Engineering Graph Features via Network Functional Blocks

Vincent W. Zheng
Advanced Digital Sciences Center, Singapore
vincent.zheng@adsc-create.edu.sg

Abstract

Graph is a prevalent data structure that enables many predictive tasks. How to engineer graph features is a fundamental question. Our concept is to go beyond nodes and edges, and explore richer structures (*e.g.*, paths, subgraphs) for graph feature engineering. We call such richer structures as network functional blocks, because each structure serves as a network building block but with some different functionality. We use semantic proximity search as an example application to share our recent work on exploiting different granularities of network functional blocks. We show that network functional blocks are effective, and they can be useful for a wide range of applications.

1 Introduction

Graph is a prevalent data structure that exists in many real-world scenarios; *e.g.*, friendship graphs in social networks, bibliography graphs in research publications, user interest graphs in e-commerce, and knowledge graph. As shown in Figure 1, there is a wide range of predictive tasks based on the graphs, including node classification, node regression, node ranking, link prediction, community detection, and so on. A fundamental question we try to address is how to better engineer graph features for these predictive tasks.

Traditionally, graph feature engineering largely focuses on the *node* level. A typical approach is to extract features from each node's content (*e.g.*, bag of words) and its interactions with neighbors (*e.g.*, node degree). Then these features are fed into some relational models for collective node classification [Zheng and Chang, 2016] or link prediction [Tang *et al.*, 2012]. More recent graph embedding methods try to learn a vector representation for each node in the graph, such that two nodes "close" on the graph have similar vector representations in a low-dimensional space [Cai *et al.*, 2018]. These learned node features can be fed into non-relational models for graph predictive tasks [Perozzi *et al.*, 2014; Grover and Leskovec, 2016].

Despite the success of node-level features, we assert that individual nodes, as the smallest network building blocks, carry limited information. There exist richer interactions among the nodes. For example, on an e-commerce graph, a

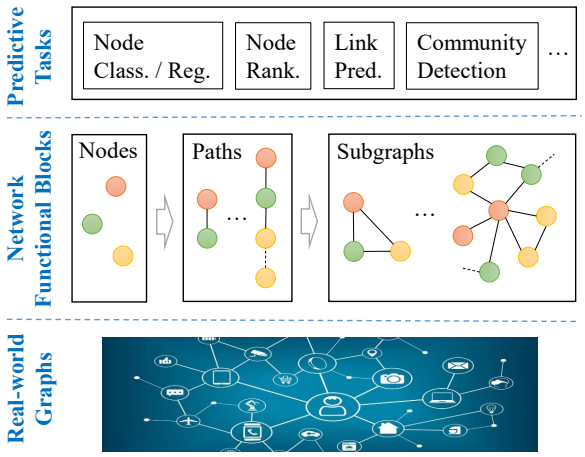

Figure 1: Exploiting network functional blocks for graph features.

path "buyer-book-seller" indicates a customer purchase behavior, whereas a *subgraph* "buyer-book&city-seller" indicate a same-city transaction. The above path and subgraph correspond to coarse-grained network building blocks. In general, we can see a graph as consisting of many building blocks, and each playing some different functionality. The functionality of each building block depends on: 1) network structure, including both nodes (*i.e.*, type, content) and their interactions (*i.e.*, network topology, interaction information); 2) graph predictive tasks, especially the task-specific supervision information. We call these network building blocks with different functionality as "Network Functional Blocks". These network functional blocks carry more information than individual nodes, and can naturally serve as useful graph features for many predictive tasks.

In this paper, we share our recent attempts to exploit the novel concept of network functional blocks for engineering graph features. Let us use *semantic proximity search* [Sun *et al.*, 2011; Fang *et al.*, 2016] as an example application to introduce our work. Semantic proximity search is closely related to node ranking and link prediction. It takes a node in a heterogeneous graph as the query, and ranks the other nodes according to a semantic relation. Consider the graph in Figure 2. The different ways that how two nodes are connected imply different kinds of semantic relations. For exam-

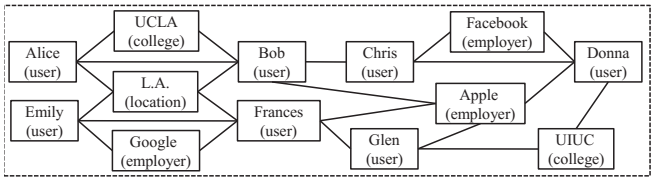

Figure 2: Semantic proximity search on a heterogeneous graph.

