The International Journal of Business Management and Technology, Volume 8 Issue 2 March-April 2024
ISSN: 2581-3889

Research Article Open Access

Node Classification Algorithms in Complex Networks Using Graph Embedding

MUSTAFA MAHYOUB SAEED MOHAMMED[1],SOROUSH AQA MAHDI[2]
Computer Science and Technology

ABSTRACT: This research introduces a novel approach to node classification in complex networks, aiming to enhance accuracy and adaptability across diverse network applications. The Dual Autoencoder Learning Method for Attribute Network Representation is proposed, leveraging two autoencoder channels to capture both structural and attribute information. The first channel uses a multi-hop attention mechanism to incorporate local and global structural aspects, while the second channel employs low-pass filtering to extract attribute information guided by structural characteristics. The innovative fusion process integrates representations from both channels, addressing limitations in existing methods. Experimental validation demonstrates superior performance compared to existing algorithms, highlighting the potential of this approach to improve node classification accuracy and adaptability. This research contributes to advancing graph embedding and node classification techniques, providing a foundation for further exploration in dynamic network environments.

Keywords: Node Classification, Algorithms, Graph Neural Networks, Machine Learning, Complex Networks, Representation Learning, Network embedding

I. BACKGROUND AND MOTIVATION

Node classification within complex networks is an essential task with applications spanning diverse domains such as social network analysis, biological network analysis, and recommendation systems. The process involves assigning nodes in a network to predefined classes or labels based on attributes, connections, or interactions, thereby facilitating insights extraction, targeted analysis, and predictive modeling within networks [1]; [2]; [3].

Node classification holds paramount significance in network analysis across a spectrum of applications. Properly classifying nodes in complex networks yields valuable insights, enhances predictive modeling, and empowers decision-making processes. Accurate node classification is indispensable for network analysis and its applications across diverse fields, including social networks, biology, and recommendation systems [4].

In recent years, algorithms rooted in graph neural networks (GNNs) have gained prominence for node classification, capturing intricate dependencies and interactions within network data. These algorithms iteratively update node representations by assimilating data from neighboring nodes, effectively encapsulating the network's local and global structure. This makes GNNs especially well-suited for tasks like node classification, where both structural and attribute information are pivotal [5].

The evolution of node classification algorithms has been profoundly impacted by the availability of extensive network data and the progress of machine learning techniques. Researchers persist in exploring innovative approaches that amalgamate diverse information sources and harness the unique characteristics of complex networks to elevate classification accuracy and scalability [6].

Accurate node classification is particularly significant in social network analysis, aiding in identifying the roles of individuals within a social structure, like categorizing users as influencers, followers, or moderators based on their activity and connections. In biological network analysis, precise classification assists in elucidating functional relationships and pathways. Additionally, in recommendation systems, accurate classification contributes to personalized recommendations by inferring user preferences from their interactions [2]; [2]; [3]).

The impact of inaccurate node classification can be profound, leading to distorted network analysis and misguided strategies. To address these challenges, advanced machine learning techniques, such as graph neural networks (GNNs), have emerged, capable of capturing intricate network structures and attributes, thereby enabling more accurate node classifications. This empowers researchers and practitioners to make informed decisions and develop more effective strategies across various applications [7].

1.1 Research Objectives

Our research objectives encompass the exploration and integration of key concepts that contribute to the advancement of node classification algorithms within complex networks. These objectives focus on leveraging potential information, representation learning techniques, Dual Autoencoder Learning Method for Attribute Network Representation, and the amalgamation of models to enhance the accuracy and robustness of node classification processes.

II. LITERATURE REVIEW

2.1 Traditional Methods for Node Classification

Traditional methods for node classification have been foundational in the field of network analysis, providing essential understanding of node roles and relationships within complex networks, such as identifying influential nodes, detecting communities, and uncovering patterns of interaction.These methods have laid the groundwork for subsequent advancements, while their underlying principles remain relevant today.One of the earliest and most straightforward approaches is the use of local network properties. In this method, nodes are classified based on their degree of centrality, which quantifies the number of connections a node possesses. Nodes with high degrees are often considered hubs, while those with lower degrees are seen as peripheral nodes. This method is rooted in the assumption that well-connected nodes play crucial roles in information diffusion and network dynamics [8].

Another classic approach involves utilizing node attributes, such as age, gender, or location, for classification, wherein classifiers are trained based on these attributes, with decision trees, support vector machines (SVMs), and naive Bayes classifiers commonly employed in attribute-based node classification to discern patterns and relationships within the network [9].

Community-based methods are also prevalent in traditional node classification. Nodes are classified based on their affiliation with certain communities or clusters within the network. Algorithms like the Girvan-Newman algorithm and modularity optimization aim to identify densely connected groups of nodes, and nodes are then classified according to the communities they belong to. This approach is particularly effective in capturing group dynamics and identifying roles within substructures [10].Furthermore, label propagation methods are widely used in traditional node classification. These methods propagate labels through the network based on local connections and label similarities. The idea is that nodes with similar attributes or connections are likely to share the same label. The process is iterative, refining labels as information spreads through the network [11].

While these traditional methods have paved the way for network analysis, they often face limitations when dealing with the complexity and dynamics of real-world networks. In response, modern approaches, such as graph neural networks (GNNs), have emerged to address these challenges by incorporating more sophisticated techniques for capturing both structural and attribute information.

2.2 Introduction to Representation Learning-Based Approaches and Their Advantages

Representation learning-based approaches have revolutionized the field of network analysis by offering powerful tools for understanding complex relationships within networks. These approaches seek to convert raw network data into meaningful and informative representations that capture underlying patterns and structures.By learning compact and expressive representations, they enable more effective analysis, visualization, and prediction in various applications.Representation learning is a sophisticated process involving the transformation of nodes within a network into low-dimensional vector spaces, where each dimension encapsulates specific aspects of the node's characteristics or interactions. This dimensionality reduction serves not only for visualization but also enhances various downstream tasks such as link prediction, node classification, and community detection. Numerous representation learning techniques have emerged, each possessing distinct strengths and advantages.

A notable technique within representation learning is network embedding. Exemplified by methods like node2vec [11] and DeepWalk [3], network embedding draws inspiration from word embeddings in natural language processing. These methods generate node embeddings by simulating random walks on the network, learning embeddings based on encountered node sequences. This approach ensures nodes with analogous structural roles are closely embedded in the vector space, effectively capturing both local and global network structures.

Graph neural networks (GNNs) constitute another class of representation learning methods that have garnered considerable attention. GNNs extend neural network architectures to directly process graph-structured data, offering a versatile and powerful approach to representation learning in complex networks. Their ability to capture intricate relationships and dependencies within graph structures contributes to their increasing significance in the evolving landscape of representation learning techniques. They operate by aggregating information from neighbouring nodes and edges, iteratively updating node representations. GNNs have demonstrated remarkable capability in capturing complex interactions, hierarchical structures, and transitive dependencies within networks [2]. They have been successfully applied in tasks like node classification, link prediction, and graph classification.

The advantages of representation learning-based approaches are multifaceted:

1. **Information Fusion:** These approaches seamlessly integrate both structural and attribute information, enabling a holistic view of nodes and their relationships. This fusion of information enhances the quality of learned representations.
2. **Non-Linearity:** Representation learning techniques can capture non-linear relationships among nodes, which are often missed by traditional linear methods. This allows for more accurate modeling of intricate network behaviors.
3. **Scalability:** Many representation learning methods, particularly GNNs, are designed to scale effectively to large networks. Their ability to process large-scale data makes them well-suited for real-world applications.
4. **Generalizability:** Learned representations can be transferred to new tasks or domains, minimizing the need for extensive retraining. This generalizability is particularly valuable in scenarios with limited labeled data.
5. **Interpretable Embeddings:** Some methods provide interpretable embeddings, allowing researchers to analyze and understand the factors contributing to node relationships and behaviors.
6. **Improved Downstream Tasks:**Enhanced node representations correlate with better outcomes in tasks like link prediction, node classification, and community detection.

As networks grow in complexity and size, representation learning-based approaches continue to evolve, addressing challenges related to scalability, heterogeneity, and dynamic networks. These techniques hold promise in enhancing our understanding of diverse networks and enabling more accurate and robust analyses.

2.3 Skip-Gram Network Embedding (SNE)

2.3.1 Explanation of the Skip-Gram model for representation learning

In recent years, notable progress in natural language processing tasks is largely credited to the widespread utilization of deep learning structures and improved word representations [4]; [13][14]. In pursuit of refining the overall capacity and simplifying the intricacies of language models, a prevalent strategy involves representing words using dense vectors. In this scheme, words with similar meanings are encoded by vectors that exhibit similarity in their arrangements [15][16].

Among the various approaches for such representations, one method that has gained extensive popularity is word2vec, an abbreviation for Word-to-Vector. The widespread adoption of this technique can be attributed to two primary factors: its computational efficiency and its capability to capture intriguing analogical relationships present in the language [17]. Notably, systems built upon the foundation of word2vec representations consistently yield substantial enhancements in performance. This underlines the profound impact that word2vec-based systems have on augmenting the outcomes of various language-related tasks.

