

Learning edge importance in bipartite graph-based recommendations

Robert Kwociński

Faculty of Mathematics
 and Computer Science

Adam Mickiewicz University
 Uniwersytetu Poznańskiego 4
 61-614 Poznań, Poland and

OLX Group

ul. Królowej Jadwigi 43,

61-872 Poznań, Poland

Email: r.kwociński@gmail.com

Tomasz Górecki

Faculty of Mathematics
 and Computer Science

Adam Mickiewicz University
 Uniwersytetu Poznańskiego 4
 61-614 Poznań, Poland

Email: tomasz.gorecki@amu.edu.pl

Agata Filipowska

Poznań University of Economics and Business

Al. Niepodległości 10,
 61-875 Poznań, Poland

and

OLX Group

ul. Królowej Jadwigi 43,

61-872 Poznań, Poland

Email: agata.filipowska@ue.poznan.pl

Abstract—In this work, we propose the P3 Learning to Rank (P3LTR) model, a generalization of the RP3Beta graph-based recommendation method. In our approach, we learn the importance of user-item relations based on features that are usually available in online recommendations (such as types of user-item past interactions and timestamps). We keep the simplicity and explainability of RP3Beta predictions. We report the improvements of P3LTR over RP3Beta on the OLX Jobs Interactions dataset, which we published.

I. INTRODUCTION

GRAPH-BASED RP3Beta model [1] is a very strong baseline on multiple recommender systems datasets [2], [3], [4]. This relatively simple model outperformed other approaches on our published OLX Jobs Interactions dataset and is currently a state-of-the-art collaborative filtering recommender system at OLX. In this work, we propose P3LTR (P3 Learning to Rank) model which generalizes the RP3Beta model.

In RP3Beta each user and item is represented as a node of the user-item bipartite graph. The recommendations are generated based on the paths of length 3 starting from a given user. The scores of these paths are calculated based on the scores assigned to the edges of the graph. Scores of the edges are directly calculated based on the degrees of the connected nodes. Hence, there is no learning process in this approach.

In P3LTR, to better leverage the importance of the user-item relation, we learn the score of a given edge based on the features of this edge. As features, we not only use node degrees but also utilize the sequence of interactions between two given nodes. It enables us to incorporate the information that the user visited the item several times and that the user not only clicked but applied for a given job, or how recent the click was.

In this work, we propose a training procedure and a loss function for the P3LTR model. We tune, train, and evaluate RP3Beta and P3LTR models on the OLX Jobs Interactions dataset.

The paper consists of 6 sections. The second section presents a literature review and formulates a research gap addressed by this work. Section III proposes the P3LTR model and describes its advantages and relation to the RP3Beta model. Section IV describes the considered dataset and hyperparameter tuning procedure. The results of our model are discussed in Section V. Section VI presents the conclusions and future perspectives.

II. RELATED WORKS

A. Recommender systems

Most digital platforms provide more choices than the user can explore in a reasonable time. Even a perfect search engine can not resolve this problem, because it requires users to know what they are looking for and to spend time providing this information. For this reason, powerful recommendation systems are developed by multiple companies, such as Netflix [5] or Amazon [6].

We usually distinguish two categories of recommendation methods: *content-based* and *collaborative filtering*. In *content-based* models [7], [8] we utilize user and item features to provide recommendations. The history of interactions between users and items is considered from the perspective of a single user. In contrast, *collaborative filtering* techniques [9], [10] do not consider additional information about users or items but utilize the rating history of all users at the same time to provide recommendations. During the last few decades several collaborative filtering recommendation techniques have been proposed: neighborhood-based (e.g., [11], [12]), matrix factorization-based (e.g., [13], [14]), graph-based (e.g., [1], [15] or Word2Vec-based (e.g., [16], [17])).

Another category of recommendation systems, *context-aware recommendation systems* (CARS) [18], [19], utilize contextual information of user-item interactions such as time or location. We can distinguish an important subcategory of these methods, *sequence-aware recommendation systems* [20], which utilizes sequentially-ordered user-item interaction logs.

