Exploring Demographic Information in Social Media for Product Recommendation

Wayne Xin Zhao, Sui Li, Yulan He, Liwei Wang, Ji-Rong Wen and Xiaoming Li

Abstract—In many e-commerce websites, product recommendation is essential to improve user experience and boost sales. Most existing product recommender systems rely on historical transaction records or website browsing history of consumers in order to accurately predict online users' preferences for product recommendation. As such, they are constrained by limited information available on specific e-commerce websites. With the prolific use of social media platforms, it now becomes possible to extract product demographics from online product reviews and social networks built from microblogs. Moreover, users' public profiles available on social media often reveal their demographic attributes such as age, gender, education, etc. In this paper, we propose to leverage the demographic information of both products and users extracted from social media for product recommendation. In specific, we frame recommendation as a learning-to-rank problem which takes as input the features derived from both product and user demographics. An ensemble method based on the gradient boosting regression trees is extended to make it suitable for our recommendation task. We have conducted extensive experiments to obtain both quantitative and qualitative evaluation results. Moreover, we have also conducted a user study to gauge the performance of our proposed recommender system in a real-world deployment. All the results show that our system is more effective in generating recommendation results better matching users' preferences than the competitive baselines.

Index Terms—e-commerce, product recommendation, product demographic, social media

1 Introduction

In recent years, e-commerce websites such as Amazon and eBay transcend geophysical barriers and make it possible for individuals or business to form transactions anywhere and anytime. With the rapid growth of e-commerce websites, *online product recommendation* has attracted much attention in the research community since product recommender systems have been shown effective in influencing consumers' purchase decisions and can potentially increase sales [1], [2], [3], [4], [5]. Existing product recommendation systems typically rely on consumers' historical transaction records in order to make relevant recommendations. As such, they are often built for specific e-commerce websites and are thus constrained by the information available there.

In this paper, rather than relying on limited information available on any specific e-commerce website, we aim to develop a generic online product recommender system by exploring a vast amount of information available externally such as that on social media platforms. It has been previously shown that purchase intent is a more common motive for mentioning a brand on Twitter[1]. As such, we propose to develop a product recommender system based on the social media data since they contain abundant information about users' purchase intents which are constantly updated in real time [6]. Mining social media data enables the capture of users' immediate purchase intents outside e-commerce websites. Furthermore, social networking platforms often contain users' public profiles, such as age, sex and/or professions, from which users' demographic characteristics can be extracted, This is particularly important to address the *cold start* problem commonly faced in recommender systems when litter information about a user is available.

Our recommender system is built based on the largest Chinese microblogging service Sina Weibo[2], in which users' purchase intents can be instantaneously discovered from their tweets and their demographic characteristics can be extracted from users' public profiles. A fast classification method is proposed to identify tweets expressing purchase intents in nearly real-time. We have previously shown that product recommendation can be casted as a learning to rank problem and Multiple Additive Regression Trees (MART) yields better performance than listwise, pairwise or other pointwise algorithms [7]. In this paper, we propose to improve over the original MART model in the following ways. Firstly, in order to cater

W.X. Zhao (corresponding author) and J.-R. Wen are with the School of Information, Renmin University of China, Beijing, China; W.X. Zhao and J.-R. Wen are also with Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China. E-mail: {batmanfly, jirong.wen}@gmail.com.

Y. He is with the School of Engineering and Applied Science, Aston University, Birmingham, United Kingdom. E-mail: y.he9@aston.ac.uk.

S. Li and X.M. Li are with the School of Electronic Engineering and Computer Science, Peking University, Beijing, China. E-mail: suili@pku.edu.cn, lxm@pku.edu.cn.

L.W. Wang is with the School of Electronics Engineering and Computer Sciences, Peking University, Beijing, China. E-mail: wanglw@cis.pku.edu.cn.

[1] http://www.brandwatch.com/wp-content/uploads/2013/02/Twitter-Landscape-2013-Extended-Version.pdf

[2] http://weibo.com

for relevance levels of different training instances, we propose weighted loss functions for learning the MART models. Secondly, we propose an attribute-based feature sampling method to be used in the process of tree node splitting in MART. Finally, as MART, being a boosting algorithm, suffers from high variance and is thus at risk of overfitting to noisy or unrepresentative training data, we develop a bagging method to wrap around MART, denoted as B-MART, which can effectively reduce the variance and meanwhile improve the recommendation accuracy. We also propose a distributed implementation of B-MART, which has been shown to be computationally more efficient in terms of both time and memory space compared to that of an undistributed implementation of the B-MART model.

To evaluate our proposed product recommender system, we collect training data through semi-automatic acquisition and conduct extensive experiments to obtain both quantitative and qualitative evaluation results. Moreover, we also conduct a user study to gauge the performance of our proposed system in a real-world deployment. All the results show that our system is more effective in generating better recommendation results than the baselines.

The remainder of the paper is organized as follows. An overview of our developed recommender system is given in Section 2. Detection of purchase intents from tweets and learning product demographics are described in Section 3 and 4 respectively. Extension of the MART model for product recommendation is presented in Section 5. Section 6 discusses the experimental results. Related work is presented in Section 7. Finally, Section 8 concludes the paper and outlines future directions.

2 Overview of the System

Our recommender system will first identify a user's purchase intents from her tweets in real-time and recommend a list of products based on a similarity measurement between the user's demographic information extracted from her online public profile and products' demographic information learned from online product reviews and following/mention networks built from SINA WEIBO.

2.1 Data collection

For the development of our system, we choose a microblogging platform, SINA WEIBO, which is the largest Chinese microblogging site. We have crawled all the information, including tweets, following relations and public profiles, for a total of 5 million active users on WEIBO via an authenticated API. A total of 1.7 billion tweets have been retrieved for these 5 million users between January 2013 and June 2013. Products used for recommendation are selected from JINGDONG[3], the largest B2C e-commerce website. Since our recommender system does not rely on any product transaction records, the system can essentially select products from any e-commerce websites for recommendation. To evaluate the performance of our system, we choose three popular product types in JINGDONG: laptop, camera and phone. In total, these three product categories contain 3,155 products and 1.13 million user reviews. To deal with such big data, we build our back-end system based on distributed indexing which allows the retrieval of information of users or products in a quick way.

2.2 System Architecture

Our recommender system consists of three major components:

- *Purchase intent detection*. This component aims to detect users' purchase intents in near real-time. In order to reduce noise, irrelevant tweets are first filtered using a manually constructed keyword list. Then a classifier trained from both textual features of tweets and users' demographic information is employed to identify tweets containing purchase intents.
- *Demographic information extraction*. Demographic information is extracted for both users and products. Users' demographic information is extracted from their online public profiles; while product demographic information is learned from online product reviews on e-commerce websites and the following/mention relations extracted on WEIBO. Both user and product demographic information is mapped into the same demographic attribute feature space for easy comparison.
- *Product recommendation*. This is the core component of the system which returns a list of recommended products to a user. We propose a novel demographic-based recommendation algorithm where similarity measurement is performed between a user and products based on features derived from their demographic information which are subsequently combined in a learning to rank framework for making product recommendation with high accuracy.

A demo system[4] has been implemented which simulates the WEIBO stream by processing our canned data collected between January and June of 2013. For ease of browsing for non-Chinese speakers, we have also produced a web page containing several interesting recommendation examples which have been translated into English[5].

[3] http://jd.com
[4] http://sewm.pku.edu.cn/metis
[5] http://162.105.205.253:8667/metisrecommendation/special_en/

2.3 Preliminary Concepts

We present some preliminary concepts before delving into the details of our system.

Purchase-intent tweet. A tweet is defined as a *purchase-intent tweet* if it explicitly expresses a desire or interest of buying a certain product [6]. Here we only consider explicit expressions of buying desires and ignore implicit purchase intents since the latter is more difficult to detect. Note that the author of a tweet may not be the one who wants to buy a product. In an example tweet below:

> Please recommend! *My son* wants to buy a *Samsung phone* less than $200.

The potential customer of *Samsung phone* is *My son*, not the author of the tweet.

Product demographics. The product demographics[6] [8], sometimes called the target audience, of a product or service is a collection of the characteristics of the people who buy that product or service. A demographic profile (often shortened as "a demographic") provides enough information about typical users of this product to create a mental picture of this hypothetical aggregate. For example, a product demographic might refer to users of single, female, middle-class, age 18 to 24, and college educated.

Knowing information such as the income status, age and tastes of users can help companies sell more, branch out to other groups and create more products that appeal to target buyers. Instead of identifying these useful demographic attributes manually, we start with the attributes in users' public profile on WEIBO. With reference to the marketing studies [9], [10], we have identified 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 [7]. We summarize the details of the demographic attributes in Table 1.

We use probabilities to characterize the demographics of a product. Formally, let $\mathcal{A}$ denote the set of attributes and $\mathcal{V}_a$ denote the set of values for an attribute a. For a product e, its demographic distribution of an attribute a is $\theta^{(e,a)} = \{\theta_v^{(e,a)}\}_{v \in \mathcal{V}_a}$, where $\theta_v^{(e,a)}$ is the proportion of the target consumers with the value v for attribute a. Since $\theta^{(e,a)}$ is a distribution, we have $\sum_{v \in \mathcal{V}_a} \theta_v^{(e,a)} = 1$. Furthermore, we use the set of attribute distributions to represent the product demographics, i.e. $\{\theta^{(e,a)}\}_{a \in \mathcal{A}}$.

