Connecting Social Media to E-Commerce: Cold-Start Product Recommendation using Microblogging Information

Wayne Xin Zhao *Member, IEEE*, Sui Li, Yulan He, Edward Y. Chang, Ji-Rong Wen *Senior Member, IEEE* and Xiaoming Li *Senior Member, IEEE*

Abstract—In recent years, the boundaries between e-commerce and social networking have become increasingly blurred. Many e-commerce websites support the mechanism of social login where users can sign on the websites using their social network identities such as their Facebook or Twitter accounts. Users can also post their newly purchased products on microblogs with links to the e-commerce product web pages. In this paper we propose a novel solution for *cross-site cold-start product recommendation* which aims to recommend products from e-commerce websites to users at social networking sites in "cold-start" situations, a problem which has rarely been explored before. A major challenge is how to leverage knowledge extracted from social networking sites for cross-site cold-start product recommendation.

We propose to use the linked users across social networking sites and e-commerce websites (users who have social networking accounts and have made purchases on e-commerce websites) as a bridge to map users' social networking features to another feature representation for product recommendation. In specific, we propose learning both users' and products' feature representations (called user embeddings and product embeddings, respectively) from data collected from e-commerce websites using recurrent neural networks and then apply a modified gradient boosting trees method to transform users' social networking features into user embeddings. We then develop a feature-based matrix factorization approach which can leverage the learnt user embeddings for cold-start product recommendation. Experimental results on a large dataset constructed from the largest Chinese microblogging service SINA WEIBO and the largest Chinese B2C e-commerce website JINGDONG have shown the effectiveness of our proposed framework.

Index Terms—e-commerce, product recommender, product demographic, microblogs, recurrent neural networks

1 INTRODUCTION

In recent years, the boundaries between e-commerce and social networking have become increasingly blurred. E-commerce websites such as eBay features many of the characteristics of social networks, including real-time status updates and interactions between its buyers and sellers. Some e-commerce websites also support the mechanism of *social login*, which allows new users to sign in with their existing login information from social networking services such as Facebook, Twitter or Google+. Both Facebook and Twitter have introduced a new feature last year that allow users to buy products directly from their websites by clicking a "buy" button to purchase items in adverts or other posts. In China, the e-commerce company ALIBABA has made a strategic investment in SINA WEIBO[1] where ALIBABA product adverts can be directly delivered to SINA WEIBO users. With the new trend of conducting e-commerce activities on social networking sites, it is important to leverage knowledge extracted from social networking sites for the development of product recommender systems.

In this paper, we study an interesting problem of recommending products from e-commerce websites to users at social networking sites who do not have historical purchase records, i.e., in "cold-start" situations. We called it *cross-site cold-start product recommendation*. Although online product recommendation has been extensively studied before [1], [2], [3], most studies only focus on constructing solutions within certain e-commerce websites and mainly utilise users' historical transaction records. To the best of our knowledge, *cross-site cold-start product recommendation* has been rarely studied before.

In our problem setting here, only the users' social networking information is available and it is a challenging task to transform the social networking information into latent user features which can be effectively used for product recommendation. To address

- *W. X. Zhao (contact author) and J. Wen are with School of Information in Renmin University of China, China. Both of them are also with Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China.*
- *S. Li and X. Li are with School of Electronic Engineering and Computer Science in Peking University, China.*
- *Y. He is with School of Engineering and Applied Science in Aston University, UK.*
- *E. Chang is with Research and Innovation at HTC.*

1. http://www.reuters.com/article/2013/04/29/net-us-sinaweibo-alibaba-stake-idUSBRE93S0DA20130429

this challenge, we propose to use the linked users across social networking sites and e-commerce websites (users who have social networking accounts and have made purchases on e-commerce websites) as a bridge to map users' social networking features to latent features for product recommendation. In specific, we propose learning both users' and products' feature representations (called user embeddings and product embeddings, respectively) from data collected from e-commerce websites using recurrent neural networks and then apply a modified gradient boosting trees method to transform users' social networking features into user embeddings. We then develop a feature-based matrix factorization approach which can leverage the learnt user embeddings for cold-start product recommendation.

We built our dataset from the largest Chinese microblogging service SINA WEIBO[2] and the largest Chinese B2C e-commerce website JINGDONG[3], containing a total of 20,638 linked users. The experimental results on the dataset have shown the feasibility and the effectiveness of our proposed framework.

Our major contributions are summarised below:

- We formulate a novel problem of recommending products from an e-commerce website to social networking users in "cold-start" situations. To the best of our knowledge, it has been rarely studied before.
- We propose to apply the recurrent neural networks for learning correlated feature representations for both users and products from data collected from an e-commerce website.
- We propose a modified gradient boosting trees method to transform users' microblogging attributes to latent feature representation which can be easily incorporated for product recommendation.
- We propose and instantiate a feature-based matrix factorization approach by incorporating user and product features for cold-start product recommendation.

2 PROBLEM FORMULATION

Given an e-commerce website, let $\mathcal{U}$ denote a set of its users, $\mathcal{P}$ a set of products and $\mathbf{R}$ a $|\mathcal{U}| \times |\mathcal{P}|$ purchase record matrix, each entry $r_{u,p}$ of which is a binary value indicating whether u has purchased product p. Each user $u \in \mathcal{U}$ is associated with a set of purchased products with the purchase timestamps. Furthermore, a small subset of users in $\mathcal{U}$ can be linked to their microblogging accounts (or other social network accounts), denoted as $\mathcal{U}^L$. As such, each user $u \in \mathcal{U}^L$ is also associated with their respective microblogging attribute information. Let $\mathcal{A}$ denote the set of microblogging features, and each microblogging user has a $|\mathcal{A}|$-dimensional microblogging feature vector $\mathbf{a}_u$, in which each entry $a_{u,i}$ is the attribute value for the i-th microblogging attribute feature.

With the notations introduced above, we define our recommendation problem as follows. We consider a cross-site cold-start scenario: a microblogging user $u' \notin \mathcal{U}$ is new to the e-commerce website, who has no historical purchase records. It is easy to see $u' \notin \mathcal{U}^L$, too, since we have $\mathcal{U}^L \subset \mathcal{U}$. We aim to generate a personalised ranking of recommended products for u' based on her microblogging attributes $\mathbf{a}_{u'}$.

Due to the heterogeneous nature between these two different data signals, information extracted from microblogging services cannot usually be used directly for product recommendation on e-commerce websites. Therefore, one major challenge is how to transform users' microblogging attribute information $\mathbf{a}_{u'}$ into another feature representation $\mathbf{v}_{u'}$, which can be used more effectively for product recommendation. Here, we call $\mathbf{a}_{u'}$ the *original* or *microblogging feature representation* and $\mathbf{v}_{u'}$ the (heterogeneous) *transformed feature representation*, respectively.

Next, we will study how to extract microblogging features and transform them into a distributed feature representation before presenting a feature-based matrix factorization approach, which incorporates the learned distributed feature representations for product recommendation. The entire workflow of our solution is shown in Figure 1 which consists of four major steps splitting into *feature mapping* and *product recommendation*, which will be discussed in Section 3 and 4 respectively.

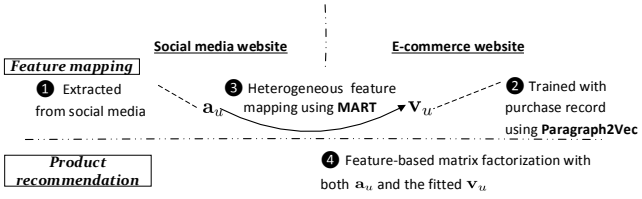

Fig. 1. The workflow diagram for our presented solution.

3 EXTRACTING AND REPRESENTING MICROBLOGGING ATTRIBUTES

Our solution to microblogging feature learning consists of three steps:

- Prepare a list of potentially useful microblogging attributes and construct the microblogging feature vector $\mathbf{a}_u$ for each linked user $u \in \mathcal{U}^L$;
- Generate distributed feature representations $\{\mathbf{v}_u\}_{u \in \mathcal{U}}$ using the information from all the users $\mathcal{U}$ on the e-commerce website through deep learning;
- Learn the mapping function, $f(\mathbf{a}_u) \rightarrow \mathbf{v}_u$, which transforms the microblogging attribute information $\mathbf{a}_u$ to the distributed feature representations

2. http://weibo.com
3. http://www.jd.com

$\mathbf{v}_u$ in the second step. It utilises the feature representation pairs $\{\mathbf{a}_u, \mathbf{v}_u\}$ of all the linked users $u \in \mathcal{U}^L$ as training data.

3.1 Microblogging Feature Selection

In this section, we study how to extract rich user information from microblogs to construct $\mathbf{a}_u$ for a microblogging user. We consider three groups of attributes.

Demographic Attributes

A demographic profile (often shortened as "a demographic") of a user such as sex, age and education can be used by e-commerce companies to provide better personalised services. We extract users' demographic attributes from their public profiles on SINA WEIBO. Demographic attributes have been shown to be very important in marketing, especially in product adoption for consumers [4]. Following our previous study [5], we identify six major demographic attributes: gender, age, marital status, education, career and interests.To quantitatively measure these attributes, we have further discretized them into different bins following our previously proposed method described in [5].

