Bias in Structural Variation of Graphs

Aayam K. Shrestha
Oregon State University
{shrestaa}@oregonstate.edu

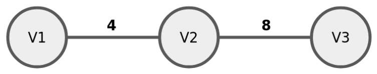
Abstract

Graphs are ubiquitous, representing a variety of natural processes as diverse as friendships between people, communication patterns, and interactions between neurons in the brain. This is, in turn, fostering an ecosystem of machine learning algorithms applied on top of features extracted from these graphs. However, As we are moving from static to dynamic/semi-structured graphs the representational variation is being more evident across the graphs. The algorithms running on top of the features so extracted may have inconsistent results across the representational variation of graphs. This distinction of algorithms behaving differently for different representation is crucial but subtle. Hence very less thought is given currently on the robustness of the performance of algorithms over the representational variations of graphs. to this end, We propose a formal framework to define "bias" of the feature queries over representational variations of graphs. We also provide empirical evidence that the algorithms do indeed behave differently over different representations over a host of algorithms over both homogeneous and heterogeneous graphs. Moreover, we propose a novel metric to characterize the optimum normalized representation of the graph database.

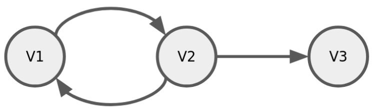
1 Introduction

The data challenges today centers more around connections, and not just discrete data. This has led to increased use and interest in representing the data in a graph structure. A few noteworthy examples being social networks for analyzing a collaborative system, Knowledge graphs for common query answering, Product Graphs for recommendations and Analysis of networks in cell/molecular biology. As more data is being created and well represented in a graph structure, it is only but natural that machine learning community is getting more invested in leveraging the value of these networks by building inference and predictive models on top of them. The most common way to derive these models from a graph structure is to extract features from these graphs and run machine learning algorithms on top of it. The feature engineering process, however, can range from random walks used in the graph used in latent embedding generation like node2vec) to handcrafted features like clustering coefficient used for predictive modeling in brain graphs.

Recently, researchers have noticed that the as in the relational world, it is quite evident that there is more than one way to represent the data in a graph, the variation of which is often underlined by their constraints. But it has been recently shown that some algorithms perform better under a particular representation than the other [25]. It is alarming, as the semantics of representation of these Graphs are getting more blurred by the day with collaborative building as well as automatic generation of the knowledge graphs. The problem is only exacerbated in dynamic graphs where the network can evolve as a function of time as a social network or a vulnerability network of a cyber-security project. Given this, it is crucial that we come up with a framework that deals with this uncertainty in a systematic way. To this end, We suggest a novel framework to define the robustness of feature queries over these type of graphs in representational variance settings. Our contributions are three-fold:



(a) A weighted graph.



(b) A directed graph.

Figure 1: Examples of weighted and directed graphs.

i) Define a framework for quantifying the robustness of a set of feature queries in terms of feature classes over representational variations of the graph. The representational variances are defined by the semantic constraints over the graph.

ii) We provide empirical evidence that the representational variance of graph data influences the performance of an algorithm over both homogeneous and heterogeneous graphs and a number of learning algorithms in various settings.

iii) We try to identify various metrics on identifying an approximately optimal representation of graphs across various classes of algorithms. [incomplete]