User profile. *User profile*, which is also called *user demographics*, is similar to *product demographics*. Formally, given a user u and the considered attribute set $\mathcal{A}$, we use $\mathcal{D}_u = \{v_a^{(u)}\}_{a \in \mathcal{A}}$ to denote the user's profile, where $v_a^{(u)}$ is the value of attribute a. To align with the demographic representation of a product, we use a binary vector to represent a user's demographic profile. For a user u with attribute a, we have $\phi^{(u,a)} = \{\phi_v^{(u,a)}\}_{v \in \mathcal{V}_a}$, where $\phi_v^{(u,a)}$ is set to 1 only when $v = v_a^{(u)}$ and 0 otherwise. On SINA WEIBO, a user is required to fill in some attributes in her public profile during registration, which can be modified later. If a user has not filled in an attribute, all its related entries are set to zero[8]. Different from the value-based representation in [7], such a demographic vector representation has the merit that it can take the results of user profile inference algorithms [12], [13], [14] as an input, where each vector element is the confidence score over the respective profile value returned by the algorithms.

TABLE 1
List of demographic attributes.

Attribute	Values
Gender	male, female
Age	1-11, 12-17, 18-30, 31-45, 46-59, 60+
Marital Status	single, engaged, loving secretly, married, relationship seeking, bereft of one's spouse, separated, divorced, ambiguous, loving
Education	literature, natural science, engineering, social sciences, medical science, art, others
Career	internet technology, designing, media, service industry, manufacturing, medicine, scientific research, management, others
Interests (Weibo tags)	travel, photographing, music and movie, computer games, Internet surfing, other

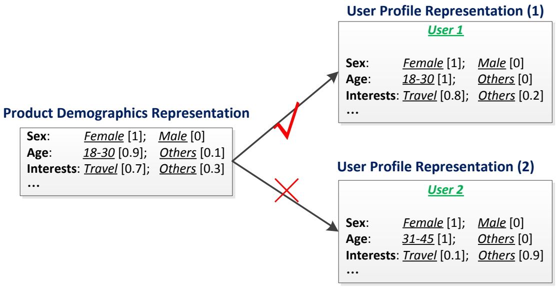
User Profile Representation (1)
User 1
Sex: Female [1]; Male [0]
Age: 18-30 [1]; Others [0]
Interests: Travel [0.8]; Others [0.2]
...
Product Demographics Representation
Sex: Female [1]; Male [0]
Age: 18-30 [0.9]; Others [0.1]
Interests: Travel [0.7]; Others [0.3]
...
User Profile Representation (2)
User 2
Sex: Female [1]; Male [0]
Age: 31-45 [1]; Others [0]
Interests: Travel [0.1]; Others [0.9]
...

Fig. 1. An illustrative example for the representations of product demographics and user profiles. For simplicity, we only show the studied value dimensions together with the corresponding probabilities. We use *Others* to combine the probabilities from the rest value dimensions.

For better understanding the representations for product demographics and user profile, we present an illustrative example in Fig. 1. In this figure, we have made the following assumptions: the product

[6] http://www.ehow.com/info_10015346_product-demographic.html

[7] Given an attribute, we collect all the unique values filled in by users in our data collection, and only keep the values with high population. We further manually group similar values. Furthermore, we discretized attribute values based on the customer segmentation [11](chapter five) in marketing and ensured balanced distribution probabilities over different values across different discretization intervals.

[8] This will make $\phi^{(u,a)}$ no longer a valid probability distribution. But as will be shown later, it does not affect the construction of demographic feature vectors.

is designed for young women who are interested in travel; user 1 is with the age of 25 while interested in travel; and user 2 is with the age of 35 while not interested in travel. Intuitively, the product is more suitable for user 1 than user 2. Our representation method provides a unified way to represent users and products, and can generate an intuitive recommendation by simply matching a product and a candidate user based on demographic-based similarities, [9]. It is more straightforward to obtain user profile representation by extracting the public profile on online social networks, while it is relatively challenging to derive a robust estimation for product demographics probabilities. We will discuss it in details in Section 4.

3 Purchase Intent Detection

The task of purchase intent detection on microblogs can be considered as a classification problem where a tweet is classified into one of the two categories (containing or not containing purchase intents). Due to the huge volume of tweets generated each day on Sina Weibo, we first perform tweet filtering by only keeping tweets that are likely to contain purchase intents. A list of *purchase indicator keywords* are compiled by an annotator major in commerce service. The Weibo stream is processed in parallel and a fast string search algorithm is used to keep tweets that contain at least one purchase indicator keyword for subsequent processing. As a preprocessing step, we need to consider the usage of negation in the tweets, which is an obstacle to accurately identify purchase intents. Specific negation identification is time-consuming and requires considerable training data [15]. For consideration of efficiency, we generate several negation purchase patterns by using the list of *purchase indicator keywords* and a small lexicon of Chinese negation words. We filter tweets with negation purchase patterns, e.g., "I would not like to buy a phone."

We then train an intent classifier using a combination of highly discriminative n-gram features and Part-of-Speech (POS) tags. For example, for a phrase "buy VB cheap JJ", we generate a feature "buy-VB cheap-JJ" and call it *lexical-POS* feature. In addition, we also incorporated users' demographic features as additional context to supplement the textual evidence. This can be explained by the following example. Given two tweets respectively from a female and a boy, both tweets have mentioned "transformer model". Intuitively, the boy's tweet is more likely to contain a purchase intent. As such, we also extract users' demographic features containing the six attributes presented in Table 1 for intent classifier training.

We have conducted evaluation on the purchase intent classification results on our manually annotated 10,000 tweets and obtained 81.8% in F-measure using a combination of lexical-POS and demographic features trained on Support Vector Machines (SVMs) with the RBF kernel.

4 Product Demographics Extraction

Users' demographic information can be obtained directly from their public profiles in Sina Weibo. In this section, we focus on extracting product demographic information, also called *target audience* or *target consumer*, from social media. Previously, product demographics are derived from either user surveys [16] or commercial purchase records [3]. We focus on the six demographic attributes shown in Table 1 and propose to leverage demographic related knowledge from online product reviews and the following/mention information on microblogs.

4.1 Demographics Extraction from Online Product Reviews

In *online product reviews*, users might explicitly mention demographic related information in addition to their opinions. For example, in a sentence "I *bought my son* this phone" from a product review, the phrase "buy my son" indicates that the current product is suitable for the author's son, who is a potential target consumer. It can be observed that "`buy somebody something`" is an important pattern for extracting the demographic knowledge which is embedded in a phrase *my son*, called a *demographic phrase*.

We propose a bootstrapping approach to iteratively learn the patterns and extract demographic phrases in Algorithm 1. The approach starts with some seed patterns such as "`buy somebody something`" and "`a gift to somebody`". In each iteration, we apply existing patterns to extract new demographic phrases with the function $\mathsf{ExtractDemographicPhrases}(\cdot,\cdot)$, and then learn new patterns with the extracted phrases with the functions $\mathsf{GeneratePatterns}(\cdot,\cdot)$ and $\mathsf{ExtractTopFrequentPatterns}(\cdot)$. To generate patterns with $\mathsf{GeneratePatterns}(\cdot,\cdot)$, for each demographic phrase, we first extract the proceeding n_1 tokens and the following n_2 tokens, and then combine the $(n_1 + n_2)$ tokens as the candidate patterns. We have found that many single-token patterns are noises, and patterns with more than two tokens yield little improvement when applied on short review text as in our experiments. As such, we only consider two-token patterns and require $(n_1 + n_2)$ to be 2, where $0 \leq n_1 \leq 2$ and $0 \leq n_2 \leq 2$. The extraction algorithm improved upon our previously proposed method [7] by adding a pattern filtering step (Lines $16 - 18$). The demographic phrases identified by a good pattern should not deviate from the previously identified demographic phrases

[9] For example, we can sum the corresponding demographic-based probabilities for each attribute: user 1 will be assigned to a value of *2.52* by having $1 \times 1 + 0.9 \times 1 + 0.7 \times 0.8 + 0.3 \times 0.2$, while similarly user 2 will be assigned to a value of *1.44*.

too much. Here we use the Jaccard coefficient to measure the similarity between extracted phrases and empirically set the threshold δ to 0.3. With the additional filtering step, the demographic phrase extraction accuracy has been significantly improved from 66% to 88.5% when evaluating on the top 400 most frequent candidate phrases, i.e., a total of 363 demographic phrases are judged to be related to some demographic characteristics. We use these 363 demographic phrases to retrieve in the collection of 1.13 million JINGDONG reviews, and have found that about 10% reviews have contained at least one demographic phrase.

