Two-Stage Session-based Recommendations with Candidate Rank Embeddings

José Antonio Sánchez Rodríguez
Zalando SE
Berlin, Berlin
jose.antonio.sanchez.rodriguez@zalando.de

Jui-Chieh Wu
Zalando SE
Berlin, Berlin
jui-chieh.wu@zalando.de

Mustafa Khandwawala
Zalando SE
Berlin, Berlin
mustafa.khandwawala@zalando.de

ABSTRACT

Recent advances in Session-based recommender systems have gained attention due to their potential of providing real-time personalized recommendations with high recall, especially when compared to traditional methods like matrix factorization and item-based collaborative filtering. Nowadays, two of the most recent methods are Short-Term Attention/Memory Priority Model for Session-based Recommendation (STAMP) and Neural Attentive Session-based Recommendation (NARM). However, when these two methods were applied in the similar-item recommendation dataset of Zalando (Fashion-Similar), they did not work out-of-the-box compared to a simple Collaborative-Filtering approach.

Aiming for improving the similar-item recommendation, we propose to concentrate efforts on enhancing the rank of the few most relevant items from the original recommendations, by employing the information of the session of the user encoded by an attention network. The efficacy of this strategy was confirmed when using a novel **Candidate Rank Embedding** that encodes the global ranking information of each candidate in the re-ranking process. Experimental results in Fashion-Similar show significant improvements over the baseline on Recall and MRR at 20, as well as improvements in Click Through Rate based on an online test. Additionally, it is important to point out from the evaluation that was performed the potential of this method on the next click prediction problem because when applied to STAMP and NARM, it improves the Recall and MRR at 20 on two publicly available real-world datasets.

CCS CONCEPTS

• **Retrieval tasks and goals → Recommender systems**; • **Computing methodologies → Neural networks**;

KEYWORDS

Neural networks, Session-based recommendation, Two-layer approach, Candidate rank embedding

ACM Reference Format:
José Antonio Sánchez Rodríguez, Jui-Chieh Wu, and Mustafa Khandwawala. 2019. Two-Stage Session-based Recommendations with Candidate Rank Embeddings . In *Proceedings of Workshop on Recommender Systems in Fashion, 13th ACM Conference on Recommender Systems (recsysXfashion'19).* ACM, New York, NY, USA, 6 pages.

1 INTRODUCTION

Recommendation systems have become a major tool for users to explore websites with an extensive assortment. These websites offer different recommendation products in different locations that try to meet the specific needs of the user.

For Zalando, one of the most important products is the similar-item recommendation. It is shown to all the customers that reach the product page, which offers them different alternatives related to the product that the user is browsing. In Figure 1, this page and the fashion-similar item recommendation is illustrated. In order to improve this recommendation product, a dataset that contains users' history view events and their clicks on the recommended items was collected in early 2018 for further research.

Recent studies on Session-based Recommendations have shown significant improvements compared to collaborative filtering approaches in several datasets [6, 15, 18]. Following the state-of-the-art approaches, two of the best-performing Session-based recommenders, STAMP [11] and NARM [10] were selected to predict the next clicked item in the similar item recommendation and compare the results with a basic collaborative filtering algorithm. However, we were unable to obtain any improvements from the baseline in terms of recall@20.

This counter-intuitive phenomenon was also observed and reported by other studies in the literature [3, 12]. The reasons behind the scene can be multi-faceted and complicated. For example, one possible cause could be the bias from the feedback loop existing in the production system. Alternatively, it could appear because of the intrinsic user behavior presented in the dataset. Figuring out the exact reasons and applying counter-measures to fight against this phenomenon belongs to another research area and is therefore not the focus of this paper.

On the other hand, the objective of this work is to improve the similar item recommendation in an online test. Prior an online test, the offline evaluation metrics must be improved in order to reduce the risk of impacting users negatively and to prioritize different online experiments. Our hypothesis is that with the proper use of personalized information, the resulting model should be able to outperform the baseline algorithms in both offline and online evaluations.

In order to effectively use the personalized information of user sessions, we have to overcome the performance issue of directly applying session-based models to the Zalando Fashion-Similar dataset.

