Predicting User Reaction to Recommended Items in Job Advertisement

Elie G MAGAMBO † Hayato YAMANA †

† Graduate School of Fundamental Science and Engineering, Waseda University, 3-4-1 Okubo, Shinjuku-ku Tokyo, Japan † Faculty of Science and Engineering, Waseda University, 3-4-1 Okubo, Shinjuku-ku Tokyo, Japan

E-mail: † {eliemagambo ,yamana}@yama.info.waseda.ac.jp

Abstract: Job searching services provide suitable job items to their users using recommender systems. The recommender systems exploit users' information from their profiles, match them with job items' details and calculate similarity between the users and job items to be recommended. Measuring the usefulness of a recommender system can be cumbersome; however analyzing the actions the user took towards the recommenced job items can provide valuable information. The information gathered can help in improving the quality of future recommended job items. In this research, we create personalized feedback features that capture the relevancy of the recommended job items by the system to users by learning their past post-click actions among bookmark, reply and delete that users performed in the past to job items that are similar to the recommended ones. Using those features, we predict future post-click users' actions that they will perform on newly recommended job items using SVM and feed forward neural networks. We do a comparative analysis between both predictors when personalized feedback features are used and when they are omitted. We saw an improvement of 14% in prediction accuracy when personalized feedback features are used compare to when they are omitted, however all predictors guaranteed same accuracy in both cases.

Keywords: Recommender systems, classifier, personalization, feedback features, neural networks.

1. INTRODUCTION

Online job searching services serve as recruiting platforms for companies looking for new employees and also as job market places for people looking for new jobs. A typical use case would start with a given company wishing to hire new manager, engineers, etc. The advantage of using the platform for the company is to reach a wide audience of competent candidates around the world.

The company posts those job-openings to the platform such as Xing[1] hoping that suitable candidates will reply to its' job offers.

On the other hand, candidates registered on the platform as job seekers would later on browse those job-item-openings depending on how they seem to match their interests. Consequently, the job offers are published at high rates on those recruiting platforms, which make it hard for users to go back and forth through the job posted. We have to note that since job offers have expiration dates, it's crucial to provide a way to organize those job items that are suitable for the users. A common challenge in job advertisement is that users aren't always looking for jobs every day, which makes it even more important to take into account granular details about the actions the users undertook during their active window of job searching.

Recommender systems are common solutions to such challenge. The main objective of recommender systems in job advertisement is to find suitable job items and recommend them to users in a ranked list. They also take into account of users' reactions toward recommended job items so that they can provide him high match users' interest.

Among Common problem recommender systems face is cold start [1, 2] where users haven't interacted with any items, or just joined the platform. It has been subjected to many researches

[1] http://xing.com Job advertisement web service

and exploiting information from users' profiles details has been useful in solving it. Another problem is serendipity [3] where a recommender system provides users with items that are not useful by populist models but turns out to be useful to users.

The focus of this research however, considers that the user has already clicked on one of the recommended items and has a possibility to take a set of different actions towards the item. The targeted actions are bookmark, reply and delete. In short, the problem we are trying to address can be stated as follow: "*Given that a user has clicked on a recommended item, is he going to bookmark it, reply it or delete it from the recommended list?"*

Solving this problem helps in making meaningful job recommendation to users and more importantly, filtering the unwanted ones. The remainder of this paper is divided as follow: Section 2 covers related works that have been done in similar context and contrast this research with them, Section 3 covers our approach. Section 4 discusses about the experiment settings, results and section 5 concludes this research and provides recommendations and future work.

2. RELATED WORK

We use personalized feedback features to extract aggregated information about how often the user has performed an action on certain concepts from the titles, tags and other attributes.

Chakrabarti et al. [4] introduced interaction feedback features that capture ad-placement on web pages as ad relevance feedback. For example, word w in certain region r of web page p has a popularity score $t_{p(r)w}$. then, the total score of region r of web page p is computed as follow[4]:

$$M_{p(r)w} = 1\big(w \epsilon\, p(r)\big).t_{p(r)w} \quad (1)$$

If word w is not present in region r of web page p, the feature contribution becomes zero.

In the same context, we also adopt the above features to capture user tendency toward a certain action. In contrast with [4], we capture user interaction feedback from certain concepts presented in the title and tags of a recommended job item. They are personalized in a sense because they are user dependent. As the user has different choice among action to undertake, we extend these features to every action among bookmark, reply and delete.

