Node Context Selection in Transformer-Based Graph Representation Learning Models

Tianze Wang
KTH Royal Institute of Technology
Stockholm, Sweden
tianzew@kth.se

Amir H. Payberah
KTH Royal Institute of Technology
Stockholm, Sweden
payberah@kth.se

Vladimir Vlassov
KTH Royal Institute of Technology
Stockholm, Sweden
vladv@kth.se

***Abstract*—Transformer models have great potential in Graph Representation Learning (GRL) for efficiently scaling the learning process on large datasets and solving many challenges presented in Graph Neural Networks, e.g., oversmoothing and suspended animation. To represent each node of a graph, Transformer models as input usually take a node together with the *node context*, i.e., a set of other nodes that serve as learning context for the target node. However, current GRL Transformer models mainly consider the graph topology when selecting the node context for each target node. In this work, we demonstrate the important role of node features in selecting the node context. Specifically, we propose a hybrid approach for selecting node context that considers both the graph topology and the semantic similarities between node features. Through the empirical evaluations, we show the advantages of our hybrid node context selection method for a downstream classification task on various datasets compared to selection methods that only consider graph topology or semantic similarities. The best classification accuracy improvements of our proposed hybrid methods over the baseline methods on each dataset range from 0.77% to 6.05%.**

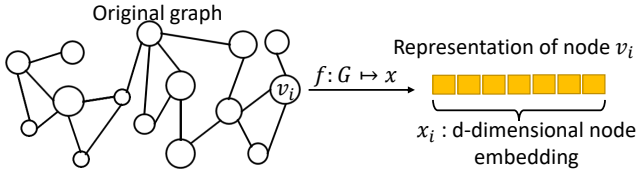
***Index Terms*—Graph Representation Learning, Transformer Models, Node Context Selection, Graph-Bert**

I. Introduction

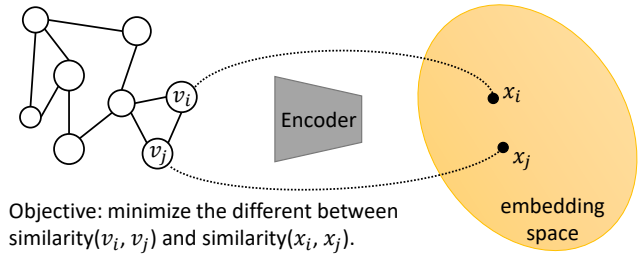
Graphs are data structures that model a set of entities (represented as graph nodes) and relationships between the entities (represented as graph edges). Graphs provide unified representations of complex data in many domains, such as social networks [1], molecule graphs [2], and knowledge graphs [3]. In Graph Representation Learning (GRL) [4]–[6], we need to create fix-sized low-dimensional vectors (graph embeddings) to represent the key information of the graph, e.g., node features and edge connections (Figure 1a and 1b).

The learned graph embeddings can be used in various downstream Machine Learning (ML) and Deep Learning (DL) tasks (Figure 1c), such as knowledge graph completion [7], link prediction [8], and node classification [9]. However, graph data are generally from a non-Euclidean space (e.g., a node can be connected to an arbitrary number of neighboring nodes); thus, the majority of ML/DL models cannot be directly used to create graph embeddings as they require input data from Euclidean space (e.g., a grid-like structure).

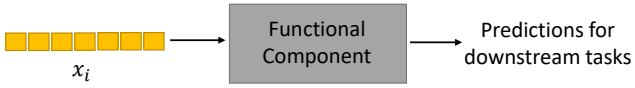
Graph Neural Networks (GNNs) [10] are common approaches to tackle the above problem. GNNs are state-of-the-art methods in GRL to create low-dimensional vectorized representation for nodes, edges, or the whole graph (Figure 1).



(a) In GRL, we want to learn fix-sized low-dimension vector embeddings to represent certain aspects of a graph, e.g., nodes, edges, or the entire graph. In most cases, embeddings are learned on the node level to represent the structure and feature information of each node.



(b) We want a GRL encoder to learn node embeddings so that two nodes v_i and v_j that are similar in the original graph would have embeddings x_i and x_j close to each other in the embedding space.



(c) The learned node embeddings can then be used in downstream tasks, e.g., node classification.

Fig. 1: Graph Representation Learning.

GNNs typically employ a neighborhood aggregation method through message passing between neighbor nodes [11]. In this view, the representation of each node is generated by iteratively collecting and aggregating information from its neighbors. While the neighborhood aggregation schema achieves state-of-the-art performance in many graph-related tasks [5], [12], some challenges come along with this way of creating representations. For example, over-smoothing [13] occurs when a GNN has too many layers, leading to very similar representations for every node in the graph, which usually leads to inferior performance for downstream tasks.

The suspended animation problem [14] occurs when we use a deep GNN with many layers, and the GNN stops responding to the input.

One way to overcome the above challenges is to employ Transformer models [15]–[20] for GRL. Transformers [21] are DL models that adopt self-attention mechanisms and learn to weigh the importance of different input parts when generating the output. They can capture long-range dependencies [22] and can be easily adapted for parallel training. Moreover, the scalable aspect of Transformer models can help tackle the challenges of learning over large graphs, e.g., social networks. Transformers have succeeded in many domains, including text generation [23], natural language understanding [24], image classification [25], and source code clustering [26].

One approach of using Transformers to create node representations is to pass a *target node* (the node we want to create representation for) together with its *target node context* to Transformer that generates a low-dimensional embedding as the representation for that target node. In general, the context for a target is its surrounding objects or entities that can provide and highlight appropriate interpretations for the meaning of the target entity. For example, the context of a pixel in an image is its surrounding pixels, and the context of a word in a sentence is its preceding and succeeding words. In a graph, the target node context serves as a learning context to create embeddings for the target node. The context contains the nodes (usually the target node's nearby neighbors) typically selected based on some relative proximity measurement on the topology of the graph, such as PageRank proximity [27]. For example, `Graph-Bert` [16], which is a state-of-the-art Transformer-based graph embedding model, selects target nodes' context using a method that mainly considers the input graph topology, i.e., the structure of the graph. However, such methods might be suboptimal as they do not consider the semantic features that come with each node.

In this work, we study the impact of node context selection for Transformer models on learning graph representations. Specifically, we propose different node context selection methods and empirically compare their impact on the performance of node classification that utilizes the learned graph representations. Our contribution can be formulated as follows.

