A Comparison of Different Embedding Methods on Session Based Recommendation with Graph Neural Networks

Mustafa Aker , Çağrı Emre Yıldız, Yusuf Yaslan

akermu@itu.edu.tr, yildizcag@itu.edu.tr, yyaslan@itu.edu.tr
Istanbul Technical University, Turkey

Keywords: session-based recommendation, graph embedding, gated graph neural networks

Abstract

Predicting users' next behavior based on their previous actions is one of the most valuable but also difficult task in the e-commerce and e-marketing fields. Recommendation systems that build upon session-based data try to bring a solution to that desire. The ultimate goal of this type of recommendation system is trying to make the best predictions about the succeeding item. Sequential order of the items within a session is also kept in mind in such systems. Recently proposed SR-GNN (Session Based Recommendation with Graph Neural Networks) has benefited from graph theory and proven its adequacy about being the state-of-art session-based recommendation model. Furthermore, there are some parts exist that can improve the overall performance. The current model uses primitive embedding type which is the simplest way of representing the items, attributes, and their relationships between each other. This study brings the SR-GNN recommendation model with different types of graph embedding techniques which are widely used in a variety of research areas. Aim of this research is investigating the the effect of the embedding types to the SR-GNN. The proposed variety of embedding techniques that applied to SR-GNN show similar but slightly worse results compared to the original SR-GNN embedding. The experimental results obtained on two real datasets show that the performance of the SR-GNN model is not affected by the embedding models and the power of the model comes from the gated graph neural network model.

1 Introduction

Recommendation task based on previous interests and interactions of users is quite important for many e-commerce companies. Analyzing and processing the substantial data which piles up cumulatively within the ever-growing internet has become one of the main goals of the companies and researchers working in these areas. The more accurate the recommendations are, the more users will be drawn from other ecosystems that are similar. Eventually, this situation might start a chain reaction by creating more precise recommendation systems, since the growing amount of data within the system will better sustain the offered model [1].

Prediction task of succeeding items based on interaction/purchase history data can be handled as two different aspects of whether the current session is related to a registered user or not. If historical data is associated with profiles, the problem becomes easier to solve[2]. Collaborative filtering techniques, which take account of both item relations and user relations, perform well in making predictions powered by historical data. However, for datasets that do not have such profiling because of privacy regulations, lack of data, or cold start, it requires another type of approach to make predictions relying on such session-based historical data.

SR-GNN is one of the state-of-art models in the field of session-based recommendation systems. According to the SR-GNN model, items are represented as nodes, and sessions are represented as graphs. In this manner, incoming and outgoing edges of the graph represent the relation between items as well as the embeddings among the nodes. In the original paper [3], embedding of these nodes is calculated as the normalized weights of nodes in a session. Incoming and outgoing edges represented as two different matrices. For simplicity, outgoing edge matrix is calculated by getting transpose of the incoming edge matrix.

From the perspective of the embedding scheme, it is possible to represent the node by highlighting different attributes and very special characteristics by using different embedding techniques. Additionally, it is possible to reduce and fix the size of the data required for representing the specific node in a better way. The first studies in this area have been revealed in the early 2000s [4]. Compared to the earlier methods, recent methods are better in terms of accuracy and performance thanks to deep learning approaches with the growing enormous-sized data available to researchers. As Cai et al. [5] mention, graph embedding operation can map the graph and node structure into lower dimensional space while ensuring the key features of the graph remain the same. In favor of these concentrated data, node-item labeling, node-item recommendation, and note-item prediction tasks are handled in a more efficient way.

Table 1: Summary of notation

Summary of notation	
V	Set of vertices in the graph
E	Set of edges in the graph
G	Graph of (V, E)
d	Number of dimensions
A	Embedding of graph G in d dimension $\|V\| \times d$
G_i	Embedding of v_i in d dimension $(1 \times d)$
W	Adjacency matrix of G
S	Sampling strategy (Either Breadth-First or Depth-First Sampling)
$N_s(A_i)$	Neighbors of v_i based on the sampling strategy S
$\phi(G)$	Objective function of given embedding

Graph embedding methods can be split into 3 different groups regarding the operation schemes of these methods. These groups can be listed as graph adjacency matrix factorization-based methods, random walk-based methods, and deep learning-based methods.

