A Ranker Ensemble for Multi-objective Job Recommendation in an Item Cold Start Setting

Murat Yagci
Bogazici University
murat.yagci@boun.edu.tr

Fikret Gurgen
Bogazici University
gurgen@boun.edu.tr

ABSTRACT

Real-life recommender systems often have multiple objectives, and considering only a single stakeholder's perspective can be inadequate. ACM Recsys 2017 challenge requires such a multi-objective job recommender system taking into account job seeker satisfaction, recruiter satisfaction, balance between relevance and revenue, and scalability in a primarily item cold start setting. In this paper, we discuss our findings, and a possible solution to this problem. Making use of interaction profiles and content data, a hybrid of ranking algorithms is proposed to optimize the required objectives. This solution was able to achieve 8th position during A/B testing, and some of the ideas can be useful in a more complex ensemble.

CCS CONCEPTS

• **Information systems → Learning to rank**; **Recommender systems**; • **Computing methodologies → Learning from implicit feedback**;

KEYWORDS

Recommender systems, Multi-objective optimization, Implicit feedback, Cold start

1 INTRODUCTION

A multi-objective recommender system tries to find a good balance among several performance criteria required by the particular system. One example is a job recommender system where a good recommendation preferably satisfies multiple stakeholders of the system: job seekers, recruiters, and service providers.

In this year's ACM Recsys challenge [1], XING[1] opens up a rare real-life multi-objective job recommendation dataset to the research community. The data includes content information of job seekers and job ads as well as different types of implicit feedback from job seekers and recruiters. Given a cold start item (job ad) which has recently appeared in the system, the challenge problem is to find job seekers such that when the ad is recommended, the job seeker is satisfied, the recruiter is satisfied, the platform revenue is optimized using a solution scalable to millions of users and items. The offline part of the challenge allows participants to develop algorithms on historical data, whereas the final evaluation is performed online in an A/B testing environment in which recommendations are consumed in real time by the job seekers and recruiters existing in XING's system.

In this paper, we discuss our findings, and a possible solution to the challenge problem. To optimize the required objectives, we propose a ranker ensemble based on interaction profiles and content data available in the system. The same solution framework is used for both offline and online parts of the challenge with suitable top-N selection strategies.

The paper is structured as follows: In Section 2, we provide a possible problem representation, and propose a solution framework. Then, in Section 3, we describe the experimental setting and show experimental results. We conclude in Section 4.

2 METHOD

2.1 Problem definition and representation

Assume a set of users (job seekers) and items (job ads) in a job recommender system. Given a target subset of users and another target subset consisting of cold start items, the research problem is to find a maximum of N target users for each target cold start item such that the all-in-one multi-objective evaluation measure detailed in Section 3.1 is maximized. An additional constraint is that each target user can be recommended a maximum of M target items.

The challenge data includes a set of content features for both job seekers and job ads. Many of these features are bilateral allowing a match-based comparison. Furthermore, there are bipartite interactions between many job seekers (including about 70% of target job seekers) and past job ads with positive and negative semantics. This allows building interaction-based rankers in which job seekers can be represented with their interaction profiles rather than their content features.

We opt for building a feature vector representation for a given user-item interaction. This vector contains match-based features, each corresponding to a similarity score between the job seeker and the job ad in an explicit or latent space. It also contains some additional user-specific features, and time-based features. Such feature representations have been successfully used in recent challenges [4, 6].

The high level components of our solution are given in Figure 1. First, two rankers based on interaction profiles are blended. Then, a content-based ranker similar to the challenge baseline is trained. Best recommendations from both recommenders are further refined through different top-N selection strategies to produce final recommendations. We give details of this solution next.

[1] www.xing.de

RecSys Challenge '17, August 27, 2017, Como, Italy

978-1-4503-5391-5/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3124791.3124798

