Compositional Network Embedding for Link Prediction

Tianshu Lyu[1], Fei Sun[2], Peng Jiang[2], Wenwu Ou[2], Yan Zhang[3*]
[1,3] Department of Machine Intelligence, Peking University [2] Alibaba Group, Beijing, China,
[1] lyutianshu@pku.edu.cn [2] {ofey.sf, jiangpeng.jp, santong.oww}@alibaba-inc.com [3] zhy@cis.pku.edu.cn

ABSTRACT

Almost all the existing network embedding methods learn to map the node IDs to their corresponding node embeddings. This design principle, however, hinders the existing methods from being applied in real cases. Node ID is not generalizable and, thus, the existing methods have to pay great effort in cold-start problem. The heterogeneous network usually requires extra work to encode node types, as node type is not able to be identified by node ID. Node ID carries rare information, resulting in the criticism that the existing methods are not robust to noise. To address this issue, we introduce *Compositional Network Embedding*, a general *inductive* network representation learning framework that generates node embeddings by combining node features based on the "*principle of compositionally*". Instead of directly optimizing an embedding lookup based on arbitrary node IDs, we learn a composition function that infers node embeddings by combining the corresponding node attribute embeddings through a graph-based loss. For evaluation, we conduct the experiments on link prediction under three different settings. The results verified the effectiveness and generalization ability of compositional network embeddings, especially on unseen nodes.

CCS CONCEPTS

• **Computing methodologies → Learning latent representations**; *Knowledge representation and reasoning*.

KEYWORDS

Network Embedding Principle of Compositionally

ACM Reference Format:
Tianshu Lyu[1], Fei Sun[2], Peng Jiang[2], Wenwu Ou[2], Yan Zhang[3]. 2019. Compositional Network Embedding for Link Prediction. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347023

1 INTRODUCTION

Graph structure plays a key role in information retrieval, *e.g.*, OSN and e-commerce. Recently, there has been a surge of work to represent nodes as low-dimensional dense vectors, called *Network Embedding*, in order to analyze the graph-structured data.

Background and related work. As the downstream machine learning tasks usually involve the predictions relevant to the nodes, the existing methods usually map node IDs into a latent space by a lookup table. However, we argue that many challenges the existing methods struggle with may stem from this design philosophy.

First of all, node ID is not generalizable. Therefore, most of the existing methods cannot infer the embeddings of the unseen nodes that do not appear in the training phase (*e.g.*, cold-start items in recommendation system). Dynamic network embeddings generate unseen node embeddings in incremental [5] or inductive way [8]. However, both of these two kinds of methods require the unseen nodes have connections to the observed network, which is not always practical or true. The second challenge is particularly for the heterogeneous network (*e.g.*, a *customer-product-brand* network). Node IDs do not carry node types inherently. Consequently, it is inappropriate for the embedding methods to learn the mapping function between node IDs and types. A conventional way for heterogeneous network embedding is to project different types of nodes to different latent spaces. This kind of methods [1, 4, 16] have to put great effort in aligning embeddings of different spaces. The last one is the balance between network topology sensitivity and robustness. In the real cases, the graph is always with noise (*e.g.*, a Real Madrid fan clicked a FC Barcelona team jersey is a spurious edge). However, most of the existing methods represent edge by a pair of node IDs, giving little hint about the edge itself. Therefore, robustness is hardly discussed in the network embedding fields [3]. The methods based on random-walk sampling [7] are not able to distinguish noises. In addition, the node embeddings relying on aggregating neighborhood attributes [8, 17] might be seriously affected by the wrong edges [3].

In this paper, we draw inspiration from the "*principle of compositionally*" [6], an influential theory in NLP area, in order to inherently tackle the above problems. It states the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them. Instead of learning a distinct embedding vector for each node ID, the proposed model, CNE, trains a composition function that learns to derive the node embedding by combining the corresponding node attribute embeddings (rather than the neighborhood embeddings as GNNs [8, 17]). The attribute embeddings are shared across all nodes in the network. Specifically, CNE learns the attribute embeddings and the composition function by considering the node proximity indicating from the graph, which is flexibly captured by random walks. The node embeddings, as the intermediate results of the framework, are encouraged to be more similar by the unsupervised graph-based loss function when the nodes are in close proximity.