Chapelle et al. [5] defined conversion estimation task as the probability that a user will convert after they click. Conversion could be subscribing to mailing list, making a reservation or purchasing a product. In our case, post click actions are limited and are the same for every job items whereas in [5], individual advertiser predefines the actions the user will perform as success of their advertisement and the scope of [5] was limited on predicting if any conversion will happen or not after the click.

Pacuk et al. [6] worked on job recommendation problem. They selected candidate jobs that might be suitable for a user depending on title and tags similar to those he has clicked upon before and ranked them according their probability to be clicked by the user. Although the context is similar to our work, the tasks differ in that they limited their work on predicting if the user will click or not whereas for us we assume that the user has already clicked and we are trying to predict if the user will bookmark, reply or delete.

Another work in same context was done in [7].They used Gradient boosted trees as weak learners that estimate probabilities for every recommended job item. Additionally, they used hawk process [8] to model time decay of recommended item since it has been initially created on the platform. Finally, they make an ensemble model that combines the probabilities estimation from weak learners as a final score for each recommended item. Although this work had better results, it also limits its scope on predicting if user will click or not click on the recommended item.

3. METHODOLOGY

We adopt personalized feedback features [4] between the recommended item and the user and personalize them to the user by binding them to actions to express his tendency to perform one among bookmark, reply or delete actions. We achieve this goal by aggregating action related past activities on items with similar attributes

values as the recommended item's attributes values. Finally, we use two state of the art predicting algorithms to compare their performance when features are bound to certain action of the user versus when they don`t differentiate between actions. In this section we describe feature engineering process and algorithms used to analyze the results of our experiment.

3.1 Feature Engineering

3.1.1 Personalized feedback features

Our intuition is that a user clicks on job titles that are mostly interesting to him/her.

Now, we assume that we have a set of job concepts extracted from job titles and users' click data for recommended jobs. Given an action, i.e., bookmark, reply or delete, we define relevancy of those actions and the users' clicked jobs' concepts as how many times the user has acted upon those concepts. Formally, we define that relevancy as follow:

Given concept $c \in C$, a set of all concepts from all titles of jobs; action $t \in T$, a set of action {bookmark, reply, delete}; user $u \in U$, a set of users; a title of recommended job item as a set of concepts and their respective count as $C_{ct}=\{c_1:count(c_1),c_2:count(c_2),\dots,c_n:count(c_n)\}$, where $count(c_i)$ represents how many times a user has interacted with concept c_i w.r.t to action t. We define feature x_t as follow:

$$x_{t,u} = \frac{1}{tavg}\sum_{i=1}^{n} a\,(count(c_i)) \;\; with \;\; a\begin{cases} a = 1 \; if \; c_i \in C_{ct} \\ a = 0 \end{cases} \qquad (2)$$

,where $tavg$ is the average number of concepts in a title. This feature allows us to capture information about the relevance of the concepts in the job title for a given user. If the recommended job title has programming concept in it, $x_{bookmark,u}$ will be expressing how many times the user u has bookmarked job titles with programming concept. If u has not interacted with jobs that have programming in their title before, $count(programming)$ will be zero meaning that programming hasn't been interesting to user u in the past. Note that the order of appearance of concepts in title is not relevant; it's the frequency of interaction with each concept in the title for the user u that counts. We extended this feature engineering process to all actions among bookmark, reply and delete which led us to have $x_{bookmark,u}$, $x_{reply,u}$ and $x_{delete,u}$ to get feedback on the intent to perform each of those actions when user u is recommended with a job item with title that has concepts he has often bookmarked, replied or deleted. The process is the same for job item tags as they are also expressed in terms of tag concepts.

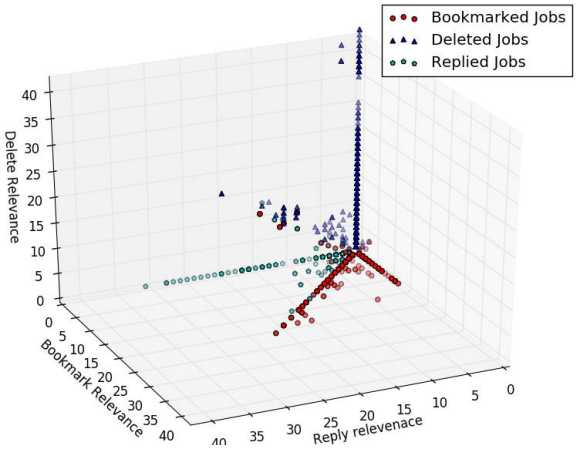

Figure 1: Distribution of Job titles over their relevance to user actions

For example a programmer might delete jobs that have a career level of CEO of a programming company; if such jobs come into recommended items because the title concepts are among most replied by the programmer, this feature will allows telling that this is not among his favorite career levels and may result into deletion.