In this work, we extend a graph-based collaborative-filtering approach that does not utilize additional contextual information. Our approach is a sequence-aware recommendation method that utilizes timestamps and types of interactions.

B. Graph-based recommendations

Many graph-based recommender systems are focused on producing the item and/or user embeddings. Some of them utilize the graph structure to produce random walks which are used as an input for the model which produces embeddings. For instance, Node2vec [21], or DeepWalk [22] utilize a SkipGram model [23]. In recent years, several collaborative filtering methods based on graph convolutional neural networks have been proposed [15], [24], [25], [26].

Another type of graph-based recommendation systems directly utilizes the graph structure to calculate the scores of items, usually by utilizing a user-item bipartite graph. Cooper *et al.* [27] proposed simple and efficient P3 and P3alpha methods which outperformed more complex and computationally demanding techniques [28], [29], [30]. Paudel *et al.* [1] extended this work by proposing the RP3Beta model, an extension of P3alpha which recommends popular items less often. These methods were recently used as a benchmark by Dacrema *et al.* [2], [3] and Anelli *et al.* [4] who compared them with several state-of-the-art neural recommendation methods. P3Alpha and RP3Beta demonstrated a very good performance against other baselines and neural models. The RP3Beta model provided the most accurate recommendations on some of the considered datasets (i.e., Pinterest [31], CiteULike-a [32] and MovieLens1M [33]; on MovieLens1M authors used their own random splits). In our previous work, as yet unpublished, we showed that RP3Beta outperforms other methods on the OLX Jobs Interactions dataset. It is currently the state-of-the-art collaborative-filtering recommendations technique deployed at OLX Jobs.

C. Research gap

The RP3Beta model does not utilize any information about user-item relations. Additionally, there is no learning process that could optimize model parameters. Hence it is not possible to learn the importance of edges in the user-item bipartite graph. In this work, we fill this gap by proposing a machine learning model which generalizes RP3Beta.

III. PROPOSED METHOD

A. Model

Let \mathcal{U} be a set of users and \mathcal{I} a set of items. For each user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ let r_{ui} be the score assigned by the model to the pair (u, i) . We denote the matrix of all user-item scores by \mathbf{R} .

We represent users and items as the nodes of the bipartite graph, where edges represent interactions between users and items. Let $\mathcal{N}(x)$ represent the set of nodes connected with the node x .

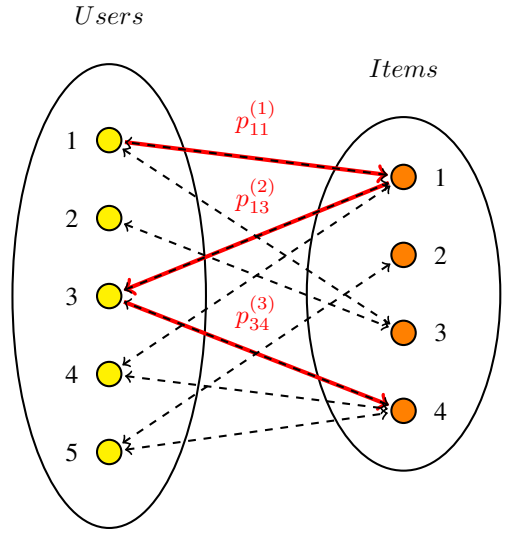


Fig. 1. Path of length 3 with edge scores. The path is highlighted by bold red line. Dashed lines represent the interactions between users and items.

For a given user $u \in \mathcal{U}$ our model recommends the items with the highest score r_{ui} , excluding the items which the user interacted with.

