How Integration helps on Cold-Start Recommendations

Cheng Guo
Tsinghua University
guoc15@mails.tsinghua.edu.cn

Hongyu Lu
Tsinghua University
luhy16@mails.tsinghua.edu.cn

Shaoyun Shi
Tsinghua University
shisy13@mails.tsinghua.edu.cn

Bin Hao
Tsinghua University
haob15@mails.tsinghua.edu.cn

Bin Liu
Tsinghua University
l-b15@tsinghua.edu.cn

Min Zhang
Tsinghua University
z-m@tsinghua.edu.cn

Yiqun Liu
Tsinghua University
yiqunliu@tsinghua.edu.cn

Shaoping Ma
Tsinghua University
msp@tsinghua.edu.cn

ABSTRACT

The RecSys Challenge 2017 focuses on recommending proper or potentially interested users for job postings. Different from traditional recommendation tasks, most of the items (jobs) to be recommended are cold-items without interaction history. To address this problem, this paper introduces an effective integration method. The whole framework consists integrations in 4 different levels: feature-level, model-level, data-level and approach-level. First, we extract features from users' and jobs' profiles and make further abstractions. Second, we improve the model of Wide and Deep Learning, a method that aggregates both embedding and numeric features. Particularly, to receive better performance, both online and offline user-item interactions are used to train the model. Other methods such as content-based Linear Regression, Item-neighbor, Historical Enhancement and Xgboost are also taken into consideration. Finally, we learn lessons from the field of social choice and experiments show that results of aggregation by voting from different methods give a better performance, receiving 29732 in offline test and 6903 in online test.

CCS CONCEPTS

•Information systems →Recommender systems;

1 INTRODUCTION

Certain recommender systems are designed to satisfy diverse needs under different circumstances. Although both the categories or concepts are rapidly increasing, the main base of recommender systems almost remain the same, which are Content-based recommendation and Collaborative Filtering based recommendation. Methods of CF has received great success in accuracy and shows an acceptable effectiveness in handling real-world conditions where both users and items are updating each day. However, there are shortcomings which cannot be overcome in these kinds of methods. It is widely realized that Collaborative Filtering is not able to deal with cold-start problems, when "new" users or items are added into the system without interaction histories. Another problem is their defects in explaining why a list of certain items is recommended to a user, for the reason that "latent factors" used by CF are not that clear and convincing. On the contrary, Content-based methods have natural advantages in settling such problems. Each user/item, whether "old" or "new", with his/her/its own profiles/features, could be directly added into a content-based model. Explanations are also easier to be welcomed by telling a user the items recommended are with "obvious" factors where he/she shows interests. Content-based methods are more modeled as classification problems. Beyond the traditional KNN [6], LR [5] methods, classifiers with more effectiveness and higher accuracy like Xgboost [1] are made used of in this field of studies. Besides, Deep Learning frameworks, like NCF [4] and Wide and Deep [2, 7], are another excellent and premium machine learning ideas, which become very popular in recent years and widely used in recommendations.

The ACM RecSys Challenge 2017 [3] focuses on the problem of job recommendations on XING, given users' (job roles, career level, discipline, etc.) and items' profiles (industry, discipline, career level demanded etc.) and user-item interactions (clicking, bookmarking, replying, etc.). Compared to common recommendation tasks, most of the items (jobs) are "cold", which inspires us to think about the problem in the way of content-based ideas. Tremendous researches show that it is difficult for individual methods to get considerable performances and satisfy all the demands in certain scenarios. As a result, it is natural to come up with ideas to integrate various ideologies, letting each one show its special prowess. Based on this, this paper introduces an integration model especially for cold-start recommendation tasks. The paper is composed as follows: Section 2 gives a brief introduction of features and further analyses on them. In Section 3, we apply the structure of Wide and Deep Learning and make adjustments for the real task. Section 4 shows the reason of Data-level Integration and how the model is trained. Section 5 gives detailed instructions on how social choice can affect the results of recommendation systems. Methods on how our results are generated for online evaluation is talked about in Section 6. In the final part, evaluations and conclusions are given based on the whole work. Figure 1 gives an overview of our model.

