IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 January 2023, accepted 16 February 2023, date of publication 27 February 2023, date of current version 3 March 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3249356*

APPLIED RESEARCH

# Comparison of Real-Time and Batch Job Recommendations

**ROBERT KWIECIŃSKI[1,2], GRZEGORZ MELNICZAK[2], AND TOMASZ GÓRECKI[1]**
[1]Faculty of Mathematics and Computer Science, Adam Mickiewicz University, 61-614 Poznań, Poland
[2]OLX Group, 61-872 Poznań, Poland

Corresponding author: Robert Kwieciński (r.kwiecinskipl@gmail.com)

**ABSTRACT** Collaborative filtering recommendation systems are traditionally trained in a batch manner and are designed to produce personalized recommendations for a large number of users at the same time. However, in many industrial use cases, it is reasonable to produce recommendations in real time, taking account of very recent user interactions. In this work, we present the implementation of batch and real-time recommendation systems using the example of the RP3Beta model, a simple scalable graph-based model that outperforms multiple more advanced models. Our approach can be utilized by any recommendation system if user-to-item recommendations can be obtained based on item-to-item recommendations. We show that it covers multiple common recommendation models, especially collaborative filtering approaches where user features are not available. We also provide the results of A/B tests comparing these two approaches in a real-world scenario of a job recommendation task, conducted with almost 200,000 OLX users. We report at least 10% more users applying for recommended job ads when using a real-time instead of a batch approach. We believe that our results can help other organizations to take informed decisions about whether to make the effort of moving from a batch to a real-time recommendation setting.

**INDEX TERMS** A/B tests, collaborative filtering, job recommendations, real-time recommendations, RP3Beta.

## I. INTRODUCTION

Recommendation systems are widely utilized by the largest internet services, including Twitter [1], Netflix [2], and Amazon [3]. They usually make use of information about previous interactions between users and items not only when training the models, but also when producing recommendations for users. It has been observed that the most recent user interactions are usually the most important for producing recommendations [4].

Collaborative filtering [5] is a very common recommendation technique, and provides recommendations to users based solely on their interactions. Such systems are not able to provide recommendations to new users until their interactions have been processed. Hence, the inclusion of recent user interactions in the recommendation process is even more important for such models.

The associate editor coordinating the review of this manuscript and approving it for publication was Claudio Zunino.

Recommendation systems usually apply processes consisting of two phases: training the model and generating the recommendations [6]. We can distinguish three types of approach with respect to their behaviour when the user performs an action (e.g., visits or purchases some item):

1) the model IS NOT retrained and the user recommendations ARE NOT affected,
2) the model IS NOT retrained, but the user recommendations ARE affected,
3) the model IS retrained and the user recommendations ARE affected.

The first case, which we refer to as *batch recommendations* [7], is the cheapest and the easiest to develop. This solution is good when a large number of recommendations are generated at the same time (e.g., for sending email recommendations to all users). Such recommendations can also be presented to users on demand, but the user experience might be worse because the recommendations do not take

account of the most recent user interactions (i.e., interactions which happened after the last recalculation of the model).

In the second case, which we refer to as *real-time recommendations* [8], the most recent user interactions impact the recommendations for the user in question, but do not impact the recommendations produced for other users (because the model is not updated). For instance, the model may consist of item embeddings learned once a day, whereas the user representation can be calculated on demand as an average of the representations of the items with which the user interacted. Unlike in the first case, such approaches address concept drifts of users and are able to provide recommendations for new users immediately after their first interaction. In the case of collaborative filtering, these methods usually suffer from a cold-start problem for items.

In the third case, the model constantly learns from a continuous stream of interactions. Such approaches, known as *stream-based recommender systems* [6], [9], require the greatest engineering effort. Even in the case of collaborative filtering, these methods are able to recommend an item after some user has interacted with it [10].

OLX[1] is a classified platform, where users post their advertisements regarding jobs, real estate, services, and goods. It is a part of the OLX Group headquartered in Amsterdam with above 10 thousand employees, which is a part of Naspers[2] owned by Prosus.[3] The OLX brand, founded in 2006, is present in around fifteen countries and serves hundreds of millions of users monthly. Advertisements are usually available only for a few weeks, and users are often anonymous (i.e., the only information we have about them is their browsing history), which creates a challenging environment for the problem of recommending relevant ads to the users.