Given a recommended job item i_k to user u where k means the desired career level of that job, an action t to perform on that recommended item, $l_{u_t,k}$, a list of job items that user u has interacted with that have career level k with regards to t and $l_{u,k}$ a list of total items with career k the user interacted regardless of t, we define career frequency feature with regard to t as:

$$cf_t(i_k,u,t) = \frac{|\,l_{u_t,k}|}{|\,l_{u,k}\,|} \qquad (3)$$

We extend the same process to industry-id, discipline-id country and region that are common attributes between a user and a job item.

3.1.2 User-Item related features

We generate the same type of features as the ones in the previous section but regardless of tasks.

Using analogy of career frequency features cited in previous section, we formally define career frequency regardless of action as follow:

Given a recommended job item i_k to user u where k means the desired career level of that job, $l_{u,k}$, a list of job items that user u has interacted with career level k, l_u a list of total job items the user interacted with, we define career frequency feature as:

$$clf_u(i_k, u) = \frac{|\ l_{u,k}\ |}{|\ l_u\ |} \qquad (4)$$

Job title and its tags related features express how many times the user has interacted with the item by using an aggregated value of all actions the user has done on related concepts.

For career level, if the recommended job has career level as professional, the career level frequency feature will express how many times the user has interacted with this job that requires professionals as career level. Note that this feature will combine the number of how many times the user has deleted, bookmarked and replied items with career level professional which is not the case for equation (3).

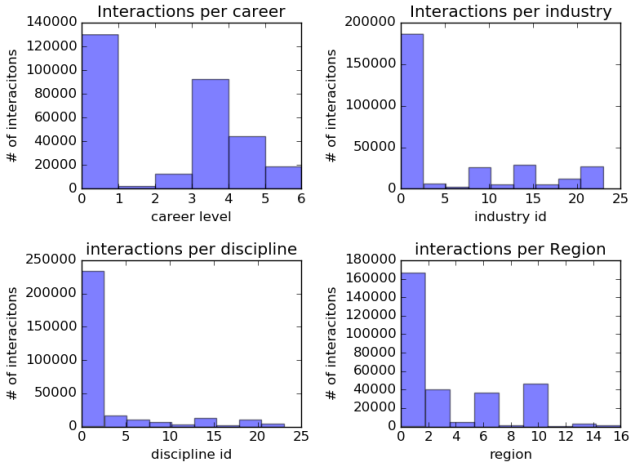

Figure 1 : User-item interactions distribution over item attributes

We use overlap [9] to measure similarity of common attributes between user and recommended job item.

Given a user u, item i and a set of common attributes $A=\{at_1, at_2, \ldots\ldots, at_n\}$, similarity in attributes is calculated as follow:

$$f(u,i,A) = \frac{1}{n}\sum_{j=1}^{n} s(u_{at_j}, i_{at_j}) \begin{cases} 0 & if(at(j)_u \neq at(j)_i) \\ 1 & if(at(j)_u = at(j)_i) \end{cases} \qquad (5)$$

We estimate a pseudo-session by selecting all user interactions on the same day when the first click on the item happened. This was motivated by the fact that jobs before they get replied, had been clicked or bookmarked and consequently had more interactions during the day compared to non-replied items therefore we generated session-look alike to generate a feature that express if the recommended item is the most clicked in that pseudo-session.

Intuitively this also means that deleted items have at least one click in the session therefore might be easier to separate from the rest.

In total we ended up with 30 features as shown in Tables 1.