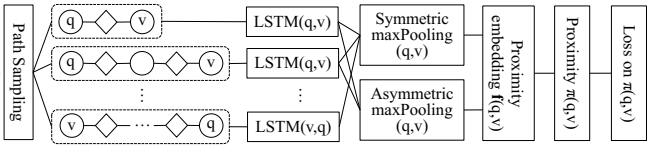

Figure 3: Overall training framework for ProxEmbed.

ple, Alice and Bob both attend UCLA, hence they are *schoolmates*; whereas Chris and Donna both work for Google, hence they are *colleagues*. Given this semantic difference, we can take any user node (*e.g.*, Alice) as a query, and ask "who are likely to be her *schoolmates*?" As the answer, we shall have Bob rank higher than the other user nodes. Semantic proximity search empowers many applications; *e.g.*, recommending connections in a professional graph such as LinkedIn, categorizing friends in a social graph such as Facebook, discovering advisors or advisees in a bibliography graph such as DBLP, and linking user identities in an e-commerce graph such as Taobao. In the following sections, we will introduce how we expand from nodes to both paths and subgraphs to engineer graph features for solving semantic proximity search.

Challenges: there exist many challenges in using network functional blocks for graph features. First of all, real-world graphs can be noisy and heterogeneous, how to generate "good" network functional blocks from these graphs? Secondly, given the rich structure of paths and subgraphs, as well as additional supervision information, how to properly model the network functional blocks for generating features? Last, but not the least, since the graph can be large and there may exist time constraint in online prediction, how to efficiently generate and use network functional blocks?

2 From Nodes to Paths

Node-based features are not ideal for semantic proximity search. For example, node embedding [Perozzi *et al.*, 2014; Dai *et al.*, 2016] tries to produce a low-dimensional vector for each single node. A typical approach to using such node embedding for proximity search is to measure the similarity between two node vectors, but it is suboptimal, because 1) it does not explicitly model the connecting structure between two possibly distant nodes; 2) it does not answer how to best compute a score from two vectors (*e.g.*, cosine similarity, Euclidean distance, or weight matrix).

To address the challenge of network functional block generation, we propose to use *paths* as the network functional blocks to approximate the connecting structure between two nodes. Paths, as bigger network functional blocks than nodes, can capture richer semantics. There is some successful prior work, such as PathSim [Sun *et al.*, 2011] and PRA [Lao and Cohen, 2010]. But they often rely on explicitly engineered path patterns. We choose to efficiently sample paths from the graph and use them for embedding to learn graph features. In practice, graphs are imbalanced (*e.g.*, much more user nodes than other types of nodes). To avoid a certain type of nodes to dominate the paths, we take node type into account in path sampling. For example, in each step of random walk at a node v, we can randomly sample one node type, and then randomly sample one neighbor of that type as the next step. More sophisticated path sampling is possible [Grover and Leskovec, 2016] and worth further studying.

To address the challenge of network functional block modeling, in [Liu *et al.*, 2017] we propose a **ProxEmbed** (Proximity Embedding) framework as shown in Figure 3. After the careful path sampling, we obtain a set of paths between a query node q and a candidate target node v as their connecting network structure. Then, we devise a recurrent neural network with *discounted path pooling* to embed multiple paths with various types of nodes and varying lengths into a vector. To differentiate the contribution of each node and each path, we can also introduce node attention and path attention in embedding [Zhao *et al.*, 2017a]. Finally, we supervise the model training with labels to identify the functionality of each path in semantic proximity search.

To address the challenge of prediction efficiency, we choose to offline sample the paths, build an index and train the model. In general, path sampling can be done in a complexity linear to the number of nodes in a graph. In the online stage, given a query, we can easily retrieve the paths between a query q to each candidate target v based on the offline index, and then predict the proximity score for ranking.

3 From Paths to Subgraphs

Although paths are useful for encoding the distance between two nodes and capture certain proximity semantics, they are low-order representations of the connecting structure. In other words, with such path-based formulations, we decompose a full connecting structure between two nodes into independent paths, each of which can only uncover partial information. In contrast, subgraphs are high-order representations, and we expect them to do better in capturing richer semantics.

