Revisiting the performance evaluation of knowledge-aware recommender systems: are we making progress?

Marina Ananyeva[1,2,*,†], Oleg Lashinin[2,*,†] and Maria Kuznetsova[2,*,†]

[1]National Research University Higher School of Economics, 20 Myasnitskaya St, Moscow, 101000, Russian Federation
[2]Tinkoff, 38A Khutorskaya St, Moscow, 127287, Russian Federation

Abstract

Knowledge-aware recommender systems incorporate side information to improve recommendation performance. The authors of new algorithms are usually focused on developing new ideas behind the proposed methods and comparing their models with existing knowledge-aware recommender models. Meanwhile, some commonly used state-of-the-art general top-n recommender models are ignored as potential baselines. In this study, we compare previously proposed knowledge-based recommender systems with simple and computationally effective recommender models (EASE and ItemKNN) that do not use any additional information about users and items. Our results on three datasets show that claimed effect of using side information in recommender systems is still questionable.

Keywords

recommender systems, knowledge-based models, evaluation

1. Introduction

In recent years, knowledge-aware recommender systems have gained great popularity and development. Most of the approaches are based on knowledge graphs (KG-based). Other types of knowledge-aware recommender models are out of the scope of this work. A typical knowledge graph contains two subgraphs. The first one is the user-item graph, in which vertices correspond to users and items. They are connected by edges that reflect interactions between users and items. The second subgraph contains additional information about users and items. It has intersecting nodes with the first graph and may have other additional vertices. Edges of the second subgraph represent side information that connects some nodes of the second subgraph. As a result, KG-based models not only use interactions between users and items but also enrich the models with various types of side information. Such algorithms rely on collaborative signals and additionally model other hidden causes of interactions. For instance, a movie recommender model may recommend film X because the target user prefers films of a similar genre or likes other films from the same director. In this example, the model cannot retrieve such causes purely from interactions. Therefore, the use of knowledge graphs is reasonable and should improve the quality of recommendations.

However, it is difficult to estimate the relative gain from the use of additional information. In the field of recommender systems, there are no standard testing protocols, mandatory baselines for comparison with new algorithms, and unified quality ranking metrics. Researchers, in particular, use a variety of methods to split data for training and testing samples [1]. Ranking metrics are frequently computed for recommendation lists of differing k lengths [2]. All this leads to the fact that the results of models on the same metrics and datasets may differ from paper to paper. Therefore, it is impossible to say unequivocally about the effectiveness of existing KG-based state-of-the-art models.

Moreover, when the authors propose a new KG-based model they often compare it only with other KG-based models, or with weak conventional top-n recommender baselines. We have found that some of the matrix factorization-based approaches are often used to demonstrate the superiority of using knowledge graphs. However, according to [2], well-tuned baselines such as widely-known ItemKNN [3], graph-based models without additional information, and even linear models can outperform deep learning approaches under the same task. Additionally, according to benchmarks [4], simple baselines are computationally effective and often more scalable to real-world applications. Thus, the benefit of using knowledge graphs in recommender systems should be further studied.

In this study, we compare highly cited KG-based models with computational effective collaborative filtering (CF-based) models, namely ItemKNN and EASE [5]. The

4th Edition of Knowledge-aware and Conversational Recommender Systems (KaRS) Workshop @ RecSys 2022, September 18–23 2023, Seattle, WA, USA.
*Corresponding author.
†These authors contributed equally.
m.ananyeva@tinkoff.ru (M. Ananyeva); o.a.lashinin@tinkoff.ru (O. Lashinin); ext.mekuznetsova@tinkoff.ru (M. Kuznetsova)

contributions of this paper can be summarized as follows:

- We aggregated information about datasets, ranking metrics, and evaluation strategies. For experiments, we included baselines for 12 graph-based recommender models (8 of them use knowledge graphs) published from 2016 to 2020. Also, we discuss some methodological issues in the evaluation of KG-based models.
- We conducted experiments on three real-world datasets with hyperparameters tuning both for baselines and KG-based models. Our study sheds light on the questionable recommendation quality of KG-based models.