3.2 Predicting algorithms

SVM [10] is an algorithm that maximizes the margin between the training sets patterns that can be applied to various classification functions. Maximizing the margin is done by solving the following optimization problem

Table 1: Personalized feedback and user-item features

Feedback feature(FeaturesX Actions)	
title_concepts_action tag_concept_raction career_action discipline_action industry_action country_action region_action	**7x3=21**
User-item features on interactions	
title_concepts_interactions tag_concept_interactions career_interactions discipline_interactions industry_interactions country_interactions region_interactions attribute_similarity most_interacted_during_day	**9**
Total	**30**

$$max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \, y_i y_j K(x_i, x_j) \quad (6)$$

$$Subject\ to \sum_{i=1}^{n} \alpha_i \, y_j = 0 \quad (7)$$

$$and\ 0 \leq \alpha_i \leq C\ , i = 1,2, \dots n \quad (8)$$

, where x is the training training vector, and $y \epsilon \{-1,1\}$ a label associated to x depending on the class it belong to, α, a parameter vector associated with the decision hyper-plane, $K(x_i, x_j)$ is the kernel function measuring the distance between x_i *and* x_j and C the upper bound of hyper-plane parameter α. SVM can be applied to multi-classification in two ways; One versus all, where one class is labeled positive and the rest of other instances are considered negative and One versus One, where pairs of classes' instances are trained one being positive and another negative in n given classes leading to $\frac{n(n-1)}{2}$ classifiers.

In our research we used the latter as the former makes the dataset imbalanced.

Feed forward neural networks [11] are artificial neural networks that let information flow in one direction from input to hidden layers and from hidden layers to output layers. They can have zero or more hidden layers. Each layer is composed of neurons.

A Single neuron in hidden or output layer takes input values from previous layer neurons; apply to them with weights, bias and passes the results through the activation function before sending the output the next layer or produces final output in case of output layer.

In this research, in the hidden layer, the results are feed to the sigmoid as an activation function and the output of that function is fed to the output layer neurons.

Each neuron's output in hidden layer can be defined as follow:

Given input vector $(x_1, x_2, x_3 \dots, x_n,) \epsilon\ \mathbb{R}^n$; and $W(w_0. w_1, w_2, \dots, w_n) \epsilon\ \mathbb{R}^{n+1}$, as a weight vector associated to that input, b, being the bias; the output of each neuron in our hidden layer is given by the following function:

$$h(x_1, .., x_n, b) = \frac{1}{1 + e^{-(w_0 b + w_1 x_1 + \cdots + w_n x_n)}} \quad (9)$$

Output and hidden layers parameters are learnt using the back propagation algorithm [12].

A softmax function is used to output the probability of each class and cross entropy loss function for all wrong predictions we've made.

4 EXPERIMENT AND DISCUSSION

In this section we describe the details about our experimental settings, dataset and discuss about our findings.

4.1 Experiment settings

We used an 8GB RAM with a CPU of type core i5. We didn`t take advantage of GPU usage while training neural network.

For SVM model training, we initialized our model with the following parameters:

Table 2: SVM Parameters settings

Parameters	Values
C: Penalty for the error term	1
γ: Kernel Parameter	0.01
K: Kernel Type	RBF[2]
Decision function shape	One versus One

Neural network weights were randomized initially and were adjusted using Gradient Descendent optimizer with a learning rate of 0.01. Bias for hidden layer and output layer were all initialized to 1. We used one hidden layer with neurons that are equal to the size of input; 30; because we didn't see any significant improvement in training accuracy by using additional hidden layer or number of neurons on 30 epochs.

We use Tensorflow[3] as our development environment.

4.2 Dataset

The dataset comprises of impressions, interactions, items and users.

[2] Radius basis Function

[3] https://www.tensorflow.org/

Xing provided around 10 million of impressions. Each represents which items were shown on the home page by the exiting Xing job recommender to which user in which week of the year. A typical record in impressions has user- id; ID of the user in users records; year, week of the year, items; a comma separated list (not set) of items that were displayed to the user; as attributes.

It provided around 8 million of interactions that are job-items user clicked, replied to, bookmarked or deleted; we are only interested in the deleted, replied to or bookmarked. Each interaction comprises of user-id, item-id, interaction-type and the time the interaction happened. 1 million job-items were also provided. Each has attributes such as title, tags, career level, industry-id, discipline-id, country, region, active during data collection, etc.

We downsized the dataset of 42^{nd} and 43^{rd} weeks as training set and predicted user post-click actions by using the dataset of 44^{th} week. The resulting dataset comprise 22,344 users, 52,946 job items and 140,958 interactions between users and items.

We note a relative small number of interactions exist due to the fact that users aren't looking for jobs every day and that the dataset was designed primary for the challenge. The organizer of the challenge created artificial users who have few to zero interactions to add noise to the dataset.

4.3 Results and discussion.

We did our experiment into two phases; phase one using feedback features and omitted them in phase two.