Earlier results indicate that with a single session-based model it seems to struggle in capturing global information between items in the dataset. To address this issue, we propose to use Candidate Rank Embeddings (CRE) together with a session-based recommender to collect such information from the pre-trained/pre-calculated model.

Following the approach of Covignton et. al. [2], in the first stage a **Candidate Generator** was employed to take a given user's click history X_u as input to generate a sorted list of candidates C of size k. Specifically, Collaborative filtering was used as the Candidate Generator. In the second stage, C and X_u are fed to the **Re-ranker** to produce fine-tuned recommendations. Borrowing the user session encoder of STAMP E_{STAMP}, X_u is encoded and used in the Re-ranker together with an innovative Candidate Rank Embeddings (CRE). CREs learns the personalized candidate rank preference along with their item preference in the process of training the model to optimize the prediction of the next item of interest. By using CREs, the Re-ranker was enabled to incorporate implicit information from the Candidate Generator as prior knowledge that helps to guide and calibrate the training of the Re-ranker with the objective of improving the order of the original recommendation.

The offline experiments showed that our CRE-enhanced model can indeed outperform the collaborative filtering baseline. The good performance on the Fashion-Similar dataset suggests the CRE trick may also be applicable to other recommendation tasks such as general next click prediction. This hypothesis was confirmed by combining the CRE-enhanced model with two baselines, STAMP and NARM, and evaluate on the YooChoose 1/4 and Diginetica next click prediction datasets.

The main contribution of this study lies in the following two aspects:

- Candidate Rank Embeddings were used together with a session-based recommender to enhance the performance of I2I-CF in a two-stage approach. The model outperforms the baselines on a Fashion-Similar dataset in terms of Recall and MRR at 20. Also, the improvement was confirmed with an online test where it was observed significant improvements in Click Through Rate.
- Experiments and analysis were done to compare the baselines I2I-CF, STAMP and NARM, and the proposed method using those baselines as Candidate Generators on the task of predicting the next click. The results show that the model with CREs improves both Recall@20 and MRR@20 of these baselines on two publicly available datasets.

2 RELATED WORK

2.1 Session-based Recommender Systems

The concept of using user interaction history as a sequence has been introduced to recommender systems with the success of the Recurrent Neural Network (RNN) family in the sequence modeling field. Before the thrive of these models, the user interactions were mainly used in simpler methods, such as, item-item collaborative filtering [1] or matrix factorization [8, 9]. One of the early attempts of using the click sequences of users as input data and considering recommendation as a next target prediction was proposed in [4]. Authors of DREAM [17] used pooling to summarize the current basket and later, another vanilla RNN was used to recommend the next basket. With the advantage of Gated Recurrent Unit (GRU) over vanilla RNN been proven for longer sequences, GRU-based recommender systems have been proposed with different loss functions, such as Bayesian Personalized Ranking (BPR) [13], TOP-1 [18] and their enhancements [6]. These models have shown better results than matrix factorization based methods over public sequence prediction datasets.

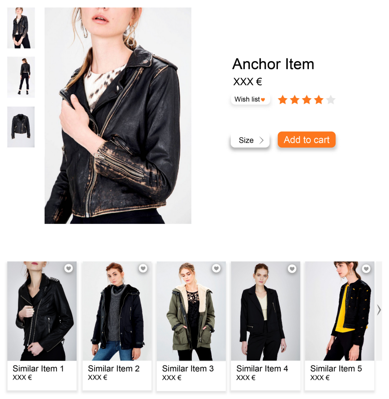

Figure 1: The product page offers similar items to the anchor.

Various authors have proposed different improvements to the GRU family approach. In [15] two techniques were proposed to address the data sparsity issue and combat the behavior distribution shift problem. In addition, the NARM architecture [10] further increased the prediction power of GRU-based session recommenders by adding an attention layer on top of the output. Despite of the high recall and Mean Reciprocal Rank (MRR), the training process of NARM takes longer time than pure GRU approaches.