1) We compare the performance of node context selection methods that only consider the graph topology with methods that only consider the semantic similarity of the node features.
2) We propose a novel two-step approach for node context selection combining the topological structure of the graph and the semantic meaning of nodes in the graph.
3) We empirically evaluate different node context selection approaches and analyze the results.

II. Preliminaries

In this section, we present the preliminaries of our work. We start with definitions of GRL, then discuss node context selection and an overview of `Graph-Bert`.

A. Graph Representation Learning

Let $G(V, E)$ be a graph, where $v_i \in V$ is a node in the graph, and $e_{ij} = (v_i, v_j) \in E$ is an edge that represents a connection between two nodes v_i and v_j. In GRL, we aim to find a function f that, for the given graph G, creates d-dimension embeddings $\mathbf{x} \in \mathbb{R}^d (d \in N)$ representing certain aspects of the given graph G. For example, $\mathbf{x}$ could represent a node, an edge, or even the entire graph. Such d-dimensional embeddings can later be used as input to other functions to accomplish specific tasks, such as node classification, link prediction, and graph clustering.

There are different approaches for f to create embeddings. Traditional methods such as node2vec [28] and struc2vec [29] work based on random walks on the original graph to create representations such that the distances of two representations $\mathbf{x}_i$ and $\mathbf{x}_j$ in the embedding space reflect the similarity of two nodes v_i and v_j in the original graph G (Figure 1b). However, state-of-the-art methods in GRL use GNNs, such as Graph Convolutional Networks (GCN) [12] and GraphSAGE [5]. To create a node embedding for a node v_i in the graph, a GNN defines a neural network based on the neighborhood structure that collects and aggregates information from all the neighbor nodes of v_i within certain distances, e.g., within 2-hop neighbors.

B. Node Context Selection

To create a representation $\mathbf{x}_i$ for a target node v_i, we need to define a set of nodes C_{v_i}, i.e., the node context of the target node v_i, that serves as the learning context for creating $\mathbf{x}_i$. One of the simplest and most common ways to define node context is to select nodes that are the direct neighbors of the target node v_i. One can also extend the node context by adding neighbors that are further away from v_i. The majority of GNN models [5], [11], [30] create x_i for a target node v_i through message passing by collecting and aggregating information from the local neighborhood of v_i.

Another way to define node context is through top-k intimacy sampling based on an intimacy matrix $S \in \mathbb{R}^{n \times n}$ where n is the number of nodes in the graph G. Each entry $s_{ij} \in S$ is a scalar value that measures the similarity of node v_i and v_j. The similarity here can come from different perspectives, and we show more details about different similarity measurements and node context in Section III. A larger value of s_{ij} represents that nodes v_i and v_j are more similar. In this way, the node context C_{v_i} is a set of k nodes (v_i not included) that are most similar to node v_i according to the intimacy matrix S.

For example, personalized `PageRank` algorithms can define the intimacy matrix as below:

$$\mathbf{S} = \alpha \cdot \left(\mathbf{I} - (1-\alpha) \cdot \bar{\mathbf{A}}\right)^{-1}, \tag{1}$$

where α is a value between 0 and 1 that represents the probability of restart (teleport) during the random walk process. $\mathbf{I} \in \mathbb{R}^{n \times n}$ is an identity matrix, and $\bar{\mathbf{A}}$ is the normalized adjacency matrix. $\bar{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ for a directed graph and $\bar{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ for an undirected graph where $\mathbf{A}$ denotes

the adjacency matrix of graph G and D is the degree matrix of graph G.

C. GraphBert

One of the state-of-the-art methods for applying Transformer models on graphs is `Graph-Bert` [16], where the neural network used is solely based on the attention mechanism without any graph convolution or aggregation. Figure 2 shows a quick summary of `Graph-Bert`.

As we see in Figure 2, to create a d-dimensional representation for a target node v_i using `Graph-Bert`, we first need to define a surrounding node context, which is a set of nodes $v_j \in C_{v_i}$. `Graph-Bert` uses a top-k intimacy sampling approach based on the intimacy matrix S defined in equation 1 capturing the influence between two nodes based on `PageRank` algorithms. Although nodes in the local neighborhood of the target node usually have a higher influence score under such metric, nodes that strongly influence the target node may be far away.

After defining the node context for the target node, we need to prepare the input vectors for the Transformer-based encoder containing: (1) raw node feature vector, (2) Weisfeiler-Lehman absolute role vector, (3) intimacy-based relative positional vector, and (4) hop based relative distance vector. These four vectors capture different information about each node and are aggregated using vector summation. The aggregated vector is then passed to the Transformer-based encoder to get the representation of the target node. The learned representation for the target node is then combined with functional components, e.g., an ML classifier, to optimize the training objective of downstream tasks, e.g., classifying papers into different research areas.

III. Node Context Selection

In this section, we start by motivating the importance of node context, and then we introduce different methods for node context selection.

A. We Shall Know A Node By The Company It Keeps

Representations of nodes using adjacency matrices sit in a high dimensional space (probably non-Euclidean), which may cause many problems as the dimension can vary quite a bit for graphs with different numbers of nodes. In GRL, we would like to create a fix dimensional embeddings to represent the graph. For example, in node classification tasks, we want to generate node embedding for each node in the graph and use it for downstream classification tasks.

To achieve a good classification task performance, nodes should be optimized to be located in the embedding space according to their class labels. That is, nodes with similar classification labels in the original graph are expected to be close to each other in the embedding space, while nodes with different labels would be further away from each other. However, such an objective might be challenging to achieve as the node label distributions for different classification tasks might vary greatly even for the same input graph. This implies the distribution of node embedding in the embedding space might need to vary significantly for different tasks. Furthermore, even if we could create embeddings with such desired properties, they may have poor generalizability for other downstream tasks since the node embedding space is heavily optimized for a specific downstream task (e.g., binary classification of whether a preprint manuscript is accepted or rejected vs. a multi-class classification on which area a paper belongs to). Nevertheless, we can still work on the general notion of optimizing the embedding space not from the downstream task perspective but on the similarities presented in the input graph.

One of the most straightforward ways of defining similarity in a graph is by the edge connection. In this view, we can consider two directly connected nodes as the most similar. A pair of nodes whose shortest path among them is 2-hop are generally considered more similar than another pair of nodes that are further, e.g., 5-hop away. There are different ways to measure similarities using graph connection apart from counting the shortest path length between two nodes in a graph. Personalized `PageRank` [27] is a way to measure the influence of all the other nodes in the graph on a specific target node. In general, this view of similarity captures the information stored in the graph topology, i.e., how nodes are connected to each other in a graph.