Algorithm 1: Bootstrapping algorithm for extracting demographic phrases from online reviews.

```
Input: review sentence corpus $\mathcal{R}$, seed extraction patterns $\mathcal{P}^{(seed)}$
Output: an set of identified patterns $\mathcal{P}$ and a set of identified demographic phrases $\mathcal{R}$;
$\mathcal{P} \leftarrow \mathcal{P}^{(seed)}$, $\mathcal{P}' \leftarrow \mathcal{P}^{(seed)}$;
$\mathcal{R}' \leftarrow \emptyset$, $\mathcal{R} \leftarrow \emptyset$;
repeat
    $\mathcal{R}' \leftarrow \emptyset$;
    for each pattern $p \in \mathcal{P}'$ do
        $\mathcal{R}_p \leftarrow \emptyset$;
        for each sentence $s \in \mathcal{R}$ do
            if p exists in s then
                $\mathcal{R}_p \leftarrow \mathcal{R}_p \cup$ ExtractDemographicPhrases ($p$,$s$);
            end
        end
        if Jaccard($\mathcal{R}_p$, $\mathcal{R}$) $\leq \delta$ and $p \notin \mathcal{P}^{(seed)}$ then
            $\mathcal{R}_p \leftarrow \emptyset$;
            Remove $p$ from $\mathcal{P}'$;
        end
        $\mathcal{R}' \leftarrow \mathcal{R}_p \cup \mathcal{R}'$;
    end
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}'$, $\mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}'$;
    $\mathcal{R}' \leftarrow \emptyset$, $\mathcal{P}' \leftarrow \emptyset$;
    for each sentence $s \in \mathcal{R}$ do
        for each demographic phrase $m \in \mathcal{R}'$ do
            $\mathcal{P}' \leftarrow \mathcal{P}' \cup$ GeneratePatterns($s, m$);
        end
    end
    $\mathcal{P}' \leftarrow$ ExtractTopFrequentPatterns($\mathcal{P}'$);
until No new pattern is identified;
return An set of identified extraction patterns $\mathcal{P}$ and a set of identified demographic phrases $\mathcal{R}$;
```

Given a product e, we obtain a set of its related demographic phrases $\mathcal{R}_e$ which have at least 10 occurrences in our data. We then map these phrases into six demographic dimensions as have been previously described in Table 1 using a list of pre-defined rules. For example, given a phrase "my little son", we can map it into two dimensions "male"(sex) and "12∼17"(age). Currently, for most of the demographic phrases, we only map them to three dimensions, namely *sex*, *age* and *marital status*. As future work, we will consider how to automatically learn more attribute-level correspondences for demographic phrases. In this example, the target consumer is "the author of the tweet" (parents) while the real product adopter is "the son of the author". Aiming to learn the match correspondence between products and users, we focus on the real adopter in this paper instead of the the consumer. We maintain a counter $\#(a, v)$ which counts the number of times phrases being mapped to value v of the attribute a. Finally, we use Laplace smoothing (a.k.a add-one smoothing) to estimate the demographic distribution of a product e for an attribute a as follows

$$\theta_v^{(e,a)} = \frac{\#(a, v) + 1}{\sum_{v' \in \mathcal{V}_a} \#(a, v') + |\mathcal{V}_a|}, \tag{1}$$

where $\mathcal{V}_a$ is the set of all possible values of attribute a. $\theta_v^{(e,a)}$ is a valid probability since $\sum_{v \in \mathcal{V}_a} \theta_v^{(e,a)} = 1$. Thus, for each attribute of a product, we can model the aggregated statistics as a distribution over a set of all possible attribute values, and a higher probability value indicates a more prominent characteristic of the product for that attribute. Equation 1 can be considered as a special case of the Dirichlet smoothing [17] where an non-informative prior is employed. In practice, we can apply Dirichlet smoothing method to produce more meaningful estimations with the domain knowledge as the prior, which can be from product designers or domain experts.

4.2 Demographics Extraction from Microblogs

It is also possible to extract product demographic information from *microblogs*. The rationale behind is that users might express positive opinions on a certain product. Such users can be treated as potential target audience for the product. As a result, the profiles of these users can be aggregated as the product demographics. We detect users' endorsements on a certain product in microblogs by considering two most common user behaviors below:

- *following*. A brand or a product usually has its official account on microblogs. We take the followers of these official accounts as the potential target audience.
- *mentioning*. Given a product, we retrieve tweets containing the product name through simple keyword matching. We then identify the polarity $\{positive, negative\}$ of each tweet using a machine learning approach [18]. Only tweets with *positive* polarity are treated as the supporting evidence, and their authors are considered as the product's target audience. Here, we only consider two cases (1) a model name is explicitly mentioned, e.g., iPhone 6; (2) a product type is explicitly stated, e.g., "Apple watches".

For the target uses selected from the aforementioned *following* or *mentioning* categories, we further perform spam user filtering by considering their followings/followers, tweet content and interactions with other users.[10] It might be the case that in both

[10] We distinguish normal users from spam users using the following three conditions: (1) An normal user should have a balanced number of tweets and retweets; (2) A normal user should not include any keywords relating to products or brands in her the nickname or profile description. (3) A normal user should not publish many tweets containing keywords relating products or brands.

following and *mentioning* categories, users may not be directly related to a specific product (e.g., "iPhone 5S"), instead they may be related to a brand (e.g., "iPhone") or a company (e.g., "Apple"). We use $\mathcal{U}_e$ to denote target users of a product e, and $\mathcal{U}_{b_e}$ to denote the target audience of a brand or company of a product e. The demographic distribution of a product e for attribute a taking the value v is calculated by combing the estimation computed from $\mathcal{U}_e$ and $\mathcal{U}_{b_e}$ through the Jelinek-Mercer (JM) smoothing method:

$$\theta_v^{(e,a)} = (1-\lambda)\times\frac{\sum_{u\in\mathcal{U}_e}\mathbf{1}[u.a=v]}{\sum_{v'}\sum_{u\in\mathcal{U}_e}\mathbf{1}[u.a=v']}+\lambda\times\frac{\sum_{u\in\mathcal{U}_{b_e}}\mathbf{1}[u.a=v]}{\sum_{v'}\sum_{u\in\mathcal{U}_{b_e}}\mathbf{1}[u.a=v']}, \tag{2}$$

where $\mathbf{1}[\cdot]$ is the indicator function which returns 1 only when the condition is true and λ is the interpolation coefficient, which is empirically set to be 0.3 in our experiments. We use Laplace smoothing in Eq. 1 to avoid zero values, while use Jelinek-Mercer (JM) smoothing to linearly combine the product-level and brand-level estimations.

5 A Ranking based Approach to Product Recommendation

In the previous sections, we have presented how to identify purchase-intent tweets and learn product demographics from social media. In this section, we present a recommendation approach based on the learning-to-rank framework.

5.1 The Learning to Rank Framework for Product Recommendation

Learning to rank was originally proposed for information retrieval [19]. In training, a number of queries and their corresponding retrieved documents with relevance levels (w.r.t the queries) are given, and the objective of learning is to construct a ranking function (model) which achieves the best results in ranking of the training data by minimizing a loss function. In retrieval (testing), given a query, the system returns a ranked list of documents in a descending order of the relevance scores which are calculated using the ranking function.

To formulate our product recommendation problem as a ranking task, we first make a connection between product recommendation and information retrieval. In our task, a purchase-intent tweet can be considered as a query and an adopted product can be understood as a relevant document. For convenience, we use the term "*query*" to denote a purchase-intent tweet. We only consider brand or type based purchase intents instead of model-specific purchase intents. For example, a tweet "I would like to buy my son a Samsung Galaxy II" contains model-specific purchase intents, in which the target product has been specified. We would not consider such cases in our task.

We assume that there are a set of purchase-intent tweets (i.e. queries) $\mathcal{Q} = \{q^{(1)}, q^{(2)}, ..., q^{(m)}\}$ during training. A purchase-intent tweet (query) $q^{(i)}$ is associated with a set of $n^{(i)}$ candidate products $\{p_1^{(i)}, ..., p_{n^{(i)}}^{(i)}\}$. For each candidate product, let $y_j^{(i)}$ denote the judgment on product $p_j^{(i)}$ with respect to query $q^{(i)}$. The value of $y_j^{(i)}$ can be either discrete or continuous, and a higher value indicates a better recommendation for the query $q^{(i)}$. We mainly consider three relevance levels for y: relevant, partially-relevant and non-relevant, which correspond to the score values of 2, 1 and 0 respectively. To derive the y values for training, we can collect a few query-decision pairs, where a decision refers to the adopted product. For each query, there is only one relevant product which matches a user's final purchase decision. If a product model does not match the final product purchased but has the same brand, it is treated as partially-relevant. For all the other cases, products are treated as non-relevant. A feature vector $\boldsymbol{x}_j^{(i)}$ can be constructed for each candidate query-product pair $(q^{(i)}, p_j^{(i)})$. The aim of the learning task is to derive a ranking function f such that, for each feature vector $\boldsymbol{x}_j^{(i)}$ (corresponding to $q^{(i)}$ and $p_j^{(i)}$), it outputs a recommendation score $f(\boldsymbol{x}_j^{(i)})$ for product ranking[11].

To apply the learning to rank algorithms, there are two important issues to consider: *candidate product generation* and *construction of feature vectors*.