2. Related work

2.1. Graph-based models

There are approaches that use the graph structure of interactions without additional information. The idea of exploiting only user-item graphs for recommendation is researched for a long time. Users and items are represented as nodes and interactions between them as edges. Early approaches like ItemRank [6], P3α [7], and P3β [8] utilize random walks to propagate user preferences. Lately, graph neural networks were applied to retrieve high-order connections between nodes and retrieve non-trivial collaborative signals. To do this, such models as PinSage [9], GC-MC [10], and NGCF [11] are developed based on graph convolution operation. Recently, simplified versions of Graph Convolution Networks (GCN) were proposed. For example, the authors of LightGCN [12] removed nonlinearities and collapsing multiple weight matrices and outperformed previous state-of-the-art NGCF [11]. Lately, researchers presented SGL [13] with a self-supervised task and additional data augmentation.

2.2. KG-based models

Models with external knowledge can be divided into 3 main groups: embedding-based (KGE), path-based, and hybrid methods [14].

Embedding-based methods use information from the knowledge graph to enrich the representation of users and items. As a rule, the main idea of such models is based on the integration of standard collaborative filtering and embeddings from the knowledge graph. For example, CKE [15] combines various types of additional data about items: structural, textual, and visual information. All representations are then aggregated together to calculate final recommendations. In KTUP [16] the researchers do a graph completion task. MKR [17] model connects two tasks - constructing a recommender system and a vector representation of a KG, that allows solving them simultaneously in multitasking learning mode.

Path-based approaches exploit the idea that similar entities in a graph are connected by close relationships. For example, PER [18] treats KG as a heterogeneous information network and extracts latent features from meta-paths of varying length and nature to represent heterogeneous relationships between users and elements. HeteRec [19] leverages the meta-path similarities to enrich the user-item interaction matrix. However, the first path-based approaches involved the manual formation of meta-paths, which led to sub-optimal results and limited their use for different recommendation scenarios.

In [20], the authors proposed combining KGE and path-based approaches into the RippleNet model. The general idea of the method is that the user's embedding is formed from the embeddings of items with which the user interacted in the past, as well as from their neighbors on a graph with a given depth. Other examples of hybrid models are KGCN [21], where the representation of the object is formed by aggregating the embeddings. In KGNNLS [22] authors prove that label smoothness regularization is equivalent to the label propagation problem and use a leave-one-out loss to evaluate the importance of each relationship type to the user. KGAT [23] model uses the attention mechanism to distinguish the importance of the influence of different neighbors.

2.3. Surveys of KG-based RecSys models

There are a few comprehensive surveys about graphs-based recommender models. In paper [24], researchers provide a comprehensive overview of graph-based models for recommendations. Recent work [25] analyzes existing work in KG-based recommendations and outlines future directions. But this paper does not contain independent experiments with well-tuned collaborative filtering baselines. Both papers do not include extra experiments with tuning hyperparameters of baselines. Although there are a lot of evaluation research studies with a comparison of conventional top-N recommendation models [2, 26, 27], there are no such articles on KG-based models. To the best of our knowledge, we are the first to provide experiments with a wide range of KG-based models and compare them with strong CF-based baselines.

3. Experiments

A typical knowledge graph can be divided into two subgraphs. The first one is a user-item graph, gathered from interaction data. And the second subgraph represents additional knowledge. To study the performance of using additional information in recommender models, we are

Table 1
Graph-based recommender models with or without using side information. Non-graph baselines are in **bold** and target models baselines are underlined. A bipartite graph G_1 is defined as a set of triplets $\{(u, y_{ui}, i) | u \in U, i \in I\}$, where U, I are sets of users and objects, respectively vertices, and connections y_{ui} represent interactions. In the User-Item graph, in addition to interactions, vertices with external features of the item are added.