Text Attributes

Recent studies have revealed that microblogs contain rich commercial intents of users [5], [6]. Also, users' microblogs often reflect their opinions and interests towards certain topics. As such, we expect a potential correlation between text attributes and users' purchase preferences. We perform Chinese word segmentation and stopword removal before extracting two types of text attributes below.

Topic distributions. Seroussi et al. ([7]) proposed to extract topics from user-generated text using the Latent Dirichlet Allocation (LDA) model for recommendation tasks. Follow the same idea, we first aggregate all the microblogs by a user into a document, and then run the standard LDA to obtain the topic distributions for each user. The benefits of topics distributions over keywords are two fold. First, the number of topics is usually set to $50 \sim 200$ in practice, which largely reduces the number of dimensions to work with. Second, topic models generate condense and meaningful semantic units, which are easier to interpret and understand than keywords.

Word embeddings. Standard topic models assume individual words are exchangeable, which is essentially the same as the *bag-of-words* model assumption. Word representations or embeddings learned using neural language models help addressing the problem of traditional bag-of-word approaches which fail to capture words' contextual semantics [8], [9]. In word embeddings, each dimension represents a latent feature of the word and semantically similar words are close in the latent space. We employ the Skip-gram model implemented by the tool word2vec[4] to learn distributed representations of words. Finally, we average the word vectors of all the tokens in a user's published document as the user's embedding vector.

Network Attributes

In the online social media space, it is often observed that users connected with each other (e.g., through following links) are likely to share similar interests. As such, we can parse out latent user groups by the users' following patterns assuming that users in the same group share similar purchase preferences.

Latent group preference. Since it is infeasible to consider all users on WEIBO and only keeping the top users with the most followers would potentially miss interesting information, we propose to use topic models to learn latent groups of followings as in [10]. We treat a following user as a token and aggregate all the followings of a user as an individual document. In this way, we can extract latent user groups sharing similar interests (called "following topics"), and we represent each user as a preference distribution over these latent groups.

Temporal Attributes

Temporal activity patterns are also considered since they reflect the living habits and lifestyles of the microblogging users to some extent. As such, there might exist correlations between temporal activities patterns and users' purchase preferences.

Temporal activity distributions. We consider two types of temporal activity distributions, namely daily activity distributions and weekly activity distributions. The *daily activity distribution* of a user is characterised by a distribution of 24 ratios, and the i-th ratio indicates the average proportion of tweets published within the i-th hour of a day by the user; similarly *weekly activity distribution* of a user is characterised by a distribution of seven ratios, and the i-th ratio indicates the average proportion of tweets published within the i-th day of a week by the user.

We summarize all types of features in Table 1.

TABLE 1
Categorisation of the microblogging features. The number of feature dimensions are shown in parentheses.

Categories	Features
Demographic Attributes	Gender (2), Age (6), Marital status (10), Education (7), Career (9), Interests (6)
Text Attributes	Topic distributions (50), Word embeddings (50)
Network Attributes	Latent group preference (50)
Temporal Attributes	Daily activity distribution (24), Weekly activity distribution (7)

4. https://code.google.com/p/word2vec

3.2 Distributed Representation Learning With Recurrent Neutral Networks

In Section 3.1, we have discussed how to construct the microblogging feature vector $\mathbf{a}_u$ for a user u. However, it is not straightforward to establish connections between $\mathbf{a}_u$ and products. Intuitively, users and products should be represented in the same feature space so that a user is closer to the products that she has purchased compared to those she has not. Inspired by the recently proposed methods in learning word embeddings using recurrent neutral networks [8], [9], we propose to learn user embeddings or distributed representation of user $\mathbf{v}_u$ in a similar way.

Learning Product Embeddings

Before presenting how to learn user embeddings, we first discuss how to learn product embeddings. The neural network methods, word2vec, proposed in [8], [9] for word embedding learning can be used to model various types of sequential data. The core idea can be summarised as follows. Given a set of symbol sequences, a fixed-length vector representation for each symbol can be learned in a latent space by exploiting the context information among symbols, in which "similar" symbols will be mapped to nearby positions. If we treat each product ID as a word token, and convert the historical purchase records of a user into a timestamped sequence, we can then use the same methods to learn product embeddings. Unlike matrix factorization, the order of historical purchases from a user can be naturally captured.

We consider two simple recurrent neutral architectures proposed in [11] to train product embeddings, namely, the Continuous Bag-Of-Words model (CBOW) and the Skip-gram model. The major difference between these two architectures lies in the direction of prediction: CBOW predicts the current product using the surrounding context, i.e., $Pr(p_t|\text{context})$, while Skip-gram predicts the context with the current product, i.e., $Pr(\text{context}|p_t)$. In our experiments, the context is defined as a window of size 4 surrounding a target product p_t which contains two products purchased before and two after p_t. More formally, each product p_t is modeled as a unique latent embedding vector $\mathbf{v}_{p_t}$, and the associated context vector is obtained to average the vectors of the context information as $\mathbf{v}_{context}$. For CBOW, the conditional prediction probability is characterized by a softmax function as follows

$$Pr(p_t|\text{context}) = \frac{\exp(\mathbf{v}_{p_t}^\top \cdot \mathbf{v}_{context})}{\sum_p \exp(\mathbf{v}_p^\top \cdot \mathbf{v}_{context})}.$$

To optimize for computing exponential sum probabilities, *hierarchical softmax* and *negative sampling techniques* are commonly used to speed up the training process. At each training iteration, we sample a target product together with their context window, and then update the parameters with Stochastic Gradient Descent (SGD) using the gradients derived by backpropogation. Learning for Skip-gram is done in a similar way, which is omitted here.

Learning User Embeddings

Given product embeddings, if we can learn user embeddings in a similar way, then we can explore the correlated representations of a user and products for product recommendation. We borrow the idea from the recently proposed Paragraph Vector (para2vec) method [9], which learns feature representations from variable-length pieces of texts, including sentences, paragraphs, and documents. We implement a simplified version of para2vec at the sentence level as follows. The purchase history of a user can be considered as a "sentence" consisting of a sequence of product IDs as word tokens. A user ID is placed at the beginning of each sentence, and both user IDs and product IDs are treated as word tokens in a vocabulary in the learning process. During training, for each sentence, the sliding context window will always include the first word (i.e., user ID) in the sentence. In this way, a user ID is essentially always associated with a set of her purchase records (a context window of 4 products at a time). We can then use the same learning procedure in word2vector for the estimation of $Pr(\text{context}|p_t)$ and $Pr(p_t|\text{context})$. We present an illustrative example of these two architectures in Fig. 2. After learning, we separate user embeddings from product embeddings and use $\mathbf{v}_u$ and $\mathbf{v}_p$ to denote the learnt K-dimensional embedding for user u and product p respectively.

The rationales of applying para2vec to model purchase data can be explained below. First, the user embedding representation for each user ID reflects the users' personalized purchase preference; Second, the surrounding context, i.e., product purchases, is used to capture the shared purchase patterns among users. Compared to the traditional matrix factorization [12], the (window-based) sequential context is additionally modeled in addition to user preference, which is expected to potentially yield better recommendation results.

3.3 Heterogenous Representation Mapping using Gradient Boosting Regression Trees

We have presented how to construct a microblogging feature vector $\mathbf{a}_u$ from a microblogging site and learn a distributed representation $\mathbf{v}_u$ from an e-commerce website respectively. In the cross-site cold-start product recommendation problem we considered in this paper (i.e., make a product recommendation to a user u who has never purchased any products from an e-commerce website), we can only obtain the microblogging feature vector $\mathbf{a}_u$ for user u. The key idea is to use a small number of linked users across sites as a bridge to learn a function which maps the original feature

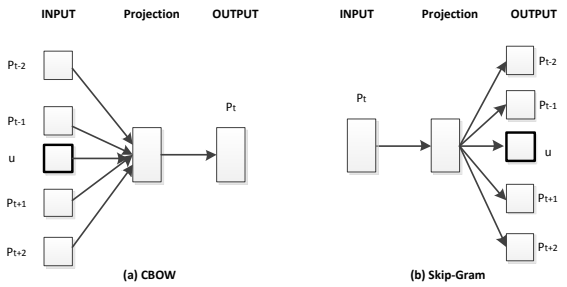

Fig. 2. Two architectures to learn both product and user embeddings. Here u denote a user ID. The major difference between para2vec and word2vec lies in the incorporation of user ID as additional context.

representation $\mathbf{a}_u$ to the distributed representation $\mathbf{v}_u$. Specifically, we can construct a training set consisting of feature vector pairs, $\{\mathbf{a}_u, \mathbf{v}_u\}_{u \in \mathcal{U}^L}$ and cast the feature mapping problem as a supervised regression task: the input is a microblogging feature vector $\mathbf{a}_u$ and the output is a distributed feature vector $\mathbf{v}_u$.