1.1 Embedding Methods

1.1.1 Graph adjacency matrix factorization-based methods

Matrix factorization was used for rerepresenting the session items and their relations with reduced dimensionality with concentrated information. This operation is required for more complex matrix operations such as matrix decomposition by allowing reduced operation time. First studies built upon matrix factorization, carried out in the early 2000s by Tenenbaum [6]. One of the most well-known and widely used embedding type considered singular value decomposition (SVD) which is built upon eigenvalue decomposition. The main goal of this procedure is to represent the matrix elements with lower dimensions [7]. For an embedding with m nodes: (Table 1)

$$A_{m\times d} = \begin{bmatrix} \uparrow & \dots & \uparrow \\ u_1 & \dots & u_k \\ \downarrow & \dots & \downarrow \end{bmatrix}_{m\times k} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix}_{k\times k} \begin{bmatrix} \leftarrow v_1^T \rightarrow \\ \vdots \\ \leftarrow v_k^T \rightarrow \end{bmatrix}_{k\times d} \quad (1)$$

$$A_{m\times d} = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \dots + \sigma_k u_k v_k^T$$

A novel approach, Isomap, that detects nonlinear degrees of freedom unlike principal component analysis and multidimensional scaling became able to operate on human handwriting and face emotion datasets [5]. Laplacian eigenmaps method considers each node pair closer if weight between those nodes is higher and tries to minimize the following objective function [4]:

$$\Phi(G) = \frac{\sum_{i,j} |G_i - G_j|^2 W_{ij}}{2} \quad (2)$$

Graph factorization proposes a different approach for factorizing the graph by trying to split the overall graph into smaller ones while considering minimizing the number of neighboring nodes instead of edges [8]. HOPE [9] proposed more recently compared to other methods. The main goal is seeking out for an indirect relationship between each node.

1.1.2 Random walk-based methods

Random walk-based methods mostly originated from the word2vec [10] algorithm, which treats words as nodes placed in a directed graph and operates within the skip-gram model to obtain their embedding representations. As an extension of the word2vec algorithm, the DeepWalk [11] algorithm picks a random node from the graph and traverses randomly until fixed steps are completed or the adaptive length is reached. Within these walks, the latent relation between each node pair is calculated and node embeddings are determined. Node2vec [12] is another method that utilizes a random walk scheme in favor of finding the best representation of nodes stated in the graph. Considering this method, the main objectives are representing each node in low dimensional feature space and trying to maximize the deciding the presence probability of the next neighboring node by maximizing the following objective function:

$$\sum_i log Pr(N_s(A_i)|\boldsymbol{\phi}(A)) \quad (3)$$

It combines bread first search node sampling and depth-first search node sampling to obtain vertex sequences. Struc2vec [13] offers a similar approach within node2vec and DeepWalk since it includes the DeepWalk phase among the steps of this procedure. Correspondingly, first, it tries to determine the hidden relationships among nodes via building a multilayered network, and then by using this generated network, the constructional similarity of the nodes is calculated. Lastly, the DeepWalk algorithm is executed for the remaining steps of the offered procedure.

1.1.3 Deep learning-based methods

Structural Deep Network Embedding (SDNE) uses deep autoencoders to obtain latent (unsupervised) and first-order (supervised) relationships between each node pair among the graph structure. It combines autoencoder and Laplacian eigenmaps schemes together to extract substantial node features from the graph.

GCN is a semisupervised learning method that operates on graph-structured networks. One of the advantages of the model is, it can be executed with linear time complexity. In essence, GCN extracts the hidden layer features and finds crucial features about both localized graph structure and nodes of the network.

Large-scale Information Network Embedding (LINE) [14] tries to solve the problem of representing the information in lower-dimensional space. The model works with all types of graphs which are directed, undirected, and weighted. The method proposes an efficient way

compared to similar methods while computing first-order and second-order relationships of all nodes.