Apart from the edge connection between two nodes, each node might also come with a set of features. For example, in a paper citation network, each node represents a paper, and edges represent the citation between papers. One node might also have some features describing the paper that it represents, e.g., year of publication, venue, and a word2vec vector that represents the paper's content. Such vectors can be used to measure the similarity between two nodes. In this view, two vectors describing two data mining papers would probably be closer to each other in the vector space than two vectors, one describing a data mining paper and another describing a computational biology paper.

In this work, we aim to study if we can use the notion of similarity between nodes and graph topology to generate node embeddings to improve the performance of downstream node classification tasks.

B. Node Context Selection Methods

To generate an embedding for a target node in a graph, `Graph-Bert` uses the encoder of a Transformer model by taking the target node and a surrounding node context (a set of nodes of predefined size) as input. This subsection gives an overview of the different methods for selecting node context.

1) Topological context: `Graph-Bert` explores a top-k intimacy sampling and implements the intimacy score based on the personalized `PageRank` method. In the end, the node context of a target node includes k nodes that have the highest influence on the target node according to the personalized `PageRank` method. We shall refer to this method of sampling the node context for a target node as `topological` hereafter, as the node context selection mainly considers the topology

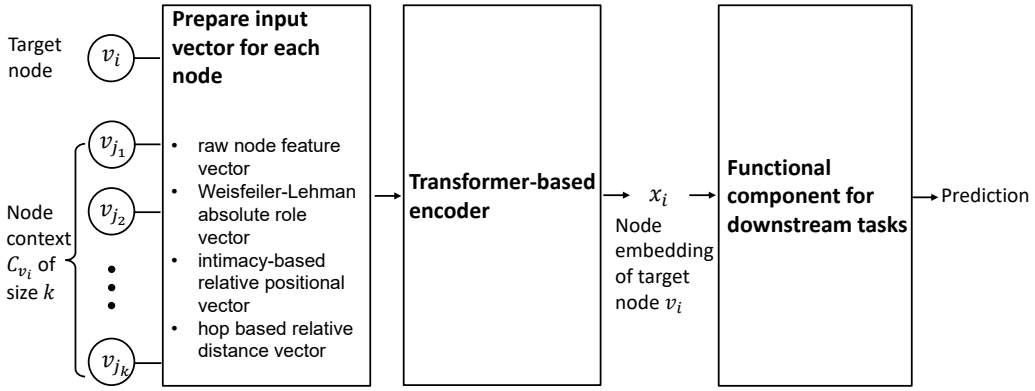


Fig. 2: `Graph-Bert` creates low dimensional representations for target node v_i solely based on Transformer and attention mechanism without graph convolution. `Graph-Bert` learns to generate representation x_i for the target node v_i using both the target node and a node context $C_{v_i} = \{v_{j_1}, v_{j_2}, \cdots, v_{j_k}\}$ and the learned representations can later be used for downstream ML tasks, e.g., node classification.

in the graph. Figure 3a shows how `topological` context works.

2) Semantic context: While `topological` context uses graph topology as a proxy for node similarity, another way to measure node similarity is based on node features, usually composed of a fixed dimension vector for each node. How these feature vectors are generated depends on what each node in a graph represents. For example, if each node represents an image, such features vector could be the features generated by some pre-trained Computer Vision (CV) models, or if each node has some raw text associated with it, e.g., descriptions of a product, title, and abstract of a paper, such feature vectors could be anything that can represent the semantic meaning of those raw texts. It could be as simple as zero-one vectors that indicate if each word in a given vocabulary has appeared in the raw text or not, or it could be feature vectors generated by word2vec [31] or Bert [24] models.

Depending on the type of feature vector used, there are different metrics we can use to measure the similarity between semantic features. In this work, we use either Jaccard or cosine similarity as our similarity measurement. For example, we could use Jaccard similarity to measure the distance between zero-one vectors and cosine similarity to measure the distance between vectors generated by word2vec [31] or Bert [24] models. We will refer to the method that selects node context based on the similarity between semantic features of nodes as `semantic` since it captures how similar nodes are with each other based on what each node describes, as shown in Figure 3b.

3) Hybrid context: In addition to creating node context for a target node using only graph topology or only node features, we propose to combine the topological context with the semantic context. We call this combination a `hybrid` context. We study if such a hybrid approach could lead the `Graph-Bert` to generate node embeddings that can improve the node classification accuracy in the downstream classification task. Figure 3c shows how to use `hybrid` to define a node context. Specifically, to create a node context of size k, we first use topological similarity to preselect a larger set of nodes of size $m \times k$ and then use semantic similarity to select the top k nodes from the preselected $m \times k$, which serve as the final node context of the target node. Here, m is a predefined hyperparameter on how far away we search during preselection. We will refer to the hybrid method as `hybrid` hereafter.

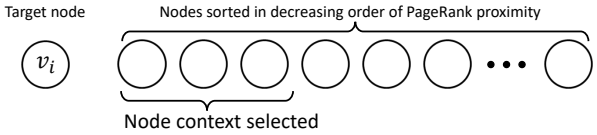
The task of the `hybrid` method can be simplified as defining a node context by finding the most *similar neighbors* of the target node. There are two aspects that we need to consider: (a) *neighbor*: the nodes in the node context should reflect graph structure, and (b) *similar*: the nodes in the node context should also be similar in terms of the semantic node features compared to the target node. In other words, nodes that are far away from each other in the graph and have quite different feature vectors should not appear in each other's node context, as they are probably not able to efficiently learn from each other, at least when not explicitly made clear to GRL models that they are kind of counterexamples to each other.

IV. Evaluation

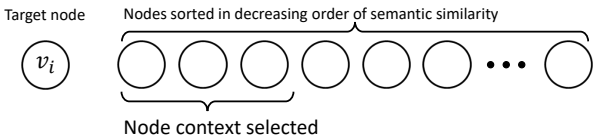
In this section, we present the empirical evaluation of different node context selection methods. We start by introducing the datasets, tasks, and experiment setup. Then, we present and analyze the experiment results. In the end, we discuss the differences between the node context selected using different selection methods.