2 Background

2.1 Homogenous Graphs

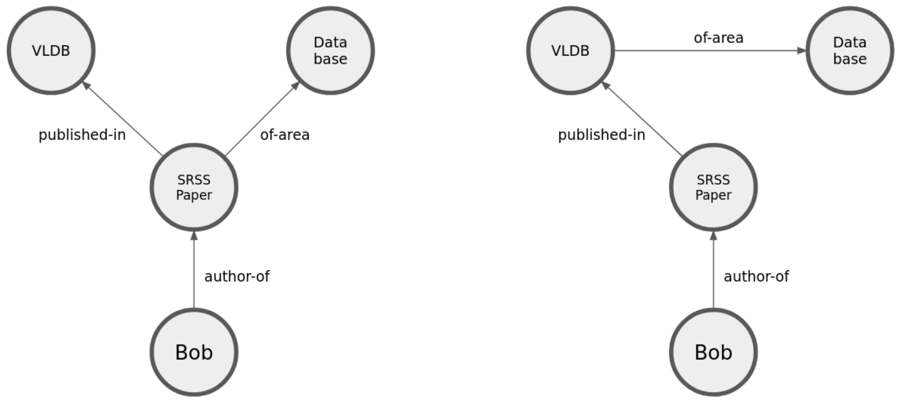
A graph is said to be homogeneous if both nodes and edges belong to a single type respectively. The homogeneous graph can be further categorized as the weighted (or directed) and unweighted (or undirected). The undirected and unweighted homogeneous graph is the most basic graph embedding input setting. A number of embedding techniques have been explored under this setting, e.g., [15, 18, 26, 22] They treat all nodes and edges equally, as only the basic structural information of the input graph is available. Intuitively, the weights and directions of the edges provide more information about the graph and may help represent the graph more accurately in the embedded space. For example, in Fig. 1(a), v2 should be embedded closer to v1 than v3 because the weight of the edge e_{12} is higher. Similarly, v2 in Fig. 1(b) should be embedded closer to v1 than v3 as v2 and v1 are connected in both directions. The above information is lost in the unweighted and undirected graph. However, the advantages of exploiting the weight and direction property of the graph edges have been explored in [21, 8, 19, 9, 2, 13].

2.2 Knowledge Graphs

A graph is said to be heterogeneous if both nodes and edges belong to one of the multiple types respectively. A knowledge graph can be viewed as an instance of the heterogeneous graph as the entities (nodes) and relations (edges) are usually of different types. For example, in a publication related knowledge graph constructed from DBLP, the types of entities can be "paper", "conference", "person", etc and the types of relations can be "author-of", "published-in", "chaired-by". Knowledge Graphs are also more generally described as property graphs where each node and edge can have properties on top of being of different types. But it can be shown that a property graph can be normalized into a heterogenous graph whose nodes are entities and edges are subject-property-object triple facts. Here each edge is represented in the form (head entity, relation, tail entity) $< h, r, t >$ For example, in Fig. 2(a), can be presented as 3 triplets: < Bob, author-of, SRSS-paper >, < SRSS-paper, published-in, VLDB > < SRSS-paper, of-area, Database >

2.3 Variations in Knowledge Graphs

It has been evident that there are multiple ways representing the same data in a relational perspective and much work has been done in the context of Data exchange and schema mapping management in a relational domain. Naturally, such variations in representation exist in Graph structures as well. As an example, consider two bibliographic datasets from DBLP (dblp.uni-trier.de) and SIGMOD Record (sigmod.org/ publications) whose fragments are shown in Figure 2.Intuitively, these databases



(a) DBLP Schema. (b) Sigmod Schema.

Figure 2: fragments from two Variations of Graph datasets.

represent essentially the same set of relationships between the same set of paper, conference, and research area entities. But, each dataset has its own way of organizing these entities and their relationships. For example, DBLP connects each paper to its research areas and conferences. Given that all papers in a conference share the same set of research areas, one can also choose the structure in Figure 1(b) for this information and connects research areas directly to conferences instead of papers.

People have looked into designing schema independent learning algorithms for certain domains like similarity search [25]. However, it is worthwhile to explore a framework to handle the variability independent of the learning algorithm. While there exist multiple types of variation, they can be broadly classified into Variations generated by knowledge preserving and lossy transformations. For the purpose of our investigation, we will be only focusing on knowledge preserving transformations.

we look into knowledge preserving transformation as being facilitated by the underlying constraints in the dataset. For eg., the functional dependency (paper, conference) -> (paper, area) is what allows the two variants of the dataset to be knowledge preserving.