The score r_{ui} is calculated 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. Following the idea used in the RP3Beta model, we factorize this score as:

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

where $p_{xy}^{(k)}$ is the score assigned to the edge connecting nodes x and y in the k -th layer, $k = 1, 2, 3$. The edge scores of a given path are illustrated in Fig. 1. With this assumption, the calculation of the scores is simplified in the following way:

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

where $\mathbf{P}^{(k)} = (p_{xy}^{(k)})$, $\mathbf{P}^{(1)}, \mathbf{P}^{(3)}$ are $|\mathcal{U}| \times |\mathcal{I}|$ matrices and $\mathbf{P}^{(2)}$ is $|\mathcal{I}| \times |\mathcal{U}|$ matrix.

In this work we propose to calculate the edge scores as the function of node features f_n and edge features f_e , i.e.:

$$p_{xy}^{(k)} = \phi^{(k)}(f_x^n, f_y^n, f_{xy}^e),$$

where f_x^n is feature vector of node x , f_{xy}^e is a feature vector of the edge connecting nodes x and y and $\phi^{(k)}$ can be any real-valued function (e.g., neural network). We will call the functions $\phi^{(k)}$ **feature encoders** and propose them below.

Assume that for each (user, item) pair we know the type of interactions between them with corresponding timestamps. Let $\mathcal{E} = \{e_1, e_2, \dots, e_{|\mathcal{E}|}\}$ be a set of all possible types of interactions (e.g., click, reply, purchase).

Then we define the following features:

- $\text{deg}(x)$ – degree of the node x (number of distinct users/items which interacted with x),
- $\text{rec}(x, y)$ – number of days which passed between the most recent user (x or y) interaction with the item (y or x) and the most recent interaction of this user with any item,
- $\text{ev}(e_i, x, y)$ – number of interactions of type e_i between x and y ,
- $\text{ev}(x, y)$ – number of interactions between x and y .

Then the score is calculated as:

$$p_{xy}^{(k)} = (\text{deg}(y))^{-d^{(k)}} \cdot e^{-\text{rec}(x,y)r^{(k)}} \cdot \sigma \left(\sum_{i \in |\mathcal{E}|} \frac{\text{ev}(e_i, x, y)}{\text{ev}(x, y)} e_i^{(k)} + b_e^{(k)} \right) \cdot \sigma(\text{ev}(x, y)e^{(k)} + b^{(k)}),$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ and $d^{(k)}$, $r^{(k)}$, $e_i^{(k)}$, $b_e^{(k)}$, $e^{(k)}$, $b^{(k)}$ are model parameters.

B. Model parameters

The total number of P3LTR model's parameters equals $3 \cdot (5 + |\mathcal{E}|)$.

The RP3Beta model is not learnable and has only two hyperparameters: α and β . Our model is equivalent to RP3Beta model when $d^{(1)} = d^{(2)} = \alpha$, $d^{(3)} = \beta$ and all other parameters equal 0.

RP3Beta is a generalization of P3Alpha [27] (for $\beta = 0$), P3 [27] (for $\alpha = 1$, $\beta = 0$) and #3-Paths [27] (for $\alpha = 0$, $\beta = 0$), so naturally P3LTR also includes these methods.

Parameters $d^{(k)}$ tell us how impactful (destination) nodes of a given degree should be, i.e., $d^{(k)} = 0$ means that we treat all nodes equally, $d^{(k)} > 0$ means that we reduce the impact of nodes with a greater degree, $d^{(k)} < 0$ means that we increase the impact of nodes with the greater degree.

Parameters $r^{(k)}$ are used to utilize the recency of interactions, i.e., when $r^{(k)} > 0$ more recent interactions have higher importance, when $r^{(k)} = 0$ the recency of interactions has no impact on recommendations and when $r^{(k)} < 0$ the older interactions are more impactful.

Parameters $e_i^{(k)}$, $b_e^{(k)}$ are designed to utilize the information about type of interactions between users and items. The parameters associated with events requiring higher user engagement (e.g., replying to an offer) might have higher values than the others (e.g., the parameter associated with visiting an offer).

