World Wide Web (2019) 22:325–345
https://doi.org/10.1007/s11280-018-0568-z

User click prediction for personalized job recommendation

Miao Jiang[1] · Yi Fang[2] · Huangming Xie[3] · Jike Chong[4] · Meng Meng[3]

Received: 30 December 2015 / Revised: 24 November 2017 / Accepted: 8 April 2018 /
Published online: 23 April 2018

Abstract Major job search engines aggregate tens of millions of job postings online to enable job seekers to find valuable employment opportunities. Predicting the probability that a given user clicks on jobs is crucial to job search engines as the prediction can be used to provide personalized job recommendations for job seekers. This paper presents a real-world job recommender system in which job seekers subscribe to email alert to receive new job postings that match their specific interests. The architecture of the system is introduced with the focus on the recommendation and ranking component. Based on observations of click behaviors of a large number of users in a major job search engine, we develop a set of features that reflect the click behavior of individual job seekers. Furthermore, we observe that patterns of missing features may indicate various types of job seekers. We propose a probabilistic model to cluster users based on missing features and learn the corresponding prediction models for individual clusters. The parameters in this clustering-prediction process are jointly estimated by EM algorithm. We conduct experiments on a real-world testbed by comparing various models and features. The results demonstrate the effectiveness of our proposed personalized approach to user click prediction.

Keywords Click prediction · Personalization · Job recommendation · Missing data

✉ Miao Jiang
miaojiang1987@gmail.com

Yi Fang
yfang@scu.edu

[1] Indiana University, 901 E 10th, Bloomington, IN 47408, USA

[2] Computer Engineering Department, Santa Clara University, 500 El Camino Real, Santa Clara, California 95053, USA

[3] LinkedIn, 2029 Stierlin Ct, Mountain View, California, USA

[4] Acorns Grow, Inc., 19900 MacArthur Blvd, Suite 400, Irvine, CA 92612, USA

1 Introduction

Major job search engines such as *Simply Hired*,[1] *Indeed*,[2] and *Glassdoor*[3] aggregate tens of millions of job postings online to provide job seekers with portals to find the employment opportunities that match their interests. To provide the most relevant job postings for a job seeker, a job search engine rely on the search queries provided by the job seeker to infer the job seeker's intent. As new job openings get created by employers on a daily basis, it is not the best option for a user to make a daily basis search. In order to catch the earliest job opportunities, many job seekers choose to engage with the job search engine through email alerts, where job postings relevant to the job seeker's interests are directly sent to their emails. Recommendation through email alerts is an important form of engagement for job search engines, as it turns a short-term engagement such as search into a long-term engagement such as email subscription. Figure 1 shows an example of job recommendations through email alerts by *Simply Hired*.

A query for job search involves a keyword and location pair, e.g. software engineer, San Francisco, CA. Such a query returns job openings in a local job market in which a job seeker can explore employment opportunities. For popular search queries, there are often hundreds to thousands of relevant job postings everyday, and only a limited number can fit in an email (e.g., SimplyHired's email alert service recommends a maximum number of 22 jobs in an email). If the search algorithm relies only on the search query, the limited number of jobs that can fit in an email may not contain the job posting that would be most interesting and relevant to the job seeker. Consequently, engagement with the email could be negatively affected. On the other hand, if the history of a job seeker's online activities is available, we can infer the user intent, and customize job recommendations for her.

This paper focuses on predicting user clicks on job postings for the purpose of personalizing job recommendation results through email alerts. To the best of our knowledge, no prior work exists in the literature on personalized job recommendation through email alerts. Our contributions can be summarized as follows:

- We present the architecture of a real-world job recommender system that currently serves a large number of online job seekers.
- We develop a set of user-specific features that reflect the click behavior of individual job seekers and groups. These features are derived from observations of click behaviors of a large user base in a major commercial job search engine. These features can be used by machine learning models to learn the preferences of specific job seekers.
- We analyze the missing data in the features and observe that the patterns of missing features may indicate various types of job seekers. We argue that a single model may not be able to capture the wide variate of all the users and jobs. We proposed a probabilistic clustering-prediction model to train different prediction models for different user clusters based on missing features. The clustering and prediction are done in a unified probabilistic framework.
- We conduct experiments on a real-world testbed by comparing various models and features. The results demonstrate the effectiveness of our proposed personalized approach to click prediction.

[1] http://www.simplyhired.com/

[2] http://www.indeed.com/

[3] http://www.glassdoor.com

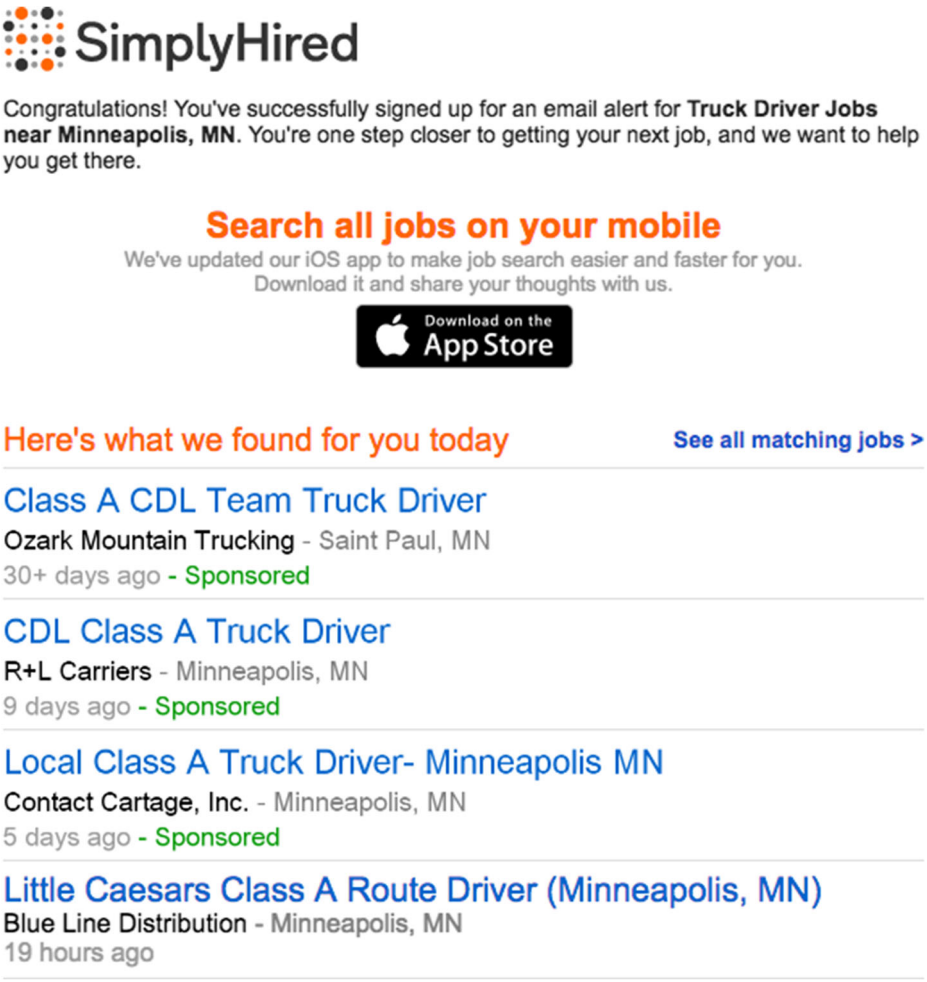

Figure 1 An example of *Simply Hired*'s email alert service for job recommendation