CNE generate embeddings for the unseen nodes by applying the learned composition function to their attributes. Different types of nodes correspond to different node attributes and composition methods. The type differences are naturally captured by CNE. CNE is also not sensitive to the edges that related to two nodes with rarely co-occurred attributes.

[*]Tianshu worked as an intern at Alibaba. Fei Sun is the corresponding author.

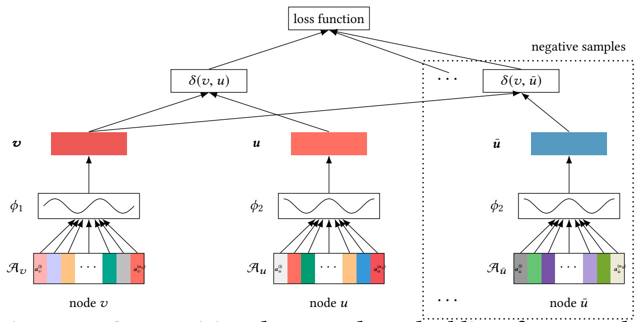

Figure 1: Compositional network embedding framework. For each node v and its neighbor u, we randomly sample K negative node $\bar{u}$. The objective of our framework is to distinguish the positive node u from the negative node $\bar{u}$ using the embeddings derived from their internal attributes.

We evaluate our models on link prediction task under different settings, including the prediction of missing edges, edges of unseen nodes, and multi-node-type edges. The results demonstrate that CNE possesses higher expressive capability, stronger generalization capacity, and more flexibility on heterogeneous network.

2 COMPOSITIONAL NETWORK EMBEDDING

In this section, we will present our idea of compositional network embedding (CNE) framework. The core idea behind our approach is that we learn how to derive the node embeddings from the embeddings of the features they carried. As Fig. 1 illustrated, CNE framework contains two key parts: (i) composition function that generates node embeddings from their internal attributes; (ii) unsupervised graph-based loss function for learning the parameters of composition function and attribute embeddings.

2.1 Embedding Composition

As Fig. 1 illustrated, the embedding generation process for a node v_i is very straightforward, applying the composition function ϕ to the node v_i's features embeddings $\mathcal{A}_i$=[$\boldsymbol{a}_i^{(1)}, \dots, \boldsymbol{a}_i^{(n_i)}$] as:

$$\boldsymbol{v}_i = \phi(\mathcal{A}_i) = \phi\left(\boldsymbol{a}_i^{(1)}, \dots, \boldsymbol{a}_i^{(n_i)}\right)$$

where, $\boldsymbol{a}_i^{(j)} \in \mathbb{R}^d$ is the embedding of node v_i's attributes $a_i^{(j)}$, d is the dimensionality of $\boldsymbol{a}_i^{(j)}$, n_i is the length of v_i's features. Here, we model the composition function ϕ as a neural network. In this context, the composition function is equivalent to an encoder. Thus, we also use *encoder* to refer to the *composition function* later. It is worth emphasizing that feature a and its embedding $\boldsymbol{a}$ are shared across all nodes in the network. In this way, after the model has been trained and the parameters are fixed, we can generate the embedding for an unseen node by feeding these shared feature embeddings through the composition function ϕ.

2.2 Composition Function (Encoder)

CNE is a quite general framework for network embedding. Here, the "generality" is reflected in the choices for feature design and composition function design. For example, we can use text (*e.g.*, product title) as the feature for the networks like product network, or image as the feature for Flickr image relationships. According to the different features, we can design the corresponding encoders, from simply concatenate, mean, and sum operator to complex models like GRU and CNN.

In this paper, without loss of generality, we mainly focus on the network with text features to introduce the framework for the sake of simplicity. Here, we use an RNN encoder with GRU as composition function ϕ to encode text features. The last hidden state are used as the representations of the node features $\mathcal{A}_i$, which we also treat as the node embedding of node v_i. While we use GRU here, any other types of encoder can be used so long as we can back-propagate through it.