RecSys Challenge '17, Como, Italy
© 2017 ACM. 978-1-4503-5391-5/17/08...$15.00
DOI: 10.1145/3124791.3124796

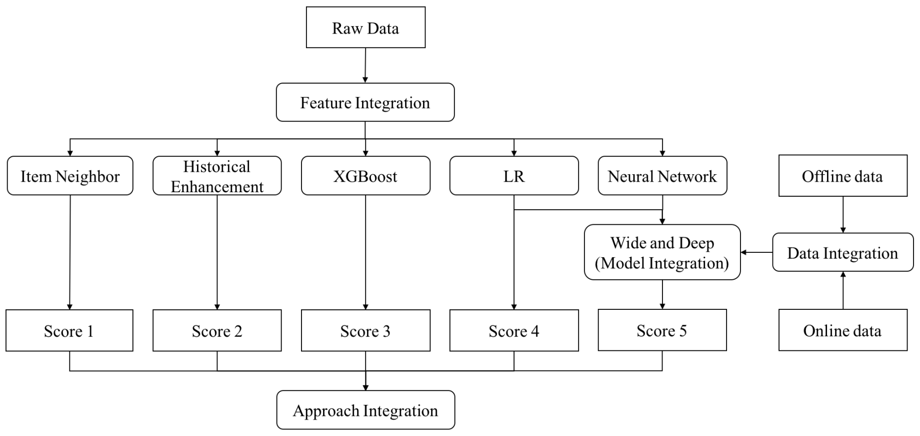

Figure 1: The conceptual diagram of the overall recommendation model.

2 FEATURE-LEVEL INTEGRATION

The main goal of recommender systems is to predict the preference of a certain user for a certain item. Accordingly, we extract features of users, items and user-item pairs, hoping that integration of these different features may help improving the performance of our algorithms.

2.1 User Features

To comprehensively capture users' preferences, we try to use different levels of user information. Firstly, we assume that the users' interests are related to their attributes. Therefore, we extract users' attribute-level features directly from users' profiles, such as user's jobrole, career level and country, which contain coarse-grained information of a user.

In addition, we also analyze users' historical interactions to understand users' habits and preferences. Different users have different interaction habits, such as the probability of a user clicking on a new item, the ratio of a certain item attribute in ones history, and the titles/tags of user-interacted items. Furthermore, by considering short and long term changes in user behaviors, we specify the time windows of different sizes, such as one or two weeks.

Users together with their preferences, could be modeled more precisely by combining users' attributive information and historical interactions.

2.2 Item Features

Similarly, items have attributive information as well, such as titles and industries. We directly extract these information as item attributive features. Furthermore, to make full use of text information of the items, we take advantage of a word embedding method (Word2Vec) to extract the semantic information from item titles and tags.

Although the challenge this year focuses mainly on cold-start problem, there is still a small part of target items with historical interactions. Therefore, this part of item-history features are extracted from their previous interactions.

2.3 Pair Features

Features of user-item pair are also extracted to describe the degree of suitability between a certain user-item pair. Firstly, we find that some attributes appear simultaneously in users and items, for example, the country and career level. Based on these corresponding attributes, we can measure the similarity between users and items. For attributes that contain sequences, such as title and tags, we calculate both the match numbers and Jaccard similarity. For numerical attributes, such as career level, we use the difference between them. And for category attributes, such as country, we use a "match or not" indicator. Secondly, we believe that the similarity between current item and a user's historical interacted items indicates ones interests in current item. For example, the frequency of current item attributes in a user's history reflects the probability of his/her being interest in this item.

To analyze effectiveness of features, we calculate the Pearson correlations between each feature and the interaction label, serving as reference in feature selections. Some of the effective features and their Pearson correlations are shown in the following Table 1.

3 MODEL-LEVEL INTEGRATION

One of our model is Wide&Deep, which is put forward by Google recently [2]. It combines the deep neural networks and wide linear models. The motivation is to combine the benefits of two different kind of models, shown in Figure 2. Deep neural networks can generate dense embeddings from sparse features and have a strong ability to process unseen features. While sometimes they may over-generalize. Therefore, combination with linear model can help recommend more relevant items, which can memorize linear relationship between input features and output results.