3.1 Augmenting Paths with Subgraphs

Path is suitable for encoding distance, whereas subgraph is a higher-order structure. We try to combine frequent subgraph patterns with paths as a new kind of network functional block.

To address the challenge of network functional block generation, in [Liu *et al.*, 2018b] we propose to use *subgraph-augmented path* ("s-path" for short) as network functional blocks to approximate the connecting structure between two user nodes. Consider a path between a query user Alice and a target user Donna in Figure 4a. In fact, Alice and Bob not only attended the same college UCLA, but also live in the same city L.A.. Such information is missing in the path, but possible to be captured by a subgraph. We are inspired by [Benson *et al.*, 2016] to exploit frequent subgraph patterns.

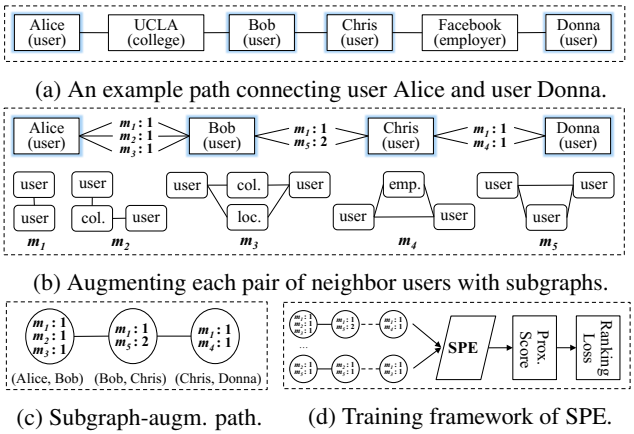

Figure 4: Augmenting paths with subgraphs.

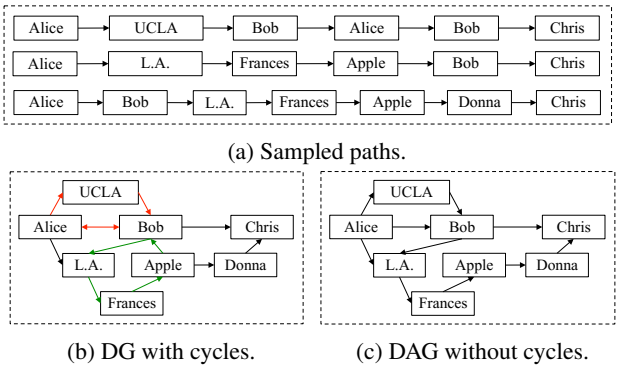

Figure 5: Proximity embedding with different structures.

Suppose we already have some offline mined frequent subgraph patterns in Figure 4b, such as "user-user" (m_1), "user-college-user" (m_2), "user-college & location-user" (m_3) and so on. Then we can replace the linear path between Alice and Bob with richer subgraph instances for m_1, m_2 and m_3. In this way, we have a more complete picture of the semantic relation between Alice and Bob. In general, such s-paths are able to leverage both path's *distance awareness* (*i.e.*, able to model multi-hop connections between two nodes) and subgraph's *high-order structure* (*i.e.*, richer than linear paths).

To address the challenge of network functional block modeling, we design a novel **SPE** (Subgraph-augmented Path Embedding) framework as shown in Figure 4. In SPE, we first take into account the structure of each subgraph, as well as the fact that not all the subgraphs are useful for a particular semantic relation (*e.g.*, m_5 is less indicative than m_2 for *schoolmates*). Besides, we also consider that s-paths are noisy in and among themselves. That is, not all the nodes in an s-path are useful; *e.g.*, if Alice and Donna are *schoolmates*, then node (Alice, Bob) in the s-path, which implies a *schoolmates* relation, is more important than the other nodes in the same s-path. Similarly, not all the s-paths are useful; *e.g.*, an s-path constructed from Alice–Emily–Frances–Donna in Figure 2 is less indicative than the one in Figure 4c for the *schoolmates* relation, since it has no clear signal. Hence, we introduce a *hierarchical attention* mechanism to automatically weigh the subgraphs in each s-path's node, the nodes in each s-path, and the s-paths between every two candidate users. Finally, we embed all the s-paths together into a vector; then we compute a proximity score and the ranking loss for model training.