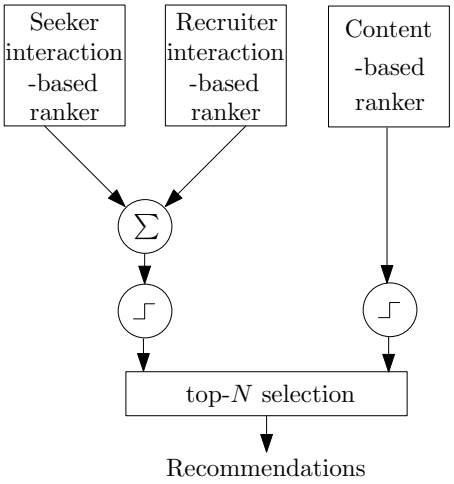

Figure 1: Ranker ensemble for multi-objective job recommendation

2.2 Interaction-based rankers

A recommender solely based on content features is the baseline of this challenge. We observe that relying on a job seeker's behavior (interaction data) in the system is often more reliable, in case such data exists. The main reason is that the content features based on a résumé or form-based data can be misleading due to several reasons like being outdated or incomplete. On the other hand, job ad content is often more elaborate and reliable. Based on this observation, we use a job seeker's positive interaction data to build her profile. More specifically, every interacted item contributes its content to her profile, such that in the end, each item content feature has a weight proportional to the number of times it appears in her interaction history. For example, if a user interacts positively with 5 distinct job ads, and 3 of these have the career level feature "Professional", then the weight of this feature in her profile is 3/5.

The job seeker interactions can often be considered positive with varying degrees of confidence. The delete interaction, on the other hand, has negative semantics, and a considerable impact on the evaluation measure. When training a classifier-based ranking model, we observe that labeling delete interactions as negative examples is not very helpful since deleted items do not usually have discriminative content. Instead, delete interactions can be used to decide, for example, if a job seeker is prone to deleting. Therefore, we opt for choosing random negative examples from the set of job ads not existing in a job seeker's interaction history.

For training and prediction, the rankers take input feature vectors which represent user-item interactions mainly in terms of job seeker profiles and job ad contents. We highlight relatively important features in these vectors in Table 1, and clarify them below:

- *Match-based features.* We try different methods to quantify match-based features. The best results are achieved as follows: Each match-based feature corresponds to a similarity score between a job seeker and a job ad. If a job ad content feature appears in the user profile, then their similarity score is nonzero and equal to the weight of this feature in the user profile. Some job ad content features may involve multiple features. For example, in the case of job title match, we sum the weights of overlapping job title features and normalize by the number of job title features appearing in the item content.
- *Latent semantic features.* SVD-based feature corresponds to similarity of a job seeker and a job ad in a latent semantic space. First, job seekers and job ads are placed in a single sparse matrix where a sparse row vector represents appearance counts of job title and tags features in a job seeker profile or a job ad content. Row vectors are then converted into TF-IDF vectors. We apply singular value decomposition (SVD) on the resulting matrix to map job seekers and job ads onto a k-dimensional latent space. Cosine similarity is used to find the feature value for a given user-item pair in this space.

Table 1: Numerical and categorical features representing an interaction between a job seeker and a job ad

Feature	*Importance*
Match-based features	
Job title	0.322
Career level	0.041
Discipline	0.116
Industry	0.041
Country	0.028
Region	0.104
Employment	0.028
Tags	0.082
Latent semantic features	
SVD-based similarity using job title/tags	0.118
Job seeker features	
Number of positive interactions	0.052
Number of negative interactions	0.002
Is willing to change job	0.001
Last interaction time	0.041
Has interaction last week	0.001

We train a binary classifier using this feature vector representation to obtain a relevance score for each eligible user-item pair. Any classifier capable of producing a probabilistic output can be used. We stick to gradient boosted decision trees (GBDT) classifier due to its performance on web-scale data with mixed feature types.

Since user and item repositories can be quite large, a general issue in web-scale ranking problems is the computational cost of predicting a relevance score for all possible user-item pairs, and sorting them to find topmost ranking items/users. To narrow down candidate pairs, we find the following strategies useful:

- *Rule-based pruning.* For example, it is usually inappropriate to recommend student jobs to managers, and vice versa.
- *Target item similarity.* Find content-based Jaccard similarity between target items and older items. Pair target items with target users who have positively interacted with the topmost similar older items. This is quite effective, but it can become costly with increasing number of target items.

- *Target user-item similarity.* Ignore a user-item pair if target job seeker profile and target job ad content have no match on some important features like job title and/or discipline.

2.2.1 Job seeker and recruiter satisfaction

The multi-objective nature of this challenge is pronounced in the evaluation measure with a relatively high score for correctly predicted recruiter interactions. To incorporate some sort of reciprocity [3], the interaction-based ranker is trained with job seeker interactions (clicks, bookmarks, and applications) and recruiter interactions separately. Since the predicted relevance scores are compatible, given a new user-item pair for prediction, we blend the scores from two rankers using simple aggregation methods like weighted average or harmonic mean. We observe that the two rankers create a useful diversity and their combination always improves the evaluation score up to 10% compared to a single ranker.

Reciprocal recommendation is problematic in the job recommendation domain since the recruiter interactions are often very sparse, lagging, harder to interpret, and possibly hidden due to business rules. This problem becomes more obvious especially in the online part of the challenge with even scarcer recruiter signals.