Assume that $\mathbf{v}_u$ contains K dimensions, we need to learn a set of K functions $\{f^{(i)}\}_{i=1}^K$, and the i-th function $f^{(i)}$ takes the original feature vector of a user u as the input and returns the corresponding i-th transformed feature value $v_{u,i}$, i.e., $v_{u,i} = f^{(i)}(\mathbf{a}^{(u)})$. We extend the Multiple Additive Regression Tree (MART) [13] method to learn feature mapping functions since it is powerful to capture higher-order transformation relationship between input and output.

A brief Introduction of MART

Gradient boosting algorithms aim to produce an ensemble of weak models that together form a strong model in a stage-wise process. Typically, a weak model is a J-terminal node Classification And Regression Tree (CART) [14] and the resulting gradient boosting algorithm is called Multiple Additive Regression Tree (MART) [13]. An input feature vector $\mathbf{x} \in \mathbf{R}^d$ is mapped to a score $F(\mathbf{x}) \in \mathbf{R}$.

The final model is built in a stage-wise process by performing gradient descent in the function space. At the mth boosting,

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta \rho_m h_m(\mathbf{x}; \mathbf{a}), \tag{1}$$

where each $h_m(\cdot)$ is a function parameterised by $\mathbf{a}_m$, $\rho_m \in \mathbf{R}$ is the weight associated with the mth function, and $0 < \eta \leq 1$ is the learning rate. The learning procedure of gradient boosting consists of two alternative steps in the m-th iteration: first fit a new component function h_m by using the steepest-descent method, and then minimize the loss function to derive the ensemble weight ρ_m for the learnt learner. At each iteration, we use the regularized squared error function to learn a new CART component: we first derive a set of disjoint regions $\{R_j\}$ which covers the space of all the joint values of the input feature vector, and then set the region fitting coefficient for R_j to the average of "pseudo responses" of the instances falling in R_j.

Completeness-Based Feature Sampling

An issue about the gradient boosting algorithm is that it tends to overfit the training data. It has been previously shown that the incorporation of randomized feature sampling improves the tree based ensemble methods in Random Forest [15]. Inspired by the idea, we propose to use an attribute-level importance sampling method where each attribute is assigned with an importance score and at each node split in building the MART trees, we only sample a fraction of attributes (empirically set to $\frac{2}{3}$) based on each attribute's importance score instead of enumerating all the attributes. Once an attribute is sampled, its corresponding attribute value features will be selected subsequently. The importance score of each attribute is set to the proportion of the attribute values that can be extracted from the users' public profiles on SINA WEIBO. Another benefit of completeness-based sampling is that attributes with a larger proportion of missing values will be more likely to be pushed to the leaf nodes, which alleviates the missing value problem in regression trees.

Fitting Refinement

Here we propose two methods to refine the fitted values. First, the fitting quality relies on the number of available linked users since insufficient training data would hurt the performance of the regression method. Recall that we can learn the user embeddings for all the users on an e-commerce website. We create a super user embedding vector $\mathbf{v}^{(sup)}$ by averaging all available user embeddings. When the training data is limited, we require that the fitted vector should not deviate from $\mathbf{v}^{(sup)}$ too much.

Second, we fit each dimension separately with an individual MART model. Based on our data analysis, we found that the values of some dimensions from the same user might be correlated. We compute pairwise Pearson Correlation Coefficient (PCC) for every two dimensions using all the learnt user embeddings from the e-commerce website, and construct the correlation matrix $\mathbf{W}^{K \times K}$, where each entry $w_{i,j}$ indicates the correlation degree between two dimensions. We convert all negative values to zero.

We then propose to take into account both methods to refine the initially fitted value $\mathbf{v}_u^{(0)}$ in the following way

$$\begin{aligned} \min \quad & \sum_k (v_{u,k} - v_{u,k}^{(0)})^2 + \mu_1 \sum_k (v_{u,k} - v_{u,k}^{(sup)})^2 \\ & + \mu_2 \sum_{k,k',k \neq k'} w_{k,k'} (v_{u,k} - v_{u,k'})^2, \end{aligned} \tag{2}$$

where μ_1 and μ_2 are the tuning parameters. The parameter μ_1 is used to "smooth" the data when the number of training instances is small or a user has very little microblogging information. While in other cases, μ_1 can be simply set to a small value, e.g., 0.05. For μ_2, we have found a value of 0.05 usually gives good performance. By setting the derivative w.r.t. $v_{u,k}$ to 0, we derive an iterative formula as follows

$$v_{u,k} \leftarrow \frac{v_{u,k}^{(0)} + \mu_1 v_{u,k}^{(sup)} + \mu_2 \sum_{k',k' \neq k} w_{k,k'} v_{u,k'}}{1 + \mu_1 + \mu_2 \sum_{k',k' \neq k} w_{k,k'}}. \quad (3)$$

Summary

We have built a single learner for each dimension in the transformed feature representation $\mathbf{v}_u$ using a modified gradient boosting trees model. The reason why we choose MART is that its components are regression trees, and trees are shown to be effective to generate high-order and interpretable knowledge using simple plain features [14], [16], [17]. Note other tree-based ensemble methods can apply here, such as Random Forest (RF)[15]. In our experiments, we have found MART is sightly better than RF, and therefore we adopt MART as the fitting model.

4 Applying The Transformed Features to Cold-Start Product Recommendation

Once the MART learners are built for feature mapping, the original microblogging feature vectors $\mathbf{a}_u$ are mapped onto the user embedding $\mathbf{v}_u$. In this section, we study how to incorporate $\{\mathbf{a}_u, \mathbf{v}_u\}$ into the feature-based matrix factorization technique. In specific, we develop our recommendation method based on the recently proposed SVDFeature [18]. Our idea can also be applied to other feature-based recommendation algorithms, such as Factorization Machines [19].

4.1 The General SVDFeature Framework for Product Recommendation

SVDFeature [18] is built based on the traditional matrix factorization approach, and it considers factorization in three aspects, namely global features (also called as *dyadic features*), user features and item features. It can be formulated for the task of product recommendation as follows

$$\begin{aligned} & \hat{r}_{u,p}(\boldsymbol{\alpha}^{(u)}, \boldsymbol{\beta}^{(p)}, \boldsymbol{\gamma}^{(u,p)}) \\ = \; & \mu + \sum_j b_j^{(G)} \gamma_j^{(u,p)} + \sum_j b_j^{(U)} \alpha_j^{(u)} + \sum_j b_j^{(P)} \beta_j^{(p)} \\ & + (\sum_j \alpha_j^{(u)} \mathbf{x}_j)^\top (\sum_j \beta_j^{(p)} \mathbf{y}_j), \end{aligned} \quad (4)$$

where $\boldsymbol{\alpha}^{(u)} \in \mathbb{R}^{N_\alpha}$, $\boldsymbol{\beta}^{(p)} \in \mathbb{R}^{N_\beta}$ and $\boldsymbol{\gamma}^{(u,p)} \in \mathbb{R}^{N_\gamma}$ are the input vectors consisting of the features of user u, the features of product p and the global features for the pair (u, p) with the lengths of N_α, N_β and N_γ respectively. Here, $b_j^{(G)}$, $b_j^{(U)}$ and $b_j^{(P)}$ are the global, user and product bias parameters respectively. The latent vectors $\mathbf{x}_j$ and $\mathbf{y}_j$ capture the j-th user feature and the j-th product feature respectively. Let $\{\mathbf{x}_j\}$ and $\{\mathbf{y}_j\}$ denote the set of all user features and product features respectively. Note that $\{\mathbf{x}_j\}$ are shared by all the users, $\{\mathbf{y}_j\}$ are shared by all the products, and the global features and bias values do not have any corresponding latent vectors. In summary, a user-product pair corresponds to a feature vector concatenated by global features, user features and product features. The response value to be fitted indicates whether the user has purchased the product or not.

Feature Coding with the Side Information

We discuss how to incorporate the user and product information into the SVDFeature framework.

Coding users and products: For users, we reserve the first $|\mathcal{U}|$ dimensions in the user input vector. Each user u is coded as a vector of $|\mathcal{U}|$-dimensional vector consists of a "1" in the uth dimension and "0" in other dimensions; Similarly, we can reserve the first $|\mathcal{P}|$ dimensions in the product input vector to code the products. Formally, we have

$$\alpha_j^{(u)} = \begin{cases} 1, & j = u; \\ 0, & j \neq u. \end{cases} \qquad \beta_j^{(p)} = \begin{cases} 1, & j = p; \\ 0, & j \neq p. \end{cases}$$

Coding microblogging attributes: Given a user u, we use the dimensions from $(|\mathcal{U}|+1)$-th to $(|\mathcal{U}|+|\mathcal{A}|)$-th to code her microblogging attribute vector $\mathbf{a}_u$. For $i = 1$ to $|\mathcal{A}|$, we have $\alpha_{|\mathcal{U}|+i}^{(u)} = a_{u,i}$. Here we follow [20] to directly incorporate microblogging attributes. In practice, a subset of features $\mathcal{A}'$ can be identified with expertise knowledge instead of using the full set of features in $\mathcal{A}$.