According to Kipf and Welling [2], the skip-gram model employed by word2vec can be elucidated as follows:

Imagine we have a text training corpus with T words: $\{w_1, \cdots, w_T\}$.The provided dataset serves as the foundation for constructing a lexicon, designated as $\mathcal{D} = \{w_1, \cdots, w_W\}$,. This lexicon organizes words according to their prevalence within the dataset, with the arrangement reflecting their respective frequencies. For every term w present in the lexicon

$\mathcal{D}$, two distinct vector representations are allocated: the input vector, often referred to as word embedding v_w, and the output vector, known as context embedding v'_w, These vectors are generated randomly, drawing from a normal distribution, as elucidated by Sen in 2008 [18]. This process ensures the creation of unique and initially arbitrary vector representations for each term in the lexicon, forming the basis for subsequent computational operations and analyses.

These embeddings allow the computation of the conditional probability, $p(w_O \mid w_I)$, using the Softmax function, which quantifies the likelihood between any pair of words w_I and w_O in the lexicon $\mathcal{D}$. This approach establishes a foundational framework for analyzing semantic relationships within the corpus:

$$\hat{p}(w_O \mid w_I) = \frac{\exp(v'_{w_O}{}^T v_{w_I})}{\sum_{w=1}^{W} \exp(v'_w{}^T v_{w_I})}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(i)$$

The skip-gram model aims to maximize, where v'^T denotes the transformation of vector v', and $v'^T v$ represents the inner product of the vectors v' and v:

$$E = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c\leq j\leq c, j\neq 0} \log \hat{p}(w_{t+j} \mid w_t) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(ii)$$

The expression pertains to the radius of the context window with the center removed at w_t, denoted by the variable c.

It is important to highlight that in our notation, we employ $\hat{p}$ to denote the estimated probability derived from the vectors, deviating from the conventional skip-gram model literature notation, which typically uses just p. In the following sections, we will thoroughly examine the analysis of both the estimated probability, denoted as $\hat{p}$, derived from the vectors, and the ground truth probability represented as p, extracted from the training corpus. This differentiation in notation is made in anticipation of upcoming discussions where we will delve into the nuances and implications of these probability estimates.

Since the computational complexity of formula (i) is considerable, involving $O(W)$ inner products and exponentials, researchers often opt not to directly apply it in practical scenarios. Instead, simplified iterations of formula (1) find widespread use in various applications, as observed in the works of Ahmed [19] and Sen [18]. The adoption of these simplified versions allows for more efficient computational processes, rendering the model more feasible and applicable to real-world situations.

However, Mikolov et al. [20] introduced an efficient and effective method for approximating $\hat{p}(w_O \mid w_I)$. Rather than computing $\sum_{w=1}^{V} \exp(v'_{w_T} v_{w_I})$ across the entire vocabulary, they used $\sum_{k=1}^{K} \exp(v'_{w_k} T_{w_I})$ as an approximation. This involves randomly selecting words $\{w_1, w_2, \cdots, w_K\}$ from the distribution $P(w), where K$ is approximately $2-5 for large datasets and 5-20$ for smaller ones. The preferred distribution $P(w)$ is the $unigram distribution U(w) raised to the power$ of $3/4$, denoted as $P(w) = U(w)^{3/4}/Z$..

Thus, to maximize $\log \hat{p}(w_O \mid w_I)$, the goal is to maximize $v'_{w_O}{}^T v_{w_I}$ and minimize $\sum_{k=1}^{K} \exp(v'_{w_k}{}^T v_{w_I})$,. This is why the terms $w_1, w_2, \cdots, w_K$ are referred to as negative samples. Additionally, in practical applications, the exponential function is often substituted with the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$ to prevent underflow. In summary, Mikolov et al. [21] aim to maximize this expression:

$$\log \sigma(v'_{w_O}{}^T v_{w_I}) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P(w)} \log \sigma(-v'_{w_k}{}^T v_{w_I}), \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(iii)$$

for $w_I = w_t$ and $w_O = w_{t+j}$ in the formula (ii).

Utilizing Formula (iii) on word embeddings establishes the foundational language model for neural network training. Notably, the skip-gram model, particularly when incorporated with negative sampling (SGNS), produces highly significant and meaningful results, enhancing the efficacy of the language model. Utilizing pre-trained word

embeddings from models employing SGNS not only demonstrates high performance across various NLP tasks but also reveals intriguing analogical relationships [3].

Conversely, by examining Formula (iii), it becomes evident that the technique of negative sampling itself lacks any mystical properties; rather, it offers a straightforward and efficient means to approximate the conditional probability $\hat{p}(w_{t+j} \mid w_t)$.. Our assertion is that the effectiveness of the SGNS model can be attributed to the interplay of the skip-gram algorithm and the preference for dense vector embeddings over one-hot vectors when representing words. By employing Formula (iii), the skip-gram model facilitates the convergence of embedding vectors associated with words sharing similar contexts in the vector space. This synergy between the skip-gram algorithm and the use of dense vector embeddings enhances the model's capacity to capture nuanced semantic relationships among words, ultimately contributing to its overall efficacy.This convergence facilitates the encapsulation of semantic information and analogical relationships within the distribution of vectors, contributing to the model's capacity to capture nuanced linguistic nuances and improve the overall quality of word embeddings.

2.3.2 Learning Rules and Relationships in the Skip-Gram Model

Within this section, our objective is to gain a comprehensive understanding of how the skip-gram model learns from the provided data. Our methodology initiates by formulating the gradient equation for both input and output vectors, offering a detailed examination of the intricate relationships existing between the skip-gram model and the competitive learning mechanisms at play. By unraveling the gradient equations, we aim to shed light on the intricate dynamics that govern the learning process of the skip-gram model, thereby providing valuable insights into its underlying mechanisms.This exploration delves into the underlying dynamics of how the model adapts and refines its representations through the learning process, shedding light on the interplay between input and output vectors and their crucial role in shaping the model's understanding of the intricate relationships within the given dataset.

$We express the average log probability E in a different ways as follows$

$$
\begin{aligned}
E &= \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log \hat{p}(w_{t+j} \mid w_t) \\
&= \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\left(v'^{T}_{w_{t+j}} v_{w_t} - \log\left(\sum_{w=1}^{W}\exp\left(v'^{T}_{w} v_{w_t}\right)\right)\right) \qquad \ldots\ldots\ldots\ldots (iv) \\
&= \frac{1}{T}\sum_{t=1}^{T}\left(\left(\sum_{-c\le j\le c, j\neq 0} v'^{T}_{w_{t+j}} v_{w_t}\right) - 2c\log\left(\sum_{w=1}^{W}\exp\left(v'^{T}_{w} v_{w_t}\right)\right)\right)
\end{aligned}
$$

$For a fixed word w_s in the vocabulary, the gradient of its input vector w_s can be written as:$

$$
\begin{aligned}
\frac{\partial E}{\partial v_{w_{s,i}}} &= \frac{1}{T}\sum_{t=1, w_t=w_s}^{T}\sum_{c\le j\le c, j\neq 0}\left(v'_{w_{t+j,i}} - \frac{\sum_{w=1}^{W}\exp\left(v'^{T}_{w} v_{w_t}\right) v'_{w_i}}{\sum_{w=1}^{W}\exp\left(v'^{T}_{w} v_{w_t}\right)}\right) \\
&= \frac{1}{T}\sum_{t=1, w_t=w_s}^{T}\sum_{c\le j\le c, j\neq 0}\left(v'_{w_{t+j,i}} - \sum_{w=1}^{W}\frac{\exp\left(v'^{T}_{w} v_{w_s}\right)}{\sum_{\tilde{w}=1}^{W}\exp\left(v'^{T}_{\tilde{w}} v_{w_s}\right)} v'_{w_i}\right) \qquad \ldots\ldots\ldots\ldots (v) \\
&= \frac{1}{T}\sum_{t=1, -c\le j\le c, w_t=w_s}^{T}\sum_{\substack{j\neq 0 \\ \text{ind}}}\left(v'_{w_{t+j,i}} - \sum_{w=1}^{W}\hat{p}(w \mid w_s) v'_{w_i}\right).
\end{aligned}
$$

$Hence, the gradient formula for the entire input vector v_{w_s} can be written like this:$

$$\frac{\partial E}{\partial v_{w_s}} = \frac{1}{T}\sum_{t=1,-c\le j\le c,w_t=w_s}^{T}\left(v'_{w_{t+j}} - \sum_{w=1}^{W}\hat{p}(w \mid w_s)v'_w\right)$$
$$= \frac{1}{T}\sum_{t=1,w_t=w_s}^{T}\sum_{-c\le j\le c,j\neq 0,w_{t+j}=w}\left(v'_w - \sum_{\tilde{w}=1}^{W}\hat{p}(\tilde{w} \mid w_s)v'_{\tilde{w}}\right) \quad \dots\dots\dots\dots(vi)$$
$$= \frac{1}{T}\sum_{t=1,w_t=w_s}^{T}\sum_{\substack{-c\le j\le c\\ j\neq 0,w_{t+j}=w}}\left(\left(1-\hat{p}(w \mid w_s)\right)v'_w - \sum_{\tilde{w}=1,\tilde{w}\neq w}^{W}\hat{p}(\tilde{w} \mid w_s)v'_{\tilde{w}}\right)$$

$This means that the word at the position $t+j$ in the training data is the same as the $word w$ in the vocabulary $when w_{t+j} = w$.