A. Datasets and Tasks

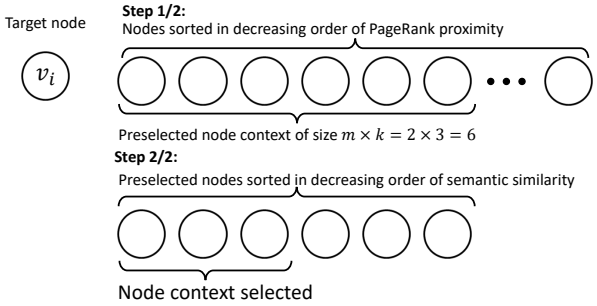
We conduct our empirical evaluation on four datasets `Cora`, `CiteSeer`, `PubMed` as in `Graph-Bert`, and `ogbn-arxiv`, a larger dataset than the first three datasets. Table I provides the statistics for all four datasets. All four datasets are paper citation datasets where each node represents a paper and each edge represents a citation between two papers. We consider node classification as the downstream task where we would like to classify a node into one of



(a) `topological` node context with $k = 3$. The final selected node context are the ones that are most similar to the target node v_i in terms of `PageRank` proximity.



(b) `semantic` node context with $k = 3$. Nodes selected in the final node context are the most similar ones to the target node v_i in terms of semantic similarity, e.g., cosine similarity between node features.



(c) `hybrid` node context with $k = 3$ and $m = 2$. This approach first uses `topological` to preselect nodes and then uses `semantic` to select the final node context from the preselected ones. The output node context contains nodes that are similar to the target node v_i considering both graph structure and node features.

Fig. 3: Demonstration of how different node context selection methods work. The semantic similarity of a node is calculated relative to the target node.

the predefined classes that represent the area that the paper belongs.

TABLE I: Dataset statistics.

Dataset	#Nodes	#Edges	node feature	#Classes
`Cora`	2,708	5,429	0-1	7
`CiteSeer`	3,312	4,715	0-1	6
`PubMed`	19,717	44,338	TF/IDF	3
`ogbn-arxiv`	169,343	1,166,243	skipgram	40

The `Cora` dataset contains a citation graph of ML papers where each node represents one paper, and each edge represents a citation between two papers. Each node in the `Cora` dataset has a 0/1-valued word vector indicating whether or not a node contains the corresponding word in the given vocabulary. The vocabulary contains 1433 unique words. Each node in the `Cora` dataset belongs to one of the following seven categories: "Case Based", "Genetic Algorithms", "Neural Networks", "Probabilistic Methods", "Reinforcement Learning", "Rule Learning", and "Theory".

The `CiteSeer` dataset contains a citation graph of scientific publications. Each node represents a paper and has a 0/1-valued word vector similar to the ones in the `Cora` dataset, except that the vocabulary contains 3703 unique words. Each node in the `CiteSeer` dataset belongs to one of the following six categories: "Agents", "AI", "DB", "IR", "ML", and "HCI".

The `PubMed` dataset is a citation graph of scientific publications. Each node represents a paper and has a TF/IDF weighted word vector [32] with a vocabulary size of 500. Each paper in the `PubMed` dataset belongs to one of the following three categories: "Diabetes Mellitus, Experimenta", "Diabetes Mellitus Type 1", and "Diabetes Mellitus Type 2".

The `ogbn-arxiv` dataset is a citation graph between Computer Science (CS) arXiv[1] papers. Each node represents a paper and has a 128-dimensional feature vector that is obtained by averaging word embeddings in the title and abstract of the paper. Each word embedding is generated using the skip-gram model [33]. Apart from the feature vector, the `ogbn-arxiv` dataset also includes the raw text of titles and abstracts for each paper. We use pretrained Sentence-BERT (SBERT) [34] to generate a second 768-dimensional feature vector for each paper from a single sentence that is based on the raw text of its title and abstract. Each paper in the `ogbn-arxiv` dataset belongs to one of the 40 categories[2] within CS papers in arXiv, e.g., cs.AI - Artificial Intelligence.

B. Experiment Setup

We implement different ways of constructing node context and test the impact of node context selection on the final node classification accuracy with different node context size k. For `hybrid`, we also test different multiples m. The experiment code can be accessed from here [3].

For the `Cora`, `CiteSeer`, and `PubMed` datasets, we repeat each experiment configuration 50 times to minimize the impact of randomness and stochastic on the result. For the `ogbn-arxiv` dataset, since each individual experiment takes much longer to run than the first three datasets, we repeat 10 times due to the time limit. We used the same train, validation, and test split as `Graph-Bert` for `Cora`, `CiteSeer`, and `PubMed` and used the recommended splits by ogbn for the `ogbn-arxiv` dataset.

We conducted experiments on the `Cora`, `CiteSeer`, and `PubMed` dataset on a local workstation with AMD Ryzen Threadripper 2920x 12-Core Processor, 128 GB of RAM, and NVIDIA GeForce RTX 2070 Super GPUs. For the experiments on `ogbn-arxiv`, we use SNIC clusters to satisfy the GPU memory requirements. For each repeated run, we give the `Graph-Bert` model with different context 200 epochs of training and report the test accuracy of the model at the time when it reaches the highest validation accuracy. The final test

[1]https://arxiv.org/
[2]https://arxiv.org/archive/cs
[3]https://github.com/bwhub/ncgt

accuracy reported is an average across repeated runs for the given configuration.

C. Results and Analysis

Table II shows the result on the Cora dataset. We test two different variants for measuring semantic similarity, i.e., Jaccard and cosine similarities, since the node features in the Cora dataset are zero-one vectors indicating if a word appears in the given vocabulary. For hybrid context, we also tested different values of k and m, i.e., $k = 5, 7, 10, 20, 30, 50$, and $m = 2, 5, 7, 10, 20$ to show the effect of the size of node context and preselection neighborhood on the final classification accuracy. We reported the average test accuracy and standard deviations over 50 repeated runs for each experiment configuration.

Table II shows that semantic is performing worse than topological regardless of the similarity of the measurement used, i.e., cosine or Jaccard. This could be partially explained by the fact that semantic context performs a selection of node context for a target node only based on similarity measurements over all the nodes in the entire graph. Such a global view would let the Graph-Bert model lose focus on the connection between nodes in the local neighborhood of the target node, meaning that the Graph-Bert model will not learn much about the structure information in the graph represented in the edge and the final representation of a node is mostly related to the feature it has.

The topological context, on the other hand, does not suffer from a loss of structure information by selecting the node context of a target node respecting the connections within the local neighborhood. The hybrid context selects node context for a target node considering both the structure of the local neighborhood and the semantic similarities between node features. Thus, the better selection of node context lead to better classification performance in general when comparing to semantic and topological, e.g., 0.8424 versus 0.8115 when comparing hybrid, Jaccard, $m = 5$ and topological for $k = 10$.

