Robust Neural Model for Searching over Incomplete Graphs

RADIN HAMIDI RAD, University of Toronto, Canada
EBRAHIM BAGHERI, University of Toronto, Canada
MEHDI KARGAR, Toronto Metropolitan University, Canada
DIVESH SRIVASTAVA, AT&T Chief Data Office, USA
JAREK SZLICHTA, York University, Canada

The task of searching over large keyword graphs aims to identify a subgraph where the nodes collectively cover the input query keywords. Although finding an exact solution to this problem is NP-hard, we address it by proposing a novel graph neural network representation learning technique specifically tailored for graphs with missing information. We propose a novel keyword graph representation learning method that incorporates complementary aspects of graphs: global, local, adjusted, and feature semantics. Considering these multiple aspects, our approach remains robust and resilient to missing information. We adopt and fine-tune a transformer-based model to aggregate the various features of a graph to generate rich representations, recognizing the pivotal role of keywords in this task. We show through experiments on real-world data that our method outperforms the state-of-the-art approaches and is particularly robust in the face of missing values, underscoring its ability to effectively handle incomplete graphs.

CCS Concepts: • **Information systems → Novelty in information retrieval**; **Question answering**.

Additional Key Words and Phrases: Graph Representation Learning, Search in Graph, Incomplete Graph, Graph Neural Network, Information Retrieval

ACM Reference Format:
Radin Hamidi Rad, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, and Jarek Szlichta. 2024. Robust Neural Model for Searching over Incomplete Graphs. *ACM Trans. Intell. Syst. Technol.* x, x, Article x (x 2024), 31 pages. https://doi.org/XXXXXXX.XXXXXXX

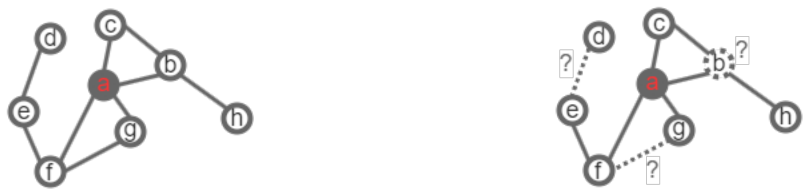
1 Introduction

With the increasing popularity of graphical structures for information representation, such as social networks, and knowledge graphs, the ability to perform effective graph search is becoming increasingly important. Keyword search over graphs enables the identification of a subgraph that covers the input keywords. Researchers have shown that this problem is NP-Hard [26]. Consequently, many tractable yet non-deterministic solutions for this problem have been developed to address this challenge. Conventional methods employ greedy or approximation approaches to find subgraph solutions for this problem, such as utilizing proximity ranking [5] and approximation search [26]. More recent approaches have incorporated graph embedding techniques, such as graph convolutional neural networks [29] and graph attention networks [52] to generate embedding vectors to approximate graph search [16, 41, 42, 48, 55].

Authors' Contact Information: Radin Hamidi Rad, radin.rad@utoront.ca, University of Toronto, Toronto, ON, Canada; Ebrahim Bagheri, ebrahim.bagheri@utoronto.ca, University of Toronto, Toronto, ON, Canada; Mehdi Kargar, kargar@torontomu.ca, Toronto Metropolitan University, Toronto, ON, Canada; Divesh Srivastava, divesh@research.att.com, AT&T Chief Data Office, USA; Jarek Szlichta, szlichta@yorku.ca, York University, Toronto, ON, Canada.

ACM 2157-6912/2024/x-ARTx
https://doi.org/XXXXXXX.XXXXXXX

These approaches do not however account for *missing* or *incomplete* information with a graph. An *incomplete graph* represents a set of nodes where some of their attributes and/or relationships between them are missing or unknown. This indicates that the dataset or the knowledge about the nodes and their relationships are only partially known. An abstract example of an incomplete graph and its comparison with the equivalent complete version of it can be seen in Figure 1. This incompleteness can result from various factors, including data collection limitations, privacy concerns, technical errors, or the inherent dynamic nature of the relationships being modelled [63]. Consequently, analyzing and interpreting incomplete graphs require special consideration.



