Augmenting Graph Convolutional Networks with Textual Data for Recommendations

Sergey Volokhin[1](✉), Marcus D. Collins[2], Oleg Rokhlenko[2], and Eugene Agichtein[1,2]

[1] Emory University, Atlanta, Georgia, USA
svolokh@emory.edu
[2] Amazon, Seattle, WA, USA
{collmr,olegro,eugeneag}@amazon.com

Abstract. Graph Convolutional Networks have recently shown state-of-the-art performance for collaborative filtering-based recommender systems. However, many systems use a pure user-item bipartite interaction graph, ignoring available additional information about the items and users. This paper proposes an effective and general method, *TextGCN*, that utilizes rich textual information about the graph nodes, specifically user reviews and item descriptions, using pre-trained text embeddings. We integrate those reviews and descriptions into item recommendations to augment graph embeddings obtained using LightGCN, a SOTA graph network. Our model achieves a 7-23% statistically significant improvement over this SOTA baseline when evaluated on several diverse large-scale review datasets. Furthermore, our method captures semantic signals from the text, which are not available when using graph connections alone.

Keywords: Graph convolutional networks · Product recommendations · Textual augmentation

1 Introduction

Graph neural network (GNN) approaches to recommendation models have grown in popularity in recent years [20], which is natural since so much of the information in these systems is easily mapped to a graph structure. While there is still some controversy over whether graph-embedding methods outperform more conventional recommendation systems [4], the appeal of GNN systems is strong. It has long been clear that side information and additional knowledge, typically social connections between users, or structured knowledge about items, enhance any recommendation system [19]. However, the use of *unstructured* information about items or users has lagged, despite the availability of vast quantities of unstructured text in the form of user reviews and item descriptions. We are aware of only a couple of examples where such unstructured information has been used in GNN recommender systems [16].

Our intuition is that unstructured review text and item descriptions capture a great deal of semantic and behavioral information unavailable from the purely topological structure of a user-item interaction graph. We posit that this unstructured text may also contain information that can't be found in conventional knowledge graphs either. For instance, particular users may express what they like about items differently. We not only want to find similar users in terms of *what* items they like or what actors or characters, or attributes they seem to gravitate towards. We want to find similar users in terms of *how* they describe those items and attributes.

At the same time, many GNN recommender systems are increasingly complex, while in at least some cases, it has been shown that the sophisticated mixing and attention mechanisms used might even hinder recommendation accuracy [9]. Therefore, we seek to take the simplest approach that we can find to incorporate unstructured review and item description data into a GNN framework. We will show that a simple means of incorporating unstructured text into a GNN recommender improves the performance of a popular baseline system, LightGCN [9], by a similar amount as much more sophisticated approaches. In summary, our contributions are:

1. We explore ways to augment interaction-based Graph Recommender Systems with textual information for improved node representation and introduce a simple and general approach for integrating both graphical and textual representations of users and items.
2. We experimentally demonstrate the effectiveness of our combined model for recommendation performance, with 6-21% improvements across the evaluation metrics.

We also release the code[3] we have written to the scientific community for transparency and reproducibility.

2 Related Work

Recommending items to users is a naturally graph-oriented problem, and Graph Convolution Networks have recently achieved significant gains in recommender systems [20]. A few of these systems attempt to incorporate additional information about the users or items in the graph, with some success. We discuss them in this section to put our contributions in context.

Many systems that use additional information about entities or users do so by augmenting the user-item graph with additional nodes. For instance, TGCN [1] includes a third class of nodes called *tags* which encode additional structured metadata. KGAT [17] includes in the graph categorical entities connected to items (for example, actor or director entities are connected to movie items). Mei *et al.* [12] choose instead to connect additional entity nodes directly to users, constructing an interaction graph with user-entity and user-item edges. KCAN [3] also attempts to encode knowledge graphs alongside the user-item interaction graph but uses much more complex methods to incorporate that information into both user and item representations; it achieves mostly minor improvements over KGAT.

MKGAT (for Multi-modal KGAT) [16], is the most similar approach to ours. MKGAT first encodes the user-item interaction graph to produce user and item embeddings. It then concatenates those with embeddings of text and images. These first two steps are similar to our approach. However, MKGAT then adds an additional graph attention network. We use much simpler linear layers or gradient-boosted decision trees to combine the text and interaction graph vectors into a regression and achieve higher relative improvements over our baselines. MKGAT does not use review or product description text but uses the text associated with entities to build their representation. Moreover, none of the additional information is used directly to augment user representations.