Considering the time consumption problem of the RNN families, authors of Short-Term Attention model [11] managed to substantially reduce the training and prediction time by replacing the RNN structure with simpler components such as the feature average layer and feed-forward networks. As an improvement, the STAMP model, in addition, encodes the user feature from their click session with an attention operation. However, despite the promising results on several public datasets, there were studies reporting that some state-of-the-art session-based models do not outrun simple collaborative filtering methods on a certain datasets [3, 12]. We found the same phenomenon in our dataset generated by an online system, and that motivates us to design a solution to have it benefit from personalization without performance degradations.

2.2 Two-Stage Approaches

Two-stage approaches are widely adopted in different recommendation tasks in various domains. Paul Covington et. al. proposed the use of two neural networks to achieve video recommendation in YouTube [2]. With a clear separation of candidate selection and ranking generation, their solution targets especially at multimedia recommendations. This cascading approach was used to solve performance issues, mainly due to the enormous amount of available videos on Youtube.

Other studies have employed this technique to improve accuracy. Rubtsov et. al. [14] applied a two-stage architecture to improve the quality of music playlist continuation. Similar to us, they used collaborative-filtering to generate candidates and a more complicated algorithm based on gradient boosting for the final prediction. Likewise, we found that the use of the two-stage architecture makes it easy to improve the model performance when applying session-based recommendations to the fashion similar item recommendation. Besides applying a two stage approach to use the session information, we modeled the user-rank preference with Candidate Rank Embeddings.

3 PROBLEM STATEMENT

The Session-based Recommendation problem is usually modeled as predicting the next item of interest for the user, providing their interaction history. Given an assortment of items I where all possible items come from, the short-term history X_u of a user u consists of a sequence of l items $x_0, x_1, \cdots, x_{l-1} \in I$ that u has interacted with so far. Session-based Recommenders aim to shift the next item $x_l = y_{u^+}$ that the user will interact with to the top position of the recommendation list when given X_u. A collection of user interaction sequences is denoted as a dataset $\mathcal{D}$, it is composed of N pairs of user sequence and target item $\{(X_u, y_{u^+})$, for $u = 1, 2, \cdots, N\}$.

A session-based recommender produces a sorted list L_Y from all items in a subset $Y \subseteq I$. In most cases, Y is equivalent to I, but it can also come from a selected set of candidates from another recommender. To obtain L_Y, a score s_y for each item $y \in Y$ is calculated and all the items from Y are ranked by their scores in a descending order. The scores of items in Y is denoted as S_Y and the function *rank* yields the item list L_Y with the new order according to S_Y:

$$L_Y = rank(S_Y) \tag{1}$$

In the following sections, we use $V_x \in \mathbb{R}^d$ to represent the latent vector for an item x. To represent a matrix of feature vectors of items, the notation V_Q is used; where Q is a list or a set of items. The shape of V_Q is $\mathbb{R}^{d \times |Q|}$.

4 TWO-STAGE RECOMMEDER WITH CANDIDATE RANK EMBEDDINGS

Using the naming convention from [2], the first recommender of the cascade is considered as the **Candidate Generator G**, while the second, known as the **Re-ranker R**, ranks the most relevant items from the output of **G**. Both **G** and **R** take a candidate set and a specific user history as input parameters, and later assign a score to each candidate. The final recommendation of the proposed method L_Y is calculated as follows:

$$\begin{aligned}
L_{Y_G} &= rank(S_{Y_G}) = rank(G(Y,\ X_u,\ \theta)) \\
C,\ \overline{C} &= L_{Y_G}[:k],\ L_{Y_G}[k:] \\
L_{C_R} &= rank(S_{C_R}) = rank(R(C,\ X_u,\ w)) \\
L_Y &= L_{C_R} : \overline{C}
\end{aligned} \tag{2}$$

Where C denotes the k most relevant candidates computed by G and $\overline{C}$ indicates the rest in Y. θ and w denotes the trainable parameters of G and R respectively. In this study, we set θ and w to be independent and do not share parameters. The training process of the proposed model is also two-staged. G is trained first (in case of training needed). The training of R starts after a well-trained G is obtained, and only considers the top-ranked k candidates coming from G, as described in Eq 2. The parameters θ and w are optimized using $\mathcal{D}$. Figure 2 illustrates the overall architecture of the model.

