Job Recommendation with Hawkes Process

An Effective Solution for Recsys Challenge 2016

Wenming Xiao[†] and Xiao Xu[†] and Kang Liang[†] and Junkang Mao[†] and Jun Wang[‡]*
[†]YunOS BU, Alibaba Group, 969 West Wenyi Road, Hangzhou 311121, China
[‡]Search BU, Alibaba Group, 500 108th Ave NE, Bellevue WA 98004, USA
{wenming.xiaowm, alfred.xx, liangkang.lk, junkang.mjk, j.wang}@alibaba-inc.com

ABSTRACT

The RecSys Challenge 2016 focuses on the prediction of users' interest in clicking a job posting in the career-oriented social networking site *Xing*. Given users' profile, the content of the job posting, as well as the historical activities of users, we aim in recommending top job postings to users for the coming week. This paper introduces the winning strategy for such a recommendation task. We summarize several key components that result in our leading position in this contest. First, we build a hierarchical pairwise model with ensemble learning as the overall prediction framework. Second, we integrate both content and behavior information in our feature engineering process. In particular, we model the temporal activity pattern using a self-exciting point process, namely *Hawkes Process*, to generate the most relevant recommendation at the right moment. Finally, we also tackle the challenging cold start issue using a semantic based strategy that is built on the topic modeling with the users profiling information. Our approach achieved the highest leaderboard and full scores among all the submissions.

CCS Concepts

•**Information systems** → **Recommender systems;** *Learning to rank;* •**Theory of computation** → Boosting;

Keywords

Recommendation Systems, Top-N Ranking, Point Process, Ensemble learning

1. INTRODUCTION

The Internet and the social network dramatically promotes the needs of consuming various type of data service for online users. For instance, the online media websites provide the convenience for accessing the latest domestic and international news. The well-known online stream sites *Netflix* and *Youku* attract millions of subscribers and feed them with all types of video contents. In addition, the rapid growth of professional social networks like *Linkedin* and *Xing* build their online service based business models by selling access to information about their users to recruiters and sales professionals. In order to provide better services and improve the efficiency, it is critical for those online service providers to predict the intent of the users and present them the proper content from the tremendous amount of information warehouse. Such a process is typically refered as standard recommendation systems, where the user profiling, content of the items, as well as the user historical activities are digested to model users' intent or interest.

RecSys Challenge 2016 is organized for such a purpose of matching the users' interest with certain digital content [2]. In particular, *Xing*, the German based career-oriented social networking site, generously provides tremendous of rich data in an anonymous format for RecSys Challenge 2016. By offering personal profiles, groups, discussion forums, event coordination, and other common social community features on both xing.com and the corresponding mobile apps, the key objective for *Xing* is to match candidates with appropriate job opportunities or recruiters. For this years' edition of the RecSys Challenge, the task is specifically designed to predict users' clicks on job postings. This contest can be formed as a standard recommendation problem with the goal of generating a list of job posts to a certain user.

Recommendation systems [15] have been intensively studied in recent years due to its wide applications many domains, ranging from music recommendation to e-commerce platform to financial services. Briefly speaking, there are two basic categories of recommendation systems, i.e., collaborative and content-based filtering [14, 17]. In addition, many researchers studied hybrid techniques to improve the accuracy of the recommendation systems [5]. However, most of the existing methods heavily rely on user feedback signals, e.g. user-item clicks in this contest, which suffer from challenging issues like the sparseness and nosiness of the data. Furthermore, the temporal characteristics is often overlooked in the traditional recommendation systems. In this paper, we will introduce our strategy for winning RecSys Challenges 2016. In particular, we build a pairwise ranking model with ensemble learning as the overall prediction framework. In addition, we specifically employ Hawkes Process to model the temporal patterns of the users' historic activity that leads to recommending relevant items at any given moment. Finally, we will discuss our remedy for the cold start issue in this contest.

*Corresponding author: W. Xiao and J. Wang.

RecSys Challenge '16, September 15 2016, Boston, MA, USA
 ISBN 978-1-4503-4801-0/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2987538.2987543

Table 1: User Profiles

Fields	Type	Comments
Job Role	Categorical	including Unknown, Student, Beginner, Experienced, Manager, Executive, and Senior Executive
Discipline	Categorical	including Consulting, HR, etc.
Industry	Categorical	including Internet, Automotive, Finance, etc.
Country	Categorical	including Germany, Austria, Switzerland, Others
Entries	Numeric	the number of CV entries
Experience	Numeric	the number of years in the job
Education	Categorical	including Unknown, Bachelor, Master, PhD
Employment	Categorical	including Full-Time, Part-Time, Freelancer, Intern, Voluntary, and Unknown

2. PROBLEM STATEMENT AND SETTING

Given the profile information of the users, the content of job posting, and the historical log of users activities, the key task of this contest is to recommend a list of job posts, which the users might interact with in the next week. In this section, we will introduce the data, discuss the problem setting, as well as describe the evaluation metric.