2.3 Learning The Parameters of CNE

In order to learn useful and predictive embeddings in an unsupervised way, we apply a graph-based loss to the output node embeddings, and tune the parameters via stochastic gradient descent. The graph-based loss function encourages nearby nodes to have similar embeddings, while enforcing the embeddings of disparate nodes are highly dissimilar. Specifically, CNE is trained end-to-end in a *siamese* framework, as illustrated in Fig. 1. Formally, for node v and its neighbor $u \in \mathcal{N}(v)$ ($\mathcal{N}(v)$ is the neighbor set for v), we define the max-margin (*i.e.*, hinge) loss function [2, 12] as:

$$\mathcal{L}(v,u) = \sum_{k=1}^{K} \max\Big(0, m - \delta(v,u) + \delta(v,\bar{u}_k)\Big) \qquad (1)$$

where $\bar{u}_k$ is a negative sample randomly sampled from the whole node set $\mathcal{V}$, K is the number of negative samples; m is the margin between the the positive node pairs and the negative node pairs, usually set as 1; δ is the score function to measure the similarity between two nodes, we define as:

$$\delta(v,u) = \cos(\boldsymbol{v},\boldsymbol{u}) = \cos\Big(\phi_1(\mathcal{A}_v), \phi_2(\mathcal{A}_u)\Big)$$

where, ϕ_1 and ϕ_2 are encoders for node v and u respectively.

Instead of getting the vector $\boldsymbol{v}$ (and $\boldsymbol{u}$) via an embedding look-up, CNE generates them by combining the feature embedding carried with node v (and u). The compositional embedding makes it possible to interact between network topology and feature similarity.

Intuitively, the goal of the max-margin objective is to rank the correct neighbor u of node v higher than any other random node $\bar{u}$ with a margin m. Importantly, other loss functions (*e.g.*, likelihood objective in DeepWalk) are also valid in our framework. Here, we choose hinge loss for its better performance in our experiments. It is worth noting that CNE can also be trained in a supervised manner, by simply replacing (or augmenting) the unsupervised loss (Eq. 1) with a task-specific objective.

Neighborhood definition. As mentioned in the loss function, the neighborhood definition is a key part in the training stage. In this paper, we define the neighborhood based on random walks as the same in DeepWalk for its effectiveness and efficiency. Specifically, we first start truncated random walks with length l at each node. After that, the *neighbors* of node v can be defined as the set of nodes within a window size w in each random walk sequence.

2.4 CNE for Various Kinds of Networks

As we presented above, CNE is a general framework for network representation learning. In CNE, node embeddings are mainly determined by the nodes' attributes and the encoders (*i.e.*, models and parameters). Here, we will discuss how to apply CNE to various kinds of networks by adjusting the setting of node attributes and encoders accordingly.

Directed network. In this case, edge direction is supposed to be preserved. For this purpose, we learn two encodings, in-degree encoder ϕ_1 and out-degree encoder ϕ_2, per node in order to rightly predict the edge directionality.

Heterogeneous network with multiple node types. In this case, different types of nodes usually carry with features from different fields. Item attributes, for instance, seem to be completely different from the user demographic characteristics in recommendation field. CNE is naturally capable of mapping different types of nodes to the same low-dimension space. We can simply employ different models as encoders according to the types of nodes.

3 EXPERIMENTS

In this section, we test the performance on three sub-tasks of link prediction, to demonstrate the model's ability and flexibility on both homogeneous and heterogeneous network.

Baselines To verify the effectiveness of our proposed model, we compare it with several strong baselines, including:

- **SGNS** [13]. It represents the node as the sum of the corresponding word embeddings which are learned by applying SGNS to the text sequences generated from node features.
- **DeepWalk** [15]. It derives the node embeddings by combining random walks and Skip-Gram model.
- **CANE** [18]: It learns context-aware embeddings for nodes with mutual attention mechanism and associated text information.
- **TriDNR** [14]. It learns node embeddings by jointly modeling the network structure, node-content, label-content correspondence.
- **GraphSAGE** [8]. It is an inductive model that generates node embeddings by aggregating features from a node's neighborhood.