Logistic Regression serves as a bridge to implement the combination. To formalize the prediction, let Y denote the class label, $\mathbf{x}$ denote the original features, then we have:

$$P(Y=1|\mathbf{x}) = \sigma(\mathbf{w}_{wide}^{\top}[\mathbf{x}, \phi(\mathbf{x})] + \mathbf{w}_{deep}^{\top}a^{(l_f)} + b) \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, $\phi(\mathbf{x})$ is the cross product transformation of the original features, and b is the bias. $\mathbf{w}_{wide}$ is the

Table 1: Features of different levels

	Level	Feature Name	Pearson Correlation
User Features	Attribute-level	u_career_level	0.1449
		u_experience_n_entries_class	0.0966
	History-level	u_pos_ratio	0.8435
		u_delete_ratio	0.8341
Item Features	Attribute-level	i_is_payed	0.1630
		i_career_level	0.0858
	History-level	i_pos_ratio	0.6383
		i_delete_ratio	0.6468
Pair Features	Attribute-level	p_career_level_gap	0.1093
		p_tags_jobroles_match_num	0.0631
	History-level	p_tag_neghistag_match_num	0.8123
		p_title_neghistitle_match_num	0.7379
		p_country_hisratio	0.5000

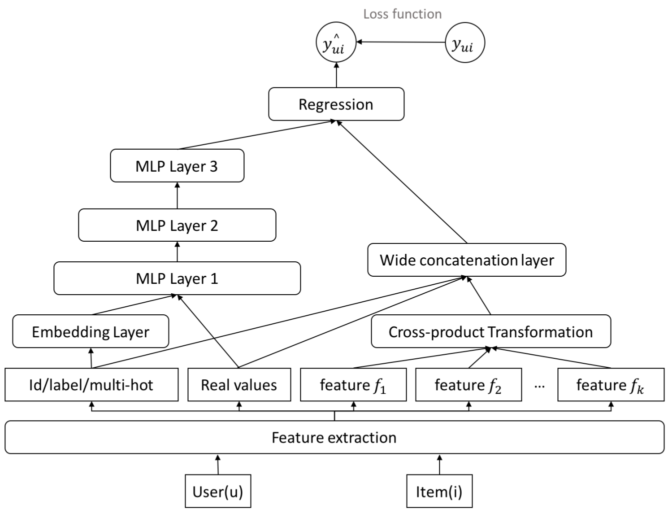

Figure 2: The model of Wide and Deep model adjusted for RecSys Challenge 2017 task.

vector of all wide model weights, and $\mathbf{w}_{deep}$ is the weights applied on the final activations $a^{(l_f)}$.

The cross product transformations on the linear part is an experience guide added artificially. It represents features' simultaneous appearance. For example, we have a 2-dimension one-hot vector for one's gender, and a 2-dimension one-hot vector for one's country (if Germany or not). Then we can generate a 4-dimension one-hot vector, which represents if one is a German man, woman or not. We can define the cross-product transformation as following:

$$\phi_k(\mathbf{x}) = \prod_{i=1}^{d} x_i^{c_{ki}} \quad c_{ki} \in \{0, 1\} \tag{2}$$

where ϕ_k represents the k-th transformation, c_{ki} is a boolean variable denotes whether ϕ_k is related to the i-th feature x_i.

The interaction data is organized by each user-job pairs. Categorical features are fed into an embedding layer and then concatenated with other features as the input of hidden layers on deep neural network. Cross-product of some categorical features are fed into the wide linear part. In our experiments, real-value features are input into both the deep neutral network and the wide linear part, which is found to improve the final result.

After some trials, we set the classify label as whether a user deleted the recommendation without any other positive interaction. In the offline test, for every job, we submitted 100 users who are supposed to have the smallest probability to delete the recommendation.

4 DATA-LEVEL INTEGRATION

A new dataset is provided in the online evaluations, independent of which is for the previous offline tests. Features could only be extracted based on this new dataset since both users' and items' ids are rehashed, leading to problems on how to update the model. One of the simplest ways is to retrain the model, while it's obviously not wise because there remains much information within the old dataset. For deep learning models, more training data usually means better performances. Fortunately, Our model do not use ids as input and it's designed to learn the relationships between input features and output labels. This relationship are supposed to be maintained across different dataset. As a result, the model can be trained with both online and offline data at same time. Another point is that, combining the two datasets and retraining may consume long time and much computing resource. Therefore, a model is designed to load the original model for offline evaluation and continue training process with the new data.