The primary objective of the skip-gram model is to maximize the average log probability denoted as E. Although, in practical implementations, researchers commonly use gradient descent to minimize $-E$ for programming convenience, a comprehensive theoretical understanding demands the application of gradient ascent concerning $\partial E/\partial v_{w_s}$ to augment E within our learning rule. Therefore, the learning rule governing the update of the input vector v_{w_s} is explicitly defined as follows. This formulation establishes a theoretical basis for comprehending how the skip-gram model optimizes the log probability and iteratively refines the input vectors throughout the learning process. This theoretical foundation provides valuable insights into the underlying mechanisms through which the skip-gram model enhances its performance and refines its representation of input vectors over successive iterations.

$$v_{w_s}^{\text{new}} = v_{w_s}^{\text{old}} + \frac{\eta}{T}\sum_{\substack{t=1,-c\le j\le c\\ w_t=w_s}}^{T}\sum_{j\neq 0,w_{t+j}=w}\left(\left(1-\hat{p}(w \mid w_s)\right)v'_w - \sum_{\tilde{w}=1,\tilde{w}\neq w}^{W}\hat{p}(\tilde{w} \mid w_s)v'_{\tilde{w}}\right) \dots\dots\dots\dots(vii)$$

The parameter η represents the learning rate in the context.

In simple terms, when a vector $\vec{b}$ is added to another vector $\vec{a}$, the latter moves in the direction of $\vec{b}$ or experiences a reduction in the angle between them. Conversely, $subtracting vector \vec{b} from vector \vec{a}$ (i.e. $\vec{a} - \vec{b}$) causes $\vec{a}$ to move away from $\vec{b}$, $increasing the angle$ between the $two vectors$.Upon scrutinizing $the terms within the large bracket of formula (4)$, it is evident that both the term $\left(1-\hat{p}(w \mid w_s)\right)$ and each $\hat{p}(\tilde{w} \mid w_s)$ are consistently positive, given that $0 < \hat{p}(w \mid w_s) < 1$ for any word w. In essence, this implies that, on an intuitive level, the vector $\left(1-\hat{p}(w \mid w_s)\right)v'_w$ is is added to v_{w_s}, while vectors $\hat{p}(\tilde{w} \mid w_s)v'_{\tilde{w}}$ for all $\tilde{w} \neq w$ are subtracted from v_{w_s}. Consequently, during gradient ascent, the input vector v_{w_s} tends to move towards the output vector v'_w corresponding to the word w present in the context window of $word w_s$. Simultaneously, the gradient ascent causes $v_{w_s} to move away from all other output vectors v'_{\tilde{w}}$ except for v'_w.

This mechanism introduces a distinctive form of competitive learning. When a word w emerges in the context of the word w_s, it engages in competition with all other words to draw the input vector w_s toward its designated output vector v'_w. It is essential to underscore a noteworthy contrast between the gradient ascent employed in the skip-gram model and the conventional winner-takes-all (WTA) algorithm typically associated with competitive learning.In the skip-gram model, the competitive learning process involves words vying for influence over the input vector based on their contextual associations. Unlike the conventional winner-takes-all approach, where only the most influential element claims dominance, the skip-gram model employs a more nuanced mechanism. Multiple words can contribute to the adjustment of the input vector, reflecting a collaborative influence rather than a singular winner. This distinction underscores the sophistication and adaptability of the skip-gram model's competitive learning strategy in capturing the intricate relationships between words and their contexts.

In the context of backpropagation in the winner-takes-all (WTA) paradigm, a distinctive feature is the setting of gradients for all losing neurons to zero, leading to a situation colloquially described as "the winner takes all while the losers stand still" [20]. Contrasting this with the gradient ascent mechanism employed in the skip-gram model, the dynamics for the non-winning neurons (comprising all words w ˜ other than w) take a more challenging turn. These non-winning words not only fail to advance their positions but actively contribute to pushing the input vector v_{w_s} away

from their respective output vectors$v_{\tilde{w}}^{'}$. In essence, the competitive learning rule governing input vector updates in the skip-gram model can be encapsulated as "the winner strengthens its advantage, while the losers experience an even more pronounced setback." This distinction highlights the contrast in outcomes between WTA's backpropagation, where losing neurons remain static, and the skip-gram model's gradient ascent, where non-winning words play a role in actively displacing the input vector from their associated output vectors. The nuanced nature of this competitive learning rule contributes to the skip-gram model's ability to refine word embeddings and capture intricate semantic relationships in a dynamic manner.

This distinctive approach in the skip-gram model's competitive learning mechanism underscores the nuanced dynamics of how the model refines its representations. Unlike the WTA algorithm's winner-takes-all approach, where losing neurons remain static, the skip-gram model's losers actively contribute to pushing the input vector away from their associated output vectors. This nuanced interplay enhances the model's adaptability, fostering a more dynamic and responsive learning process.

The application of $SGNSalignswithouranalysis$: $Maximizingformula$ (3) involves optimizing $theinnerproductv_{w_0}^{'T}v_{w_I}$, simultaneously minimizing the $innerproductsv_{w_k}^{'T}v_{w_I}$ for all words w_k. This signifies that the $inputvectorv_{w_I}$ corresponding to the word w_I is drawn closer to the output vector $v_{w_0}^{'}$, when the word w_0 is part of the context of w_I. At the same time, v_{w_I} is $pushedawayfromtheoutputvectorsv_{w_k}^{'}$,, $wherew_k$ represents $randomlyselectedwords(negativesamples)$. Essentially, the $negativesamples$ serve to simulate all "loser" words $\tilde{w} \neq w_0$. As it is impractical to compute $\hat{p}(\tilde{w} \mid w_I)$ for all words $\tilde{w}$ other than w_0, SGNS randomly selects a subset of words to act as "losers," enabling the differentiation of the winning word w_0[10].

As a result of the gradient ascent update, the input vector v_{w_s} undergoes a progressive alignment with the output vectors $v_w^{'}$ corresponding to words w present in the context of w_s, Simultaneously, it diverges from the output vectors $v_{\tilde{w}}^{'}$ associated with words $\tilde{w}$ not present in the context. This dynamic mechanism facilitates the capture of semantic information by strategically positioning embedding vectors within the vector space, a concept well-supported by previous studies [13]; [14]. The evolving alignment and divergence contribute to the model's ability to discern and represent nuanced semantic relationships.

Likewise, in the case of $anoutputvectorv_{w_s}^{'}$, $thegradientofE$ with respect to each of its dimensions is expressed as:

$$\frac{\partial E}{\partial v_{s,i}^{'}} = \frac{1}{T}\sum_{t=1}^{T}\left(\left(\sum_{-c\leq j\leq c, j\neq 0, w_{t+j}=w_s} v_{w_{t,i}}\right) - 2c\frac{\exp(v_{w_s}^{T}v_{w_t})v_{w_{t,i}}}{\sum_{\tilde{w}=1}^{W}\exp(v_{\tilde{w}}^{T}v_{w_t})}\right)$$

$$= \frac{1}{T}\sum_{t=1}^{T}\left(n_{t,c,w_s}\cdot v_{w_{t,i}} - 2c\cdot\hat{p}(w_s \mid w_t)\cdot v_{w_{t,i}}\right) \quad \ldots\ldots\ldots\ldots (ix)$$

$$= \frac{1}{T}\sum_{t=1}^{T}\left(n_{t,c,w_s} - 2c\cdot\hat{p}(w_s \mid w_t)\right)v_{w_{t,i}}$$

Within this context, n_{t,c,w_s} signifies the frequency of occurrences of the word w_s within a window of radius-c, excluding the central word w_t. Much like the gradient of the input vector v_{w_s}, the term $\hat{p}(w_s \mid w_t)$ represents an estimation of the conditional probability $p(w_s \mid w_t)$ derived from the existing set of vectors $\{v_w, v_w^{'}\}_{w=1}^{W}$. Consequently, the all-encompassing gradient formula for the entire output vector $v_{w_s}^{'}$ can be articulated as follows:

$$\frac{\partial E}{\partial v_{w_s}^{'}} = \frac{1}{T}\sum_{t=1}^{T}\left(n_{t,c,w_s} - 2c\cdot\hat{p}(w_s \mid w_t)\right)v_{w_t}. \ldots\ldots\ldots\ldots (x)$$

This formula integrates the information regarding the frequency of word occurrences in the specified context window, along with the estimated conditional probability based on the current set of vectors. It provides a comprehensive expression for the gradient of the output vector, encapsulating the interplay of contextual information and the model's estimation of conditional probabilities in the learning process. This gradient formula serves as a crucial

element in the optimization process, guiding the adjustments of output vectors to enhance the overall performance of the skip-gram model.

