Combination of Content-Based User Profiling and Local Collective Embeddings for Job Recommendation

Vasily Leksin[1], Andrey Ostapets[1], Mikhail Kamenshikov[1], Dmitry Khodakov[1], and Vasily Rubtsov[1,2]

[1] Avito, Russia
http://www.avito.ru
[2] National Research University Higher School of Economics, Russia

Abstract. We present the approach to the RecSys Challenge 2017, which ranked 7th. The goal of the competition was to prepare job recommendations for the users of the social network for business Xing.com. Our algorithm consists of two different models: Content-based User Profiling and Local Collective Embeddings. The first content-based model contains many hand-tuned parameters and data insights, so it performs fairly well on the task of the challenge despite its simplicity. The second model is based on Matrix Factorization and may be applicable to a wide range of cold-start recommendation tasks. The combination of these two models have shown the best performance on local validation.

Keywords: recommender system; cold-start problem; Local Collective Embeddings

1 Introduction

Unlike last year's competition [3] ACM RecSys Challenge 2017 focuses on the problem of a cold-start vacancy recommendations for a business social network XING. The Challenge consists of two phases. The first phase is offline evaluation: fixed historical dataset and fixed target users and items for which recommendations should be submitted. Very few target items were in Interactions data. The second phase is online evaluation: a new portion of target users and items was released every day. The recommendations submitted at the online stage were rolled out in XING's live system and pushed to real users. Both phases aimed at achieving the following tasks: given a new job posting the goal was to identify those users who (a) might be interested in getting the job posting as a push notification with recommendation and (b) were also proper candidates for the given job position. For online and offline evaluation, the same metrics and the same structure of datasets were used. The top teams from the offline phase were allowed to participate in the online evaluation. Online part determined the winners of the challenge.

The task of offline phase was standard cold-start recommendation problem while the online stage was focused on the push notifications and was designed by the following scheme:

- Every 24 hours teams needed to send a submission to the organizers;
- For each job posting, a team needed to provide a list of ≤ 250 users;
- Each user could receive ≤ 1 push per day.

The recommendations were delivered to the users through the following channels: activity stream in the mobile applications, jobs marketplace, emails, and recruiter tools. Some challenges that the participating teams needed to solve:

- Balance between user interests and the interests of recruiters
- Balance between relevance and revenue
- Item cold-start problem
- Smart targeting of push recommendations

2 Data Description

In the challenge, the semi-synthetic dataset from XING was provided: artificial users and events were added, text tokens from job postings titles and descriptions were replaced by numeric IDs and random noise was added in order to anonymize the data.

The organizers have provided datasets:

- **Interactions** — interactions that the user performed on the job posting items (clicked, bookmarked, replied, recruiter interest or deleted) as well as details about items shown to users by the existing recommender (impressions).
- **Users** — details about those users who appear in the above datasets: job roles, career level, discipline, industry, location, experience, and education.
- **Items** — job postings details that were/will be recommended to the users: title, career level, discipline, industry, location, employment type, tags, created time and flag if item was active during the test.

On the online phase datasets contained 1M users, 853K items, 88.7M impressions, 4.2M interactions (84% clicked, 8.5% deleted, 4.3% bookmarked, 2.3% replied, 0.8% recruiter interest). Every 24 hours the organizers provided a new portion of target users and items for prediction.

3 Evaluation Measure

Let T be a list of target items. For each target item $i \in T$ a recommender selects users to whom that item is pushed as a recommendation. Leaderboard score is computed as follows:

$$S = \sum_{i \in T} s(i, r(i)). \tag{1}$$

Here, recommendations $r(i)$ specifies the list of users who will receive the given item as a push recommendation.

Let $U_r = r(i)$ be a list of recommended users. The score function $s(i, U_r)$ sums up the success rates of the users U_r and the item-based success rate:

$$s(i, U_r) = \text{itemSuccess}(i, U_r) + \sum_{u \in U_r} \text{userSuccess}(i, u), \tag{2}$$

where userSucess scores a user-item pair according to the user interactions:

$$\text{userSuccess}(i, u) = \rho(u, r) * \text{premiumBoost}(u), \tag{3}$$

where ρ is event type weight and premiumBoost is a boost coefficient for premium user

$$\rho(u, r) = \begin{cases} -10, \text{if delete only,} \\ 1, \text{if clicked,} \\ 5, \text{if bookmarked or replied,} \\ 20, \text{if recruiter interest,} \end{cases} \tag{4}$$

$$\text{premiumBoost}(u) = \begin{cases} 2, \text{if the user is premium,} \\ 1, \text{otherwise.} \end{cases} \tag{5}$$

$$\text{itemSuccess}(i, U_r) = \begin{cases} w(i), \text{if} \quad \exists u \in U_r : \text{userSuccess}(i, u) > 0 \\ 0, \text{otherwise,} \end{cases} \tag{6}$$

where w is a boost coefficient for a paid item.

