Artificial Intelligence Research and Development
A. Cortés et al. (Eds.)

doi:10.3233/FAIA220325

Contextual TV Show Recommendation

Paula Gómez DURAN [1], Jordi VITRIÀ
Departament de Matemàtiques i Informàtica, Universitat de Barcelona

Abstract.
Recommender systems are a form of artificial intelligence that is used to suggest items to users of digital platforms. They use large data sets to infer models of users' behavior and preferences in order to recommend items that the user may be interested in. Following the trend imposed by digital media companies and willing to adapt to the media consumption habits of their customers, TV broadcasters are starting to realize the potential of recommender systems to personalize the access to their online catalog. By understanding what viewers are watching and what they might like, TV broadcasters can improve the quality of their programming, increase viewership, and attract new viewers.

In this work, we analyze one specific group of users that TV broadcasters must take into account when creating a recommender system: non-logged users. In this scenario the challenge is to use contextual information about the interaction in order to predict recommendations, as it is not feasible to use any kind of information about the user. We propose a method to leverage data from other type of users (logged users and identified devices) by using Graph Convolutional Networks in order to come up with a more accurate recommender system for unidentified users.

Keywords. Recommender Systems, Graph Convolutional Network, Context-aware recommendations, Public Media Service

1. Introduction

Recommender systems are a family of artificial intelligence tools that are used to suggest items for users of digital platforms, such as online media (OM) platforms and public-service media (PSM) platforms. In this scenario, they are specifically used to optimize user's engagement. Collaborative filtering has been identified as one of the best technical approaches to accomplish this task [3, 5, 19], but the case of PSM platforms faces some specific requirements, such as the presence of a special group of users that are not present in OM and which we aim to analyse in this work: non-identified users. This kind of users is very typical of PSMs, as on most of these platforms there is no requirement for a user to be registered and cookies cannot always be guaranteed to track users' sessions.

Over the last recent years, context-aware recommendations [11] have aroused in order to end up with more accurate recommendations. Context-aware recommendations [1] take into account all aspects surrounding a user's situation when making suggestions.

[1]Corresponding Author: Paula Gómez Duran, Gran Via de les Corts Catalanes 585, 08007, Barcelona, Spain; E-mail: paula.gomez@ub.edu.

This can include factors such as time, location, device, weather, previous social media interactions, and other activities taking place at the same time. However, a lot of this research have been focused on the conventional context-aware problem, in which the model includes context as an important, but not essential, feature in order to enrich the information about the user-item interaction [8, 14]. Hence, those cases where there are no recorded user interactions (continuous cold-start problem) and where context-state is set as the only input (and thus it becomes essential), are under-represented scenarios in recommender systems research.

In this work we state that, in spite of the fact that PMS cannot always guarantee very specific contexts, pure-contextual recommendation problem can be addressed in a positive way. In this scenario we propose to leverage collaborative information from other users by using Graph Convolutional Networks to generate richer predictions than not just those corresponding to the most probable items for a given context specification. Therefore, our only approach is to use a pure contextual recommender system in order to predict recommendations, as it is not feasible to use any other kind of information. By working with a dataset of real interactions from a Catalan PMS, we will show how to build a pure contextual recommender system that maximizes the hit rate and leverage all the previous collaborative data available, even not needing it to have any information about the existing user or items.

2. Related work

Pagano et al. [11] were among the first to identify the huge importance of context in recommenders. They stated that context should be taken into account *per se* instead of being used as an ”extra” feature that could only help to get better predictions. They claimed we should move from context-aware to context-driven recommendation models.

Based on the work of Adomavicius and Tuzhilin [1], we can consider that there are two different views of context: representational and interactional. The first one refers to observable attributes that are known a priori and that do not change over time, while the second one refers to the cyclical relationship between context and activity that makes the context change over time. They reckon that fully observable factors are easier to handle than unobservable ones and they present three different ways of including context into a traditional recommender system model: pre-filtering, post-filtering or modeling. In an experimental evaluation, Panniello et al. [12] found that the optimal choice of a pre-filtering or post-filtering strategy strongly depends on the particular recommendation problem. Thus, in this work we will address the context with the modeling strategy.

