Attention Models in Graphs: A Survey

JOHN BOAZ LEE, WPI, USA
RYAN A. ROSSI, Adobe Research, USA
SUNGCHUL KIM, Adobe Research, USA
NESREEN K. AHMED, Intel Labs, USA
EUNYEE KOH, Adobe Research, USA

Graph-structured data arise naturally in many different application domains. By representing data as graphs, we can capture entities (*i.e.*, nodes) as well as their relationships (*i.e.*, edges) with each other. Many useful insights can be derived from graph-structured data as demonstrated by an ever-growing body of work focused on graph mining. However, in the real-world, graphs can be both large – with many complex patterns – and noisy which can pose a problem for effective graph mining. An effective way to deal with this issue is to incorporate "attention" into graph mining solutions. An attention mechanism allows a method to focus on task-relevant parts of the graph, helping it to make better decisions. In this work, we conduct a comprehensive and focused survey of the literature on the emerging field of graph attention models. We introduce three intuitive taxonomies to group existing work. These are based on problem setting (type of input and output), the type of attention mechanism used, and the task (*e.g.*, graph classification, link prediction, *etc.*). We motivate our taxonomies through detailed examples and use each to survey competing approaches from a unique standpoint. Finally, we highlight several challenges in the area and discuss promising directions for future work.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning**; **Logical and relational learning**; • **Mathematics of computing** → **Graph algorithms**; *Combinatorics*; *Graph theory*; • **Information systems** → **Data mining**; • **Theory of computation** → **Graph algorithms analysis**; **Streaming, sublinear and near linear time algorithms**.

Additional Key Words and Phrases: Attention mechanism; graph attention; deep learning; graph attention survey

ACM Reference Format:
John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. 2019. Attention Models in Graphs: A Survey. *ACM Trans. Knowl. Discov. Data.* 0, 0, Article 00 (August 2019), 25 pages. https://doi.org/0000001.0000001

1 INTRODUCTION

Data which can be naturally modeled as graphs are found in a wide variety of domains including the world-wide web [Albert et al. 1999], bioinformatics [Pei et al. 2005], neuroscience [Lee et al. 2017], chemoinformatics [Duvenaud et al. 2015], social networks [Backstrom and Leskovec 2011], scientific citation and collaboration [Liu et al. 2017], urban computing [Zheng et al. 2014], recommender

Authors' addresses: John Boaz Lee, WPI, MA, USA, jtlee@wpi.edu; Ryan A. Rossi, Adobe Research, San Jose, CA, USA, rrossi@adobe.com; Sungchul Kim, Adobe Research, San Jose, CA, USA, sukim@adobe.com; Nesreen K. Ahmed, Intel Labs, Santa Clara, CA, USA, nesreen.k.ahmed@intel.com; Eunyee Koh, Adobe Research, San Jose, CA, USA, eunyee@adobe.com.

1556-4681/2019/8-ART00 $15.00
https://doi.org/0000001.0000001

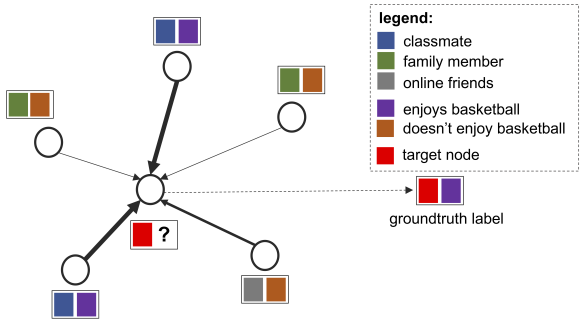

Fig. 1. Attention is used to assign importance to each type of neighbor. The link size denotes how much attention we want to apply to each neighbor. In this example, we see that by using attention to focus on a node's classmates, we can better predict the kind of activity the target is interested in. Best viewed with color.

systems [Deng et al. 2017], sensor networks [Aggarwal et al. 2017], epidemiology [Moslonka-Lefebvre et al. 2011], anomaly and fraud analysis [Akoglu et al. 2015], and ecology [Allesina et al. 2005]. For instance, interactions between users on a social network can be captured using a graph where the nodes represent users and links denote user interaction and/or friendship [Backstrom and Leskovec 2011]. On the other hand, in chemoinformatics, we can build molecular graphs by treating the atoms as nodes and the bonds between atoms as edges [Duvenaud et al. 2015].

A large body of work – which we broadly categorize as the field of graph mining [Aggarwal and Wang 2010] – has emerged that focuses on gaining insights from graph data. Many interesting and important problems have been studied in this area. These include graph classification [Duvenaud et al. 2015], link prediction [Sun et al. 2011], community detection [Girvan and Newman 2002], functional brain network discovery [Bai et al. 2017], node classification and clustering [Perozzi et al. 2014], and influence maximization [He and Kempe 2014] – with new problems constantly being proposed.

In the real-world, however, graphs can be both structurally large and complex [Ahmed et al. 2014; Leskovec and Faloutsos 2006] as well as noisy [He and Kempe 2014]. These pose a significant challenge to graph mining techniques, particularly in terms of performance [Ahmed et al. 2017].

Various techniques have been proposed to address this issue. For instance, the method proposed by [Wu et al. 2015] utilizes multiple views of the same graph to improve classification performance while [Zhang et al. 2016] leverages auxiliary non-graph data as side views with the same motivation. Another popular technique involves the identification of task-relevant or discriminative subgraphs [Shi et al. 2012; Zhu et al. 2012].

The fields of neuroscience and artificial intelligence have a long history of interactions. Many recent advances in artificial intelligence have been inspired by the study of the neural capabilities of the brain in humans and other animals [Hassabis et al. 2017]. One key example is the recent work in deep learning on *attention*. The key insight is that the brain does not learn by implementing a global optimization principle in a uniform neural network. The brain is modular and is comprised by distinct as well as interacting sub-modules performing key functions, such as language, memory, etc. [Marblestone et al. 2016]. Attention is an important brain-inspired function that has been utilized in many ways in recent deep learning methods allowing improvements in learning from limited data.

Recently, new methods have emerged to address the challenges facing relational (graph) learning problems by incorporating *attention* to improve graph deep learning methods. An attention mechanism aids a model by allowing it to "focus on the most relevant parts of the input to make

decisions" [Velickovic et al. 2018]. Attention was first introduced in the deep learning community to help models attend to important parts of the data [Bahdanau et al. 2015; Mnih et al. 2014]. The mechanism has been successfully adopted by models solving a variety of tasks. Attention was used by [Mnih et al. 2014] to take glimpses of relevant parts of an input image for image classification; on the other hand, [Xu et al. 2015] used attention to focus on important parts of an image for the image captioning task. Meanwhile [Bahdanau et al. 2015] utilized attention for the machine translation task by assigning weights which reflected the importance of different words in the input sentence when generating corresponding words in the output sentence. Finally, attention has also been used for both the image [Yang et al. 2016] as well as the natural language [Kumar et al. 2016] question answering tasks. However, most of the work involving attention has been done in the computer vision or natural language processing domains.

Applying attention to graph data comes with its own unique set of challenges. For instance, when generating captions from an image in the image captioning task [Xu et al. 2015], an object that may be particularly relevant could be situated in any part of the image. On the other hand, for graph-based task the structure of the graph has to be carefully considered. For instance, for the task of node classification on social networks [Kipf and Welling 2017] where the goal is to identify the label of a node, we typically attend to a node's neighborhood and ignore the nodes in the rest of the graph [Velickovic et al. 2018]. Figure 1 shows a motivating example of when attention can be useful in a graph setting.

There has been a growing interest in attention models for graphs recently and various techniques have been proposed [Choi et al. 2017; Feng et al. 2016; Han et al. 2018; Lee et al. 2018; Ma et al. 2017; Velickovic et al. 2018]. Although attention is defined in slightly different ways in all these papers, the competing approaches do share common ground in that attention is used to allow the model to focus on task-relevant parts of the graph. We discuss and define, more precisely, the main strategies used by these methods to apply attention to graphs in Section 3. In particular, the main advantages of using attention on graphs can be summarized as follows:

(1) Attention allows the model to avoid or ignore noisy parts of the graph, thus improving the signal-to-noise (SNR) ratio [Lee et al. 2018; Mnih et al. 2014].
(2) Attention allows the model to assign a relevance score to elements in the graph (for instance, the different neighbors in a node's neighborhood) to highlight elements with the most task-relevant information, further improving SNR [Velickovic et al. 2018].
(3) Attention also provides a way for us to make a model's results more interpretable [Choi et al. 2017; Velickovic et al. 2018]. For example, by analyzing a model's attention over different components in a medical ontology graph we can identify the main factors that contribute to a particular medical condition [Choi et al. 2017].

In this paper, we conduct a comprehensive and focused review of the literature on graph attention models. To the best of our knowledge, this is the first work on this topic. We introduce three different taxonomies to group existing work into intuitive categories. We then motivate our taxonomies through detailed examples and use each to survey competing approaches from a particular standpoint. In particular, we group the existing work by the kind of attention that is used, by the task (*e.g.*, node classification, graph classification, *etc.*), and by the problem setting (defined by the type of input graph and the primary problem output).

Our first taxonomy introduces the main strategies that have been proposed in the literature for applying attention in graphs. Our second taxonomy groups methods by application area; this shows the reader what problems have already been tackled while, perhaps more importantly, revealing important graph-based problems where attention models have yet to be applied. In previous work, different methods take different kinds of graphs (*e.g.*, homogeneous graphs, heterogeneous graphs,

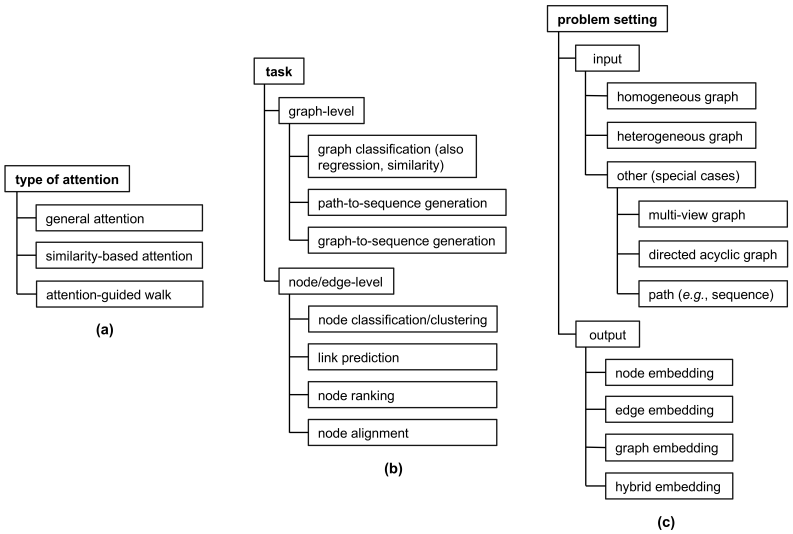

Fig. 2. Proposed taxonomies to group graph attention models based on (a) type of attention used, (b) task or problem, and (c) problem setting.

directed acyclic graphs, *etc.*) with different properties (*e.g.*, attributed, weighted or unweighted, directed or undirected) as input. These methods also differ by their outputs (*e.g.*, node embedding, link embedding, graph embedding). We use the third and final taxonomy, which is our main taxonomy, to provide a comprehensive survey of the field from the perspective of problem setting. We illustrate our proposed taxonomies in Figure 2.