2 Related work

The early job recommender systems relied on Boolean search which often could not satisfy job seekers' complex information needs [21]. Later, classic recommendation techniques such as collaborative filtering and content based filtering were applied to recommend jobs [10]. Malinowski et al. [21] proposed a probabilistic model that estimates the probability that an applicant likes a job. [15] further extended the work by proposing a unified multi-layer framework to support the matching of individuals for jobs and team members who are willing to collaborate with them. A machine learning based approach was applied to job recommendation by extracting a number of job transitions from publicly available employee profiles on the Web [26]. Some works attempted to build user profiles by extracting the meta-data from resumes including demographics, education, experience, language skills, and so on [21]. [27] constructed user profiles by passively detecting the click streams and

user behaviors. [9] collected and analyzed the job related information from various social media sources. A survey on job recommendation can be found in [1]. More recently, [33] investigated the time effect, i.e., when is the right time to make a job recommendation. They proposed a hierarchical proportional hazards model by characterizing the tenure between two successive decisions and related factors. A hybrid job recommender system is proposed by combining content similarity with the PageRank algorithm [39].

Several implementations of job recommender systems were presented including CASPER [5], Proactive [17], and *Absolventen* [14]. The goal of CASPER was to personalize job search in two steps [5]: a server side similarity-based retrieval of jobs and a client side case-based personalization. Specifically, to construct user profile, implicit feedback from various user actions was used including applying for a job, emailing, reading, and visiting a job. The authors showed that implicit feedback was a valuable input for personalized job search as it could be monitored without much extra effort. *Proactive* [17] was a job recommender in which every job advertisement included a bookmark option. The bookmarked jobs were then stored in the user profile and then passed as input to the actual recommender which identified similar jobs based on content. Lee and Brusilovsky [18] extended these profiles by inferring preferences from implicit negative feedback. Specifically, if a job seeker read a job description without adding it to the bookmarks, it was deemed as negative feedback. Their experimental results showed that adding of negative feedback was helpful in order to distinguish between relevant and non-relevant jobs. *Absolventen*[4] is an Austrian job board for graduates and a recommender system is used to suggest appropriate jobs to applicants. It was shown that only around half of the registered job seekers fill out the resume, for the other half no personalized recommendations can be generated. To improve this, the recommender system has been enhanced with implicit relevance feedback and the impacts of this approach have been examined in [14]. Four different user actions for implicit feedback have been identified including reading of a job description, bookmarking, applying and searching for jobs.

Some job recommender systems also offer job seeker recommendations [21, 32, 37] (i.e., given a job, find the most qualified candidates). In fact, job recommendation problem is bidirectional recommendation between job seekers and jobs [37]. In the Bilateral Person-Job Recommender [21], the preferences of the recruiter were accumulated with the previously selected candidates, whereas the job seeker's profile consisted of information from relevant jobs. A probabilistic model was established that predicts whether a job seeker is relevant or not. This people recommendation task is closely related to expert search (i.e., given an expertise area, find the people who have the expertise), which is an active research area in the field of information retrieval. A comprehensive survey on expertise retrieval can be found in [3].

Click prediction has been extensively investigated as a core component in sponsored search for ranking ads. The state-of-the-art sponsored search systems typically rely on a machine learned model to predict the clickability of ads returned for a user search query. In previous studies [2, 12], the historical click information for each ad is shown to be effective for predicting the future click probability of the ad. Cheng and Cantú-Paz [8] further studied the personalization of click models in sponsored search so that the users would be presented ads most relevant to them. However, it has been shown that practical sponsored search systems often contain many ads without adequate historical click-through data, even after aggregation at different levels [34]. Consequently, there are many features missing in the click prediction models, which deteriorates the predictive performance of those machine learning based models.

[4] https://www.absolventen.at/

The presence of missing data also often poses a great challenge to many recommender systems. Much existing work applied data smoothing methods to fill the missing values. For example, [36] proposed a clustering based smoothing technique to predict all the missing data based on the ratings of most similar users in the similar clusters. Ma et al. [20] propose a missing data prediction algorithm taking information of both users and items into account. Marlin et al. [23] systematically investigated the effect of *missing at random* in collaborative filtering.

Another area of related work is learning to rank (L2R) for recommendation. L2R employs a machine learning approach to rank documents and has attracted a lot of attention in the information retrieval community. The L2R techniques can be loosely grouped into three categories: pointwise, pairwise, and listwise [19]. Since recommender systems aim at providing users with personalized items with a descending order of relevance, personalized recommendation can be naturally casted as a ranking problem. Recent top-N recommendation techniques have exploited learning to rank approaches including pointwise ordinal model for predicting personalized item ratings [16], pairwise Bayesian personalized ranking [28], and listwise collaborative filtering technique such as CLiMF [31] and TFMAP [30]. Several of the above models including [28, 30, 31] have utilized implicit feedback to optimize binary relevance data ranking metrics.

3 A real-world personalized job recommender

We present the personalized job recommender designed in the context of a job search engine at *Simply Hired*, an online job aggregator website. The website provides a search engine for

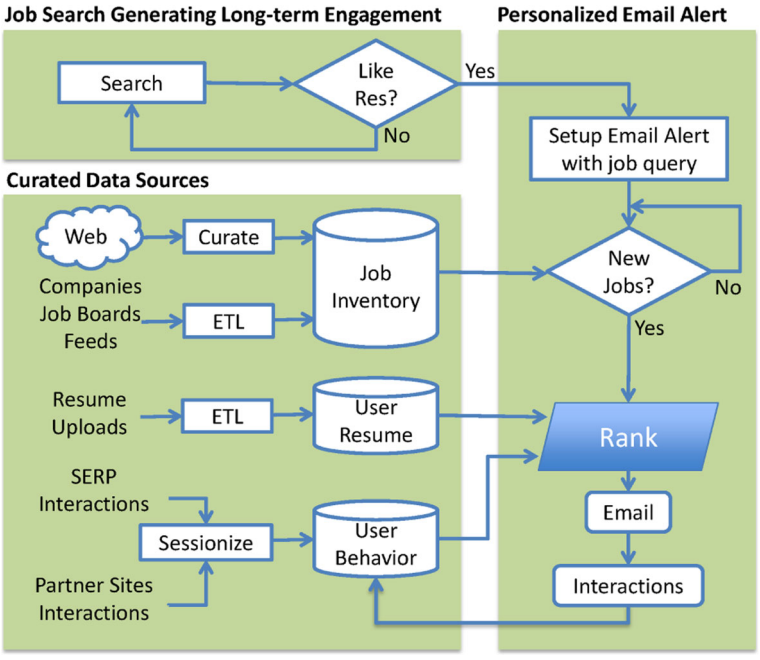

Figure 2 Architecture of a real-world personalized job recommender

online job postings based on more than 10 million jobs postings that it aggregates from various sources to satisfy more than 30 million unique visitors from 24 countries every month.