(a) Graph G (b) Graph G with missing values

Fig. 1. A comparison between a graph without any missing values (a) and a graph with missing values (b). Incomplete graphs can be missing edges, nodes or attributes.

While many existing methods assume that graphs are fully observable and complete, in real-world applications, graphs often suffer from missing values, rendering them incomplete. For instance, researchers have extensively applied graph representation learning methods to tabular data [15, 53]. However, when translating tabular data into graphical structures, different forms of missing information, commonly referred to as *missing values* can manifest as missing edges, nodes or attributes [17].

Thus, the study of graphs with missing values is of critical importance [11, 57]. Some of the (non-comprehensive) reasons for graph incompleteness can be enumerated as follows: (1) *Technical errors:* Occur due to software bugs, hardware failures, or network disruptions, leading to missing nodes or edges [2]. (2) *Information confidentiality:* Results in intentional data omissions to protect privacy, such as anonymized or hidden relationships in social networks [64]. (3) *Data source complexity:* Arises from challenges in integrating heterogeneous sources with different formats and inconsistencies, often leading to missing information [35, 40]. (4) *The dynamic nature of data:* Fast-changing systems, such as social networks and financial markets, make it difficult to maintain an up-to-date graph [18, 36, 37]. (5) *Resource constraints:* Such as computational limitations, storage capacity, or restricted data access, can prevent comprehensive data collection, leading to partial graphs [10]. (6) *Intentional abstraction:* This is sometimes employed to simplify graphs for specific analyses, omitting less relevant details [7]. (7) *System evolution:* Leads to outdated graphs when networks change over time but are not continuously updated [1]. These factors collectively contribute to the incompleteness of graph representations across various domains.

In the context of methods that facilitate graph search, particularly the cutting-edge techniques that employ graph representation learning [14, 29, 52], the presence of missing information poses significant limitations. This is primarily because the majority, if not all, of these methods operate under the assumption that the underlying graphical structure is complete. Consequently, when dealing with an incomplete graph, the reliability of graph metapaths or message-passing techniques used for learning representations may be compromised. These methods are not designed to handle missing information and, therefore, lack robustness when confronted with such scenarios. As a result, the performance of state-of-the-art neural methods on incomplete graphs falls short of the desired levels [22].

Our research is motivated by the belief that a robust method for handling missing information should encompass multiple complementary aspects of the graph during the process of learning representations. Our approach emphasizes the importance of considering various aspects, including *local*, *global*, and *adjusted contexts* of nodes, along with the *semantic content* of nodes, in order to develop a robust graph representation learning method for incomplete graphs. By incorporating these different contextual factors, our approach enables the generation of robust and reliable representations using available subsets of data. The method effectively addresses the challenges posed by incomplete information, even in scenarios where only limited information is available for certain subgraphs.

The **main contributions of our work** are as follows:

- We propose a transformer-based architecture specifically designed for keyword search over *incomplete graphs*—a problem setting that remains underexplored compared to traditional graph completion or classification tasks. Our method is designed to operate effectively even when critical structural or semantic information is missing.
- Our approach exhibits robustness in the face of incomplete graph structures by capturing both local (micro), global (macro) and adjusted contexts of subgraphs, as well as node semantics. Thus, our model can generate reliable representations even when significant keywords or edges are missing from the graph.
  - Global Semantics: By using anchor nodes, our proposed approach captures the broader context of each node within the entire graph. This helps differentiate nodes that might appear similar locally but are situated differently globally.
  - Local Semantics: Utilizing the Weisfeiler-Lehman algorithm, our proposed approach captures the structural context of nodes within their immediate neighborhoods. This provides a detailed local view of the graph's structure.
  - Adjusted Semantics: A k-hop strategy helps in balancing the local and global contexts, creating a robust representation that considers both immediate neighbors and more distant nodes.
  - Feature Embedding: We propose to adopt an encoding method to transform sparse keyword occurrences into dense, low-dimensional vectors, capturing the interaction between keywords and preserving as much information as possible.
- Through extensive experiments over real-world data, we demonstrate that our novel method outperforms the current state-of-the-art techniques in keyword search over incomplete graphs. Our model showcases a more robust performance, even in cases where substantial information is missing. This can be attributed to its ability to leverage a range of complementary features for learning the graph representation, even in the presence of missing information.