5 APPROACH-LEVEL INTEGRATION

5.1 Methodologies

5.1.1 Logistic Regression. In some sense, recommendation could be considered as a classification problem (usually 5-level in rating predictions and 2-level in item-recommendations) given a certain item and user. Logistic Regression is a classic regression model commonly used to address such problems, where the output represents the probability that user u is interested in item i. The equation is as follow, where $Score(i, u)$ is the prediction score on a item-user

Table 2: An example of Approval Voting

Voter	Ballot	A	B	C	D
RS_1	A C	1	0	1	0
RS_2	A B C	1	1	1	0
RS_3	A	1	0	0	0
Score		3	1	2	0

pair, f_e is the feature extraction function and || means a stitching operation.

$$Score(i,u) = (1 + e^{-\theta(f_e(i)||f_e(u))})^{-1} \tag{3}$$

5.1.2 Historical Enhancement. Considering both item attributes and user histories, we found that an item may be clicked because its attributes matched user's preferences. Accordingly, we assumed that a user's preferences of an item can be reflected in his/her history. For example, a user, who clicked jobs mostly from Germany, is more likely to be interested in jobs whose country is Germany. Given a user u, with its clicked items $H(u)$, and an item i, with its attributes set $\{a_i^1, a_i^2 \dots a_i^k\}$, we calculated the probability of u clicks i by combining the probability of u clicked a item whose k-th attribute is same as i.

$$Score(i,u) = \prod_{k=1}^{K} \frac{\sum_{i' \in H(u)} I(a_{i'}^k = a_i^k)}{|H(u)|} \tag{4}$$

5.1.3 Item-neighbour. The Item-neighbour idea comes from the traditional Item-based KNN method, while here it's not necessary to set a hyper parameter K. Similarities between items are calculated based on their metadata. We use tf-idf weightings modeling items into vectors, and cosine distance to measure similarity between item i and j, where $sim(i,j) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}|| ||\vec{j}||}$.And the prediction score for an i-u pair is:

$$Score(i,u) = \sum_{e \in E(u)} \omega(e) sim(i(e), i) \tag{5}$$

where $E(u)$ is is the set of events for user u, and $i(e)$ is the item of event e. $\omega(e)$ is defined as same as the RecSys Challenge evaluation metric: $\omega(clcik) = 1$, $\omega(bookmark) = 5$, $\omega(reply) = 5$, $\omega(delete) = -10$, $\omega(recruiter's\ interest) = 20$.

5.1.4 XGBoost. XGBoost is an effective open-source software library which provides the gradient boosting framework, also considered as a baseline method in this task. As an improvement of the baseline, the regression targets are chosen as: -1 for delete only, 0 for impression, 1 for click only, 2 for recruiter interest, and 3 for rely or bookmark. Such targets receive a best performance in our experiments within the same framework. In the regression part, several types of features are used including match features, similarity features, and users' history statistics. To further improve the result, we tune a threshold of minimum score(0.35) on the validation set, in order to filter out low-score pairs.

5.2 Voting System

Social choice theory is a theoretical framework for analysis of combining individual opinions, preferences, interests, or welfares to reach a collective decision or social welfare in some sense. The main idea of social choice coincides with the purpose to aggregate the results of different approaches, in order to get a reasonable and acceptable result on each aspect. Voting systems, or Electoral systems, are main applications based on Social Choice theories. We choose four effective and efficient voting methods: Approval, Copeland, Borda and Weighted Borda to implement the Integration of different recommender systems. In this task, we consider different recommender systems as voters who give ballots (recommendation results) on candidates (users to be recommended in this task) individually, and get the final result through the proposed voting system.

Table 3: An example of Borda/Weighted Borda Voting

Voter	Ballot	Borda				Weighted Borda			
		A	B	C	D	A	B	C	D
$RS_1(\omega_1 = 0.3)$	C B D A	0	2	3	1	0	0.6	0.9	0.3
$RS_2(\omega_2 = 0.2)$	B D C A	0	3	1	2	0	0.6	0.2	0.4
$RS_3(\omega_3 = 0.5)$	B A D C	2	3	0	1	1	1.5	0	0.5
Score		2	8	4	4	1	2.7	1.1	1.2

5.2.1 Approval. In traditional sense, Approval voting is a single-winner electoral system. Each "voter" may "approve" of (select) any number of candidates and the winner is the most-approved "candidate". The idea could be easily promoted to "multi-winner" elections. Suppose there are m recommender systems, each giving top-n recommended users to item i as input. For example, for item i, system RS_k gives the user list U_k, candidates (users) in U_k are approved while others are not. And the final result U_{output} is a ranked list according to the approved number from the ballots. Table 2 shows a simple case with 3 methods and 4 candidates each. Approval Voting has a linear computation complexity of $O(mn)$. While its disadvantages are also obvious that it ignores the order of candidates on each ballot, where user at position 1 and position 20 should be different.