2.3 Content-based ranker

Unfortunately, there are also a significant number of cold start target job seekers (about 30%), and those with very few interactions. We train a second ranker to compensate for such job seekers based on their less informative content features. This ranker is similar to the interaction-based ranker except the match-based features and latent semantic features in Table 1 now depend on bilateral content features instead of interaction profiles. Furthermore, the interaction-based job seeker features are not always applicable.

It is also possible to improve content-based ranker with reciprocity by training two separate rankers for job seeker and recruiter interactions. However, this alternative is not further investigated in this work.

2.4 top-N selection and balancing revenue

Some job seekers in the system are premium (paying members) and some job ads are paid. The evaluation measure rewards pushing successful recommendations involving such seekers and ads. However, the penalty is also severe when a premium job seeker is pushed a bad recommendation.

We only match a target job ad with a premium target job seeker, if their relevance score is above some predefined threshold. Slightly boosting relevance scores of premium targets to improve their ranking has not improved the challenge evaluation score. In the end, we choose not to bias rankings towards premium job seekers.

For recommending job seekers to as many paid and unpaid job ads as possible, we use several strategies for top-N selection. In all strategies, we assume existence of candidate target job seekers for the target job ads. These candidates are obtained from the rankers, and have predicted relevance scores above some predefined threshold. The candidates are then sorted in descending order of relevance. We summarize these strategies as follows:

(1) *Simple aggregation.* Simply select top-N job seekers from candidates of both rankers. Prefer as many recommendations as possible from interaction-based rankers.
(2) *Improving aggregate diversity.* Obtain a maximum bipartite matching between eligible target job seekers and target job ads. To this end, we apply a graph-theoretic algorithm similar to [2]. The idea is to find a maximum bipartite matching several times iteratively. At each iteration, we exclude job ads and job seekers whose capacities (N or M) are full. This strategy can be especially useful when the possible number of recommendations is quite restricted as in the online part of the challenge. It allows to recommend job seekers to as many distinct job ads as possible.
(3) *Round robin.* Select the next job seeker from a job ad's candidate list by examining job ads in a round robin fashion. Accept selected recommendation if capacities (N or M) are not full.

In the offline part of the challenge, the first strategy is applicable since there is no restriction on the number of recommendations to a single job seeker (M = number of target items). However, monitoring the maximum number of recommendations to a target job seeker is still useful, since it can be balanced with a combination of above-mentioned strategies. In the online part of the challenge, there is a further restriction such that each job seeker can receive 1 recommendation only ($M = 1$). In this case, we first pick topmost ranked job seekers for every job ad, and apply the second strategy a few times. We then use the third strategy to enrich the final set of recommendations.

2.5 Scalability

The proposed solution is based on rankers with shallow tree ensembles, low-cost blending schemes, and efficient strategies for candidate and top-N selection. It is scalable to millions of users and items in the training phase, and able to serve daily load of target users and items in the prediction phase.

3 EXPERIMENTS

3.1 Dataset and evaluation

The challenge consists of an offline period for ideation and developing solutions, and an online period for A/B testing. The interactions in the offline dataset span a period of 3 months with the interactions of last week constituting a test set unavailable to challenge participants. There are 1.5 million job seekers and 1.3 million job ads with content information, of which 74,840 job seekers and 46,559 cold start job ads are targets known to have interactions in the test set. About 70% of target job seekers have positive interactions which show a skewed multiplicity distribution with a mode and a median of 1 and 4, respectively. The ratio of delete and unique positive job seeker interactions is about 1/6. About 23% of target job seekers are premium, and 27% of target job ads are paid. The online period involves 5-week A/B testing of different algorithms from the participants. Again, a long-term dataset is provided with comparable descriptive statistics. However, new target cold start job ads (up to about 13,000), and new target job seekers (about 50,000) are given daily.

```
score(targetItems) =
targetItems.map(item => score(item, recommendations(item))).sum

score(item, users) =
users.map(u => userSuccess(item,u)).sum + itemSuccess(item, users)

  userSucess(item, user) =
    (
        if (clicked) 1 else 0
      + if (bookmarked || replied) 5 else 0
      + if (recruiter interest) 20 else 0
      - if (delete only) 10 else 0
    ) * premiumBoost(user)

  premiumBoost(user) = if (user.isPremium) 2 else 1

  itemSuccess(item, users) =
    if (users.filter(u => userSuccess(item, u) > 0).size >= 1) {
      if (item.isPaid) 50
      else 25
    } else 0
```