The structure of this paper is as follows: Section 2 presents an extensive discussion of the related work and how we differ from them. Section 3 provides a clear definition of the problem and principles to search in incomplete graphs. Section 4 presents the proposed method, including model architecture and training. Section 5 presents our extensive experiments. Finally, Section 6 concludes this paper.

2 Related Work

The domain of incomplete graph search, particularly in the context of keyword search, has gained significant attention in recent years. Several methodologies have been proposed to handle missing information in graphs, spanning from traditional graph search algorithms to machine learning-based techniques. In this section, we categorize and analyze the key approaches in the literature, highlighting their strengths and limitations.

Traditional Graph Search Approaches. Traditional graph search algorithms are primarily designed for complete graphs with well-defined structures [23, 59]. These algorithms assume full visibility of all graph components, which makes them inefficient and inaccurate when faced with missing information. The main challenge in incomplete graphs lies in either inferring the missing elements or developing robust search strategies that can operate despite incomplete data. One notable example is BANKS (Browsing ANd Keyword Searching) [5], which models relational databases as graphs, where tuples represent nodes, and relationships form edges. It uses a proximity-based ranking system to enable schema-free keyword searches. Despite being widely adopted, its effectiveness relies heavily on the underlying database schema. Highly interconnected or complex schemas may reduce its efficiency.

Probabilistic Models for Incomplete Graph Search. Probabilistic models address the uncertainty in incomplete graphs by estimating the likelihood of missing connections or attributes. BLINK [23] (Bayesian-information and Linkage-disequilibrium method) optimizes keyword searches by leveraging advanced indexing and querying strategies. While these models improve search accuracy, they come with high computational costs and require complex implementations to maintain indexing structures efficiently.

Dimensionality Reduction Techniques. Dimensionality reduction techniques help manage large-scale, high-dimensional graph data. Principal Component Analysis (PCA) and Kernel PCA (KPCA) [19, 63] simplify the feature space, improving search accuracy. KPCA outperforms PCA by capturing non-linear relationships, making it more effective for complex incomplete graphs. However, these approaches still struggle with missing data, leading to unstable performance in highly incomplete graphs.

Machine Learning-Based Methods. Graph Neural Networks (GNNs) have emerged as powerful tools for handling incomplete graphs. These models learn latent representations of nodes and predict missing edges to improve keyword search. Graph Convolutional Networks (GCN) [29] use local node attributes and immediate neighbors to handle missing data but suffer from the oversmoothing problem, where nodes become indistinguishable after multiple layers. A variant, ChebConv [14], mitigates oversmoothing using Chebyshev polynomial expansion, but deeper architectures may still be affected. Graph Attention Networks (GAT) [52] introduce attention mechanisms to prioritize the most relevant neighbor nodes. This approach enhances feature aggregation and model interpretability, but it is computationally expensive, particularly for large-scale graphs. GraphSAGE [21] extends GNNs by generating node embeddings that incorporate both structural and attribute information. It enables inductive learning for unseen nodes, making it scalable. However, performance is highly dependent on the aggregation function and the size of sampled neighborhoods. Recently, T2-GNN [25] was proposed as a general framework to enhance the robustness of GNNs on incomplete graphs using a teacher-student distillation mechanism. Unlike prior methods, T2-GNN separately models feature and structure completion through independent teacher models, ensuring that missing features do not negatively impact structure learning and vice versa. A dual-distillation mechanism is then employed to transfer knowledge from the teacher models to the student GNN, improving its ability to learn representations despite missing information. However, T2-GNN is inherently a supervised method, relying on labeled data for effective knowledge transfer in its teacher-student distillation framework. This dependence on labeled data can limit its applicability in real-world scenarios where annotations are scarce or expensive to obtain.