2.4 Node2Vec

Node2Vec [11] is an algorithmic framework for learning continuous feature representations for nodes in networks. It is one of the widely used deep learning based graph embedding algorithms which employs Random walks. It preserves higher-order proximity between nodes by maximizing the probability of occurrence of subsequent nodes in fixed length random walks. Moreover, it uses biased random walks that provide a trade-off between breadth-first (BFS) and depth-first(DFS) graph searches. With good hyper-parameters tuning the algorithm is able to preserve the community structure as well as structure equivalence between nodes.

2.5 TransE

TransE [6] is one of the most popular translational distance model for graph embeddings. Translational distance models exploit the distance-based scoring functions. They measure the probability of a triplet as a distance between two entities. It represents both entities and relations as vectors in the same space, say $\mathbf{R}^d$. Given a triplet <h,r,t>, the relation is interpreted as a translation vector r so that the embedded entities h and t can be connected by r with low error, i.e., $h + r \approx t$ when <h, r, t>holds. In multi-relational data, Fig. 2a we can get something like Alice + author-of $\approx$ SRSS Paper and SRSS Paper + Published-in $\approx$ Avatar. The scoring function is then defined as the (negative) distance between $h + r$ and t , i.e.,

$$f_r(h,t) = - \mid\mid h + r - t \mid\mid_{\frac{1}{2}}$$

The score is expected to be large if <h, r, t> holds.

2.6 DistMult

DistMult [24] tackles the link prediction task by optimizing a scoring function containing a bilinear product between vectors for each of the subject and object entities and a diagonal matrix for each relation. In simple words each entity is represented as a vector of latent features and each relation is represented as a matrix which models featurewise interactions between the latent vectors. Note that the relationship matrix is a diagonal matrix per relation type, which limits the linear transformation performed on entity vectors to a stretch. The score of a fact <h,r,t> is defined by a bilinear function.

$$f_r(h,t) = h^T diag(r) t = \sum_{i=0}^{d-1} [r]_i \cdot [h]_i \cdot [t]_i$$

where $h, t \in \mathbf{R}^d$ are vector representations of the entities, and $diag(r) \in \mathbf{R}^d$is a diagonal matrix associated with the relation. This score captures pairwise interactions between only the components of h and t along the same dimension (see also Fig. 3b), and reduces the number of parameters to $\mathcal{O}(d)$ per relation. However, since $\mathbf{h}^T diag(\mathbf{r})\mathbf{t} = \mathbf{t}^T diag(\mathbf{r})\mathbf{h}$ for any h and t, this over-simplified model can only deal with symmetric relations which is clearly not powerful enough for general KGs.

2.7 ComplEx

ComplEx [20] extends DistMult by introducing a complex-valued embeddings so ast to better model asymmetric relations. In ComplEx, entity and relation embeddings <h,r,t> no longer lie in a real space but a complex space, say Cd. The score of a fact <h,r,t> is defined as

$$f_r(h,t) = Re(h^T diag(r)\hat{t}) = Re\left(\sum_{i=0}^{d-1} [r]_i \cdot [h]_i \cdot [\hat{t}]_i\right)$$

where $\hat{t}$ is the conjugate of t and $Re(\cdot)$ means taking the real part of a complex value. This scoring function is not symmetric any more, and facts from asymmetric relations can receive different scores depending on the order of entities involved. [12] recently showed that every ComplEx has an equivalent HolE, and conversely, HolE is subsumed by ComplEx as a special case in which the conjugate symmetry is imposed on embeddings.

2.8 Holographic Embeddings (HoLE)

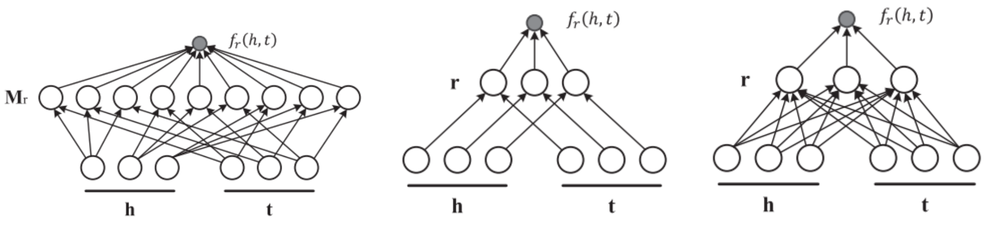
Hole [16] combines the expressive power of RESCAL [17] with the efficiency and simplicity of DistMult. It represents both entities and relations as vectors in $\mathbf{R}^d$. Given a triplet <h,r,t> the entity representations are first composed into h ? $h \star t \in \mathbf{R}^d$ by using the circular correlation operation [44], namely