Figure 2: Multi-objective evaluation measure

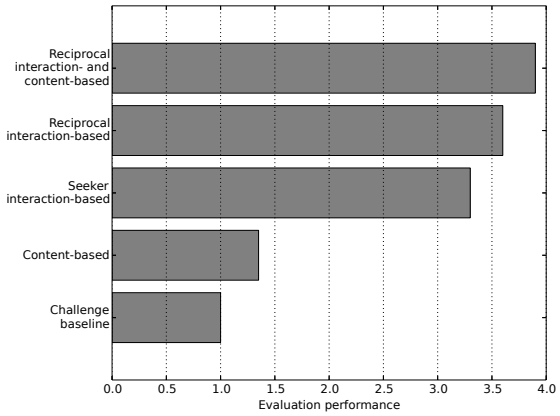

Figure 3: Comparative performance of different rankers with respect to challenge evaluation measure

In both offline and online parts, the evaluation criteria are similar except the restrictions on N and M. In the former, $N = 100$, and M is unrestricted, whereas in the latter, $N = 250$, and $M = 1$. The evaluation measure[2] expresses different expectations from the challenge solution such as reciprocity and revenue maximization as a single score. This measure is given in Figure 2. As an additional evaluation criterion, the online challenge has a restricted 24-hour time window for receiving the daily target items and users, and submitting recommendations to the system.

We use the Python ecosystem for this solution, specifically scikit-learn [5] for GBDT. The setup is a commodity computer with 4 cores and 8 GB main memory.

3.2 Experimental results

We split the offline training dataset into two using a time split at the beginning of last week. Cold start items of the last week together with their interactions constitute a validation set which helps us tune parameters for the final solution. The following parameters achieve the results in this paper: All rankers use a GBDT with 100 trees and a learning rate of 0.10. For training, we sample all positive and a balanced number of negative user-item pairs. The number of latent dimensions, $k = 50$, for SVD-based similarity feature. Relevance score thresholds are chosen in a range between 0.80-0.95 before top-N selection. As described in Section 2.4, the top-N selection strategy is task dependent.

For the interaction-based rankers, we report importance of chosen features in Table 1. These values correspond to how much each feature contributes to the reduction of impurity in all nodes across all trees. Similar feature importance values are obtained for the content-based ranker.

The improvement obtained using different ranking options is given in Figure 3 in a comparative fashion. We note that throughout our solution, we do not make use of impressions from XING's own recommender.

The online solution achieved 8th position with 70% of the average score of the top 3 scorers. The limited availability of interaction data during the online evaluation period was somewhat restrictive for retraining interaction-based models as well as monitoring daily performance. The available setup was able to produce predictions in about an hour for the daily A/B testing.

[2] http://2017.recsyschallenge.com

4 CONCLUSIONS

We discuss our findings and solution for the multi-objective job recommendation problem in an item cold start setting. We show that interaction-based rankers quickly outperform the content-based baseline with respect to the evaluation measure. We obtain the best recommendation candidates using a hybrid of rankers based on reciprocal interactions as well as content features. The proposed top-N selection strategies further refine the candidates for obtaining the final recommendations considering the problem constraints and objectives.

To further improve the performance, one possibility is to perform more extensive feature engineering. Another one is to combine results from as many diverse learners as possible exploiting their individual strengths.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments. This work is supported in part by Bogazici University BAP project no. 13241.

REFERENCES

[1] F. Abel, Y. Deldjoo, M. Elahi, and D. Kohlsdorf. 2017. RecSys Challenge 2017: Offline and Online Evaluation. In *Proceedings of 11th ACM Conference on Recommender Systems (RecSys '17)*.

[2] G. Adomavicius and Y. O. Kwon. 2011. Maximizing aggregate recommendation diversity: A graph-theoretic approach. In *Proceedings of workshop on Novelty and Diversity in Recommender Systems (ACM RecSys '11)*, Vol. 816. Ceur-ws, 3–10.

[3] L. Li and T. Li. 2012. MEET: A Generalized Framework for Reciprocal Recommender Systems. In *Proceedings of 21st ACM International Conference on Information and Knowledge Management (CIKM '12)*. 35–44.

[4] A. Pacuk, P. Sankowski, K. Węgrzycki, A. Witkowski, and P. Wygocki. 2016. RecSys Challenge 2016: Job Recommendations Based on Preselection of Offers and Gradient Boosting. In *Proceedings of ACM Recsys '16 Challenge*. 10:1–10:4.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[6] A. M. Yagci, T. Aytekin, and F. S. Gurgen. 2015. An Ensemble Approach for Multi-label Classification of Item Click Sequences. In *Proceedings of ACM Recsys '15 Challenge*. 7:1–7:4.