Hybrid Models for Incomplete Graph Search. Hybrid models combine multiple methodologies to overcome the limitations of individual approaches. The Structure-Attribute Transformer (SAT) [11] was among the first to propose a hybrid approach. It integrates transformers with graph neural networks to handle missing data by jointly considering structural and attribute information. This model enhances data imputation techniques in graph analysis but may introduce noise, negatively impacting search performance. Later, D^2PT [32] was introduced as a dual-channel GNN designed to improve information propagation in incomplete graphs. Unlike conventional GNNs, D^2PT employs a diffused propagation then transformation (DPT) backbone that enhances long-range message passing while maintaining efficiency. Additionally, it constructs a global graph based on semantic similarities, enabling information propagation even for stray nodes that would otherwise remain disconnected. To align these two learning processes, D^2PT utilizes a contrastive prototype alignment mechanism, ensuring effective knowledge transfer between the original and global graphs. However, D^2PT is not specifically designed for search-based tasks on real-world datasets, where labels are not always available or well-structured, potentially limiting its adaptability to large-scale retrieval problems. Recently, KS-GNN [22] employs message-passing and feature aggregation for more accurate keyword searches in incomplete graphs. It integrates a three-term loss function: (1) Reconstruction loss (similar to encoder-decoder architectures), (2) Subgraph keyword-based node similarity, and (3) Keyword frequency regularization. Despite its effectiveness, message-passing mechanisms struggle with missing edges, making KS-GNN less stable for highly incomplete graphs.

Table 1. Comparison of methods for incomplete graph search.

Method	**Handles Missing Data**	**Scalability**	**Supports Keyword Search**	**Learns Node Representations**	**Structural & Attribute Integration**	**Unsupervised Learning**	**Robust to High Incompleteness**
BANKS [5]	✗	✓	✓	✗	✗	✓	✗
BLINK [23]	✓	✗	✓	✗	✗	✓	✗
PCA [19]	✗	✓	✗	✗	✗	✓	✗
KPCA [63]	✗	✓	✗	✗	✗	✓	✗
GCN [29]	✗	✓	✗	✓	✗	✓	✗
GAT [52]	✗	✓	✗	✓	✓	✓	✗
GraphSAGE [21]	✗	✓	✗	✓	✓	✓	✗
SAT [11]	✗	✗	✓	✓	✓	✓	✗
D^2PT [32]	✓	✓	✗	✓	✓	✗	✗
T2-GNN [25]	✓	✗	✗	✓	✓	✗	✗
KS-GNN [22]	✓	✓	✓	✓	✗	✓	✗
Proposed Method	✓	✓	✓	✓	✓	✓	✓

Comparison of Methods. Table 1 summarizes the capabilities of different methods for incomplete graph search. The table highlights key characteristics such as scalability, structural and attribute integration, and robustness to highly incomplete graphs. Notably, while some traditional and probabilistic methods provide efficient search capabilities, they lack resilience in sparse graphs. Machine learning-based approaches, particularly hybrid models, demonstrate improved robustness but often require high computational resources.

3 Problem Definition

The task of keyword search in graphs is focused on finding a set of nodes from within the larger graph such that they collectively cover the set of keywords provided by the user in the form of a query. For instance, when searching an academic graph for a query such as *graph neural networks*

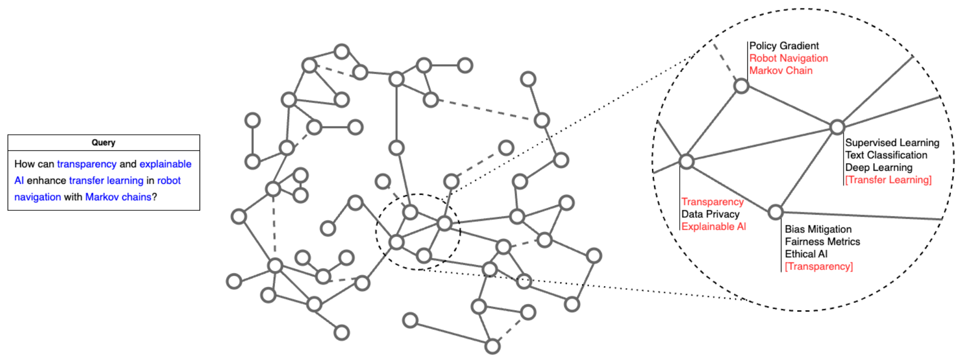


Fig. 2. A case study of keyword search in an incomplete graph. The selected nodes collectively cover the query keywords, but missing edges and attributes introduce challenges that affect structural connectivity and semantic completeness.

for protein structures, the objective will be to find and retrieve a set of nodes (e.g., papers) that contain the query keywords. In this paper, we are aiming to address special case of this task where the underline graph is incomplete. In other words, some of the graph attributes or edges may be missing.