To address the challenge of prediction efficiency, we take a similar path sampling and indexing approach as ProxEmbed. Yet we also offline mine frequent subgraph patterns and use them to match the input graph for subgraph instance indexing. In general, subgraph pattern mining is NP-hard; hence, we choose to control the size of subgraph patterns. Besides, based on the applications, we also propose to constrain the subgraph patterns to involve at least two users, and devise a symmetry-based matching algorithm to speed up the pattern matching [Fang *et al.*, 2016]. Thanks to the offline indexing, both online s-path generation and prediction can be efficient.

3.2 Assembling Paths into Subgraphs

Exploiting frequent subgraph patterns as network functional blocks has been shown powerful, yet mining them takes effort. We look for alternatives to generate the equally powerful network functional blocks in a more efficient manner.

To address the challenge of network functional block generation, in [Liu *et al.*, 2018c] we propose to use *DAGs* (directed acyclic graphs) as the network functional blocks to approximate the connecting structure between two nodes. Next, we motivate why we choose DAGs. Let us start by discussing the implications of using paths in Figure 5a. On the one hand, the paths are *directional*. This coincides with the intuition of modeling proximity from one node to another. Thus we are looking for a directional structure to model the proximity. On the other hand, the paths are limited in expressiveness. Modeling each path independently and aggregating them later does not fundamentally address this limitation. Thus we are also looking for a richer structure than paths. In all, we arrive at the choice of using DGs (directed graphs) to model the connections from one node to another. However, DGs can have cycles, which are not ideal for the proximity search task. For example, consider an example DG between Alice and Chris in Figure 5b. In practice, Alice can get to know Chris through she knowing a schoolmate Bob and Bob knowing Chris personally. Having an extra cycle among Alice, Bob and UCLA can make the distance between Alice and Chris becomes longer, thus weaker to explain their relation. Besides, cycles are not preferred for inference by probabilistic graphical models [Koller and Friedman, 2009], thus not suitable for our representation learning as well.

To address the challenge of network functional block modeling, we propose a novel **D2AGE** (Distance-aware DAG Embedding) model. Specifically, for a DAG, when combining the outputs of the predecessor nodes as the input of the successor, different predecessors should have varying contributions for a target relation due to their varying distances from the start node. For example, consider a DAG in Figure 5c. Chris' predecessors are Bob and Donna. The distance from the start node Alice to Bob is much shorter than that to Donna. Thus the connection between Alice and Bob is stronger than that between Alice and Donna. When combining the embedding of Bob and Donna as the inputs for Chris, we have to differentiate them by their own distance from the

	Methods	Symmetric				Asymmetric	
		Linkedin		Facebook		DBLP	
		school.	*collea.*	*class.*	*family*	*advisor*	*advisee*
NDCG@10	DeepWalk	0.518	0.506	0.689	0.575	0.743	0.379
	MGP	0.568	0.546	0.843	0.732	0.718	0.403
	Metapath2vec	0.524	0.509	0.702	0.560	0.769	0.403
	ProxEmbed	0.652	0.606	0.851	0.743	0.765	0.405
	SPE	**0.690**	**0.686**	**0.866**	0.761	**0.782**	0.413
	D2AGE	0.678	0.639	0.856	**0.765**	0.779	**0.417**
MAP@10	DeepWalk	0.391	0.369	0.575	0.455	0.659	0.264
	MGP	0.305	0.333	0.729	0.651	0.663	0.280
	Metapath2vec	0.395	0.378	0.589	0.436	0.683	0.281
	ProxEmbed	0.541	0.492	0.801	0.678	0.690	0.283
	SPE	0.581	**0.575**	**0.822**	0.704	0.700	**0.298**
	D2AGE	**0.587**	0.528	0.808	**0.713**	**0.703**	0.290

Table 1: Comparison with the baselines with 100 labels.

start node. Note that such distance-awareness can happen at any node in the DAG with multiple predecessors. Therefore, we introduce a *recursive distance discount mechanism* when embedding the DAGs. That is, for each node v in a DAG, we assign different weights to its different predecessors w.r.t. their distances from the start node. Generally, the further v is from the start node, the weaker their connection is. Finally, in D2AGE, we recursively apply the distance discounts from multiple predecessors in DAG embedding. Given the training supervision, we devise an end-to-end solution.