Coding user embeddings: Given a user u, we use the dimensions from $(|\mathcal{U}|+|\mathcal{A}|+1)$-th to $(|\mathcal{U}|+|\mathcal{A}|+K)$-th to code her distributed feature vector (user embedding) $\mathbf{v}_u$. For $k = 1$ to K, we have $\alpha_{|\mathcal{U}|+k}^{(u)} = v_{u,k}$.

Coding product embeddings: Given a product p, we use the dimensions from $(|\mathcal{P}|+1)$-th to $(|\mathcal{P}|+K)$-th to code the product embedding $\mathbf{v}_p$. For $k = 1$ to K, we have $\beta_{|\mathcal{P}|+k}^{(p)} = v_{p,k}$.

Coding the global user-product feature: Since we have both user embeddings and product embeddings, we can incorporate a global feature to denote a similarity degree between a user and a product. The idea is that a user is more likely to buy a product which is closer in the unified latent feature space, therefore the corresponding entry should receive a larger global bias value. We define a global feature as follows

$$\gamma_1^{(u,p)} = \text{sim}(\mathbf{v}_u, \mathbf{v}_p),$$

where the cosine similarity is used to implement the function $\text{sim}(\cdot,\cdot)$.

With these coded features, for a user-product pair (u,p), we have the following factorization formula

$$\begin{aligned}&\hat{r}_{u,p}(\boldsymbol{\alpha}^{(u)},\boldsymbol{\beta}^{(p)},\boldsymbol{\gamma}^{(u,p)}) \\ =\ & \mu + b_1^{(G)}\gamma_1^{(u,p)} + \sum_j b_j^{(U)}\alpha_j^{(u)} + \sum_j b_j^{(P)}\beta_j^{(p)} + \\ & \left(\mathbf{x}_u + \sum_{i=1}^{|\mathcal{A}|} a_{u,i}\mathbf{x}_i + \sum_{k=1}^{K} v_{u,k}\mathbf{x}_k\right)^\top \left(\mathbf{y}_p + \sum_{k=1}^{K} v_{p,k}\mathbf{y}_k\right).\end{aligned} \tag{5}$$

We use Θ to denote the parameters to learn, $\{\mu, b_1^{(G)}, \{b_j^{(U)}, \mathbf{x}_j\}, \{b_j^{(P)}, \mathbf{y}_j\}\}$[5].

Parameter Learning

We employ the pairwise ranking model for parameter learning. Given a user u, we generate the positive-negative pairs of products (p,p') in which u has purchased p (called *positive*) but not p' (called *negative*). The pairwise ranking model assumes that the fitted value for the purchased product is larger than the one that has not been purchased by a user, i.e., $Pr(\hat{r}_{u,p} > \hat{r}_{u,p'})$. Furthermore, we use the sigmoid function as the loss function

$$Pr(\hat{r}_{u,p} > \hat{r}_{u,p'}) = \frac{1}{1+e^{-(\hat{r}_{u,p}-\hat{r}_{u,p'})}}.$$

Note that for pairwise ranking, we do not need to learn the user bias parameters $\{b_j^{(U)}\}$. With the above partial-order rank probability function, the overall regularized ranking loss function can be written as follows

$$\begin{aligned}\mathcal{L} =\ & -\sum_{u\in\mathcal{U}}\sum_{(p,p')\in\mathcal{D}_u} \log\frac{1}{1+e^{-(\hat{r}_{u,p}-\hat{r}_{u,p'})}} + \sum_j \lambda_1 \parallel \mathbf{x}_j \parallel_2^2 \\ & + \sum_j \lambda_2 \parallel \mathbf{y}_j \parallel_2^2 + \lambda_3 \parallel b_1^{(G)} \parallel_2^2 + \lambda_4 \sum_j \parallel b_j^{(P)} \parallel_2^2,\end{aligned}$$

where $\mathcal{D}_u$ denotes the positive-negative pairs for user u, and λs are the coefficients for ridge regularization. By minizing the loss function $\mathcal{L}$, we use the stochastic gradient descent method (SGD) to learn the model parameters. Given a training instance consisting of a user u and a positive-negative pair (p,p'), the derivatives at this instance for updating the model parameters are presented as follows

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{x}_u} &= -e^u_{p>p'}\left\{\Delta\mathbf{y}_{p,p'} + \sum_{k'=1}^{K}\mathbf{y}_{k'}\Delta v_{p,p',k'}\right\} + 2\lambda_1\mathbf{x}_u, \\ \frac{\partial\mathcal{L}}{\partial\mathbf{x}_i} &= -a_{u,i}e^u_{p>p'}\left\{\Delta\mathbf{y}_{p,p'} + \sum_{k'=1}^{K}\mathbf{y}_{k'}\Delta v_{p,p',k'}\right\} + 2\lambda_1\mathbf{x}_i, \\ \frac{\partial\mathcal{L}}{\partial\mathbf{x}_k} &= -v_{u,k}e^u_{p>p'}\left\{\Delta\mathbf{y}_{p,p'} + \sum_{k'=1}^{K}\mathbf{y}_{k'}\Delta v_{p,p',k'}\right\} + 2\lambda_1\mathbf{x}_k, \\ \frac{\partial\mathcal{L}}{\partial\mathbf{y}_p} &= -e^u_{p>p'}\bar{\mathbf{x}}^u + 2\lambda_2\mathbf{y}_p, \\ \frac{\partial\mathcal{L}}{\partial\mathbf{y}_{p'}} &= e^u_{p>p'}\bar{\mathbf{x}}^u + 2\lambda_2\mathbf{y}_{p'}, \\ \frac{\partial\mathcal{L}}{\partial\mathbf{y}_k} &= -e^u_{p>p'}\left(v_{p,k}\bar{\mathbf{x}}^u - v_{p',k}\bar{\mathbf{x}}^u\right) + 2\lambda_2\mathbf{y}_k, \\ \frac{\partial\mathcal{L}}{\partial b_1^{(G)}} &= -e^u_{p>p'}\left(\gamma_1^{(u,p)} - \gamma_1^{(u,p')}\right) + 2\lambda_3 b_1^{(G)}, \\ \frac{\partial\mathcal{L}}{\partial b_j^{(P)}} &= -e^u_{p>p'}\left(\beta_j^{(p)} - \beta_j^{(p')}\right) + 2\lambda_4 b_j^{(P)},\end{aligned}$$

where $\Delta\mathbf{y}_{p,p'} = \mathbf{y}_p - \mathbf{y}_{p'}$, $\Delta v_{p,p',k'} = v_{p,k'} - v_{p',k'}$, $e^u_{p>p'} = 1 - Pr(\hat{r}_{u,p} > \hat{r}_{u,p'})$, $\bar{\mathbf{x}}^u = \mathbf{x}_u + \sum_{i=1}^{|\mathcal{A}|} a_{u,i}\mathbf{x}_i + \sum_{k=1}^{K} v_{u,k}\mathbf{x}_k$ and $\bar{\mathbf{y}}^p = \mathbf{y}_p + \sum_{k=1}^{K} v_{p,k}\mathbf{y}_k$.

Applications in Cold-Start Product Recommendation

With the learnt models, we can recommend products from e-commerce websites to users in online social networking websites. In this scenario, the only information available is the microblogging features of users, i.e., $\mathbf{a}_u$. Using MART, we can derive the fitted user embeddings, i.e., $\hat{\mathbf{v}}_u = f(\mathbf{a}_u)$. We consider the following variants to rank candidate products with our proposed methods:

- Only with the fitted user embeddings

$$\hat{r}_{u,p} = bias + \left(\sum_{k=1}^{K}\hat{v}_{u,k}\mathbf{x}_k\right)^\top\left(\mathbf{y}_p + \sum_{k=1}^{K} v_{p,k}\mathbf{y}_k\right), \tag{6}$$

- With both the fitted user embeddings and microblogging feature vectors

$$\hat{r}_{u,p} = bias + \left(\sum_{i=1}^{|\mathcal{A}|} a_{u,i}\mathbf{x}_i + \sum_{k=1}^{K}\hat{v}_{u,k}\mathbf{x}_k\right)^\top\left(\mathbf{y}_p + \sum_{k=1}^{K} v_{p,k}\mathbf{y}_k\right), \tag{7}$$

where $bias = b^{(G)}\cdot\text{sim}_{cos}(\hat{\mathbf{v}}_u,\mathbf{v}_p) + b_p^{(P)}$. Note that all the above ranking formulae do not use the user latent vector $\mathbf{x}_u$. In another words, we do not require users made any purchases before recommending products to them. Thus, our proposed recommendation framework can be applied for cold-start recommendation.

5 EXPERIMENTS

We present experimental setup first before discussing our results.

5.1 Experimental Setup

Our task requires data from both an e-commerce website and an online social networking site.