For SVM, we reached a score of 91% of accuracy using personalized feedback features and 77% after omitting them. As Table 3 and Table 4 show, feedback features improve our result by 14%.

High precision and recall on deleted recommendation was intuitively obvious since most of the deleted items have almost one click which is delete, however replied items precision was slightly low because users tend to bookmark items to reply them later therefore causing an overlap between replied and bookmarked job items.

On the other hand, we didn't get significant improvement in using neural network as they performed almost the same as SVM, reaching a prediction accuracy of 91% with using personalized feedback features and 77% by omitting them. More details about the predictions results of both algorithms are listed in table 3 and 4.

5 CONCLUSION

In this paper we introduced a way of engineering and using personalized feedback features to predict post-click actions the user might take after visiting a recommended job item. Using feedback features give captures strong signals from user item interactions. We compared the behavior of SVM and neural networks using and without using personalized feedback features which yielded same performance in terms of accuracy however the introduction of feedback features improved the accuracy of both classifiers accuracy at 14%.

This research can be useful in ranking items by giving high ranking scores to items that this approach predicts that they will be replied or bookmarked and also help filter items that a user is likely to delete in his recommended items list.

In our future work, we will be including the case where users clicks on item but doesn`t take an action so we can confirm the effectiveness of the findings in this research.

This research also didn't expand its' scope to cold start case as it focus more on user past history therefore might not be able to recommend suitable items to new users. In future work we might explore how we can exploit nearest neighbors of the new comer for suitable recommendation.

Table 3: SVM vs Neural nets Predictions with personalized feedback features

	SVM			Feed Forward Neural nets		
Measures	Precision	Recall	F1-score	Precision	Recall	F1-score
Bookmarks	0.97	0.44	0.61	0.98	0.42	0.58
Replies	0.77	0.99	0.87	0.77	0.98	0.86
Deleted	0.99	1.00	1.00	0.99	1.00	0.99
Avg total	0.93	0.92	0.91	0.93	0.91	0.90
Accuracy : **0.91**				**0.91**		

Table 4 : SVM vs Neural nets Predictions w/o personalized feedback features

	SVM			Feed Forward Neural nets		
Measures	Precision	Recall	F1-score	Precision	Recall	F1-score
Bookmarks	0.88	0.13	0.23	0.58	0.59	0.59
replies	0.67	0.60	0.63	0.63	0.66	0.64
Deleted	0.80	0.99	0.89	0.89	0.87	0.88
Avg total	0.78	0.77	0.73	0.78	0.78	0.78
Accuracy : **0.77**				**0.77**		

6 REFERENCES

[1] S. Loh, F. Lorenzi, R. Granada, D. Lichtnow, L.K. Wives, J.P. Oliveira, Identifying similar users by their scientific publications to reduce cold start in recommender systems, Proc. of the 5th Int'l Conf. on Web Information Systems and Technologies (WEBIST2009), pp. 593–600, 2009.

[2] S.T. Park, W. Chu, Pairwise preference regression for cold-start recommendation, in: Proceedings of the 2009 ACM Conference on Recommender Systems, pp. 21–28, 2009.

[3] Ge, M., Delgado-Battenfeld, C., & Jannach, D. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In RecSys (2010)

[4] Chakrabati, D., Agarwal, D., AND Josifovsnki, V. 2008. Contextual advertising by combining relevance with click feedback. In Proceedings of the 17th international conference on World Wide Web. 417–426.

[5] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. Transactions on Intelligent Systems and Technology, 2013.

[6] Recsys Challenge 2016: job recommendations based on pre-selection of offers and gradient boosting. 10th ACM Conference on Recommender Systems - RecSys 16

[7] W. Xiao, X. Xu.,K. Liang,J Mao,J Wang 2016. Job Recommendation with Hawkes Process. 10th ACM Conference on Recommender Systems - RecSys '16

[8] Hawkes, A. G. (1971a): "Spectra of some self-exciting and mutually exciting point processes," Biometrika, 58, 83—90

[9] C. Stanfill and D. Waltz. Toward memory-based reasoning. CACM, 29(12):1213–1228, 1986.

[10] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pages 144–152. ACM Press, 1992.

[11] C. M. Bishop, Neural Networks for Pattern Recognition. New York: Oxford Univ. Press, 1995

[12] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning representations by back-propagating errors. *Nature*, 323, 533--536.