We argue that these taxonomies allow readers to learn about different graph attention methods from different perspectives. In general, the discussion pertaining to each of the taxonomies can be considered separately. For instance, if the reader is interested in a particular problem, they may refer to the second taxonomy (Fig. 2b) to see what other methods have been proposed for this problem. If their problem domain requires them to work on a particular type of graph, they can then focus on the third taxonomy (Fig. 2c) to see what other methods have been proposed that match their problem setting. The first taxonomy (Fig. 2a) allows them to quickly identify the kind of strategies that have been employed previously to implement attention.

We use Graph Attention Networks [Velickovic et al. 2018] as an example to show the connections between the different taxonomies and also to highlight their individual focus. Graph Attention Networks are a variant of Graph Convolutional Networks [Kipf and Welling 2017] with attention. The method is given an attributed homogeneous graph (input) and it uses a series of "message passing" steps to learn individual node embeddings (output) for each of the nodes in the graph. Previously [Kipf and Welling 2017], at each step, a node's new embedding is calculated by taking a weighted sum of its neighbors' embeddings – with the weights being fixed. The method in [Velickovic et al. 2018] differs from previous work in that a general attention mechanism (type of attention) is used to assign attention coefficients to neighbor embeddings. This defines a probability distribution over the "messages" to be aggregated, allowing the model to prioritize more task-relevant messages. The method was evaluated on node classification (task). A classification layer took the final node embeddings as input and used these to predict each node's label, this allowed the model to be trained end-to-end.

Additionally, we also summarize the challenges that have yet to be addressed in the area of graph attention and provide promising directions for future work.

1.1 Main contributions

The main contributions of this work are as follows:

(1) We introduce three intuitive taxonomies for categorizing various graph attention models and survey existing methods using these taxonomies. To the best of our knowledge, this is the first survey on the important field of graph attention.
(2) We motivate each taxonomy by discussing and comparing different graph attention models from the taxonomy's perspective.
(3) We highlight the challenges that have yet to be addressed in the area of graph attention.
(4) We also provide suggestions on potential areas for future work in this emerging field.

1.2 Scope of this article

In this article, we focus on examining and categorizing various techniques that apply attention to graphs (we give a general definition of attention in Sec. 2). Most of these methods take graphs as input and solve some graph-based problem such as link prediction [Sun et al. 2011], graph classification/regression [Duvenaud et al. 2015], or node classification [Kipf and Welling 2017]. However, we also consider methods that apply attention to graphs although the graph is only one of several types of input to the problem.

We do *not* attempt to survey the vast field of general graph-based methods that do not explicitly apply attention, multiple work have already been done on this with each having a particular focus [Cai et al. 2018; Getoor and Diehl 2015; Jiang et al. 2013].

1.3 Organization of the survey

The rest of this survey is organized as follows. We start by introducing useful notations and definitions in Section 2. In Section 3, we introduce the main strategies used to implement attention in graphs (Fig. 2a). We then proceed to categorize existing work broadly by the type of task they solve (Fig. 2b) in Section 4. Sections 5 through 7 cover discussion of related work using our main taxonomy (Fig. 2c). We organized Sections 5-7 such that the methods in existing work are grouped by the main type of embedding they calculate (*e.g.*, node embedding, edge embedding, graph embedding, or hybrid embedding); these methods are then further divided by the type of graphs they support. In Section 8, we discuss challenges as well as interesting opportunities for future work. Finally, we conclude the survey in Section 9.

2 PROBLEM FORMULATION

In this section, we define the different types of graphs that appear in the discussion; we also introduce related notations.

Definition 1 (Homogeneous graph). *Let $G = (V, E)$ be a graph where V is the set of nodes (vertices) and E is the set of edges between the nodes in V.*

Further, let $\mathbf{A} = \left[A_{ij}\right]$ be the $n \times n$, for $n = |V|$, adjacency matrix of G where $A_{ij} = 1$ if there exists $(v_i, v_j) \in E$ and $A_{ij} = 0$ otherwise. When $\mathbf{A}$ is a binary matrix (*i.e.*, $A_{ij} \in \{0, 1\}$), we consider the graph to be *unweighted*. Note that $\mathbf{A}$ may also encode edge weights; given a weight $w \in \mathbb{R}$ for $(v_i, v_j) \in E$, $A_{ij} = w$. In this case, the graph is said to be *weighted*. Also, if $A_{ij} = A_{ji}$, for any $1 \leq i, j \leq n$, then the graph is *undirected*, otherwise it is *directed*.

Given an adjacency matrix $\mathbf{A}$, we can construct a right stochastic matrix $\mathbf{T}$ – also called the transition matrix – which is simply $\mathbf{A}$ with rows normalized to 1. Also, given a vertex $v \in V$, let Γ_v

Table 1. Summary of notation. Matrices are bold upright roman letters; vectors are bold lowercase letters.

G	(un)directed (attributed) graph
$\mathbf{A}$	adjacency matrix of the graph $G = (V, E)$
$\mathbf{T}$	stochastic transition matrix for G constructed by normalizing over the rows in $\mathbf{A}$
n, m	number of nodes and edges in the graph
k	number of embedding dimensions
d	node attribute dimension
Γ_i	neighbors of i defined using some neighborhood
$\Gamma_i^{(\ell)}$	ℓ-neighborhood $\Gamma_i^{(\ell)} = \{j \in V \mid \text{dist}(i, j) \leq \ell\}$
$\text{dist}(i, j)$	shortest distance between i and j
$\mathbf{X}$	$n \times d$ input attribute matrix
$\mathbf{x}$	a d-dimensional feature vector
x_i	the i-th element of $\mathbf{x}$
$\mathbf{Z}$	output node embedding matrix (or vector $\mathbf{z}$ in the case of graph embeddings)

be the set of vertices in node v's neighborhood (for instance, a popular definition for neighborhood is simply the one-hop neighborhood of v, or $\Gamma_v = \{w|(v, w) \in E\}$).

Definition 2 (Heterogeneous graph). *A heterogeneous graph is defined as $G = (V, E)$ consisting of a set of node objects V and a set of edges E connecting the nodes in V. A heterogeneous graph also has a* node type mapping function *$\theta : V \rightarrow \mathcal{T}_V$ and an* edge type mapping function *defined as $\xi : E \rightarrow \mathcal{T}_E$ where $\mathcal{T}_V$ and $\mathcal{T}_E$ denote the set of node types and edge types, respectively. The type of node i is denoted as $\theta(i)$ (which may be an author, a paper, or a conference in a heterogeneous bibliographic network) whereas the type of edge $e = (i, j) \in E$ is denoted as $\xi(i, j) = \xi(e)$ (*e.g.*, a co-authorship relationship, or a "published in" relationship).*

Note a heterogeneous graph is sometimes referred to as a typed network. Furthermore, a bipartite graph is a simple heterogeneous graph with two node types and a single edge type. A homogeneous graph is simply a special case of a heterogeneous graph where $|\mathcal{T}_V| = |\mathcal{T}_E| = 1$.

Definition 3 (Attributed graph). *Let $G = (V, E, \mathbf{X})$ be an attributed graph where V is a set of nodes, E is a set of edges, and $\mathbf{X}$ is an $n \times d$ matrix of* node input attributes *where each $\bar{\mathbf{x}}_i$ (or $\mathbf{X}_{i:}$) is a d-dimensional (row) vector of attribute values for node $v_i \in V$ and $\mathbf{x}_j$ (or $\mathbf{X}_{:j}$) is an n-dimensional vector corresponding to the jth attribute (column) of $\mathbf{X}$. Additionally, we may also have an $m \times d'$ matrix $\mathbf{X}'$ of* edge input attributes. *These may be real-valued or not.*

Definition 4 (Multi-view graph). *$G = (V, \{\mathbf{A}_1, \cdots, \mathbf{A}_c\})$ denotes a multi-view graph with c views. An $n \times n$ adjacency matrix $\mathbf{A}_i$ captures the relationship between different nodes from the perspective of the i-th view.*

Definition 5 (Directed acyclic graph (DAG)). *A directed graph with no cycles.*

The different "classes" of graphs defined in Definitions 1-3 can be directed or undirected as well as weighted or unweighted. Other classes of graphs arise from composing the distinct "graph classes" (Definition 1-3). For instance, it is straightforward to define an attributed heterogeneous network $G = (V, E, \mathcal{T}_V, \mathcal{T}_E, \mathbf{X})$.

Definition 6 (Path). *A path P of length L is defined as a sequence of* unique *indices $i_1, i_2, \ldots, i_{L+1}$ such that $(v_{i_t}, v_{i_{t+1}}) \in E$ for all $1 \leq t \leq L$. The length of a path is defined as the number of edges it contains.*

Unlike walks that allow loops, paths do not and therefore a walk is a path iff it has no loops (all nodes are unique). Note that a path is a special kind of graph with a very rigid structure where all the nodes have at most 2 neighbors. We now give a general, yet formal, definition of the notion of *graph attention* as follows:

Definition 7 (Graph Attention). *Given a target graph object (*e.g.*, node, edge, graph, etc),* v_0 *and a set of graph objects in* v_0*'s "neighborhood"* $\{v_1, \cdots, v_{|\Gamma_{v_0}|}\} \in \Gamma_{v_0}$ *(the neighborhood is task-specific and can mean the* ℓ*-hop neighborhood of a node or something more general). Attention is defined as a function* $f' : \{v_0\} \times \Gamma_{v_0} \rightarrow [0, 1]$ *that maps each of the object in* Γ_{v_0} *to a relevance score which tells us how much attention we want to give to a particular neighbor object. Furthermore, it is usually expected that* $\sum_{i=1}^{|\Gamma_{v_0}|} f'(v_0, v_i) = 1$.

A summary of the notation used throughout the manuscript is provided in Table 1.

3 TYPES OF GRAPH ATTENTION MECHANISM

In this section, we describe the three main types of attention that have been applied to graph data. While all three types of attention share the same purpose or intent, they differ in how attention is defined or implemented. In this section, we provide examples from the literature to illustrate how each type is implemented, in general.

Similarity-based attention which was proposed earlier in the NLP domain in works such as [Bahdanau et al. 2015; Graves et al. 2014] makes the assumption that "the most relevant keys are the most similar to the query" [Galassi et al. 2019]. As an example, in our context, when a model decides to attend to a node v^*'s neighborhood using similarity-based attention, it will prioritize nodes that have similar representations as v^*. In practice, similarity can be based on a similarity function like cosine similarity or more loosely calculated using a dot product or even vector subtraction.