$$[h \star t]_i = \sum_{k=0}^{d-1} [h]_k \cdot [t]_{(k+i)\%d}$$

The compositional vector is then matched with the relation representation to score that triplet, i.e.,

$$f_r(h,t) = r^T(h \star t) = \sum_{i=0}^{d-1} [r]_i \sum_{k=0}^{d-1} [h]_k \cdot [t]_{(k+i)\%d}$$

Circular correlation makes a compression of pairwise interactions (see also Fig. 3c). So HolE requires only $\mathcal{O}(d)$ parameters per relation, which is more efficient than RESCAL. Meanwhile, since circular correlation is not commutative, i.e. $h \star t \neq t \star h$, HolE is able to model asymmetric relations as RESCAL does.



(a) RESCAL. (b) DistMult. (c) HoLE.

Figure 3: Simple illustrations of RESCAL, DistMult, and HolE. The figures are adapted from [16]

3 Approach

We fix a countably infinite set of node ids denoted by $\mathcal{V}$. Let $\mathcal{L}$ be a finite set of labels. A *database* D over $\mathcal{L}$ is a directed graph (V, E) in which V is a finite subset of $\mathcal{V}$ and $E \subseteq V \times L \times V$. This definition of graph databases is frequently used in the graph data management literature [3, 23, 10]. We denote an edge from node u to node v whose label is a as (u, a, v). We say that $(u, a, v) \in D$ whenever $(u, a, v) \in E$. Similarly, we say that $v \in D$ whenever $v \in V$.

Database constraints restrict the instances of a schema. They are usually expressed as logical formulas over schemas [5, 4, 1]. Two widely used type of constraints are *tuple-generating* and *equality-generating* dependencies [5, 4]. A tuple-generating dependency (*tgd* for short) over schema $\mathcal{L}$ is in the form of $\forall \bar{x}(\phi(\bar{x}) \rightarrow \exists \bar{y} \psi(\bar{x}, \bar{y}))$ where $\bar{x}$ and $\bar{y}$ are sets of variables, and ϕ and ψ are logical formulas in a query language over $\mathcal{L}$. A *full tgd* does *not* have any existential variable in its conclusion. An equalitygenerating dependency (*egd* for short) over schema $\mathcal{L}$ is in the form of $\forall \bar{x}(\phi(\bar{x}) \rightarrow x_1 = x_2)$ where $\bar{x}$ is a set of variables, $x_1, x_2 \in \bar{x}$, and ϕ is logical formulas in a query language over $\mathcal{L}$.

3.1 feature query and feature class

In machine learning, a feature is simply an individual measurable property or characteristic of the object/entity under evaluation/analysis. They are the basic building blocks of the dataset and have a major impact on the performance of algorithms learning from these datasets. The features in some sense define the theoretical limit of learnability from a particular dataset. Features are easy to spot when we are working with normalized relational data as each column represent a feature. Needless to say one can improve the quality of your dataset's features with processes like feature selection and feature engineering.

When we are tasked with learning directly over a database the feature-set we are dealing with is not that clear as there are almost infinite queries that can be run over the database to get many data-points. But it is worth noting that the expressivity of the queries comes from the query language and are capped in terms of the expressivity of the query language itself. The framework for representing feature queries on these query language has already been explored in [14] for classifiers along with the question of separability(learnability). Much of the same literature can be followed in graph databases as well with few minor differences.