Model	Year	#citations*	Datasets	Baselines	Metrics	Graph	Motivation
Models with bipartite user-item graph							
NGCF [11]	2019	797	Gowalla, Yelp2018, Amazon-Books	**MF, NeuMF, CMN,** HOP-Rec, GC-MC, PinSage	Recall@20, NDCG @20	Bipartite	High-order embedding propagation
DGCF [28]	2020	91	Gowalla, Yelp2018, Amazon-Books	**MF**, GC-MC, NGCF, DisenGCN, **MacridVAE**	Recall@20, NDCG @20	Bipartite	Disentangled representation
LightGCN [12]	2020	492	Gowalla, Yelp2018, Amazon-Books	NGCF, **Mult-VAE**, GRMF	Recall@20, NDCG @20	Bipartite	Without complex operations and non-linearity
SGL [13]	2021	59	Yelp2018, Amazon-Books, Alibaba-iFashion	NGCF, LightGCN, **Mult-VAE**, **DNN+SSL**	Recall@20, NDCG @20	Bipartite	Self-supervision
KGE models							
CKE [15]	2016	941	MovieLens-1M IntentBooks	**BPRMF**, BPRMF+TransE, PRP, PER, **LibFM**, **CMF**, **CTR** **BPRMF+SDAE**	MAP@K, Recall@K K: 20, 40, 60, 80, 100	Item	Structural, textual, visual embeddings + CF
CFKG [29]	2018	181	Amazon	**BPR, BPR_HFT, VBPR, DeppCoNN**, CKE, **JRL**	Precision, Recall, HR, NDCG K:10	User-Item	Heterogenous user-item knowledge graph
RippleNet [20]	2018	471	MovieLens-1M Book-Crossing Bing-News	CKE, **SHINE, DKN,** PER, LibFM + TransR, Wide&Deep	Precision@K, Recall@K, F1@K K: 1, 2, 5, 10, 20, 50, 100	Item for each user	Preference propagation
KGCN [21]	2018	280	MovieLens-20M Book-Crossing Last.FM	**SVD, LibFM**, LibFM + TransE, PER, CKE, RippleNet	Recall@K K: 1, 2, 5, 10, 20, 50, 100	Item for each user	Graph Convolutional
MKR [17]	2019	214	MovieLens-1M Book-Crossing Last.FM Bing-News	PER, CKE, DKN, RippleNet, LibFM + TransR, Wide&Deep	Recall@K Precision@K K: 2, 5, 10, 20, 50	Item	Multi-task learning, cross&compress unit
Hybrid graph-based models							
KTUP [16]	2019	212	MovieLens-1M	**FM, BPRMF,** CFKG, CKE, CoFM	Recall@10, Precision@10 F1 score@10, Hit ratio@10, NDCG @10	Item	Graph completion, preference translation
KGAT [23]	2019	605	Amazon-Books Last-FM Yelp2018	**FM, NFM,** CKE, CFKG, MCRec, RippleNet, GC-MC	Recall@20, NDCG @20	Bipartite + item	Attention
KGNN-LS [22]	2019	205	MovieLens-20M, Book-Crossing, Last.FM, Dianping-Food	**SVD, LibFM,** LibFM+TransE, PER, CKE, RippleNet	Recall@K K: 2, 10, 50, 100	Item for each user	Label smoothing

*The data is current as of 2022-03-25.

firstly focused on graph models without features. Then, we move on to models with additional information.

Thus we perform experiments to answer the following research questions:

1. **RQ1:** Does the graph structure help to better capture the collaborative signal from the interaction data?
2. **RQ2:** Do graph-based models that use additional external information perform better than models without features?

3.1. Datasets

We consider three real-world datasets from different areas of application and varying sparsity degrees.

MovieLens-1M[1]: This dataset contains ratings for movies from 1 to 5. We use 20-core filtering and consider any grade as positive feedback to achieve a lesser degree of sparsity of the interaction matrix.

Amazon-Books[2]: This dataset contains information about the ratings of books purchased by Amazon users. 30-core filtering is used to ensure the dataset's quality and match the available computing resources.

TTRS [30] is a real dataset of financial transactions of 50,000 randomly selected clients of Tinkoff bank, covering spending in all areas of life - from buying food and household appliances to paying for various forms of leisure. Users with less than 10 interactions were excluded.

[1]https://grouplens.org/datasets/movielens/1m/

[2]http://jmcauley.ucsd.edu/data/amazon/

Table 2
Performance comparison on three datasets. The best model in each semantic group is in **bold**, the models are sorted in ascending chronological order. The best model for each dataset and metric pair is in **green**, the second best model in **violet**, and the third best model in **blue**.