2.1 Data

The data contains three key components: the users' profile information, the content of job posting, and the interactions between users and job posts. There are a total 1.5 millions users and 1.35 million job posts. The user profile and the posts are processed and transformed into Bag-of-Word representation. For instance, the job role information of users is mapped to a total of $12,000$ key words and there are averagely two words per job role. The job tags are represented by $93,000$ words and each job tag has around 8.2 words. Table 1 shows some important fields contained in the user profile information. The training data are captured from the 34-th week to the 45-th week in 2015, which contains around 8.8 million interactions between $780,000$ active users and $1,020,000$ posts. For the user interactions, there are four types:

1 the user clicked on the item;

2 the user bookmarked the item;

3 the user clicked the reply or application button under the item;

4 the user deleted the item from a list;

In this contest, the first three types are treated as positive feedback equivalently and the last one is regarded as a negative feedback. For the testing period, the interactions of the 46-th week are used to generate the ground truth. For the testing set, there are $150,000$ users and $320,000$ active jobs.

2.2 Evaluation Metric

In this contest, we use the success rate of the top-K recommendation as the basic evaluation metric, which is the standard use case at *Xing*. In particular, if the user interacts with one of the top-K recommended job posting, the recommendation is considered as a success. Here the value of K is set as 30 and the total number of users is $N = 150,000$ for the training dataset. Therefore, we formulate a personalized ranking problem to generate the list of job posts as S, where $r_i = R(u_i)$ is the recommended job posts for the user u_i and $i = 1, \cdots, N$. Note that for the real recommendation systems, the most reliable evaluation results are typically performed by online AB testing [15]. In other words, each recommender will deploy the results using a randomly selected traffic (i.e. users) and compare the results based on the measurement of the real user interactions. However, such an online evaluation strategy requires significant efforts of engineering engagement.

In this contest, the organizer provides an alternative way to perform offline evaluation. For the testing period and target users, the system recorded their behavior and generate a ground truth set of user-item tuples T. Here $t_i = T(u_i)$ gives the list of items, which the user u_i interacted with during the testing period. Hence, given the predicted recommendation S, the online evaluation system will produce a score $score(R, T)$ to measure how relevant of the recommendation to the ground truth. The scoring function is defined as:

$$score(R, T) = \sum_{i=1}^{N} s(u_i) \tag{1}$$

$$s(u_i) = 20\,(P_2 + P_4 + R + UserSuccess) + 10(P_6 + P_{20}).$$

Here P_2, P_4, P_6, P_{20} represent the precision at the top k positions, i.e., $k = 2, 4, 6, 20$ and R is the recall. The value of "UserSuccess" is binary with one indicating at least one recommended item was interacted by the user and zero being no hit on the recommended items. The value $s(u_i)$ measures the quality of the recommendation of user u_i and the final score $score(R, T)$ is the summarization over all the N users. Since there are over $320,000$ job posts for the testing set, it is fairly challenging to recommend no more than 30 to the users. In addition, the ground-truth data is dependent on various factors, such as search and recommendation algorithms and user interfaces that were deployed on *Xing*'s site. Note that the captured ground-truth data will influence the offline evaluation process.

dependent to the real deployment of the *Xing*'s recommendation system. If a job post never shows in the impression of *Xing*'s online system, there will not be possible to collect any interation between the user and the post.

3. METHODOLOGY

In this section, we will present the overall framework for job recommendation and also describe the process of modeling temporal patterns of users' interactions with Hawkes process.

3.1 Overall Framework

In this contest, we build a hierarchal ranking model with ensemble learning as the overall framework. In this hierarchical framework, the first layer absorbs original features to build multiple ranking models and generate the relevance scores for a user-item pair. The second layer integrates the above relevance score and the the information extracted

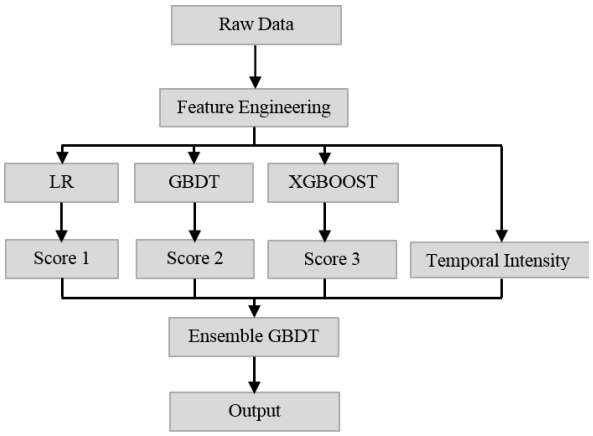

Figure 1: The conceptual diagram of the overall recommendation model.

from users' temporal behavior pattern in an ensemble way to produce the final ranking score. The conceptual diagram of the proposed framework is illustrated in Figure 1.