On the other hand, *general* attention generalizes the idea of attention and doesn't explicitly require similarity [Luong et al. 2015]. For instance, Graph Attention Networks or GAT [Velickovic et al. 2018] uses this type of attention to identify v^*'s neighbors that hold task-relevant or discriminative information even if they do not share similar features/embeddings with v^*. This type of attention is suitable for scenarios where we don't equate similarity to relevance.

Various methods have used walks on graphs to sample from graphs [Grover and Leskovec 2016; Perozzi et al. 2014; Vishwanathan et al. 2010]. Methods based on *attention-guided* walks rely on an attention mechanism to bias the walk on a graph. This can be for different reasons, for instance [Abu-El-Haija et al. 2018] uses attention to bias the walk sampling process which in turn has an effect on the learned node embeddings. On the other hand, [Lee et al. 2018] uses an attention-guided walk to learn walk-based subgraph embeddings. The reader can refer to [Galassi et al. 2019] for an in-depth discussion on how attention coefficients are calculated by methods in the NLP domain.

In the discussion below, we now give some examples of how the three types of attention are implemented in practice. Recall from our general definition of attention in Def. 7 that we are given a target graph object (*e.g.*, node, edge, graph, *etc*) v_0 and a set of graph objects in v_0's "neighborhood" $\{v_1, \cdots, v_{|\Gamma_{v_0}|}\} \in \Gamma_{v_0}$. Attention is defined as a function $f' : \{v_0\} \times \Gamma_{v_0} \rightarrow [0, 1]$ that maps each of the objects in Γ_{v_0} to a relevance score.

3.1 General attention

Given the corresponding attributes/features $\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_{|\Gamma_{o^*}|}$ for $v_0, v_1, \cdots, v_{|\Gamma_{v_0}|}$, attention weights can be learned via:

$$\alpha_{0,j} = \frac{e_{0,j}}{\sum_{k \in \Gamma_{v_0}} e_{0,k}}$$

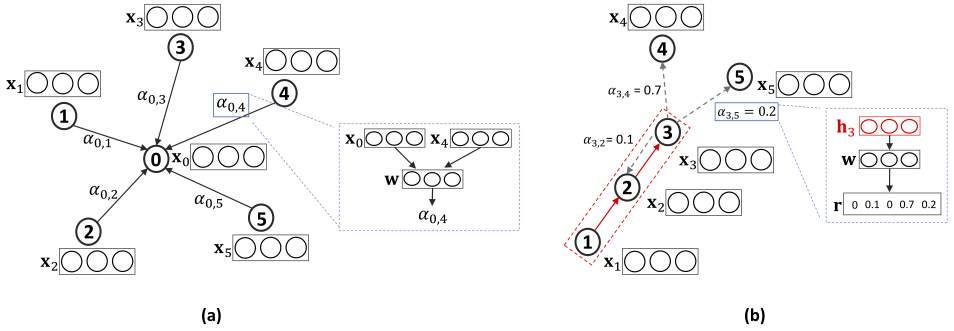

Fig. 3. (a) Given a target object v_0, we assign importance weights $\alpha_{0,i}$ to the objects i in our neighborhood. This can be done by learning a function that assigns importance by examining (possibly) the hidden embeddings of v_0 and v_i, $\mathbf{x}_0$ and $\mathbf{x}_i$. (b) The hidden embedding $\mathbf{h}_3$ represents information $(\mathbf{x}_1, \cdots, \mathbf{x}_3)$ we've integrated after a walk of length $L = 3$, we input this into a ranking function that determines the importance of various neighbor nodes and this is used to bias the next step. For both examples, we use $\mathbf{w}$ to represent the trainable parameters of the attention function.

where $e_{0,j}$ is node v_j's relevance to v_0. In practice, this is typically implemented using softmax with a trainable function learning v_j's relevance to v_0 by considering their attributes. The implementation in GAT [Velickovic et al. 2018] illustrates this:

$$\alpha_{0,j} = \frac{\exp\Big(\text{LeakyReLU}\Big(\mathbf{a}[\mathbf{W}\mathbf{x}_0||\mathbf{W}\mathbf{x}_j]\Big)\Big)}{\sum_{k\in\Gamma_{v_0}}\exp\Big(\text{LeakyReLU}\Big(\mathbf{a}[\mathbf{W}\mathbf{x}_0||\mathbf{W}\mathbf{x}_k]\Big)\Big)}$$

where $\mathbf{a}$ is a trainable attention vector, $\mathbf{W}$ is a trainable weight matrix mapping the input features to the hidden space, and $||$ represents concatenation. An illustration of this is shown in Fig. 3a.

3.2 Similarity-based attention

Again, given the corresponding attributes or features, the second type of attention can be learned similarly as above except for a key difference. We call this approach similarity-based attention as more attention is given to object's that share more similar hidden representations or features, this is also often referred to in the literature as alignment [Bahdanau et al. 2015]. To illustrate this, we use the definition given in AGNN [Thekumparampil et al. 2018]:

$$\alpha_{0,j} = \frac{\exp\Big(\beta \cdot \cos\Big(\mathbf{W}\mathbf{x}_0, \mathbf{W}\mathbf{x}_j\Big)\Big)}{\sum_{k\in\Gamma_{v_0}}\exp\Big(\beta \cdot \cos\Big(\mathbf{W}\mathbf{x}_0, \mathbf{W}\mathbf{x}_k\Big)\Big)}$$

where β is a trainable bias and "cos" represents cosine-similarity; like before, $\mathbf{W}$ is a trainable weight matrix to map inputs to the hidden space. Note that this is very similar to the above definition. The difference is that the model explicitly learns similar hidden embeddings for objects that are relevant to each other since attention is based on similarity or alignment.

3.3 Attention-guided walk

While the first two types of attention focuses on choosing relevant information to integrate into a target object's hidden representation, the third type of attention has a slightly different purpose. We use GAM [Lee et al. 2018] to illustrate this idea. GAM takes a series of steps on an input graph and encodes information from visited nodes using an RNN to construct a subgraph embedding. The

RNN's hidden state at a time t, $\mathbf{h}_t \in \mathbb{R}^h$ encodes information from the nodes that were previously visited by the walk from steps $1, \cdots, t$. Attention is then defined as a function $f' : \mathbb{R}^h \rightarrow \mathbb{R}^k$ that maps an input hidden vector $f'(\mathbf{h}_t) = \mathbf{r}_{t+1}$ to a k-dimensional rank vector that tells us which of the k types of nodes we should prioritize for our next step. The model will then prioritize neighboring nodes of a particular type for the next step. We illustrate this in Fig. 3b.

4 GRAPH TASKS

In general, graph-based methods can be broadly split into two groups by the kind of problem they tackle: node-level or graph-level. When mining richly structured datasets where links (or relationships) exist between different nodes (or objects), with node-level tasks we are typically interested in answering questions pertaining to a node (or a group of nodes). When performing node classification [Lu and Getoor 2003], for instance, we might ask "what does the structure of a node's neighborhood and its neighbors' information tell me about the node's label?"

When the task is at the graph-level, on the other hand, each object in the dataset is typically represented by a graph and we are now interested in answering questions pertaining to entire graphs. For instance, given a set of molecular graphs we might be interested in knowing what the structure of a molecule can tell us about its solubility or toxicity (graph classification/regression [Duvenaud et al. 2015]).

The different graph-based methods we survey in this paper solve a variety of graph problems by framing them as machine learning tasks. To solve the task at hand using a machine learning algorithm, we need to represent pertinent objects (*e.g.*, nodes, edges, subgraphs, or even entire graphs) as vectors. In Sec. 5-7, we discuss in more detail how these attention-based embeddings or vector representations can be learned using various approaches.

4.1 Node-level

Several attention-based methods have been proposed recently for the popular tasks of node classification and link prediction [Abu-El-Haija et al. 2018; Feng et al. 2016; Qu et al. 2017; Thekumparampil et al. 2018; Velickovic et al. 2018; Wang et al. 2018; Zhao et al. 2017; Zhou et al. 2019]. Although the methods differ from one another in the approach they take and the assumptions they make, they all share a common strategy. This entails the use of an attention mechanism to help learn node and/or edge embeddings which are then fed as input to a model for the task at hand (*i.e.*, node classification or link prediction).

Methods such as AttentionWalk [Abu-El-Haija et al. 2018] and GAKE [Feng et al. 2016] first learn node embeddings in an unsupervised manner. The learned node embeddings are then used as input for the downstream tasks of node classification and link prediction. Attention is used in [Abu-El-Haija et al. 2018] to allow the model to select an ideal "context distribution" which has an effect on the learned embeddings. On the other hand, GAKE uses attention to highlight objects in a node's "context neighborhood" which have more representative power which again has an effect on the node's learned embedding.

GAT [Velickovic et al. 2018] and AGNN [Thekumparampil et al. 2018] are variants of the popular graph convolutional network (GCN) introduced by [Kipf and Welling 2017]. These methods incorporate an attention mechanism to the "message-passing" step in a GCN to help them identify more task-relevant messages for each node. Unlike the previous methods, these models are end-to-end trainable.

On the other hand, [Qu et al. 2017] and [Wang et al. 2018] study node classification and/or link prediction in the context of multi-view graphs while [Zhou et al. 2019] considers a heterogeneous graph. When learning node embeddings in a multi-view graph, the different views of the graph have to be considered and attention is used to highlight the informative views for each node. This is

the strategy employed by both [Qu et al. 2017] and [Wang et al. 2018]. In a heterogeneous graph, on the other hand, different meta-paths can exist between a pair of nodes with each type of meta-path representing a distinct relationship between the nodes. HAHE [Zhou et al. 2019] used an attention mechanism to highlight more task-relevant meta-paths when learning node embeddings for node classification.

Apart from node classification and link prediction, an attention-based method has also been proposed for the task of top-K item recommendation on heterogeneous networks [Hu et al. 2018] which we can view as a kind of node ranking [Sun et al. 2009a]. Finally, MEgo2Vec [Zhang et al. 2018] is a method that tackles the problem of user or node alignment on social networks.

4.2 Graph-level

Multiple works have also studied graph-level tasks such as graph classification and graph regression. In this setting, an attention mechanism which helps the model to attend to more task-relevant parts of a graph is used to learn a graph embedding. Methods like EAGCN [Ryu et al. 2018], Modified-GAT [Shang et al. 2018], GAM [Lee et al. 2018], and Dipole [Ma et al. 2017] learn attention-guided graph embeddings for the more standard tasks of graph classification and regression. It is not hard to see how these can be applied to the related problem of graph similarity search [Zheng et al. 2013].