5.2.2 Copeland. Copeland's method is a Condorcet method in which candidates are ordered by the number of pairwise victories, minus the number of pairwise defeats. As a method which gives a final score of each candidate, it's also easy to be promoted to multi-winner elections. The ballot of each voter is a ranked list according to his/ her opinion, which is just consistent with this task. Table 4 shows a simple case with 3 methods and 4 candidates. It's easy to conclude that the computation complexity of Copeland Voting is $O(mn^2)$.

5.2.3 Borda. Borda or Borda Count is a single-winner election method in which voters rank options or candidates in order of preference. The ballots in Borda is the same as Copeland. And the voting method could also be promoted to multi-winner cases because of its use of scores to describe a candidate's importance, as shown in Table 3. The final score of each candidate (user) u in recommended list given by RS_j given item i could be measured as follow, where RCN is the required candidate number:

$$Score(i,u) = \sum_{j}^{m} \sum_{i}^{n} (RCN - RS_j rank(i,u)) \tag{6}$$

Table 4: An example of Copeland Voting

Voter	Ballot	(A, B)	(A, C)	(A, D)	(B, C)	(B, D)	(C, D)	Score
	C B D A	False	False	False	False	True	True	
	B D C A	False	False	False	True	True	False	
	B A D C	False	True	True	True	True	False	
Result		(B, A)	(C, A)	(D, A)	(B, C)	(B, D)	(D, C)	
A		-1	-1	-1				-3
B		1			1	1		3
C			1		-1		-1	-1
D				1		-1	1	1

5.2.4 Weighted-Borda. As mentioned before, Borda is a voting method that concerns both order of each result given by recommender systems and computation complexity. While problems remain: whether different recommendation methods are of a same importance, or are they in the same level of reliability? To improve traditional Borda, we learn lessons from Weighted Voting, where voters are not the same but with different weights. The higher value of a voter's weight, the more he/she will influence the final results. Accordingly, the final score of each candidate (user) u is as follow, where ω_j is a hyper-parameter to measure a recommender system's reliability:

$$Score(i,u) = \sum_{j}^{m} \sum_{i}^{n} \omega_j (RCN - RS_j rank(i,u)) \tag{7}$$

The weight ω_j of each recommender system is difficult to set ordinarily, but fortunately, they could be estimated by large-scale offline experiments. Scores of the methods separately are used as reference to give their weights where $\omega_j = \frac{offline\ score(RS_j)}{\sum_k offline\ score(RS_k)}$, also shown in Table 3.

6 FROM OFFLINE TO ONLINE

Online recommendation task asks participants to submit users that may be interested in the provided items every single day. Online test breaks the limit of the original recommender system of XING, providing opportunities for users that may never perform interactions, leading to a completely change in concepts. Except for that, the main difference in demand from offline tests is that each user is only permitted to be recommended no more than once, and for each item the number is limited to 250. The task could be considered as an optimization problem, trying to recommend the best items to proper users. In the step of Approach Integration, we make use of voting in two ways: voting from users to "select" item lists, and voting from items to "select" user lists. Algorithm 1 shows the process of the selection from users for online test, where K is a hyper-parameter.

```
for $i$ in candidate_items do
    for $u$ in iu_voting($i$) do
        if ($u$ is recommended)or($Rank(ui\_voting(u), i) > K$)
         then
            continue;
        else
            recommended_list($i$).append($u$);
        end
        if recommended_list($i$).size $> 250$ then
            break;
        end
    end
end
```

Algorithm 1: Process of the selection from users for online test.

7 EVALUATIONS

The RecSys Challenge allows participators to submit results for specified items in both offline and online tests, which could be used as evaluations for our models. For online scores, scores per item are calculated according to the feedback interactions to make them comparable, for the number of items published everyday differs. Table 5 shows results on both offline and online tests. All the parameters are set on validation set to make sure the model gives the best performance.

The line "Before Feature Integration" means that models are trained on the original given features. From Table 5, all the methods receive an improvement in offline tests after Feature Integration. Great promotions haven been received in methods of Historical Enhancement, XGBoost, Logistic Regression and Neural Network, while the change is slight in Item-neighbour. The reason for this phenomenon is that user-item interactions put much larger influence rather than users' and items' profile features.

Neural Network shows a better performance than other simple methods, further more , Wide and Deep model, by integrating both deep learning and wide linear part, receives an even better score.