To address the challenge of prediction efficiency, we take a path assembling and cycle removal approach to generate the DAGs in both offline and online stages. In contrast, a brute force approach to generate a DAG from a start node q to an end node v on a graph is to use BFS (Breadth First Search), starting from q. However, running BFS on the whole graph is expensive, taking an $O(|V| + |E|)$ complexity with V and E as the node set and edge set respectively. It is costing in the online stage for new query nodes to compute their DAGs. Our path assembling and cycle removal approach has two insights. On the one hand, sampling paths is much more efficient and can be done offline. On the other hand, they can be easily assembled into more complex structures such as DGs. As we can control the number of paths to assemble for two nodes and the length of these paths, we guarantee the induced DGs to have a constant graph size. Thus running BFS on such a small DGs to generate DAGs is easy.

4 Empirical Studies

We empirically study the performance of the above three approaches of network functional blocks for semantic proximity search. For datasets, we use three heterogeneous graphs, including a professional network *LinkedIn*, a social network *Facebook*, and a bibliography network *DBLP*. We consider different semantic relations on the datasets, including *schoolmate* and *colleague* in LinkedIn, *family* and *classmate* in Facebook, *adivsor* and *advisee* in DBLP. Here, *adivsor* and *advisee* are asymetric relations, whereas the others are symmetric ones. Details of datasets, ground truth and training setup can be found in [Liu *et al.*, 2018a]. We adopt NDCG and MAP as node ranking evaluation metrics.

We compare our models ProxEmbed, SPE and D2AGE with the following state-of-the-art baselines: DeepWalk [Perozzi *et al.*, 2014], MGP [Fang *et al.*, 2016] and Metapath2vec [Dong *et al.*, 2017]. As shown in Table 1, our models tend to outperform the baselines on all the datasets. DeepWalk is an indirect approach for semantic proximity search, since it focuses on node-level features rather than the direct connecting structure between two nodes. MGP gives decent results thanks to using frequent subgraph patterns for proximity estimation. But it requires additional efforts of frequent subgraph mining and it is unable to learn implicit graph features. Metapath2vec has a comparable performance with DeepWalk. Surprisingly, it does not seem to benefit much from its meta-path patterns. This may be because the meta-path patterns are limited and manually designed, which are unlikely to be optimal. Among our methods, SPE tends to perform well in most datasets, which shows the value of exploiting explicit subgraph pattern as network function blocks. D2AGE has comparable performance with SPE and sometimes outperforms it. Since the network functional block generation in D2AGE is more efficient than SPE, the above observation suggests that distance-aware DAGs are very effective network functional blocks for semantic proximity search.

5 Discussions

Network functional blocks can be useful in many graph predictive applications, in addition to semantic proximity search. For example, we can use path modeling for community-based question answering [Zhao *et al.*, 2017b], or use interactive path structure modeling for e-commerce user ID matching [Liu *et al.*, 2018a], or use cascade modeling for detrimental prescribing cascade detection [Hoang *et al.*, 2016] and social media diffusion prediction [Wang *et al.*, 2017].

There is still a lot of room to further develop network functional blocks for graph feature engineering. On the one hand, designing network functional blocks can expand from explicit patterns to implicit ones. For example, community is an important structure in graphs, but it is often implicit due to probabilistic assignments [Cai *et al.*, 2017]. There has been some attempt to use community to assist node embedding [Cavallari *et al.*, 2017]. Another example is graph convolution, which finds implicit higher-order subgraph patterns [Meng *et al.*, 2018] or implicit spectral patterns from k-th order neighborhood [Kipf and Welling, 2017]. On the other hand, using network functional blocks shall extend from academic settings to industrial ones. In practice, graphs easily have billions of nodes, and continue to grow and evolve over time. How to efficiently generate meaningful network functional blocks from such a billion-scale graph, and put it to work in a distributed setting, is underexplored. Besides, how to enable the incremental update of network functional block generation and model training also remains an open question.

6 Conclusion

Network functional block is a useful concept to guide graph feature engineering. We have made several attempts to exploit different kinds of network functional blocks, ranging from paths to subgraph-augmented paths and distance-aware DAGs. These rich network functional blocks are shown effective in the semantic proximity search application. Various network functional blocks can be further explored for a wide

range of graph predictive tasks. In future, we are particularly interested in developing network functional blocks for graph feature engineering in a large-scale and dynamic setting.