5.2 Candidate Product Generation

Recall that we have identified the purchase-intent tweets in *Purchase Intent Detection*. The authors of these tweets are treated as potential consumers. In this section, we study how to generate candidate products based on users' profiles and their purchase-intent tweets.

Requirement identification A user's requirement of a certain product can be derived from the identified purchase-intent tweets. We consider an example here:

I want to buy a *Samsung phone* for <u>less than $200</u>.

In this example, we need to detect the user's requirement of *price* $<$ \$200 for a brand, *Samsung phone*. To build an attribute-based filter for products, we have collected all the product related information from JingDong, such as brand, price, and screen size. Regular expressions have been compiled for each information field and users' requirements are identified by using the regular expression patterns. We generate a list of candidate products which fulfil the identified requirements.

Dealing with explicit mentions of target consumers By default, the author of a purchase-intent tweet

[11] To be more specific, the values of y are needed to be given in training, while in test we obtain the values of y by using the predicted output from the learnt ranking function f, and an item with a larger value for y will be ranked in a higher position, i.e., of more importance for recommendation.

will be the potential customer of its related product. However, we have noticed that there are a considerable number of purchase-intent tweets, in which the potential consumer is not the author of the tweets. See the following example:

I want to buy my son a *Samsung phone.*

The author of this tweet is a female. But the target consumer is *"my son"*. The recommender system needs to be able to identify explicit mentions of target consumers and makes recommendations accordingly. We have previously described an automated approach for the extraction of demographic phrases for product demographic learning (Section 4.1). This approach can also be used to identify the explicit mention of target consumers.

Candidate product list pruning Although we only select the products which fulfil the identified requirements, the number of candidate products is still large which is not suitable for the learning to rank algorithms. For this reason, we select at most 50 best-sale products among the candidate products as the input to the learning to rank algorithms in the next step.

5.3 Constructions of Feature Vectors

A crucial step to the learning to rank algorithms is the construction of a feature vector for each training instance. We mainly consider two types of features:

Query-independent product features. This type of features are independent of any specific query (purchase-intent tweet). Intuitively, product sales follows "the rich get richer" phenomenon, i.e., a user is more likely to buy a product with more successful sales history and more positive comments. Based on this intuition, we use the following features: the sales history of a product (*sale*) and the overall rating score of a product (*rating*). In our dataset, we have found that popular products usually have similar high rating scores. Thus, we also derive the polarity scores based on the opinionated user reviews. Here we follow the unsupervised method in [20] with an open Chinese opinion lexicon to transform review content into polarity scores in the interval of $[-1, 1]$. The third feature used is the overall polarity score of a product (*polarity*). For new products, their feature values are assigned with the aggregated feature values at the brand level.

Query-dependent product features. We quantify the match degree between a user and a product based on their demographic attribute values (see Table 1) and use it as a query-dependent feature. Formally, given a user u and a candidate product e, we derive an attribute vector for attribute a with each value set to the product between the probabilities in the demographic distribution of e and u:

$$x_{a,v}^{u,e} = \theta_v^{(e,a)} \times \phi_v^{(u,a)}, \tag{3}$$

where $\theta^{(e,a)}$ and $\phi^{(u,a)}$ are the demographic distributions of products and users defined in Section 2.3. Then we concatenate all the attribute vectors to form the final feature vector. For distributions $\phi_v^{(e,a)}$, there would be exactly one nonzero entry in the attribute vector $\{x_{a,v}^{u,e}\}_{v \in \mathcal{V}_a}$, which indicates the match degree between a product and a user. Intuitively, if a user closely matches a product in terms of their demographic profiles, the user will be a representative of the target consumers of the product. Recall that we have learnt the demographic distributions from both online reviews and microblogs. Thus, for each attribute, we can obtain two different attribute vectors, and concatenate them into a single vector.

6 The Gradient Boosting Trees for Product Recommendation

Section 5 has presented a recommendation approach based on the learning-to-rank framework. In this section, we focus on the extension of the gradient-boosting trees, a commonly used ranking and regression algorithm.

There are in general three categories of approaches to learn the ranking function f: pointwise, pairwise and listwise [19]. Compared to the pointwise approaches, the pairwise and listwise approaches usually require much more data to learn a robust ranking function. As such, pointwise approaches might be more suitable when the training data are limited, especially when facing with the cold start problem. One of the pointwise approaches, Multiple Additive Regression Tree (MART), has been shown to be effective in many applications [21], [22]. Therefore, in this paper, we focus on extending MART for product recommendation.

6.1 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) [23] and the resulting gradient boosting algorithm is called Multiple Additive Regression Tree [23], [24], [25], i.e. MART. An input feature vector $\mathbf{x} \in \mathbf{R}^d$ is mapped to a score $F(\mathbf{x}) \in \mathbf{R}$. In product recommendation, the fitting values are the relevance labels of the candidate products. We define three relevance levels, i.e., full-match, brand-match, mismatch with scores of 2, 1 and 0 respectively. The final model is built in a stage-wise process by performing gradient descent in the function space. At the mth boosting,

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta \rho_m h_m(\mathbf{x}; \mathbf{a}), \tag{4}$$

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. Shrinkage with small learning rates (i.e. $0 < \eta < 1$) is one important regularization technique which can improve the model's generalization performance compared to that without shrinkage (i.e. $\eta = 1$).

We first describe the general procedure for standard MART. Let $\{(\mathbf{x}_i, y_i)\}_1^N$ be a training set of N instances with each consisting of a feature vector and a relevance value. The general learning procedure for gradient boosting methods includes two major steps in each iteration

$$\mathbf{a}_m = \arg\min_{\mathbf{a}} \sum_{i=1}^{N} \big\{ - g_m(\mathbf{x}_i) - \beta h(\mathbf{x}_i; \mathbf{a}) \big\}^2, \tag{5}$$

$$\rho_m = \arg\min_{\rho} \sum_{i=1}^{N} \mathcal{L}\Big(y_i, F_{m-1}(\mathbf{x}) + \rho h(\mathbf{x}_i; \mathbf{a}_m) \Big), \tag{6}$$

where $-g_m(\mathbf{x})$ is the unconstrained negative gradient and is defined as follows

$$-g_m(\mathbf{x}_i) = -\left[\frac{\delta \mathcal{L}(y_i, F(\mathbf{x}_i))}{\delta F(\mathbf{x}_i)} \right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}. \tag{7}$$

The main idea of gradient boosting learning is that we fit a function $h_m(\mathbf{x})$ by using the steepest-descent method, i.e. $h_m(\mathbf{x}) = -\rho_m g_m(\mathbf{x})$. This gradient is defined only at the points in the training set and cannot be generalized to other values. Therefore, Eq. 5 aims to produce $\mathbf{h}_m = \{h(\mathbf{x}_i; \mathbf{a}_m)\}_1^N$ most parallel to $-\mathbf{g}_m = \{-g_m(\mathbf{x}_i)\}$, which is the most highly correlated with $-g(\mathbf{x})$ over the data distribution. In Eq. 6, we further minimize the loss function to derive the ensemble weight for the mth base learner.

6.2 Extending MART for Product Recommendation

In this part, we present several extensions to the above MART model for product recommendation: 1) we proposed to use the weighted loss function for weight-sensitive training; 2) we proposed to use feature sampling for avoid over-fitting for the new feature representation given in Section 2.3, which is different from the original feature representation in [7]; 3) we propose to bag the MART models to reduce the variance while keeping a high accuracy.

6.2.1 Model Learning with Weighted Loss Function

In real e-commerce recommender systems, different training instances may have varying relevance levels, e.g., a full-match product should be assigned with a larger weight than a brand-match product. Here we propose to incorporate different weights to model the relevance levels of training instances. With the weighted squared error function, we can have the following loss function

$$\min \sum_{i=1}^{N} \mathcal{L}(y_i, \hat{y}_i) = \min \sum_{i=1}^{N} w_i \big(y_i - \hat{y}_i \big)^2, \tag{8}$$

where $\hat{y}_i$ is a fitted value of the ith instance by the model and w_i is the relevance score (or weight) of the ith instance. We have three types of products in our training data, relevant (match a specific product), partially-relevant (match at a brand level) and non-relevant. The weights of training instances falling into each of these three types are empirically set to 4, 2 and 1 respectively.

With the new loss function, we first compute the pseudo response $\{\tilde{y}_i\}_{i=1}^N$, where $\tilde{y}_i = -g_m(\mathbf{x}_i) = w_i \big(y_i - F_{m-1}(\mathbf{x}_i) \big)$. Then we use the pseudo responses $\{\mathbf{x}_i, \tilde{y}_i\}_{i=1}^N$ to fit a regression tree and obtain a set of disjoint regions $\{R_{m,j}\}$. To learn the coefficient $\gamma_{m,j}$ for $R_{m,j}$, we minize the following loss function

$$\gamma_{m,j} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{m,j}} w_i (y_i - F_{m-1}(\mathbf{x}_i) - \gamma)^2, \tag{9}$$

which can be derived as

$$\gamma_{m,j} = \frac{\sum_{\mathbf{x}_i \in R_{m,j}} \big(w_i (y_i - F_{m-1}(\mathbf{x}_i)) \big)}{\sum_{\mathbf{x}_i \in R_{m,j}} w_i}.$$

The learning procedure for MART with weighted loss function is as follows. In the mth iteration,

- For $i = 1, ..., N$, compute the pseudo response $\tilde{y}_i$, where $\tilde{y}_i = -g_m(\mathbf{x}_i) = w_i(y_i - F_{m-1}(\mathbf{x}_i))$.
- Use the training set with pseudo responses $\{(\mathbf{x}_i, \tilde{y}_i)\}_{i=1}^N$ to fit a regression tree. We can obtain a set of disjoint regions $\{R_j\}$, and then set the region coefficient γ_j of R_j as $\gamma_{m,j} = \text{WeightedAverage}_{\mathbf{x}_i \in R_{m,j}} \{(y_i - \hat{y}_i)\}$.
- Add the learnt mth tree into the ensemble learner. Note that the ensemble weight ρ has already been incorporated into the coefficient γ.