Parameters $e^{(k)}$, $b^{(k)}$ were introduced to include the information about the frequency of interactions. Higher frequency is an indicator of higher user engagement and might be used to increase the importance of a particular item.

C. Model training

The goal of training the model is to learn the values of the parameters of our feature encoders $\phi^{(k)}$. We describe three

components of this process: a forward pass for a single user, a training loop, and a loss function.

1) *Forward pass for a single user*: We can present our model from the perspective of a message-passing paradigm used in graph convolutional neural networks [15], [24], [25], [26]. For the initial representation ($k = 0$) we set the score $r_u^{(0)} = 1$ for the node representing the given user and the score 0 for all other nodes. Then for all nodes x and for $k = 1, 2, 3$ we perform the message passing:

$$r_x^{(k)} = \sum_{y \in \mathcal{N}(x)} r_y^{(k-1)} \phi^{(k)}(f_x^n, f_y^n, f_{xy}^e).$$

This process can be interpreted as spreading the message across the graph. At the beginning, we send the message to the neighboring nodes depending on their relevancy calculated by $\phi^{(1)}$. Then each of these nodes sends the message to their neighbors with respect to the relevancy calculated by $\phi^{(2)}$. This process could be continued, but for efficiency reasons and based on the results of Cooper *et al.* [27], we limit it to 3 steps.

2) *Training loop*: A training process is described by Algorithm 1.

Algorithm 1 Training loop of the P3LTR model.

for iteration = 1, 2, ..., iterations **do**

 Update edge weights of the graph based on feature encoders

 Set the loss to 0.

for i = 1, 2, ..., batch size **do**

 Pick a random target user (by default: random user who interacted with at least 2 items) and take his most recent interacted item as a *validation node*.

 Make a forward pass for this user and calculate the scores of top k items and the score and position of a validation node.

 Calculate the loss for this user and add it to the current loss.

end for

 Backpropagate the loss and update the weights of feature encoders.

end for

3) *Loss function*: The idea of our loss function is to score the validation item higher than the other items. Let us define

$$\text{ratio} = \frac{\text{avg score of top k items}}{\text{validation node score}}.$$

To stabilize the training, we additionally calculated the sum of squares of the parameters and multiplied it by a constant regularization parameter. We considered three loss functions:

- **ratio**: ratio + regularization term,
- **log ratio**: $\ln(\text{ratio})$ + regularization term,
- **boosted log ratio**: $\ln(\text{ratio}) \cdot \text{validation node position} + \text{regularization term}$.

The idea of the log ratio was inspired by BPR loss [34] function and the idea of the boosted log ratio was inspired

by WARP loss [35]. The best loss function was chosen during the hyperparameter optimization.

D. Model advantages

We would like to emphasize the following advantages of the proposed approach:

- 1) P3LTR generalizes RP3Beta which is a strong baseline model.
- 2) P3LTR directly utilizes the information about the user-item relationship. For instance, our model may be used for encoding the importance of ratings in the explicit feedback dataset used for the top N recommendations task if we treat each rating as a different type of interaction.
- 3) P3LTR utilizes additional information regarding the users and the items. In our collaborative filtering dataset, we used only node degrees as such features, but we can easily extend the model to include additional user and item features.
- 4) P3LTR is an explainable model from two perspectives: we can explain because of which items a given item is recommended and explain why some items are more influential on recommendations.
- 5) P3LTR directly utilizes the information of the node's neighbors. Such an approach might give better results than embedding-based approaches for users with a low number of interactions.
- 6) The training pipeline is used only for optimizing the weights of feature encoders. Hence it can be trained sporadically (or even just once) and be utilized for providing predictions every day.
- 7) The model prediction is almost as efficient, as RP3Beta. The difference is in the preprocessing stage, where in P3LTR, we need to additionally calculate features and pass them through feature encoders.