Acknowledgments

This research is supported by National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme, and Alibaba Group under its Alibaba Innovative Research (AIR) programme.

References

[Benson *et al.*, 2016] Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 353(6295):163–166, 2016.

[Cai *et al.*, 2017] Hongyun Cai, Vincent W. Zheng, Fanwei Zhu, Kevin Chen-Chuan Chang, and Zi Huang. From community detection to community profiling. *PVLDB*, 10(7):817–828, 2017.

[Cai *et al.*, 2018] Hongyun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *TKDE*, 2018.

[Cavallari *et al.*, 2017] Sandro Cavallari, Vincent W. Zheng, Hongyun Cai, Kevin Chen-Chuan Chang, and Erik Cambria. Learning community embedding with community detection and node embedding on graphs. In *CIKM*, pages 377–386, 2017.

[Dai *et al.*, 2016] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, pages 2702–2711, 2016.

[Dong *et al.*, 2017] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, pages 135–144, 2017.

[Fang *et al.*, 2016] Yuan Fang, Wenqing Lin, Vincent W. Zheng, Min Wu, Kevin Chen-Chuan Chang, and Xiaoli Li. Semantic proximity search on graphs with metagraph-based learning. In *ICDE*, pages 277–288, 2016.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[Hoang *et al.*, 2016] Tao Hoang, Jixue Liu, Nicole Pratt, Vincent W. Zheng, Kevin C. Chang, Elizabeth Roughead, and Jiuyong Li. Detecting signals of detrimental prescribing cascades from social media. *Artificial Intelligence in Medicine*, 71:43–56, 2016.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[Koller and Friedman, 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[Lao and Cohen, 2010] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, 2010.

[Liu *et al.*, 2017] Zemin Liu, Vincent W. Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. Semantic proximity search on heterogeneous graph by proximity embedding. In *AAAI*, 2017.

[Liu *et al.*, 2018a] Zemin Liu, Vincent W. Zheng, Zhou Zhao, Zhao Li, Hongxia Yang, Minghui Wu, and Jing Ying. Interactive paths embedding for semantic proximity search. In *KDD*, 2018.

[Liu *et al.*, 2018b] Zemin Liu, Vincent W. Zheng, Zhou Zhao, Hongxia Yang, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. Subgraph-augmented path embedding for semantic user search on heterogeneous social network. In *WWW*, pages 1613–1622, 2018.

[Liu *et al.*, 2018c] Zemin Liu, Vincent W. Zheng, Zhou Zhao, Fanwei Zhu, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. Distance-aware DAG embedding for proximity search on heterogeneous graphs. In *AAAI*, 2018.

[Meng *et al.*, 2018] Changping Meng, S. Chandra Mouli, Bruno Ribeiro, and Jennifer Neville. Subgraph pattern neural networks for high-order graph evolution prediction. In *AAAI*, 2018.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, pages 701–710, 2014.

[Sun *et al.*, 2011] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. *PVLDB*, 4(11), 2011.

[Tang *et al.*, 2012] Jie Tang, Alvis Cheuk M. Fong, Bo Wang, and Jing Zhang. A unified probabilistic framework for name disambiguation in digital library. *IEEE Trans. Knowl. Data Eng.*, 24(6):975–987, 2012.

[Wang *et al.*, 2017] Jia Wang, Vincent W. Zheng, Zemin Liu, and Kevin Chen-Chuan Chang. Topological recurrent neural network for diffusion prediction. In *ICDM*, pages 475–484, 2017.

[Zhao *et al.*, 2017a] Zhou Zhao, Ben Gao, Vincent W. Zheng, Deng Cai, Xiaofei He, and Yueting Zhuang. Link prediction via ranking metric dual-level attention network learning. In *IJCAI*, pages 3525–3531, 2017.

[Zhao *et al.*, 2017b] Zhou Zhao, Hanqing Lu, Vincent W. Zheng, Deng Cai, Xiaofei He, and Yueting Zhuang. Community-based question answering via asymmetric multi-faceted ranking network learning. In *AAAI*, pages 3532–3539, 2017.

[Zheng and Chang, 2016] Vincent W. Zheng and Kevin Chen-Chuan Chang. Regularizing structured classifier with conditional probabilistic constraints for semi-supervised learning. In *CIKM*, pages 1029–1038, 2016.