6.2.2 Attribute-Based Feature Sampling

In our previous work [7], a demographic based feature corresponds to a demographic attribute (e.g., "gender"), while in this paper a demographic based feature corresponds to an attribute value (e.g., "female") in order to capture more fine-grained information (See Eq. 3). Using the value-level feature representation generates much more features and therefore 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 [26]. 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 $\frac{2}{3}$) instead of enumerating all the attributes based on each attribute's importance score. Once an attribute is sampled, its corresponding attribute value features will be selected. The importance score of each attribute is set to the proportion of the attribute that has its value filled in the users' public profiles in Sina Weibo, which will be further discussed in Section 6.4. As will be shown in our experimental results, using the proposed attribute-based feature sampling yields significant improvement over the original feature representation in [7] (See Table 3).

6.2.3 Bagging the MART

Boosting algorithms often have relatively high variance. As previous studies showed in [21], boosting mainly reduces the bias while bagging mainly reduces variance. Therefore, we consider a combination of boosting and bagging techniques to achieve a system with high accuracy and low variance. Bagging, also called bootstrap aggregating, combines outputs from multiple models each trained from a subset of an original training set created by sampling with replacement. To apply the bagging technique, we first generate multiple MART models, each of which is trained on a different bootstrap sample. We apply the sampling with replacement technique to generate multiple bootstrap samples, each of which contains the same size of training instances as in the original training dataset.

With the learnt multiple MARTs, we next study how to combine the ranks of different sub-models. To deal with different score scales of different sub-models, we consider the use of multiple rank aggregation techniques such as Borda count [27]. As Borda count may lead to ties, we follow the method in [21] by normalizing the scores of product generated by each sub-model for each query. In specific, for each query we linearly scale the scores of its corresponding products such that the product with the highest score will have a score of 1.0 and the product with the lowest score will have a score of 0. We call our extended gradient boosting trees as ***MART*** and the bagged gradient boosting trees as ***B-MART***.

7 Experiments

We evaluate the performance of recommending products on Jingdong to users who have expressed explicit purchase intents on Sina Weibo.

7.1 Experimental setup

Semi-automatic acquisition of training data. The performance of learning to rank algorithms highly relies on the amount of training data. Previous studies tend to use transaction records (users' actual purchase decisions) from online e-commerce websites as supervised information [1]. However, such data are not always available as e-commerce companies could choose not to reveal their transaction data. Even with the availability of transaction data, it is still difficult to link users across microblogs and e-commerce websites, which makes it infeasible for the evaluation of our current task.

We propose to acquire training data semi-automatically motivated by the following two example tweets from the same user:

> (*Oct 1, 2012*) Please recommend! I want to buy my son a Samsung phone.
> (*Oct 4, 2012*) Done! I *have bought* Galaxy II for my lovely son!

In the first tweet, a user expressed her desire to buy a Samsung phone for her son. The user subsequently reported that she has purchased a product. The above example forms a query-decision pair, which can be naturally converted into a training instance. To extract such query-decision pairs, we first identify purchase-intent tweets. Then for each tweet, a set of tweets from the same author which were subsequently published in a week time have also been retrieved. We compile some cue phrases which are indicative of reporting purchase decisions, e.g., "*have bought* something" and use them to identify those tweets potentially reporting purchasing decisions. We subsequently invite two human judges to manually examine whether these tweets actually report buying decisions and discard the false positives, and the overall Cohen's kappa coefficient is 0.87 indicating very high agreement.[12] The final set of query-decision (*qd*) pairs can be used as training instances for the learning to rank algorithms. To generate the training dataset, for each query-decision pair, we add three partially-relevant (match at a brand level) and at most 50 non-relevant best-sale products. We choose three popular product types, phone, camera and laptop, for the construction of our dataset for the evaluation of product recommendation. Although our model is tested on these three product types, it can be easily extended to other types. We select these three types because the related purchase intents widely exist on Sina Weibo, and the products themselves are with clear product demographics. The statistics of these three product types is summarized in Table 2.

TABLE 2
Statistics of the dataset for product recommendation.

Types	#brands	#models	#*qd* pairs	#instances
phone	57	1,584	170	4,732
camera	25	724	496	12,741
laptop	25	829	437	11,795

[12] On Sina Weibo, all the tweets from a user can be publicly seen by other registered users. The judges log into their own Weibo accounts and check the validity of each candidate query-product pair online. Each user's public profile of a user is also checked and spam users are removed. The workload for each judge is about $5 \sim 7$ times the number of *qd* pairs in Table 2, i.e., only $1/7 \sim 1/5$ of the originally detected *qd* pairs are finally kept as training data.

Evaluation metrics. The evaluation metrics adopted here are precision at k ($p@k$), Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG), which are commonly used in information retrieval. For $p@k$ and MRR, we consider the relevance at the product model level, i.e. the specific product actually purchased as described in a tweet is treated as the only relevant product. To compute $p@k$, we calcuate the proportion of queries for which we have made the correct recommendations in the top k positions. MRR is the average of the reciprocal of the highest ranks of the relevant documents given a query Q. NDCG takes into account both the ranking positions and the quality grade of products, and is defined as follows:

$$NDCG(\mathcal{Q},k) = \frac{1}{|\mathcal{Q}|}\sum_{j=1}^{|\mathcal{Q}|} Z_{k,j} \sum_{m=1}^{k} \frac{2^{R(j,m)}-1}{\log_2(1+m)}, \qquad (10)$$

where $\mathcal{Q}$ is the set of queries, $Z_{k,j}$ is the normalizer for the jth query which denotes the ideal ranking score for the top k positions, and $R(j,m)$ is the relevance level of the mth product for the jth query. For $R(j,m)$, we mainly consider three relevance levels: relevant (2), partially-relevant (1) and non-relevant (0). For each query, there is only one relevant product which matches a user's final purchase decision. If a product model does not match the final product purchased but has the same brand, it is treated as partially-relevant. For all the other cases, products are treated as non-relevant.

Methods to compare. We evaluate our proposed recommender system using the representative learning to rank algorithms from each of the categories, pointwise, pairwise and listwise:

- *Pointwise*: Logistic Regression;
- *Pairwise*: RankSVM [28], RankBoost [29];
- *Listwise*: Listnet [30], AdaRank [31].

We also use the implementation of MART in our previous work [7] as an additional baseline, denoted as $MART_{old}$. In this paper, $MART_{old}$ has been extended in the following ways: 1) new feature representation (Eq. 3); 2) weighted loss function (Eq. 8); 3) feature sampling (Sec. 5.2.3).

The open source toolkits, RankLib[13] is used for the implementation of the learning to rank algorithms. For fairness, we follow the default parameter settings in RankLib. For each algorithm, five-fold cross-validation is performed and the results are averaged over five such runs. As historical transaction records on JINGDONG is not available, we cannot compare our system with existing recommendation approaches relying on purchase history [1], [5]. Instead, we built three baselines with each of them using one of the three query-independent features to rank products from our generated candidate product list, i.e. product sales (*sales*), product polarity scores (*polarity*), and product rating scores (*rating*).

7.2 Product Recommendation Results

It can be observed from Table 3 that the learning to rank methods outperform the three simple baselines. Furthermore, the performance of the pointwise approaches is much better than that of pairwise and listwise approaches. The listwise approaches give the worst performance compared to the other two learning-to-rank categories. One main reason is that it is relatively difficult to collect enough evidence from training data to infer a total order for candidate products since there is usually only one correct produce decision[14]. Overall, the best performance is given by *MART* in the pointwise category. In specific, it can be observed that our proposed $MART$ improves over $MART_{old}$. We also notice that the performance in the *Camera* category is consistently better than that in the other two product categories. A possible reason is that, compared to *Phone* and *Laptop*, the *Camera* category contains fewer famous brands and models, and most users may simply buy from those well-known classic models (such as *Canon 5D Mark III*). By computing the percentage of top five popular products in each category, we have the following statistics: Laptop (3.8%) < Phone (6.9%) < Camera (9.5%), which indicates that *popularity* is an important factor to consider in recommendation and contributes more in the Camera category.

Bagging is likely to reduce the variance by aggregating multiple models. We randomly generate 100 subsets of training samples, with each containing 90% of the queries in the original training set. Then we run MART and B-MART on each of these 100 training subsets and average the results and compute the variance. It can be observed from Table 4 that bagging in general reduces the variance and also improves the overall performance.