Context-aware recommendation has become increasingly sophisticated in the recent years [8, 14, 15]. On the one hand, lots of efforts have been made to model *sessions*, which are basically sequences of user interactions, as contextual signals. As a session has usually an aim, there are many different recommender systems that characterize sessions in order to provide better recommendations. On the other hand, using contextual factors in order to figure out the availability patterns of some items have also been widely researched. However, most of the research has been focused on session-based or context-aware models where at least some information of the user is still available (i.e. ID from logged user, session of the user, cookies of the user, ...) while not big efforts have been put in those extreme cases where user and item dynamics make the challenge be a continuous cold start problem.

Since the first models, which simply considered item/context combinations as inputs of the recommender (i.e., a TV program was transformed into 2 different items when it was consumed from TV or it was consumed from a desktop), some more sophisticated models [9, 10, 18] have aroused which open up new possibilities for exploiting new context-variables which help improve the models. However, they require some specific information such as content descriptions or location information to, at least, encode somehow the preferences of a given recommendation (like "simulating" user's ID).

Our work differs from the SOTA methods as we aim to solve the recommendation problem in a more generic way for one specific group of users that TV broadcasters must take into account when creating a recommender system: non-logged users.

3. Logged users' behavior as a context

In this section, we briefly describe the dataset from the Catalan PMS that we have analyzed in the experiments and the method we propose to leverage the information about logged users.

3.1. Dataset

We obtained an anonymized dataset corresponding to historical data of user views of the online catalogue of a Catalan public broadcaster, TV3, collected over diverse platforms (SmartTVs, website, mobile apps) throughout a whole calendar year (2021). The raw data contained information on user interactions indicating *userID* (including fake IDs for non-logged users), *itemID* (a one hot encoding with a '1' activating the position which references the item number) and some contextual information of the interaction such as device or timestamp (using multi-hot encoding to represent the active context combinations). Items were identified at the single episode level (e.g., morning and evening news had separate IDs, as had each episode of TV series as well). As it would not be helpful for the recommender to have to rank different episodes of the same TV show, we aggregated all episodes from the same program into the same *itemID*. This resulted in a large reduction in the number of items.

We identified three groups of users: those who were logged into the platform (logged users), those who we could track by the cookies (cookies users) and those who had just one interaction, either because they did just interact once or due to the impossibility of tracking them (non-logged users).

We report in table 1 the statistics from the non-logged users, which we aim to analyse, and from the logged users dataset, which we have used in order to leverage the past behaviour and the data flow structure that actually occurs in the PMS platform.

	#users	**#items**	**#interactions**
`Logged`	10,919	509	69,161
`Non-Logged`	251,561	613	15,590,592

Table 1. Dataset statistics in terms of number of users, number of items and number of interactions among them.

The first subgroup of the data is conformed by those users who have a profile on the broadcaster platform. Even though no user information is collected, the fact that they

are logged in is very useful in order to ensure continuity of the data consumption and leverage all user records.

We applied some filters in order to consider data corresponding to a minimum of three interactions per user and also a minimum frequency of three visualizations per item, with the aim of removing outliers and work with more stable data. Due to the diverse nature of the platforms where items are offered, the public TV broadcaster does not require to be logged in and so that is the reason why the dataset is smaller in terms of the number of users (reflected in the summary statistics in table 1).

After filtering, we adapted the data to build a dataset which consist of $<$*userID, itemID, rating, timestamp*$>$ tuple interactions.

The second subgroup collected is conformed by those users who had just one interaction captured in the platform. In order to determine which are those context that are relevant when trying to characterize an item consumption, we have done some user-profiling analysis by applying some effective graph-based clustering methods such as Markov clustering algorithm or more classic clustering methods such as K-Means. By applying them we could analyze which context dimensions where the ones affecting each cluster and, in fact, we concluded the dimensions that actually affect a user: the device from which the consumption is done, the period of the day (late night, early morning, morning, noon, evening, night), and whether it is weekend or not.

After analysing the most representative context, we adapted the data to build a dataset which consist of $<$*itemID, rating, deviceID, periodID, isweekend*$>$ tuple interactions, such as for example $<$*237, 1, 3, 4, 0*$>$ would mean that item 237 was consumed ($rating = 1$) in a mobile phone ($deviceID = 3$), during the early morning ($periodID = 4$) of a weekday ($isweekend = 0$).