In this work, we compare the batch and real-time recommendation approaches as deployed at OLX for job recommendations. Our main motivation for implementing real-time recommendations infrastructure was the need of providing recommendations to the users just after their first interaction with job ads and improving the quality of the recommendations after every new interaction. We do not consider stream-based models, because of their high implementation costs relative to the improvement that we expect. We believe that the abilities of stream-based approaches to address the problems of *concept drift* and *cold-start for items* [7] are less important in the case of our job recommendations than in other domains (e.g., news recommendations [10]). Below we list our main contributions.

- We outline the required infrastructure to provide real-time recommendations by presenting the whole pipeline from data acquisition to serving the recommendation to the users. Such a solution was successfully deployed at multiple markets of the OLX.
- We describe the process of calculating the recommendations from the RP3Beta model within our real-time recommendations infrastructure.
- We report the improvement of the real-time RP3Beta approach over the batch RP3Beta approach by providing the results of A/B tests conducted with OLX users. To our knowledge, there is no research providing an online comparison of batch and real-time recommendations.
- We show that our solution can be generalized for recommendation models that produce user-to-item recommendations by aggregating the item-to-item recommendations of the items that the user has interacted with.

The rest of the paper is organized as follows. Section II presents related work, including the mathematical formulation of the RP3Beta model. Section III presents the details of our implementation of the RP3Beta model in batch and real-time settings. Section IV presents the results of A/B tests conducted with users of OLX. Conclusions are presented in section V.

## II. RELATED WORK

### A. RECOMMENDATION SYSTEMS IN THE INDUSTRY

Recommendation methods for millions of customers and millions of items are successfully utilized in the industry for at least 25 years. In 1998, Amazon launched item-based collaborative filtering [3], [8]. Recommendation systems gained attention in 2007 when Netflix's organized the Netflix Price competition and offered one million dollars for creating a recommendation model outperforming the Netflix algorithm by 10% in terms of RMSE [2]. In 2016, Netflix described their recommendation systems and announced that 80% of hours streamed at Netflix come from recommendations [11].

Within the last few years, we observe industrial applications of more advanced recommendation techniques. In 2018, Pinterest described Pinsage [12], a graph convolutional neural network model trained on a graph with 3 billion nodes and 18 billion edges, which was deployed at Pinterest and outperformed previous models. In 2022, Pinterest published work describing ItemSage, a new recommendation method based on Pinsage and transformer architecture, which deployment increased Gross Merchandise Value (GMV) by up to 7% [13]. Lacic et al. [14] published the results of A/B tests conducted on the Austrian job platform Studo Jobs to evaluate the performance of utilizing embeddings for real-time job recommendations. In 2020, LinkedIn reported a significant improvement in their recommendation system achieved by utilizing deep transfer learning for creating domain-specific job understanding models [15].

In our previous work [16] we reported the impact of sending email and push notifications with job recommendations to millions of OLX users actively looking for a job. We observed significantly more users applying for jobs when sending them

[1]https://www.olxgroup.com/brands/olx
[2]https://www.naspers.com/
[3]https://www.prosus.com/

recommendations from the ALS model, and even more when we used the RP3Beta model.

### B. BATCH AND REAL-TIME RECOMMENDATIONS

Viniski et al. [7] noted that *recommender systems are traditionally trained in a batch fashion* and examined several popular examples, such as SVD [17], BPRMF [18] and NeuMF [19]. Sharma et al. [1] described the evolution of recommendation methods at Twitter and the process of supplementing batch with real-time processing. They mentioned that around 2012 *nearly all Twitter graph-based recommendations were generated in batch at roughly daily intervals*. They observed that the system performed well for users whose recommendations had been recently recomputed, which supported their intuition that *day-old batch recommendations did not seem to exploit the advantages of Twitter*.