The size of node context, i.e., k, also plays an important role in the performance of the Graph-Bert model on node classification. We generally observe a trend of better performance when using a larger k for topological and hybrid context. This could be explained by the fact that a larger node context allows the Graph-Bert model to have more comparisons of the nodes within a given node context. Thus, the Graph-Bert model can create node embeddings where similar nodes (both in structure and feature semantic similarities) can sit close to each other in the latent representation space. For example, increasing k from 5 to 7 boosts the accuracy of hybrid, cosine, $m = 7$ from 0.8187 to 0.8398.

The size of the preselection node context ($m \times k$) also impacts the final classification accuracy. Increasing m increases the size of the preselection neighborhood and usually leads to better classification performance. For example, increasing m from 2 to 5 increase the accuracy from 0.8341 to 0.8424 for hybrid, Jaccard when $k = 10$. Too large of a m, on the other hand, can potentially decrease the performance as too large of a preselection node context would fade out the benefit of the preselection step. Such a node selection process undermines the impact of graph structure on the selection process, resulting in a node context that is similar to one selected using only semantic context. For example, when $k = 50$ increasing m from 10 to 20 decrease the classification accuracy from 0.8284 to 0.8089 for hybrid context with Jaccard as similarity measurement.

Table III follows the same schema as Table II and shows the result on the CiteSeer dataset. Overall, experiment results on CiteSeer are similar to those on Cora. This could be because Cora and CiteSeer resemble each other in many aspects, e.g., the number of nodes and edges in the graph, node feature type, and the number of categories for node classification. One interesting observation is that cosine similarity seems to be slightly better than Jaccard similarity most of the time. This could be an indication that similarity measurements (e.g., Jaccard and cosine) and feature representations (e.g., zero-one vector) also affect the performance of downstream classification tasks (see the results on PubMed and ogbn-arxiv datasets).

Table IV shows the result on PubMed dataset. Overall, Jaccard is a better similarity measurement of node features on PubMed dataset. For example, semantic context always has better classification performance when using Jaccard similarity compared to cosine similarity and would sometimes even outperform the topological context, e.g., $k = 5, 20, 30, 50$. The advantages of Jaccard similarity also appear when using hybrid context as the performance is generally better than those using cosine similarity. This may indicate that the node features (TF/IDF weighted word vectors) in PubMed dataset are better suited with Jaccard similarity than cosine similarity as a measurement of the distance between node features.

Table V shows the experiment results on ogbn-arxiv dataset. We test the impact of node context selection for two sets of features, skip-gram and SBERT [34]. Skip-gram rows in Table V show the experiment results using the original node feature that comes with the ogbn-arxiv dataset and the SBERT rows show the results where we use features generated by a pretrained SBERT model [34].

Overall, the trend is the same as previous results on Cora, CiteSeer, and PubMed, where hybrid has the best performance overall, and semantic is worse than topological. However, compared to node features generated by skip-gram, SBERT generates context-aware features for text summarization and usually performs better. Our experiment result shows that using SBERT node features is better than using skip-gram features for the hybrid model (Graph-Bert model with hybrid context) as it can achieve higher accuracy with the prior feature. This indicates that apart from node context selection, the quality of node features that serve as input to the hybrid model also plays an important role in the model's performance. The better the input node features are, the higher the node classification accuracy we can reach with

TABLE II: Average accuracy and standard deviation of different node contexts on Cora dataset.

Node Context	k=5	k=7	k=10	k=20	k=30	k=50
semantic, cosine	0.7917±0.0119	0.7908±0.0132	0.7875±0.0141	0.7886±0.0134	0.7886±0.0104	0.7904±0.0099
semantic, jaccard	0.7828±0.0152	0.7923±0.0121	0.7895±0.0100	0.7848±0.0129	0.7852±0.0107	0.7896±0.0081
topological	0.8177±0.0081	0.8157±0.0102	0.8115±0.0103	0.8286±0.0099	0.8221±0.0108	0.8285±0.0119
hybrid, cosine, m=2	0.8109±0.0073	0.8072±0.0088	0.8322±0.0077	0.8339±0.0057	0.8361±0.0060	0.8370±0.0084
hybrid, cosine, m=5	0.8194±0.0079	0.8355±0.0064	0.8420±0.0072	0.8334±0.0100	0.8405±0.0091	**0.8380±0.0054**
hybrid, cosine, m=7	0.8187±0.0047	**0.8398±0.0068**	0.8365±0.0081	0.8409±0.0081	0.8403±0.0074	0.8360±0.0070
hybrid, cosine, m=10	0.8227±0.0071	0.8265±0.0077	0.8285±0.0076	**0.8469±0.0067**	**0.8434±0.0077**	0.8284±0.0056
hybrid, cosine, m=20	0.8203±0.0065	0.8264±0.0071	0.8270±0.0086	0.8326±0.0079	0.8259±0.0072	0.8117±0.0067
hybrid, jaccard, m=2	0.8047±0.0091	0.8050±0.0085	0.8341±0.0068	0.8362±0.0049	0.8368±0.0066	0.8370±0.0072
hybrid, jaccard, m=5	**0.8285±0.0062**	0.8359±0.0045	**0.8424±0.0043**	0.8328±0.0100	0.8419±0.0064	0.8377±0.0063
hybrid, jaccard, m=7	0.8222±0.0053	0.8381±0.0048	0.8368±0.0056	0.8371±0.0074	0.8431±0.0055	0.8352±0.0065
hybrid, jaccard, m=10	0.8255±0.0073	0.8307±0.0078	0.8295±0.0070	0.8449±0.0085	0.8410±0.0071	0.8284±0.0062
hybrid, jaccard, m=20	0.8192±0.0073	0.8280±0.0082	0.8220±0.0088	0.8330±0.0071	0.8198±0.0076	0.8089±0.0072

TABLE III: Average accuracy and standard deviation of different node contexts on CiteSeer dataset.