Figure 2 illustrates the architecture of the personalized job recommender. When job seekers like the search results they are seeing, they often opt to receive further emails regarding similar opportunities. The creation of email alerts allows the website to engage with the job seeker over longer periods of time, providing the opportunity for the website to learn from user behavior to produce more relevant personalized job recommendations. A significant portion of the engagement on the *Simply Hired* website is driven by email alerts that inform the job seekers about similar job openings on a daily or weekly basis. A number of curated data sources are used in this process. The curated job inventory is aggregated from a number of job sources:

- Company websites: Automated crawlers roam on the Web to collect and curate employment related sites and extract job postings into semi-structured records with fields such as job title, company name, normalized location, and job category classification.
- Company and Job Board feeds: some companies and job boards send feeds of jobs to the job aggregator in exchange for the traffic they bring to the company and job board pages.

A job seeker also has the option to upload their resumes to provide more information about their existing roles, skill sets and interests for the personalized job recommender. These key categories of information are extracted automatically from the resumes. Historic user behavior is also a powerful source of information that provide insights from a collaborative filtering perspective. These information can be extracted from job seekers' behavior with search engine result page (SERP) as well as partner websites. The interactions are analyzed from both the click level as well as the session level.

With the availability of long-term engagements generated from job searches, and curated data sources such as job inventory, user resumes, and user behavior, we have a platform for personalized job recommendation. In producing an email alert, we first check to see if there are any new relevant jobs that has been curated into the job inventory database since the last sent email alert. If there is, the personalized job recommender ranks the relevant job postings according to descending order, such that the most relevant job posting is listed on top. The resulting ranked list is then sent out as an email alert. User interactions with the sent email alerts are recorded and becomes job seeker behavior logs that are used for further personalization.

This paper focuses on the ranking component which scores jobs by the click model to estimate the probability they will be clicked by the email alert subscriber. This estimate is crucial for job search engines as it influences user experience and revenue. A simple measure of the clickability is the content similarity between job description and user query. In this paper, we follow a machine learning approach by collecting click and non-click events from logs as training samples [8], where each sample represents a user-job pair with a number of user-independent and user-specific features that indicate the job clickability. Section 4.2 introduces these features in detail.

4 Data collection and features

4.1 Data collection

We created a testbed from *Simply Hired*'s databases to study the characteristics of the data and evaluate click prediction. We collected job and user interaction information of email

alerts from July 11th − October 19th, 2013. All personally identifiable information were removed from the dataset. We use the current data and historical data as follows.

- Current data: We use the anonymized user behavior on October 19, 2013 as current data, which serves as labels for observable user click behavior. To avoid inaccuracies and ambiguities caused by accounting for email clients that do not load images by default, or pre-caches images by default, we base our analysis only on emails that have generated at least one click, assessing the job postings that received clicks as positive samples, and the impressed but un-clicked job postings as negative examples.
- Window data: We collect five days' (Oct 14-Oct 18) user behaviors in the same way as we collect for current data. We collect it day by day and make an integrations. We collect window data in order to make models (See Section 6)
- Historical data: To personalize job recommendation, we build user profiles with features extracted from the 75 days prior to the specific date. We focus personalized job recommendation on job seekers who have produced at least one click in the past 75 days, and have engagement with us to the specific date. History data is a relative concept. For example, for Oct 19th, 2013, the historical data would be from Aug 5th to Oct 18th. However, for October 18th, history date would be from Aug 4th to October 17th. Since Historical data will be utilized to extract user features, which are introduced in section 4.1.2. It is important to distinguish historical data with window data though they have override.

The testbed includes a sample of 167,965 users and 1,231,547 jobs with a total of 1,944,673 user-job view events.

4.2 Features

An accurate machine learning based click model relies on the careful design of features. In this work, we derive the features from the attributes which contain important information about job, users, and their behavior. Table 1 includes the descriptions and examples of the attributes. We can generate both user-independent and user-specific features from these attributes as follows. Table 2 shows the list of the features.

Table 1 Attributes used to generate features

Attribute	Description	Example
Job title	The job title associated with each job	Software Engineer
Company name	The company name associated with each job	Google
CBSA	Job location associated with each job	Minneapolis, MN
ONET	A code which indicates job category	"53-3032.00" for trailer-truck driver
Source type	Primary or job boards as job source type	job_board
Source name	Direct company, staffing agency or job board names as job source	Experteer
Crawler ID	The crawler used for locating the job posting	121
Doc score	The relevancy between job description and user query	0.1680
Query	The search keywords that the user used to create an email alert	police officer

Table 2 22 features used in the click models

14 User-independent features
Doc score, company, CBSA, ONET, crawler id, job source type, job source name, query,
job title & company, query & company, job title, CBSA & job title, CBSA & company, query & job title
8 User-specific features (based on user's click history)
company, job location, ONET, job title & query, CBSA, job source type, job source name, query

4.2.1 Job features

Each job has different attributes such as a company name, a job title, a location, a job source, a job category, and so on. Each attribute has 1,000 - 100,000 categories. For example, job location is represented by 1098 CBSA (Core-Based Statistical Area) as defined by United States Office of Management and Budget;[5] job category is represented by 1,110 ONET (Occupational Information Network) code;[6] job source is represented by a sample of 7,000 *Simply Hired* crawlers that are indexing jobs from around the Web.

In this study, we converted thousands of categorical attributes to numerical feature values. We performed Bayesian analysis on each attribute to compute the conditional probability a click given an attribute based on job seeker behavior on *Simply Hired*, specifically as follows:

$$jobfeature = P(click|attribute) - P(click) \tag{1}$$

where $P(click|attribute)$ is calculated as the number of clicks on the attribute divided by the total number impressions of the attributes. $P(click)$ is the total number of clicks divided by the total number of impressions. Equation (1) can capture the degree of bias of the given attribute. For example, if $P(click|\text{“}SanJose\text{”}) = 0.05$ and $P(click) = 0.04$, it shows that the click on “San Jose” is more likely to happen, which may indicate the popularity of the job location. Table 3 contains some examples of CBSA locations and their corresponding feature values calculated based on (1).

We use a time window of 14 days to accumulate the clicks and impressions. We also construct features with combined attributes such as job location & company name to capture the interaction between attributes. We consider these combined features because different job popularity may be affected by locations, companies and other factors. For example, job seekers may prefer to work in Microsoft as a software engineer than work in U.S. Bank with the same title. Also, people may prefer New York over San Francisco Bay area for financial industry.

These features quantify job seeker overall preferences between different companies, likely behavior within different geographical regions, and within different job categories. To reduce noise, we only consider the case when there are sufficient observations for the attribute. Through extensive experiments, we found the threshold of 100 impressions could strike a good balance between sufficient samples and the availability of the feature for click prediction. If there are not enough observations to compute the feature, we would assume the feature is 0 denoting the case of missing values. This often happens when job seekers used rare queries or clicked jobs that contained rare company name, title or job source. Section 5.1 investigates the missing data issue in detail.

[5] http://www.whitehouse.gov/sites/default/files/omb/bulletins/2013/b-13-01.pdf

[6] http://www.onetonline.org

Table 3 Examples of CBSA and the corresponding feature values calculated based on (1)

CBSA	Location	Feature Value
45020	Sweetwater, TX	0.0146
11820	Astoria, OR	0.0002
26940	Indianola, MS	−0.0231
31620	Magnolia, AR	−0.0247

4.2.2 User features

For personalization, we extract personalization features (See Table 2, 8 User-specific features) based on user engagement history in the past 75 days, as our statistics show that more than 65% of users have click history in the preceding 75 days. The companies, job titles, queries, job locations and job categories in a users click history are used to infer their background and job search preference. The user features are extracted differently depending on their types.