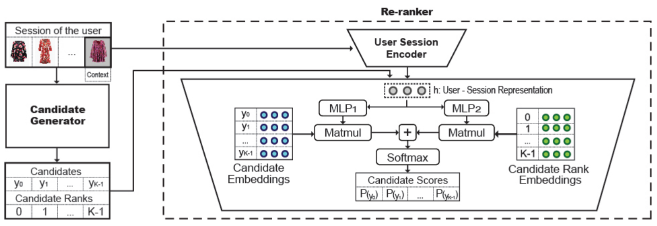

Figure 2: The model architecture: The Candidate Generator is trained first and treated as a black box. Then, the Re-ranker is trained to re-score the candidates provided by the Generator. For calculating these scores, two components are considered: the candidates and the rank preference of them.

At inference time, given a specific user interaction history X_u, both, G and R, take it as an input and operate sequentially. More details about the Candidate Generator and the Re-ranker are described in Sections 4.1 and 4.2

4.1 The Candidate Generator

The Candidate Generator G can be an arbitrary recommender that takes a user session X_u as input and ranks the set Y_G. For training G we consider $Y_G = I$.

The selection of the algorithm for G depends mostly on the characteristics of the dataset $\mathcal{D}$ and the performance of the algorithm on generating high quality candidates.

4.2 The Re-ranker

The Re-ranker R takes the same user click sequence X_u, as G does, but concentrates on ranking a smaller set of candidates C determined by the Candidate Generator it connects with.

We employ a variant of STAMP in R to include the candidate rank information in the model. Specifically, the encoder for the user click history E_{STAMP} is reused. E_{STAMP} is a simple element-wise multiplication between the history representation h_s and the anchor representation item h_t described in [11].

$$h_u = E_{STAMP}(X_u) = h_s \odot h_t \tag{3}$$

The STAMP encoder E_{STAMP} was chosen because of its architectural simplicity and promising performance in predicting the user preference.

Before defining the calculation of the scores, Candidate Rank Embeddings must be introduced. Given a sequence X_u, we obtain a sorted list of candidates $C = [y_0, y_1, \cdots, y_{k-1}]$ of size k from G. The **Candidate Rank** r_i is an integer ranging from $[0, k)$ which denotes the position of the i^{th} item in C. Each Candidate Rank is associated with a Candidate Rank Embedding. Therefore, CREs are positional embeddings shared among different candidate lists produced by G. The CREs for a Re-ranker that takes k candidates into account can be represented as a matrix W_{CR} of shape $\mathbb{R}^{k \times d_{CRE}}$.

With the CRE defined the scores are specified in the following equation:

$$\begin{aligned} h_e &= MLP_1(h_u) = tanh(W_{e1}^{\intercal} tanh(W_{e2}^{\intercal} h_u + b_{e2}) + b_{e1}) \\ h_r &= MLP_2(h_u) = tanh(W_{r1}^{\intercal} tanh(W_{r2}^{\intercal} h_u + b_{r2}) + b_{r1}) \\ D_R(h_u, C) &= softmax(V_C^{\intercal} h_e + W_{CR}^{\intercal} h_r) \end{aligned} \quad (4)$$

Where $W_{e1}, b_{e1}, W_{e2}, b_{e2}, W_{r1}, b_{r1}, W_{r2}, b_{r2}$ are learnable weight matrices and bias terms of the feed-forward network layers, V_C is the item embeddings of the candidates and W_{CR} is the candidate rank embedding matrix. Note that W_{CR} is the same for all user sequences X_u because they depend only on the rank of the candidates. All the embeddings are initialized randomly and trained together with the model.

We train R to predict the next click that is in C by using the Cross Entropy loss.

The main difference between our solution and STAMP is the use of W_{CR} and two non-linear projections MLP_1 and MLP_2. These projections are used to approximate the click probability of the candidates and the rank preference of users. The first projection focuses on predicting the embedding of the target item, while the second one focuses on predicting the embedding of the position of the target item in the ranked candidate list.

The intuition behind learning the rank preference is that the information from the output of G can flow into the model, and a balance between the newly-learned item preference and the old rank can be obtained by summing up two user preferences.