On the other hand, various work [Bahdanau et al. 2015; Luong et al. 2015; Xu et al. 2018; Zhou et al. 2018] have also considered the problem of generating sequences from input graph data. Most notably, in contrast to simpler sequence-to-sequence models, graph2seq [Xu et al. 2018] is a method that can output a sequence given a general graph. This can be applied to tasks where we are given a graph capturing various relationships between different entities and are asked to answer a text query by reasoning using the provided information.

Finally, GRAM [Choi et al. 2017] applied attention to a medical ontology graph to help learn attention-based embeddings for medical codes. While the problem they studied was the problem of classifying a patient record (described by certain medical codes), the novelty of their work was in applying attention on the ontology graph to improve model performance.

5 ATTENTION-BASED NODE/EDGE EMBEDDING

In this section – as well as the two succeeding ones – we introduce various graph attention models and categorize them by problem setting. To maintain uniformity across the sections (Sec. 5 - Sec. 7), we group the methods by the type of graphs they take as input. The first subsection covers the most common type of graphs (homogeneous graphs). The second subsection covers heterogeneous networks or graphs which have received a lot of attention recently in graph mining [Dong et al. 2017; Hu et al. 2018; Liu et al. 2018c; Shi et al. 2017; Zhou et al. 2019]. Finally, we use the last subsection to discuss methods that work on all other special types of graphs (*e.g.*, directed acyclic graphs, multi-view graphs, paths).

For easy reference, we show all of the surveyed methods in Table 2, taking care to highlight where they belong under each of the proposed taxonomies. We now begin by defining the traditional node embedding problem.

Definition 8 (Node Embedding). *Given a graph $G = (V, E)$ with V as the node set and E as the edge set, the objective of node embedding is to learn a function $f : V \rightarrow \mathbb{R}^k$ such that each node $i \in V$ is mapped to a k-dimensional vector $\mathbf{z}_i$ where $k \ll |V|$. The node embedding matrix is denoted as $\mathbf{Z}$.*

The learned node embeddings given as output can subsequently be used as input to mining and machine learning algorithms for a variety of tasks such as link prediction [Sun et al. 2011], node classification [Velickovic et al. 2018], community detection [Girvan and Newman 2002], and role

Table 2. Qualitative and quantitative comparison of graph-based attention models.

	Input					Output				Mechanism				Task							
Method	**Homogeneous graph**	**Heterogeneous graph**	**Multi-view graph**	**Directed acyclic graph**	**Path**	**Node embedding**	**Edge embedding**	**Graph embedding**	**Hybrid embedding**	**General attention**	**Similarity-based attention**	**Attention-guided walk**	**Other**	**Node/Link classification**	**Link prediction**	**Node ranking**	**Node alignment**	**Graph classification/reg.**	**Seq-to-seq generation**	**Graph-to-seq generation**	**Other**
AttentionWalks [Abu-El-Haija et al. 2018]	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗
GAKE [Feng et al. 2016]	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
GAT [Velickovic et al. 2018]	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
AGNN [Thekumparampil et al. 2018]	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
PRML [Zhao et al. 2017]	✓	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
HAHE [Zhou et al. 2019]	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
MVE [Qu et al. 2017]	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗
I^{2}MNE [Wang et al. 2018]	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
JointD/E + SATT [Han et al. 2018]	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗
MCRec [Hu et al. 2018]	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
SPE [Liu et al. 2018c]	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗
MEgo2Vec [Zhang et al. 2018]	✓	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Modified – GAT [Ryu et al. 2018]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
GAM [Lee et al. 2018]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
EAGCN [Shang et al. 2018]	✗	✗	✓	✗	✗	✓	✗	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
Dipole [Ma et al. 2017]	✗	✗	✗	✗	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
CCM [Zhou et al. 2018]	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
RNNSearch [Bahdanau et al. 2015]	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗
Att – NMT [Luong et al. 2015]	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗
graph2seq [Xu et al. 2018]	✓	✗	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
GRAM [Choi et al. 2017]	✗	✗	✗	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓

discovery [Rossi and Ahmed 2015]. We now define the problem of attention-based node embedding as follows:

Definition 9 (Attention-based Node Embedding). *Given a graph $G = (V, E)$ with V as the node set and E as the edge set, attention-based node embedding learns a function f' which defines a probability distribution over the elements in a target node i's neighborhood, and a function $f : V \rightarrow \mathbb{R}^k$ which maps each node $i \in V$ to an embedding vector $\mathbf{z}_i$. The function f utilizes f' to determine which elements in a node's neighborhood to attend to or prioritize when calculating the node's embedding.*

Above, we defined attention-based node embedding; note that attention-based edge embedding can also be defined similarly. Here, we use a single section to discuss both node and edge embeddings since there has not been a lot of work on attention-based edge embeddings and also because the two problems are quite similar [Cai et al. 2018]. We now categorize the different methods that calculate attention-based node/edge embeddings based on the type of graph they support.

5.1 Homogeneous graphs

Most work that calculate attention-based node embeddings focus on homogeneous graphs [Abu-El-Haija et al. 2018; Thekumparampil et al. 2018; Velickovic et al. 2018; Zhang et al. 2018; Zhao et al. 2017]. Also, all the methods assume the graph is attributed although [Abu-El-Haija et al. 2018] only needs node labels (in this case, the attribute size $d = 1$).

The order of the discussion here is as follows. We start by introducing methods that are more similar to traditional unsupervised node representation learning approaches (*e.g.*, DeepWalk, node2vec) [Abu-El-Haija et al. 2018; Feng et al. 2016]. We then introduce methods that extend graph convolutional networks by adding attention [Thekumparampil et al. 2018; Velickovic et al. 2018]. Finally, we introduce other approaches for learning attention-based node embeddings [Zhang et al. 2018; Zhao et al. 2017].

The method proposed by [Abu-El-Haija et al. 2018], called AttentionWalks, is most reminiscent of popular node embedding approaches such as DeepWalk and node2vec [Grover and Leskovec 2016; Perozzi et al. 2014] in that a random walk is used to calculate node contexts. Given a graph G with its corresponding transition matrix $\mathbf{T}$, and a context window size c, we can calculate the expected number of times walks started at each node visits other nodes via:

$$\mathbb{E}[\mathbf{D}|a_1, \cdots, a_c] = \mathbf{I}_n \sum_{i=1}^{c} a_i(\mathbf{T})^i.$$

Here $\mathbf{I}_n$ is the size-n identity matrix, a_i, for $1 \leq i \leq c$, are learnable attention weights, and $\mathbf{D}$ is the walk distribution matrix where D_{uv} encodes the number of times node u is expected to visit node v given that a walk is started from each node. In this scenario, attention is thus used to steer the walk towards a broader neighborhood or to restrict it within a narrower neighborhood (*e.g.*, when the weights are top-heavy). This solves the problem of having to do a grid-search to identify the best hyper-parameter c as studies have shown that this has a noticeable impact on performance [Perozzi et al. 2014] – note that for DeepWalk the weights are fixed at $a_i = 1 - \frac{i-1}{c}$.

Another attention-based method that is very similar in spirit to methods like DeepWalk, node2vec, and LINE [Grover and Leskovec 2016; Perozzi et al. 2014; Tang et al. 2015] in that it uses co-occurrence information to learn node embeddings is GAKE [Feng et al. 2016]. It is important to point out that GAKE builds a graph, commonly referred to as a knowledge graph, from knowledge triplets (h, t, r) where h and t are terms connected by a relationship r. Given three triplets (Jose, Tagalog, speaks), (Sato, Nihongo, speaks), and (Jose, Sinigang, eats) we can construct a simple heterogeneous graph with three types of nodes, namely {person, language, food}, and two types of relationships, namely {speaks, eats}. However we categorize GAKE as a homogeneous graph method as it doesn't seem to make a distinction between different kinds of relationships or node types (see metapath2vec [Dong et al. 2017] for a node embedding method that explicitly models the different kinds of relationships and nodes in a heterogeneous graph). Instead, the method takes a set of knowledge triplets and builds a directed graph by taking each triplet (h, t, r) and adding h and t as the head and tail nodes, respectively, while adding an edge from the head to the tail (they also add a reverse link).

The main difference between GAKE and methods such as DeepWalk, node2vec, and even AttentionWalks is that they include edges to define a node's context. Note that AttentionWalks follows previous approach [Perozzi et al. 2014] and define a node's context by other nodes within a context window. They define three different contexts to get related subjects (nodes or edges), formally they maximize the log-likelihood:

$$\sum_{s \in V \cup E} \sum_{c \in \Gamma_s} log \ \Pr(s \mid c)$$

where Γ_s is a set of nodes and/or edges defining the neighborhood context of s. To get the final node embedding for a given subject s in the graph, they use attention to obtain the final embedding $\mathbf{z}_s = \sum_{c \in \Gamma'_s} \alpha(c)\, \mathbf{z}'_c$ where Γ'_s is some neighborhood context for s, $\alpha(c)$ defines the attention weights for context object c and $\mathbf{z}'_c$ is the learned embedding for c.

On the other hand, methods like GAT [Velickovic et al. 2018] and AGNN [Thekumparampil et al. 2018] extend graph convolutional networks [Kipf and Welling 2017] by incorporating an explicit attention mechanism. Recall that a GCN is able to propagate information via an input graph's structure using the propagation rule:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})$$

where $\mathbf{H}^{(l)}$ indicates the learned embedding matrix at layer l of the GCN with $\mathbf{H}^{(0)} = \mathbf{X}$. Also, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_n$ is the adjacency matrix of an undirected graph $\mathbf{A}$ with added self loop. The matrix $\tilde{\mathbf{D}}$, on the other hand, is defined as the diagonal degree matrix of $\tilde{\mathbf{A}}$ so, in other words, $\tilde{D}_{i,i} = \sum_j \tilde{A}_{i,j}$. Hence, the term $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ computes a symmetric normalization (similar to the normalized graph Laplacian) for the graph defined by $\tilde{\mathbf{A}}$. Finally, $\mathbf{W}^{(l)}$ is the trainable weight-matrix for level l and $\sigma(\cdot)$ is a nonlinearity like ReLU, Sigmoid, or tanh.

A GCN works like an end-to-end differentiable version of the Weisfeiler-Lehman algorithm [Shervashidze et al. 2011] where each additional layer allows us to expand and integrate information from a larger neighborhood. However, because we use the term $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}$ in the propagation, we are essentially applying a weighted sum of the features of neighboring nodes normalized by their degrees. GAT and AGNN essentially introduce attention weights a_{uv} to determine how much attention we want to give to a neighboring node v from node u's perspective. The two methods differ primarily in the way attention is defined (we expound on this in Sec. 3). Furthermore, [Velickovic et al. 2018] introduce the concept of "multi-attention" which basically defines multiple attention heads (*e.g.*, weights) between a pair of objects in the graph.