3.2. Pure-contextual recommendation

As it has been mentioned in previous sections, the most commonly addressed problems in contextual recommendation are those stating that users' preferences could be different across different contextual situations. However, very few research has been done into finding models which suit best for those scenarios where no information from user is collected and in which, therefore, an extreme cold-start problem scenario is created. At that stage, if no information is collected from user nor any information from content can be leveraged (PMS do not always characterize all content they have), the only option left is to build a pure-contextual recommender by modifying some existing models in order to allow them dropping the *userID* dimension.

Pagano et al. [11] propose some models where *userID* is just dropped. However, it is non-trivial how to do it, as it would imply changes when evaluating the system or performing negative sampling when training it. Besides, unless the number of unique contexts you have is on the order of number of users in your recommendation data, in which case you would be able to consider each unique context combination as a particular user who consumed an item, a vanilla RS will not be able to succeed this way. In fact, for those cases where the number of unique context is on much minor order than the number of items (e.g. maybe 10, 20, 30 contexts vs 600 items), addressing the problem as a multi-class classification task might be the best option.

In this work, we address the case where, given a unique combination of contexts (which are not enough precise to be used as an identifier for a given interaction), a classi-

fier is able to predict the probability of each item (multi-class problem) to be consumed. In other words, we try to find out a probability for consuming an item under a specific context combination.

3.3. How to leverage historical data from logged users?

As the scenario we are approaching requires to use a **pure contextual recommender system** in order to predict items, we have chosen to treat it as a classification problem, where the items on the database are represented in the output of the network in terms of a probability distribution.

The ideal scene would be to have as many content information as we could about the items to enrich the classifier, as besides contextual information, specifically in those scenarios where unique context combinations are on the order of tens, some extra information is needed in order to provide to the classifier more notions about the items and thus avoid recommending lots of items with the same probability under a given context. However, for some PMS and specifically in the case we show, most of this information have not been consolidated in the same format yet, it is not accessible for legal reasons or sometimes it is not even linked to the item id in consume data (they might be used in the platform for different purposes and sometimes they are not in charge of the same section on the company).

3.3.1. Graph Convolutional Embeddings

In front of this situation, we propose to use the collaborative data available from **logged-users**, fit into a Graph Convolutional Neural Network (GCN), in order to leverage the bias that this kind of models capture and introduce it into the classification model. This way, we would be able to enrich the pure-contextual recommender just in a straightforward manner by leveraging the inductive bias that those models supply.

Specifically, this approach leverages the work proposed by Duran et al. [4] to use Graph Convolutional Embeddings in collaborative recommender systems. We created a *bi-parted graph* by selecting $\{userID - itemID\}$ from the **logged-users** interactions in order to build an adjacency matrix (A). Then, we fit this matrix A into a Graph Convolutional Embedding Layer in order to generate the item embeddings. After performing the convolution, those embeddings will be connected implicitly by user's nodes in such a way that the resulting model is biased by the graph neighborhood structure: items that are similar will have similar embeddings. Of course, this measure of similarity is what we want to take from the **logged-users** data of the OM.

To perform the embeddings we use the following equation:

$$E = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}HW \tag{1}$$

where E are the resulting items' embeddings, A is the adjacency matrix of the graph that we have just built from **logged-users**, H is the identity matrix (because for the first layer should be the node's features and we do not have content information for any node) and W are the weights we want to learn. Note that in GCN, the D is the degree matrix computed from the $\hat{A}$, which is just the A matrix with the self-connections added.

However, as the information that we want to propagate is the one referring to the item nodes, in order to avoid loosing information as we train the model and specifically for those items with few connections, we apply some modifications to the initial **GCE layer** and we end up with Equation 2.

The major component of graph convolution is the neighborhood message passing, which allows the supervision signal to flow in the graph and propagate through the edge. Recent works [7, 17] have demonstrated that stacking proper layers of convolution may lead to better performance, especially when the supervision signal is sparse. In RS data, the edges between nodes in the constructed graph are relatively sparse, as a result a single layer of convolution is not enough for sufficient information propagation. However, simply stacking more convolution layers would also increase the risk of over-fitting because the number of trainable parameters would also increase. To address the problem, we utilize a "batched" operation in one single layer to perform multi-hop information propagation with only one parameter matrix. We define the embedding matrix after k-hop propagation as:

$$E = (\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}})^k HW \tag{2}$$

This enables the node to get information from k-th order neighbourhood with the same number of parameters as a single naive graph convolution layer. Thus, it helps to alleviate the data sparsity problem of recommendation scenarios, specially in this case where the user nodes which connect several items are never actively trained.