Consequently, the rule for updating the output vector v'_{w_s} through gradient ascent is as follows:

$$v_{w_s}^{\text{new}} = v_{w_s}^{\text{old}} + \frac{\eta}{T}\sum_{t=1}^{T}\left(n_{t,c,w_s} - 2c \cdot \hat{p}(w_s \mid w_t)\right)v_{w_t} \dots\dots\dots\dots (xi)$$

Examining formula (5) reveals that the softmax definition $\hat{p}(w_s \mid w_t) = \exp(v'^{T}_{w_s} v_{w_t}) / \left(\sum_{\tilde{w}=1}^{W} \exp(v'^{T}_{\tilde{w}} v_{w_t})\right)$ implies that $\hat{p}(w_s \mid w_t)$ will generally be small (less than 10^{-3}) for most word pairs (w_s, w_t). The typical window size c ranges from 3 to 5, meaning that multiplying $2c$ will not significantly increase $2c \cdot \hat{p}(w_s \mid w_t)$). As a result, the expression $n_{t,c,w_s} - 2c \cdot \hat{p}(w_s \mid w_t)$ will typically yield a positive value when the word w_s appears within the context window of w_t, given that n_{t,c,w_s} is at least one in such instances. Conversely, if w_s is absent from the context window of w_t, n_{t,c,w_s} will be zero, causing the expression $n_{t,c,w_s} - 2c \cdot \hat{p}(w_s \mid w_t)$ to be negative.

In essence, when the word $w_t = w$ at position t in the training corpus encompasses w_s within its context window, it effectively "attracts" the output vector v_(w_s)^' of the word w_s towards its own input vector $v_{w_t} = v_w$ Conversely, when w_s is not present in the context window of w_t, the word $w_t = w$ must actively "repel"v'_{w_s}^' away from its own input vector v_w. This dynamic mirrors a competitive learning process, where each word $w_t = w$ in the training corpus competes with others to influence the movement of the output vector v'_{w_s}^' towards its own input vector $v_{w_t} = v_w$. Failing to do so results in the word w_t=w losing in this competition, causing v'_{w_s}^' to move away from v_w. This competitive learning rule aligns with the principle of "winners capitalize on their gains, while losers experience further losses."As a consequence, words with w_s in their context window emerge victorious in influencing the output vector, while those without w_s in their context window face defeat, leading to a dynamic and adaptive adjustment of the model's parameters during the learning process.Failure to do so results in the word $w_t = w$ losing in this competition, leading to v'_{w_s} moving away from v_w. This competitive learning rule follows the principle of "winners capitalize on their gains, while losers experience further losses." Consequently, words with w_s in their context window emerge victorious, whereas those without w_s in their context window face defeat.

Upon revisiting formula (3), it becomes evident that in each training step, there exists only one distinct input vector v_{w_I}. Consequently, the role played by w_I should be perceived as "multifaceted": it emerges as the "victor" in relation to the actual word w_O, yet it assumes the position of the "defeated" entity when compared to all the negatively sampled words [22]. It is crucial to recognize that the competitive learning mechanism dictating the modification of the output vector in the skip-gram model undergoes a notable transformation in the Skip-Gram with Negative Sampling (SGNS). In more straightforward terms, SGNS introduces a bias towards the input vectors, diverging from the theoretical expectation that the roles of input and output vectors in the skip-gram model should be comparable, as proposed by Mnih and Hinton in 2009 [23].

In essence, while the skip-gram model typically maintains a symmetrical treatment of input and output vectors in competitive learning, SGNS introduces a departure from this symmetry by exhibiting a bias towards the input vectors. This shift in the competitive learning mechanism has implications for how the model adapts and refines its representations during training. The theoretical anticipation of input and output vector equality in skip-gram is challenged by the introduced bias in SGNS, emphasizing the need to consider the nuanced variations between these two approaches for a comprehensive understanding of their respective dynamics and performance implications.

2.4 Self-Coding Model for Node Classification

In the realm of node classification within complex networks, self-coding models have emerged as a novel and promising approach. These models fundamentally revolve around the concept that nodes within a network inherently encode valuable information about themselves and their local interactions. Self-coding approaches prioritize the examination of a node's immediate neighborhood, emphasizing the importance of considering both the structural connections and attributes of neighboring nodes for classification tasks [24]. While these models hold significant promise in their ability to leverage intrinsic node properties for classification, they also face potential limitations and challenges.

Self-coding models place great emphasis on the encoding of information within each node, treating them as self-contained entities. This approach allows the models to effectively capture the unique features, characteristics, and

context of individual nodes within the network [25]. By doing so, self-coding models aim to unveil the intricate structural dependencies within the network, including the propagation of information, influence dynamics, and the diffusion of attributes [29]. However, one potential limitation of self-coding models lies in their reliance on local neighborhood information. While this approach enables them to capture fine-grained details of node interactions, it may also limit their ability to capture global network properties and long-range dependencies. Additionally, self-coding models may face challenges in dealing with noisy or incomplete data, as they heavily rely on the information encoded within individual nodes. Noise or missing information in node attributes or connections could lead to inaccuracies in classification outcomes and hinder the model's performance. Moreover, the computational complexity of self-coding models may pose challenges, particularly for large-scale networks, requiring substantial computational resources and time for training and inference. Thus, while self-coding models offer promising avenues for node classification in complex networks, addressing these limitations and challenges is crucial to realizing their full potential in practical applications.

2.4.1 Utilization of Dynamic Network Characteristics

Dynamic network characteristics play a pivotal role in self-coding models, enabling them to adapt to the evolving nature of network structures and node behaviors. These dynamic characteristics are harnessed in several key ways:

Temporal Information: Self-coding models thoughtfully incorporate temporal aspects, such as the timing of interactions or updates to node attributes. This incorporation allows the models to discern and account for temporal dependencies between nodes. It further aids in distinguishing between short-term and long-term effects within the network, thereby contributing to more nuanced and context-aware node classification [5].

Dynamic Embeddings: Dynamic network characteristics underpin the generation of dynamic node embeddings that evolve alongside the network. These embeddings are instrumental in capturing the changing roles and positions of nodes, facilitating more precise node classification. They are particularly adept at representing the evolving nature of network relationships and behaviors [30].

Influence Propagation: Self-coding models leverage dynamic network characteristics to model influence propagation and information diffusion. This modeling is vital for understanding the persistent impact of influential nodes over time. Such nodes may continue to exert influence on their neighbors, and self-coding models adeptly account for these lasting effects, thereby enhancing the accuracy of node classification [24].

Adaptive Learning: Adaptability is a hallmark of self-coding models. These models employ adaptive learning mechanisms that fine-tune their representations and predictions as the network undergoes changes. This adaptability ensures that the models remain relevant and effective in dynamic network scenarios, where structures and behaviors are in a constant state of flux [26].

Self-coding models in node classification offer a research-driven avenue for comprehending the dynamic complexities of complex networks. These models excel at encoding node information, considering local interactions, and harnessing dynamic network characteristics. Consequently, they hold the promise of delivering more accurate, context-aware, and adaptable node classification methodologies, particularly in environments where network structures and node behaviors exhibit temporal evolution.

2.5 Encoder Gated Recurrent Unit (ENGRU)

The Encoder Gated Recurrent Unit (ENGRU) is an innovative algorithm that integrates the principles of self-coding and Recurrent Neural Networks (RNNs). To understand ENGRU's significance, it's essential to first delve into the fundamentals of RNNs, particularly Gated Recurrent Units (GRUs)[11].

2.5.1 Recurrent Neural Networks (RNNs) and GRUs

Recurrent Neural Networks (RNNs) belong to the domain of artificial neural networks, specifically designed to handle sequential data processing. Renowned for their prowess in tasks involving sequences, such as time series analysis and natural language processing, RNNs excel in capturing dependencies and patterns over time. Their utility extends to

dynamic network analysis, showcasing their effectiveness in unraveling intricate relationships and evolving patterns within dynamic systems [11].

RNNs feature a distinctive architecture that enables them to retain memory of previous inputs, rendering them well-suited for tasks where context and temporal dependencies are pivotal. In the realm of dynamic network analysis, where interactions and connections among entities evolve over time, the inherent sequential processing capabilities of RNNs render them invaluable. Their aptitude for capturing temporal dependencies fosters a deeper understanding of the evolving nature of networks, positioning RNNs as a beneficial choice for analyzing and modeling dynamic systems.

A significant challenge encountered in training conventional RNNs is the issue of vanishing gradients, which impedes the networks' capacity to capture long-range dependencies in sequences. Gated Recurrent Units (GRUs) were introduced as a remedy to address this problem. GRUs represent a subtype of RNN architecture that incorporates gating mechanisms to regulate information flow within the network. These gates, comprising reset and update gates, empower GRUs to selectively retain or discard information from previous time steps. This selective mechanism enhances their ability to model long-range dependencies more effectively compared to traditional RNNs [27].