As mentioned, historical data corresponds to a current day's data. When we extract user-specific features, we would examine this users' current behavior has been ever appears in historical data sets. For example, in current data, we find a user with ID 10001 has a clicked a job title "software engineer" on current date, Oct19th, so when we extract feature, we would search all his clicked jobs. If he has ever clicked a job with title "software engineer", we would give higher value.

Historical features will be extracted as a probabilistic calculation. The formula would be $feature = \frac{score}{\texttt{total\_occurance}}$. Score would be assessed in different way which would be described later. Total occurrence means that the total number of clicks made by the user in past 75 days.

Here are the methods to compute the score by data types:

For company locations, we assess similarity by exact match. Specifically, if there is any difference in the location values, the feature would get 0; otherwise, it would be 1, since different locations indicate different preferences. Similarly, we apply the same method to the attributes of company name, historical job source, historical job source, and CBSA.

For job titles and queries, we compute the cosine similarity between current and historical attribute values. These values are often in long strings and thus it is unlikely to have exact match. On the other hand, partial match may indicate similar job functions, e.g., "Software Engineer vs "Software Engineer II". Thus, we treat each string as a vector and apply the vector space model with term frequency [22] as the weighting scheme to compute their similarities.

For ONET category, we use a hierarchical match based on the tree-based taxonomy that ONET classification code is designed with. Specifically, each ONET code is interpreted as three levels marked by "-" and ".". For example, the ONET code of 15-1133.00 for software engineer is grouped into "15", "15-1133", and "15-1133.00". The match is scored by the degree of partial match of the code. We assign scores based on the levels of hierarchy. For the first level, we assign 0.5 if we have a match. For the second and third, we assign each of them as 0.25 since they are not as important as the first hierarchy.

4.2.3 Query document features

Query Document Features correspond to Table 2's doc score in user independent feature. We introduce it separately because we have different method in extraction. For each job in

the email alerts, we also computed a contextual relevance score, based on similarity between job and email alert query with boosts and penalties from location proximity, job age, and other features. The range of contextual relevance score can be affected by the length of the query, which can differ for different email alert query. We normalized contextual relevance score for all jobs in one email alert using the maximum contextual relevance score in that email alert. Normalized Doc score was used as a feature in our models.

5 Probabilistic clustering and prediction model

5.1 Motivation

Based on the data collected, we find that 46.5% of feature values were missing. We plot the percentage of missing values in each individual feature in Figure 3. As we can see, 9 out of the 22 features have more than 70% of missing values and only 4 features have less than 10% of missing values. We also calculate the percentage of missing features for each job seeker and plot the histogram (normalized by the total number of job seekers) in Figure 4. The histogram shows that more than 18% of the job seekers have around 45% of missing features. The majority of the users have around 40% - 60% of missing features. These two figures indicate that there exist a lot of missing values in the features, which could be caused by various reasons in the real-world context.

We further investigate the missing features vs observed features by analyzing the patterns of the features that are missing. Specifically, we randomly choose 10,000 instances and transform the original instance-feature matrix to a binary matrix in which 1 indicates the corresponding entry is observed and 0 means missing. Principal Component Analysis (PCA) is then applied to reduce the dimensionality of the instances. We plot the instances in the space of the first two principal components in Figure 5. As we can see, these instances seem to form clusters. The experiments in Section 6.4.4 demonstrate that these clusters

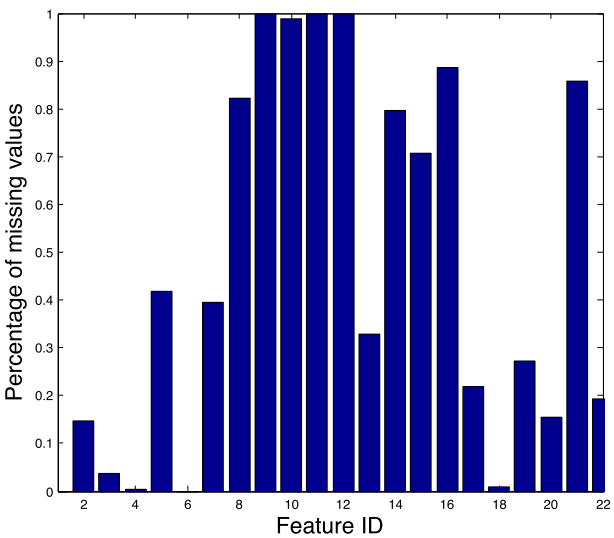

Figure 3 The percentage of missing values in each individual feature. The feature IDs are based on the orders in Table 2

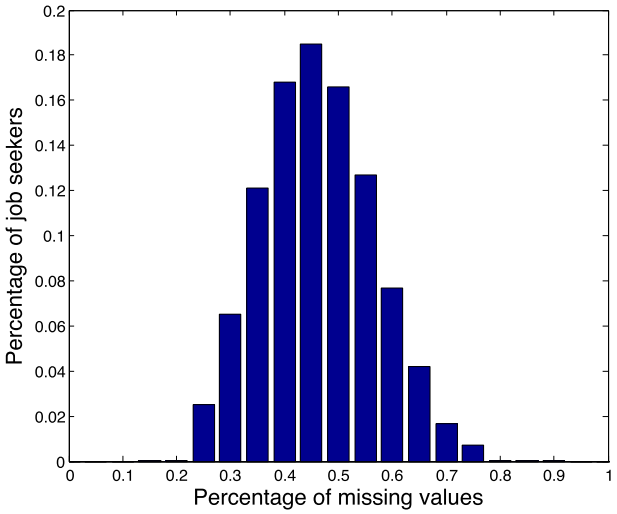

Figure 4 The histogram of the percentage of missing features for each job seeker

characterize different types of jobs and job seekers. Thus, training one flat machine learning model may not be able to capture a wide variety of jobs and job seekers. To capture the multimodality of the job seekers is especially crucial to provide personalized job recommendations. It is worth noting that these clusters are derived from the missing features. In other words, the patterns of missing values can be leveraged to extract valuable information instead of being discarded by many prior methods.