$$w(i) = \begin{cases} 50, \text{if the item is paid,} \\ 25, \text{otherwise.} \end{cases} \tag{7}$$

Evaluation measure shows the business value of different types of events from different types of users and different types of vacancies.

For local validation in the first phase we used the last week of interactions. In the second phase, we used *userSuccess* score as a local score and recommendations were built only for those users and items that were present in the all currently available data from the daily chunks.

4 Models

In our solution, we used two different approaches: Content-Based User Profiling and Local Collective Embeddings [6].

4.1 Content-Based User Profiling

Our first content-based [4] model is based on ranking user-item pairs using content information from a user profile and user interaction history. The motivation for the content-based approach is below: we tried experimenting with classical XGBoost, but it delivered rather bad results; so, we built a simple algorithm where each feature has its own weight and it was possible to understand how the score for each prediction is formed.

Let U be a set of target users and I be a set of target items. For each pair (u_k, i_k) we check whether it has matches between User Job Title and Item Job Title (or tags) to reduce the number of the observed pairs. Then we apply following rankers to make top-N recommendations.

- **User Title ranker** — firstly, we calculate 3 types of inverse document frequency for each token as

$$F_t = \log(\frac{\#\text{unique tokens}}{\#\text{token occurrences}}) \tag{8}$$

  where UF_t stands for User Title, IF_t stands for Item Title, and TF_t stands for Item Tags. For each user u we calculate the token weight as:

$$\text{score}_t = \frac{20 * UF_t * IF_t * TF_t}{\sqrt{|u|}} \tag{9}$$

  where $|u|$ is a number of tokens in the user's title. The final ranker score for a pair (u_k, i_k) is calculated as a sum of token scores in intersection between User Title and Item Title (tags).

$$\text{score}(u, i) = \sum_t score_t \tag{10}$$

- **User Interest ranker** — this ranker is quite similar to User Title ranker described above, but instead of scoring tokens from User Job Title, we score tokens from items target user interacted with. This scoring includes Item Job Title and Item Job Tags with weights based on the interaction type.
- **Industry, Discipline and Career Level rankers** — those rankers also use information from both user profiles and user interactions. The Profile Career Level ranker is based on the Career Level difference between user and item: $CLD(u, i) = |u_{CL} - i_{CL}|$

$$w_{CL}(u, i) = \begin{cases} 1.2, if CLD(u, i) \leq 1 \\ 0.7, if CLD(u, i) = 3 \\ 0.5, if CLD(u, i) \geq 4 \end{cases} \tag{11}$$

$$\text{score}(u, i) = w_{CL} * \text{score}(u, i) \tag{12}$$

  As the features Industry and Discipline are categorical, we multiply the pair score by $w > 1$ having exact match, and by $w < 1$ otherwise. Interactions

rankers are slightly different from the profile ones, and the main purpose of those rankers is to modify the score according to the ratio of parameter matches and the total user interactions.
- **User Behavior ranker** — different users have different behavior patterns. That ranker is based on user interaction history. If the user has all positive interactions, we add a constant to his score, if the user has all negative interactions, we subtract a constant from his score. Also, we multiply user score by a $w < 1$, which is based on the positive/negative interactions ratio.
- **Premium Ranker** is a simple ranker that multiplies the score by 1.1 for a premium user and by 1.05 for a premium item. Weights were first set manually based on our expertise and then they were optimized on the local validation set using a parameter grid.

Final predictions were made by applying all rankers to user-item pairs, thresholding, and taking top-100 users for each item.

4.2 Local Collective Embeddings

Local Collective Embeddings (LCE) [6] is a matrix factorization model that exploits items' properties and past user preferences while enforcing the manifold structure exhibited by the collective embeddings. This model addresses the cold-start problem. The motivation for LCE approach is as follows: matrix factorization and topic modeling are well-proven methods, but in this challenge, we faced the necessity of a hybrid solution, which is LCE.