Unlike all methods described above, our approach uses additional information to augment both user and item representations.

While it is well known that side information improves graph recommender performance [16,17,20], our model is distinguished in two ways. First, we use raw text from reviews and descriptions, which requires no processing. Second, we have taken a much simpler approach to incorporate that text, which proves equally effective in terms of the relative improvement of each recommendation metric. For instance, MKGAT [16] compares its base model with and without text inputs and finds that using unstructured text improves recall by 3.1% and nDCG by 3.5%. As shown below, the simple approach of TextGCN improves recall @20 by 8-18% and nDCG@20 by 10-31% over LightGCN.

3 Methodology

In this section, we describe our TextGCN framework, which uses the original LightGCN framework as a starting point. First, we discuss two baseline improvements, not involving text, which we have applied to both LightGCN and TextGCN. We sought to improve on the original baseline to help confirm that the improvements we observe when including text indeed are due to the additional information contained in the text over what is available in the graph. Next, we describe how the textual information is used to improve upon the baselines in our proposed model.

3.1 Non-text Related Improvements

The LightGCN paper [9] primarily focused on simplifying the general GNN architecture and left several optimizations unexplored.

Activation Function. We replaced the softplus activation function used in LightGCN with SELU [11] when calculating the Bayesian Personalized Ranking (BPR) loss [15]. Using SELU yielded better results for both our baseline models and those incorporating text features. Therefore every result in this work was computed using SELU as an activation function.

[3] `https://github.com/sergey-volokhin/TextGCN`

Negative Sampling. The original LightGCN work randomly sampled one negative example for each positive example per user. We conducted several experiments with more complex sampling functions and report below results with the best of these, Dynamic Negative Sampling (DNS) [21], which improved all metrics. DNS first ranks all the items and then selects negatives with the lowest score. Details about these experiments are shown in §5.1.

3.2 TextGCN

Table 1. Notation used in this paper

Symbol	Definition
d_i	vector or textual description of item i
$r_{u,i}$	vector of review written by user u about item i
$\mathcal{N}_x$	set of neighbors of node x
$\mathbf{t}_x$	textual representation of node x
$\mathbf{e}_x$	LightGCN vector representing node x

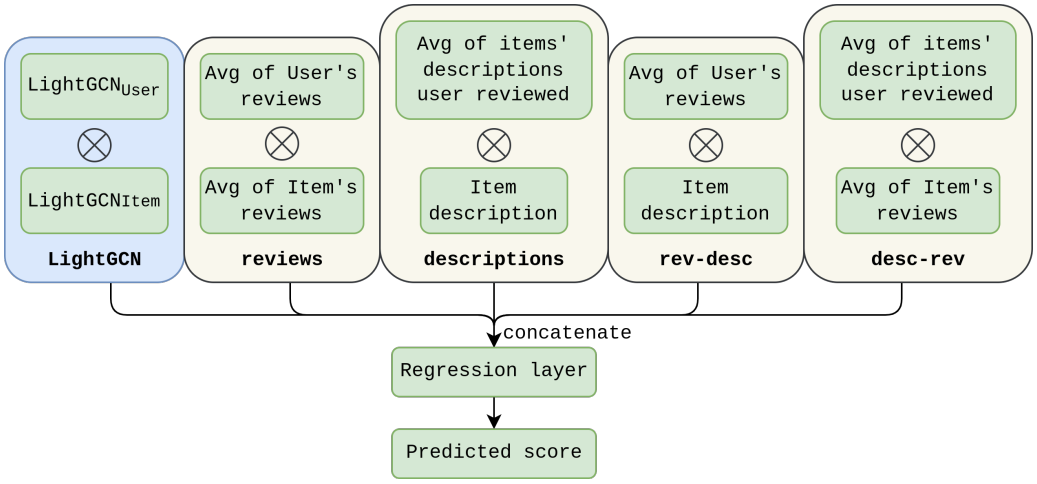

Fig. 1. Architecture of the proposed model. We use 5 features, 4 of which involve textual information (reviews and item descriptions). The LightGCN model is pretrained on user-item interaction graph and frozen, so the first feature does not change when training the regression layer. $\otimes$ represents dot product

Figure 1 shows the architecture of the model. First, we train and freeze the LightGCN model. Then we combine textual representations of items and users to create 4 additional features. The features are mathematically defined in Table 2 using notations from Table 1. Finally, we train a regression layer which predicts the scores for user-item pairs.