IV. EXPERIMENTAL SETUP

A. Dataset

We utilized the OLX Jobs Interactions dataset which is publicly available on Kaggle¹. In our previous work, as yet unpublished, we compared several collaborative filtering non-neural approaches. The RP3Beta model outperformed other approaches in terms of accuracy and efficiency and, after online A/B tests, has been deployed at OLX.

The dataset contains 65 502 201 events made on <http://olx.pl/praca> by 3 295 942 users who interacted with 185 395 job ads in 2 weeks of 2020. Each event contains 4 pieces of information: user id, item id, event type (e.g., click or reply) and timestamp.

It is important to note that users usually do not interact with many job ads (average: 20, median: 6, first quartile: 2, third quartile: 18).

¹<https://www.kaggle.com/datasets/olxdatascience/olx-jobs-interactions>

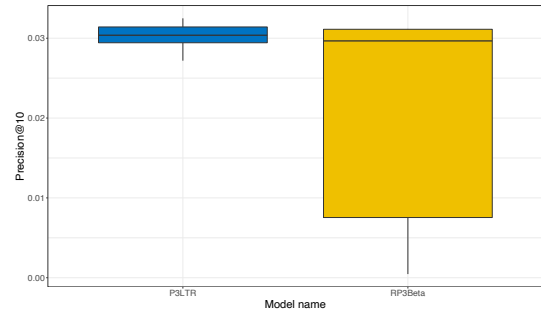


Fig. 2. Precision for each model depending on parameters.

B. Train-test splitting

We split the events into train and test sets by time, i.e., 20% of the newest events (approximately 2.8 days) were included in the test set. We filtered out from the test set all user-item pairs which appeared in the train set (to avoid recommending already seen items).

C. Hyperparameter tuning

For the sake of efficiency, we extracted 20% of users and 20% of items from the original train set and, according to the train-test splitting technique described in the previous section, we divided them into train and test sets used for validation. For each model, we defined the hyperparameter space and performed 100 iterations chosen by Bayesian optimization using Gaussian processes. We were optimizing for precision@10 [36] calculated on 30 thousand users. In Fig. 2 we can observe that the hyperparameters significantly affect the performance of tuned models. Therefore, tuning was essential for providing reliable results for compared methods. We can also see that choosing suboptimal hyperparameters for the RP3Beta model can result in very poor performance, which is not the case for P3LTR. We report the optimal hyperparameters in Table I.

V. RESULTS

We used the best found hyperparameters to train our model on the full dataset and generate recommendations for all 619 389 users in the test set. We compared the following methods:

- **P3LTR**,
- **RP3Beta**,
- **P3**: which is the RP3Beta model for $\alpha = 1$ and $\beta = 0$,
- **#3-Paths**: which is the RP3Beta model for $\alpha = 0$ and $\beta = 0$. We initialized all the parameters of our P3LTR model to zeros, which makes the #3-Paths model equivalent to the P3LTR model before the learning process.

In this section, we compare the accuracy and diversity of these models. We will also discuss the parameters of our P3LTR model.

A. Accuracy

In Table II we list common accuracy evaluation metrics calculated with respect to the top 10 recommendations. The

TABLE I
MODEL HYPERPARAMETERS.

Model	Model hyperparameters
RP3beta	{'alpha': 0.61447198, 'beta': 0.1443548}
P3LTR	{'regularization': 0.001, 'learning_rate': 0.02, 'batch_size': 153, 'iterations': 80, 'top_k': 205, 'loss': 'log_ratio'}

TABLE II
ACCURACY RESULTS. ALL PRESENTED METRICS WERE DESCRIBED IN [36].

Model	P3LTR	RP3Beta	P3	#3-Paths
precision	0.0515	0.0484	0.0481	0.0391
recall	0.0817	0.0783	0.0782	0.0611
ndcg	0.0798	0.0759	0.0755	0.0599
mAP	0.0414	0.0393	0.0390	0.0302
MRR	0.1423	0.1365	0.1363	0.1107
LAUC	0.5408	0.5391	0.5391	0.5305
HR	0.3242	0.3131	0.3147	0.2605

values should not be directly compared with the results achieved on other datasets, because metrics heavily depend on the distribution of the dataset (for example high sparsity) and the train/test splitting strategy. We can observe that our method P3LTR outperforms RP3Beta with respect to all listed metrics.