Method	MovieLens-1M		TTRS		Amazon-books	
	Recall@10	NDCG@10	Recall@10	NDCG@10	Recall@10	NDCG@10
Pop	0.072	0.126	0.233	0.134	0.012	0.011
ItemKNN [3]	0.163	0.126	0.322	0.240	**0.149**	**0.149**
EASE [5]	**0.205**	**0.314**	**0.389**	**0.290**	**0.158**	**0.157**
NGCF [11]	0.172	0.268	0.343	0.254	0.082	0.076
DGCF [28]	0.172	0.265	0.345	0.254	0.087	0.082
LightGCN [12]	**0.173**	**0.269**	**0.352**	**0.262**	0.088	0.083
SGL [13]	**0.179**	**0.271**	**0.354**	**0.264**	**0.103**	**0.097**
CKE [15]	0.165	0.263	0.349	0.257	0.077	0.072
CFKG [29]	0.154	0.241	0.324	0.240	0.067	0.062
RippleNet [20]	0.121	0.203	0.301	0.222	0.036	0.033
KGCN [21]	0.146	0.234	0.316	0.235	0.051	0.046
MKR [17]	0.139	0.224	0.331	0.246	0.047	0.043
KTUP [16]	0.159	0.251	0.331	0.244	0.061	0.056
KGAT [23]	0.169	0.273	0.342	0.254	0.076	0.071
KGNNLS [22]	0.146	0.235	0.317	0.234	0.052	0.047

3.2. Experimental Settings

Evaluation metrics. We use the standard metrics NDCG@k and Recall@k averaged over the entire sample to assess the recommendations' quality. Due to paper space limitations, we report metrics only with k = 10.

Evaluation protocol. For each dataset, 80% of the interactions for each user were randomly selected as the training sample. We also leave 10% of interactions for validation and 10% for the test set. One negative example is randomly selected from a uniform distribution for each interaction during the training phase. At the stage of validation and testing, we rank positive examples relative to all items in the dataset.

Baselines. Three popular non-graph models based on statistics were chosen as baselines:

1. Pop is a recommendation of the most popular items in the dataset. Popularity is calculated by the frequency of the object's occurrence among all interactions;
2. ItemKNN [3] – an item-based approach to recommending objects that are close in the cosine similarity measure;
3. EASE [5] is a state-of-the-art linear model based on the idea of an autoencoder. It has an analytical solution for a convex loss function.

Hyperparameter Fitting. To review the current state of KG-based models, we took the popular open library for building recommender systems RecBolee[3] [4]. We took all the graph models implemented in Recbole of version 1.0.0 and analyzed the relevant articles for the datasets, metrics, baselines, data preprocessing, and pipeline for building recommender systems. Table 1 presents the summary statistics.

We found a few issues with evaluating KG-based models and consequently claimed the value of additional knowledge for recommendation quality.

Firstly, the authors include matrix factorization (BPR, SVD) and factorization machines (FM, LibFM) as collaborative filtering baselines. Meanwhile, KNN-based, VAE-based, and Item-based models are ignored. Moreover, MKR [17] and RippleNet [20] even are not compared with CF baselines. Therefore, we cannot conclude that KG-based models outperform algorithms based only on user-item interactions.

Secondly, researchers from most papers do not properly tune hyperparameters of CF-based models. According to [2], it leads to the weak performance of baselines and possible superiority of proposed KG-based methods.

To fill this gap, we conduct comprehensive experiments with tuning both baselines and KG-based models.

Implementation details. All calculations were performed on the Tesla V100 GPU, experiments were carried out on the basis of the RecBole framework. For all models, we tune hyperparameters with random greed-search optimizing NDCG@10 within 24 hours for each

[3]https://github.com/RUCAIBox/RecBole

Table 3
Training time comparison on the MovieLens-1M dataset.