7.3 Feature Analysis

Results with different feature types. As have been previously mentioned, we have a set of three query-independent features, *s̲ales*, *p̲olarity* and *r̲ating*, denoted as *spr*. We also have two types of query-dependent features *Weibo* generated from microblogs in SINA WEIBO, and *JD* derived from online reviews in JINGDONG. We build our recommender systems using our proposed extended version of *MART*, trained from different types of the features. The results are presented in Table 5. We can see that 1) using the three

[13] https://sourceforge.net/p/lemur/wiki/RankLib/
RankLib might assign equal scores to items during ranking. In this case, we further sort the items of equal scores by their sales volume.

[14] For the listwise approach, each training instance is an ordered list. However, the relative order between non-relevant products is not possible to obtain in our training data.

TABLE 3

Performance comparison of product recommendation on three datasets for $p@k$ and $NDCG@k$. $***$ and $**$ indicates that the improvement that $MART$ over the best baseline $MART_{old}$ is significant at the levels of 0.001 and 0.01. The improvements of $MART$ over the other baselines are significant at the level of 0.001.

Types	Metrics	Baselines			Pointwise			Pairwise		Listwise	
		sales	polarity	rating	$MART_{old}$	$MART$	Regression	RankSVM	RankBoost	Listnet	AdaRank
PHONE	p@5	0.303	0.248	0.124	0.310	**0.386*** **	0.303	0.338	0.359	0.234	0.138
	MRR	0.132	0.155	0.062	0.190	**0.210** **	0.132	0.189	0.170	0.110	0.073
	NDCG@5	0.226	0.162	0.073	0.248	**0.278** **	0.226	0.240	0.250	0.132	0.094
CAMERA	p@5	0.577	0.043	0.050	0.624	**0.638** **	0.549	0.551	0.574	0.535	0.142
	MRR	0.391	0.030	0.029	0.418	**0.430** **	0.370	0.379	0.404	0.381	0.081
	NDCG@5	0.368	0.044	0.044	**0.417**	0.411	0.354	0.352	0.378	0.347	0.102
LAPTOP	p@5	0.219	0.041	0.029	0.403	**0.419** **	0.216	0.146	0.216	0.162	0.105
	MRR	0.148	0.028	0.024	0.245	**0.277** **	0.129	0.095	0.142	0.095	0.063
	NDCG@5	0.142	0.050	0.041	0.265	**0.278** **	0.134	0.110	0.143	0.105	0.086

TABLE 4

Comparison of the mean and variance for MART and B-MART.

METRICS	Phone				Camera				Laptop			
	Mean		Variance ($\times 10^{-3}$)		Mean		Variance ($\times 10^{-3}$)		Mean		Variance ($\times 10^{-3}$)	
	MART	B-MART	MART	B-MART	MART	B-MART	MART	B-MART	MART	B-MART	MART	B-MART
p@5	0.307	0.308	3.58	2.98	0.602	0.614	2.20	1.67	0.370	0.375	1.49	1.43
MRR	0.189	0.195	1.09	1.06	0.382	0.404	0.96	0.54	0.252	0.253	0.61	0.51
NDCG@5	0.247	0.253	1.10	0.96	0.386	0.407	0.58	0.35	0.248	0.257	0.35	0.31

query-independent features alone (*spr*) gives strong baseline results; 2) the combination of all the features achieves the best performance; and 3) demographic features derived from online reviews in JINGDONG and those extracted from microblogs are effective to improve the recommendation performance. Combining these two types of demographic features (*Weibo + JD*) yields a competitive performance compared to that of using all the features.

TABLE 5

Performance comparison with different type of features for *MART*. "*Both*" denotes "*Weibo + JD*".

Types	Metrics	spr	Weibo	JD	Both	all
p@5	PHONE	0.255	0.407	0.324	**0.434**	0.345
	CAMERA	0.601	0.579	0.442	0.610	**0.633**
	LAPTOP	0.422	0.289	0.257	0.346	**0.432**
NDCG@5	PHONE	0.178	0.295	0.226	**0.290**	0.262
	CAMERA	0.380	0.379	0.295	0.388	**0.408**
	LAPTOP	0.254	0.209	0.164	0.238	**0.278**

Qualitative analysis of demographic features. We present in Table 6 some learned demographic features from online reviews for a phone product, Samsung Galaxy S4. In specific we only show the associated demographic features for "color". It can be observed that the demographic distributions indeed varies with different colors. Young females prefer *white* phones while young males like *black* phones more.

Table 7 shows some learned product demographic features of two different phone brands, Apple and Samsung, from microblogs. We have a couple of interesting observations, 1) Samsung has a more balanced sex distribution; 2) Apple products are more preferred by the consumers in the IT field.

TABLE 6

Samples of the learnt product demographics based on online reviews. Real numbers denote the learned weights of the corresponding attribute values.

Galaxy S4 (White) Galaxy S4 (Blue) Galaxy S4 (Black)	$(Gender, [“male”, 0.271], [“female”, 0.729])$ $(Gender, [“male”, 0.688], [“female”, 0.312])$ $(Gender, [“male”, 0.852], [“female”, 0.148])$
Galaxy S4 (White) Galaxy S4 (Blue) Galaxy S4 (Black)	$(Age, [“ < 45”, 0.931], [“ \geq 45”, 0.069])$ $(Age, [“ < 45”, 0.755], [“ \geq 45”, 0.245])$ $(Age, [“ < 45”, 0.650], [“ \geq 45”, 0.350])$

TABLE 7

Samples of the learnt product demographic based on microblogs.

Apple	$(Gender, [“male”, 0.593], [“female”, 0.407])$
	(Career, [“IT”, 0.28], [“management”, 0.219] , [“media”, 0.172], [“industry”, 0.139])
	(Tag, [“music&movie”, 0.316], [“travel”, 0.249] , [“Internet surfing”, 0.163], [“computer games”, 0.161])
Samsung	$(Gender, [“male”, 0.503], [“female”, 0.497])$
	(Career, [“management”, 0.252], [“IT”, 0.223] , [“industry”, 0.252], [“media”, 0.223])
	(Tag, [“computer games”, 0.281], [“travel”, 0.27] , [“music&movie”, 0.209], [“Internet surfing”, 0.188])

Relative attribute importance. Tree-based methods offer additional feasibility to learn relative importance of each attribute. The results of relative attribute importance allows an easy interpretation of the recommendations generated by MART. Inspired by the method introduced in [25], we calculate a statistic of the relative importance of each attribute for MART based on the training data. Recall that a demographic 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. An attribute is associated with a set of attribute-value features, and we can sum the contributions of its attribute-value features as the attribute contribution. Formally, the overall contribution of attribute a is an average of its contributions over all regression trees with J terminal nodes:

$$\hat{I}_a^2 = \frac{1}{M}\sum_{m=1}^{M} \hat{I}_a^2(T_m) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{J-1} \mathbf{1}(v_t^{(m)} \in \mathcal{V}_a)\Delta_t^{(m)}, \quad (11)$$

where $v_t^{(m)}$ stands for the splitting feature of the t-th node of the m-th regression tree, and $\Delta_t^{(m)}$ is the reduction of the squared error by the current split at the t-th node of the m-th regression tree. The results are shown in Figure 5. As a comparison, we also include the relative importance of query-independent features using a similar method.

From Figure 5 we have some interesting observations: 1) *Sales* is an very important feature across all the product types, and is more prominent in the *Camera* category. This is in line with our previous observation in Section 7.2 that the recommendation performance in the *Camera* category is better than the other two product categories if simply ranking products based on their *sales* volume. 2) Demographic attributes are important to consider for product recommendation. Also, the demographic information derived from Weibo appear to be more effective than those derived from Jingdong. 3) Among all the demographic attributes, *gender*, *tag* and *age* are the top three most important attributes overall.

7.4 Inferring Missing User Demographic Profiles

In Table 1, we have presented the six demographic attributes. It is worth noting that not all users have filled in all the attributes in their public profiles. We want to check how the system performance is affected by the completeness of user profiles. We first analyze the dataset of three product categories as shown in Table 2, where a query-decision pair corresponds to a unique Weibo user. The average number of attributes a user has filled in for each product category is as follows: 2.73 ± 1.10 (laptop), 2.66 ± 1.12 (camera) and 2.81 ± 1.07 (phone). Our evaluation results show that our system generally works well when a user has filled in three or more attributes. To examine the applicability of our system on a large Weibo user population, we compute the proportions of the user demographic attributes that can be found in the public profiles of a total of 5 million Weibo users: *gender* (100%), *age* (36.7%), marital status (4.6%), education (26.3%), career (12.9%) and *tags* (65.7%). It can be seen that apart from marital status, all the other attributes have considerable filled-in values, especially the top two most important attributes *gender* and *age*.