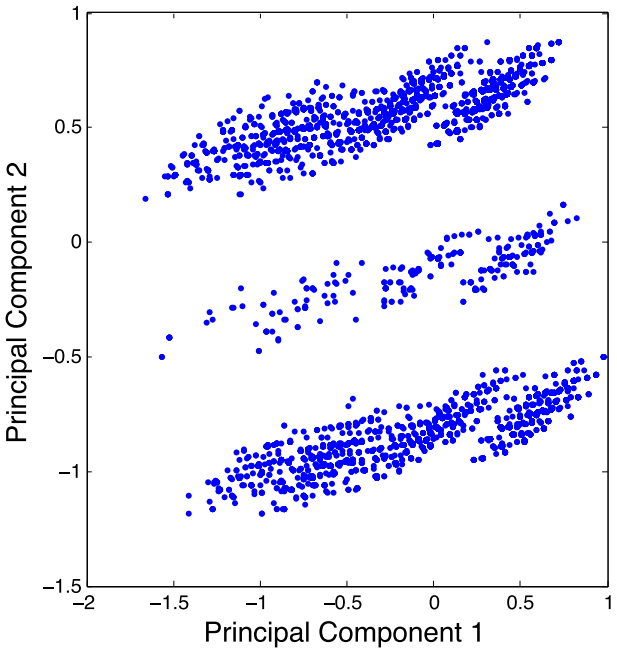

Figure 5 10,000 user-job instances projected to the space of the first two principal components by PCA

5.2 The PCP Model

In this section, we present a unified probabilistic model that can capture various types of jobs and job seekers and meanwhile learn personalized prediction strategies for the job seekers. Formally, given the pair (u, b) for user u and job b, our goal is to estimate the conditional probability $P(c|(u, b))$ that u will click on b. c is a binary click variable $c \in \{1, -1\}$ where 1 means click and -1 is non-click. A conventional click prediction approach would train a single machine learning model such as logistic regression as exemplified in the prior work [8] based on the observed features. As shown in Section 5.1, job seekers demonstrate a variety of clicking behaviors and a single model is not flexible enough to accommodate the multimodality of users and jobs. Thus, we assume that there are a total of K types of user-job (u, b) pairs and $P(c|(u, b))$ can then be decomposed as follows by probability chain rule and marginalization:

$$P\left(c|(u,b)\right) = \sum_{z=1}^{K} P\left(z|(u,b)\right) P\left(c|(u,b),z\right) \tag{2}$$

where z denotes the latent type of user-job pair (u, b) and $P(z|(u, b))$ measures the probability that (u, b) belongs to the type z. It is noticeable that this is a soft version of categorization which leads to a probabilistic membership assignment of (u, b) to latent types. As shown in Section 5.1, the types of users and jobs can be determined by how the observed features are missing. Specifically, we model $P(z|(u, b))$ by a soft-max function $\frac{1}{Z_{(u,b)}} \exp\left(\sum_{i=1}^{N_m} \alpha_{zi} m_{i,(u,b)}\right)$ where $m_{i,(u,b)} \in \{0, 1\}$ is the i^{th} missing pattern feature for (u, b) and α_{zi} is the weight associated with the feature for type z. N_m is the total number of missing pattern features and $Z_{(u,b)}$ is the normalization constant $Z_{(u,b)} = \sum_{z=1}^{K} \exp\left(\sum_{i=1}^{N_m} \alpha_{zi} m_{i,(u,b)}\right)$ to ensure $P(z|(u, b))$ is a proper probability. Moreover, $P(c|(u, b), z)$ measures the probability of click/non-click given (u, b) *and* the type z. It can be modeled by logistic regression as follows:

$$P\left(c|(u,b),z\right) = S\left(c \sum_{j=1}^{N_f} \beta_{zj} f_{j,(ub)}\right) \tag{3}$$

where $S(\cdot)$ is the sigmoid function defined as $S(x) = \frac{1}{1+\exp(-x)}$. $f_{j,(ub)}$ is the j^{th} user-job feature for (u, b) and β_{zj} is the weight associated with the feature. It is worth noting that β_{zj} depends on type z. In other words, for each type of missing patterns, there will be a corresponding logistic regression model and different types will have different prediction models trained for them. The proposed model can be viewed as a prediction process as follows: given the missing pattern features $m_{\cdot,(u,b)}$ for the user-job (u, b), we first determine the type z of (u, b), and then use the logistic regression model for z to predict the click variable c. The proposed model accomplishes this joint clustering-classification process by maximizing the following data log likelihood function with respect to the parameters α and β. We denote it as the probabilistic clustering-prediction (PCP) model.

$$\begin{aligned} L(\alpha,\beta) &= \sum_{(u,b)} \log\left(\sum_{z=1}^{K} P\left(z|(u,b)\right) P\left(c_{(u,b)}|(u,b),z\right)\right) \\ &= \sum_{(u,b)} \log\left(\sum_{z=1}^{K}\left(\frac{1}{Z_{(u,b)}} \exp\left(\sum_{i=1}^{N_m} \alpha_{zi} m_{i,(u,b)}\right)\right) \times S\left(c_{(u,b)} \sum_{j=1}^{N_f} \beta_{zj} f_{j,(u,b)}\right)\right) \end{aligned} \tag{4}$$

We can use the EM algorithm [4] to estimate the parameters. The E-step can be derived as follows by computing the posterior probability of z given (u, b) and the observed click or non-click $c_{(u,b)}$:

$$P\left(z|(u,b),c_{(u,b)}\right) = \frac{\exp\left(\sum_{i=1}^{N_m}\alpha_{zi}m_{i,(u,b)}\right) S\left(c_{(u,b)}\sum_{j=1}^{N_f}\beta_{zj}f_{j,(u,b)}\right)}{\sum_{z=1}^{K}\exp\left(\sum_{i=1}^{N_m}\alpha_{zi}m_{i,(u,b)}\right) S\left(c_{(u,b)}\sum_{j=1}^{N_f}\beta_{zj}f_{j,(u,b)}\right)} \tag{5}$$

By optimizing the auxiliary function [4], we can derive the following M-step update rules,

$$\alpha_{z\cdot}^{*} = \arg\max_{\alpha_{z\cdot}} \sum_{(u,b)} P\left(z|(u,b),c_{(u,b)}\right)\log\left(\frac{1}{Z_{(u,b)}}\exp\left(\sum_{i=1}^{N_m}\alpha_{zi}m_{i,(u,b)}\right)\right) \tag{6}$$

$$\beta_{z\cdot}^{*} = \arg\max_{\beta_{z\cdot}} \sum_{(u,b)} P\left(z|(u,b),c_{(u,b)}\right)\log\left(S\left(c_{(u,b)}\sum_{j=1}^{N_f}\beta_{zj}f_{j,(u,b)}\right)\right) \tag{7}$$

The above M-step can be efficiently optimized by the L-BFGS method [25]. The number of latent clusters K can be obtained by maximizing the sum of log-likelihood and some model selection criteria. In the experiments, we choose Bayesian information criterion (BIC) [29], which is a measure of the goodness of fit of an estimated statistical model, defined as $\max 2L - r\log(K)$ where r is the number of parameters in the statistical model. The pseudocode of the PCP algorithm is shown in Algorithm 1. The computational complexity of the algorithm is $O(KN_uN_b\max\{N_m, N_f\}N_{iter})$ where N_u is the number of users, N_b is the number of jobs, and N_{iter} is the maximum number of iterations specified in the EM algorithm.

PCP can be conceptually viewed as a mixture of logistic regression models [4] but with the mixture weights parametrized by a soft-max function over missing features. It is also similar to the expert and query dependent probabilistic models for expert search [38]. PCP can go beyond logistic regression and use any learning to rank model as the prediction model in the clustering-prediction process. The parameter estimation would be similar for other prediction models while the exact derivation of EM algorithm would be different.