2.5.2 Description of the ENGRU Algorithm

The Encoder Gated Recurrent Unit (ENGRU) algorithm represents an innovative integration of self-coding principles with the power of RNNs, specifically GRUs. ENGRU leverages the capabilities of both self-coding and RNNs to enhance node classification in dynamic networks. Here's an overview of the ENGRU algorithm:

1. **Node Self-Coding:** ENGRU begins by encoding each node's attributes and structural information, treating nodes as self-contained entities similar to self-coding models. This encoding is essential for capturing the unique characteristics of each node within the dynamic network [7].
2. **Sequential Information:** ENGRU incorporates the sequential nature of dynamic networks. It considers the temporal evolution of the network by processing nodes and their encoded information sequentially over time. This temporal aspect is a critical element for capturing the dynamic changes in network structures and behaviors [19].
3. **GRU Integration:** ENGRU employs Gated Recurrent Units (GRUs) to model the temporal dependencies within the dynamic network. GRUs are well-suited for this task due to their ability to capture long-range dependencies while avoiding vanishing gradient issues. The GRUs serve as the temporal processing units within ENGRU [27].
4. **Dynamic Embeddings:** Through the integration of self-coding and GRUs, ENGRU generates dynamic node embeddings. These embeddings evolve over time, capturing the changing roles and positions of nodes within the dynamic network. The dynamic embeddings are instrumental for accurate and context-aware node classification [31].
5. **Adaptive Learning:** ENGRU's adaptive learning mechanisms ensure that the model remains relevant and effective as the network evolves. It dynamically adjusts its representations and predictions to accommodate changing network structures and behaviors [28].

The Encoder Gated Recurrent Unit (ENGRU) algorithm represents a sophisticated fusion of self-coding principles and the power of Gated Recurrent Units (GRUs). By combining self-coding's ability to capture node-specific information with the temporal modeling capabilities of GRUs, ENGRU offers a potent approach for node classification in dynamic networks. This integration enables accurate, context-aware, and dynamic node classification, making ENGRU a promising algorithm for a wide range of applications in dynamic network analysis.

III. RESEARCH METHOD

3.1 Introduction

In this section, we describe the research methodology used for the implementation and evaluation of the Representation Learning Method employing Dual Autoencoders. Representation learning is a key aspect of machine learning, and Dual Autoencoders have shown promising results for learning meaningful representations from complex data structures.

This section explains the experimental framework that we adopted for applying Dual Autoencoders in the context of representation learning and assessing their effectiveness.

3.2 Conceptual Framework and Symbolic Representation

The first step of our research methodology is to explain the essential concepts related to the problem domain. We establish the mathematical framework that supports the design of our model, which involves selecting the appropriate algorithms and parameters for our representation learning method. We introduce the Attribute Network Representation Learning, a core component of our research paradigm, which aims to learn low-dimensional vector representations of nodes in attribute networks by combining structural and attribute information.

3.3 Data Preprocessing

The next step of our research methodology is to focus on data preprocessing, which is crucial for ensuring the quality and robustness of our representation learning method. We use four real-world datasets from different domains, namely Cora, Citeseer, Pubmed, and wiki, which are publicly available and widely used for network representation learning tasks. These datasets consist of networks of nodes with various attributes and labels, representing scientific papers, blogs, or users. We preprocess the data by normalizing the attribute values, removing isolated nodes, and splitting the data into training, validation, and test sets. We also optimize the model's objective function, which consists of two terms: a reconstruction loss that measures the fidelity of the learned representations, and a regularization loss that encourages the representations to be smooth and discriminative.

3.4 Experimental Implementation and Analysis

The final step of our research methodology is to conduct experiments to evaluate the performance of our representation learning method. We use the experimental method as it allows us to test the real-world applicability of our model and to identify potential challenges and limitations that may not be captured by mathematical analysis or simulation-based approaches.

In the following sections, we provide more details on each aspect of our experimental implementation and analysis, and we discuss the main findings and implications of our experiments. Through this systematic presentation, we aim to offer a comprehensive understanding of the principles and practicalities involved in applying Dual Autoencoders for representation learning within the scope of our research.

IV. RESULTS

4.1.1 Network Representation using Dual Autoencoders and Attribute Information

To provide a clear understanding of the model and algorithm being proposed, this paper begins by defining relevant concepts and the primary symbolic representations used in the algorithm. The main symbols are listed below.

Table 1: Symbolic Notations and Their Meanings in Attribute Network Representation with Dual Autoencoders

Symbol	Meaning
V	refers to the collection of nodes within the network.
E	signifies the collection of edges connecting these nodes.
A	stands for the collection of node attributes
n	represents the total number of nodes in the network, denoted as, $\|V\|$
m	signifies the number of node attributes, represented as, $\|A\|$
X	is the attribute matrix, with dimensions of, $n \times m$
M	represents the adjacency matrix characterizing the network's connectivity
e_{ij}	denotes the weight associated with the connection between nodes v_i and v_j
v_i	stands for a node labeled as $i, v_i \in V$
d	signifies the dimension of the vector, a representation of the node, with d being significantly larger than t.
y_i	represents the vector that serves as the representation of node v_i
Y	denotes the matrix encompassing the representation vectors of all nodes within the network

In practical scenarios, real-world networks often manifest as graph structures, encompassing diverse examples like social networks, citation networks, and more. Within these tangible networks, nodes typically possess a set of associated attribute information, effectively constituting attribute vectors. For instance, in the context of social networks, a node corresponds to a user, and the attributes linked to this user may encompass details such as gender, age, location, and their network connections. Consequently, in the realm of network representation learning research, a network characterized by nodes enriched with diverse attribute information is commonly referred to as an "attribute network."

An attribute network can be conceptualized as $G = (V, E, A)$,, where V is the set of nodes $V = \{v_1, v_2, \cdots, v_n\}$, and n is the total number of nodes. The network encompasses edges (E) and attributes (A), providing a comprehensive framework for understanding the relationships and characteristics within the network.E signifies the collection of neighboring edges connecting these nodes, $E = \{e_{ij}\}$. A, on the other hand, designates the set of node attributes, $A = \{a_1, a_2, \cdots, a_m\}$, with m representing the number of attributes. The measure of attribute similarity between nodes (v_i, v_j) is determined by evaluating the resemblance between the attribute vector x_i of node v_i and the attribute vector x_j of node v_j. This assessment provides insights into the level of similarity in attributes between any given pair of nodes in the network.

In the process of representation learning for an attribute network, the fusion of the adjacency matrix and attribute matrix yields the ultimate low-dimensional representation vectors for nodes. These vectors serve as enriched inputs for machine learning algorithms, enabling effective handling of various network analysis tasks, including but not limited to node clustering and link prediction. This integration of attribute-based representations enhances the model's ability to discern patterns and relationships within the network, fostering more accurate and insightful analyses.

To express the concept of attribute network representation learning symbolically, for a given attribute network, we aim to acquire a mapping function denoted as $f: v_i \rightarrow v_j \in Y^d$, where '$i$' belongs to the set of nodes, V. The mapping function described is intricately tied to the structural and attribute features of individual nodes, where 'd' denotes the dimensionality of the resulting representation vector for each node. This connection emphasizes that the final representation of nodes is influenced by both the structural and attribute aspects, capturing a comprehensive view of the node's characteristics in the designated vector space.

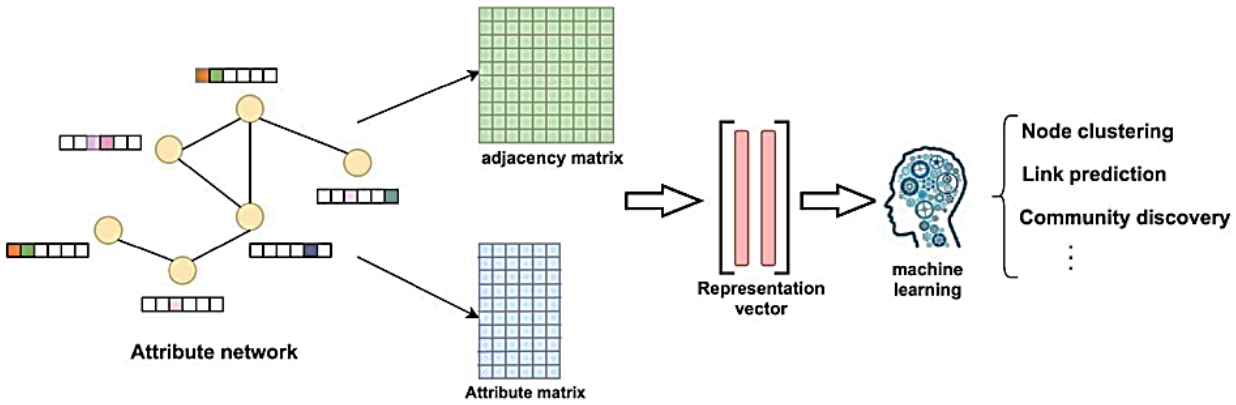

Figure 1: Node Classification in Attribute Networks using Machine Learning

Figure 1 illustrates how machine learning can be used to classify nodes in attribute networks based on their connections and attributes. The figure consists of five steps:

- Step 1: An attribute network is shown, where each node has a color-coded attribute and is connected to other nodes by edges.
- Step 2: The attribute network is converted into two matrices: an adjacency matrix that represents the connections between nodes, and an attribute matrix that shows the attributes of each node.
- Step 3: Both matrices are transformed into a representation vector, which captures the essential information from the matrices in a compact form.