Parameters Setting CNE is trained using Adam [11] with initial learning rate of 0.0008 and batch size of 256. For fair comparison, all baselines are with the same random-walk relevant settings as our proposed model. The node embedding dimension of all models is set to 512. For all models that leverage text features, we build a vocabulary of top 40,000 words and learn the word embeddings from scratch. For GraphSAGE, we use mean aggregator and train word vectors trained by SGNS. For models based on random walks (*e.g.*, DeepWalk, TriDNR, and CNE), we set the length of truncated random walks as l=20, set window size w=4, and randomly sample K=4 negative nodes for each positive node pair.

Task and Evaluation Metrics We use the link prediction (LP) task to evaluate the ability of our proposed model under different settings. We randomly remove a portion of existing edges from the network and use the left network to train each network embedding model. For testing, we randomly choose one thousand nodes from the network and use the learned node embeddings to predict the unobserved links. Intuitively, a larger similarity, *e.g.*, the cosine similarity, implies that the two nodes may have a higher propensity to be linked. In this way, we can employ Precision@k and Recall@k to evaluate the LP performance.

3.1 Task 1: LP on Homogeneous Network

This task is to validate that CNE possesses higher representation power although the network is incomplete.

Table 1: Precision of different training set size (Task 1).

Method	10%		30%		50%		70%		90%	
	P@10	P@100	P@10	P@100	P@10	P@100	P@10	P@100	P@10	P@100
SGNS	0.131	0.036	0.132	0.036	0.131	0.036	0.133	0.037	0.134	0.037
DeepWalk	0.208	0.105	0.220	0.117	0.237	0.123	0.237	0.123	0.244	0.125
TriDNR	0.190	0.062	0.203	0.072	0.215	0.078	0.213	0.082	0.219	0.086
CANE	0.315	0.130	0.320	0.133	0.328	0.133	0.315	0.130	0.316	0.131
GraphSAGE	0.216	0.086	0.252	0.111	0.263	0.120	0.261	0.123	0.236	0.120
CNE	**0.374**	**0.155**	**0.369**	**0.157**	**0.389**	**0.161**	**0.390**	**0.166**	**0.414**	**0.174**

Table 2: Precision of unseen test nodes (Task 2).

Method	10%		30%		50%	
	P@10	P@100	P@10	P@100	P@10	P@100
SGNS	0.016	0.003	0.016	0.003	0.016	0.003
TriDNR	0.019	0.003	0.021	0.004	0.030	0.005
CNE	**0.034**	**0.008**	**0.039**	**0.009**	**0.042**	**0.009**

We use Amazon Baby category dataset [9], consisting of 71,317 item's metadata. The homogeneous network is constructed from *co-view* relations, resulting in an item graph with 47,185 nodes and 1,166,828 edges. The portion of removed edges is varied between 10% and 90%. We treat the product title as node feature.

Results As shown in Table 1, CNE achieves much better scores than all the baselines. Note that DeepWalk and CANE require that all nodes in the graph are present during training of the embeddings. For fairness of comparison, we only use the nodes presented in the training network to construct the test set.

Importance of network topology. It is a little surprising that DeepWalk is a strong baseline in our experiments, indicating that structure provides rich information for link prediction. SGNS is based on node features solely, getting the word similarity only from word co-occurrence statistics. The gaps between CNE and SGNS suggest that network topology can improve the attribute embeddings for network embedding. This result is in consistent with the finding in [10].