Algorithm 1 Probabilistic Clustering and Prediction (PCP) Model

Input: Feature values $m_{i,(u,b)}$, $f_{j,(u,b)}$ and click c for all the user-job pairs (u, b), tolerance value for convergence, maximum number of iterations N_{iter}
Output: Parameters α_{zi}, β_{zj}

1: **procedure** EXPECTATION-MAXIMIZATION
2: Initialize the parameters by the standard normal distribution $\alpha_{zi} \leftarrow \mathcal{N}(0, 1)$,
3: $\beta_{zj} \leftarrow \mathcal{N}(0, 1)$
4: Do
5: Perform the Expectation step using (5)
6: Perform the Maximization step using (6)
7: Until the difference of the log likelihood estimates in (4) between the current iteration and the last iteration is less than tolerance value OR the number of iterations exceeds the maximum

6 Experiments

6.1 Setup

The data used in the experiments were sampled from the *Simply Hired* traffic logs of user-job interaction in email alerts in the period of 75 days as shown in Section 4.1. Each sample of the data is a user-job view event labeled as click or non-click by a subscribed user of the *Simply Hired*'s email alert service for job recommendation. As we mentioned in Section 4, the data is categorized as current data, window data and historical data. After we extract features for current data and window data (Note: historical data itself is utilized to extract features, for window data, we also need to extract historical features). We set current data as testing data by hide the click fact. We integrate the window data as training data and we will build model upon that. Overall, the data have 1,964,773 user-job view events, which includes 1,695,896's training data. We adopt the evaluation metrics that were used for click prediction in sponsored search: precision-recall (P-R) curves and area under curve metric (AUC) [8]. The precision is defined as the number of user-job pairs clicked on by users divided by the total number of user-job pairs labeled as click by the model, and recall is defined as the number of user-job pairs labeled as click by the model divided by the total number of actually clicked user-job pairs. In the experiments, we also show precision at various recall levels ranging from 0.025 to 0.2. This range is chosen based on the TREC evaluation guideline which specifies that the range of recall from 0 to 0.2 in Precision-Recall curve characterizes high precision performance [13]. High precision performance is important for job recommendation since we can only recommend a limited number of jobs for a particular user.

In the experiments, we compare the performance of the proposed probabilistic clustering-prediction (PCP) model with logistic regression. As pointed out in Section 5, any learning to rank model can be plugged into the clustering-prediction process as the prediction model. As the goal of our modeling is to demonstrate the usefulness of missing features in click prediction, we leave the development of other similar models in the future work. On the other hand, we do use the state-of-the-art learning to rank models as the baselines for comparison. We also investigate the effect of various features in click prediction (e.g., personalized vs non-personalized).

6.2 Baseline methods

To compare with our proposed PCP model, we have applied five different learning to rank (L2R) models. In this section, we will make a brief description of each of the state-of-art learning models. Learning to rank models [19] has been widely used in many applications in information retrieval. It is usually referred as machine learning techniques for training models to solve a ranking problem. Originally, it has been popular in applications of ranking documents. In our work, it could be effectively rank the job postings in a descending order, which definitely meets the purpose of job recommendations. Below are the five L2R baseline methods. In our experiments, we use their implementations in RankLib[7] which is a popular library of learning to rank algorithms.

– **Coordinate Ascent** [24]. It is a list-wise linear model for information retrieval which uses coordinate ascent to optimize the model's parameters. When optimizing the loss

[7] https://sourceforge.net/p/lemur/wiki/RankLib/

function, it sequentially doing optimization in one dimension at a time. It cycles through each parameter and optimizes over it while fixing all the others.

- **RankBoost** [11]. It is a pair-wise boosting technique for ranking. Training proceeds in rounds. It starts with all document pairs being assigned with an equal weight. At each round, the learner selects the weak ranker that achieves the smallest pair-wise loss on the training data with respect to the current weight distribution. Pairs that are correctly ranked have their weight decreased and those that are incorrectly ranked have their weight increased so that the learner will focus more on the hard samples in the next round. The final model is essentially a linear combination of weak rankers.
- **ListNet** [7]. ListNet utilizes a listwise ranking loss based on the probability distribution on permutations. Specifically, it first defines the permutation probability distribution based on the scores given by a scoring function. Then it defines another permutation probability distribution based on the ground truth label. For the next step, ListNet uses the KL divergence between these two distributions to define its listwise ranking loss. A neural network is then used as the model and gradient descent as the optimization algorithm to learn the scoring function.
- **RankNet** [6]. RankNet is probably the first learning-to-rank algorithm used by commercial search engines [19]. In RankNet, the loss function is defined on a pair of documents, but the hypothesis is defined with the use of a scoring function. Given two documents associated with a training query, a target probability is constructed based on their ground truth labels. Then, the modeled probability Pu,v is defined based on the difference between the scores of these two documents given by the scoring function. Similar to ListNet, A neural network is used as the model.
- **AdaRank** [35]. The idea of AdaRank is similar to that of RankBoost except that it is a list-wise approach. Hence, it directly maximizes any desired IR metric such as NDCG and MAP whereas RankBoost's objective is to minimize the pair-wise loss.

6.3 Research questions

A set of experiments are designed to address the following questions of the proposed research:

- Can machine learning based models with a multitude of features outperform the content based approach with a single feature? (Section 6.4.1)
- Can the proposed PCP model improve over a flat machine learning model such as logistic regression by leveraging the missing features? (Section 6.4.1)
- How does PCP perform compared to the state-of-the-art learning to rank models? (Section 6.4.2)
- What is the effect of the personalization features in predicting user clicks? (Section 6.4.3)
- Do the clusters learned by PCP correspond to various types of jobs and job seekers in the real world? Would the learned prediction models be different for different clusters? (Section 6.4.4)

6.4 Results

6.4.1 PCP vs logistic regression vs non-machine learning based approach

In this section, we compare machine learning based approach with content matching based approach. Specifically, we train a logistic regression model on all the 22 features shown in

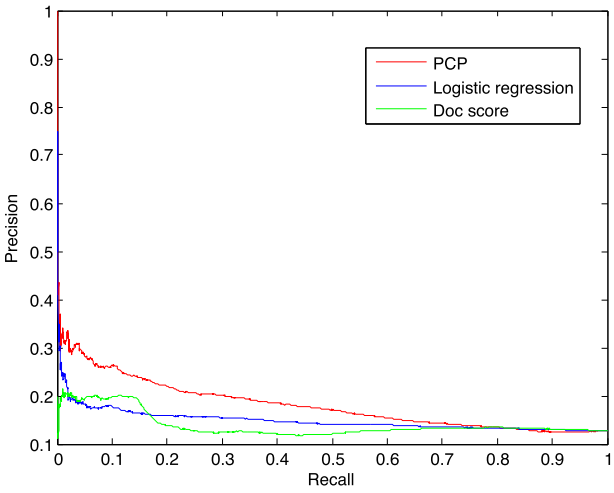

Figure 6 Precision-Recall curves for PCP, Logistic regression, and Doc score

Section 4.2 and compare it with the method that only uses the document relevance score (Doc score). Figure 6 plots the Precision-Recall curve and Table 4 shows the corresponding AUC scores. As we can see, logistic regression can gain 5.7% improvement over Doc score in AUC. We also show the results of the proposed PCP model in Figure 6 and Table 4. PCP has improved 22.3% in AUC over the baseline and substantially outperforms logistic regression. As pointed out in Section 6.1, PCP consists of multiple logistic regression models for various types of users and jobs. The substantial performance gain by PCP over logistic regression demonstrates the effectiveness of utilizing the patterns of missing features to cluster user-job pairs.