PRML [Zhao et al. 2017] is another approach that learns edge embeddings but they use a different strategy to define attention. In [Zhao et al. 2017], a path-length threshold L is defined and a recurrent neural network [Gers et al. 2000] learns a path-based embedding for paths of length $1, \cdots, L$ between pairs of nodes u and v. Attention is defined in two ways. First, attention is used to identify important nodes along paths and this helps in calculating the intermediary path embeddings. Second, attention then assigns priority to paths that are more indicative of link formation and this is used to highlight important or task-relevant path embeddings when these are integrated to form the final edge embedding. We can think of this approach as an attention-based model that calculates the Katz betweenness score for pairs of nodes [Liben-Nowell and Kleinberg 2003].

MEgo2Vec is an attention-based method which solves the problem of user alignment across two social networks [Zhang et al. 2018]. To determine whether a user v_i (from the first network) and a user u_i (from the second network) refer to the same individual (for notational convenience, we use the same subscript i for the pair of nodes), the method first extracts two ego networks corresponding to v_i and u_i, respectively. An ego network is a vertex-induced subgraph consisting of a focal node v^* (in this case, v_i or u_i) and all nodes directly connected to v^*. A single matched ego network is then constructed by grouping the focal users into a pair (v_i, u_i) and matching their neighbors into pairs (v_j, u_j) – based on whether the corresponding neighbors are likely to be the same individual. Note that each user pair is represented by a single node in the matched network. An edge is drawn between two user pairs (v_j, u_j) and (v_k, u_k) if both v_j and v_k as well as u_j and u_k were connected in their respective ego networks. A node embedding is then learned for the focal

pair (v_i, u_i) by calculating the following embeddings:

$$\tilde{\mathbf{v}}_i = \sigma(\sum_{j \in \mathcal{N}_i^M} \alpha_{ji}\mathbf{v}_j), \;\; \tilde{\mathbf{u}}_i = \sigma(\sum_{j \in \mathcal{N}_i^M} \alpha_{ji}\mathbf{u}_j).$$

Here, σ is a non-linearity, $\mathcal{N}_i^M$ is the set of indices referring to the focal node's neighbor pairs in the matched network, $\mathbf{v}_j$ is the initial node embedding of a node v_j from the first network, $\mathbf{u}_j$ is the initial node embedding of a node u_j from the second network, and α_{ji} is the attention weight determining the importance of a neighbor pair (v_j, u_j). The attention weights are calculated to prioritize (1) rarer and thus more discriminative neighbor pairs, (2) neighbor pairs that are more similar as these are more likely to be the same individual, and (3) neighbor pairs that have the same semantic relationship with the focal pair across social networks. The final embedding of the focal pair (v_i, u_i) is simply $|\tilde{\mathbf{v}}_i - \tilde{\mathbf{u}}_i|$.

5.2 Heterogeneous graph

The only work, to the best of our knowledge, that has been proposed to calculate attention-guided node embeddings for heterogeneous graphs is HAHE [Zhou et al. 2019]. To learn a node embedding, the method first selects P meta-paths and uses each path p to create a distinct view of the data characterized by the adjacency matrix $\mathbf{A}^{(p)}$. $\mathbf{A}_{ij}^{(p)}$ is simply the count of meta-paths of type p between nodes i and j. The reason why we do not consider this to be a method for multi-view graph is because an arbitrary number of meta-paths P can be selected to form a multi-view graph with P views from a given heterogeneous graph.

P separate node embeddings are then learned for each node with respect to the P views. For each view p, a neighborhood aggregation scheme with attention is used to learn a node's embedding. This technique is almost identical to the ones proposed by GAT and AGNN and is used to prioritize more relevant neighborhood features. More specifically, a node i's p-th view-specific embedding is calculated via:

$$\mathbf{h}_i^{(p)} = \sigma(\sum_{j \in \mathcal{N}_i^{(p)}} \alpha_{ij}^{(p)}\mathbf{W}^{(p)}\mathbf{x}_j^{(p)}),$$

where σ is a non-linearity, $\mathcal{N}_i^{(p)}$ is node i's neighbors in view p, $\mathbf{W}^{(p)}$ is a trainable weight matrix, $\mathbf{x}_j^{(p)}$ is node j's feature vector, and $\alpha_{ij}^{(p)}$ is a trainable attention weight. A second attention mechanism is then utilized to assign weights to each of the P view-specific embeddings for a node so they can be aggregated into one final node embedding.

5.3 Other special cases

Multi-view graphs are graphs which are comprised of multiple edge sets which are defined over the same set of nodes [Qu et al. 2017; Wang et al. 2018]. Each view (represented by a distinct edge set) in the graph captures interactions between entities from a particular point of view. For instance, when studying brain networks, a multi-view graph with two complementary views can be used to represent the structural (view one) and functional (view two) connections between brain regions [Liu et al. 2018a].

MVE is a method for learning node embeddings on multi-view graphs [Qu et al. 2017]. Here, an attention mechanism is used to identify the informative views for each node. The embeddings from more informative views are then given more weight when calculating each node's final node embedding. In essence, the attention mechanism used by MVE is quite similar to the second attention mechanism of HAHE.

I^{2}MNE [Wang et al. 2018] is another attention-based method for learning node embeddings on multi-view graphs. Like MVE, the proposed method also uses attention to select more informative or task-relevant views from which to learn the final node embeddings. However, while learning view-specific node embeddings the method utilizes a second attention mechanism to identify intra-view neighbors whose features are more relevant to a particular node. This is similar to the self-attention proposed in GAT [Velickovic et al. 2018] and also used by HAHE [Zhou et al. 2019]. In fact, minus the preprocessing step in HAHE to convert a heterogeneous graph to a multi-view graph, the two methods (HAHE and I^{2}MNE) are very similar.

6 ATTENTION-BASED GRAPH EMBEDDING

Similarly, we begin the discussion by defining the traditional graph embedding problem.

Definition 10 (Graph Embedding). *Given a set of graphs, the objective of graph embedding is to learn a function $f : \mathcal{G} \rightarrow \mathbb{R}^k$ that maps an input graph $G \in \mathcal{G}$ to a low dimensional embedding vector $\mathbf{z}$ of length k; here $\mathcal{G}$ is the input space of graphs. Typically, we want to learn embeddings that group similar graphs (*e.g.*, graphs belonging to the same class) together.*

The learned graph embeddings can then be fed as input to machine learning/data mining algorithms to solve a variety of graph-level tasks such as graph classification [Lee et al. 2018], graph regression [Duvenaud et al. 2015], and graph-to-sequence generation [Xu et al. 2018]. We now define the problem of attention-based graph embedding as follows:

Definition 11 (Attention-based Graph Embedding). *Given the same inputs as above, we learn a function $f : \mathcal{G} \rightarrow \mathbb{R}^k$ that maps each input graph $G \in \mathcal{G}$ to an embedding vector $\mathbf{z}$ of length k. Additionally, we learn a second function f' that assigns "attention weights" to different objects (e.g., nodes, edges, or subgraphs) in a given graph to allow the model to prioritize more task-relevant parts of the graph when calculating its embedding.*

6.1 Homogeneous graphs

Several methods have been proposed for learning graph embeddings on homogeneous graphs [Lee et al. 2018; Ryu et al. 2018; Xu et al. 2018].

[Ryu et al. 2018] simply use the formulation of GAT [Velickovic et al. 2018] and make a few adjustments to the way attention is calculated. However, aside from the change in how attention weights are calculated, there isn't much difference between GAT and the method proposed in [Ryu et al. 2018]. To obtain the final graph embedding for the task of graph regression on molecular graphs, the proposed method adds fully connected layers after the final GAT layer to flatten the per-node outputs.

On the other hand, [Xu et al. 2018] propose graph2seq which solves the natural language question answering task. Given a set of facts (in the form of sentences) their approach is quite unique in that they convert the input into an attributed directed homogeneous graph. To obtain a "static" graph embedding that is used for the sequence generation task, they first learn node embeddings for each node in the graph. Node embeddings are formed by concatenating forward and backward representations for each node. These representations are derived by aggregating information from each node's forward (neighbors traversed using forward links) and backward (similarly, neighbors traversed using reverse links) neighborhoods, respectively. The static graph embedding is obtained by pooling the individual node embeddings or by simply aggregating them.

Attention, in the case of graph2seq however, is applied by also attending to the individual node embeddings during sequence generation. Since each node embedding $\mathbf{z}_i$ captures information in the region around node i (we can think of this as a neighborhood focused around i), attention

allows us to prioritize a particular part of the graph when generating each word in the output sentence (sequence). We can view this as a second "dynamic" time-dependent graph embedding which captures the intuition that different parts of the graph can be associated, primarily, with different concepts or words.

Finally, in a previous work [Lee et al. 2018], we proposed GAM which uses two types of attention to learn a graph embedding. The main idea is to use an attention-guided walk to sample relevant parts of the graph to form a subgraph embedding. In GAM, we took a walk of length L. Let $1 \leq i \leq L$ be the i-th node discovered in the walk and $\mathbf{x}_i$ the corresponding node attribute vector, an RNN is used to integrate the node attributes $(\mathbf{x}_1, \cdots, \mathbf{x}_L)$ to form a subgraph embedding $\mathbf{s}$ (the subgraph or region covered during the walk). During each step, an attention mechanism is used to determine which neighboring node is more relevant to allow the walk to cover more task-relevant parts of the graph. To get the final graph embedding, we deploy z "agents" to sample from various parts of the graph yielding embeddings $\{\mathbf{s}_1, \cdots, \mathbf{s}_z\}$. A second attention mechanism is then used to determine the relevance of the various subgraph embeddings before these are combined to form a graph embedding. In other words an attention mechanism is defined as a function $\alpha : \mathbb{R}^k \rightarrow [0, 1]$ which maps a given subgraph embedding to a relevance score. The graph embedding is thus defined as $\sum_{i=1}^{z} \alpha(\mathbf{s}_i)\, \mathbf{s}_i$.

6.2 Heterogeneous graphs

[Liu et al. 2018c] studied the problem of semantic user search on heterogeneous graphs. The goal is to predict the relevance (or proximity) of two users w.r.t. some desired semantic user relationship. The traditional approaches only considered meta-paths between users to solve this problem which [Liu et al. 2018c] considers to be insufficient for capturing the rich semantic relationship between users. To solve this problem, [Liu et al. 2018c] proposed SPE which calculates "subgraph-augmented path embeddings."

They start by defining an "s-node" embedding which contrary to what its name implies is an aggregate subgraph embedding. An s-node is defined by a pair of user nodes (i, j) and captures the subgraphs between users i and j. For instance, for a given user pair (i, j) with L different subgraph patterns between them, the s-node embedding is defined as $\mathbf{y}_{ij} = \sum_{l=1}^{L} \alpha_l \mathbf{x}_l$. Here, α_l is a trainable attention weight, and $\mathbf{x}_l$ is the initial subgraph embedding for subgraph l.