In our next set of experiments, we want to find out whether the performance of our recommender system can be improved if we can infer the absent demographic attribute values in users' public profiles. Inferring users' demographic profiles is itself a challenging research problem. In our experiments here, we take a relatively simple approach based on the "homophile" property of user connections in social networks. In specific, we assume that users who *follow* each other are more alike compared to those they do not [32], [12]. Therefore, we fill in the missing demographic attribute values of a user by averaging over the values of all the users that she follows. Formally, given attribute a and user u, we first calculate the occurrence frequency of an attribute value:

$$\phi_v^{(u,a)} \propto \sum_{u' \in \mathcal{F}_u} \mathbf{1}[v_a^{(u')} = v] \quad (12)$$

where $\{\phi_v^{(u,a)}\}$ is the demographic distribution of user u for attribute a, $\mathcal{F}_u$ is the set of u's following users and $\mathbf{1}[\cdot]$ is an indicator function which only returns 1 when the statement is true and 0 otherwise. Then we normalize $\{\phi_v^{(u,a)}\}_{v \in \mathcal{V}_a}$ to make it a valid probability distribution. The performance evaluation results before and after filling in the missing demographic attribute values are shown in Table 8. We can observe that by filling in the missing demographic attribute values, the performance has been improved to an extent. In our future work, we will investigate more principled ways such as the methods introduced in [12], [13], [14] for more accurate inference of user demographic attributes. Also, we can attach confidence scores to the inferred demographic attribute values before incorporating them into our recommender system.

TABLE 8
Comparison of performance with original and inferred user demographic profiles.

Metrics	Phone		Camera		Laptop	
	ori.	*inf.*	*ori.*	*inf.*	*ori.*	*inf.*
p@5	**0.386**	**0.386**	0.636	**0.643**	0.413	**0.438**
MRR	**0.210**	0.204	0.417	**0.431**	**0.279**	**0.279**
NDCG@5	0.278	**0.282**	0.404	**0.412**	0.274	**0.285**

7.5 User Study

In this section, we evaluate the proposed method in a real-world deployment through user study. We compare our proposed *B-$MART$* method, an ensemble of 100 $MART$ components with the previously implemented $MART_{old}$ in [7].

We adopt the balanced interleaving method [33] used for the evaluation of search engines to compare the performance of $MART_{old}$ and *B-$MART$*.[15]

To conduct the user study, we first randomly selected 5,000 real users, and then sent an invitation

[15] Balanced interleaving method reflects the intuition that the results of the two rankings A and B should be interleaved into a single ranking I in a balanced way, which ensures that any top k results in I always contain the top k_a results from A and the top k_b results from B, where k_a and k_b differ by at most 1.

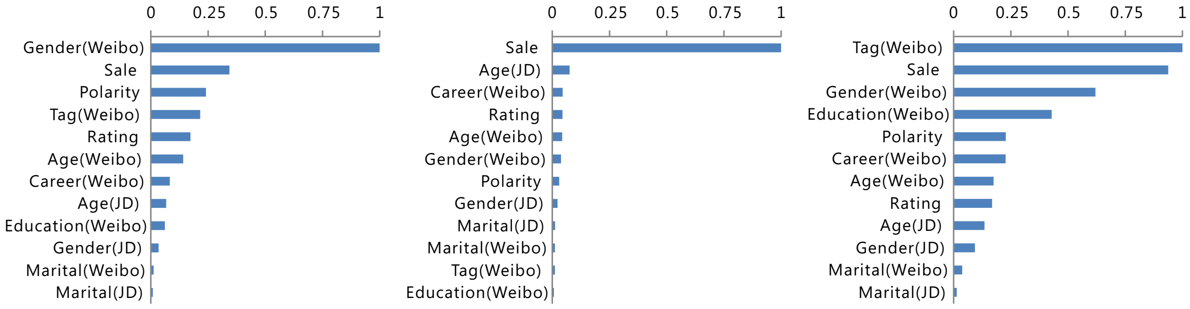

Fig. 2. Phone

Fig. 3. Camera

Fig. 4. Laptop

Fig. 5. Attributes ranked by their relative importance on the the three product categories.

message to each of them using the API provided by SINA WEIBO. Finally, 47 users agreed to participate in this study with small monetary incentives provided to them. For each user, our system generates a ranked list of K products ($K = 10$) based on the user's microblogging profile for each of three product types respectively, i.e., phone, camera and laptop. Each recommended product is also accompanied with a detailed product description and user reviews. The balanced interleaving method is then employed to combine the rankings produced by $MART_{old}$ and B-$MART$ into a single ranking. And the user is required to select a potential product that she is likely to purchase. A user has an option to select nothing if she cannot find any interesting product.

We use the measure proposed in [33] to quantify the degree of preference in this interleaving experiment. Given two comparison systems A and B, we first define two quantities $wins(A)$ and $wins(B)$. In evaluation, $wins(A)/wins(B)$ will be incremented by 1 for any query where A/B was preferred, and $ties(A, B)$ is incremented by 1 with an equal preference. *Preference* is defined to be a higher original position of a selected product in a compared ranking. Queries without clicks are ignored. We can now define the statistic

$$\Delta_{AB} = \frac{wins(A) + \frac{1}{2}ties(A, B)}{wins(A) + wins(B) + ties(A, B)} - 0.5, \qquad (13)$$

where Δ_{AB} reflects the degree how system A is better than B. It is easy to verify that the following cases: (1) $\Delta_{AB} = 0$ if $wins(A) = wins(B)$; (2) $\Delta_{AB} > 0$ if $wins(A) > wins(B)$; (3) $\Delta_{AB} < 0$ if $wins(A) < wins(B)$. Another important property is that Δ_{AB} monotonously increases with the increasing of $wins(A)$ by fixing $wins(B)$ and $ties(A, B)$. Table 9 reports the results of our user study. The values of $P@10$ are also reported which measure whether any of the top ten recommendations matches users' preferences. It can be observed that B-$MART$ is consistently better than $MART_{old}$ in terms of Δ_{AB}. By looking into the values for $wins(A)$ and $wins(B)$, we can see that our system is more capable to generate product recommendations which closely match users' preferences. The values of $P@10$ indicate that about 50% of the test users can find suitable products in the top ten recommendations generated by our system, which shows a great potential to be used in real-world applications.

TABLE 9
Evaluation results via user study. A and B denote B-$MART$ and $MART_{old}$ respectively. Removing queries without clicks, the values of $wins(A)$, $wins(B)$ and $ties_{AB}$ are shown in parentheses.

Statistics		Phone	Camera	Laptop
Δ_{AB}		7.4% (16,9,22)	7.6% (16,9,21)	9.0% (15,7,22)
P@10	A	0.489 (+41.2%)	0.522 (+9.2%)	0.500 (+25.0%)
	B	0.340	0.478	0.409

8 RELATED WORK

Our work is mainly related to four lines of research:

Product recommendation

Early work on product recommendation mainly relies on collaborative filtering which makes recommendations based on matching users with similar interests [34], [35], [5]. Collaborative filtering suffers from the "cold start" problem. Recently, there have been some attempts to incorporate information from online social networks into recommendation [36]. In particular, demographic information has been shown to be useful to improve the recommendation performance [37], [38], [3]. Our work is related to the aforementioned research. Nevertheless, we presented the first study which identifies users with purchase needs on social networks and make product recommendations by jointly considering both users' and products' demographic information. Some design issues and suggested guidelines for the development of product recommender systems have been previously discussed in [2], [4]. We have followed the suggested guidelines but proposed different methodologies for the implementation of our system.

With the rapid growth of online e-commerce services, online review mining has become a hot research topic [39]. In particular, it has been shown that online review are useful to improve the results of product ranking or recommendation. Liu et al. ([40]) proposed to use a sentiment model to predict sales performance; while in [41], composite rating scores were derived from aggregated reviews collected from multiple websites using different statistic- and heuristic-based methods and were subsequently used to rank products and merchants. Ganu et al. ([42]) derived text-based ratings of item aspects from review text and then grouped similar users together based on the topics and sentiments that appear in the reviews. Zhang et al. ([43], [44]) extracted explicit product features (i.e. aspects) and user opinions by phrase-level sentiment analysis on user reviews for product recommendation.

Demographic information has been important for recommender systems [35]. Typically, many existing studies utilize the demographic information obtained directly from user websites [45], [3] or questionnaires [38]. Besides the users' registered profile, Seroussi et al. ([46]) proposed to extract topics from user-generated text using the Latent Dirichlet Allocation (LDA) model, termed as text-based user attributes. Both types of attributes were then integrated into a matrix factorization model for rating prediction. Korfiatisa and Poulos ([37]) proposed to build a demographic recommender system by extracting service quality indicators (star ratings) and consumer types from hotel reviews. They defined different demographic groups by consumer types based on the assumption that different types of travelers assess each quality indicator differently.

Learning to rank

Learning to rank was originally proposed in information retrieval, where it aims to incorporate various features in a formal way to improve the ranking performance of the retrieved results [19]. In this paper, we formulate the product recommendation task as a learning to rank problem and evaluate various learning to rank algorithms on both the query-dependent and query-independent features derived from social media data. The learning to rank algorithms we tested include pointwise approaches, MART [47] and RandomForest [26], pairwise approaches, RankSVM [28] and RankBoost [29], and listwise approaches, ListNet [30] and AdaRank [31].