Linden et al. [8] proposed an *item-to-item collaborative filtering* approach for serving personalized real-time recommendations on a large scale, and deployed the solution at Amazon. Their algorithm for each item calculates a list of similar items in an offline phase. The personalized user recommendations are provided by aggregating the lists of items similar to the items with which a given user interacted. Such an approach is widely applied [6], because calculating item similarities is an essential part of many popular recommendation algorithms [20], [21], [22].

Several studies have been carried out to compare batch or real-time recommendations against stream-based recommendations [7], [23]. To our knowledge, there is no research comparing batch and real-time recommendations. In this work, we fill that gap by providing the results of an A/B test that we conducted. We hope that our results can help other organizations to make informed decisions on whether to make the effort of moving from a batch to a real-time recommendation setting.

### C. RP3Beta MODEL

In this section, we present the RP3Beta model proposed by Paudel et al. [22]. Following Dacrema et al. [24] we treat this model as a generalization of P3Alpha [25] and directly calculate the scores instead of random walk approximations. In our previous work [16] we showed that RP3Beta outperformed other popular collaborative filtering approaches (namely, ALS [26], LightFM [27], Prod2Vec [20], [28], and SLIM [21]) on the OLX Jobs Interactions dataset. The RP3Beta model is currently deployed at OLX.

Let us denote the set of users by $\mathcal{U}$ and the set of items by $\mathcal{I}$. We represent our data as a bipartite graph in which the parts are users and items, while edges represent interactions between them. Let $\mathcal{N}(x)$ be the set of neighbours of the node $x$.

The model calculates a score $r_{ui}$ representing the relevance of item $i \in \mathcal{I}$ for user $u \in \mathcal{U}$, as the sum of the scores assigned to the paths of length 3 connecting $u$ and $i$, i.e.:

$$r_{ui} = \sum_{i' \in \mathcal{N}(u)} \sum_{u' \in \mathcal{N}(i')} p(u, i', u', i),$$

where $p(u, i', u', i)$ is the score assigned to the given path. The score of the path is calculated as the product of the scores of the edges:

$$p(u, i', u', i) = p^{(1)}_{ui'} p^{(2)}_{i'u'} p^{(3)}_{u'i},$$

where $p^{(1)}_{ui'} = \frac{1}{|\mathcal{N}(u)|^\alpha}$, $p^{(2)}_{i'u'} = \frac{1}{|\mathcal{N}(i')|^\alpha}$ and $p^{(3)}_{u'i} = \frac{1}{|\mathcal{N}(u')|^\alpha |\mathcal{N}(i)|^\beta}$.

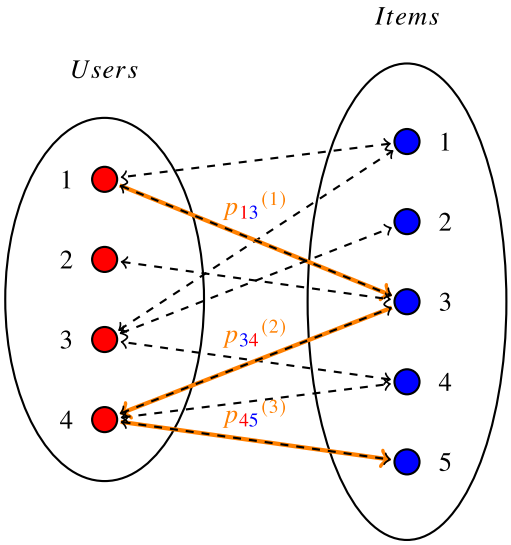


**FIGURE 1.** Path of length 3 with edge scores. The path is highlighted as a bold orange line. Dashed lines represent interactions between users and items.

The edge scores of a given path are illustrated in Fig. 1.

We can efficiently calculate these scores by representing the model in the matrix form:

$$\boldsymbol{R} = \boldsymbol{P}^{(1)} \boldsymbol{P}^{(2)} \boldsymbol{P}^{(3)}, \tag{1}$$

where $\boldsymbol{R} = (r_{ui})$ and $\boldsymbol{P}^{(k)} = (p^{(k)}_{xy})$. Note that $\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(3)}$ are $|\mathcal{U}| \times |\mathcal{I}|$ matrices and $\boldsymbol{P}^{(2)}$ is an $|\mathcal{I}| \times |\mathcal{U}|$ matrix.