Node Context	k=5	k=7	k=10	k=20	k=30	k=50
semantic, cosine	0.6991±0.0058	0.7008±0.0055	0.6991±0.0054	0.6984±0.0063	0.7032±0.0106	0.7083±0.0097
semantic, jaccard	0.6973±0.0048	0.6980±0.0041	0.6990±0.0061	0.6980±0.0061	0.7008±0.0090	0.7063±0.0093
topological	0.6977±0.0049	0.6970±0.0046	0.6964±0.0045	0.6952±0.0045	0.6962±0.0055	0.6982±0.0073
hybrid, cosine, m=2	0.6961±0.0049	0.6968±0.0046	0.7034±0.0088	0.7061±0.0106	0.7138±0.0120	0.7158±0.0078
hybrid, cosine, m=5	0.7108±0.0063	**0.7116±0.0083**	**0.7152±0.0077**	0.7159±0.0065	0.7131±0.0090	0.7124±0.0071
hybrid, cosine, m=7	0.7058±0.0061	0.7099±0.0070	0.7082±0.0051	0.7182±0.0045	**0.7199±0.0056**	0.7155±0.0078
hybrid, cosine, m=10	**0.7144±0.0064**	0.7049±0.0060	0.7042±0.0051	0.7079±0.0058	0.7153±0.0073	0.7114±0.0078
hybrid, cosine, m=20	0.7040±0.0072	0.7039±0.0072	0.7091±0.0105	0.7102±0.0083	0.7077±0.0099	0.7088±0.0114
hybrid, jaccard, m=2	0.6980±0.0049	0.6977±0.0046	0.7030±0.0086	0.7056±0.0105	0.7131±0.0119	0.7127±0.0106
hybrid, jaccard, m=5	0.7137±0.0073	0.7110±0.0061	0.7147±0.0095	0.7161±0.0069	0.7143±0.0085	**0.7160±0.0062**
hybrid, jaccard, m=7	0.7042±0.0061	0.7085±0.0076	0.7094±0.0049	**0.7186±0.0052**	0.7172±0.0084	0.7140±0.0068
hybrid, jaccard, m=10	0.7140±0.0069	0.7053±0.0061	0.7059±0.0053	0.7081±0.0062	0.7170±0.0068	0.7100±0.0060
hybrid, jaccard, m=20	0.7049±0.0064	0.7038±0.0060	0.7056±0.0094	0.7090±0.0087	0.7092±0.0090	0.7072±0.0121

TABLE IV: Average accuracy and standard deviation of different node contexts on PubMed dataset.

Node Context	k=5	k=7	k=10	k=20	k=30	k=50
semantic, cosine	0.6530±0.0117	0.6616±0.0099	0.6761±0.0112	0.6883±0.0106	0.6908±0.0125	0.6928±0.0114
semantic, jaccard	0.7281±0.0127	0.7358±0.0084	0.7499±0.0036	0.7691±0.0038	0.7764±0.0071	0.7572±0.0109
topological	0.6942±0.0089	0.7391±0.0097	0.7520±0.0111	0.7326±0.0104	0.7387±0.0095	0.7499±0.0078
hybrid, cosine, m=2	0.7359±0.0111	0.7492±0.0097	0.7428±0.0085	0.7701±0.0094	0.7915±0.0069	0.7915±0.0074
hybrid, cosine, m=5	0.6866±0.0079	0.7147±0.0095	0.7385±0.0078	0.7640±0.0079	0.7734±0.0090	0.7814±0.0088
hybrid, cosine, m=7	0.7118±0.0077	0.7203±0.0078	0.7444±0.0100	0.7551±0.0084	0.7602±0.0071	0.7655±0.0100
hybrid, cosine, m=10	0.7219±0.0077	0.7429±0.0089	0.7514±0.0118	0.7459±0.0088	0.7533±0.0089	0.7531±0.0089
hybrid, cosine, m=20	0.7336±0.0067	0.7289±0.0069	0.7239±0.0057	0.7428±0.0109	0.7397±0.0084	0.7486±0.0085
hybrid, jaccard, m=2	**0.7468±0.0085**	**0.7551±0.0088**	0.7510±0.0071	0.7788±0.0080	0.7893±0.0061	0.7989±0.0078
hybrid, jaccard, m=5	0.7208±0.0120	0.7250±0.0094	0.7597±0.0075	**0.7905±0.0057**	**0.7992±0.0052**	0.8059±0.0070
hybrid, jaccard, m=7	0.7075±0.0079	0.7454±0.0100	0.7646±0.0084	0.7896±0.0065	0.7972±0.0076	**0.8061±0.0067**
hybrid, jaccard, m=10	0.7086±0.0105	0.7544±0.0070	**0.7721±0.0056**	0.7875±0.0043	0.7948±0.0043	0.8025±0.0076
hybrid, jaccard, m=20	0.7363±0.0073	0.7503±0.0058	0.7589±0.0088	0.7744±0.0099	0.7885±0.0060	0.8018±0.0068

the hybrid model.

We also compare the hybrid model with some baseline models: Multilayer Perceptron (MLP), GCN and GraphSAGE. MLP is a fully connected neural network classifier that takes node features as input and classifies each node feature into different categories without considering graph structure. GCN and GraphSAGE, on the other hand, represent baseline GNN models that learn representations for graph nodes considering graph structure and classify nodes into different categories. All the baseline models have two layers of neural network, 32 hidden channel, and are trained for 200 epochs for a fair comparison with the hybrid model.

Table VI shows the result of MLP, GCN, GraphSAGE, and the hybrid model (the best accuracies in Table V) on ogbn-arxiv dataset. MLP has the worst performance compared to all the other models since it does not utilize citation information between papers encoded on the edges of a graph. GCN and GraphSAGE utilize such structural information and achieve better classification performance than MLP. The type of node features also plays an essential role in the final classification accuracy. We can observe from Table VI that all models achieve better classification accuracy when using node features created by the SBERT model compared to node features created by the skip-gram model.

This confirms our hypothesis that the SBERT model can create better node features than the skip-gram model. In our view, there are the following potential reasons that lead to features of better quality: (1) compared to the skip-gram model, SBERT can create context-aware word embeddings that can better summarize the semantic of each word in the given

TABLE V: Average accuracy and standard deviation of different node contexts on ogbn-arxiv dataset.

Node Context	k=5	k=7	k=10	k=20	k=30
semantic, cosine, skipgram	0.6635±0.0033	0.6620±0.0039	0.6640±0.0034	0.6645±0.0056	0.6634±0.0039
semantic, cosine, SBERT	0.6967±0.0027	0.6971±0.0026	0.6974±0.0024	0.6971±0.0021	0.6962±0.0039
topological, skipgram	0.6709±0.0035	0.6721±0.0031	0.6741±0.0038	0.6844±0.0031	0.6884±0.0026
topological, SBERT	0.6967±0.0034	0.6983±0.0028	0.6974±0.0029	0.6985±0.0031	0.6993±0.0059
hybrid, cosine, m=2, skipgram	0.6720±0.0043	0.6747±0.0031	0.6791±0.0028	0.6855±0.0022	0.6900±0.0028
hybrid, cosine, m=2, SBERT	0.6954±0.0027	0.6971±0.0036	0.6969±0.0034	**0.7009±0.0028**	0.6994±0.0044
hybrid, cosine, m=7, skipgram	0.6745±0.0033	0.6798±0.0043	0.6807±0.0037	0.6827±0.0035	0.6858±0.0021
hybrid, cosine, m=7, SBERT	0.6985±0.0037	**0.6992±0.0041**	**0.6987±0.0053**	0.6989±0.0053	0.6984±0.0047
hybrid, cosine, m=10, skipgram	0.6742±0.0028	0.6774±0.0052	0.6798±0.0039	0.6844±0.0043	0.6853±0.0038
hybrid, cosine, m=10, SBERT	**0.6990±0.0016**	0.6989±0.0035	**0.6987±0.0035**	0.7002±0.0023	**0.7026±0.0031**

TABLE VI: Average accuracy and standard deviation of baseline models on ogbn-arxiv dataset.