Session-based recommendation systems became widely popular in recent years. Making more precise predictions by considering both short-term history data and long-term history data becomes available by the model offered by [15] which is called STAMP. Later, the SR-GNN model, which is proposed by [3] built upon the STAMP model and improved the performance of its predecessor. In the model SR-GNN, embeddings are calculated as normalized weights between each node pair within a session.

Considering a session-based dataset, understanding and extracting the hidden relationship between each item within the overall item graph -also called item network- happened to be a vital task to get better predictions via session-based recommendation models. Different embedding schemes were adopted for different tasks over time, however, none of them tried on any of the session-based datasets.

In this study, our aim is to investigate the node embeddings over session-based recommendation system models. To achieve this, we have implemented different node embedding algorithms and calculated the related node embedding representation of each item provided by $DIGINETICA$[1] and $YOOCHOOSE$[2] datasets. Within these calculations, the SR-GNN model is fed by precalculated node representation data instead of the default one, which is normalized weights among nodes. In the scope of In this research, singular value decomposition and Laplacian eigenmaps embeddings are implemented as matrix factorization based methods, DeepWalk, and node2vec embeddings are implemented as random walk-based methods, LINE embedding implemented as a deep learning-based method, additionally, word2vec embedding has been implemented as an ancestor method of DeepWalk and node2vec algorithms.

Each embedding calculation has been performed upon generating the overall network provided by datasets. To create a global map structure, each new node within the session was added cumulatively to the network. Edges that have been introduced to the global map are attached to the global map as their weights are assigned as 1. For those edges that reappear while traversing the dataset, their weights increased according to their reappearance instead of re-adding them to the global map (their weights increased by 1 for each re-appearance case). Following this procedure, a global network is created for the operation of calculating node embeddings considering their weights over the graph structure.

Native SR-GNN method consists of 2 main subprocesses; the first one is the graph embedding phase and the other one is the gated graph neural network phase. Generated embedding values are given as input values for the first step which is the embedding phase and the predictions on the graph neural networks phase performed accordingly.

Our aim is to draw attention to the effect of the embedding mechanism on session-based recommendation models. For evaluating the performance of the offered models, the same dataset that SR- GNN method reached, its maturity has been used to make better and more precise interpretations of the proposed model. Graph embedding phase of the SR-GNN method is enriched by altering the default embedding method with 6 different embedding methods (SVD, Laplacian eigenmaps, word2vec, DeepWalk, node2vec, and LINE).

2 Experiments

2.1 Datasets and Data preparation

We have built and conducted several experiments within the context of this study. For each case SR- GNN method is executed multiple times and the average of Recall@20 and MRR@20 values have been used to neglect the effect of the sampling procedure of the SR-GNN method (Figure 1). We published an open-source *library* [3] that includes implementations of each graph embedding technique with the SR-GNN model.

To be able to demonstrate embeddings' effect on the SR-GNN model, we have used the same dataset which is used by Wu et al. [3] in the SR-GNN model's paper . YOOCHOOSE dataset is originally published as part of the RecSys Challenge 2015 and contains six months of user click data obtained from e-commerce websites. The second dataset, DIGINETICA, was published as part of the CIKM Cup 2016 and the transactional part of this dataset is used for evaluation. For data cleaning, the steps proposed by Li et al. [16] and Liu et al. [15] are followed for consistency. From both datasets, sessions with a single item are removed and after that, all items appearing less than 5 sessions are also removed. That leaves 7,981,580 sessions and 37,483 items in YOOCHOOSE, and 204,771 sessions and 43097 items in DIGINETICA dataset. For YOOCHOOSE dataset, only the most recent 1/64 fraction is used. Wu et al. [3], Li et al.[16], and Liu et al. [15] also use the most recent 1/4 fraction, however, this setup exceeded our hardware memory limit.