We experimented with using both reviews and items' descriptions (see §4 for a description of the data sources) in both the user and item node representations.

We can represent *users* by using the average of the reviews they have written (Eqn. 1) or by using the average of the descriptions of the items they have reviewed (Eqn. 2):

$$\mathbf{t}_u = \frac{\sum_{j \in \mathcal{N}_u} r_{u,j}}{|\mathcal{N}_u|} \equiv avg(r_u) \tag{1}$$

$$\mathbf{t}_u = \frac{\sum_{j \in \mathcal{N}_u} d_j}{|\mathcal{N}_u|} \equiv avg_u(d) \tag{2}$$

We can represent *items* by using either average of the reviews written about them (Eqn. 3) or using their descriptions (Eqn. 4):

$$\mathbf{t}_i = \frac{\sum_{v \in \mathcal{N}_i} r_{v,i}}{|\mathcal{N}_i|} \equiv avg(r_i) \tag{3}$$

$$\mathbf{t}_i = d_i \tag{4}$$

Table 2. All features used in the TextGCN model, and the average weights of the corresponding neurons in the regression layers across all datasets.

name	feature	weight in pred. layer
LightGCN	$\mathbf{e}_u \cdot \mathbf{e}_i$	1.47 ± 0.08
reviews	$avg(r_u) \cdot avg(r_i)$	8.66 ± 4.45
descriptions	$avg_u(d) \cdot d_i$	24.44 ± 2.27
rev-desc	$avg(r_u) \cdot d_i$	-12.80 ± 2.79
desc-rev	$avg_u(d) \cdot avg(r_i)$	-7.23 ± 4.66

We use those representations to create features for the model, the list of features that the model uses can be found in Table 2. Each feature is constructed by applying dot product on different user and item representations, and is then fed into a regression layer which estimates the score for that user-item pair. For instance, the feature *rev-desc* is the dot product of the average vector of the user's embedded reviews and the vector of the item's embedded description: $avg(r_u) \cdot d_i$.

3.3 Semantic Similarity

In our experiments mixing unstructured text with user-item interaction graphs, we need to compute the similarity between the textual representations of items and users. We used the Sentence-Transformers [14] framework of SOTA sentence and text embeddings to achieve that. Specifically, we use the "all-MiniLM-L6-v2"[4] model trained for semantic search and clustering tasks. Although larger models could further improve the quality, optimizing the specific language model is out of the scope of our work.

4 Data

Previous research has made use of the well-known Amazon Reviews data from 2014 [8] in the Books domain (herein "Books'14"), and we used that data, among others, to validate our code. However, the Books'14 data does not include textual item metadata like descriptions. Therefore, we use a newer version of that data for our experiments, Amazon Reviews 2018 [13], also in the Books domain (herein, "Books'18"), which includes textual item descriptions. Table 3 lists the statistics of the data.

Books'18 has a different distribution than Books'14, so for consistency, we sub-sampled Books'18 to have a similar ratio of users to items as in Books'14 (herein "Sampled'18") and used that as the final dataset in Books domain. Despite those three datasets being very similar in nature and structure, the results we have obtained for Sampled'18 were four times better than results we obtained for Books'14 on 3/4 metrics.

We calculated several sparsity-related metrics on each dataset included in Table 3 to investigate this discrepancy. These metrics demonstrate that Books'18 is much sparser than Books'14. While sub-sampling Books'18 does bring the number of users, items, and total samples closer to Books'14, centrality measures remain smaller than those for Books'14. That seems counter-intuitive: higher edge sparsity should result in lower recommendation performance. Nonetheless, this observation shows that it is important not to compare results across different generations of review data or even across different subsets of a single generation of data.

Sparsity metrics and other statistics for data from the other domains we use in this work ("Toys and Games", "Movies", and "Electronics") are also shown in Table 3.