Models	Node Feature	
	skipgram	SBERT
MLP	0.5021±0.0026	0.6134±0.0021
GCN	0.6863±0.0026	**0.7098±0.0027**
GraphSAGE	0.6806±0.0037	0.7073±0.0042
hybrid	**0.6900±0.0028**	0.7026±0.0031

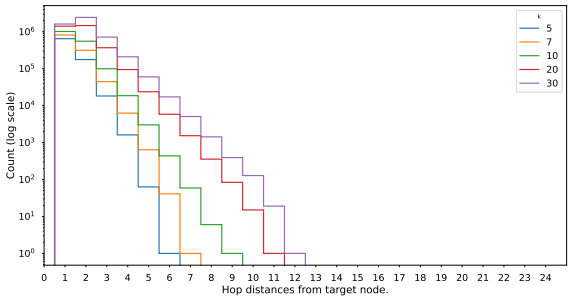


Fig. 4: topological hop distance distribution on ogbn-arxiv dataset.

raw text, (2) SBERT is pretrained on large-scale datasets and has better generalizability, and (3) the node feature created by SBERT has higher dimensionality than the features created by the skip-gram (768 versus 128) and has potentially more representative power.

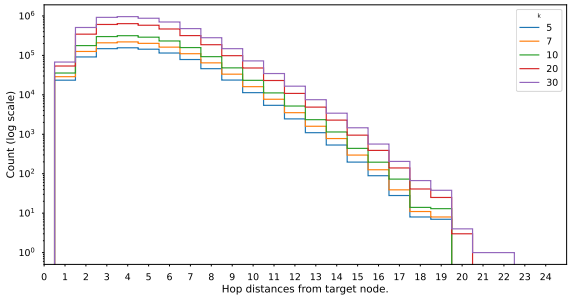
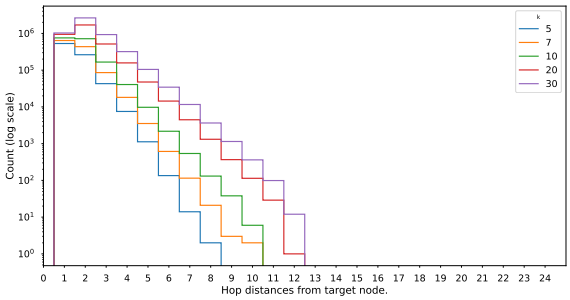
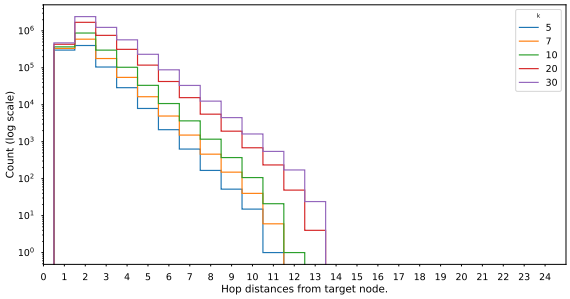


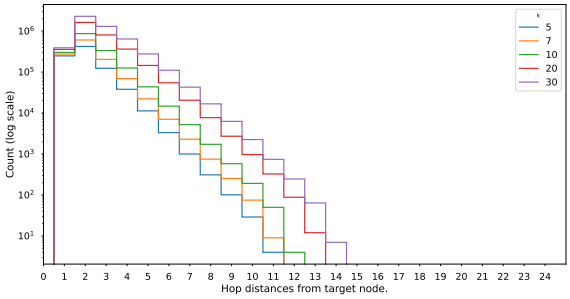
Fig. 5: semantic (SBERT, cosine) hop distance distribution on ogbn-arxiv dataset.



(a) $m = 2$.



(b) $m = 7$.



(c) $m = 10$.

Fig. 6: hybrid (SBERT, cosine) hop distance distribution on ogbn-arxiv dataset.

D. The effect of node context selection

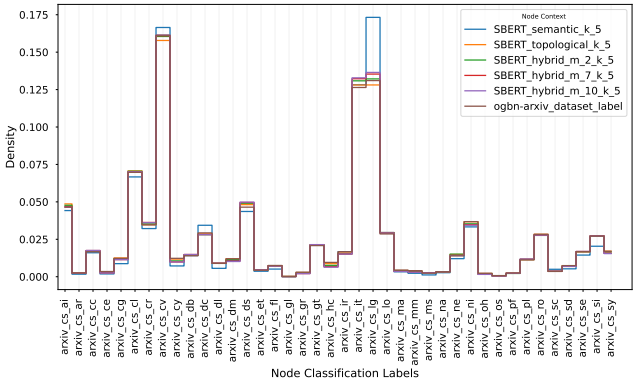
To further understand the potential reason for differences in node classification performances, we also compare the differences between the node context defined using topological, semantic, and hybrid.

Figures 4, 5, 6a, 6b, and 6c show the distance between the target node and the nodes in its node context on the

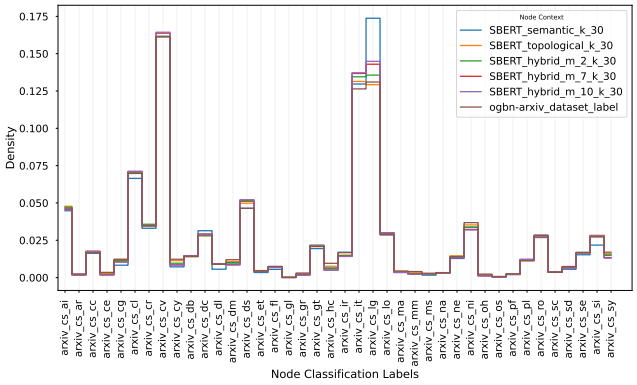
ogbn-arxiv dataset using topological, semantic, and hybrid. Overall, we can observe that most of the nodes in the node context are a few hops neighbors of the target node, and as we increase the size of node context k, more and more nodes that are further away from the target node starts to show in the node context.