Social network mining

Commercial intent detection and analysis has been performed on the Web data [48] and social media data such as Twitter [6]. We also consider automatically detecting purchase intents from microblogs but with different feature representation and with an additional incorporation of users' demographic features. Furthermore, we have explored efficient implementation of purchase intent detection in order to achieve the near real-time performance when dealing with large amount of social stream data.

In recent years, there has also been some work on identifying individual's demographic characteristics such as age, gender and interests from social media data [13], [14]. In this paper, we extract users' demographic information from their public profiles in SINA WEIBO and perform inference of missing demographic attributes by considering friends' information in WEIBO. We will explore more sophisticated methods for inferring users' demographic attributes in future work.

9 CONCLUSIONS

In this paper we have presented a novel demographic based product recommender system which detects users' purchase intents from their microblogs in near real-time and makes product recommendations based on matching the users' demographic information extracted from their public profiles with product demographics learned from microblogs and online reviews. We have shown that recommendation can be framed as a learning-to-rank problem which takes as input features derived from both product and user demographics. An ensemble method based on the gradient boosting regression trees is extended in a number of ways to make it suitable for our recommendation task. Our experimental results on large-scale microblog data crawled from SINA WEIBO have verified the feasibility and effectiveness of our proposed recommender system. In addition, we have conducted a user study to gauge the performance of our system in a real-world deployment. The results have also shown that our system is more effective in generating recommendation results better matching users' preferences. We believe our study will have profound influence on both research and industrial communities.

Limitations and future work: Our approach is developed based on the key idea by representing users and items with the same demographic attributes, which allows a direct comparison for product recommendation. There are two major limitations in our approach. First, it is difficult to work on those product types which do not receive considerable attention on online social media, e.g., *purified water*. Second, currently we only consider the explicitly set attributes on microblogs. As such, our approach cannot work well in cases when users' demographic attributes are missing in their microblogging profiles. Our future work will mainly focus on solving the second limitation by automatically filling users' demographic attributes and extracting hidden representation patterns by leveraging from information revealed from both content and social network data.

10 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 Grant No. 61502502 and No. 61573026, the pilot project under Baidu open cloud service platform under Grant No. 4333150064, and the National Key Basic Research Program (973 Program) of China under Grant No. 2014CB340403. Xin Zhao was also partially supported by 2015 HTC Young Scholar Program.

References

[1] J. Wang and Y. Zhang, "Opportunity model for e-commerce recommendation: Right product; right time," ser. SIGIR '13, 2013.

[2] F. von Reischach, F. Michahelles, and A. Schmidt, "The design space of ubiquitous product recommendation systems," ser. MUM '09, 2009.

[3] M. Giering, "Retail sales prediction and item recommendations using customer demographics at store level," *SIGKDD Explor. Newsl.*, vol. 10, no. 2, Dec. 2008.

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

[5] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, Jan. 2003.

[6] B. Hollerit, M. Kröll, and M. Strohmaier, "Towards linking buyers and sellers: Detecting commercial intent on twitter," ser. WWW '13 Companion, 2013.

[7] X. W. 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 *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, 2014, pp. 1935–1944.

[8] M. Baker and S. Hart, *The Marketing Book*. Routledge; 6 edition (November 1, 2007), 2007.

[9] G. Sridhar, "Consumer involvement in product choice - a demographic analysis," *XIMB Journal of Management*, 2007.

[10] V. A. Zeithaml, "The new demographics and market fragmentation," *Journal of Marketing*, vol. 49, pp. 64–75, 1985.

[11] K. Tsiptsis and A. Chorianopoulos, *Data Mining Techniques in CRM: Inside Customer Segmentation*. John Wiley & Sons, Ltd, 2010.

[12] Y. Dong, Y. Yang, J. Tang, Y. Yang, and N. V. Chawla, "Inferring user demographics and social strategies in mobile social networks," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '14, 2014, pp. 15–24.

[13] A. Mislove, B. Viswanath, K. P. Gummadi, and P. Druschel, "You are who you know: Inferring user profiles in online social networks," ser. WSDM '10, 2010.

[14] B. Bi, M. Shokouhi, M. Kosinski, and T. Graepel, "Inferring the demographics of search users: Social data meets search queries," ser. WWW '13, 2013.

[15] B. Zou, G. Zhou, and Q. Zhu, "Negation focus identification with contextual discourse information," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, June 2014, pp. 522–530.

[16] "Us demographic and business summary data," *Product guide*, 2012.

[17] C. Zhai and J. D. Lafferty, "A study of smoothing methods for language models applied to information retrieval," *ACM Trans. Inf. Syst.*, vol. 22, no. 2, pp. 179–214, 2004.

[18] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," ser. ACL '04, 2004.

[19] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, Mar. 2009.

[20] P. D. Turney, "Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02, 2002, pp. 417–424.

[21] Y. Ganjisaffar, R. Caruana, and C. V. Lopes, "Bagging gradient-boosted trees for high precision, low variance ranking models," in *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '11, 2011, pp. 85–94.

[22] H. Zhang, E. Riedl, V. A. Petrushin, S. Pal, and J. Spoelstra, "Committee based prediction system for recommendation: KDD cup 2011, track2," in *Proceedings of KDD Cup 2011 competition, San Diego, CA, USA, 2011*, 2012, pp. 215–229.

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

[24] J. H. Friedman, "Stochastic gradient boosting," *Comput. Stat. Data Anal.*, vol. 38, no. 4, pp. 367–378, Feb. 2002.

[25] ——, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

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

[27] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 1, pp. 66–75, Jan. 1994.

[28] T. Joachims, "Training linear svms in linear time," ser. KDD '06, 2006.

[29] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, pp. 933–969, 2003.

[30] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," ser. ICML '07, 2007.

[31] J. Xu and H. Li, "Adarank: A boosting algorithm for information retrieval," ser. SIGIR '07, 2007.

[32] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "Twitterrank: finding topic-sensitive influential twitterers," in *WSDM*, 2010.

[33] O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue, "Large-scale validation and analysis of interleaved search evaluation," *ACM Trans. Inf. Syst.*, vol. 30, no. 1, pp. 6:1–6:41, Mar. 2012.

[34] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," ser. WWW '01, 2001.

[35] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE TKDE*, vol. 17, no. 6, Jun. 2005.

[36] P. Symeonidis, E. Tiakas, and Y. Manolopoulos, "Product recommendation and rating prediction based on multi-modal social networks," ser. RecSys '11, 2011.

[37] N. Korfiatis and M. Poulos, "Using online consumer reviews as a source for demographic recommendations: A case study using online travel reviews," *Expert Syst. Appl.*, vol. 40, no. 14, 2013.

[38] L. Qiu and I. Benbasat, "A study of demographic embodiments of product recommendation agents in electronic commerce," *Int. J. Hum.-Comput. Stud.*, vol. 68, no. 10, pp. 669–688, Oct. 2010.

[39] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[40] Y. Liu, J. Huang, A. An, and X. Yu, "ARSA: A sentiment-aware model for predicting sales performance using blogs," in *SIGIR*, 2007.

[41] M. McGlohon, N. S. Glance, and Z. Reiter, "Star quality: Aggregating reviews to rank products and merchants." in *ICWSM*, 2010.

[42] G. Ganu, Y. Kakodkar, and A. Marian, "Improving the quality of predictions using textual information in online user reviews," *Inf. Syst.*, vol. 38, no. 1, 2013.

[43] 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.

[44] Y. Zhang, H. Zhang, M. Zhang, Y. Liu, and S. Ma, "Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification," in *SIGIR*, 2014.

[45] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artificial Intelligence Review*, vol. 13, no. 5-6, 1999.

[46] Y. Seroussi, F. Bohnert, and I. Zukerman, "Personalised rating prediction for new users using latent factor models," in *ACM HH*, 2011.

[47] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, vol. 29, pp. 1189–1232, 2000.

[48] H. K. Dai, L. Zhao, Z. Nie, J.-R. Wen, L. Wang, and Y. Li, "Detecting online commercial intention (oci)," in *WWW '06*, 2006.

PLACE PHOTO HERE

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 and journals such as ACL, EMNLP, COLING, ECIR, CIKM, SIGIR, SIGKDD, ACM TOIS, ACM TIST and IEEE TKDE.

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.

PLACE PHOTO HERE

Yulan He is a Reader at the School of Engineering and Applied Science, Aston University, UK. She received her PhD degree from Cambridge University working on statistical models to spoken language understanding. She has published over 100 articles with most appeared in high impact journals and at top conferences. Her research interests include natural language processing, statistical modelling, text and data mining, sentiment analysis, and social media analysis.

PLACE PHOTO HERE

Liwei Wang is a professor in the Department of Machine Intelligence at Peking University, China. He has a BS and MS in electronics engineering from Tsinghua University and a PhD in applied mathematics from Peking University. His honors include the *Pattern Recognition Letters* Top Cited Article 2005-2010 and a coauthored paper that won the MIRU outstanding paper award. He has been named by IEEE Intelligent Systems as among "AI's 10 to Watch" in 2011, which celebrates 10 rising stars in the field of artificial intelligence (AI).

PLACE PHOTO HERE

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).

PLACE PHOTO HERE

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.