The RP3Beta model recommends the items with the highest score excluding the items with which the user has interacted.

## III. MODELS

In this section, we present batch and real-time versions of the RP3Beta model used at OLX. We describe the real-time infrastructure in detail. We also add a remark regarding the utilization of the same concept for other recommendation models.

### A. BATCH RP3Beta MODEL

In the batch approach, we directly produce and store the recommendations for each user based on (1). This is a reasonable approach for sending emails to users when a huge

number of users receive recommendations at the same time. It can also be used if we need to produce recommendations at the user's request. To achieve this, we deployed an API which returns a list of recommendations based on the user identifier. However, these recommendations are generated during the batch process (several times a day) and may not take account of the most recent user interactions. The greatest advantage of this approach is its simplicity and low cost compared with real-time recommendations.

### B. REAL-TIME RP3Beta MODEL

The scores defined in (1) can be calculated in the following way:

- Calculate matrix $\boldsymbol{A} = \boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$, which is an $|\mathcal{I}| \times |\mathcal{I}|$ sparse matrix. This is done in batch mode – several times per day.
- Produce the recommendations for a given user by multiplying the user representation (which is the row from matrix $\boldsymbol{P}^{(1)}$ representing the given user) by the pre-calculated matrix $\boldsymbol{P}^{(2)}\boldsymbol{P}^{(3)}$. This is done on demand when the recommendations for a given user need to be displayed.

Since the matrix $\boldsymbol{A}$ is calculated in batch mode, we are not able to take account of information about new items which have had interactions since the matrix was calculated for the last time. As a result, if all items with which a user has interacted are very new, then we cannot provide any recommendations to that user. On the other hand, recommendations based on these items might not be accurate anyway, because our collaborative filtering approach would be based on a small number of users who had interacted with them. Hence, we believe that calculating this part of equation (1) in batch mode instead of real-time mode has a low impact on the overall quality of the recommendation system.

The number of nonzero entries of the matrix $\boldsymbol{A}$ is twice the number of distinct item pairs visited by the same user. It might easily exceed billions in real-world applications. Hence, to speed up the calculation of recommendations, we decided to restrict each column of the matrix $\boldsymbol{A}$ to the $N$ largest values. By choosing $N \geq |\mathcal{I}|$ we do not make any modifications to the matrix $\boldsymbol{A}$. In Figure 2 we can observe how the precision@10 [29] depends on the choice of $N$ in the case of our dataset with around $10^5$ items. We report precision because we believe it is highly correlated with our business metrics (similarly to [30]). Additionally, we observed a similar behaviour of other offline metrics depending on $N$ (including recall, NDCG, MAP, MRR, LAUC, HR [29]). Based on this, we decided to choose $N = 60$.

### C. REAL-TIME RP3Beta ARCHITECTURE

The high-level architecture used to efficiently serve personalized real-time user-to-item recommendations is presented in Figure 3.

User interaction data is constantly being streamed into and is stored in two distinct systems – a big-data platform and a fast key-value store containing the recent user-item

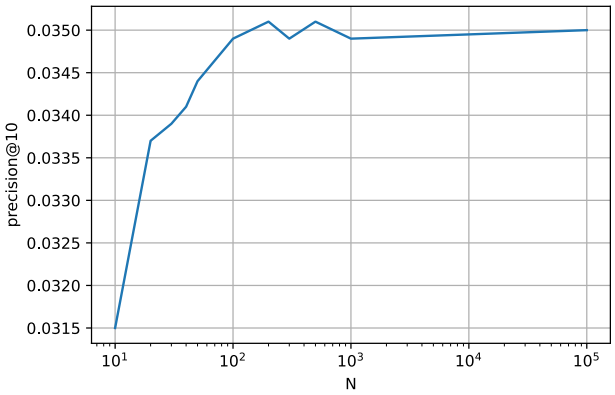


**FIGURE 2.** Precision@10 after restricting each column of matrix $\boldsymbol{A}$ to the $N$ largest values, for $N \in \{10, 20, 30, 40, 50, 100, 200, 300, 500, 1000, 10^5\}$.

interactions. The big-data platform is used to query and aggregate a high volume of data used as input for the batch item-to-item prediction step, the output of which is again stored in a fast item key-value lookup of $N$ similar items along with the similarity scores.