The Precision-Recall curve in Figure 6 demonstrates that PCP generally achieves much higher precision than the baselines when recall is small. Table 5 further shows the precision of the three approaches at various recall levels ranging from 0.025 to 0.2. This range characterizes high precision performance as specified in the TREC evaluation guideline [13], which is important for the job recommendation application in this paper. As we can see in the table, the gap in precision between PCP and logistic regression generally becomes larger as recall goes smaller. The three methods shows very similar performance when recall level is greater than 0.8.

6.4.2 PCP vs learning to rank

We further compare PCP with the state-of-the-art learning to rank (L2R) models. Specifically, five L2R models are chosen due to their effectiveness in various ranking tasks [19]. These five models cover the full spectrum of L2R models, i.e., pointwise: coordinate ascent

Table 4 AUC of PCP, logistic regression, and document relevance score (Doc score)

Doc score	Logistic Regression	PCP
0.5161	0.5455 (+5.7%)	0.6314 (+22.3%)

The numbers in the parenthesis show the relative improvement over the baseline of Doc score

Table 5 Precision of PCP, Logistic regression (LR), and Doc score at various recall levels ranging from 0.025 to 0.2

	Recall level							
	@0.025	@0.050	@0.075	@0.100	@0.125	@0.150	@0.175	@0.200
PCP	0.2989	0.2836	0.2597	0.2625	0.2461	0.2376	0.2274	0.2203
Doc score	0.2023	0.1944	0.1971	0.1970	0.1978	0.1863	0.1537	0.1387
LR	0.1911	0.1816	0.1740	0.1785	0.1687	0.1643	0.1601	0.1595

(CA); pairwise: RankNet and RankBoost; listwise: AdaRank and ListNet. Figure 7 plots the Precision-Recall curve and Table 6 shows the corresponding AUC scores. As we can see, PCP outperforms all the L2R models in AUC. These results are encouraging since we only use logistic regression as the prediction model in PCP. In the future work, we will explore the state-of-the-art L2R models in PCP and the performance could get further improved. Among the various learning to rank models, only RankBoost yields comparable performance with PCP. AdaRank and ListNet generate the lowest AUC scores, which indicates that listwise L2R methods may not be suitable for this task.

Table 7 contains the results of the high precision region. In general, PCP, RankBoost, and Coordinate Ascent achieve higher precision when recall level is lower. The other three L2R methods yield stable but low precision performance over various recall levels. Similar to the pattern shown in Table 5, the gap in precision between PCP and the other methods generally becomes larger as recall goes smaller, which shows the advantage of PCP over L2R in job recommendation.

6.4.3 Effect of user-specific features

In this section, we study the effect of the user-specific features in click prediction. We run PCP on three sets of features: user-independent, user-specific, and all the features. The user-independent features and user-specific features are introduced in Section 4.2.

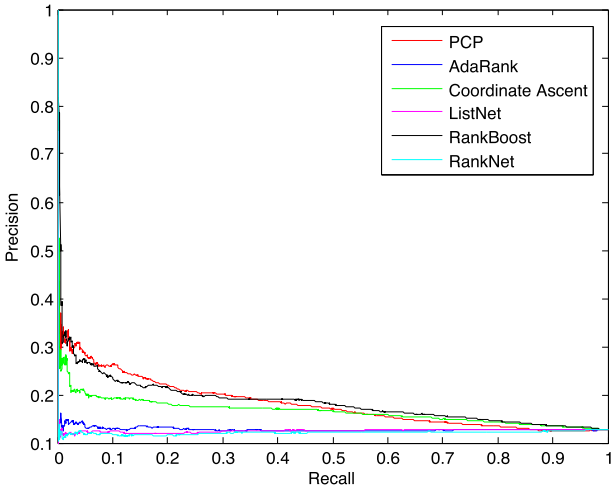

Figure 7 Precision-Recall curves for PCP and various learning to rank models

Table 6 AUC of PCP and various learning to rank models

AdaRank	CA	ListNet	RankBoost	RankNet	PCP
0.493	0.525	0.499	0.491	0.536	0.631

Table 7 Precision of PCP and the state-of-the-art learning to rank methods at various recall levels ranging from 0.025 to 0.2

	Recall level							
	@0.025	@0.050	@0.075	@0.100	@0.125	@0.150	@0.175	@0.200
PCP	0.2989	0.2836	0.2597	0.2625	0.2461	0.2376	0.2274	0.2203
AdaRank	0.1444	0.1341	0.1318	0.1286	0.1264	0.1352	0.1320	0.1322
CA	0.2070	0.2046	0.1944	0.1904	0.1930	0.1900	0.1850	0.1822
ListNet	0.1188	0.1261	0.1246	0.1251	0.1201	0.1194	0.1209	0.1202
RankBoost	0.2892	0.2691	0.2508	0.2355	0.2244	0.2190	0.2205	0.2162
RankNet	0.1186	0.1181	0.1186	0.1177	0.1159	0.1168	0.1151	0.1159

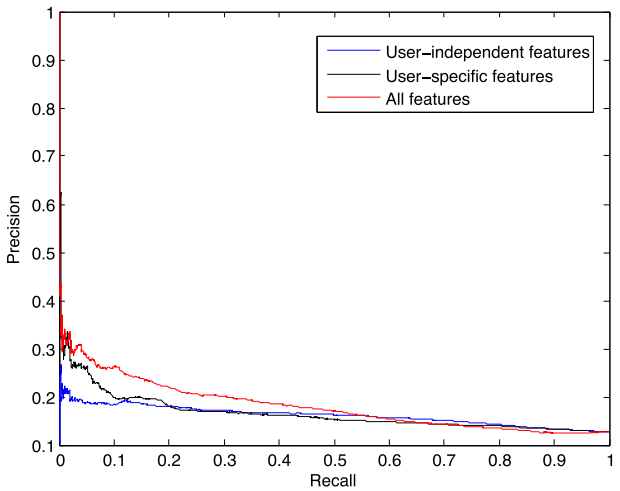

Figure 8 Precision-Recall curves for PCP with various sets of features

Table 8 AUC of PCP with various sets of features

User-independent	User-specific	All
0.574	0.586	0.6314

Table 9 Precision of PCP with various sets of features at various recall levels ranging from 0.025 to 0.2

	Recall level							
	@0.025	@0.050	@0.075	@0.100	@0.125	@0.150	@0.175	@0.200
All	0.2989	0.2836	0.2597	0.2625	0.2461	0.2376	0.2274	0.2203
User-independent	0.1911	0.1874	0.1869	0.1834	0.1896	0.1864	0.1823	0.1805
User-specific	0.2662	0.2600	0.2198	0.2006	0.1964	0.1988	0.1954	0.1803

Figure 8 plots the Precision-Recall curve and Table 8 shows the corresponding AUC scores. Table 9 contains the precision of PCP with various sets of features at various recall levels ranging from 0.025 to 0.2. All these results show that the user-specific features alone yield comparable performance with the user-independent, but if we combine both features, the improvement could be substantial. These results show the effectiveness of considering user-specific features in user click prediction.