Importance of jointly-training topology and attributes. As node attributes can alleviate the data sparsity problem to some extent, they do enhance the structure-based embeddings when being utilized in a reasonable way. Comparing with DeepWalk, the results show that models leveraged nodes' attributes (*e.g.*, CANE and CNE) can achieve better performances, especially on small training set. Although these methods take advantage of the nodes' attributes, they choose different ways. TriDNR and CANE all optimize the attribute embeddings indirectly through the node embeddings (The former uses a shadow model and the latter uses deep CNN); while GraphSAGE uses a fixed pre-trained attribute embeddings. The results show that our proposed CNE outperforms these baselines with a significant gap. This indicates that directly optimizing the node attribute embeddings and the encoders in an end-to-end way can act as a powerful form of regularization to improve the performances. In addition, comparing results from different groups, it is easy to find that CNE is more stable than other baselines, improving consistently along with the increase of the training set.

3.2 Task 2: LP for Unseen Nodes

This task is to test the ability of CNE on generating the embedding for unseen nodes. We use the same experimental setting as Task 1 except for the choice of test set. In this task, we focus on the nodes that did not appeared in the training set. GraphSAGE, DeepWalk, and CANE are not available for this task as they can not calculate the

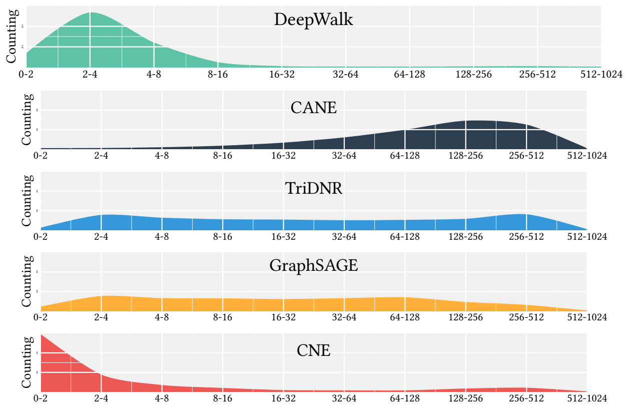

Figure 2: The distribution of the similarity ranks of the subsequently clicked item.

node embeddings without nodes structural information. Therefore, only the results of the rest baselines are presented in Table 2. In this task, we only conduct the experiments on networks with portion of training edges varied from 10% to 50% since keeping more edges will not be able to generate enough unseen nodes for test.

Results Table 2 includes the comparison between different baselines. At most of time, CNE is the best one. CNE encodes the network structure by feature embeddings and encoders, while TriDNR separately models the structural and semantic embeddings. When TriDNR is without structural information, the representation power of the semantic embeddings is too weak to accomplish link prediction. CNE on the contrary does not suffer from such trouble as long as it captures sufficient connections between network structure and semantic distances. Meanwhile, in the training of TriDNR, the word embeddings interact with network topology. Thus, it outperforms SGNS, whose word embedding is based on the context (word co-occurrence statistics). Note that Task 2 is much harder than Task 1, as the unseen nodes have quite few edges.

3.3 Task 3: LP on Multi-node-type Network

This task is designed to validate that CNE is pretty flexible to model the network of multiple node types. In this experiment, we collect users' behavior sequences from TaoBao, a popular online shopping website. The task is to predict a user's subsequent behavior (n+1) given the past n behaviors. Baselines need a homogeneous network and we construct a item graph by connecting two items if they are viewed in the same session. The network consists of 8744 nodes and 29,976 edges. All the baselines provide a node embedding for each item and the behavior sequence is represented by a *record* vector by adding n items embeddings up. Meanwhile, we construct a heterogeneous network of two kinds of nodes, user and item. Each user node is associated with n item viewing records within a session. CNE models the user node by n GRU encoders and produce the user embedding (*record* vector) by adding up the last hidden layer of n encoders. In our experiments, we let n equal 4.

Compared with the above tasks, Task 3 is more challenging as each record only corresponds to one correct answer, the most subsequent item. We calculate the cosine similarity between the candidate items' vector and the *record* vector, ranking the candidates in the descending order. Note that the record products are filtered out from the candidate. We test 1000 users' records and count the positions of the correct answers in the 1000 ordered lists.