The *interaction store* – depending on implementation – can reflect the most recent user behaviour in almost real time, while the *item-to-item recommendation store* is updated only after completion of the batch prediction step.

When requesting personalized recommendations for a user, the *aggregator* component retrieves up to $M$ most recent interactions $\mathcal{I}_u = \{i_k : k \leq M\}$ from the *interaction store*. Then for each of the unique returned items a list of similar items $S(\mathcal{I}_u) = \left\{\left\{\left(i', s_{ii'}\right) : i' \in \mathcal{S}_N(i)\right\} : i \in \mathcal{I}_u\right\}$ is fetched from the *item-to-item recommendation store*, where $s_{xy}$ is the similarity score between $x$ and $y$, and $\mathcal{S}_N(x)$ is the set of the $N$ items most similar to item $x$ (excluding $x$ itself). Since an item $i'$ can appear multiple times with different similarity scores (for different $i$), the final result needs to be flattened and aggregated, which is described by Algorithm 1. Note that instead of *sum* we could use any aggregate function, e.g., *max, mean, median* (and their weighted versions).

**Algorithm 1** Calculations Performed by the *aggregator* Component

```
R = dict()
for item ∈ I_u do
    for similar_item, similarity_score ∈ S_N(item) do
        append similarity_score to R[similar_item]
    end for
end for
for similar_item, similarity_scores ∈ R do
    R[similar_item] = sum(similarity_scores)
end for
```

Using this setup allows the retrieval of personalized user-to-item recommendations within seconds of a user interaction, with the list being continuously updated as the user reacts to the recommendations presented.

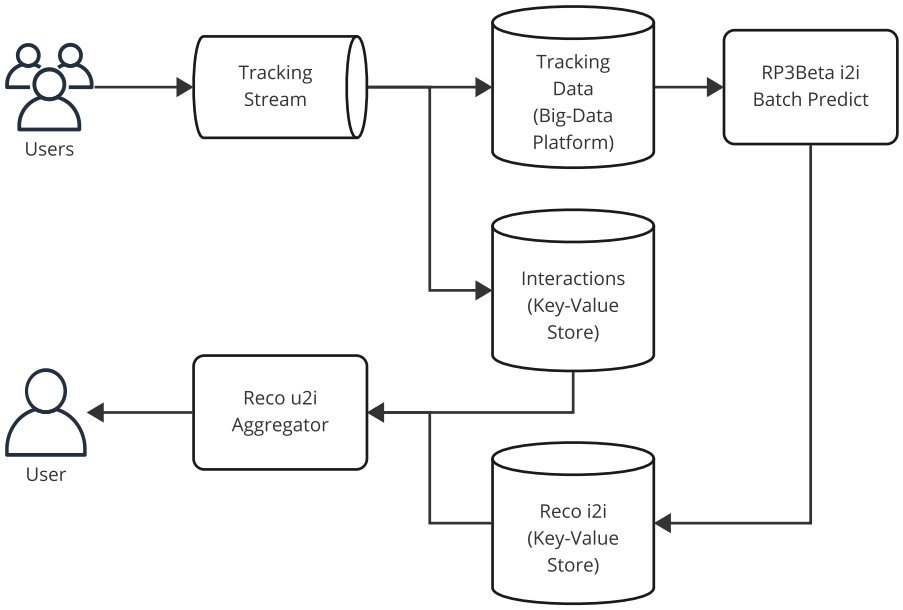


**FIGURE 3.** Real-time recommendations architecture at OLX.

### D. REMARK ON APPLICABILITY TO OTHER RECOMMENDATION MODELS

Most recommendation models, not necessarily graph-based, are able to provide item-to-item recommendations in a batch mode (for all the items at the same time). Hence we can utilize the architecture described in subsection III-C to provide real-time recommendations based on the user's latest interactions with these items. For some models, such recommendations may not be the same as the user-to-item recommendations produced by the model in a usual way (defined within the model). For instance, in matrix factorization methods [31], the recommendations are calculated based on user embeddings learned during the training procedure. Two users with the same set of interactions might have different embeddings, hence different recommendations produced in the usual way. In our approach, such users would receive exactly the same recommendations.