Model	Best model						Overall tuning	
	#epochs	Sec./epoch		Overall training time, min.	GPU RAM, GB		#hparams/ greed size	Tuning time, h.
		Train	Eval		Train	Eval		
Pop	1	1,95	6,66	0,14	0,00	0,00	-	0,0
ItemKNN	1	1,68	12,20	0,23	0,00	0,00	24/24	0,5
EASE	1	1,65	8,68	0,17	0,00	0,00	6/6	0,7
NGCF	93	23,32	5,79	45,12	0,31	0,32	115/2400	24
DGCF	168	139,22	5,93	406,42	4,92	4,92	15/180	24
LightGCN	262	7,58	5,73	58,12	0,17	0,17	80/80	24
SGL	92	29,35	5,64	53,65	0,41	0,41	79/100	24
CKE	239	8,55	5,91	57,60	0,39	0,39	60/60	19,3
CFKG	83	3,98	10,05	19,41	0,17	0,17	60/60	15
RippleNet	10	30,47	9,06	6,59	0,99	1,00	83/100	24
KGCN	66	6,83	11,51	20,17	0,16	0,18	5/5	5
MKR	70	10,91	14,43	29,56	0,44	0,56	96/180	24
KTUP	212	8,64	13,06	76,67	0,18	0,18	101/240	24
KGAT	98	27,13	5,64	53,52	1,18	2,72	89/375	24
KGNNLS	66	12,82	11,69	26,96	0,19	0,19	5/5	5,1

"Overall training time, min." = ("Train sec./epoch" + "Eval sec./epoch") * "#epochs" / 60

"#hparams" - number of randomly iterated hyperparameters from the grid

"Tuning time" - time spent iterating hyperparameters from the grid (maximum - 24 hours)

model. Hyperparameter ranges were taken from [4]. Our experiments can be reproduced using our open-source repository[4]. Model launch code is available there, as well as hyperparameter grids and fitted optimal values.

3.3. Results

Table 2 presents NDCG@10 and Recall@10 values in our experiments.

RQ1: Graph models in all datasets outperform the Pop baseline but demonstrate poor performance compared to the state-of-the-art non-graph model EASE [5] and ItemKNN [3] on Amazon Books. Although the EASE model was published in 2019 and could not be used as a baseline in most models, the authors of the DGCF [28], LightGCN [12], and SGL [13] should have considered including it, as well as ItemKNN [3], in their analysis. Even though EASE and ItemKNN use less information based on interaction data alone, they are often able to model a behavioral signal better than knowledge graph models and are easier to implement.

RQ2: Approaches with a bipartite user-item graph that are not enhanced with additional connections with item characteristics demonstrated the next best quality after EASE. Despite the fact that both models with a bipartite user-item graph and a knowledge graph incorporate information on interactions, the experimental results suggest that KG-models capture the collaborative signal worse. Along with the model's complication, it may be anticipated that more emphasis is devoted to the KG part of the learning process and less to the collaborative one. This indirectly verifies the advantages of the most basic CKE [15] KG-model with a distinct CF part.

Table 3 displays training time and memory costs for all models on the MovieLens-1M dataset with optimal hyperparameters. It can be shown that all baseline models do not require the usage of a GPU and are trained in less than one minute. The inclusion of graphs in models increases training time by more than 40 times. RippleNet, for example, had the quickest training time on the MovieLens-1M dataset, with a result of 6.6 minutes, and the longest was DGCF, which trained on the Amazon-Books dataset for more than 8 hours.

4. Conclusion

In this study, we examined the quality and required resources for training the developed KG-based recommendation models from different perspectives. Furthermore, we uncovered a few methodological issues that contribute to the stated effectiveness of KG-based models, which may be addressed by incorporating simple linear base-

[4]https://github.com/kg-based-recsys-eval/kg_based_recsys_eval

lines and adjusting its hyperparameters in the experimental set-up. Our comprehensive experiments on three real-world datasets reveal that a basic linear model EASE outperforms any KG-based approach. Moreover, the early-adopted ItemKNN approach shows better quality in making recommendations than some graph models on several datasets. Our findings indicate that researchers should include and adjust such simple baselines in their experiments and compare new algorithms to them.

References

[1] Z. Meng, R. McCreadie, C. Macdonald, I. Ounis, Exploring data splitting strategies for the evaluation of recommendation models, in: Fourteenth ACM conference on recommender systems, 2020, pp. 681–686.

[2] M. F. Dacrema, P. Cremonesi, 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, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 101–109. URL: https://doi.org/10.1145/3298689.3347058. doi:10.1145/3298689.3347058.

[3] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th international conference on World Wide Web, 2001, pp. 285–295.

[4] W. X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, et al., Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 4653–4664.

[5] H. Steck, Embarrassingly shallow autoencoders for sparse data, in: The World Wide Web Conference, 2019, pp. 3251–3257.