[4] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

Table 3. Data statistics (centralities are multiplied by 10^4)

Data	#users	#items	#samples	Sparsity	Degree centr.		Eigenvector centr.	
					mean	median	mean	median
Books'14	53k	92k	2.9M	99.938%	2.29	1.39	11.48	4.13
Books'18	174k	96k	4M	97.605%	0.86	0.45	5.12	0.89
Sampled'18	92k	58k	2.7M	99.949%	1.79	1.06	7.98	1.43
Movies	268k	78k	3.1M	99.961%	1.09	0.50	9.80	4.55
Toys	64k	32k	0.75M	99.963%	1.32	0.83	12.36	5.40
Electronics	139k	40k	2.1M	99.984%	0.43	0.23	4.45	1.62

5 Experiments

We first describe the baselines we use for comparison and then the results of our experiments adding unstructured text representations to those baselines. The results of all models described in this section are shown in Table 4.

5.1 Baselines

Collaborative Filtering Baseline. The first baseline does not use graphs and works as a sanity check to ensure that a basic CF model does not outperform our much more complex approach. We use the "implicit"[6] Python library to build several CF systems from the user-item interaction matrix. The results of the best CF system–BayesianPersonalizedRanking in all cases–are shown in Table 4, marked as "CF (BPR)".

Graph-Based Baselines. We experiment with LightGCN [9] and several derivatives of it as baselines. LightGCN uses 3 graph propagation layers, with a simple mean aggregation over neighbor nodes, normalized symmetrically by the degree of each node. The final node representation is a simple average over the three layers' outputs (the formulas are available in the original paper [9]).

Single Layer. In the original LightGCN paper, the best model for the Books'14 data uses only the outputs from the final (i.e., third) layer. However, in our experiments, it did not perform as well as the version which takes the average of all layers. The authors called this the "Single" variation. We put the results for it in Table 4.

Alternate Aggregators. Before selecting LightGCN as our base model, we evaluated several other Graph Convolutional Networks, all of which are available in the Python `torch_geometric` [5] package (GCN [10], GAT (v1 [18] and v2), GraphSAGE [7]), however, all of them performed worse than LightGCN. Results are shown in Table 4.

Dynamic Negative Sampling (DNS). The authors of the LightGCN paper have noted [9] that more advanced negative sampling techniques could improve LightGCN, and we decided to try one such sampling method to evaluate whether the improvements obtained using additional unstructured text would still appear when using improved sampling methods. Following [21], we rank 1000 random items for each user, pick the 40 lowest ranked items, then pick 5 random positives, and train on the Cartesian product of those 2 sets (200 samples per user).

Per §3.1, we use 1-to-1 sampling, and the activation function in the BPR loss is SELU. Results for all those models can be found in Table 4.

5.2 TextGCN

We leverage the semantic information in encoded item descriptions and reviews by combining it with the final node vectors from the LightGCN network. The formulas for all features are described in Table 2.

Table 4. All the Baseline and Experimental models that were run on Sampled'18 dataset. Names of TextGCN models reflect which layer is used on top of LightGCN for the final score prediction. 'Linear' is the TextGCN Baseline model with a linear top layer. Metrics averaged over 5 runs

Model	Recall	Precision	Hit rate	nDCG
CF (BPR)	0.1422	0.0391	0.4662	0.1029
Graph Baselines:				
SAGE	0.0963	0.0263	0.3479	0.0674
GAT	0.1366	0.0359	0.4452	0.0984
GATv2	0.1384	0.0364	0.4503	0.0993
GCN	0.1419	0.0374	0.4584	0.1029
"Single"	0.1162	0.0318	0.4081	0.0833
LightGCN	0.1690	0.0455	0.5210	0.1244
LightGCN w DNS	0.1813	0.0490	0.5467	0.1353
TextGCN:				
XGBoost	0.1539	0.0372	0.4736	0.1075
GBDT	0.1749	0.0453	0.5308	0.1304
Linear	0.1833	0.0460	0.5308	0.1350
Linear w DNS	**0.1923**	**0.0485**	**0.5481**	**0.1428**

We have experimented with combining different representations. For example, we can represent users by concatenating their LightGCN node vector with the user-averaged item description vector and represent items similarly computing $(\mathbf{e}_u||avg_u(d)) \cdot (\mathbf{e}_i||d_i)$ as a feature. However, we found these were highly correlated with other existing features and so degraded the performance. We have omitted these from the table.

Finally, we freeze the LightGCN embeddings when training this final regressor. We also experimented with back-propagating the error signal back through the LightGCN model (unfreezing the model), but we achieved better results with frozen graph and text embeddings. Therefore we show only experiments with frozen embeddings below.