Recommendations produced by our approach are identical (for large enough values of $M$ and $N$) to recommendations produced in a usual way for models where user-to-item recommendations can be obtained by aggregating item-to-item recommendations (e.g., SLIM [21]). Below we briefly prove that this property is satisfied by models which calculate the (user, item) score as the dot product of user and item embeddings, if a user embedding is calculated as the weighted average of the item embeddings with which the user interacted (e.g., LightGCN [32]). Assume that a user $u$ interacted with items $(i_1, i_2, \ldots, i_t)$ and this user's representation is calculated as $\sum_{k=1}^{t} w_k \boldsymbol{e}_{i_k}$, where $w_1, w_2, \ldots, w_t$ are weights and $\boldsymbol{e}_i$ is the embedding of item $i$. Then the score of item $i$ for user $u$ is calculated as:

$$r_{ui} = \left(\sum_{k=1}^{t} w_k \boldsymbol{e}_{i_k}\right) \cdot \boldsymbol{e}_i = \sum_{k=1}^{t} w_k \boldsymbol{e}_{i_k} \cdot \boldsymbol{e}_i = \sum_{k=1}^{t} w_k s_{i_k,i},$$

where $s_{i',i}$ is the similarity score between $i'$ and $i$, and $\cdot$ is a dot product. Hence, the user-to-item recommendations can be calculated based on item-to-item similarities.

The accuracy of our approach relies heavily on the quality of item-to-item similarities. The only user-specific information taken into consideration during the prediction is user interactions. As a result, our approach may not perform well when user features are more important than user interactions. In particular, our approach can not provide any recommendations for users without interactions.

## IV. ONLINE A/B TEST

Users of OLX receive 30 personalized job recommendations on a dedicated page. The recommendations come from two types of models: collaborative filtering and content-based. The final list of recommendations is built based on these two sources, using a blending algorithm. For simplicity, we can assume that the importance of these two sources is similar. We conducted an A/B test to compare batch and real-time recommendations: in variant A the collaborative filtering recommendation model was the batch RP3Beta model, whereas in variant B it was the real-time RP3Beta model. Each user was randomly assigned to one of these variants (independently with equal probability) and could not change the variant during the experiment. The experiment lasted twelve days.

The recommendations of the batch RP3Beta model and the batch step of the real-time RP3Beta model (i.e., the calculation of matrix $\boldsymbol{A}$) were calculated several times a day, based on the last seven days of user interactions. The experiment was conducted in multiple markets (countries) in parallel and the models were trained for each market separately. In the biggest market, our training datasets consisted of around 35 million interactions performed by

**TABLE 1. Results of A/B test comparing batch and real-time RP3Beta models. The difference was calculated as the uplift of the B group over the A group for provided metrics.**

| Variant | Users | converted users | % converted users |
|---|---|---|---|
| A (batch) | 92,835 | 3,731 | 4.02% |
| B (real-time) | 92,194 | 4,081 | 4.43% |
| Difference | -0.69% | 9.38% | 10.14% |

2 million users regarding 160 thousand job ads. We published an analogous dataset from a different period on Kaggle.[4] We conducted a more comprehensive analysis of this dataset in our previous work [16].

The results of the experiment are presented in Table 1. A converted user is a user who clicked on at least one recommended job ad and then applied for that job. We observe a 10.14% higher percentage of converted users in variant B. The $p$-value of the chi-squared test [33] equals 0.00001, which confirms the statistical significance of these results. Taking into consideration that only the collaborative filtering recommendation system was changed, we were satisfied with the results and decided to replace batch RP3Beta recommendations with real-time RP3Beta recommendations at OLX.