A subgraph-augmented path embedding can then be learned for two users u and v by forming a path comprised of s-nodes from u to v. In particular, a path with n s-nodes can be formed with the following user pairs $(u, s_1), (s_1, s_2), (s_2, s_3), \cdots, (s_n, v)$. A series of s-node embeddings for a path is fed as input into an RNN to learn a subgraph-augmented path embedding. This time attention is applied to differentiate between the contributions of different s-nodes along the path. In particular, the path embedding for a path between u and v with n s-nodes is defined as $\mathbf{z}_{uv} = \sum_{i=1}^{n} \alpha'_i \mathbf{h}_i$ where α'_i is a trainable attention weight and $\mathbf{h}_i$ is the i-th hidden vector of the RNN. Finally, to calculate proximity between users u and v, multiple path embeddings corresponding to different subgraph-augmented paths between users u and v are considered.

MCRec [Hu et al. 2018] is a method which studies the problem of item recommendation for users. Like SPE [Liu et al. 2018c], we are also interested in calculating the relevance/proximity between two nodes in a heterogeneous network. However, in this case, it is between a user node and an item node. MCRec employs a strategy that is similar to SPE by learning embeddings for meta-paths between a user u and an item i and uses attention to highlight meta-paths that capture semantically-important relationships between a user and an item. However, unlike [Liu et al. 2018c], the paths are node subgraph-augmented paths but simply meta-paths. In particular, MCRec [Hu

et al. 2018], SPE [Liu et al. 2018c], and PRML [Zhao et al. 2017] all examine the paths, in general, between two nodes to for link prediction or recommendation/ranking.

6.3 Other special cases

Attention was originally studied in the Computer Vision and NLP domains and various RNN models using attention on sequence-based tasks were proposed [Bahdanau et al. 2015; Luong et al. 2015]. Since sequences are technically no different from paths, we introduce some notable attention models under this setting.

A sequence (*e.g.*, a sentence) of length L can be represented as a directed attributed graph of length L – the i-th attribute vector $\mathbf{x}_i$ of node i is then a representation of the i-th component in the sequence (a one-hot word embedding or a word2vec embedding, for instance). In the proposed methods [Bahdanau et al. 2015; Luong et al. 2015; Ma et al. 2017], the node attribute vectors $\{\mathbf{x}_i, \cdots, \mathbf{x}_L\}$ are fed one after the other into an RNN. Recall that in the simplest case, a recurrent neural network calculates a hidden embedding $\mathbf{h}_i$ for each input i via the rule

$$\mathbf{h}_i = \sigma(\mathbf{W}_h \mathbf{x}_i + \mathbf{U}_h \mathbf{h}_{i-1}) + \mathbf{b}_h,$$

where $\mathbf{W}_h$ and $\mathbf{U}_h$ are trainable weight matrices for the input $\mathbf{x}_i$ and the previous hidden embedding $\mathbf{h}_{i-1}$, respectively; $\mathbf{b}_h$ is a bias vector and $\mathbf{h}_0$ is usually initialized to the zero vector.

In this case, we can think of $\mathbf{h}_i$ as node i's node embedding which integrated information from the sub-path encompassing nodes $1, \cdots, i$. Attention can then be applied on node embeddings to generate an attention-guided graph embedding. In particular, the method proposed in [Bahdanau et al. 2015] called RNNSearch defined the graph embedding as $\mathbf{z} = \sum_{i=1}^{L} \alpha_i \mathbf{h}_i$ where attention is assigned to each hidden embedding $\mathbf{h}_i$ depending on its relevance to the task.

When the length of the path L is large, however, attending over all the hidden embeddings as in RNNSearch may not yield the best results. [Luong et al. 2015] proposed to use two attention mechanisms, the first is similar in spirit to that of [Bahdanau et al. 2015]. However, the second attention allows the model to select a local point within the input and focus attention there. More concretely, depending on the needs of the task, the second attention mechanism outputs a position $1 \leq p \leq L$ and the graph embedding is constructed from the hidden node embeddings $\mathbf{h}_{p-D}, \cdots, \mathbf{h}_{p+D}$ where D is an empirically selected attention window size.

Finally, Dipole [Ma et al. 2017] applied an attention model similar to that of [Bahdanau et al. 2015] to sequential medical data. They used it to diagnose or predict medical conditions from the sequence. In practice, both [Ma et al. 2017] and [Bahdanau et al. 2015] used a bi-directional model which processes an input sequence using two RNNs, one taking the input sequence in its original order and another which takes the input sequence in reverse order.

EAGCN [Shang et al. 2018] was used to study the task of graph regression on molecular graphs. The method represents a molecule as a multi-view graph where each view characterizes the relationship between atoms w.r.t. a particular chemical bond property. The method then uses a technique similar to the attention-based neighbor aggregation scheme used in [Wang et al. 2018; Zhou et al. 2019] to learn multiple node embeddings (from the different views) for each node. The final graph embedding is then a concatenation/sum of the attention-guided node embeddings. Apart from its similarity to the multi-view graph methods mentioned above, EAGCN is like a version of Modified - GAT that operates on multi-view graphs.

7 ATTENTION-BASED HYBRID EMBEDDING

In this section we discuss methods that apply attention on graph data. However, the calculated embeddings are "hybrid" since the methods here also take data of other modalities (*e.g.*, text) and the learned embedding is a combination of all inputs.

7.1 Homogeneous graphs

CCM [Zhou et al. 2018] tackles the problem of generating a response to an input statement. However, what sets CCM apart from other text-based sequence-to-sequence methods is that the model takes homogeneous graphs as additional inputs. The graphs are used to encode knowledge about the world and, by considering them, the model is able to generate responses that are "knowledge-aware." Each graph i is defined by a set of knowledge triplets $T_i = \{(h_1^{(i)}, t_1^{(i)}, r_1^{(i)}), \cdots, (h_{n_i}^{(i)}, t_{n_i}^{(i)}, r_{n_i}^{(i)})\}$, where $h_j^{(i)}$ and $t_j^{(i)}$ are nodes and $r_j^{(i)}$ denotes a directed edge from $h_j^{(i)}$ to $t_j^{(i)}$.

When the model is given an input statement, it first retrieves a set of graphs related to the words in the input statement. For each of these graphs i, it learns a graph embedding $\mathbf{z}_i$ as follows:

$$\mathbf{z}_i = \sum_{j=1}^{|T_i|} \alpha_j^{(i)} [\mathbf{h}_j^{(i)}; \mathbf{t}_j^{(i)}]$$

where $\mathbf{h}_j^{(i)}$ and $\mathbf{t}_j^{(i)}$ are the embedding or attribute vectors of nodes $h_j^{(i)}$ and $t_j^{(i)}$, respectively, and $\alpha_j^{(i)}$ is an attention coefficient calculated by taking into account the pair of nodes as well as the relationship between them.

Finally, at each step of the response generation process, it uses attention to focus on important graph embeddings $\mathbf{z}_i$. Furthermore, The model uses attention to identify the important pairs of nodes (or triplets) within each graph. Like GAKE, we consider CCM to be a homogeneous graph method since it doesn't make an explicit distinction between different types of nodes and relationships.

JointD/E + SATT [Han et al. 2018] is another work that applies attention on graphs constructed from knowledge triplets, similar to [Feng et al. 2016; Zhou et al. 2018]. However, they also use a large text corpus that may contain references to the different terms in the knowledge graph implying certain relationships. They introduce a method that uses attention to learn a joint embedding from graph and text data for the task of knowledge graph link prediction.

Under certain settings (brain network construction, for instance) the datasets tend to be quite small and noisy [Zhang et al. 2016]. The method proposed by [Zhang et al. 2016] uses side information to regularize the graph construction process to highlight more discriminative patterns – which is useful when the output graphs are used for graph classification. Exploring the possibility of adding attention in this setting is interesting.

7.2 Heterogeneous graphs

While previous work [Han et al. 2018; Zhou et al. 2018] have applied attention on knowledge graphs – which can be considered heterogeneous graphs – these models do not distinguish between the different types of links and nodes explicitly. To the best of our knowledge, there currently does not exist any work that has considered this setting.

7.3 Other special cases

GRAM [Choi et al. 2017] is a graph-based attention model for doing classification/regression on clinical data. Clinical records can usually be described by clinical codes $c_1, \cdots, c_{|C|} \in C$ in a vocabulary C. GRAM constructs a DAG whose leaves are the codes $c_1, \cdots, c_{|C|}$ while the ancestor nodes are more general medical concepts. [Choi et al. 2017] uses an attention mechanism to learn a k-dimensional final embedding of each leaf node i (medical concept) via:

$$\mathbf{g}_i = \sum_{j \in \mathcal{A}(i)} \alpha_{i,j}\, \mathbf{e}_j.$$

where $\mathcal{A}(i)$ denotes the set comprised of i and all its ancestors and $\mathbf{e}_j$ is the k-dimensional basic embedding of the node j. The use of attention allows the model to refer to more informative or relevant general concepts when a concept in C is less helpful for medical diagnosis (*e.g.*, it is a rare concept). Since a patient's clinical visit record is represented as a binary vector $\mathbf{x} \in \{0,1\}^{|C|}$ indicating which clinical codes were present for a particular visit the learned embeddings $\mathbf{g}_1, \cdots, \mathbf{g}_{|C|}$ can be stacked to form a $k \times |C|$ embedding matrix to embed $\mathbf{x}$; this can then be inputted into a predictive model for medical diagnosis. Note that the problem of medical diagnosis is a classical supervised learning task which takes clinical code feature vectors but GRAM applies attention on a medical concept DAG for the purpose of embedding the given feature vectors.

8 DISCUSSION AND CHALLENGES

In this section we discuss additional issues and highlight important challenges for future work.

8.1 Graph generation

Several recent work [Liu et al. 2018b; You et al. 2018a,b] have also studied the problem of graph generation in an adversarial setting [Goodfellow et al. 2014]. Many important problems in drug discovery and material science are based on the principle of designing molecular structures that exhibit certain desired properties. With this in mind, [You et al. 2018a] proposed a method for generating structurally-valid molecular graphs which are also optimized based on the desired properties. Meanwhile, [Liu et al. 2018b] explored the problem of generating parse trees that represent syntactically-valid sequences (*e.g.*, SQL queries).

The graph generation problem is a challenging problem since we are dealing with a large output space of possible graphs and have to bias the generation process towards graphs that optimize some desired objective. One interesting research direction that can be considered is the exploration of novel attention mechanisms that can aid in the process of graph generation. For instance, GCPN [You et al. 2018a] uses an iterative process to gradually add edges and substructures during the graph generation process. It would be interesting to look at attention mechanisms that would allow us to pinpoint crucial steps made in the past to help inform the decision making at the current step.