To observe the effect of the embedding type on the SR-GNN model, first, we need to create an embedding vector of each item given within raw datasets. To achieve this, it is required to create a whole graph structure by considering the sessions given in the raw dataset. Each session in the raw dataset can be treated as subgraphs of the whole graph. By using these sessions, anoverall graph has been built by constructing the graph piece by piece by merging them together. If item B is observed in a session right after item A, a directed edge from A to B is created between these nodes to show the relationship between these nodes. The overall graph has been built

[1] http://cikm2016.cs.iupui.edu/cikm-cup

[2] https://recsys.acm.org/recsys15/

[3] https://github.com/yildizcag/Embedding-SRGNN

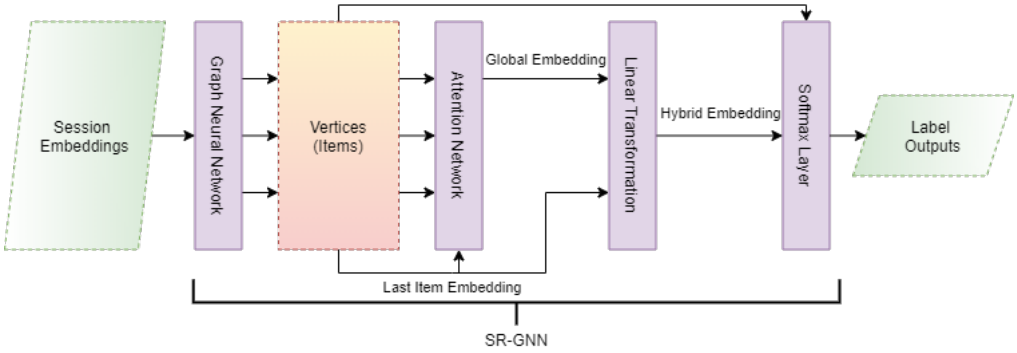


Figure 1: SR-GNN Method proposed by Wu et al. For each test, different graph embedding method is used to generate session embeddings. Result of the embedding models are then fed to SR-GNN method to generate predictions for next item in the sessions.

using this procedure. After this building phase will have accomplished, the output graph has become ready for evaluating node embeddings. In the following step, six different embedding types introduced within the context of this research were created. Then, the obtained data stored to give as input parameters to the SR-GNN neural network phase.

2.2 Evaluation Metrics

For evaluating and comparing the performances of the proposed models, the following two widely used evaluation metrics have been picked up:

Recall@20: Represents the ratio of correctly proposed items that occur inside the top-20, among the ordered item list.

$$Recall@K = \frac{n_{hit}}{N} \tag{4}$$

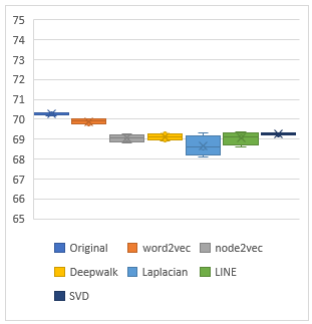
Mean Reciprocal Rank @20 (MRR@20): Average of the reciprocal ranks of the items that have been correctly recommended. If it is above 20, the value of MRR@20 is assigned as 0.

$$MRR@K = \frac{\sum_{t \epsilon G} \frac{1}{Rank(t))}}{N} \tag{5}$$

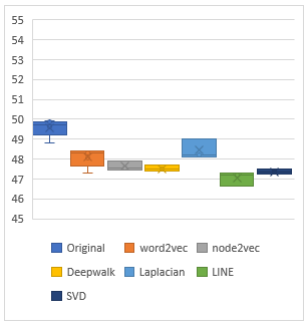
2.3 Baselines

For evaluating the performance of the proposed model, reference values need to be calculated. For this purpose, the default SR-GNN method provided by Wu et. al. [3] has been executed. For the YOOCHOOSE 1/64 dataset, an average Recall@20 score of 70.27 (Figure 2(a)) and for the DIGINETICA dataset, an average Recall@20 score of 49.57 (Figure 2(b)) has been observed.