To identify differences between the methods, we test the null hypothesis that all methods perform the same. We used the Friedman test with Iman and Davenport extension. The p -value from this test is equal to 0 which indicates that we can safely reject the null hypothesis that all the algorithms perform the same. We can therefore proceed with the post-hoc tests in order to detect significant differences among all of the methods. Demšar [37] proposes the use of the Nemenyi's test and preparing a plot to visually check the differences, the critical difference plot. In the plot, those algorithms that are not joined by a line can be regarded as different. In our case, with a significance of $\alpha = 0.05$ any two algorithms with a difference in the mean rank above 0.006 are regarded as non-equal (Fig. 3).

We can observe three disjoint groups of methods:

- 1) P3LTR,
- 2) RP3Beta and P3,
- 3) #3-Paths.

From this analysis, we see that P3LTR performs significantly better than other methods on the examined dataset.

B. Diversity

Most of the job ads refer to only one job position. Hence we should avoid recommending the same item to a great number of users. For that reason, we report also the diversity metrics in Table III. Test coverage is a fraction of items from the test set which were recommended to at least one user. We also report Shannon entropy [38] and Gini index [38]. We can see that P3LTR is the most diverse method with respect to all these metrics. We can also note that the #3-Paths method seems less

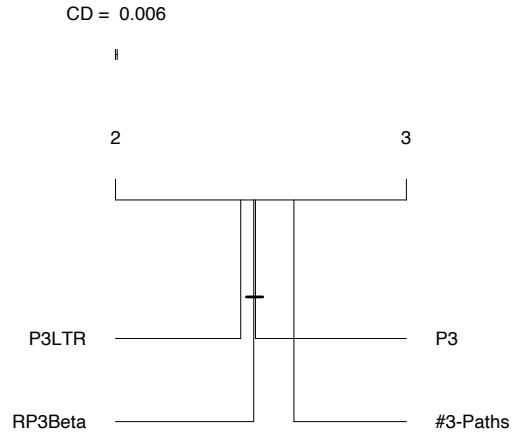


Fig. 3. Critical difference plot (for precision).

TABLE III
DIVERSITY RESULTS.

Model	P3LTR	RP3Beta	P3	#3-Paths
test coverage	0.757	0.573	0.617	0.374
Shannon entropy	10.090	9.527	9.631	8.712
Gini index	0.844	0.908	0.898	0.957

diverse than other methods. We suppose that the reason is that this method recommends the items based on the number of paths of length 3 connecting a given user and item, so the most popular items are more often recommended.

In order to decide whether to deploy a new recommendation system in production, we usually check how different are the recommendations produced by a new model compared to the old one. To assess it we calculated the overlap coefficient [39] with respect to user-item pairs. The results are reported in Table IV. We see that 70% of the top 10 recommendations provided by P3LTR and RP3Beta models are the same recommendations. We can also observe that RP3Beta and P3 provide pretty similar results on our dataset (overlap coefficient equals 84%).

C. Parameters of the P3LTR model

As we mentioned, the parameters of our model can be easily interpreted. In the Table V we report the values of $d^{(k)}$ (parameters related to node degrees) and $r^{(k)}$ (parameters related to recency).

In previous works regarding the RP3Beta model, usually positive values for α and β were chosen to discourage the model from recommending the most popular items [1], [3].

TABLE IV
AN OVERLAP OF MODELS.

Model	P3LTR	RP3Beta	P3	#3-Paths
P3LTR	100%	70%	64%	50%
RP3Beta	70%	100%	84%	68%
P3	64%	84%	100%	60%
#3-Paths	50%	68%	60%	100%

TABLE V
PARAMETERS OF THE P3LTR MODEL.