3.4. Embedding classifier

The next step is to use these graph-based item embeddings to built a better context-based classifier. Our hypothesis is that by enriching the context information with a GCE embedding layer, we will allow the model to leverage the intrinsic bias that the original database actually holds.

To this end, we propose a new context-based item classifier architecture, called **embedding classifier**, which is represented in Figure 1. On the left side of the figure, it can be seen that the model takes as an input the one-hot encoding of each context in order to compute its respective embeddings and concatenate them, thus ending up with a matrix containing a context embedding in each row. At the same time, on the right side of the figure, the model will take as an input the graph embeddings of each of the items in the dataset, thus ending up with a matrix that will contain in each row an item embedding. Depending on whether we use GCE layer or vanilla embedding layer in order to compute the item embeddings, we will leverage the historical information (or not), respectively.

Then, each of the two matrices (the matrix containing the context embeddings and the matrix containing the item embeddings) are multiplied hence giving a vector of item probabilities for each given context. Finally, those vectors are fed into a linear layer so that the probability of each given item along different contexts is combined in order to end up with a final representation that will represent the probability of an item being consumed under a given combination of contexts.

To sum up, we propose a model that can be trained end-to-end by and which leverage all the previous collaborative data available (no content information is used) just

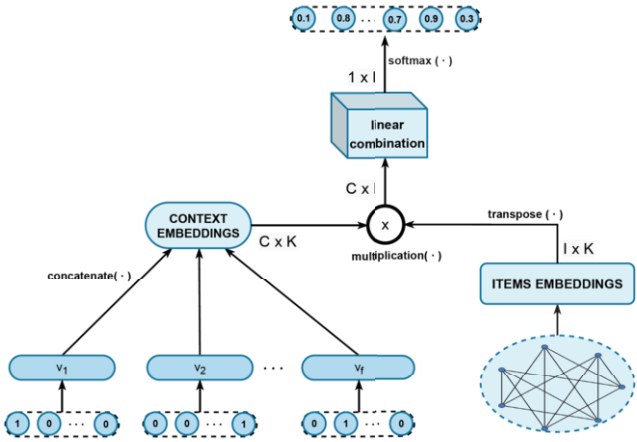

Figure 1. Schema providing an intuition of where GCE layer is applied into the **Embedding Classifier**, as explained in 4.1.

by previously building the A matrix which encode the bias contained into the *user-item* topology from the $logged - users$ data. So, the final model is a classifier where the output is a tensor containing the probability assigned to each item of being consumed under a given context combination.

4. Results

In this section we explain in detail the baseline models and the metrics we use in order to reach conclusions for the Catalan PMS dataset (TV3) in contextual recommendation. All the baseline models are state-of-the-art models we chose as realistic models that actually are used or suitable to be used by a public service broadcaster. Then, we compare each of the models with our proposed method (classifier + GCE). Note that we require the chosen models to be capable of being naturally modified to perform pure context-aware recommendations, which moreover implies in this case being treated as a classification problem. That is mainly the reason why we do not choose any recurrent model, as they would require some temporal information of the user consumption's to simulate a sequence that we actually do not have.

4.1. Baselines

We have chosen this baselines, which are used very often in PMS production scenarios for pure-contextual recommendations.

1. **Random**: This model is insensitive to data distribution. It uniformly recommends items independently of the context.
2. **MostPop**: MostPop stands for Most Popular items recommender. This model ignores must of the items and always recommends the K most popular items (the items that were consumed the most), independently of the context.

3. **MLP**: MLP stands for Multi-layer Perceptron classifier, which consists of three layers of units: an input layer representing the context, a hidden layer, and an output layer with one unit for each item to be recommended.
4. **Embedding Classifier (CEmb)**: This model we propose is inspired in Factorization Machines [13] with the aim of using embedding to characterize each item and each context option.