All models are run for 1000 epochs or until they converge and do not show any improvement for 75 consecutive epochs. Evaluation is performed every 25 epochs. All the experiments are run on 5 random vertically-sampled folds of the data: each training fold has all the users and is of size 80% of the whole dataset, and results are averaged.

Prediction Layer

Linear Layer. In this version, we use a simple dense linear layer on top of the features shown in Table 2. Despite the simplicity, it already significantly outperforms the baseline. Furthermore, we can add complexity by introducing hidden layers and increasing their sizes. For example, we added a 16-node hidden layer, which improved the results by an additional $\approx 1\%$.

Gradient Boosted Decision Trees. We experimented with gradient-boosted decision trees (GBDT) and XGBoost [2] regressors, however, they also performed worse than LightGCN (results in Table 4).

6 Results and Discussion

Table 4 shows variants of TextGCN that we described in §5.2. Surprisingly, a simple linear model did best here, and XGBoost performed worse than LightGCN. It proved too easy to overfit the ranking model to the data, and it failed to generalize well. So we use Linear version of TextGCN to run all further experiments.

Our main results for all datasets are shown in Table 5. When we compare each model with text to the corresponding baseline without text, the model incorporating text features is superior to the baseline on every metric on all the datasets. TextGCN improves recall@20 by 13.19%, precision@20 by 12.12%, hit rate by 11.19%, and nDCG by 20.25% on average over LightGCN. Adding Dynamic Negative Sampling

Table 5. Main results for all 4 domains. All metrics @20. All results are statistically significant ($p \ll 0.001$)

	Sampled'18 (Books)				Toys			
Model	recall	precis	hit	nDCG	recall	precis	hit	nDCG
LightGCN	0.1700	0.0429	0.5044	0.1222	0.0988	0.0107	0.1787	0.0571
TextGCN	0.1833	0.0460	0.5308	0.1350	0.1160	0.0125	0.2064	0.0693
LightGCN w DNS	0.1822	0.0463	0.5290	0.1330	0.1114	0.0122	0.2035	0.0662
TextGCN w DNS	0.1923	0.0485	0.5481	0.1428	0.1268	0.0136	0.2258	0.0762
	Movies				Electronic			
	recall	precis	hit	nDCG	recall	precis	hit	nDCG
LightGCN	0.1575	0.0195	0.3074	0.0939	0.0543	0.0058	0.1087	0.0299
TextGCN	0.1723	0.0212	0.3321	0.1107	0.0640	0.0067	0.1261	0.0392
LightGCN w DNS	0.1789	0.0227	0.3475	0.1123	0.0703	0.0078	0.1432	0.0474
TextGCN w DNS	0.1895	0.0239	0.3644	0.1241	0.0750	0.0082	0.1513	0.0514

Table 6. Average relative improvement of TextGCN over the corresponding baseline across all datasets

Metric	w/o DNS	w DNS
Recall	+13.19%	+8.03%
Precision	+12.12%	+7.02%
Hit	+11.19%	+6.27%
nDCG	+20.25%	+10.39%

boosts the performance of both the LightGCN and TextGCN, however, the improvement gets smaller: recall@20 by 8.03%, precision@20 by 7.02%, hit rate by 6.27%, and nDCG by 10.39%. This supports the conclusion that the actual text of user reviews or descriptions contains useful information beyond what is available in the user-item graph itself.

Table 2 shows which features the models use, as well as the average weights in the prediction layer for each feature for all TextGCN models we have trained. Those weights can act as proxies for feature importances and we can draw conclusions from them. We notice that the highest weights are for the "description" feature, which calculates the similarity between the user's average item description vector and the candidate item vector. We speculate that this is because users are attracted to similar aspects across products. Such aspects (say "lightweight") might be implicit in a user-item interaction graph if enough users interacted with the *same* set of products that shared that feature. However, using the text descriptions to represent both users and items appears to be more effective and direct.

Two other textual features, using different textual representations from each user and item (*rev-desc* and *desc-rev*) have negative importance, which suggests that there is no direct useful link between the description given to the product by the seller and the reviews written by the users.