As shown in the digram, there are three basis learning models used in the first layer, including logistical regression, gradient boosting regression tree (GBRT) [10], and a recent version of boosting tree, namely eXtreme Gradient Boosting (XGBOOST) [12]. In addition, we utilize the pairwise ranking model to generate the recommendation list [13]. For the training data with impression information, such type of ranking models will produce a list of jobs with clicked items being ranked higher than unclicked or deleted items. In particular, motivated by the regression tree using relative relevance judgments [20], we developed a pairwise version of the GBDT algorithm for the contest. The output of the basis models represents the learned relevance scores using different models. These relevance scores are treated as features to feed into the second layer of ensemble models to produce the final ranking scores. Ensemble learning has been shown to be powerful for integrating multiple weak learner [6, 21]. In particular, we also noticed that ensemble learning is also employed in the past RecSys challenge [16]. Note that for the second layer, we also provide a *temporal intensity* as additional features, which capture the temporal iteration patterns between users and items. By such an integration of relevance scores and temporal intensity, we aim in recommend the most relevant jobs at the right time.

3.2 Feature Representation

As discussed earlier, there are three important components in the *Xing*'s data:, i.e., users' profile, the content of the posts, and the historical interactions. Accordingly, we construct feature representation based on such information. In general, we divide the features into three categories: user features $\mathbf{X}_u$, post features $\mathbf{X}_i$, user-post interaction features $\mathbf{X}_{ui}$. For the users' feature, we transform the categorical features into binary representations. For instance, in Table 1, the education information is converted into 4-bit binary codes, e.g., $(0, 0, 1, 0)$ indicating a user with a Master degree. Besides the profiling information, we also compute the statistics of the users' activity as additional user related representation. Similarly, we can obtain the features of job posting based on the semantic content, such as job's description, industry, and so on. For the user-item interactions, we are given a behavior matrix between N users and M jobs as $\mathbf{B} = \{b_{ij}\}_{i,j=1}^{i=N,j=M}$, with $b_{ij} = 1$ indicating the i-th user has an interaction on the j-th job. For each type of action, we can create such a behavior matrix accordingly. Given the semantic content of the jobs, we can compute item-item similarity $\mathbf{S} = \{s_{lm}\}_{l,m=1}^{l,m=M}$, where s_{lm} measures the similarity between the i-th and j-th job. Therefore, we can compute the intent of a user to a job using the behavior and similarity matrices as $m_{ij} = \sum_l b_{il} s_{lj}$, where m_{ij} measures the potential interest of the i-th user to the j-th job. Then we can derive a feature vector $\mathbf{X} = \{\mathbf{X}_u, \mathbf{X}_i, \mathbf{X}_{ui}\}$ corresponding to a user-post pair.

3.3 Temporal Intensity with Hawkes Process

Given the historical log of user activities, e.g., click behavior, we can formulate it as a time sequence $\{t_1, t_2, \cdots, t_n\}$. Then the conditional intensity function $\lambda(t)$ can be modeled using temporal point process to characterize the portability that the next behavior will happen at time t ($t > t_n$). As discussed in [1, 18], such an intensity function $\lambda(t)$ is designed to estimate the phenomena of interests. Motivated by [9, 11], we employ a low-rank self-exciting Hawkes process to model the temporal behavior patterns, as introduced in below.

For a standard Hawkes process, it models the intensity of recurrent user activities as

$$\lambda(t) = \lambda_0 + \alpha \sum_j \gamma(t, t_j). \tag{2}$$

Here, λ_0 is a baseline intensity, the kernel function $\gamma(t, t_j)$ captures temporal dependencies, and α is a weight coefficient. The summation of $\gamma(t, t_j)$ over time gives the dependency of the user's historical activities. Note that the above standard Hawkes Process models recurrent user activities. In order to extend the estimation of temporal intensity to unseen user-item pairs, we can give the following intensity function for any user-item pair (u, i) as

$$\lambda^{u,i}(t) = \lambda_0^{u,i} + \alpha^{u,i} \sum_j \gamma(t, t_j^{u,i}). \tag{3}$$

Here $\lambda_0^{u,i}$ and $\alpha^{u,i}$ are the (u, i)-th entry of the base intensity matrix $\mathbf{\Lambda}$ and the self-exciting matrix $\mathbf{A}$. Since it is often assumed that the users' interactions and the items' characteristics can be clustered in groups, like the well-known Nextflix problem [3], here we can rely on a similar assumption and impose low-rank structures on both $\mathbf{\Lambda}$ and $\mathbf{A}$. By doing that, we can transfer the knowledge from observed user-item pairs to unseen user-item pairs [9]. Some advanced extensions can leverage the user-profile and item-content information into the base intensity matrix to formulate a context-aware Hawkes Process [7, 8]. Instead of using the estimated intensities to perform final ranking for time-sensitive recommendation, we use the temporal intensities as additional features, complementary to the previously computed relevance scores from the first layer models, to train a final ranking model, as seen in Figure 1.