4.2. Parameter settings

To ensure a fair comparison of the models' performance, we train all of them by optimizing the Cross Entropy (CE) loss as the training function[2] with the Adam optimizer[6]. We use Bayesian Optimization[2] strategy in order to tune the hyperparameters and follow the same criteria in all cases. Thus, we determine the best hyperparameters for each of the models by tuning the learning rate on the range $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$; batch size on the range $\{256, 512, 1024, 2048, 4096\}$, and dropout on range $\{0, 0.15, 0.3, 0.5\}$. The embedding size is set to 32 for all models. We run all the experiments for a maximum of 150 epochs and perform early stopping when the loss stops decreasing for more than 10 consecutive epochs.

4.3. Evaluation protocol

When evaluating a recommender system, the goal is to generate a ranked list of k items by decreasing predicted score of how likely a user is to interact with them. We split 80% of the interactions for training set and 20% for test set. In fact, we sort by timestamp and left out for testing the last 20%, thus simulating the past and future times.

The performance of each model follows from assessing the recommendation lists provided for all users through a range of metrics. In terms of measuring how good a model is, we used standard offline top-k metrics:

- **Hit Rate (HR@k):** A recall-based metric, measuring whether the test item is in the top-k positions of the recommendation list provided by the RS to a user, averaged across all users.
- **Normalized Discounted Cumulative Gain (nDCG@k):** It is a rank-sensitive measure of quality of recommendations, which provides information on the ranking position of the ground truth sample; higher scores are assigned to the top-ranked items and can be regarded as a weighted version of HR@k. This metric can be computed as an average across all users [16].

In addition, we used one metric to assess the diversity of items offered to users across the item catalogue **(coverage)**, as it is very important for PMS to be capable of guaranteeing item diversity and personalization instead of just sticking to recommend popular content

- **Coverage (Cov@k):** is the fraction of items that the RS includes as recommendations across all users on the top$k$ recommendations. A RS that consistently recommended a few very popular items to all users would have a small coverage (i.e. ItemPop model), while a RS that is able to recommend the whole item catalogue to users will have $Cov@k = 1$ (i.e. random model).

[2] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html

4.4. Performance

In this sub-section we show the results for all different models in terms of the defined metrics in the previous subsection.

Non-Logged dataset			
	HR	**nDCG**	**Coverage**
Random	0.020	0.009	1.000
MostPop	0.039	0.017	0.018
MLP	0.041	0.019	0.061
CEmb	0.042	0.019	0.062
CEmb - GCE	**0.085**	**0.041**	**0.058**

Table 2. Off-line metrics on Catalan PMS test dataset. Acronyms are the same as in Section 4.3.

In the first two rows we show the results of two opposite models that can always be applied when approaching a pure cold-start scenario: Random and MostPop. On the one hand, we show that the random model systematically offers the lowest metrics in terms of succeeding (HR and NDCG), while offering a coverage of 100%, as it goes through all items of the database to offer recommendations. On the other hand, we see that recommending the most popular items leads in general terms to a very good performance. However, this generally means to reduce the number of items from database showed (coverage) to very small number. In fact, we can see that even the performance metrics (HR, NDCG) are high compared with the model complexity, the coverage is reduced to showing just the 1.8 % items from the database.

As we have mentioned in the previous sections, the trade-off between engaging the user to the platform and offering a wide variety of items to the users in order to ease the access to all media type of content to consumers, it really something to consider for PMS. Thus, the best model should not just have very good metrics in terms of performance (as could be MostPop model in some cases) but also try to enlarge coverage to ensure showing as much items as they can from the database without being randomly shown.

In the second and third row, we show two SOTA classifiers that we have chosen: one that do not use embeddings to characterize entities (each context and item) - MLP - and one that, in fact, does - CEmb. We observe in Table 2 that both models coexist in the same range of metrics, having the model which use embeddings to characterize entities a slightly better performance in terms of HR and coverage.

As we have explained, we have chosen the embeddings' classifier (CEmb) model as it is constituted by embeddings that characterize each entity (each context or item). By chosing a model that uses embeddings, we are able to substitute the vanilla embedding layer by the GCE layer and leverage past collaborative interactions. That is the model that we see in the last row, with the one we show that leveraging past information of logged users outperforms all other models in terms of HR and NDCG metrics for more than double. Besides, the coverage of items showed to users in recommendation is just slightly reduced, and hence we end up with a good trade-off that accomplish the policy that follow a PMS.