Furthermore, since the ranking score comes from the dot product from the candidate rank embeddings and a projection from the user representation, this allows the model to learn the relationship between the user and the position of the target. For example, it gives the capability to recognize which type of users like to click the top positions or the items which co-occur with the anchor very often when G produces its candidates using co-occurrence information. This behavior can be difficult to learn with a model that considers only the user-item preference.

In Section 5.5 an analysis of the importance of using the candidate rank information is presented, where we compare our two-stage approach against one without CREs.

So far, we only tried using a one-to-one mapping between Candidate Ranks/positions and CREs. In applications with a large candidate set, having multiple ranks share one CRE could be beneficial because training signals can be shared among several unpopular positions.

5 EXPERIMENTS AND ANALYSIS

5.1 Baselines

The following recommendation algorithms are used in our experiments as the baselines.

- **Item-Item Collaborative Filtering (I2I-CF)**: it considers only the most similar items for the last-seen item x_{l-1} in the user interaction/click sequence. It is pre-calculated using a variant of the cosine similarity function described in [1].
- **Attention-based GRU (NARM)** [10]
- **Short-Term Attention Model (STAMP)** [11]

5.2 Experiment Setup

Each of the baselines is used as a Candidate Generator in two sets of separated experiments. The first set of experiments evaluate the model performance on the Fashion-Similar dataset to predict the next similar item on the carousel that the user would click. The second set of experiments compare the performance of CREs on two next item prediction datasets, when combining with different baseline models. Before training the re-ranker, we first train the corresponding baseline and keep them fixed as Candidate Generators. Not all the training sequences are used in training, the re-ranker only considers those which their target item falls within the candidate set C from G. Five percent of the training examples are randomly sampled for validation. During training, the model performance is checked every 1000 steps and the best model is selected by their performance in terms of Recall@5. Adam [7] is used to train R for 5 epochs with a learning rate of **0.001** and batch size of 512. We set the number of candidates k being returned by G to be **100**. The item embeddings and model weights of E_{STAMP} are initialized w.r.t. the best settings reported in [11]. The weights of W_{CR}, W_{e1}, W_{e2}, W_{r1}, W_{r2}, are initialized with Xavier [5]. To label the results, we use RRCRE-X as an abbreviation of re-ranking the output from an approach X with CREs.

5.3 Predicting the Fashion Similar Item

In this task, our goal is to predict the clicked products in the similar-item-recommendation carousel, given the latest l actions of a user.

Only the **Fashion-Similar** dataset was used because the other datasets are mainly used for general next click prediction tasks. The dataset was collected using the customers actions from several major european markets for multiple days. Every record in this dataset consists of a click on the similar-item recommendation, which is the target, along with the latest 12 items the user browsed before interacting with the similar-item recommendation.

The dataset was split into training and test sets by time, leaving 9 days for the training set and the last day for the test test. With this arrangement, the offline evaluation measures how well the model performs on the next day assuming we retrain our algorithms daily.

The training set contains 8353562 examples, for 1435605 users which interacted with 650228 items in total. The test set contains 624559 examples, for 233051 users which interacted with 259784 items in total.

5.4 Predicting the next click

For this task, the objective is to predict the next user interaction given the past click history. We used the following datasets:

- **YooChoose 1/4**: the dataset was preprocessed exactly as described in [15].
- **Diginetica**: we preprocessed the dataset with the approach described in [16]. Note that Wu et. al. introduced additional preprocessing to the dataset compared to [10].

Table 1: Performance of the baselines and the proposed method on predicting the similar item.

	Fashion-Similar	
	Recall@20	MRR@20
I2I-CF	**0.8106**	**0.2611**
STAMP	0.7244	0.2443
NARM	0.6989	0.2510
RRCRE-CF	**0.8381**	**0.2981**

Table 2: Performance of the baselines and the proposed method on the task of predicting the next click.