Assume that we have:

- a set of m properties stored in a matrix $X_s \in \mathrm{R}^{n \times m}$, where a row corresponds to an item and a column to an item property;
- a set of u users stored in a matrix $X_u \in \mathrm{R}^{n \times u}$, where a cell $(i; j)$ indicates whether the user j has shown interest to item i.

At the test time, we are given a new item q with description $q_s \in \mathrm{R}^{1 \times m}$, and our goal is to predict $q_u \in \mathrm{R}^{1 \times u}$, i.e., to score how likely is the user to show interest to the new item.

Define LCE optimization problem:

$$\min : J = \frac{1}{2}[\alpha||X_s - WH_s||^2 + (1-\alpha)||X_u - WH_u||^2 + +\lambda(||W||^2 + ||H_s||^2 + ||H_u||^2)]$$

s.t.

$$W \geq 0; Hs \geq 0; Hu \geq 0.$$

To regularize this matrix factorization model we use manifold assumption: if two data points x_i and x_j , in any view, are close in the intrinsic geometry of the distribution, then their representations in the low-dimensional space should also be close to each other. Let's construct a graph with n nodes where each node represents a data point. For each point, we find the p nearest neighbors and we connect the corresponding nodes in the graph.

This results in a matrix A which can later be used to measure the local closeness of two points x_i and x_j. Using the above defined weight matrix A we may measure the smoothness of the low dimensional representation as:

$$S = \frac{1}{2}\sum_{i,j=1}^{n} ||w_i - w_j||^2 A_{ij} = \sum_{i=1}^{n}(w_i^T w_i)\mathbf{D}_{ii} - \sum_{i,j=1}^{n}(w_i^T w_j)\mathbf{A}_{ij} =$$

$$= Tr(W^T DW) - Tr(W^T AW) = Tr(W^T LW),$$

where D is a diagonal matrix which entries are the row sums of A (or column, as A is symmetric), i.e., $D_{ii} = \sum A_{ij}$; $L = D - A$ is called the Laplacian matrix of the graph [10] and $Tr()$ is the trace operator.

This leads to the following optimization problem:

$$\min : J = \frac{1}{2}[\alpha||X_s - WH_s||^2 + (1-\alpha)||X_u - WH_u||^2 +$$

$$+\beta Tr(W^T LW) + \lambda(||W||^2 + ||H_s||^2 + ||H_u||^2)]$$

s.t.

$$W \geq 0; Hs \geq 0; Hu \geq 0,$$

where L is the Laplacian matrix of the graph, and β is a hyper-parameter which controls the extent to which locality is enforced.

Learning algorithm have the following Expectation Maximization (EM)[1] update rules:

$$W \leftarrow W \odot \frac{[\alpha X_s H_s^T + (1-\alpha)X_u H_u^T + \beta AW]}{[\alpha W H_s H_s T + (1-\alpha)W H_u H_u^T + \beta DW + \lambda W]}$$

$$H_s \leftarrow H_s \odot \frac{[\alpha W^T X_s]}{[\alpha W^T W H_s + \lambda H_s]};$$

$$H_u \leftarrow H_u \odot \frac{[(1-\alpha)W^T X_u]}{[(1-\alpha)W^T W H_u + \lambda H_u]};$$

where $\frac{\bullet}{\bullet}$ denotes the element-wise matrix division operator.

Figures 1-3 show the results of optimizing two main model parameters: the number of EM-iterations and the number of factors. Figure 1 shows that the objective function converges after 25 iterations. From the Figures 2 and 3, we can see that the increase in the number of factors continues to improve the local score, even after the objective function has converged.

After training LCE model, we train linear regression models on user features where target is corresponding user factor. Finally, we have as many linear regression models as a number of factors. Final recommendations are built based on weighted sum of LCE obtained factors and factors predicted by user features. This technique allowed to improve by 10% on the local score.