To illustrate the challenges of keyword search in incomplete graphs, we present a case study. Consider a user query: *"How can transparency and explainable AI enhance transfer learning in robot navigation with Markov chains?"*. As shown in Figure 2, the incomplete graph may suffer from missing edges and attributes. The objective is to select a group of nodes that ideally cover all the required keywords; however, missing information introduces additional challenges. Some keywords may be absent from the dataset and need to be inferred based on semantic relationships. Similarly, missing edges can disrupt the structural connectivity of the graph, potentially weakening important relationships between nodes. This case study highlights the need for robust methods to handle missing information while ensuring effective keyword search in incomplete graphs.

Let us define this task as follows. Given a set of keywords as the input query, we let the query be:

$$q = \left(w_1, w_2, \ldots, w_{n_q}\right) \tag{1}$$

where $(w_1, w_2, \ldots, w_{n_q})$ denotes keywords in query q with n_q being size of query keywords. Subsequently, the set of keywords for the i^{th} query is denoted as W_{q_i} and defined as follows:

$$W_{q_i} = \left(w_1^{q_i}, w_2^{q_i}, \ldots, w_{n_q}^{q_i}\right) \tag{2}$$

Furthermore, we denote the main graph being searched as $G(V, E, W)$, where V is the set of nodes, E is the set of edges connecting the nodes, and W is the set of keywords. For each $v \in V$, there is the set of associated keywords denoted as W_v and defined as follows:

$$W_v = \left(w_1^v, w_2^v, \ldots, w_{n_v}^v\right) \tag{3}$$

where n_v is the size of node keywords. Based on the mentioned definitions for the query as the input and graph as the data, we can now define the task of keyword search: the objective of keyword search on graph G is to find a subgraph $G'(V', E', W')$ where $V' \subseteq V$ and $E' \subseteq E$ and keywords associated with V' can collectively cover all the keywords in the query q_i, i.e., $W_{q_i} \subseteq W'$ [23].

The desired output of the keywords search task is to find the most efficient subgraph that not only covers the required keywords but also minimizes pre-defined costs. Thus, we can define the desired output by properly defining the scoring function that is needed to retrieve the subgraph. The widely adapted scoring function, based on [13, 20, 22, 23, 30, 50], is defined as:

$$score(v, q_i) = \sum_{j=1}^{n_q} \text{dist}_{\text{min}} \left(v, w_j^{q_i} \right) \tag{4}$$

where $dist_{\text{min}} \left(v, w_j^{q_i} \right)$ represents the path length between node v to the nearest node containing keyword $w_j^{q_i}$, which is the j^{th} keyword in q_i's keyword set.

In other words, the keyword search objective is to find the subgraph $G'(V', E', W')$ whose nodes have minimum cumulative distance from nodes that host the query keywords. The keyword search problem is defined as follows:

$$\forall v \notin V', score\,(v', q) \geq \max\left(\{score\,(v_i, q) \mid v_i \in V'\}\right) \tag{5}$$

Given our focus on incomplete graphs, we need to specifically take incomplete graph characteristics into account. Incompleteness in a graph can occur in the form of missing keywords and edges. Hence, a *desirable method* should satsify the following objectives:

Objective 1: Reduce the dimension of the output to make search feasible

While modern graph embedding techniques can effectively manage high-dimensional data, reducing output dimensionality remains a valuable strategy for enhancing search performance. High-dimensional spaces can still pose challenges such as increased computational cost and reduced algorithmic efficiency due to the “curse of dimensionality” [3, 43]. In our approach, dimensionality reduction serves to simplify the search space, improving efficiency and robustness, and reducing computational overhead. For example, in a scientific collaboration graph such as DBLP [49], nodes represent researchers, and edges represent co-authorships between these researchers. This graph is inherently high-dimensional due to the vast number of potential connections (edges) and attributes (e.g., keywords, publication venues, research topics) associated with each researcher. Current practical techniques such as KS-GNN/KS-PCA [22] and Graph Neural Networks-based methods [6, 14, 21, 29, 52] use dimensionality reduction as an instrument in this regard, enabling the extraction of meaningful low-dimensional features from high-dimensional data, thereby preserving the essential characteristics of the graph while making the search process more manageable.

Objective 2: Retain as much key information as possible despite missing keywords and edge information

The loss of keywords and edge information can severely impact the ability to perform accurate and comprehensive keyword search on graphs. For instance, consider an incomplete graph representing a product recommendation system, where nodes are products, and edges indicate that two products are often bought together e.g. an online retail dataset [9]. Due to incomplete data, some edges (product relationships) and keywords (product descriptions or tags) might be missing. To retain key information despite these missing elements, one could employ machine learning models to predict missing edges based on existing connections and shared attributes among products. For example, if product A is often bought with products B and C, and product B is often bought with product D, but the direct link between A and D is missing, the system could infer this relationship based on the shared key information. This approach helps preserve the integrity of the recommendation graph, ensuring that users receive relevant suggestions even in the face of incomplete data.

Consequently, it is crucial to devise strategies that minimize the effect of these losses by retaining as much relevant information as possible. This objective focuses on leveraging the remaining data to infer missing information or to highlight connections and patterns that are still discernible despite the incompleteness. Methods that enhance the robustness of search algorithms against missing data, such as incorporating encoding-decoding mechanisms, e.g., denoising autoencoders [4], utilizing probabilistic models to predict missing connections [28] are good examples of methods that address this objective.

Objective 3: Capture the structural properties of the graph to overcome missing edges

The structural integrity of a graph plays a pivotal role in determining the relevance and context of search results. Missing edges can disrupt the connectivity and overall structure of the graph, leading to incomplete or inaccurate search outcomes. Therefore, preserving the structural properties of the graph is paramount. This involves identifying and utilizing alternative paths or relationships within the graph that can compensate for the missing edges. For instance, in a social network such as Facebook and Twitter [33], if two individuals share many mutual connections but are not directly connected in the graph due to missing edges, a graph neural network method (e.g. GCN [29], GAT [52]) could predict the likelihood of a direct connection based on the observed structural patterns. This approach helps maintain the social graph integrity, enabling accurate analysis of community structures, influence spread, and social cohesion despite incomplete information. Additionally, leveraging graph neural networks or other advanced techniques to infer missing structural information can help maintain the integrity of the graph's topology, ensuring that the search algorithm can still navigate the graph effectively and yield relevant results despite the missing connections.

4 Methodology

In this section, we present our approach for address the three objectives (**Objectives 1-3**). In order to address objectives O_1, O_2 and O_3, we developed our proposed approach around three principles (P_1, P_2 and P_3), which directly map onto the problem objectives, thus P_1 addresses O_1, P_2 addresses O_2 and P_3 addresses O_3 as follows:

Principle 1: Complexity of subgraph identification

The problem of keyword search in graphs has been shown to be NP-hard [26]. Hence, search in incomplete graphs, as super-sets of complete graphs, is an NP-hard problem. Therefore, we utilize the graph representation learning technique to estimate the relevance of graph elements. This complies with objective O_1, which discusses the "curse of dimensionality". To address P1, we adopt a graph representation learning strategy to estimate the relevance of keywords to subgraphs through soft-matching, which is inexact yet tractable. This addresses objective O_1, by reducing the output dimension and introducing a more efficient search space.