	YooChoose 1/4		Diginetica	
	Recall@20	MRR@20	Recall@20	MRR@20
I2I-CF	0.5259	0.2001	0.3760	0.1211
STAMP	**0.6983**	0.2915	0.4834	0.1588
NARM	0.6973	**0.2921**	**0.5015**	**0.1599**
RRCRE-I2I-CF	0.5586	0.2446	0.3773	0.1220
RRCRE-STAMP	**0.7086**	**0.3133**	0.5046	**0.1677**
RRCRE-NARM	0.7029	0.3082	**0.5116**	0.1675

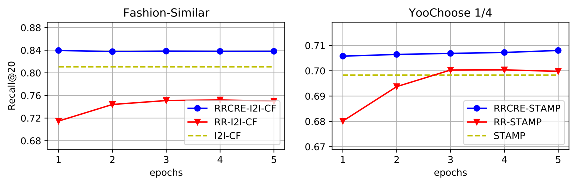

Figure 3: Our approach with and without Candidate Rank information.

5.5 Offline Results and Analysis

From Table 1 we can observe that STAMP and NARM don't show superior results in the task of predicting the similar item in the offline evaluation. The result on Fashion-Similar seems to be counter-intuitive, and further investigations are required as future work.

However, with RRCRE-I2I-CF we are able to improve I2I-CF in the similar item prediction task in terms of Recall@20 and MRR@20. It is because the model is capable of utilizing the hidden information in the ranking of the baseline together with the session information captured by the attention network from E_{STAMP}. An online test was performed to confirm the offline results, described in Section 5.6.

As shown in Table 2, STAMP and NARM perform significantly better than I2I-CF in the next click prediction task.

We also applied our method to use STAMP and NARM as Candidate Generator in the next click prediction task, the evaluation result shows that it is able to slightly improve the Recall@20 and MRR@20 of STAMP and NARM on both YooChoose 1/4 and Diginetica.

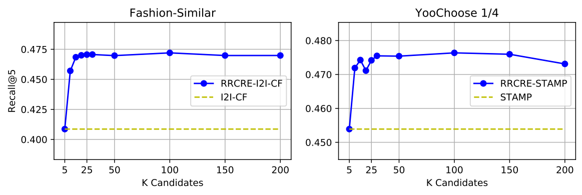

Figure 4: Recall@5 of the bests approaches on each dataset given different number of candidates to re-rank.

To understand the improvement obtained by applying the Candidate Rank Embeddings, we compared the model performance between the proposed method with and without CRE, i.e. the final scores of candidates become $D_R(h_u, C) = softmax(V_C^{\top} h_e)$. We use RR-X as an abbreviation of re-ranking the output from an approach X without CREs. The result is illustrated in Figure 3. For the Fashion-similar dataset, we can observe that simply re-ranking the most relevant candidates from I2I-CF with E_{STAMP} doesn't lead to superior results. On the other hand, when training with CREs we obtain a better result listed in Table 1. We also compare RR-STAMP and RRCRE-STAMP in one of the next click prediction datasets. It turns out that RRCRE-STAMP outperforms the baseline from epoch 1, while RR-STAMP requires more iterations. It is because RR-STAMP has to learn the next clicked target from randomly-initialized model parameters without the rank information from STAMP being presented.

Additionally, we illustrate the behaviour of the proposed method with respect to the number of candidates to re-rank. In Fig 4 we can observe improvements in recall@20 even with a small k. In addition, we found the recall plateaus or decreases when k exceeds a certain threshold. One possible reason could be that the candidates associated with the low ranks rarely appear as targets in the dataset. As a result, the CREs for these ranks could not have been well-trained and could have captured misleading information. However, since using a relatively small k simplify the effort of training and serving in production, more investigations were not done.

5.6 Online test in Zalando

From the previous section, it was observed an improvement in the offline metrics with respect to I2I-CF. As a consequence, an online test hosted by Zalando was performed to compare I2I-CF and RRCRE-I2I-CF.

RRCRE-I2I-CF was served using cpu machines running Tensorflow-Serving. For this algorithm, the recommendations are calculated in real-time. On the contrary, I2I-CF was served by using a static table stored in memory.

I2I-CF was giving static non-personalized similar recommendations and RRCRE-I2I-CF was adapting the similar item list depending on the user's previous l actions.