E-commerce data. We used a large e-commerce dataset shared by [6], which contains 138.9 million

5. In order to simplify our notations, we use $\mathbf{x}_i$ to denote $\mathbf{x}_{|\mathcal{U}|+i}$, $\mathbf{x}_k$ to denote $\mathbf{x}_{|\mathcal{U}|+|\mathcal{A}|+k}$ and $\mathbf{y}_k$ to denote $\mathbf{y}_{|\mathcal{P}|+k}$.

transaction records from 12 million users on 0.2 million products. Each transaction record consists of a user ID, a product ID and the purchase timestamp. We first group transaction records by user IDs and then obtain a list of purchased products for each user.

Microblogging data. We used our previous data [5] collected from the largest Chinese microblogging site SINA WEIBO, in which we have retrieved a total of 1.7 billion tweets from 5 million active users within a half-year time span from January 2013 to June 2013.

User linkage. We have found that WEIBO users sometimes shared their purchase record on their microblogs via a system-generated short URL, which links to the corresponding product entry on JINGDONG. By following the URL link, we can obtain the JINGDONG account of the WEIBO user[6]. We identified 23,917 linked users out of 5 million active users by scanning tweets in this way. We first filter out 3,279 users with too little information on their WEIBO public profiles. Next, we further divide users into two groups. The first group contains users with more than five product purchases, denote as $\mathcal{D}_{dense}$. The second group contains the remaining users, denoted as $\mathcal{D}_{sparse}$. The statistics of these linked users are summarized in Table 2. For privacy consideration, all the WEIBO IDs and JINGDONG IDs of all linked users are replaced by anonymized unique IDs, and all their textual information and purchase information is encoded with numeric symbols.

TABLE 2
Statistics of our linked user datasets.

Datasets	#users	#products	Average #products	Average #tweets
$\mathcal{D}_{dense}$	15,853	98,900	52.0	41.0
$\mathcal{D}_{sparse}$	4,785	6,699	2.6	35.7

5.2 Evaluation on User Embeddings Fitting

Given a linked user $u \in \mathcal{U}^L$, we have the microblogging feature vector $\mathbf{a}_u$ extracted from WEIBO and the user embedding $\mathbf{v}_u$ learnt based on her JINGDONG purchase record. We use a regression-based approach to fit $\mathbf{v}_u$ with $\mathbf{a}_u$ for heterogeneous feature mapping, and the fitted vector is denoted as $\hat{\mathbf{v}}_u$. To examine the effectiveness of the regression performance, the Mean Absolute Error (MAE) is used as the evaluation metric

$$MAE = \frac{1}{|\mathcal{T}|}\Big\{\sum_{u\in\mathcal{T}}\frac{\sum_{k=1}^{K}|v_{u,k}-\hat{v}_{u,k}|}{K}\Big\}, \quad (8)$$

where $|\mathcal{T}|$ is the number of test users. We consider three different comparison methods: (1) CART [14]; (2) MART$_{old}$, which is the original implementation as in [13]; (3) MART$_{sample}$, which is our modified implementation with feature sampling; (4) MART$_{both}$, which is our modified implementation with feature sampling and fitting refinement.

For user embedding fitting, we use $\mathcal{D}_{dense}$ for evaluation, since the users in $\mathcal{D}_{dense}$ have a considerable number of purchases for learning the ground truth user embeddings using our modified para2vec method, which are more reliable for evaluation. The dataset $\mathcal{D}_{dense}$ is split *by users* into training set and test set with three different $\frac{\#train}{\#test}$ ratios, namely 1:1, 1:4 and 1:9. We use a similar evaluation method as N-fold cross validation. Given the $\frac{\#train}{\#test}$ ratio of $1:N$, each fold will be treated as the training data exactly once and the rest $N-1$ folds are treated as the test data, the process will be repeated N times and the final results are averaged over N such runs. The number of boosting iterations for all MART variants and the values of μ_1 and μ_2 for MART$_{both}$ are optimized by N-fold cross validation.

In Table 3, we can see that when the training data is relatively large (ratio 1:1), all the MART variants give similar results and they perform consistently better than the simple CART. Interestingly, when the size of training data becomes smaller, MART$_{sample}$ and MART$_{both}$ outperforms MART$_{old}$. In specific, the performance gain achieved by MART$_{both}$ over the other two MART variants is more significant with smaller set of training data. These results show that our modifications of feature sampling and fitting refinement are very effective.

TABLE 3
Performance comparisons of MAE results for fitting user embeddings on $\mathcal{D}_{dense}$. Smaller is better.

$\frac{\#train}{\#test}$	CART	MART$_{old}$	MART$_{sample}$	MART$_{both}$
1/1	0.557	**0.515**	**0.515**	**0.515**
1/4	0.557	0.522	**0.521**	**0.521**
1/9	0.564	0.589	0.558	**0.529**

Relative attribute importance. Tree-based methods offer additional feasibility to learn relative importance of each attribute. Inspired by the method introduced in [13], we calculate a statistic of the relative importance of each attribute for MART based on the training data. Recall that in MART, each feature corresponds to an attribute value. First, we traverse through all the regression trees, and calculate for each feature its contribution to the cost function by adding up the contributions of all the nodes that are split by this feature. Here we define *feature contribution* to be the reduction of the squared error in the loss function. For each attribute, we can sum up the contributions of all of its possible attribute values as its overall contribution.

The results are shown in Figure 3. We have the following observations: 1) The text attributes occupy the

6. Note that when a user shares a purchase record on her microblog, she will be notified automatically by SINA WEIBO that her JINGDONG account would be exposed to the public.

top two rank positions[7]; 2) Within the demographic category, *Gender* and *Interests* are more important than the others. 3) The social based attributes are ranked relatively lower compared to the other two categories. It seems that demographic attributes are less important than text attributes in our dataset. One possible reason is that many demographic attribute values are missing in users' public profiles on WEIBO.[8] Nevertheless, the ranking of relative importance of attributes does not entirely depend on their completeness proportion. For example, *Interests* is more important than *Latent group preference* even though the later has a larger completeness proportion. Another possible reason is that the feature dimension for text attributes is much larger than that of demographic attributes, e.g., *Topic Distribution* has fifty feature dimensions while *Gender* only has two feature dimensions.

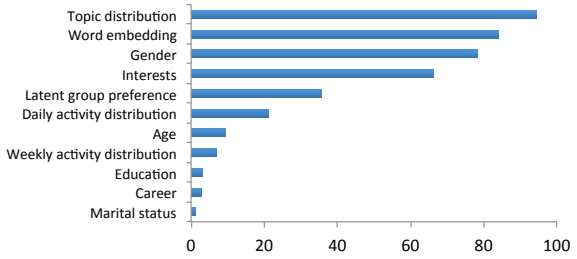

Fig. 3. Relative attribute importance ranking (corresponding to the features in Table 1).

We can also evaluate the importance of each attribute by conducting experiments on the traditional product recommendation task. We use the standard MF approach as a baseline and add attributes one at a time using the SVDFeature framework discussed in Section 4.1, then check the performance improvement yielded by the added attribute. The attribute ranking obtained in this way is similar to the ranking in Fig. 3, but the gap between text attributes and demographic attributes becomes smaller.

7. Although both topic distributions and word embeddings are used to capture the semantic characteristics of user-generated text, they have different focuses. Topic distributions are more suitable to extract topical themes from text based on word co-occurrence patterns (essentially taking the whole document as the context window) while word embeddings are more suitable to capture the semantics between words from local context windows, usually comprising 3 words before and after the target word. Hence, we keep both types of text features in our approach. It is worth noting that our method is a tree-based approach, which can effectively handle information redundancy, i.e., if a feature contains redundant information given the tree that is being constructed, it will be pushed to a lower rank during attribute selection.

8. In our dataset, the completeness proportion of demographic attributes are as follows: Gender (100%), Interests (65.7%), Age (36.7%), Education (26.3%), Career (12.9%) and Marital status (4.6%); while for text and network attributes, the proportion of completeness is about 99.1%, i.e., most users have published tweets and followed some other users.

5.3 Evaluation on Cold-Start Product Recommendation

For cold-start product recommendation, we aim to recommend products to microblog users without the knowledge of their historical purchase records.

Construction of the Evaluation Set

The evaluation set splits users into training set and test set. For the training set, we sample negative products with a ratio of 1:1 for each user, i.e., we have the same number of negative and positive products. For the test set, we randomly sample negative products with a ratio of 1:50 for each user, i.e., each positive product would involve 50 negative products. All negative products are sampled from the same product category as the corresponding positive one. For example, for "iPhone 6", we can sample "Samsung Galaxy S5" from the "Mobile Phones" category as a negative product.Given a user, we can generate a list of candidate products consisting of both positive and negative products. On average, a user has about 52 positive products and 2,600 negative products in our experimental dataset, which is indeed a challenging task. Similar to the evaluation scenario in Information Retrieval, we would like to examine the performance that a system ranks positive products over negative products.