- Step 4: The representation vector is fed into a machine learning model, which performs various tasks such as node clustering, link prediction, and community discovery.
- Step 5: The outcomes of the machine learning model are displayed, such as the clusters of nodes with similar attributes, the predicted links between nodes, and the communities of nodes that share common characteristics.

First-order proximity in a network refers to the local pairwise closeness between two vertices. When considering vertices v_i and v_j, first-order proximity between them is established if there is a direct edge connecting them, with the weight e_{ij} of the edge quantifying this proximity. Conversely, if there is no direct edge between the two vertices, there is no first-order proximity.

Besides, direct connections between vertices are relatively sparse in real-world networks. Consequently, even when two vertices exhibit significant similarities, their first-order proximity is nullified if they lack a direct connection. This omission presents a challenge, as these vertices are excluded from similarity assessments. Therefore, relying solely on first-order proximity is insufficient for preserving the structural integrity of the network.

To overcome this limitation, higher-order proximity becomes essential. This concept revolves around gauging the similarity in neighborhood network structures between vertices, acting as a crucial supplement to first-order proximity. It plays a pivotal role in preserving the overall structure of the network. If we represent the first-order proximity vector between vertex v_i and all other vertices as $N_i = (e_{i,1}, e_{i,2}, \cdots, e_{i,n})$, the higher-order proximity between vertices v_i and v_j hinges on the similarity between their respective N_i and N_j vectors. This approach extends the analysis beyond immediate neighbors, capturing more nuanced structural relationships within the network for a more comprehensive understanding.

4.1.2 Method for Learning Representations with Dual Autoencoders

This study builds upon Dual Autoencoders for Attribute Network Representation Learning (DANRL). It examines how the structure of a network and the traits of its nodes influence each other. To do this, it uses a dual-channel autoencoder setup that looks at both the network's structure and its attributes. One channel focuses on the structure, using the network's connections to determine relationships between nodes. It includes a mechanism that considers information from distant neighbors, aiding in understanding both local and global network structures. This helps create a vector representation that captures the complex relationships within the network.

In the other channel, the autoencoder works on attributes, using both node traits and network connections to create a filter. This filter gathers information from nearby nodes to build an attribute representation vector connected to the network's structure. The dual autoencoders then combine this information and feed it into a decoder to reconstruct a matrix representing the network's features. During training, node pairs are selected based on their similarity or dissimilarity, and both the structural and attribute encoders are trained together using a monitored loss function.

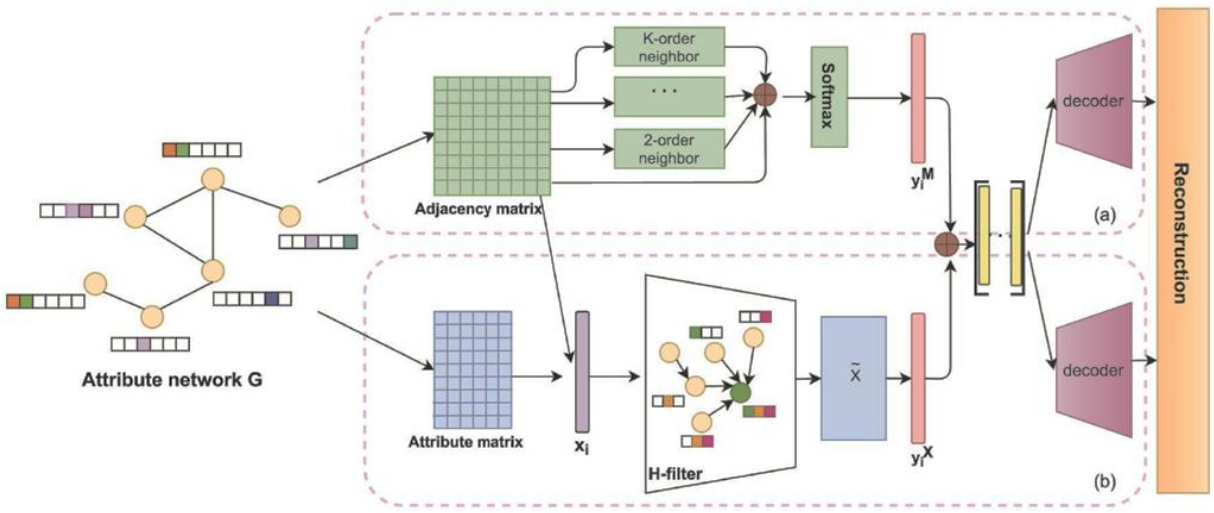

Figure 2: Attribute Network Reconstruction using Adjacency and Attribute Matrices

Diagram 2 shows how an attribute network can be reconstructed using two matrices: an adjacency matrix that captures the connections between nodes, and an attribute matrix that represents the attributes of each node. The diagram consists of the following steps:

- Step 1: An attribute network G is given, where each node has a specific attribute indicated by a colored bar, and is connected to other nodes by edges.
- Step 2: The attribute network G is converted into two matrices: an adjacency matrix that shows the presence or absence of edges between nodes, and an attribute matrix X_i that displays the attributes of each node in a tabular form.
- Step 3: The adjacency matrix is processed by a component called K-order neighbor, which analyzes the connections between nodes up to a certain distance (K). The output of this component is then passed to another component called 2-order neighbor, which further analyzes the connections between nodes up to a distance of 2. The output of this component is then fed into a softmax function, which normalizes the values and produces a vector y_i^M.
- Step 4: The attribute matrix X_i is processed by a component called H-filter, which filters out the irrelevant or noisy attributes and produces another vector y_i^X.
- Step 5: Both vectors $y_i^M$and y_i^X are inputted into two decoders, labeled as (a) and (b), respectively. These decoders are responsible for reconstructing or processing the vectors further. The final outcomes of the decoders are not shown in the diagram.

4.1.3 Enhancing Network Structure and Attribute Information Processing for Complex Attribute Networks

The network's inherent complexity poses several challenges, including cases where nodes lack first-order proximity yet exhibit similarities in network structures, and where the attribute data encompass diverse types. Furthermore, issues like missing or incomplete attribute information for some nodes further complicate the network analysis. As a result, this study commences with a comprehensive preprocessing of both node structure and attribute information.

In the context of attribute networks, the adjacency matrix functions as a comprehensive record of the first-order proximity, delineating the relationships and connections between nodes within the network. This matrix encapsulates the immediate associations and structural links, providing a fundamental representation of the network's connectivity.However, it's worth noting that first-order proximity primarily captures local network structures and may not fully represent similarities, especially when nodes share similar neighborhood structures without direct edges connecting them. For instance, in social-network communities, individuals may have common neighbors without direct connections. Hence, depending solely on first-order proximity is inadequate. To address this limitation, we leverage the column vector of the adjacency matrix to portray local structural information. Employing multi-hop attention-weighted summation, we extend our approach to encompass second-order neighbor information, enhancing the model's ability to capture more nuanced and intricate structural relationships within the network.

Node attribute information typically encompasses various data types, often without inherent size or order distinctions. Furthermore, there is no direct link between each attribute. To tackle this complexity, this study undertakes a unique encoding of attribute information, followed by the fusion of these encoded representations into the attribute vector representation of each node.As an illustration, consider node v_i; its attribute representation vector is denoted as a_i, where a_{ij} , signifies the encoded attribute vector associated with node v_i,, and the symbol $\oplus$ denotes the merging or concatenation process.

Hence we have

$$a_i = a_{i1} \oplus a_{i2} \oplus a_{i3} \oplus \dots \oplus a_{im} \qquad (xii)$$

To handle the challenge of missing or incomplete attribute information, traditional methods often resort to statistical approaches like imputation using mean or mode values. Another common method entails implementing a random adjustment mechanism, which introduces adjustments to input samples by randomly assigning missing attribute information of specific nodes to zero within the input vector.While these methods provide simplicity and transparency, they often neglect the structural intricacies of nodes, introducing biases during the integration of network information. This oversight highlights the need for more sophisticated methods that consider both attribute imputation and structural relationships within the network for a more accurate representation of node attributes.

Therefore, in this study, we propose a novel approach that integrates the first-order proximity of node structures with the node attribute matrix. This integrated framework facilitates the imputation of missing attribute information for nodes based on the attributes of their first-order neighboring nodes, thereby incorporating structural considerations into the imputation process.

Our study addresses the complex challenges posed by attribute networks by proposing a novel framework that integrates structural proximity with attribute information to handle missing or incomplete attribute data. By considering both structural relationships and attribute imputation, our approach offers a more accurate representation of node attributes in complex networks.

4.1.4 Structural Autoencoder

The structural autoencoder module, designed for unsupervised network representation learning, primarily focuses on the reconstruction of the adjacency matrix. To adeptly capture the intricate, nonlinear structural features of the network at both local and global levels, our approach incorporates a graph attention mechanism. This mechanism facilitates the learning of significance weights assigned to nodes and their respective neighbors, allowing for the aggregation of weighted message-passing processes. By leveraging this attention mechanism, the model enhances its ability to capture subtle structural nuances within the network, contributing to more effective representation learning.