6.4.4 Clusters and parameters

We obtain 7 clusters from PCP (on all the features) based on the BIC criterion in Section 5. We investigate the individual clusters and the corresponding prediction models for the clusters. We find these clusters indicate various types of jobs or job seekers. For example, we analyze the users and jobs in Cluster 1 which accounts for 30% of the whole population. We find that the users in this cluster are less likely to click on popular companies. The similar situation happens to the job sources. As for the job locations they have clicked on, we find they do not have any preference since the corresponding feature values are very low. On the other hand, they tend to have a high score on the job_queries and job_query & title features, which indicates that they have a strong preferences on job types. We also analyze Cluster 5 which includes around 22% of the populations. We found that the feature scores on the company name is relatively high, which means that users in this cluster may prefer branded companies. It seems they do not like popular job sources and job queries since they have low values on these features. Similarly, we find they have a strong preference on job types but a weak one on job locations. Based on the similar analysis, the other clusters can also be found to represent certain specific types of jobs and job seekers. We further look at the parameters β_{zj} learned for each cluster z and feature j. We find these weights are very different between clusters, which indicates different prediction models were trained for different clusters and thus leads to increased personalization for job seekers.

7 Conclusion and future work

In this paper, we ntroduce a real-world personalized job recommender system. A learning to rank approach is used to recommend jobs for job seekers based on a set of user-independent and user specific features. We further reveal the cluster structure of the missing features and propose a probabilistic model to learn preferences of various types of users and jobs. We conduct a comprehensive set of experiments to compare the proposed model with the state-of-the-art learning to rank models and demonstrate its effectiveness.

In the future work, we will go beyond logistic regression and use other L2R models as the prediction model in the proposed clustering-prediction process. Consequently, the proposed

models would become pairwise or listwise PCP instead of pointwise PCP presented in this paper. The proposed approach can also be applied to other recommendation tasks in which the data presents multimodality behaviors. Another research direction is to develop more personalization features such as dwell time, query reformulation, and the sequence of clicks, which can be aggregated and exploited in personalized job recommendation.

References

1. Al-Otaibi, S.T., Ykhlef, M.: A survey of job recommender systems. Int. J. Phys. Sci. **7**(29), 5127–5142 (2012)
2. Attenberg, J., Pandey, S., Suel, T.: Modeling and predicting user behavior in sponsored search. In: SIGKDD, pp. 1067–1076. ACM (2009)
3. Balog, K., Yi, F., de Rijke, M., Serdyukov, P., Si, L., et al.: Expertise retrieval. Found. Trends Inf. Retr. **6**(2-3), 127–256 (2012)
4. Bishop, C.M.: Pattern recognition and machine learning, vol. 1. Springer, Berlin (2006)
5. Bradley, K., Rafter, R., Smyth, B.: Case-based user profiling for content personalisation. In: Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 62–72. Springer (2000)
6. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on Machine learning, pp. 89–96. ACM (2005)
7. Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on Machine learning, pp. 129–136. ACM (2007)
8. Cheng, H., Cantú-Paz, E.: Personalized click prediction in sponsored search. In: WSDM, pp. 351–360. ACM (2010)
9. Cheng, Y., Xie, Y., Chen, Z., Agrawal, A., Choudhary, A., Guo, J.S.: A real-time system for mining job-related patterns from social media. In: SIGKDD, pp. 1450–1453. ACM (2013)
10. Färber, F., Weitzel, T., Keim, T.: An automated recommendation approach to selection in personnel recruitment. In: AMCIS, pp. 302. Citeseer (2003)
11. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences. J. Mach. Learn. Res. **4**(Nov), 933–969 (2003)
12. Graepel, T., Candela, J.Q., Borchert, T., Herbrich, R.: Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. In: ICML, pp. 13–20 (2010)
13. Harman, D.K.: The fourth text retrieval conference (TREC-4). National institute of standards and technology (1996)
14. Hutterer, M.: Enhancing a job recommender with implicit user feedback. Fakultät für Informatik, Technischen Universität Wien (2011)
15. Tobias K.: Extending the applicability of recommender systems: a multilayer framework for matching human resources. In: HICSS, pp. 169–169. IEEE (2007)
16. Koren, Y., Sill, J.: Ordrec: an ordinal model for predicting personalized item rating distributions. In: RecSys, pp. 117–124. ACM (2011)
17. Lee, D.H., Brusilovsky, P.: Fighting information overflow with personalized comprehensive information access: a proactive job recommender. In: ICAS07, pp. 21–21. IEEE (2007)
18. Lee, D.H., Brusilovsky, P.: Reinforcing recommendation using implicit negative feedback. In: User Modeling, Adaptation, and Personalization, pp. 422–427. Springer (2009)
19. Liu, T.-Y.: Learning to rank for information retrieval. Found. Trends Inf. Retr. **3**(3), 225–331 (2009)
20. Ma, H., King, I., Lyu, M.R.: Effective missing data prediction for collaborative filtering. In: SIGIR, pp. 39–46. ACM (2007)
21. Malinowski, J., Keim, T., Wendt, O., Weitzel, T.: Matching people and jobs: A bilateral recommendation approach. In: HICSS, volume 6, pp. 137–145. IEEE (2006)
22. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to information retrieval. Cambridge University Press, Cambridge (2008)
23. Marlin, B., Zemel, R.S., Roweis, S., Slaney, M.: Collaborative filtering and the missing at random assumption. ICML (2012)
24. Metzler, D., Bruce Croft, W.: Linear feature-based models for information retrieval. Inf. Retr. **10**(3), 257–274 (2007)

25. Nocedal, J.: Updating quasi-newton matrices with limited storage. Math. Comput. **35**(151), 773–782 (1980)
26. Paparrizos, I., Barla Cambazoglu, B., Gionis, A.: Machine learned job recommendation. In: RecSys, pp. 325–328. ACM (2011)
27. Rafter, R., Smyth, B.: Passive profiling from server logs in an online recruitment environment. In: ITWP, pp. 35–41 (2001)
28. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: UAI, pp. 452–461 (2009)
29. Schwarz, G., et al: Estimating the dimension of a model. Ann. Stat. **6**(2), 461–464 (1978)
30. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., Oliver, N.: Tfmap: optimizing map for top-n context-aware recommendation. In: SIGIR, pp. 155–164. ACM (2012)
31. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In: RecSys, pp. 139–146. ACM (2012b)
32. Singh, A., Rose, C., Visweswariah, K., Chenthamarakshan, V., Kambhatla, N.: Prospect: a system for screening candidates for recruitment. In: CIKM, pp. 659–668. ACM (2010)
33. Wang, J., Yi, Z., Posse, C., Bhasin, A.: Is it time for a career switch. In WWW, pp. 1377–1388 (2013a)
34. Wang, T., Bian, J., Liu, S., Zhang, Y., Liu, T.-Y.: Psychological advertising: exploring user psychology for click prediction in sponsored search. In: SIGKDD, pp. 563–571. ACM (2013b)
35. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 391–398. ACM (2007)
36. Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yong, Y., Chen, Z.: Scalable collaborative filtering using cluster-based smoothing. In: SIGIR, pp. 114–121. ACM (2005)
37. Yu, H.T., Liu, C.R., Zhang, F.Z.: Reciprocal recommendation algorithm for the field of recruitment. Int. J. Inf. Comput. Sci. **8**(16), 4061–4068 (2011)
38. Yi, F., Si, L., Mathur, A.P.: Discriminative probabilistic models for expert search in heterogeneous information sources. Inf. Retr. **14**(2), 158–177 (2011)
39. Yao, L., Helou, S.E., Gillet, D.: A recommender system for job seeking and recruiting website. In: WWW, pp. 963–966 (2013)