Methods to Compare

We consider the following methods for performance comparison:

- **Popularity (Pop)**: products are ranked by their historical sale volumes.
- **Popularity with Semantic Similarity (Pop++)**: the ranking score is a combination of two scores: (1) the popularity score S_1; (2) the cosine similarity S_2 between product description and user text information, including profile, tweets and tags. The two scores are combined by $\log(1 + S_1) \times \log(1 + S_2)$.
- **Embedding Similarities (ES)**: Similarity scores $\hat{\mathbf{v}}_u^\top \cdot \mathbf{v}_p$ between a user embedding $\hat{\mathbf{v}}_u$ and a list of product embeddings $\mathbf{v}_p$ are used to rank products.
- **MF with user attributes (MFUA)**: User attributes (including user profile and topic distributions) are incorporated into the basic matrix factorisation algorithm for product rating prediction [7]. For fairness, we also use the pairwise loss function to train the model.
- **FM without User Interactions (FMUI)**: Rendle [20] applied the Factorization Machines (FM) for "follow" recommendation in KDDCup 2012. It has been found that similar performance was obtained with or without the interactions of user features. FM without user feature interactions is equivalent to SVDFeature. We reimplement this

method in the SVDFeature framework with our extracted microblogging features.

- **Cold**$_E$: Our proposed approach which uses the fitted user embedding features and product embedding features (Eq. 6).
- **Cold**$_{D+E}$: Our proposed approach which uses the microblogging features, the product embedding features and the fitted user embedding features (Eq. 7). Especially, we only use demographic attributes here, since they have been shown important to product recommendation [21], [5].
- **Cold**$_{++}$: Since the user and product embeddings can be learned for all the users and products respectively in the e-commerce website, we can train Cold$_E$ with all the users in $\mathcal{U}$, not limited to the linked users $\mathcal{U}^L$. This variant is called Cold$_{enhanced}$.

We set the regularization coefficient to a 0.004, the iteration number to 50 and the factor number to 32 for all the methods. We use the `CBOW` architecture to learn the embedding vectors based on the purchase records from all the non-linked users and the partial purchase records from linked users in our training set. The number of dimensions of embedding vectors is set to 50. The user embedding features in the test sets for different $\frac{\#training}{\#test}$ settings are set to the values fitted using MART$_{both}$. For Cold$_{enhanced}$, we add additional 10,000 randomly selected non-linked users from $\mathcal{U}$ into the training set.

Evaluation Metrics for Product Recommendation

Five widely used metrics are used for the evaluation of product recommendation results, including $Precision@k$, $Recall@k$, the Mean Average Precision (MAP), the Mean Reciprocal Rank (MRR) and the area under the ROC Curve (AUC).

Experimental Results on $\mathcal{D}_{dense}$

We first evaluate the performance of product recommendation on $\mathcal{D}_{dense}$, where $\delta\%$ linked users are used as the training data, and the remaining $(100-\delta)\%$ linked users as the test data. To examine the performance with varying amount of training data, we set δ to 80, 50, 20 and 10, which correspond to the $\frac{\#training}{\#test}$ Split Ratios (SR) of 4:1, 1:1, 1:4 and 1:9 respectively.

The results of different methods for overall product recommendation are presented in Table 5. It can be observed that:

- Apart from the simple baseline *Popularity*, which does not rely on any training data, the performance of all other methods improves with the increasing size of the training data. *Popularity* appears to be a competitive baseline for cold-start recommendation due to the fact that negative products are selected from the same product categories as the positive ones. By incorporating the semantic similarity between users and products, it leads to negligible performance change, which indicates the simple surface similarity cannot well capture the purchase preferences.
- *FMUI* performs better than *MFUA* on the dataset with the split ratios of 1:1 and 4:1, but is worse with the other two ratios. A possible reason is that FMUI involves all the microblogging attributes and thus potentially requires more training data for a better performance. When the training data is limited, FMUI cannot gather sufficient statistics for some microblogging attributes due to data sparsity.
- Our proposed *Cold* variants are consistently better than the baselines. Interestingly, Cold$_{enhanced}$ is not sensitive to the amount of training data, which gives rather stable performance across all the three ratios. By incorporating additional demographic attributes, Cold$_{D+E}$ is consistently better than Cold$_E$, and the improvement seems more significant when the training data is abundant (at the ratio of 1:1). When the training data is limited, Cold$_{++}$ outperforms all the other methods. But with more training data, it performs slightly worse than Cold$_{D+E}$.

TABLE 4
Performance comparisons of different methods on cold-start product recommendation. $*$ indicates that our Cold method is significantly better than the best baseline at the level of 0.01.

SR	Methods	P@10	R@50	MAP	MRR	AUC
4:1	Pop	0.175	0.215	0.120	0.380	0.669
	Pop$_{++}$	0.175	0.215	0.120	0.380	0.669
	ES	0.117	0.195	0.115	0.267	0.653
	MFUA	0.212	0.245	0.136	0.495	0.701
	FMUI	0.226	0.253	0.145	0.502	0.730
	Cold$_E$	0.237	0.265	0.155	0.512	0.751
	Cold$_{D+E}$	**0.243***	**0.270***	**0.159***	**0.527***	**0.771***
	Cold$_{++}$	0.239	0.261	0.157	0.517	0.763
1:1	Pop	0.175	0.215	0.120	0.380	0.669
	Pop$_{++}$	0.175	0.215	0.120	0.380	0.669
	ES	0.117	0.195	0.115	0.267	0.653
	MFUA	0.210	0.240	0.130	0.469	0.681
	FMUI	0.215	0.241	0.125	0.481	0.687
	Cold$_E$	0.222	0.251	0.142	0.484	0.724
	Cold$_{D+E}$	**0.229***	**0.257***	**0.146***	**0.508***	**0.734***
	Cold$_{++}$	0.226	0.255	0.146	0.497	0.730
1:4	Pop	0.175	0.215	0.120	0.380	0.669
	Pop$_{++}$	0.175	0.215	0.120	0.380	0.669
	ES	0.117	0.195	0.115	0.267	0.653
	MFUA	0.202	0.231	0.126	0.449	0.693
	FMUI	0.186	0.225	0.131	0.389	0.670
	Cold$_E$	0.216	0.243	0.137	0.475	0.700
	Cold$_{D+E}$	0.218	0.248	0.137	0.477	0.705
	Cold$_{++}$	**0.220***	**0.249***	**0.140***	**0.484***	**0.715***
1:9	Pop	0.175	0.215	0.120	0.380	0.669
	Pop$_{++}$	0.175	0.215	0.120	0.380	0.669
	ES	0.117	0.195	0.115	0.267	0.653
	MFUA	0.193	0.230	0.118	0.439	0.678
	FMUI	0.172	0.225	0.117	0.411	0.668
	Cold$_E$	0.205	0.234	0.128	0.461	0.683
	Cold$_{D+E}$	0.206	0.238	0.129	0.473	0.685
	Cold$_{++}$	**0.217***	**0.245***	**0.138***	**0.482***	**0.695***

Experimental Results on $\mathcal{D}_{sparse}$

We have examined the performance of product recommendation on frequent buyers above. In real-world

applications, "long-tail" users (i.e., those with few purchases) are prevalent in e-commerce Websites. Therefore, an effective recommender system should also be capable of generating recommendations for these users. We use the users in $\mathcal{D}_{dense}$ as the training data for both user embedding fitting and matrix factorization learning, and consider the users in $\mathcal{D}_{sparse}$ as the test data for product recommendation. Since the users in $\mathcal{D}_{sparse}$ have fewer than five purchases, we only report the performance of $Recall@k$ but not $Precision@k$. We also use MAP, MRR and AUC as evaluation metrics. We can observe from Table 5 that our proposed method Cold$_E$ is consistently better than all the baselines, which indicates that the effectiveness of recommendation for long-tail users.

TABLE 5
Performance comparisons of different methods on cold-start product recommendation on $\mathcal{D}_{sparse}$. $*$ indicates that Cold$_E$ is significantly better than the best baseline at the level of 0.01.

Methods	MAP	MRR	R@10	AUC
Pop	0.175	0.125	0.120	0.684
Pop$_{++}$	0.175	0.175	0.120	0.684
MFUA	0.251	0.337	0.419	0.718
FMUI	0.252	0.337	0.421	0.720
Cold$_E$	**0.275***	**0.363***	**0.458***	**0.757***

Scalability Analysis

We present the scalability analysis for our model $Cold_E$.[9] We first analyze the time complexity for both offline parameter training and online product recommendation. For *offline parameter training*, the cost of training the MART models is $N_{tree} \times \bar{C}_{tree}$, where N_{tree} is the number of trees and $\bar{C}_{tree}$ is the average cost for generating a decision regression tree. Then, the SGD method to train Cold$_E$ has the computational complexity of $\mathcal{O}(nL\bar{F}|\mathcal{D}|)$, where n is the iteration number, L is the number of latent factors, $\bar{F}$ is the average number of non-zero features for a training instance and $|\mathcal{D}|$ is the training data size. In practice, we have found that SGD converges quickly and usually converges in $30-50$ iterations on our training set. For *online product recommendation*, when a new user arrivees, we first generate the fitted user embedding features, at most incurring a cost of $h_{max} \times N_{tree}$, where h_{max} is the maximum tree height. When making recommendation, we use Eq. 6 to score each candidate product. In Eq. 6, a user incurs a cost of $K \times L$ additions and K multiplications to derive $\sum_{k=1}^{K} \hat{v}_{u,k}\mathbf{x}_k$ and a cost of L multiplications and L additions for dot product, while $\mathbf{y}_p + \sum_{k=1}^{K} v_{p,k}\mathbf{y}_k$ for all the products are pre-computed. To generate recommendation, we further need a cost of $N_{list} \times \log N_{list}$ for ranking candidate products for a user, where N_{list} is the length of candidate product list.