Moreover, to augment the model's capability, the geometric distances between nodes are intricately computed in the peripheral Euclidean space of the embedding space. This supplementary computation contributes to a more comprehensive understanding of the structural relationships among nodes, facilitating a richer representation that incorporates both topological and geometric aspects within the network.Subsequently, these geometric distances undergo a sorting process, allowing for the incorporation of information pertaining to nodes that exhibit geometric proximity to one another into the aggregation operation.

Incorporating the graph attention layer into the learning process serves as a pivotal step in emphasizing the significance of comprehending neighboring nodes in the network.

$$\mathrm{e_{ij}} = \mathrm{attn}\left(\mathrm{y_i^M}, \mathrm{y_j^M}\right) = \sigma\left(\mu \cdot \left[W^{(1)}\mathrm{x_i^M} \oplus W^{(1)}\mathrm{x_j^M}\right]\right) \qquad (xiii)$$

In this context, the notation attn(·) signifies the presence of an attention layer, while μ and $W^{(1)}$ represent the parameters that undergo a learning process. The symbol $\oplus$ denotes the operation of vector splicing, and $\mathrm{e_{ij}}$ corresponds to the measure of importance assigned to the features of node v_j concerning node v_i. To facilitate a straightforward comparison of the importance weight coefficients across nodes, a normalization procedure is employed on $\mathrm{e_{ij}}$ using the softmax function as follows;

$$\gamma_{ij} = \mathrm{Softmax}_j(\mathrm{e_{ij}}) = \frac{\exp(\mathrm{e_{ij}})}{\sum_{k\in N_i}\exp(\mathrm{e_{ik}})} = \frac{\exp\left(\mathrm{Re}\,LU\left(\mu \cdot \left[W^{(1)}\mathbf{x}_\mathrm{i}^\mathrm{M} \parallel W^{(1)}\mathbf{x}_\mathrm{j}^\mathrm{M}\right]\right)\right)}{\sum_{k\in N_i}\exp\left(\mathrm{Re}\,LU\left(\mu \cdot \left[W^{(1)}\mathrm{x_i^M} \parallel W^{(1)}\mathbf{x}_\mathrm{k}^\mathrm{M}\right]\right)\right)} \qquad (xiv)$$

Incorporating information not only from the neighboring nodes directly connected by edges but also from nodes without direct edge connections, we apply a process known as multi-hop attention to propagate information through the graph. This process involves computing attention scores for multi-hop neighbors, which are determined based on the matrix M. where;

$$\mathrm{M} = \sum_{i=0}^{k} \theta_i \mathrm{M}^i \text{ where } \sum_{i=0}^{k} \theta_i = 1 \text{ and } \theta_i > 0 \qquad (xv)$$

Herein, θ_i represents the attenuation factor for attention weights, while M^i characterizes the length of the path between nodes, effectively expanding the reach of nodes within the acceptance domain. This domain involves nodes that share geometric proximity in the peripheral Euclidean space. We introduce a unique differential twisting function, which considers both node distance and node "proximity" within this peripheral Euclidean space. The function is designed to

capture the nuanced relationships between nodes, incorporating both spatial proximity and geometric features for a more comprehensive understanding of the network structure.

$$\rho = \frac{1}{\mathrm{n}(n-1)} \sum_{i \neq j} \frac{d_{\mathrm{M}}(\mathrm{x_i}, \mathrm{x_j})}{d_{\mathcal{E}}(\mathrm{x_i}, \mathrm{x_j})} = \frac{1}{\mathrm{n}(n-1)} \sum_{i \neq j} \frac{d_{\mathrm{M}}(\mathrm{x_i}, \mathrm{x_j})}{\sqrt{\sum_{i=1}^{n} (\mathrm{x_i} - \mathrm{x_j})^2}} \qquad (xvi)$$

Here, $d_{\mathrm{M}}(\mathrm{x_i}, \mathrm{x_j})$ denotes the proximity distance measure in a non-Euclidean space, while $d_{\varepsilon}(\mathbf{x_i}, \mathrm{x_j})$represents the distance between nodes in the peripheral Euclidean space. The parameter ρ acts as the "trade-off" factor between these two distances. A smaller ρ indicates a constrained trade-off between the non-Euclidean and Euclidean aspects, thereby emphasizing the preservation of node proximity information and geometrically similar node details. This parameter allows for a flexible adjustment, enabling the model to prioritize either the non-Euclidean or Euclidean characteristics based on the specific requirements of the network representation learning task.

Ultimately, the features of neighboring nodes located within the acceptance domain undergo weighting and summation:

$$\mathrm{Y_i^M} = \sum_{k \in \mathrm{M}} \left(\gamma_{ik} \cdot \mathrm{Y_k^M} + \gamma_{i\varepsilon} \cdot \mathrm{Y_\varepsilon^M} \right) \qquad (xvii)$$

Following the process of adaptive decoding, we acquire the representation of structural embeddings.

4.1.5 Attribute Autoencoder

The attribute autoencoder is a key component in the network representation learning framework, tailored to capture intricate and highly nonlinear attribute information associated with nodes. During the attribute learning phase, the encoder engages in feature mapping of the original node attributes within the network. To mitigate the influence of high-frequency noise in the node attributes, a Laplace smoothing filter is employed. This filter aids in generating embedded representations for these attributes, facilitating the extraction of essential patterns and nuanced information within the node attributes.Following this, the structural representation and attribute representation are judiciously combined, promoting interactive learning and maintaining coherence between them. This integration process culminates in the reconstruction of the node attribute matrix. To assess the smoothness of the attribute vector x within the graph, the initial step involves calculating the Rayleigh entropy with respect to both the graph Laplacian matrix $\mathrm{L}(\mathrm{L} = \mathrm{D} - \mathrm{M})$ and the attribute vector x:

$$R(\mathrm{L}, \mathrm{x}) = \frac{\mathrm{x^T L x}}{\mathrm{x^T x}} \qquad (xix)$$

and

$$\begin{aligned} &\mathrm{x^T L x} = \mathrm{x^T D x} - \mathrm{x^T M x} \\ &= \sum_i \mathrm{x}^2(v_i) d_i - \sum_i \sum_j \mathrm{M}_{ij}\, \mathrm{x}(v_i)\mathrm{x}(v_i) \\ &= \frac{1}{2}\left(\sum_i \mathrm{x}^2(v_i) d_i - 2 \sum_i \sum_j \mathrm{M}_{ij}\, \mathrm{x}(v_i)\mathrm{x}(v_j) + \sum_j \mathrm{x}^2(v_j) d_j \right) \qquad (xx) \\ &= \frac{1}{2} \sum_i \sum_j \mathrm{M}_{\mathrm{ij}} (\mathrm{x_i} - \mathrm{x_j})^2 \end{aligned}$$

This implies that neighboring nodes are expected to exhibit similar attribute values, and the greater the similarity, the smoother the attributes should appear. The outcome of the Rayleigh entropy computation yields the eigenvalue L, and the solution for x within the realm of $R(\mathrm{L}, \mathrm{x})$ corresponds to the eigenvector of L.

The traditional Laplace smoothing filter is conventionally defined as:

$$\mathrm{H} = \mathrm{I} - k\mathrm{L} \qquad (xxi)$$

The resulting filtered attribute vector, denoted as $\tilde{x}$, is as follows:

$$\tilde{\mathrm{x}} = \mathrm{Hx} = (\mathrm{I} - k\mathrm{L})\mathrm{x} = \sum_{i=1}^{n} (1 - k\lambda_i) p_i \mu_i = \sum_{i=1}^{n} p_i \quad {}' \mu_i' \qquad (xxii)$$

Here, μ_{i} represents the eigenvector of L, and p_i signifies the corresponding coefficient of the eigenvector.

The attribute vector matrix, following t-layer Laplacian filtering, can be expressed as $\tilde{X} = H^t X$.

In practical network analysis tasks, a symmetric normalized graph Laplacian matrix is typically employed. This involves utilizing D and L, which represent the degree matrix and the Laplace matrix, respectively, concerning the matrix M.

$$\tilde{\mathrm{M}} = \mathrm{I} + \mathrm{M} \qquad \text{(xxiii)}$$

$$\tilde{\mathrm{L}} = \tilde{\mathrm{D}}^{-\frac{1}{2}} \mathrm{L} \mathrm{D}^{-\frac{1}{2}} \qquad (xxiv)$$

As a result, the Laplace matrix is given by:

$$\mathrm{H} = \mathrm{I} - k\tilde{\mathrm{L}} \qquad (xxv)$$

For selecting appropriate values of k, we consider the maximum eigenvalues of L, denoted as $\lambda_m, k = 1/\lambda_m$. In the subsequent sections involving task evaluation and result analysis, we showcase the impact of different k values on the experimental outcomes.

In this research, we measure the similarity of attribute information between each pair of nodes within the attribute matrix after the application of smoothing and filtering, using cosine similarity calculations. The obtained similarity data between nodes is then recorded as follows:

$$\mathrm{S}_{ij}^{\mathrm{X}} = \mathrm{Cos\,Sim}(\tilde{\mathrm{X}}) = \frac{\mathrm{x_i x_j^T}}{|\widetilde{\mathrm{x_i}}||\tilde{\mathrm{x}}_{\mathrm{j}}|} \qquad (xxvi)$$

To analyze the distribution of shared attributes among different nodes, we initially identify the existence of directly connected edges between pairs of nodes using the adjacency matrix derived from the original network. Subsequently, we execute attribute encoding vector multiplications within the attribute matrix to discern the common attributes between two nodes. Similar to the structural encoder, this component of the encoder incorporates multiple layers of nonlinear functions.

