* manipulates the product representation to approximate the distribution over reviews. In other words, the LMLF does not necessarily encode the underlying product-related structure inside the product representation in the way the rating prediction model can easily decode [15].

4.4 Impact of Training / Test Data Split

Comparing the results of the original HFT paper with the results we get training over the same split, it becomes clear that models trained on different splits are not directly comparable. To further explore the importance of the chosen split for model selection, we perform experiments over five randomly selected folds and compare each model on every fold.

One of the challenges in pursuing empirical work on the Amazon review dataset is the current absence of a standard

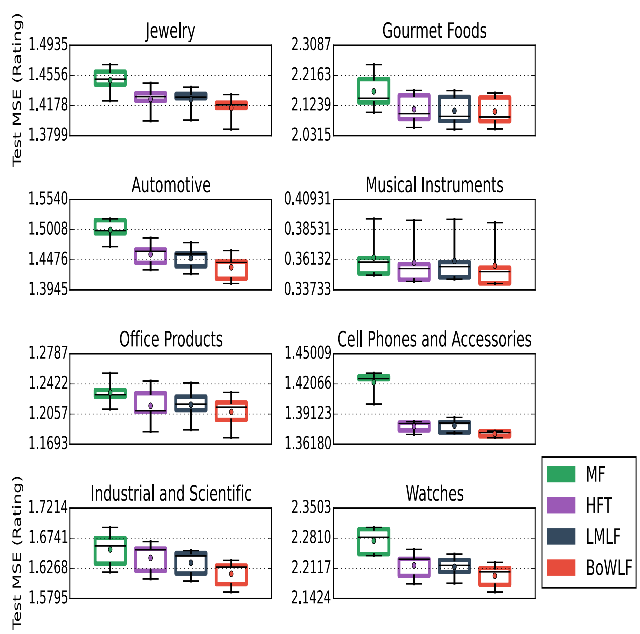

Figure 3: Box and whisker plot showing K-fold ($K = 5$) experiments. Point represents the mean over all folds. Center line represents median. Box extents represent 25^{th} and 75^{th} percentile. Whisker extents show minimum and maximum values.

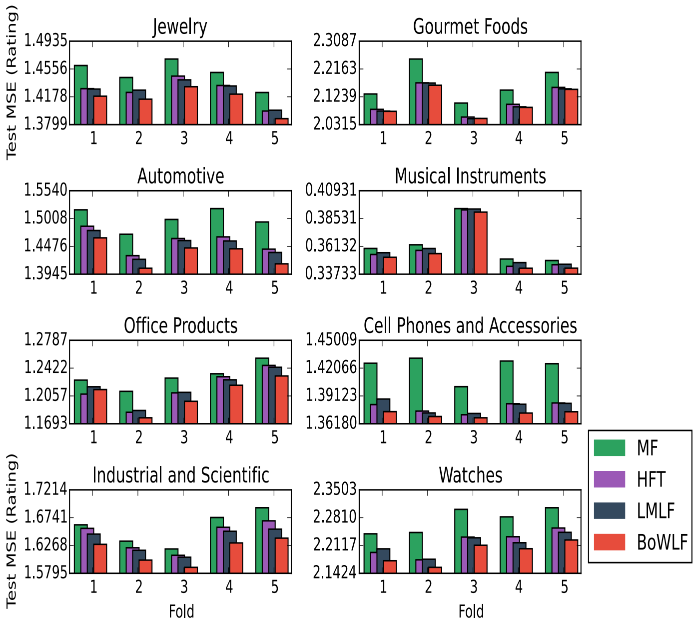

Figure 4: Bar chart showing showing K-fold ($K = 5$) experiments. Although values across folds vary, relative performance is consistent.

train / test split of the data. Here we evaluate the importance of establishing a standard data split.