While for space complexity, our major cost consists of space for MART models and latent factors. MART models take up a cost of $\mathcal{O}(\bar{N}_{node} \times \bar{C}_{node} \times N_{tree})$, where $\bar{N}_{node}$ and C_{node} denotes the average number of nodes in a MART tree and the average space cost for a single node respectively. We have a cost of $(|\mathcal{U}|+|\mathcal{P}|+K) \times L$ to store latent factors. Compared to traditional matrix factorization, it incurs an additional cost of $K \times L$. In practice, K is usually set to 50$\sim$200. We summarize the time and space cost for $Cold_E$ in Table 6.[10] It can be observed that our method is very efficient in online recommendation. When dealing with extremely large datasets, the training process can be performed in a distributed way by using SGD, and the test process can still be efficient since it only involves the MART tree traversal and latent vector operations.

TABLE 6
Running time and memory costs for our approach on $\mathcal{D}_{dense}$ with the split $\frac{\#train}{\#test}$ ratio of 1:1.

Phases	#users	Time (sec.)	Space (MB)
Training	7,927	563 (MART) 304 ($Cold_E$)	4.67 (MART) 15.72 ($Cold_E$)
Test	7,926	13.8 (MART) 5.1 ($Cold_E$)	4.67 (MART) 15.72 ($Cold_E$)

Parameter Analysis

For our methods, an important component is the recurrent neutral networks, which can be set to two simple architectures, namely *CBOW* and *Skip-gram*. We present the comparison results of our method Cold$_E$ using these two architectures in Table 7. We can see that the performance of using Skip-gram is slightly worse than that of using CBOW[11]

We also examine how the performance varies with different number of embedding dimensions from 50 to 150 with a gap of 25. We observe that the performance is relatively stable with the varying number of embedding dimensions. This is not surprising since the MART models fit each dimension independently. The optimal performance of $Cold_E$ was obtained when the dimension number is 100, which is only slightly better than that of 50. Thus, using 50 embedding dimensions

9. The *Cold* model is implemented in C++ and the MART model is implemented in JAVA. We run the program (single-thread) in the server with Intel(R) Xeon(R) CPU E5-2620 v2 2.10GHz and Ubuntu 14.04 LTS.

10. For each user, we consider a candidate list of m positive product and $50 \times m$ negative products, where m is the actual number of purchases.

11. Here Skip-gram and CBOW correspond to the variants of PV-DBOW and PV-DM for Paragraph Vector respectively. As indicated in [9], PV-DM generally works better than PV-DBOW, which is consistent with our findings. The major reason is that in CBOW (Fig. 2) the target product is emitted conditioned on both the user embedding and the surrounding product embeddings, which naturally captures both sequential purchase context and user preference.

TABLE 7
Performance comparisons of Cold$_E$ using two different architectures.

$\frac{\#training}{\#test}$	Architectures	P@10	MAP	MRR
1:1	CBOW	0.222	0.142	0.484
	Skip	0.220	0.138	0.472
1:4	CBOW	0.216	0.137	0.475
	Skip	0.213	0.133	0.462
1:9	CBOW	0.205	0.128	0.461
	Skip	0.204	0.123	0.458

would be sufficient for our recommendation tasks considering the trade-off between performance and computational complexity. For matrix factorization methods, an important parameter to set is the number of latent factors. We use Cold$_E$ and MFUA as a comparison and vary the number of latent factors from 16 to 80 with a gap of 16. The performance of two methods is relatively stable with different numbers of latent factors, and Cold$_E$ is consistently better than MFUA.

5.4 Revisiting the Effectiveness of the Distributed Representations of Users and Products

In the previous section, we have shown that the learnt product and users embeddings are effective to improve the recommendation performance. In this section, we give more insights into the effectiveness of the distributed representations.

Insights into Product Embeddings

First, we take the learnt product embeddings to conduct a quantitative similarity analysis in order to find out whether the learned product embeddings can discriminate products from different categories or brands. We compute the average similarity score between product pairs from (1) different categories and brands (DCDB); (2) same category but different brands (SCDB); and (3) same category and same brand (SCSB). As it is infeasible to calculate the similarity scores for all possible product pairs in JINGDONG, we sample 10 million product pairs randomly for each type of product pairs for computation. The results are as follows: $\text{sim}_{DCDB} = 0.0217$, $\text{sim}_{SCDB} = 0.2719$ and $\text{sim}_{SCSB} = 0.4406$. The average similarity score of $\text{sim}_{SCDB} > \text{sim}_{DCDB}$ indicates the product embeddings learned are indeed very different for products under different categories; while $\text{sim}_{SCSB} > \text{sim}_{SCDB}$ indicates the product embeddings have a good discriminative power for brands[12].

Insights into User Embeddings

We take the learnt user embeddings to conduct a quantitative similarity analysis in order to find out whether the learned user embeddings can identify users with similar purchase history.

Given a user u, we build two groups of users, denoted by $\mathcal{G}_u^A$ and $\mathcal{G}_u^B$. $\mathcal{G}_u^A$ contains the top K most similar users (a.k.a. K nearest neighbours) of user u, which are identified by the Jacarrd coefficient in terms of purchase history; $\mathcal{G}_u^B$ contains K randomly selected users. We would like to examine whether the user embedding vectors can discriminate a user in $\mathcal{G}_u^A$ from another one in $\mathcal{G}_u^B$.

Given user u together with $\mathcal{G}_u^A$ and $\mathcal{G}_u^B$, we can derive two similarity values $\text{sim}_A^{(u)}$ and $\text{sim}_B^{(u)}$, which are the average similarities with the users in $\mathcal{G}_u^A$ and the users in $\mathcal{G}_u^B$ respectively for user u. We use the cosine function to compute the similarity between two user embedding vectors. K is set to 30 in our experiments. In this way, we can obtain two arrays of similarity values $\{\text{sim}_A^{(u)}\}_{u\in\mathcal{U}}$ and $\{\text{sim}_B^{(u)}\}_{u\in\mathcal{U}}$. By constructing the paired t-test, the results have shown that the values in $\{\text{sim}_A^{(u)}\}_{u\in\mathcal{U}}$ are significantly larger than those in $\{\text{sim}_B^{(u)}\}_{u\in\mathcal{U}}$ at the level of 0.001. The average similarities for $\{\text{sim}_A^{(u)}\}_{u\in\mathcal{U}}$ and $\{\text{sim}_B^{(u)}\}_{u\in\mathcal{U}}$ are 0.090 and 0.031 respectively.

6 Related Work

Our work is mainly related to three lines of research:
Recommender systems. In recent years, the matrix fatortization approach [12], [22] has received much research interests. With the increasing volume of Web data, many studies focus on incorporating auxiliary information [23], [1], [24], [25], [26] into the matrix factorization approach. Two typical frameworks of such studies are the SVDFeature [18] and Factorization Machine [19].

There has also been a large body of research work focusing specifically on the cold-start recommendation problem. Seroussi et al. [7] proposed to make use of the information from users' public profiles and topics extracted from user-generated content into a matrix factorization model for new users' rating prediction. Zhang et al. [27] propose a semi-supervised ensemble learning algorithm. Schein [28] proposed a method by combining content and collaborative data under a single probabilistic framework. Lin et al. [10] addressed the cold-start problem for App recommendation by using the social information from Twitter. Trevisiol et al. Zhou et al. [16] experimented with eliciting new user preferences using decision trees by querying users' responses progressively through an initial interview process. Moshfeghi et al. [29] proposed a method for combining content features such as semantic and emotion information with ratings information for the recommendation task. Liu et al. [30] identified representative users whose linear combinations of tastes are able to approximate other users.

12. All the improvement is signifiant at the confidence level of 0.01.

Cross-domain recommendation. One of the key techniques for cross-domain recommendation is Transfer Learning [31], [32], and the idea is to learn transferable knowledge from the source domain, and further apply it in a target domain. Singh [33] proposed collective matrix factorization to estimate the relations of multiple entities by factorizing several matrices simultaneously while sharing parameters in the latent space. Li [34] attempted to transfer user-item rating patterns from an auxiliary matrix in another domain to the target domain through Codebooks. Hu [35] and Zhao [36] extended transfer learning to triadic factorization and active learning for cross-domain recommendation, respectively.

Social network mining. We follow the early commercial mining studies on social networking websites. Hollerit et al. [37] presented the first work on commercial intent detection in Twitter. Zhao et al. [5] first proposed to route products from e-commerce companies to microblogging users. Our work is also related to studies on automatic user profiling [38] and cross-site linkage inference [39].