3.4 Cold Start Problem

Finally, we noticed a small portion of new users and new job posts, which there were no historical interactions. For instance, in the training data (from the 34-th week to 45-th week), there are about $580,000$ new users and $129,000$ active new job posts. For such a cold start problem, we proposed

to employ a content based collaborative filtering method for *Xing*'s job recommendation. Here, we briefly describe the process.

As introduced earlier, *Xing* provides basic profile including demographic data, predefined taxonomy for some job categories, as well as users' interest. Note that users have job roles and job posts have tags and titles, which provide the raw features to compute semantic relevance between new users and job posts. Similarly, we can use demographic data, education degree, career level, work experience, etc, in the relevance computation. In the contest, we use TFIDF to capture the key semantic words and jaccard similarity to measure the relations. In addition, we also employ LDA [4] to extract latent topics. In the latent topic space, we utilize the cosine similarity compute the potential interest between a new user and a post. Finally, based on similar relevance measure, we can compute user-user and item-item similarity and generate the job recommendation for new users using a two-way nearest neighbor method, as suggested in [19].

4. CONCLUSIONS

In this draft, we have discussed the winning strategy for RecSys Challenge 2016. We summarize our key ideas that we applied to the contest in below.

1 We designed a hierarchical learning model that can capture semantic relevance as well as temporal characteristics of user-item iterations;

2 We specifically utilized a self-exciting point process, namely Hawkes Process, to model the temporal intensity of users' intent;

3 Note that there are a non-negligible amount of new users and job posts, we proposed to handle those cold start recommendation using a content based strategy;

In this contest, our strategy shows superior quality and also performs steadily. In our future study, we will perform extensive experimental evaluation to study the individual contribution from each component of this framework. In addition, considering the discrepancy between online deployment and offline testing, we are also interested in studying a better offline evaluation strategy for this type of challenges.

5. ACKNOWLEDGMENTS

We thank the organizers of RecSys Challenge 2016 and *Xing* for generously providing the opportunity and data resources for developing and testing our techniques and ideas.

6. REFERENCES

[1] O. Aalen, O. Borgan, and H. Gjessing. *Survival and event history analysis: a process point of view*. Springer Science & Business Media, 2008.

[2] F. Abel, A. Benczúr, D. Kohlsdorf, M. Larson, and R. Pálovics. Recsys challenge 2016: Job recommendations. In *Proceedings of the 2016 International ACM Recommender Systems*, 2016.

[3] J. Bennett and S. Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

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

[5] R. Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

[6] T. G. Dietterich. Ensemble learning. *The handbook of brain theory and neural networks*, 2:110–125, 2002.

[7] N. Du, M. Farajtabar, A. Ahmed, A. J. Smola, and L. Song. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 219–228. ACM, 2015.

[8] N. Du, L. Song, H. Woo, and H. Zha. Uncover topic-sensitive information diffusion networks. In *Proceedings of the sixteenth international conference on artificial intelligence and statistics*, pages 229–237, 2013.

[9] N. Du, Y. Wang, N. He, J. Sun, and L. Song. Time-sensitive recommendation from recurrent user activities. In *Advances in Neural Information Processing Systems*, pages 3492–3500, 2015.

[10] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[11] A. G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[12] C. G. ianqi Chen. Xgboost: A scalable tree boosting system. In *Proceedings of the 22th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016.

[13] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[14] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[15] F. Ricci, L. Rokach, and B. Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[16] P. Romov and E. Sokolov. Recsys challenge 2015: ensemble learning with categorical features. In *Proceedings of the 2015 International ACM Recommender Systems Challenge*, page 1. ACM, 2015.

[17] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[18] I. Valera and M. Gomez-Rodriguez. Modeling adoption and usage of competing products. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 409–418. IEEE, 2015.

[19] J. Wang, K. R. Varshney, and A. Mojsilovic. Legislative prediction via random walks over a heterogeneous graph. In *SDM*, pages 1095–1106. SIAM, 2012.

[20] Z. Zheng, K. Chen, G. Sun, and H. Zha. A regression framework for learning ranking functions using relative relevance judgments. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 287–294. ACM, 2007.

[21] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1):239–263, 2002.