The models were updated every day to be adapted to the latest user behaviour and we compared their performance in major european markets for several days.

To ensure that the recommendations satisfy the similarity constraint of the product, some filters based on category trees were applied to the output of both methods.

The results showed relative improvements in engagement based on a significant **+2.84%** ($p\text{−}value \leq 0.05$) increase in Click Through Rate. It proves that there is a positive effect of using the session of users to generate a personalize ranking of similar items and supports the offline experiment results carried-out.

6 CONCLUSION AND FUTURE WORK

In this study, the possibility of improving the fashion similar item recommendation was explored with a two-staged re-ranking approach that is able to benefit from the Candidate Rank information, the session of the user and a small set of candidates.

With this approach, the Recall@20 and MRR@20 of I2I-CF was improved on the Fashion-Similar dataset, and the success in the offline evaluation was confirmed by an online test.

The proposed approach was also confirmed to be useful to improve the performance of two advanced session-based recommendation algorithms, STAMP and NARM on the next click prediction datasets YooChoose 1/4 and Diginetica. Despite the success in the offline evaluation, further experiments are needed to confirm the impact of the proposed method in the context of session-based recommendation.

ACKNOWLEDGMENTS

The authors are immensely grateful to Alan Akbik, Andrea Briceno, Humberto Corona, Antonino Freno, Francis Gonzalez, Romain Guigoures, Sebastian Heinz, Bowen Li, Max Moeller, Roberto Roverso, Rezar Shirvany, Julie Sanchez, Hao Su, Lina Weichbrodt and Nana Yamazaki for their support, revisions, suggestions, ideas and comments that greatly helped to improve the quality of this work.

REFERENCES

[1] Fabio Aiolli. 2013. A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge. In *IIR*.

[2] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[3] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. 2018. News Session-Based Recommendations Using Deep Neural Networks. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems (DLRS 2018)*. ACM, New York, NY, USA, 15–23. https://doi.org/10.1145/3270323.3270328

[4] Robin Devooght and Hugues Bersini. 2016. Collaborative Filtering with Recurrent Neural Networks. *CoRR* abs/1608.07400 (2016). arXiv:1608.07400 http://arxiv.org/abs/1608.07400

[5] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Yee Whye Teh and Mike Titterington (Eds.), Vol. 9. PMLR, Chia Laguna Resort, Sardinia, Italy, 249–256. http://proceedings.mlr.press/v9/glorot10a.html

[6] Balá zs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. ACM, New York, NY, USA, 843–852. https://doi.org/10.1145/3269206.3271761

[7] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. https://doi.org/10.1109/MC.2009.263

[9] Rajeev Kumar, BK Verma, and Shyam Sunder Rastogi. 2014. Social popularity based SVD++ recommender system. *International Journal of Computer Applications* 87 (Jan. 2014).

[10] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. ACM, New York, NY, USA, 1419–1428. https://doi.org/10.1145/3132847.3132926

[11] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining (KDD '18)*. ACM, New York, NY, USA, 1831–1839. https://doi.org/10.1145/3219819.3219950

[12] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *CoRR* abs/1803.09587 (2018). arXiv:1803.09587 http://arxiv.org/abs/1803.09587

[13] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09)*. AUAI Press, Arlington, Virginia, United States, 452–461. http://dl.acm.org/citation.cfm?id=1795114.1795167

[14] Vasiliy Rubtsov, Mikhail Kamenshchikov, Ilya Valyaev, Vasiliy Leksin, and Dmitry I. Ignatov. 2018. A Hybrid Two-stage Recommender System for Automatic Playlist Continuation. In *Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18)*. ACM, New York, NY, USA, Article 16, 4 pages. https://doi.org/10.1145/3267471.3267488

[15] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 17–22. https://doi.org/10.1145/2988450.2988452

[16] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based Recommendation with Graph Neural Networks. In *Proceedings of The Twenty-Third AAAI Conference on Artificial Intelligence (AAAI '19)*. http://arxiv.org/abs/1811.00855

[17] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 729–732. https://doi.org/10.1145/2911451.2914683

[18] Bal á zs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015). arXiv:1511.06939 http://arxiv.org/abs/1511.06939