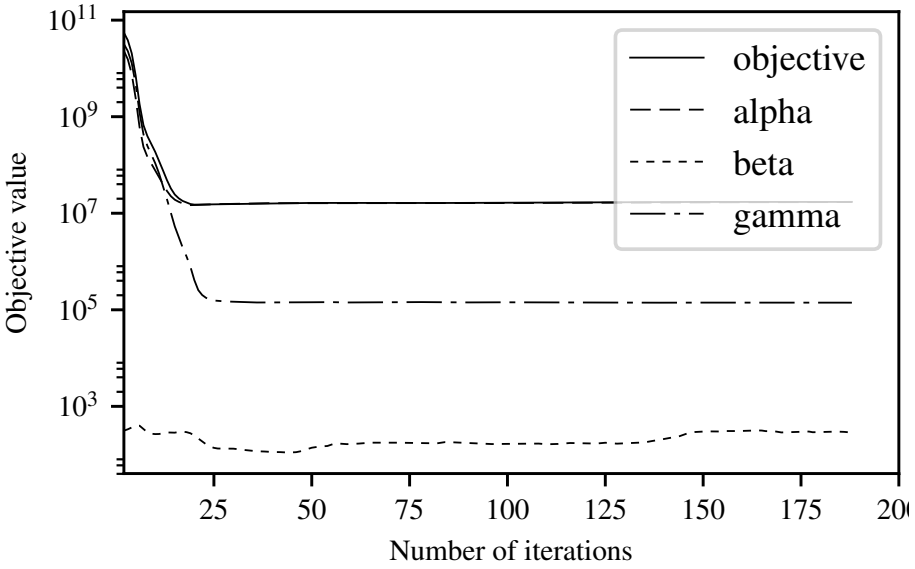

Fig. 1. EM convergence.

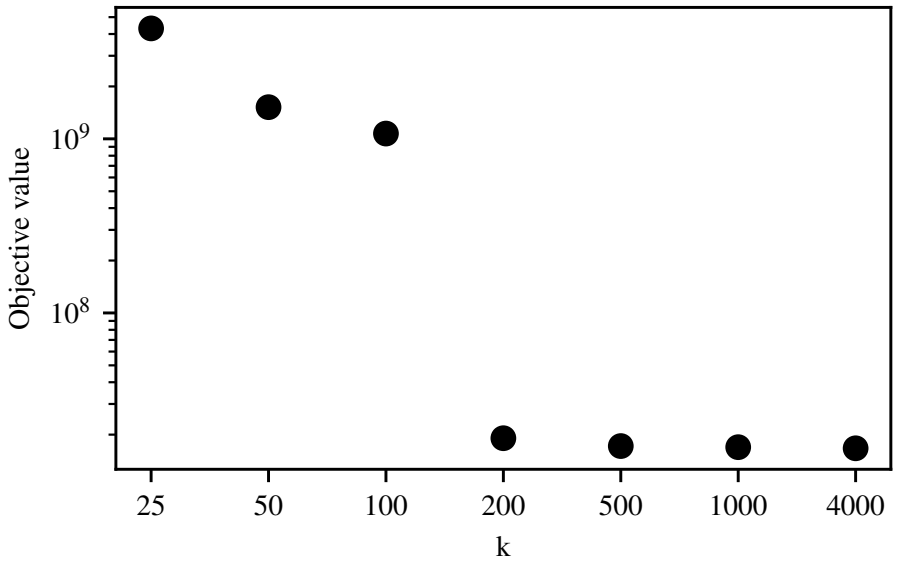

Fig. 2. Optimizing k (objective function)

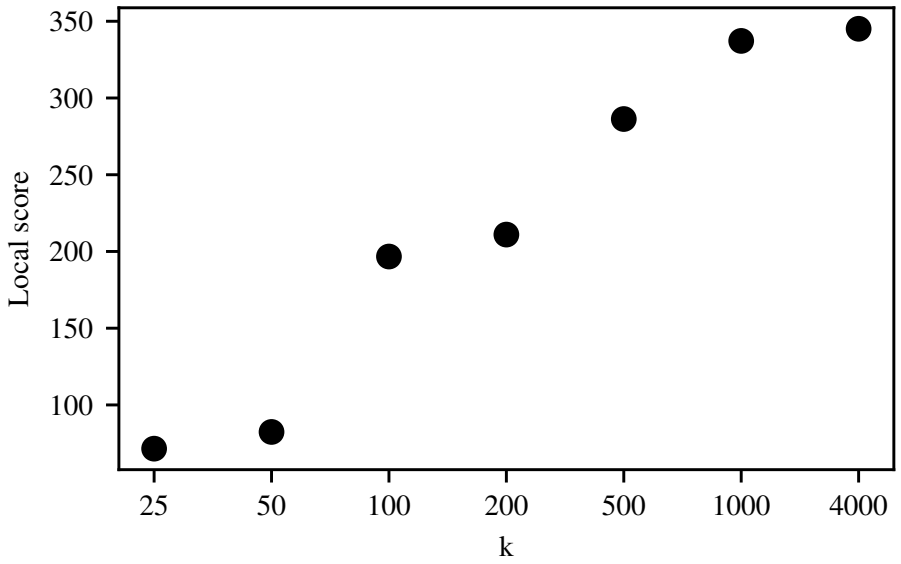

Fig. 3. Optimizing k (local score)