Fig. 3 shows the results of experiments comparing the performance of each model over different splits. We perform 5-fold validation. That is, each model is trained 5 times, each on 80% of the data and we report performance on the remaining 20%. The result on the test reveal several important points. First, we note that the variance over splits can be large, meaning that comparing across different splits could be misleading when performing model selection. On the other hand, as shown in Fig 4, the relative performance of each model is consistent over the different splits. This implies that a single random split can be used for model selection and evaluation as long as this split is held constant over all evaluated models.

Taken together, Figs. 3 and 4 illustrate the importance of standardizing the dataset splits. Without standardization, performance measured between different research groups becomes incomparable and the use of this dataset as a benchmark is rendered difficult or even impossible.

4.5 Effect of Language Model

One way to analyze models which use text information is to compare their negative log-likelihood (NLL) scores on the same test dataset. We find BoWLF has a stronger language model than HFT, which is reflected in the NLL results, and in this case it appears to contribute to a better rating prediction. As shown in Fig. 5, LMLF has a much better language model than both HFT and BoWLF, but as discussed earlier, LMLF *does not* lead to better rating predictions than BoWLF. LMLF appears to be largely equivalent to HFT in prediction strength, despite having a much better language model. As discussed above in Sec. 4.3, this suggests that the strong nonlinearity in the LSTM helps modeling

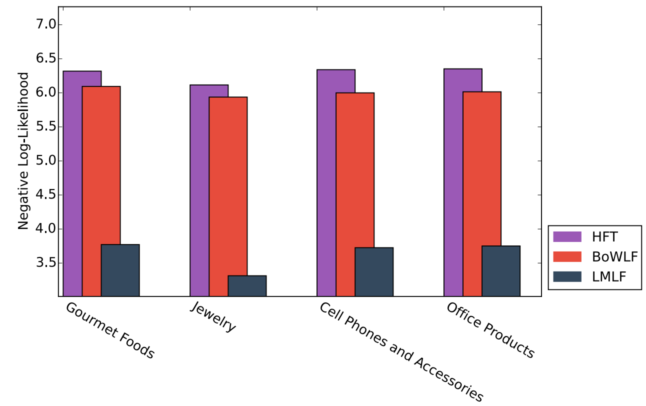

Figure 5: Bar chart showing showing negative log-likelihood (NLL) on test data for several datasets. LMLF is superior in NLL but does not improve rating prediction over BoWLF.

reviews, but not necessarily result in the linearly-decodable product representation, leading to less improvement in rating prediction.

Contrary to LDA-based approaches, the latent dimensions of the product representations learned by BoWLF do not necessarily have clear interpretations as topics. However, the neighborhoods learned by BoWLF are interpretable, where we use the cosine distance between two product representations, i.e., $1 - \frac{\gamma_i^\top \gamma_j}{\|\gamma_i\|_2 \|\gamma_j\|_2}$. The BoWLF product space neighbors seem qualitatively superior to the neighbors given by HFT as seen in Table 2. Note in particular the association of "MTR Simply Tomato Soup" with other soups by BoWLF, while HFT neighbors seem much broader, including crackers, noodles, and gummy bears. This observation is consistent with the interpretation of the differences in the mathematical form of HFT and BoWLF (as argued in Sec. 3.4). The ability of BoWLF to form peakier distributions over words, given the product representation, allows the model to be more discriminating and more closely group similar products. Furthermore, we can see that the neighbors based on the product representations from the LMLF are qualitatively worse than those from the BoWLF, which indirectly confirms that the underlying product-related structure encoded by the LMLF is more difficult to extract linearly.

While drawing firm conclusions from this small set of neighbors is obviously ill-advised, the general trend appears to hold in more extensive testing. Broadly speaking, this further strengthens the idea that stronger product representations lead to improvements in rating prediction.