Parameter	$k = 1$	$k = 2$	$k = 3$
$d^{(k)}$	0.631	0.163	0.433
$r^{(k)}$	0.081	0.008	0.028

We can see that in our machine learning approach also positive values were learned for all $d^{(k)}$ parameters.

Additionally, in RP3Beta model we have $d^{(1)} = d^{(2)} = \alpha$. However, P3LTR model chose very different values for $d^{(1)}$ and $d^{(2)}$.

Regarding recency, the model chose positive values of $r^{(k)}$ parameters. It means that the more recent interactions should have higher importance.

We do not discuss parameters related to event type e.g., viewing or replying to an ad, because they did not converge within the number of iterations we have chosen. Hence the reported results might differ when we train the model multiple times. We believe the convergence could be achieved with a greater value of a batch_size hyperparameter, but it would also significantly increase the training time.

VI. SUMMARY

In the paper, we introduced a new graph vertex ranking recommendation method which we named P3LTR. It generalizes the RP3Beta model which provides very efficient and accurate recommendations on multiple datasets. We described several strengths of our approach, including explainability and prediction efficiency. We showed that our method is superior to RP3Beta on the OLX Jobs Interactions dataset in terms of accuracy and diversity of recommendations.

The proposed method may improve the quality of recommendations currently being generated using the RP3Beta model that is implemented at OLX Jobs in a production setting.

In future work, we plan to explore more advanced feature encoders which utilize user and item features. We would like to explore and compare different loss functions for the P3LTR model. Additionally, we would like to launch A/B tests on production to measure the model's effectiveness on real users.

REFERENCES

- [1] B. Paudel, F. Christoffel, C. Newell, and A. Bernstein, "Updatable, accurate, diverse, and scalable recommendations for interactive applications," *ACM Transactions on Interactive Intelligent Systems*, vol. 7, pp. 1–34, 12 2016.
- [2] M. F. Dacrema, P. Cremonesi, and D. Jannach, "Are we really making much progress? a worrying analysis of recent neural recommendation approaches," in *Proceedings of the 13th ACM Conference on Recommender Systems*, ser. RecSys '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 101–109.
- [3] M. F. Dacrema, S. Boglio, P. Cremonesi, and D. Jannach, "A troubling analysis of reproducibility and progress in recommender systems research," *ACM Transactions on Information Systems*, vol. 39, pp. 1–49, 01 2021.
- [4] V. W. Anelli, A. Bellogín, T. Di Noia, and C. Pomo, "Reenvisioning the comparison between neural collaborative filtering and matrix factorization," in *Fifteenth ACM Conference on Recommender Systems*, ser. RecSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 521–529.
- [5] C. Gómez-Urbe and N. Hunt, "The netflix recommender system," *ACM Transactions on Management Information Systems*, vol. 6, pp. 1–19, 12 2015.
- [6] B. Smith and G. Linden, "Two decades of recommender systems at amazon.com," *IEEE Internet Computing*, vol. 21, pp. 12–18, 05 2017.
- [7] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*. Springer, 2007.
- [8] U. Javed, K. Shaukat Dar, I. Hameed, F. Iqbal, T. Mahboob Alam, and S. Luo, "A review of content-based and context-based recommendation systems," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 16, 02 2021.
- [9] R. Chen, K. Hua, Y.-S. Chang, B. Wang, 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. PP, pp. 1–1, 10 2018.
- [10] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 165–174. [Online]. Available: <https://doi.org/10.1145/3331184.3331267>
- [11] X. Ning and G. Karypis, "Slim: Sparse linear methods for top-n recommender systems," in *Proceedings of the 2011 IEEE 11th International Conference on Data Mining*, ser. ICDM '11. USA: IEEE Computer Society, 2011, p. 497–506. [Online]. Available: <https://doi.org/10.1109/ICDM.2011.134>
- [12] H. Khojamli and J. Razmara, "Survey of similarity functions on neighborhood-based collaborative filtering," *Expert Systems with Applications*, vol. 185, p. 115482, 2021.
- [13] M. Kula, "Metadata embeddings for user and item cold-start recommendations," *arXiv preprint arXiv:1507.08439*, 2015.
- [14] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ser. ICDM '08. USA: IEEE Computer Society, 2008, p. 263–272.
- [15] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, *LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation*. New York, NY, USA: Association for Computing Machinery, 2020, p. 639–648. [Online]. Available: <https://doi.org/10.1145/3397271.3401063>
- [16] 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 *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1809–1818.
- [17] O. Barkan and N. Koenigstein, "Item2vec: Neural item embedding for collaborative filtering," 09 2016, pp. 1–6.
- [18] G. Adomavicius and A. Tuzhilin, *Context-Aware Recommender Systems*. Boston, MA: Springer US, 2015, pp. 191–226. [Online]. Available: https://doi.org/10.1007/978-1-4899-7637-6_6
- [19] S. Kulkarni and S. F. Rodd, "Context aware recommendation systems: A review of the state of the art techniques," *Computer Science Review*, vol. 37, p. 100255, 2020. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1574013719301406>
- [20] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, jul 2018. [Online]. Available: <https://doi.org/10.1145/3190616>
- [21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," vol. 2016, 07 2016, pp. 855–864.