One interesting observation is that semantic tends to define node context that is further away from the target node than topological. This could be a potential reason why semantic context is getting worse performance than topological as the Graph-Bert model might not be able to efficiently learn the local neighborhood structure of the target node when using semantic context. The hybrid context, on the other hand, does not sample node context with nodes that are too far away from the target node as it combines topological and semantic context.

Another factor that might contribute to the worse performance for semantic context could be that it tends to oversample or undersample certain classes. Figure 7 shows that semantic context, compared to the label distribution of nodes in ogbn-arxiv dataset, oversamples certain classes, e.g., arxiv_cd_lg and arxiv_cd_it and undersamples other classes, e.g., arxiv_cs_cy and arxiv_cs_dl. Such an unbalanced node context could lead the Graph-Bert model to over or under-emphasize the learning of certain node classes and decrease node classification accuracy compared to hybrid and topological contexts.



(a) $k = 5$.



(b) $k = 30$.

Fig. 7: Node classification label distribution of different node context.

V. Related Work

This section shows the related work for using Transformer models for graph representation learning. We start with an overview of recent studies of Transformer models on graphs, and then we dive deeper into those more related to our work.

A. Transformer on homogeneous graphs

Several work studies on how to use Transformer models to learn graph representations on homogeneous graphs where there are only one type of nodes and one type of edges. Graph Transformer (GT) [15] generalizes Transformer models to homogeneous graphs with a focus on exploring the sparsity and positional encodings using Laplacian eigenvectors. GT also has an extended version that uses edge features in its encoding pipeline. Graph-Bert [16] demonstrates the possibility of learning graph representations only using the attention mechanism without any neighborhood aggregation and graph convolution. Furthermore, Graph-Bert shows the possibility of pretraining on general graph representation learning tasks and fine-tuning on domain-specific downstream applications.

B. Transformer on heterogeneous graphs

Some studies focus on GRL with Transformer models on heterogeneous graphs that contain various types of nodes and edges. Graph Transformer Network (GTN) [17] uses Transformer to learn and generate multi-hops meta-path by a soft selection of edge types. These meta-paths are then used by an ensemble of GCN models to create node representations for the input heterogeneous graph. Heterogeneous Graph Transformer (HGT) [18] treats heterogeneous edges as meta relations and maintains type-specific representations for nodes and edges with heterogeneous attention mechanisms. Apart from that, HGT also proposed its own mini-batch sampling method for scalable training of HGT on heterogeneous graphs.

C. Encoding Graph Structure

Dwivedi et al. [35] discuss the idea of positional encoding for GNN and introduce Learnable Structural and Positional Encodings (LSPE) that can be combined with any message-passing GNN. Graphormer [19] uses spatial, centrality, and edge encoding to effectively encode structure information that helps Transformer to better model graph data. Spectral Attention Network (SAN) [20] explores learned positional encoding that is later added to the input of Transformer models. The learned positional encoding in SAN utilizes the full Laplacian spectrum and can better represent node positions in a graph.

VI. Conclusion

In this work, we study the impact of node context selection for Transformer models in graph representation learning. Specifically, we study this problem using Graph-Bert, one

of the state-of-the-art methods of applying Transformer models for graph representation learning. We empirically compare different methods for selecting node context for a target node and find that our proposed hybrid method, which considers both (1) the topological information encoded in graph edges and (2) the semantic similarity of node features, has the best performance overall, with accuracy improvements of up to 6.05% on downstream classification tasks. We also show that increasing the quality of node features, e.g., using SBERT instead of skip-gram features, can increase the performance in graph representation learning.

Potential extensions for future work include more efficient methods for sampling nodes based on graph topology than the `PageRank` method. The impact of node context selection on the self-supervised pretraining of Transformer models for graph representation learning. The combination of contrastive learning and node context selection of Transformer models for graph representation learning.

ACKNOWLEDGMENT

Special thanks to Vangjush Komini for the generous discussion throughout this work. The computations were partially enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at Chalmers Centre for Computational Science and Engineering (C3SE) partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

REFERENCES

[1] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake news detection on social media using geometric deep learning," *arXiv preprint arXiv:1902.06673*, 2019.

[2] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," *Advances in neural information processing systems*, vol. 28, 2015.

[3] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2021.

[4] D. Zhang, J. Yin, X. Zhu, and C. Zhang, "Network representation learning: A survey," *IEEE transactions on Big Data*, vol. 6, no. 1, pp. 3–28, 2018.

[5] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1025–1035.

[6] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[7] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[8] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[9] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social network data analytics*. Springer, 2011, pp. 115–148.

[10] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[11] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International conference on machine learning*. PMLR, 2017, pp. 1263–1272.

[12] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *J. International Conference on Learning Representations (ICLR 2017)*, 2016.

[13] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI conference on artificial intelligence*, 2018.

[14] J. Zhang and L. Meng, "Gresnet: Graph residual network for reviving deep gnns from suspended animation," *arXiv preprint arXiv:1909.05729*, 2019.

[15] V. P. Dwivedi and X. Bresson, "A generalization of transformer networks to graphs," *arXiv preprint arXiv:2012.09699*, 2020.

[16] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-bert: Only attention is needed for learning graph representations," *arXiv preprint arXiv:2001.05140*, 2020.

[17] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph transformer networks," *Advances in neural information processing systems*, vol. 32, 2019.

[18] Z. Hu et al., "Heterogeneous graph transformer," in *Proceedings of The Web Conference 2020*, 2020, pp. 2704–2710.

[19] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu, "Do transformers really perform badly for graph representation?" *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[20] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] D. Bahdanau et al., "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[23] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[25] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[26] M. Hägglund, F. J. Pena, S. Pashami, A. Al-Shishtawy, and A. H. Payberah, "Coclubert: Clustering machine learning source code," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 151–158.

[27] J. Klicpera et al., "Predict then propagate: Combining neural networks with personalized pagerank for classification on graphs," in *International conference on learning representations*, 2018.

[28] A. Grover et al., "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, 2016, pp. 855–864.

[29] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 385–394.

[30] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[31] T. Mikolov et al., "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[32] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information processing & management*, vol. 24, no. 5, pp. 513–523, 1988.

[33] T. Mikolov et al., "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[34] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[35] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," *arXiv preprint arXiv:2110.07875*, 2021.