3.2 Bias in features

Schema - S is a graph schema / database
Query Language - QL is a language by which we can query a database. Eg. RPQ, CRPQ.
Query - q is a query defined using any query language.
Query set - Q is a set of queries q A feature f is all the data coming from a particular query q.
A Query Class QC is a set of all the queries that can be performed over Query Language QL with some constraints. hence $QC \subseteq QL$

Definition 1: Knowledge preserving Transformation $\mathcal{T}$
A Tranformation function $\mathcal{T}$ takes a schema S as an input and transforms it into a new-Schema.

$S' = \mathcal{T}(S)$. A transformation function $\mathcal{T}$ is said to be a knowledge preserving transformation if and only if there exists a mapping function $\mathcal{M}$ which maps each query q over S to q' over S' such that the results from both q and q' are exactly the same.

$Q' = \mathcal{M}(Q)$such that there exists $q' \in Q'$ such that $result(q') = result(q)$ for all $q \in Q$

Also let us define a mapping function generator $MapGen$ that takes an old and new schema as an input and generates the Mapping function $\mathcal{M}$ such that it can transform any query q over S to q' over S'.

Moreover we notice that these transformation are supported by the given set of constraints C over schema S. Hence we modify our definition slightly to $S' = \mathcal{T}(S, C)$. It is to be noted that there are more than one transformation function T for a given schema and set of constraints C. Let τ_{SC} be set of transformations over schema S employing constraint set C

let § be the set of schema generated from $t \in \tau$ such that.

$$\S = \{S' : S' = \mathcal{T}(S, C), \mathcal{T} \in \tau_{SC}\}$$

Q = the set of queries (features) q that our current algorithm relies upon / parameters of interest.
§ = the set of information preserving graphs S' each defined by over a set of constraints C
$\{Q'\}$ = set of transformed query set. Each element corresponding to the set of new queries, for each knowledge preserving transformation $\mathcal{T} \in \tau$
QC = set of queries that are supported by some additional constraint over the Query language as defined by user.

Now it is to be noted that for all $q \in Q$ does not guarantee that $q \in QC$

Given $Q' = \mathcal{M}(Q)$, then we define Q'_c such that $Q'_c \subset Q'$

$$Q'_c = \{q' : q' \in QC\}$$

$$Bias = \underset{Q' \in \{Q'\}}{\mathbb{E}} \left[\frac{|Q| - |Q'_c|}{|Q|} \right]$$

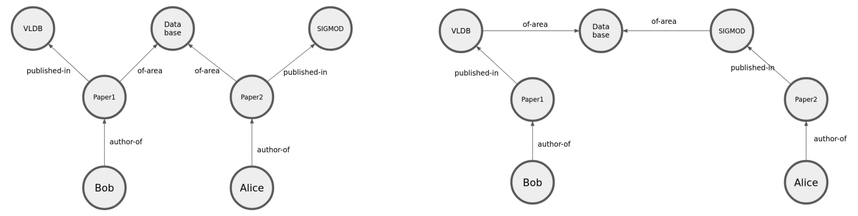
Here we can see that the Bias goes to 0 if the transformation is an identity function and no queries are mapped to any query class bigger than the query class in the Source Schema. And the Bias goes to 1 as the transformations get more complicated resulting in more transformed queries to flow out to a more complicated query class.

Example: Let us consider embedding the two variations of graph databases as shown in fig 4(a) and 4(b). let us build an embedding using node2vec algorithm with a max-walk-path of 4. It is evident that it will be able to capture the idea of two author publishing in the same area of research. (we can reach Alice from Bob in 4 hops). But if we employ the same algorithm in 4(b) then it will completely miss this relationship as the authors who publish in the same area are now 6 hop away. A quick approach to tackle this would be to just increase he max-walk-path from 4 to 6, but it is to be noted that it might not fix the problem completely as the node 6 hop is still less likely to be visited in the random walk the node 4 hop apart. Also as the hop distances increase linearly as the number of connected relations grows, it may not be even feasible to always increase the random walk length. Hence we will have two different embeddings for same data due to the different structure and hence affect the performance of analysis performed using them.