- [22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 03 2014.
- [23] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'13. Red Hook, NY, USA: Curran Associates Inc., 2013, p. 3111–3119.
- [24] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," ser. SIGIR'19. New York, NY, USA: Association for Computing Machinery, 2019, p. 165–174. [Online]. Available: <https://doi.org/10.1145/3331184.3331267>
- [25] R. van den Berg, T. Kipf, and M. Welling, "Graph convolutional matrix completion," *ArXiv*, vol. abs/1706.02263, 2017.
- [26] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 974–983. [Online]. Available: <https://doi.org/10.1145/3219819.3219890>
- [27] C. Cooper, S. H. Lee, T. Radzik, and Y. Siantos, "Random walks in recommender systems: Exact computation and simulations," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14 Companion. New York, NY, USA: Association for Computing Machinery, 2014, p. 811–816.
- [28] F. Fouss, A. Pirotte, and M. Saerens, "A novel way of computing similarities between nodes of a graph, with application to collaborative recommendation," in *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, 2005, pp. 550–556.
- [29] F. Fouss, A. Pirotte, J.-m. Renders, and M. Saerens, "Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 3, pp. 355–369, 2007.
- [30] M. Gori and A. Pucci, "Itemrank: A random-walk based scoring algorithm for recommender engines," 01 2007, pp. 2766–2771.
- [31] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4274–4282.
- [32] C. Wang and D. Blei, "Collaborative topic modeling for recommending scientific articles," 08 2011, pp. 448–456.
- [33] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, dec 2015. [Online]. Available: <https://doi.org/10.1145/2827872>
- [34] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI 2009*, 05 2012.
- [35] J. Weston, S. Bengio, and N. Usunier, "Wsabie: Scaling up to large vocabulary image annotation," 01 2011, pp. 2764–2770.
- [36] Y.-M. Tamm, R. Damdinov, and A. Vasilev, "Quality metrics in recommender systems: Do we calculate metrics consistently?" in *Fifteenth ACM Conference on Recommender Systems*, ser. RecSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 708–713. [Online]. Available: <https://doi.org/10.1145/3460231.3478848>
- [37] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.
- [38] G. Shani and A. Gunawardana, *Evaluating Recommendation Systems*. Boston, MA: Springer US, 2011, pp. 257–297. [Online]. Available: https://doi.org/10.1007/978-0-387-85820-3_8
- [39] M. Vijaymeena and K. Kavitha, "A survey on similarity measures in text mining," *Machine Learning and Applications: An International Journal*, vol. 3, pp. 19–28, 03 2016.