4.1.6 Model Optimization

Within this study, the optimization objective function of the model is formulated as a concurrent optimization of both the reconstruction error associated with the structural autoencoder and that of the attribute autoencoder. The optimization loss function is articulated as follows:

$$L_{\mathrm{loss}} = L_{\mathrm{str}} + L_{\mathrm{attr}} = \min\left(\sum_{1}^{n} \|\mathrm{y_i^M} - \mathrm{y_i}\|_2^2 + \sum_{1}^{m} \|\mathrm{y_i^X} - \mathrm{y_i}\|_2^2\right) \qquad (xxvii)$$

Derived from the aforementioned elucidation of the model's constituent elements, we can construct a description of the DANRL algorithm, which is presented in Algorithm 1.

Algorithm 1: DANRL Algorithm

Input:

$$Attributenetwork G = (V, E, A), adjacency matrix M, attribute matrix X, filter layers t;$$

Output: Node representation matrix Y;

1. $Calculateneighbornodeimportanceweighte_{ij} from$ (1).
2. $Normalizefrom$ (2).
3. $forn = 2,3, \dots, ndo$
4. $CalculateEuclideandistanceandtwistpfrom$ (4).
5. $endfor$;
6. $Obtainnodestructureembeddedrepresentationy_i^M$;.
7. $ObtainLaplacianLfrom$ (11).
8. $k \leftarrow 1/\lambda_m$;
9. $GetfiltermatrixHfrom$ (12).
10. $Getthesmoothedattributematrix\bar{X}from$ (14).
11. $Calculatenodeattributesimilaritymatrix\bar{S}_ij\hat{}Xfrom$ (13).
12. $Obtainnodeattributeembeddedrepresentationy_i^x$;.
13. $forepoch = 1,2, \dots, customdo$
14. $Updateencoderparameters$.
15. $Calculatethejointoptimizationlossfunction$.
16. $endfor$.

V. CONCLUSION

This research presents aninnovative methodology termed the Dual Autoencoder Learning Method for Attribute Network Representation. This method integrates two autoencoder channels, each dedicated to distinct aspects of network representation. The initial channel utilizes a multi-hop attention mechanism, facilitating the assimilation of high-order neighborhood information for nodes. By computing importance weights for neighboring nodes, this mechanism ensures the inclusion of both local and global structural elements within the network, enhancing the model's capacity to capture intricate relationships across different scales. This innovative approach seeks to advance the effectiveness of attribute network representation learning through a dual-channel autoencoder framework.

In contrast, the second autoencoder channel adopts a low-pass filtering strategy, iteratively extracting attribute information from the neighborhood of nodes based on the network's structural characteristics. The dual autoencoder architecture facilitates the dedicated learning of both network structure and attributes. Subsequently, an adaptive fusion process brings these learned representations together, fostering interactivity and mutual influence between the two information types. This effectively mitigates the existing limitations in network representation learning, specifically the lack of interaction between structural and attribute information, as well as the absence of joint learning for node representation incorporating both forms of information.

5.1 Implications and Applications

The proposed Dual Autoencoder Learning Method for Attribute Network Representation (DANRL) holds several important implications and offers a wide range of applications across various domains:

1. Improved Network Representation:

DANRL addresses the limitations of traditional network representation methods by simultaneously capturing both structural and attribute information, thereby offering a more comprehensive representation of complex networks.Enhanced network representations are invaluable for numerous applications, including social network analysis, recommendation systems, and biological network analysis, where understanding both network structure and attributes is crucial.

2. Enhanced Machine Learning Models:

The Dual Autoencoder Learning Method for Attribute Network Representation (DANRL) exhibits substantial potential to enhance the efficacy of machine learning models dependent on network representations, including tasks like node classification, link prediction, and community detection. Its applicability extends to real-world scenarios such as fraud detection in financial networks, content recommendation in social networks, or the identification of disease-related genes in biological networks. By leveraging DANRL, these applications can benefit from improved accuracy and robustness in extracting meaningful patterns and relationships within diverse network structures.

3. Network Anomaly Detection:
DANRL's capacity to capture both local and global structural information positions it as a valuable tool for anomaly detection in networks. Its application extends to detecting fraudulent activities in financial transactions, identifying network intrusions in cybersecurity, or uncovering unusual behavior in social networks. By leveraging DANRL, these applications can benefit from a holistic understanding of network structures, enabling more accurate identification and timely intervention in anomalous patterns or activities.

4. Attribute Network Integration:
DANRL facilitates the integration of attribute information into network analysis, which is essential in scenarios where nodes possess rich attribute data.It finds utility in recommendation systems, personalized marketing, and customer segmentation, where incorporating user attributes can lead to more tailored and effective strategies.

5. Multimodal Data Fusion:
DANRL's adaptive fusion of structural and attribute embeddings allows for the integration of multimodal data sources, enhancing the representation of diverse data types.This can be applied in multimedia content analysis, sensor networks, and healthcare, where information from various sensors or modalities needs to be combined for a comprehensive analysis.

6. Scalable Network Analysis:
DANRL's ability to capture high-order neighborhood information improves the scalability of network analysis by reducing the computational complexity.It can be employed in large-scale social networks, web graphs, and transportation networks to extract meaningful insights without overwhelming computational demands.

In summary, the DANRL algorithm has significant implications for advancing network representation learning, and its applications span a wide spectrum of domains, ranging from improving machine learning models to solving real-world problems in various fields. Its ability to seamlessly integrate structural and attribute information makes it a valuable tool for researchers and practitioners in diverse domains seeking to extract meaningful insights from complex networks.

REFERENCES

[1] P. Hamilton, W. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems* 2017.

[2] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations* 2017.

[3] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2014.

[4] J. Turian, L. Ratinov, and Y. Bengio, "Word representations: A simple and general method for semi-supervised learning," in *Association for Computational Linguistics*, pp. 384-394, 2010.

[5] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93-106, 2008.

[6] T. Shinozaki, "Competitive Learning Enriches Learning Representation and Accelerates the Fine-tuning of CNNs," *arXiv:1804.09859v1 [cs.LG]*, 2018.

[7] T. N. Kipf and M. Welling, "Representation learning on graphs: Methods and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 11, pp. 2454-2472, 2021.

[8] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509-512, 1999.

[9] T. Mitchell, *Machine Learning*, McGraw-Hill Education, 1997.

[10] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577-8582, 2006.

[11] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *International Conference on Machine Learning* 2003.

[12] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2016.

[13] Y. Bengio, D. Schuurmans, J. L. Ba, and C. Szepesvari, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137-1155, 2003.

[14] E. Huang, R. Socher, C. Manning, and A. Ng, "Improving Word Representations via Global Context and Multiple Word Prototypes," *Association for Computational Linguistics*, 2012.

[15] E. Oja, "Neural networks, principle components, and subspaces," *International Journal of Neural Systems*, vol. 1, pp. 61-68, 1989.

[16] P. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *Journal of Artificial Intelligence Research*, vol. 37, pp. 141-188, 2010.

[17] P. Turney, "Distributional semantics beyond words: Supervised learning of analogy and paraphrase," *Transactions of the Association for Computational Linguistics (TACL)*, vol. 1, pp. 353-366, 2013.

[18] P. Sen, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93-106, 2008.

[19] M. Ahmed, X. Liao, and A. T. S. Chan, "Network embedding in dynamic graphs," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

[20] T. Mikolov et al., "Distributed representations of words and phrases and their compositionality," *Neural Information Processing Systems*, 2013.

[21] T. Mikolov et al., "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference*, pp. 5528-5531, 2011.

[22] T. Kocmi and O. Bojar, "An Exploration of Word Embedding Initialization in Deep-Learning Tasks," *arXiv:1711.09160v1 [cs.CL]*, 2017.

[23] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," *Advances in Neural Information Processing Systems*, vol. 21, pp. 1081-1088, 2009.

[24] M. Ahmed, X. Liao, and A. T. S. Chan, "Network embedding in dynamic graphs," *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining*, 2018.

[25] R. Monti, F. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[26] D. Zügner and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018.

[27] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

[28] A. Trivedi, N. C. Krishnan, and T. Suel, "Predicting temporal dynamics of social media cascades," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2017.

[29] R. Roman, R. Precup, and D. David, "Second Order Intelligent Proportional-Integral Fuzzy Control of Twin Rotor Aerodynamic Systems," *Procedia Computer Science*, vol. 139, pp. 372-380, 2018.

[30] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *International workshop on artificial intelligence and statistics*, 2005.

[31] X. Dong, L. Tang, Y. Chang, and X. Zhu, "Link prediction and recommendation across heterogeneous social networks," in *Proceedings of the Eighth International Conference on Web Search and Data Mining*, 2015.