8.2 Scalability of graph attention models

The majority of methods [Bahdanau et al. 2015; Ma et al. 2017; Zhou et al. 2018] that calculate an attention-based graph embedding work for relatively small graphs and may have trouble scaling effectively to larger graphs. Methods like [Lee et al. 2018; Ryu et al. 2018; Shang et al. 2018] may be able to work on larger graphs but these have their individual shortcomings.

For instance, GAM [Lee et al. 2018] uses attention-guided walks to sample from a graph. When dealing with large graphs, these questions naturally arise: (1) Do we need longer walks and can an RNN handle these? (2) Can walks effectively capture all task-relevant information from a graph, especially if there are complex structural patterns involved? Methods like [Ryu et al. 2018; Shang et al. 2018] that apply attention to a GCN architecture seem like a step in the right direction. We can think of them as end-to-end differentiable versions of the Weisfeiler-Lehman algorithm [Duvenaud et al. 2015; Shervashidze et al. 2011] with attention and these can be trained fairly quickly especially if we train stochastically [Chen et al. 2018]. However, recent work has shown that various graph neural network approaches including GCNs cannot distinguish between certain types of graph structures [Xu et al. 2019].

There is a need to evaluate existing graph attention models on a variety of large real-world graphs to test their scalability as well as their effectiveness on these datasets. Furthermore, it would be useful to explore other ways of applying attention not simply to boost model accuracy but also to improve the model runtime.

8.3 Hierarchical graph representation learning

A large number of different graph neural networks [Duvenaud et al. 2015; Hamilton et al. 2017; Kipf and Welling 2017; Velickovic et al. 2018] have been proposed recently. By and large, these methods rely on a series of neural "message-passing" steps to aggregate information from a node's neighborhood to learn node representations. The embeddings learned by these architectures are "flat" since information is only propagated through the edges of the graph and aggregation is not done in a hierarchical manner.

Learning a hierarchical set of representations may be particularly beneficial for tasks such as graph classification which aims to learn the label of an entire graph. [Ying et al. 2018] introduced a method which can learn hierarchical representations of graphs by using a technique called differentiable graph pooling. Graph pooling, when applied repeatedly, allows the method to gradually coarsen the input graph by grouping related nodes together.

This technique is useful as it allows us to learn hierarchical representations for graphs which is akin to how a convolutional neural network is able to learn hierarchical representations for images. It may be useful to apply attention to the process of graph pooling. For instance, one question we can ask is this: can attention be used to help identify the right cluster for each node during the graph pooling step?

8.4 Attention-based methods for heterogeneous graphs

The study of heterogeneous graphs, also called heterogeneous information networks, has become quite popular in recent years with many papers published in the area [Dong et al. 2017; Lee and Adorna 2012; Sun et al. 2011, 2009a,b]. A Heterogeneous graph usually gives us a much richer semantic view of the underlying data, when leveraged, this information can benefit solutions for problems such as image search [Tsai et al. 2014], item recommendation [Hu et al. 2018], and semantic user search [Liu et al. 2018c]. Please refer to [Shi et al. 2017] for a recent comprehensive survey on heterogeneous graph-based methods.

While methods like GAKE, CCM, JointD/E+SATT are designed for knowledge graphs which can be considered as a special type of heterogeneous graph, they do not explicitly differentiate between the different kinds of links and nodes. Some recent work [Hu et al. 2018; Liu et al. 2018c; Zhou et al. 2019] have studied attention-based models for heterogeneous graphs. However, given the richness of problems that can arise from the study of heterogeneous networks, there is a need to study novel attention mechanisms that can be applied to these networks.

8.5 Inductive graph learning

Some recent work have begun to investigate methods for inductive learning on graphs [Guo et al. 2018; Hamilton et al. 2017; Rossi et al. 2018] which is different from transductive learning. The former is more useful than the latter as it can generalize to yet unseen data.

This setting is important as it allows graph-based methods to handle many real-world cases where nodes and edges in graphs can appear dynamically. It also allows us to learn general patterns in one graph dataset that can be useful in another dataset. This is similar to how transfer learning is used to train a model on one text corpora for application on another text dataset [Dai et al. 2007]; another example is when a model is trained to recognize general shapes in a large image dataset and this information is used on smaller dataset with much fewer samples [Oquab et al. 2014]. Another interesting example of transfer learning is shown by [Zhu et al. 2011] where the transfer is done across data from different domains (*e.g.*, text to images).

An interesting direction for future study is looking at attention-based inductive learning techniques that can be used to identify relevant graph patterns that are generalizable to other graph

datasets. Looking further, we can also explore attention-based techniques that do cross-domain or heterogeneous transfer learning [Zhu et al. 2011].

While the authors of methods like GAT [Velickovic et al. 2018] have conducted an initial study of inductive learning on a small graph dataset, we believe more focused experiments should be done on graph-based inductive learning taking into account a large set of datasets and settings.

8.6 Model interpretability

One of the advantages of attention is its ability to endow a model with a certain degree of interpretability and, indeed, this is the reason why attention-based models are favored in task domains where interpretability is of particular importance [Brown et al. 2018; Choi et al. 2017, 2016]. Among the work we have surveyed, some simply tout the added predictive ability which attention brings while others like HAHE [Zhou et al. 2019] and GRAM [Choi et al. 2017] have conducted some analysis of the effects and behavior of attention in their work.

We believe that there is value in studying graph-based attention in more detail and with a focus on interpretability. For instance, when studying an attention-based method for learning node embeddings it would be interesting to analyze the kind of information each type of node prioritizes and give an explanation as to why this is the case. Or, in the case of graph classification of molecular graphs [Duvenaud et al. 2015], it would be interesting to see what important molecular substructures are discovered by attention and how much these overlap with conventional knowledge.

8.7 Attention-guided attributed walk

Recently, Ahmed *et al.* [Ahmed et al. 2018] proposed the notion of *attributed walk* and showed that it can be used to generalize graph-based deep learning and embedding methods making them more powerful and able to learn more appropriate embeddings. This is achieved by replacing the notion of random walk (based on node ids) used in graph-based deep learning and embedding methods with the more appropriate and powerful notion of *attributed walk*. More formally,

Definition 12 (Attributed walk). *Let $\mathbf{x}_i$ be a k-dimensional vector for node v_i. An* attributed walk *S of length L is defined as a sequence of adjacent node types*

$$\phi(\mathbf{x}_{i_1}), \phi(\mathbf{x}_{i_2}), \ldots, \phi(\mathbf{x}_{i_{L+1}}) \tag{1}$$

associated with a sequence of indices $i_1, i_2, \ldots, i_{L+1}$ such that $(v_{i_t}, v_{i_{t+1}}) \in E$ for all $1 \leq t \leq L$ and $\phi : \mathbf{x} \rightarrow y$ is a function that maps the input vector $\mathbf{x}$ of a node to a corresponding type $\phi(\mathbf{x})$.

The type sequence $\phi(\mathbf{x}_{i_1}), \phi(\mathbf{x}_{i_2}), \ldots, \phi(\mathbf{x}_{i_{L+1}})$ is the node types that occur during a walk (as opposed to the node ids). It was shown that the original deep graph learning models that use traditional random walks are recovered as a special case of the attributed walk framework when the number of unique types $t \rightarrow n$ [Ahmed et al. 2018]. It should be noted that the node types here do not necessarily refer to node types in heterogeneous graphs but can be calculated for nodes in a homogeneous graph from their local structure.

Attention-based methods that leverage random walks (based on node ids) may also benefit from the notion of *attributed walks* (typed walks) proposed by Ahmed *et al.*[Ahmed et al. 2018].

9 CONCLUSION

In this work, we conducted a comprehensive and focused survey of the literature on the important field of graph attention models. To the best of our knowledge, this is the first work of this kind. We introduced three intuitive taxonomies to group existing work. These are based on problem setting, the type of attention mechanism used, and the task. We motivated our taxonomies through detailed examples and used each to survey competing approaches from the taxonomy's unique standpoint.

We also highlighted several challenges in the area and provided discussion on possible directions for future work.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Rami Al-Rfou, and Alex Alemi. 2018. Watch Your Step: Learning Node Embeddings via Graph Attention. In *Proc. of NeurIPS*. 9198–9208.

Charu C. Aggarwal, Amotz Bar-Noy, and Simon Shamoun. 2017. On sensor selection in linked information networks. *Computer Networks* 126, C (2017), 100–113.

Charu C. Aggarwal and Haixun Wang. 2010. *Graph Data Management and Mining: A Survey of Algorithms and Applications*. Advances in Database Systems, Vol. 40. Springer.

Nesreen K. Ahmed, Nick Duffield, Theodore L. Willke, and Ryan A. Rossi. 2017. On Sampling from Massive Graph Streams. In *Proc. of VLDB*. 1430–1441.

Nesreen K. Ahmed, Jennifer Neville, and Ramana Kompella. 2014. Network Sampling: From Static to Streaming Graphs. *ACM TKDD* 8, 2 (2014), 1–56.

Nesreen K Ahmed, Ryan Rossi, John Boaz Lee, Xiangnan Kong, Theodore L Willke, Rong Zhou, and Hoda Eldardiry. 2018. Learning Role-based Graph Embeddings. In *arXiv:1802.02896*.

Leman Akoglu, Hanghang Tong, and Danai Koutra. 2015. Graph based anomaly detection and description: a survey. *DMKD* 29, 3 (2015), 626–688.

Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. 1999. Internet: Diameter of the World-Wide Web. *Nature* 401, 1 (1999), 130–131.

Stefano Allesina, Antonio Bodini, and Cristina Bondavalli. 2005. Ecological subsystems via graph theory: the role of strongly connected components. *Oikos* 110, 1 (2005), 164–176.

Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In *Proc. of WSDM*. 635–644.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proc. of ICLR*. 1–15.

Zilong Bai, Peter B. Walker, Anna E. Tschiffely, Fei Wang, and Ian Davidson. 2017. Unsupervised Network Discovery for Brain Imaging Data. In *Proc. of KDD*. 55–64.

Andy Brown, Aaron Tuor, Brian Hutchinson, and Nicole Nichols. 2018. Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection. In *rXiv:1803.04967v1*.

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. 2018. A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications. *ACM TKDE* (2018), 1–20.

Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *arXiv:1710.10568v3*.

Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. 2017. GRAM: Graph-based Attention Model for Healthcare Representation Learning. In *Proc. of KDD*. 787–795.

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. RETAIN: An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism. In *Proc. of NeurIPS*. 3504–3512.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Transferring Naive Bayes Classifiers for Text Classification. In *Proc. of AAAI*. 540–545.

Shuiguang Deng, Longtao Huang, Guandong Xu, Xindong Wu, and Zhaohui Wu. 2017. On Deep Learning for Trust-Aware Recommendations in Social Networks. *IEEE TNNLS* 28, 5 (2017), 1164–1177.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proc. of KDD*. 135–144.

David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gomez-Bombarelli, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional Networks on Graphs for Learning Molecular Fingerprints. In *Proc. of NeurIPS*. 2224–2232.

Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. 2016. GAKE: Graph Aware Knowledge Embedding. In *Proc. of COLING*. 641–651.

Andrea Galassi, Marco Lippi, and Paolo Torroni. 2019. Attention, please! A Critical Review of Neural Attention Models in Natural Language Processing. In *arXiv:1902.02181v1*.

Felix A. Gers, Jurgen Schmidhuber, and Fred A. Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation* 12, 10 (2000), 1–20.

Lise Getoor and Christopher P. Diehl. 2015. Link Mining: A Survey. *SIGKDD Explor. Newsl.* 7, 2 (2015), 3–12.

M. Girvan and M. E. J. Newman. 2002. Community structure in social and biological networks. *PNAS* 99, 12 (2002), 7821–7826.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proc. of NeurIPS*. 2672–2680.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. In *arXiv:1410.5401*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proc. of KDD*. 855–864.

Junliang Guo, Linli Xu, and Enhong Cheng. 2018. SPINE: Structural Identity Preserved Inductive Network Embedding. In *arXiv:1802.03984v1*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proc. of NeurIPS*. 1–11.

Xu Han, Zhiyuan Liu, and Maosong Sun. 2018. Neural Knowledge Acquisition via Mutual Attention Between Knowledge Graph and Text. In *Proc. of AAAI*. 1–8.

Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. 2017. Neuroscience-inspired artificial intelligence. *Neuron* 95, 2 (2017), 245–258.

Xinran He and David Kempe. 2014. Stability of influence maximization. In *Proc. of KDD*. 1256–1265.

Binbin Hu, Chuan Shi, Wayne Xin Zhao, and Philip S. Yu. 2018. Leveraging Meta-path based Context for Top-N Recommendation with A Neural Co-Attention Model. In *Proc. of KDD*. 1531–1540.

Chuntao Jiang, Frans Coenen, and Michele Zito. 2013. A survey of frequent subgraph mining algorithms. *The Know. Eng. Review* 28, 1 (2013), 75–105.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proc. of ICLR*. 1–14.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask Me Anything: Dynamic Memory Networks for Natural Language Processing. In *Proc. of ICML*. 2397–2406.

John Boaz Lee and Henry Adorna. 2012. Link Prediction in a Modified Heterogeneous Bibliographic Network. In *Proc. of ASONAM*. 442–449.

John Boaz Lee, Xiangnan Kong, Yihan Bao, and Constance Moore. 2017. Identifying Deep Contrasting Networks from Time Series Data: Application to Brain Network Analysis. In *Proc. of SDM*. 543–551.

John Boaz Lee, Ryan Rossi, and Xiangnan Kong. 2018. Graph Classification using Structural Attention. In *Proc. of KDD*. 1–9.

Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proc. of KDD*. 631–636.

David Liben-Nowell and Jon Kleinberg. 2003. The link prediction problem for social networks. In *Proc. of CIKM*. 556–559.

Qi Liu, Biao Xiang, Nicholas Jing Yuan, Enhong Chen, Hui Xiong, Yi Zheng, and Yu Yang. 2017. An Influence Propagation View of PageRank. *ACM TKDD* 11, 3 (2017), 30:1–30:30.

Xinyue Liu, Xiangnan Kong, Lei Liu, and Kuorong Chiang. 2018b. TreeGAN: Syntax-Aware Sequence Generation with Generative Adversarial Networks. In *Proc. of ICDM*. 1140–1145.

Ye Liu, Lifang He, Bokai Cao, Philip S. Yu, Ann B. Ragin, and Alex D. Leow. 2018a. Multi-View Multi-Graph Embedding for Brain Network Clustering Analysis. In *Proc. of AAAI*. 117–124.

Zemin Liu, Vincent W. Zheng, Zhou Zhao, Hongxia Yang, Kevin Chen-Chuan Chang, Minghui Wu, and Jing Ying. 2018c. Subgraph-augmented Path Embedding for Semantic User Search on Heterogeneous Social Network. In *Proc. of WWW*. 1613–1622.

Qing Lu and Lise Getoor. 2003. Link-based Classification. In *Proc. of ICML*. 496–503.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proc. of EMNLP*. 1412–1421.

Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. 2017. Dipole: Diagnosis Prediction in Healthcare via Attention-based Bidirectional Recurrent Neural Networks. In *Proc. of KDD*. 1903–1911.

Adam H Marblestone, Greg Wayne, and Konrad P Kording. 2016. Toward an integration of deep learning and neuroscience. *Frontiers in computational neuroscience* 10 (2016), 94.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent Models of Visual Attention. In *Proc. of NeurIPS*. 2204–2212.

Mathieu Moslonka-Lefebvre, Ann Finley, Ilaria Dorigatti, Katharina Dehnen-Schmutz, Tom Harwood, Michael J. Jeger, Xiangming Xu, Ottmar Holdenrieder, and Marco Pautasso. 2011. Networks in Plant Epidemiology: From Genes to Landscapes, Countries, and Continents. *Phytopathology* 101, 4 (2011), 392–403.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In *Proc. of CVPR*. 1717–1724.

Jian Pei, Daxin Jiang, and Aidong Zhang. 2005. On Mining Cross-Graph Quasi-Cliques. In *Proc. of KDD*. 228–238.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proc. of KDD*. 701–710.

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 *Proc. of CIKM*. 1767–1776.

Ryan A. Rossi and Nesreen K. Ahmed. 2015. Role Discovery in Networks. *ACM TKDE* 27, 4 (2015), 1112–1131.

Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2018. Deep Inductive Network Representation Learning. In *Proceedings of the 3rd International Workshop on Learning Representations for Big Networks (WWW BigNet)*. 8.

Seongok Ryu, Jaechang Lim, and Woo Youn Kim. 2018. Deeply learning molecular structure-property relationships using graph attention neural network. In *arXiv:1805.10988v2*.

Chao Shang, Qinqing Liu, Ko-Shin Chen, Jiangwen Sun, Jin Lu, Jinfeng Yi, and Jinbo Bi. 2018. Edge Attention-based Multi-Relational Graph Convolutional Networks. In *arXiv:1802.04944v1*.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *JMLR* (2011), 2539–2561.

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and Philip S. Yu. 2017. A Survey of Heterogeneous Information Network Analysis. *IEEE TKDE* 29, 1 (2017), 17–37.

Xiaoxiao Shi, Xiangnan Kong, and Philip S. Yu. 2012. Transfer Significant Subgraphs across Graph Databases. In *Proc. of SDM*. 552–563.

Yizhou Sun, Rick Barber, Manish Gupta, Charu C. Aggarwal, and Jiawei Han. 2011. Co-author Relationship Prediction in Heterogeneous Bibliographic Networks. In *Proc. of ASONAM*. 121–128.

Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. 2009a. RankClus: Integrating Clustering with Ranking for Heterogeneous Information Network Analysis. In *Proc. of EDBT*. 565–576.

Yizhou Sun, Yintao Yu, and Jiawei Han. 2009b. Ranking-based Clustering of Heterogeneous Information Networks with Star Network Schema. In *Proc. of KDD*. 797–806.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *Proc. of WWW*. 1067–1077.

Kiran K. Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. 2018. Attention-based Graph Neural Network for Semi-supervised Learning. In *arXiv:1803.03735v1*.

Min-Hsuan Tsai, Charu C. Aggarwal, and Thomas S. Huang. 2014. Ranking in heterogeneous social media. In *Proc. of WSDM*. 613–622.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph Attention Networks. In *Proc. of ICLR*. 1–12.

S.V.N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph Kernels. *JMLR* (2010), 1201–1242.

Yueyang Wang, Liang Hu, Yueting Zhuang, and Fei Wu. 2018. Intra-view and Inter-view Attention for Multi-view Network Embedding. In *Proc. of PCM*. 201–211.

Jia Wu, Zhibin Hong, Shirui Pan, Xingquan Zhu, Zhihua Cai, and Chengqi Zhang. 2015. Multi-graph-view subgraph mining for graph classification. *KAIS* 48, 1 (2015), 29–54.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proc. of ICML*. 2048–2057.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proc. of ICLR*. 1–17.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2Seq: Graph to Sequence Learning with Attention-based Neural Networks. In *arXiv:1804.00823v3*.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. 2016. Stacked Attention Networks for Image Question Answering. In *Proc. of CVPR*. 21–29.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Proc. of NeurIPS*. 4805–4815.

Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay S. Pande, and Jure Leskovec. 2018a. Graph Convolutional Policy Network for Goal-Directed Molecular Graph Generation. In *Proc. of NeurIPS*. 6412–6422.

Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018b. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *Proc. of ICML*. 5694–5703.

Jingyuan Zhang, Bokai Cao, Sihong Xie, Chun-Ta Lu, Philip S. Yu, and Ann B. Ragin. 2016. Identifying Connectivity Patterns for Brain Diseases via Multi-side-view Guided Deep Architectures. In *Proc. of SDM*. 36–44.

Jing Zhang, Bo Chen, Xianming Wang, Hong Chen, Cuiping Li, Fengmei Jin, Guojie Song, and Yutao Zhang. 2018. MEgo2Vec: Embedding Matched Ego Networks for User Alignment Across Social Networks. In *Proc. of CIKM*. 327–336.

Zhou Zhao, Ben Gao, Vicent W. Zheng, Deng Cai, Xiaofei He, and Yueting Zhuang. 2017. Link Prediction via Ranking Metric Dual-level Attention Network Learning. In *Proc. of IJCAI*. 3525–3531.

Weiguo Zheng, Lei Zou, Xiang Lian, Dong Wang, and Dongyan Zhao. 2013. Graph similarity search with edit distance constraint in large graph databases. In *Proc. of CIKM*. 1595–1600.

Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Introduction to the Special Section on Urban Computing. *ACM TIST* 5, 3 (2014), 38:1–38:55.

Hao Zhou, Tom Yang, Minlie Huang, Haizhou Zhao, Jingfang Xu, and Xiaoyan Zhu. 2018. Commonsense Knowledge Aware Conversation Generation with Graph Attention. In *Proc. of IJCAI-ECAI*. 1–7.

Sheng Zhou, Jiajun Bu, Xin Wang, Jiawei Chen, Binbin Hu, Defang Chen, and Can Wang. 2019. HAHE: Hierarchical Attentive Heterogeneous Information Network Embedding. In *arXiv:1902.01475v1*.

Yin Zhu, Yuqiang Chen, Zhongqi Lu, Sinno Jialin Pan, Gui-Rong Xue, Yong Yu, and Qiang Yang. 2011. Heterogeneous Transfer Learning for Image Classification. In *Proc. of AAAI*. 1304–1309.

Yuanyuan Zhu, Jeffrey Xu Yu, Hong Cheng, and Lu Qin. 2012. Graph Classification: A Diversified Discriminative Feature Selection Approach. In *Proc. of CIKM*. 205–214.