4 Experiments

4.1 Expermient design on node2vec

Machine learning on graphs is not as straightforward as with relational data. graph analytics algorithms are generally both compute and memory incentive. In order to tackle this, it is almost always a good idea to generally find a better representation of the graph. This is done by employing different encoding methods so that graph analytics algorithms can be run on these encoding(features). DeepWalk/Node2vec has been the most popular of the bunch. In brief, The idea is to make random walks from the start node of a fixed walk length l. the collection of these walks is the document representing this node. Then we use something like doc2vec to get the embedding from this set of random walks. In sum, we are encoding a node as a representation of the nodes around it.



(a) DBLP Schema.

(b) SIGMOD Schema.

Figure 4: Snippets from two Variations of bibliography Graph Databases

We generally refer to the learned embedding as the learned features where we can run our favorite machine learning algorithms on top. But if we look more closely the actual feature set is the random walk of the graph. if we change the underlying structure of the graph database then the underlying feature set will not be the same. It is possible to translate the query set (random walk in this case) but the walk length can change in this transformation and hence the feature class changes. Hence the algorithm may be biased towards some structure given the query-set and set of constraints in the database.

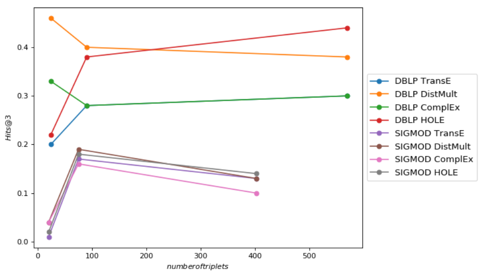
Experiments are yet to be performed but it seems natural that we may get different results for different embeddings. a dummy scenario would be a node classification problem and it was dependent on the 5th friend circle of a person. once the database is transformed and the Friends are moved beyond 5 hops, the learned embedding may not be able to capture that information and hence perform poorly.

4.2 link prediction using embeddings

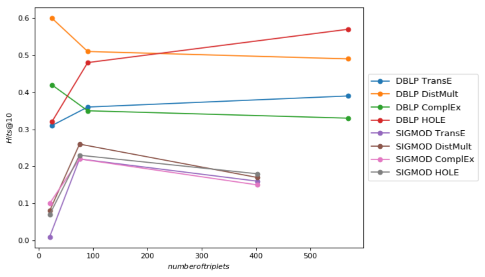
For evaluation, we use the same filtered evaluation protocol (ranking procedure) as in [7]. During training and validation we transform each triplet $< h, r, t >$ into two examples: tail query $< h, r, ? >$ and head query $<?, r, t >$. We train the model by minimizing negative log-likelihood (NLL calculated from the scoring function) of the ground truth triplet $< h, r, t >$ against randomly sampled pool of M negative triplets $< h, r, t >, t' \in \{t\}$(this applies for tail queries, head queries are handled analogically).

For each test triplet, the head is removed and replaced by each of the entities of the dictionary in turn. Dissimilarities (or energies) of those corrupted triplets are first computed by the models and then sorted by ascending order; the rank of the correct entity is finally stored. This whole procedure is repeated while removing the tail instead of the head. We report the mean of those predicted ranks and the hits@1, hits@3 and, hits@10 i.e. the proportion of correct entities ranked in the top 1, 3 and 10 respectively. In the filtered protocol we rank the validation or test set triplet against all corrupted (supposedly untrue) triplets – those that do not appear in the train, valid and test dataset.

We carry out the same evaluation procedure on 2 variations of dataset(DBLP and SIGMOD) each with 4 different embed-dings discussed previously. TransE, Complex, MultDist and Hole. We then repeat the entire experiment on multiple dataset size of same schema representations. Ideally since both the datasets are representing the same data, they should the same performance on the evaluation metrics. But this does not seem to be true in the real world.It is seen clearly in fig 5. that one representational structure of data distinctly outperforms across all embeddings and irrespective of dataset size. Hence it can be concluded that most of the deep learning embedding based methods as well, despite they not being query driven as node2vec still suffer from the variability of the performance with variability in the structure of graph databases.