In Approach level Integrations, Item-neighbour, Historical Enhancement, XGBoost and Wide & Deep serve as voters for an expected better result. Conclusions could be drawn that each of the voting method shows an acceptable performance (20000+ score), especially superior than those low-score methods such as Item-neighbour and XGBoost. The results are consistent with the idea of "fairness" in Social Choice Theory. However, Approval Voting and common Borda Voting get lower scores than the most reliable "voter" Wide and Deep, considering that not each "voter" is equal on judging which user might be interested in a certain item. Situations are improved by emphasis on pair-wise relationships from Copeland Voting. Finally, by giving each voter a weight according to its previous score, Weighted Borda receives the best score among all the methods.

Table 5: Experiment results of offline/online evaluations

		Offline Score		Online Score/Item
		Before Feature Integration	After Feature Integration	
	Item-neighbour	12372	12438	1.0180
	Historical Enhancement	13590	20450	0.9541
	XGBoost	3781	14628	—-
	Logistic Regression (Wide)	6093	24168	—-
	Neural Network (Deep)	325	25539	—-
Model Integration	Wide and Deep	903	26855	1.0210
Approach Integration	Approval	—-	22737	—-
	Copeland	—-	27739	—-
	Borda	—-	25403	—-
	Weighted Borda	—-	28082	1.2177

Table 6: Experiment on Data Integration (3 ways to use offline data)

Model	online-only	fine-tuning	retrain
positive ratio	0.174	0.311	0.382

Due to the limitations of time and submission times per day, only some of the methods with good performances are selected and applied. For online evaluation, we report the best score from each model during a single day rather than taking the average. Since items provided every single day may be significantly various. The sores are calculated from feedbacks given by the competition system, using the same interaction metric. Approach Integration still holds the best performance, followed by Wide & Deep and Item-neighbour. The relatively poor performance from Wide & Deep could be explained by its over-fitting on training set. And for Historical Enhancement, the reason lies in that there is not enough interaction history to learn items' and users' attributes.

To validate whether Data Integration makes a difference, experiments are conducted on "online-only" (model trained with online data only), "retrained" (model retrained with combination of both offline and online data) and "fine-tuning" (offline data based model trained by adding online data).In order to make comparisons, the positive ratio of impressions on a certain day (18th May) is used as an evaluation metric, and the results are shown in 6. It makes sense that the "retained" model gets the best performance, and at the same time, "fine-tuning" model also achieves a similar performance with an acceptable cost of time. Both these two models shows a significantly better performances than method of "online-only".

8 CONCLUSIONS

This paper introduces our strategy applied in RecSys Challenge 2017. Different approaches are designed and used to address the job recommendation task in a cold-start scenario. Our framework is proposed as a four-level integration and the key-ideas are listed in below:

Feature Integration of different levels can significantly affect the performance of a recommender system.

Wide and Deep Learning is an effective model to aggregate heterogeneous features. It also shows a superior performance compared with other content-based methods.

The situations in online tests share diversities with offline ones, where a non-over-fitted model is necessarily required. By adding new data into a already-trained model, balance is reached considering both time costs and performance.

Although different RSs hold various performances, they could be a good supplement of each other. Lessons are learned from Social Choice Theory to implement the Approach-level Integration and receive the best results.

Integration in these four levels do help a lot on cold-start recommendations.In our future study, a multi-level Deep and Wide model would be tested. In addition, other voting approaches and boosting methods will also be taken into consideration.

ACKNOWLEDGMENTS

We thank the organizers of RecSys Challenge 2017 and Xing for generously providing the opportunity and data resources for developing and testing our techniques and ideas. And special thanks to Weizhi Ma, Xiao Lin and Chong Chen for their constructive advice and discussions.

REFERENCES

[1] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.
[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.
[3] Abel Fabian, Yashar Deldjoo, Mehdi Elahi, and Daniel Kohlsdorf. 2017. RecSys Challenge 2017: Offline and Online Evaluation. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys '17)*. ACM, Como, ITALY, 2.
[4] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.
[5] Maja Pohar, Mateja Blas, and Sandra Turk. 2004. Comparison of logistic regression and linear discriminant analysis: a simulation study. *Metodoloski zvezki* 1, 1 (2004), 143.
[6] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. ACM, 285–295.
[7] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. 2017. Situational Context for Ranking in Personal Search. In *The International Conference*. 1531–1540.