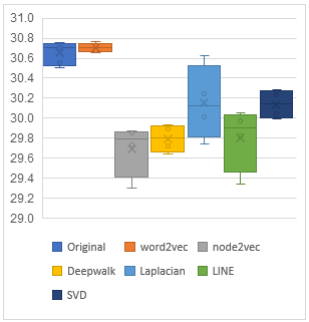
Any of the proposed embedding types could not manage to reach the performance of the original SR-GNN method, while YOOCHOOSE 1/64 dataset word2vec implementation MRR@20 score (Figure 2(c)) has a lower variance than the original method and is on par with the original method. However, the Recall@20 score is slightly worse compared to the original embedding. For the DIGINETICA dataset, Laplacian eigenmaps implementation performed second best with 48.45 Recall@20 success rate (Figure 2(d)). Other types of embedding implementation executions also performed worse on YOOCHOOSE and DIGINETICA datasets. However, their performances listed only 1.0-2.0 points in terms of Recall@20 success rate.



(a) YOOCHOOSE 1/64 dataset Recall@20 Results



(b) DIGINETICA dataset Recall@20 Results



(c) YOOCHOOSE 1/64 dataset MRR@20 Results

(d) DIGINETICA dataset MRR@20 Results

Figure 2: Recall@20 and MRR@20 scores of each embedding methods combined with SR-GNN (with yoochose 1/64 and DIGINETICA dataset)

2.4 Parameters

SR-GNN method has two different major parts which are the neural network phase and graph embedding phase and that can be executed separately. For each part, there are several hyperparameters that can be adjusted. For the neural network phase, the learning rate and hidden layer size parameters play a vital role, while the embedding dimension becomes crucial for the graph embedding phase. Within these experiments, the learning rate is adjusted as 0.001, and the hidden layer size is set to 100, and starting from 4^{th} epoch, the learning rate decays by 0.1. Furthermore, the embedding dimension is set to 128 due to performance issues.

3 Conclusion

3.1 Conclusion

In this study, we have searched for the effect of graph embedding types over session-based recommendation models -specifically SR-GNN- and made a broad comparison among these methods. We have used 2 different session-based datasets which are provided by YOOCHOOSE and DIGINETICA in the scope of ACM RecSys Challenge 2015. Six different embedding types which are SVD, Laplacian eigenmaps, DeepWalk, node2vec, word2vec, and LINE applied on graphs derived from these two datasets and these embedding calculations are given to the neural network phase of the SR-GNN method. For YOOCHOOSE dataset, among these methods including SR-GNNs default one, the default embedding method outperformed the most successful embedding method among these six, which is word2vec by 0.5 Recall@20 rate (Table 2). On the other hand, word2vec embedding average MRR@20 score is slightly higher (0.05) and it has a more consistent distribution. For this dataset; we observe that the next item appearance is less frequent in the top 20 items, but its order in the top 20 is better than the original embedding (Figure 2(c)). For the DIGINETICA dataset, the Laplacian eigenmaps embedding method outperformed six introduced embedding methods, yet, it still falls behind the default embedding method by 1% in terms of Recall@20 rate (Table 3).

Table 2: YOOCHOOSE 1/64 dataset results

Model Name	Recall@20	MRR@20
Original SR-GNN	**70.27**	30.65
SVD	69.27	30.14
LINE	69.05	29.80
word2vec	69.89	**30.68**
node2vec	69.06	29.69
DeepWalk	69.11	29.80
Laplacian Eigenmaps	68.66	30.16

Internal structure of the dataset plays a key role while making predictions based on history. It is believed that word2vec performed better on YOOCHOOSE data might be related to the difference of average session lengths of the datasets. Furthermore, the size of the training set and the average relation count between nodes affect the outcome. Since word2vec method makes predictions based on a sequence of items that are observed previously, it is not a surprising outcome that word2vec operates better on a dataset that has longer individual sessions. When it is investigated deeply, it is observed that DIGINETICA dataset has more unique nodes than YOOCHOOSE dataset, and considering DIGINETICA dataset, the edges between nodes are more spread out while YOOCHOOSE dataset has a more compact and dense structure (edges between nodes converge on specific nodes while creating the nodes having higher centrality values [17]). Accordingly, DIGINETICA dataset's graph to have a more balanced structure than YOOCHOOSE dataset's which leads to the Laplacian eigenmaps method operating better, since this method adjusts embeddings of two nodes similar if the weight of the edge between two nodes is higher [18]. The graph that has this attribute spread in a wider range becomes a more suitable operation field for the Laplacian eigenmaps method. Considering the SR-GNN method performing on a session-based dataset, representing the relationships among the nodes in a more complex way is not the best way, since it requires more computation power and does not improve the performance as expected.