(a) Hits@3 across Variations and Embedding.



(b) Hits@10 across Variations and Embedding

Figure 5: Performance metrics for different variations of graph database (DBLP and SIGMOD) across different embeddings (TransE, Hole, Complex, MultDist)

5 Discussion

Hence we have shown that the structural variation leads to variability in the performance of algorithms in a spectrum of graph analysis tasks. namely, classic graph algorithms, homogeneous graph embeddings based algorithm like node2vec as well as a heterogeneous graph like link prediction embedding algorithm like TransE, ComplEx, MultDist and Hole. To our knowledge, this is the first showcase of such bias on structural variation of knowledge graphs.

Moreover, Under the framework of treating queries as features and then applying machine learning on top of them, We define a framework for quantifying the robustness of a set of feature queries in relation to the representational variations in underlying graph databases facilitated by the semantic constraints of the graph.

We observe that a novel metric can be defined which can be optimized when searching over the space of representational variance to reduce the variance of the generated embeddings. This can in turn increase the accuracy of ML algorithms learned in downstream applications.

References

[1] Serge Abiteboul, Richard Hull, and Victor Vianu. Foundations of databases: The logical level. 1995.

[2] Amr Ahmed, Nino Shervashidze, Shravan M. Narayanamurthy, Vanja Josifovski, and Alexander J. Smola. Distributed large-scale natural graph factorization. In *WWW*, 2013.

[3] Pablo Barceló, Jorge Pérez, and Juan L. Reutter. Schema mappings and data exchange for graph databases. In *ICDT*, 2013.

[4] Catriel Beeri and Moshe Y. Vardi. A proof procedure for data dependencies. *J. ACM*, 31:718–741, 1984.

[5] Iovka Boneva, Angela Bonifati, and Radu Ciucanu. Graph data exchange with target constraints. In *EDBT/ICDT Workshops*, 2015.

[6] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[7] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *CIKM*, 2015.

[9] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *AAAI*, 2016.

[10] Wenfei Fan, Yinghui Wu, and Jingbo Xu. Functional dependencies for graphs. In *SIGMOD Conference*, 2016.

[11] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *KDD : proceedings. International Conference on Knowledge Discovery Data Mining*, 2016:855–864, 2016.

[12] Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. In *ACL*, 2017.

[13] Zhipeng Jin, Ruoran Liu, Qiudan Li, Daniel Dajun Zeng, Yongcheng Zhan, and Lei Wang. Predicting user's multi-interests with network embedding in health-related topics. *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 2568–2575, 2016.

[14] Benny Kimelfeld and Christopher Ré. A relational framework for classifier engineering. *SIGMOD Record*, 47:6–13, 2018.

[15] Tong Man, Huawei Shen, Shenghua Liu, Xiaolong Jin, and Xueqi Cheng. Predict anchor links across social networks via an embedding approach. In *IJCAI*, 2016.

[16] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. Holographic embeddings of knowledge graphs. In *AAAI*, 2016.

[17] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[18] Bryan Perozzi, Rami Al-Rfou', and Steven Skiena. Deepwalk: online learning of social representations. In *KDD*, 2014.

[19] Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie-Yan Liu. Learning deep representations for graph clustering. In *AAAI*, 2014.

[20] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *ICML*, 2016.

[21] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *KDD*, 2016.

[22] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *AAAI*, 2017.

[23] Peter T. Wood. Query languages for graph databases. *SIGMOD Record*, 41:50–60, 2012.

[24] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *CoRR*, abs/1412.6575, 2015.

[25] Yodsawalai and Chodpathumwan. Structurally robust similarity search. 2019.

[26] Xiaohan Zhao, Adelbert Chang, Atish Das Sarma, Haitao Zheng, and Ben Y. Zhao. On the embeddability of random walk distances. *PVLDB*, 6:1690–1701, 2013.