7 Conclusions

In this work, we have established that knowledge from unstructured text can be exploited to improve recommendations using Graph Neural Networks. This text captures information not present in the user-item interaction graph. By examining the computed feature importances of our ranking models, we identified that using the item description text to augment both user and item representations had the strongest positive influence on recommendation metrics. Furthermore, we have shown that we can efficiently augment lightweight graph embeddings with this text and substantially improve recommendation performance without complex models to combine the graph and textual representations.

References

1. Chen, B., Guo, W., Tang, R., Xin, X., Ding, Y., He, X., Wang, D.: TGCN: Tag Graph Convolutional Network for Tag-Aware Recommendation. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. p. 155–164. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3411927

2. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, ACM, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939785
3. Demartini, G., Zuccon, G., Culpepper, J.S., Huang, Z., Tong, H., Tu, K., Cui, P., Wang, D., Zhang, Z., Zhou, J., Qi, Y., Zhu, W.: Conditional Graph Attention Networks for Distilling and Refining Knowledge Graphs in Recommendation. Proceedings of the 30th ACM International Conference on Information & Knowledge Management pp. 1834–1843 (2021). https://doi.org/10.1145/3459637.3482331
4. Deng, Y.: Recommender Systems Based on Graph Embedding Techniques: A Review. IEEE Access **10**, 51587–51633 (2022). https://doi.org/10.1109/access.2022.3174197
5. Fey, M., Lenssen, J.E.: Fast graph representation learning with PyTorch Geometric. In: ICLR Workshop on Representation Learning on Graphs and Manifolds (2019)
6. Frederickson, B.: Fast python collaborative filtering for implicit datasets (2017), https://github.com/benfred/implicit
7. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf
8. He, R., McAuley, J.: Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: proceedings of the 25th international conference on world wide web. pp. 507–517 (2016). https://doi.org/10.1145/2872427.2883037
9. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: 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. pp. 639–648 (2020). https://doi.org/10.1145/3397271.3401063
10. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016). https://doi.org/10.48550/arXiv.1609.02907
11. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017), https://proceedings.neurips.cc/paper/2017/file/5d44ee6f2c3f71b73125876103c8f6c4-Paper.pdf
12. Mei, D., Huang, N., Li, X.: Light Graph Convolutional Collaborative Filtering With Multi-Aspect Information. IEEE Access **9**, 34433–34441 (2021). https://doi.org/10.1109/access.2021.3061915
13. Ni, J., Li, J., McAuley, J.: Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 188–197 (2019). https://doi.org/10.18653/v1/D19-1018
14. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics (2019), https://arxiv.org/abs/1908.10084
15. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. p. 452–461. UAI '09, AUAI Press, Arlington, Virginia, USA (2009). https://doi.org/10.48550/arXiv.1205.2618
16. Sun, R., Cao, X., Zhao, Y., Wan, J., Zhou, K., Zhang, F., Wang, Z., Zheng, K.: Multi-modal Knowledge Graphs for Recommender Systems. Proceedings of the 29th ACM International Conference on Information & Knowledge Management pp. 1405–1414 (2020). https://doi.org/10.1145/3340531.3411947
17. Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., Karypis, G., Wang, X., He, X., Cao, Y., Liu, M., Chua, T.S.: KGAT: Knowledge Graph Attention Network for Recommendation. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining pp. 950–958 (2019). https://doi.org/10.1145/3292500.3330989
18. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017). https://doi.org/10.48550/arXiv.1710.10903
19. Wang, S., Hu, L., Wang, Y., He, X., Sheng, Q.Z., Orgun, M.A., Cao, L., Ricci, F., Yu, P.S.: Graph learning based recommender systems: a review. In: Zhou, Z.H. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021. pp. 4644–4652. IJCAI International Joint Conference on Artificial Intelligence, International Joint Conferences on Artificial Intelligence (2021). https://doi.org/10.24963/ijcai.2021/630, 30th International Joint Conference on Artificial Intelligence, IJCAI 2021 ; Conference date: 19-08-2021 Through 27-08-2021
20. Wu, S., Sun, F., Zhang, W., Xie, X., Cui, B.: Graph neural networks in recommender systems: A survey. ACM Comput. Surv. (2022). https://doi.org/10.1145/3535101

21. Zhang, W., Chen, T., Wang, J., Yu, Y.: Optimizing top-n collaborative filtering via dynamic negative item sampling. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 785–788. SIGIR '13, Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2484028.2484126