Table 3: DIGINETICA dataset results

Model Name	Recall@20	MRR@20
Original SR-GNN	**49.59**	**16.45**
SVD	47.37	15.28
LINE	47.05	14.96
word2vec	48.09	15.40
node2vec	47.63	15.10
DeepWalk	47.52	15.23
Laplacian Eigenmaps	48.45	16.13

3.2 Future Directions

In this research, we have studied with 2 different session-based datasets which are YOOCHOOSE and DIGINETICA. In order to get broader inferences, these experiments can be rerun on different session-based datasets. From the SR-GNNs neural network phase's perspective, learning rate and hidden layer size hyperparameters which are picked as 0.001 and 100 consecutively, can be adjusted according to the embedding type. From the embedding perspective, the dimension of embedding vectors is chosen as 128. Depending on the type of the research field, this parameter can be adjusted correspondingly. In this way, case-specific embedding models can be built.

References

[1] Junghwan Cho, Kyewook Lee, Ellie Shin, Garry Choy, and Synho Do. How much data is needed

to train a medical image deep learning system to achieve necessary high accuracy? *arXiv:1511.06348 [cs]*, January 2016. arXiv: 1511.06348.

[2] David Ralph, Yunjia Li, Gary Wills, and Nicolas G. Green. Recommendations from cold starts in big data. *Computing*, 102(6):1323–1344, June 2020.

[3] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based Recommendation with Graph Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:346–353, July 2019. arXiv: 1811.00855.

[4] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 585–591, Cambridge, MA, USA, January 2001. MIT Press.

[5] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications. *arXiv:1709.07604 [cs]*, February 2018. arXiv: 1709.07604.

[6] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, December 2000. Publisher: American Association for the Advancement of Science Section: Report.

[7] Charles F. Van Loan. Generalizing the singular value decomposition. *SIAM Journal on Numerical Analysis*, 13(1):76–83, 1976.

[8] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, WWW '13, pages 37–48, New York, NY, USA, May 2013. Association for Computing Machinery.

[9] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric Transitivity Preserving Graph Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1105–1114, New York, NY, USA, August 2016. Association for Computing Machinery.

[10] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Red Hook, NY, USA, December 2013. Curran Associates Inc.

[11] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '14, pages 701–710, New York, NY, USA, August 2014. Association for Computing Machinery.

[12] Aditya Grover and Jure Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 855–864, New York, NY, USA, August 2016. Association for Computing Machinery.

[13] Leonardo F. R. Ribeiro, Pedro H. P. Savarese, and Daniel R. Figueiredo. struc2vec: Learning Node Representations from Structural Identity. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394, August 2017. arXiv: 1704.03165.

[14] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 1067–1077, Republic and Canton of Geneva, CHE, May 2015. International World Wide Web Conferences Steering Committee.

[15] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1831–1839, New York, NY, USA, July 2018. Association for Computing Machinery.

[16] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, and Jun Ma. Neural attentive session-based recommendation. *CoRR*, abs/1711.04725, 2017.

[17] U Kang, Spiros Papadimitriou, Jimeng Sun, and Hanghang Tong. Centralities in Large Networks: Algorithms and Observations. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, Proceedings, pages 119–130. Society for Industrial and Applied Mathematics, April 2011.

[18] Palash Goyal and Emilio Ferrara. Graph Embedding Techniques, Applications, and Performance: A Survey. *Knowledge-Based Systems*, 151:78–94, July 2018. arXiv: 1705.02801.