There are two further issues worthy of mention:

1) In the batch recommendation system we additionally changed the order of RP3Beta recommendations based on the matching between user profiles and recommended ads. We have seen in the past that reranking increases the number of converted users by around 2.5%. Hence, this reranked version of the RP3Beta model was our baseline which we wanted to improve using real-time recommendations (where we did not make any reranking).

2) When conducting the experiment we made an error in the implementation of real-time recommendations, namely, we accidentally transposed matrix $\boldsymbol{A}$. It was not possible to rerun the experiment comparing batch and real-time recommendations, because it was already proven that the real-time model, even with the implementation mistake, is superior to the batch approach. Hence, we conducted a follow-up experiment comparing the wrong and fixed implementation of the real-time model. The experiment lasted 10 days. We observed 4.7% more converted users in the fixed variant.

Hence, the advantage of real-time recommendations over batch recommendations may be even greater than reported in Table 1.

## V. SUMMARY

In this work, we described the OLX implementation of the RP3Beta model in two versions: batch and real-time. We discussed the mathematical and architectural aspects of these approaches. Then we provided the results of online A/B tests conducted on OLX users. We reported that replacing the batch RP3Beta model with the real-time RP3Beta model increases the number of users replying to recommended job ads by at least 10%.

Even though our experiments were conducted using the example of the RP3Beta model, we can use any recommendation system if the recommendations for a user can be calculated based on items similar to the items with which that user has interacted. Additionally, we believe that in domains where users are more likely to change their preferences, the impact of utilizing a real-time recommendation system may be even greater.

[4] https://www.kaggle.com/datasets/olxdatascience/olx-jobs-interactions

## REFERENCES

[1] A. Sharma, J. Jiang, P. Bommannavar, B. Larson, and J. Lin, "GraphJet: Real-time content recommendations at Twitter," *Proc. VLDB Endowment*, vol. 9, no. 13, pp. 1281–1292, Sep. 2016.
[2] J. Bennett and S. Lanning, "The Netflix Prize," in *Proc. KDD Cup Workshop*, New York, NY, USA, 2007, p. 35.
[3] B. Smith and G. Linden, "Two decades of recommender systems at Amazon.com," *IEEE Internet Comput.*, vol. 21, no. 3, pp. 12–18, May 2017.
[4] R. Kwieciński, T. Górecki, and A. Filipowska, "Learning edge importance in bipartite graph-based recommendations," in *Proc. Ann. Comput. Sci. Inf. Syst.*, Sep. 2022, pp. 227–233.
[5] R. Chen, Q. Hua, Y.-S. Chang, B. Wei, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018.
[6] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel, "StreamRec: A real-time recommender system," in *Proc. ACM SIGMOD Int. Conf. Manage. data*, New York, NY, USA, Jun. 2011, pp. 1243–1246.
[7] A. D. Viniski, J. P. Barddal, A. D. S. Britto Jr., F. Enembreck, and H. V. A. D. Campos, "A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start," *Expert Syst. Appl.*, vol. 176, Aug. 2021, Art. no. 114890.
[8] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.
[9] M. Al-Ghossein, T. Abdessalem, and A. Barré, "A survey on stream-based recommender systems," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–36, May 2021.
[10] M. Jugovac, D. Jannach, and M. Karimi, "StreamingRec: A framework for benchmarking stream-based news recommenders," in *Proc. 12th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2018, pp. 269–273.
[11] C. A. Gomez-Uribe and N. Hunt, "The Netflix recommender system: Algorithms, business value, and innovation," *ACM Trans. Manage. Inf. Syst.*, vol. 6, no. 4, pp. 1–19, Jan. 2016.
[12] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Jul. 2018, pp. 974–983.
[13] P. Baltescu, H. Chen, N. Pancha, A. Zhai, J. Leskovec, and C. Rosenberg, "ItemSage: Learning product embeddings for shopping recommendations at pinterest," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2022, pp. 2703–2711.
[14] E. Lacic, M. Reiter-Haas, T. Duricic, V. Slawicek, and E. Lex, "Should we embed? A study on the online performance of utilizing embeddings for real-time job recommendations," in *Proc. 13th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2019, pp. 496–500.
[15] S. Li, B. Shi, J. Yang, J. Yan, S. Wang, F. Chen, and Q. He, "Deep job understanding at LinkedIn," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, Jul. 2020, pp. 2145–2148.
[16] R. Kwieciński, A. Filipowska, T. Górecki, and V. Dubrov, "Job recommendations: Benchmarking of collaborative filtering methods for classifieds," 2023, *arXiv:2301.07946*.
[17] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc. KDD Cup Workshop*, 2007, pp. 39–42.