5. Conclusions

In this paper we have focused on one specific group of users that TV broadcasters must take into account when creating a recommender system: non-logged users. Not a lot of efforts have been put into facing this type of problem, which is basically building a pure-contextual recommender system that end up working in a scenario in which all data follows the cold-start problem distribution: no information at all about the user, session, etc. Thus, we approach the challenge of predicting recommendations for those type of users by using contextual information about the interaction and seeking to leverage data from those logged users, who we state that implicitly hold a bias that describe how data is willing to be consumed.

In the end, by working with a dataset of real interactions from a public Catalan TV broadcaster, we show in the results how building a contextual recommender system that maximizes the hit rate and leverage all data available can actually be accomplished in order to come up with better recommendations for unidentified users.

6. Acknowledgements

This research was partially funded by the Government of Catalonia's Agency for Business Competitiveness (ACCIÓ) under project PICAE (Comunitat RIS3Cat Media). P.G. and J.V. acknowledge funding from projects Nos. RTI2018-095232-B-C21 (MINECO/FEDER, UE) and 2017SGR1742 (Generalitat de Catalunya).

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[2] James Bergstra, Brent Komer, Chris Eliasmith, Dan Yamins, and David D Cox. Hyperopt: a python library for model selection and hyperparameter optimization. *Computational Science & Discovery*, 8(1):014008, 2015. .

[3] Xuegang Chen, Mingna Xia, Jieren Cheng, Xiangyan Tang, and Jialu Zhang. Trend prediction of internet public opinion based on collaborative filtering. In *2016 12th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*, pages 583–588. IEEE, 2016.

[4] Paula G. Duran, Alexandros Karatzoglou, Jordi Vitria, Xin Xin, and Ioannis Arapakis. Graph convolutional embeddings for recommender systems. *IEEE Access*, 9: 100173–100184, 2021. ISSN 21693536. . URL http://arxiv.org/abs/2103.03587.

[5] John Hannon, Mike Bennett, and Barry Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206, 2010.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv e-print*, 2014.

[7] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17.

[8] Miklas Strøm Kristoffersen, Sven Ewan Shepstone, and Zheng-Hua Tan. The importance of context when recommending tv content: Dataset and algorithms. *IEEE Transactions on Multimedia*, 22(6):1531–1541, 2019.

[9] Augusto Q Macedo, Leandro B Marinho, and Rodrygo LT Santos. Context-aware event recommendation in event-based social networks. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 123–130, 2015.

[10] Ante Odić, Marko Tkalčič, Jurij F Tasič, and Andrej Košir. Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 25(1):74–90, 2013.

[11] Roberto Pagano, Paolo Cremonesi, Martha Larson, Balázs Hidasi, Domonkos Tikk, Alexandros Karatzoglou, and Massimo Quadrana. The contextual turn: From context-aware to context-driven recommender systems. In *Proceedings of the 10th ACM conference on recommender systems*, pages 249–252, 2016.

[12] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 265–268, 2009.

[13] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010. .

[14] Nícollas Silva, Heitor Werneck, Thiago Silva, Adriano CM Pereira, and Leonardo Rocha. A contextual approach to improve the user's experience in interactive recommendation systems. In *Proceedings of the Brazilian Symposium on Multimedia and the Web*, pages 89–96, 2021.

[15] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z Sheng, Mehmet A Orgun, and Defu Lian. A survey on session-based recommender systems. *ACM Computing Surveys (CSUR)*, 54(7):1–38, 2021.

[16] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. A theoretical analysis of ndcg type ranking measures. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *Proceedings of the 26th Annual Conference on Learning Theory*, volume 30 of *Proceedings of Machine Learning Research*, pages 25–54, Princeton, NJ, USA, 12–14 Jun 2013. PMLR. URL https://proceedings.mlr.press/v30/Wang13.html.

[17] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[18] Jiancan Wu, Xiangnan He, Xiang Wang, Qifan Wang, Weijian Chen, Jianxun Lian, and Xing Xie. Graph convolution machine for context-aware recommender system. *Frontiers of Computer Science*, 16(6):1–12, 2022.

[19] Jinfeng Zhong and Elsa Negre. Towards improving user-recommender systems interactions. In *2022 IEEE/SICE International Symposium on System Integration (SII)*, pages 816–820, 2022. .