Our work is built upon these studies, especially in the areas of cross-domain and cold-start recommendation. Though sharing some similarities, we are dealing with a very specific task of highly practical value, cold-start product recommendation to microblogging users. To the best of our knowledge, it has not been studied on a large data set before. The most relevant studies are from [40], [41] by connecting users across eBay and Facebook. However, they only focus on brand- or category-level purchase preference based on a trained classifier, which cannot be directly applied to our cross-site cold-start product recommendation task. In addition, their features only include gender, age and Facebook likes, as opposed to a wide range of features explored in our approach. Lastly, they do not consider how to transfer heterogeneous information from social media websites into a form that is ready for use on the e-commerce side, which is the key to address the cross-site cold-start recommendation problem.

7 Conclusions

In this paper, we have studied a novel problem, *cross-site cold-start product recommendation*, i.e., recommending products from e-commerce websites to microblogging users without historical purchase records. Our main idea is that on the e-commerce websites, users and products can be represented in the same latent feature space through feature learning with the recurrent neural networks. Using a set of linked users across both e-commerce websites and social networking sites as a bridge, we can learn feature mapping functions using a modified gradient boosting trees method, which maps users' attributes extracted from social networking sites onto feature representations learned from e-commerce websites. The mapped user features can be effectively incorporated into a feature-based matrix factorisation approach for cold-start product recommendation. We have constructed a large dataset from WEIBO and JINGDONG. The results show that our proposed framework is indeed effective in addressing the *cross-site cold-start product recommendation* problem. We believe that our study will have profound impact on both research and industry communities.

Currently, only a simple neutral network architecture has been employed for user and product embeddings learning. In the future, more advanced deep learning models such as Convolutional Neural Networks[13] can be explored for feature learning. We will also consider improving the current feature mapping method through ideas in transferring learning [31].

Acknowledgements

The authors thank the anonymous reviewers for their valuable and constructive comments. The work was partially supported by National Natural Science Foundation of China under the grant number 61502502, the National Key Basic Research Program (973 Program) of China under the grant number 2014CB340403 and the Innovate UK under the grant number 101779.

References

[1] J. Wang and Y. Zhang, "Opportunity model for e-commerce recommendation: Right product; right time," in *SIGIR*, 2013.
[2] M. Giering, "Retail sales prediction and item recommendations using customer demographics at store level," *SIGKDD Explor. Newsl.*, vol. 10, no. 2, Dec. 2008.
[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, Jan. 2003.
[4] V. A. Zeithaml, "The new demographics and market fragmentation," *Journal of Marketing*, vol. 49, pp. 64–75, 1985.
[5] W. X. Zhao, Y. Guo, Y. He, H. Jiang, Y. Wu, and X. Li, "We know what you want to buy: a demographic-based system for product recommendation on microblogs," in *SIGKDD*, 2014.
[6] J. Wang, W. X. Zhao, Y. He, and X. Li, "Leveraging product adopter information from online reviews for product recommendation," in *ICWSM*, 2015.
[7] Y. Seroussi, F. Bohnert, and I. Zukerman, "Personalised rating prediction for new users using latent factor models," in *ACM HH*, 2011.
[8] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.
[9] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014.
[10] J. Lin, K. Sugiyama, M. Kan, and T. Chua, "Addressing cold-start in app recommendation: latent user models constructed from twitter followers," in *SIGIR*, 2013.
[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.
[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[13] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

13. http://deeplearning.net/tutorial/lenet.html

[14] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[15] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, Oct. 2001.

[16] K. Zhou, S. Yang, and H. Zha, "Functional matrix factorizations for cold-start recommendation," in *SIGIR*, 2011.

[17] T. Chen, H. Li, Q. Yang, and Y. Yu, "General functional matrix factorization using gradient boosting," in *ICML*, 2013.

[18] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, "SVDFeature: A toolkit for feature-based collaborative filtering," *Journal of Machine Learning Research*, vol. 13, 2012.

[19] S. Rendle, "Factorization machines with libfm," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 3, May 2012.

[20] ——, "Social network and click-through prediction with factorization machines," in *KDDCup*, 2012.

[21] B. Xiao and I. Benbasat, "E-commerce product recommendation agents: Use, characteristics, and impact." *MIS Quarterly*, vol. 31, pp. 137–209, 2007.

[22] Y. Shi, X. Zhao, J. Wang, M. Larson, and A. Hanjalic, "Adaptive diversification of recommendation results via latent factor portfolio," in *SIGIR*, 2012.

[23] L. Hong, A. S. Doumith, and B. D. Davison, "Co-factorization machines: Modeling user interests and predicting individual decisions in twitter," in *WSDM*, 2013.

[24] J. Tang, H. Gao, H. Liu, and A. Das Sarma, "etrust: Understanding trust evolution in an online world," in *SIGKDD*, 2012.

[25] H. Ma, T. C. Zhou, M. R. Lyu, and I. King, "Improving recommender systems by incorporating social contextual information," *ACM Trans. Inf. Syst.*, vol. 29, no. 2, 2011.

[26] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma, "Explicit factor models for explainable recommendation based on phrase-level sentiment analysis," in *SIGIR*, 2014.

[27] M. Zhang, J. Tang, X. Zhang, and X. Xue, "Addressing cold start in recommender systems: a semi-supervised co-training algorithm," in *SIGIR*, 2014.

[28] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *SIGIR*, 2002.

[29] Y. Moshfeghi, B. Piwowarski, and J. M. Jose, "Handling data sparsity in collaborative filtering using emotion and semantic based features," in *SIGIR*, 2011.

[30] N. N. Liu, X. Meng, C. Liu, and Q. Yang, "Wisdom of the better few: cold start recommendation via representative based rating elicitation," in *ACM RecSys*, 2011.

[31] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE TKDE*, vol. 22, no. 10, pp. 1345–1359, 2010.

[32] W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction," in *AAAI*, 2010.

[33] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *SIGKDD*, 2008.

[34] B. Li, Q. Yang, and X. Xue, "Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction," in *IJCAI*, 2009.

[35] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu, "Personalized recommendation via cross-domain triadic factorization," in *WWW*, 2013.

[36] L. Zhao, S. J. Pan, E. W. Xiang, E. Zhong, Z. Lu, and Q. Yang, "Active transfer learning for cross-system recommendation," in *AAAI*, 2013.

[37] B. Hollerit, M. Kröll, and M. Strohmaier, "Towards linking buyers and sellers: Detecting commercial intent on twitter," in *WWW Companion*, 2013.

[38] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: Inferring user profiles in online social networks," in *WSDM*, 2010.

[39] R. Zafarani and H. Liu, "Connecting corresponding identities across communities," in *ICWSM*, 2009.

[40] Y. Zhang and M. Pennacchiotti, "Recommending branded products from social media," in *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, 2013, pp. 77–84.

[41] ——, "Predicting purchase behaviors from social media," in *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, 2013, pp. 1521–1532.

Wayne Xin Zhao is currently an assistant professor at the School of Information, Renmin University of China. He received the Ph.D. degree from Peking University in 2014. His research interests are web text mining and natural language processing. He has published several referred papers in international conferences journals such as ACL, EMNLP, COLING, ECIR, CIKM, SIGIR, SIGKDD, AAAI, ACM TOIS, ACM TKDD, ACM TIST, IEEE TKDE, KAIS and WWWJ.

PLACE PHOTO HERE

Sui Li is currently a PhD student at the School of Electronic Engineering and Computer Science, Peking University, China. He received his BEng degree in Computer Science from Peking University in 2014, China. His research mainly focuses on Web mining and machine learning.

Yulan He is a Reader at the School of Engineering and Applied Science, Aston University, UK. She received her PhD degree from the University of Cambridge in 2004. She has published extensively in natural language processing, text and data mining, sentiment analysis, and social media analysis. She has served as an Area Chair in EMNLP 2015, NAACL 2016 and co-organised ECIR 2010.

PLACE PHOTO HERE

Edward Y. Chang is the Vice President of Research and Innovation at HTC since July 2012, heading software and hardware future technology research and development. Prior to his HTC post, Ed was a director of Google Research and was a full professor of Electrical Engineering at the University of California, Santa Barbara (UCSB). Ed has served on ACM (SIGMOD, KDD, MM, CIKM), VLDB, IEEE, WWW, and SIAM conference program committees, and co-chaired several conferences including MMM, ACM MM, ICDE, and WWW. He is a recipient of the NSF Career Award, IBM Faculty Partnership Award, and Google Innovation Award.

Ji-Rong Wen is a professor at the School of Information, Renmin University of China. Before that, he was a senior researcher and group manager of the Web Search and Mining Group at MSRA since 2008. He has published extensively on prestigious international conferences/journals and served as program committee members or chairs in many international conferences. He was the chair of the "WWW in China" track of the 17th World Wide Web conference. He is currently the associate editor of ACM Transactions on Information Systems (TOIS).

Xiaoming Li is a professor at the School of Electronic Engineering and Computer Science and the director of Institute of Network Computing and Information Systems in Peking University, China. He is a Senior member of IEEE and currently served as Vice president of China Computer Federation. His research interests include search engine and web mining, and web technology enabled social sciences.