[6] M. Gori, A. Pucci, V. Roma, I. Siena, Itemrank: A random-walk based scoring algorithm for recommender engines., in: IJCAI, volume 7, 2007, pp. 2766–2771.

[7] C. Cooper, S. H. Lee, T. Radzik, Y. Siantos, Random walks in recommender systems: exact computation and simulations, in: Proceedings of the 23rd international conference on world wide web, 2014, pp. 811–816.

[8] B. Paudel, F. Christoffel, C. Newell, A. Bernstein, Updatable, accurate, diverse, and scalable recommendations for interactive applications, ACM Transactions on Interactive Intelligent Systems (TiiS) 7 (2016) 1–34.

[9] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 974–983.

[10] R. v. d. Berg, T. N. Kipf, M. Welling, Graph convolutional matrix completion, arXiv preprint arXiv:1706.02263 (2017).

[11] X. Wang, X. He, M. Wang, F. Feng, T.-S. Chua, Neural graph collaborative filtering, in: Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval, 2019, pp. 165–174.

[12] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 639–648.

[13] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, X. Xie, Self-Supervised Graph Learning for Recommendation, Association for Computing Machinery, New York, NY, USA, 2021, p. 726–735. URL: https://doi.org/10.1145/3404835.3462862.

[14] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, Q. He, A survey on knowledge graph-based recommender systems, IEEE Transactions on Knowledge and Data Engineering (2020).

[15] F. Zhang, N. J. Yuan, D. Lian, X. Xie, W.-Y. Ma, Collaborative knowledge base embedding for recommender systems, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 353–362.

[16] Y. Cao, X. Wang, X. He, Z. Hu, T.-S. Chua, Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences, in: The world wide web conference, 2019, pp. 151–161.

[17] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, M. Guo, Multi-task feature learning for knowledge graph enhanced recommendation, in: The world wide web conference, 2019, pp. 2000–2010.

[18] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, J. Han, Personalized entity recommendation: A heterogeneous information network approach, in: Proceedings of the 7th ACM international conference on Web search and data mining, 2014, pp. 283–292.

[19] X. Yu, X. Ren, Y. Sun, B. Sturt, U. Khandelwal, Q. Gu, B. Norick, J. Han, Recommendation in heterogeneous information networks with implicit user feedback, in: Proceedings of the 7th ACM conference on Recommender systems, 2013, pp. 347–350.

[20] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, M. Guo, Ripplenet: Propagating user preferences on

the knowledge graph for recommender systems, in: Proceedings of the 27th ACM international conference on information and knowledge management, 2018, pp. 417–426.

[21] H. Wang, M. Zhao, X. Xie, W. Li, M. Guo, Knowledge graph convolutional networks for recommender systems, in: The world wide web conference, 2019, pp. 3307–3313.

[22] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, Z. Wang, Knowledge-aware graph neural networks with label smoothness regularization for recommender systems, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 968–977.

[23] X. Wang, X. He, Y. Cao, M. Liu, T.-S. Chua, Kgat: Knowledge graph attention network for recommendation, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 950–958.

[24] S. Wu, F. Sun, W. Zhang, X. Xie, B. Cui, Graph neural networks in recommender systems: a survey, ACM Computing Surveys (CSUR) (2020).

[25] J. Chicaiza, P. Valdiviezo-Diaz, A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions, Information 12 (2021) 232.

[26] V. W. Anelli, A. Bellogín, T. Di Noia, D. Jannach, C. Pomo, Top-n recommendation algorithms: A quest for the state-of-the-art, arXiv preprint arXiv:2203.01155 (2022).

[27] S. Rendle, W. Krichene, L. Zhang, Y. Koren, Revisiting the performance of ials on item recommendation benchmarks, arXiv preprint arXiv:2110.14037 (2021).

[28] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, T.-S. Chua, Disentangled graph collaborative filtering, in: Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval, 2020, pp. 1001–1010.

[29] Y. Zhang, Q. Ai, X. Chen, P. Wang, Learning over knowledge-base embeddings for recommendation, arXiv preprint arXiv:1803.06540 (2018).

[30] S. Kolesnikov, O. Lashinin, M. Pechatov, A. Kosov, Ttrs: Tinkoff transactions recommender system benchmark, arXiv preprint arXiv:2110.05589 (2021).