5. DISCUSSION

We develop two new models (BoWLF and LMLF) which exploit text reviews to regularize rating prediction on the Amazon Reviews datasets. BoWLF achieves state of the art results on 27 of the 28 datasets, while LMLF outperforms HFT (but not BoWLF) as dataset size increases. Additionally, we explore the methodology behind the choice of data split, clearly demonstrating that models trained on different data subsets cannot be directly compared. Performing K-fold crossvalidation ($K = 5$), we confirm that BoWLF

Product	HFT	BoWLF	LMLF
Extra Spearmint Sugarfree Gum	Hong Kong Fu Xiang Yuan Moon Cakes French Chew - Vanilla Peck's Anchovette	Dubble Bubble Gum Trident Sugarless White Gum Gold Mine Nugget Bubble Gum	Gumballs Special Assorted Bazooka Bubble Gum Gourmet Spicy Beef Jerky
Dark Chocolate Truffle	Tastykake Kreamies Kakes Ceam .. Miko - Awase Miso Soyabean Paste Haribo Berries Gummi Candy	Ritter Sport Corn Flakes Chocolate Chocolate Dobosh Torte Sugar Free, Milk Chocolate Pecan Turtles	Fantis Grape Leaves Grape Flavoring Tutti Fruitti Flavoring
MTR Simply Tomato Soup	Wellington Cracked Pepper Crackers Maggi Instant Noodles Haribo Gummi Candy	MTR Mulligatawny Soup hai Kitchen Coconut Ginger Soup Miko - Awase Miso Soyabean Paste	Muir Glen Organic Soup Soy Ginger Saba Noodles Alessi Soup

Table 2: Nearest neighbors (cosine similarity) based on product representations estimated by HFT, BoWLF and LMLF, for Gourmet Foods dataset. Qualitatively, the ability to regularize the product representations seems to correlate well with the quality of the neighbourhoods formed in product representation space.

achieves superior performance across dataset splits. The resulting product neighborhoods measured by cosine similarity between product representations are intuitive, and correspond with human analysis of the data. Overall we find that BoWLF has a 20.29% average improvement over basic matrix factorization and a 5.64% average improvement over HFT.

We found that the proposed LMLF slightly lagged behind the BoWLF. As we discuss above, we believe this could be due to the nonlinear nature of language model based on a recurrent neural network. This nonlinearity results in the product-related structure underlying reviews being nonlinearly encoded in the product representation, which cannot be easily extracted by the linear rating prediction model. However, this will need to be further investigated in addition to analyzing the exact effect of language modeling on prediction performance.

Acknowledgements

We would like to thank the developers of Theano [3, 2] and Pylearn2 [6], for developing such a powerful tool for scientific computing. We are grateful to Compute Canada and Calcul Québec for providing us with powerful computational resources.

6. REFERENCES

[1] Y. Bao, H. Fang, and J. Zhang. TopicMF: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, 2014.

[2] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[3] J. Bergstra, F. Bastien, O. Breuleux, P. Lamblin, R. Pascanu, O. Delalleau, G. Desjardins, D. Warde-Farley, I. J. Goodfellow, A. Bergeron, and Y. Bengio. Theano: Deep learning on gpus with python. In *Big Learn workshop, NIPS'11*, 2011.

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

[5] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[6] I. J. Goodfellow, D. Warde-Farley, P. Lamblin, V. Dumoulin, M. Mirza, R. Pascanu, J. Bergstra, F. Bastien, and Y. Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[7] A. Graves. Generating sequences with recurrent neural networks. Technical report, arXiv:1308.0850, 2013.

[8] G. E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*, volume 1, pages 1–6, Edinburgh, Scotland, 1999. IEE.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[10] A. Ilin and T. Raiko. Practical approaches to principal component analysis in the presence of missing values. *The Journal of Machine Learning Research*, 11:1957–2000, 2010.

[11] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[12] G. Ling, M. R. Lyu, and I. King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 105–112, New York, NY, USA, 2014. ACM.

[13] 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*, RecSys '13, pages 165–172, New York, NY, USA, 2013. ACM.

[14] T. Mikolov. *Statistical Language Models based on Neural Networks*. PhD thesis, Brno University of Technology, 2012.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations: Workshops Track*, 2013.

[16] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[17] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender systems handbook*, volume 1. Springer, 2011.

[18] 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.