4.3 Blending

After parameter tuning we had local score of 649 for Content-Based User Profiling and 405 for LCE. We generated 500 recommended items for each user from both models. Then we got the final score for user-item pair as a weighted sum of scores of two models:

$$\text{score}_{final} = 0.8 * \text{score}_{LCE} + \text{score}^{0.15}_{content-based}.$$

The coefficient choice for blending was based on the distribution of scores. The content-based algorithm produced scores from 0 to 20K (approximately). LCE scores were in 0-1 range. Therefore, the content-based model score was raised to power $p < 1$. Then the exact values of the parameters were selected on the local validation set. This fine-tuning scheme of two rankers by their linear combination is similar to the scheme from [2]. The local score for the mixed model was 702. It means that the LCE model allowed to increase the local score by 8.1%.

5 Conclusions

In our approach, we used two different models: Content-based User Profiling and Local Collective Embeddings. In this Challenge, the content-based approach have shown better results on local validation than the Local Collective Embeddings. A possible reason for the fact that the more sophisticated LCE algorithm has lost in quality to the simpler content-based method is that LCE was not

customized enough for the task of competition. In particular, it was trained on binary data and did not take into account the different types of features (numeric, categorical, binary). However, the composition of the two algorithms had a better score.

The underlying rules leading to the observed results are as follows: within the content-based approach it is very important to understand how the score for each prediction is formed. In the Matrix Factorization approach [5], it is very important to choose an appropriate generative model which can include user and item side features, for example, an extension on LCE. Further improvement of the LCE algorithm may incorporate the user features to the model (as an additional term similar to the item features) as well as different probability distributions for binary, categorical and numeric features.

The second phase of the competition can be considered as an A/B test. Obviously, it is hard to carry out for organizers (it is necessary to split into groups correctly, collect solutions from participants, etc.), but it may stimulate participants since they can compete in real-time and observe how the algorithm shows itself comparing to other algorithms. Daily feedback in the online phase was too rare to use it for automatic parameter tuning or multi-armed bandits approach. In the offline stage, it was also important not to use features which we would not be able to use in the online stage.

The main scientific contribution of the paper answers the question of how to use Local Collective Embeddings with user's side features. Originally, LCE was not designed to include user features.

References

1. Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. The Elements of Statistical Learning. Springer New York Inc., New York, NY, USA, Chapter 8.5 The EM algorithm, 236243.
2. Dmitry I Ignatov, Sergey I Nikolenko, Taimuraz Abaev, and Jonas Poelmans. 2016. Online recommender system for radio station hosting based on information fusion and adaptive tag- aware profiling. Expert Systems with Applications 55 (2016), 546558.
3. Vasily Leksin and Andrey Ostapets. 2016. Job Recommenda- tion Based on Factorization Machine and Topic Modelling. In Proceedings of the Recommender Systems Challenge (RecSys Challenge 16). ACM, New York, NY, USA, Article 6, 4 pages. https://doi.org/10.1145/2987538.2987542
4. Francesco Ricci, Lior Rokach, and Bracha Shapira (Eds.). 2015. Recommender Systems Handbook. Springer, Chapter 4 Semantics- Aware Content-Based Recommender Systems, 119159. https://doi.org/10.1007/978-1-4899-7637-6
5. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. 2000. Application of Dimensionality Reduction in Recommender System A Case Study. In IN ACM WEBKDD WORKSHOP.
6. Martin Saveski and Amin Mantrach. 2014. Item Cold-start Rec- ommendations: Learning Local Collective Embeddings. In Pro- ceedings of the 8th ACM Conference on Recommender Sys- tems (RecSys 14). ACM, New York, NY, USA, 8996. https://doi.org/10.1145/2645710.2645751