Table 3: A user click record and the top ranked products provided by each method. Words of similar meaning are in the same color (large size, casual, feminine, hot weather)

	Rank	Product Title
Click Record	1	Spring green loose mid-sleeve casual T-shirt.
	2	Pierced lace off shoulder 3/4 sleeve loose blouse.
	3	Plus size floral printed slimming princess dresses.
DeepWalk	1	Cotton plain loose white t-shirt.
	2	Spring and summer outlet high-waist shorts.
	3	Ethnic style Thailand Napal summer holiday long dress.
CANE	1	Original design fashion loose hip pants.
	2	Ethnic style Thailand Napal summer holiday long dress.
	3	Summer sleeveless wrinkled dress.
TriDNR	1	Puff sleeve elegant floral printed blouse.
	2	Extra size slimming pierced long scarf wrap shawl.
	3	Spring and summer sleeveless casual jumpsuits.
GraphSAGE	1	Korean summer beautiful dress.
	2	Hong-kong embroidery dress.
	3	Korean summer fashion v-neck hoodie.
CNE	1	Summer flower figure-flattering princess dress.
	2	Slimming cold shoulder empire waist fairy dress.
	3	Pink colorful dotted silk long-sleeve blouse.

Results The distribution of the similarity ranks of the correct answers are presented in Fig.2. CNE tends to place the correct answer in the top three positions, and DeepWalk tends to place the correct answer in little bit farther positions. Meanwhile, it can not be ignored that a bit of correct answers appear in much farther positions provided by CNE. Besides, the other baselines that utilize node attributes also have the similar phenomenon. The reason is that a lot of products with similar attributes but long geodesic distances may be ranked in the front of the list. GraphSAGE, CANE, and TriDNR have very poor performances on this task. The text feature seems to badly confuse their predictions.

Case Study We also check the items in the top positions calculated by different methods in Table 3. The first 3 lines are the product titles in the click records and the following items are recommended by different methods. The record indicates that the user aims at large size, casual, feminine clothes in hot weather. The top ranked items provided by the baselines only have partial features and most of them are identical words. Compared with the record products, products recommended by CNE are with semantically similar titles although the words are different. This case shows that CNE learns to capture the feature relevance and is capable of predict the users' interests.

4 CONCLUSION

In this paper, we open a new frontier in network embedding by introducing the "*principle of compositionally*" to node embeddings. To address the main limitations of the existing approaches, we proposed a novel approach that can efficiently generate embeddings for unseen nodes by combining their internal attributes. Experiments demonstrate the effectiveness and generalization ability of compositional network embeddings.

5 ACKNOWLEDGEMENT

This work is supported by NSFC under Grant No.61532001 and MOE-ChinaMobile under Grant No. MCM20170503.

REFERENCES

[1] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, and Thomas S. Huang. 2015. Heterogeneous Network Embedding via Deep Architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 119–128.

[2] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research* 12 (2011), 2493–2537.

[3] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial Attack on Graph Structured Data. In *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80. PMLR, StockholmsmÃďssan, Stockholm Sweden, 1115–1124.

[4] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 135–144.

[5] Lun Du, Yun Wang, Guojie Song, Zhicong Lu, and Junshan Wang. 2018. Dynamic Network Embedding : An Extended Approach for Skip-gram based Network Embedding. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2086–2092.

[6] Gottlob Frege. 1948. Sense and reference. *The philosophical review* 57, 3 (1948), 209–230.

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

[8] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, 1025–1035.

[9] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *Proceedings of International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 507–517.

[10] Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 941–951.

[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of International Conference for Learning Representations*.

[12] Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and Pollution: Delving into Cross-Space Mapping for Zero-Shot Learning. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 378–387.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and Their Compositionality. In *Proceedings of Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, 3111–3119.

[14] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-Party Deep Network Representation. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence, 1895–1901.

[15] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, New York, NY, USA, 701–710.

[16] Meng Qu, Jian Tang, Jingbo Shang, Xiang Ren, Ming Zhang, and Jiawei Han. 2017. An Attention-based Collaboration Framework for Multi-View Network Representation Learning. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*. ACM, New York, NY, USA, 1767–1776.

[17] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *Companion Proceedings of the The Web Conference*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 953–960.

[18] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-Aware Network Embedding for Relation Modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 1722–1731.