[18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proc. 25th Conf. Uncertainty Artif. Intell.*, May 2012, pp. 452–461.
[19] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 1–15.
[20] O. Barkan and N. Koenigstein, "ITEM2VEC: Neural item embedding for collaborative filtering," in *Proc. IEEE 26th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Sep. 2016, pp. 1–6.
[21] X. Ning and G. Karypis, "SLIM: Sparse linear methods for top-N recommender systems," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 497–506.
[22] B. Paudel, F. Christoffel, C. Newell, and A. Bernstein, "Updatable, accurate, diverse, and scalable recommendations for interactive applications," *ACM Trans. Interact. Intell. Syst.*, vol. 7, no. 1, pp. 1–34, Dec. 2016.
[23] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen, "Streaming ranking based recommender systems," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, Jun. 2018, pp. 525–534.
[24] M. F. Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? A worrying analysis of recent neural recommendation approaches," in *Proc. 13th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2019, pp. 101–109.
[25] C. Cooper, S. H. Lee, T. Radzik, and Y. Siantos, "Random walks in recommender systems: Exact computation and simulations," in *Proc. 23rd Int. Conf. World Wide Web*, New York, NY, USA, Apr. 2014, pp. 811–816.
[26] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proc. 8th IEEE Int. Conf. Data Mining*, Dec. 2008, pp. 263–272.
[27] M. Kula, "Metadata embeddings for user and item cold-start recommendations," 2015, *arXiv:1507.08439*.
[28] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, Aug. 2015, pp. 1809–1818.
[29] Y.-M. Tamm, R. Damdinov, and A. Vasilev, "Quality metrics in recommender systems: Do we calculate metrics consistently?" in *Proc. 15th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2021, pp. 708–713.
[30] A. Mogenet, T. A. N. Pham, M. Kazama, and J. Kong, "Predicting online performance of job recommender systems with offline evaluation," in *Proc. 13th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2019, pp. 477–480.
[31] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Comput.*, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[32] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "LightGCN: Simplifying and powering graph convolution network for recommendation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, New York, NY, USA, Jul. 2020, pp. 639–648.
[33] M. L. McHugh, "The Chi-square test of independence," *Biochemia Medica*, vol. 23, no. 2, pp. 143–149, 2013.

**ROBERT KWIECIŃSKI** received the M.Sc. degree in mathematics from Adam Mickiewicz University, Poznań, Poland. He is currently pursuing the industrial Ph.D. degree in computer science with cooperation between Adam Mickiewicz University and OLX Group. Since 2018, he has been with OLX Group, where he is currently a Senior Data Scientist. His main research interest and the topic of his Ph.D. dissertation is recommender systems, which is developed and implemented for the OLX classifieds platform in the jobs category, in multiple countries in which the platform operates.

**GRZEGORZ MELNICZAK** received the M.A. degree in operations research from the Poznań University of Economics and Business, Poland, in 2007, and the M.Sc. degree in applied mathematics from Adam Mickiewicz University, Poznań, Poland, in 2009. He is currently with OLX Group as a Machine Learning Engineer with a particular interest in recommendations and distributed systems.

**TOMASZ GÓRECKI** received the M.Sc. and Ph.D. degrees in mathematics from the Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland, in 2001 and 2005, respectively, and the Habilitation degree in computer science from the Systems Research Institute, Polish Academy of Sciences, in 2015. He is currently an Assistant Professor with Adam Mickiewicz University. He is the author of over 90 scientific papers and three books. His main research interests include methods of artificial intelligence, machine learning, and time series analysis, and their applications.