Principle 2: Addressing missing information

Most, if not all, real-world graphs suffer from missing information. As such, our proposed method should be able to preserve key information from the graph because of its importance for the sake of searching to find correct target nodes. In addition, our proposed method should be able to capture a range of different complementary information from the graph. Because of missing edges, the proposed method must be able to capture the graph structure and maximally use that to share information with the neighbourhood nodes for better connectivity and richer embedding vectors. Also, due to missing keywords, the proposed method should be able to capture information about keywords based on other available information besides the limited set of observable keywords that are available in the graph. This can be achieved by considering other pertinent node attributes, e.g., importance and co-occurrence probability. This principle addresses objective O_2, trying to retain as much key information as possible.

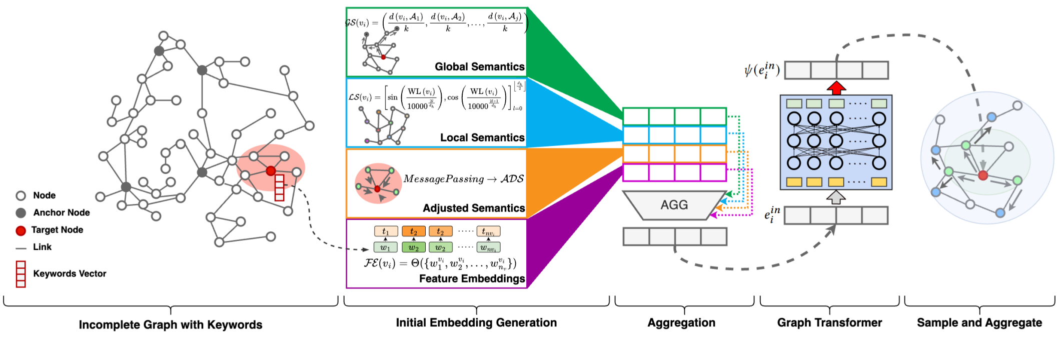


Fig. 3. The overall architecture of our proposed approach. The architecure comprises a transformer-based neural model that integrates global, local, and adjusted semantics for robust representation learning. Using anchor nodes, Weisfeiler-Lehman algorithm and k-hop message passing techniques to capture global, local, and adjusted semantics, respectively. The semantics is used as positional encoding along with feature embeddings to train a transformer architecture, enabling an efficient keyword-based subgraph search over incomplete graphs.

Principle 3: Navigating missing edges

A common challenge in incomplete graphs is the absence of certain edges, which may disrupt the ability to accurately represent the true connections within the graph. This absence not only obscures the graph's underlying structure but also hampers the effective retrieval of relevant subgraphs based on keyword search. To confront this issue head-on, our approach should involve leveraging predictive modelling to infer missing edges based on existing patterns of connectivity and node attributes within the graph. By predicting these missing edges, we aim to reconstruct a more complete picture of the graph topology, thereby enhancing the integrity and utility of the keyword search process. This proactive strategy specifically addresses objective O_3, ensuring that the proposed method remains robust and effective even in the face of incomplete connectivity data.

To satisfy these three principles, we propose a neural architecture for learning representations for an incomplete graph that captures the global, local, and adjusted semantics of the graph. To address these challenges and meet principles, our proposed method utilizes unsupervised learning through a transformer-based neural architecture for generating rich embedding vectors that are suitable for searching over incomplete graphs. The architecture of our proposed approach is shown in Figure 3. We have chosen a transformer architecture for the proposed method for multiple reasons. First, the keyword search task requires identifying nodes that collectively cover a given set of keywords. In a Transformer-based model, we can represent node features as a sequence of tokens, similar to how words are processed in natural language tasks. This tokenized representation allows the model to learn contextual relationships between node attributes, improving the accuracy of keyword matching. Second, unlike GNNs, which rely on predefined structural positions, Transformers can use semantic embeddings as positional information. This allows the model to understand the relationships between nodes based on their feature similarities rather than their adjacency in the graph. As a result, even when graph structure is incomplete, the model can still organize and retrieve relevant nodes effectively. Additionally, Transformers offer scalability advantages. Self-attention mechanisms allow parallel processing of nodes, making the approach computationally efficient for large-scale graphs. Unlike deep GNNs, which suffer from over-smoothing when stacking multiple layers, Transformers maintain distinct feature representations even in deep architectures, ensuring better discrimination between nodes.

In what follows, we will explain the representation learning process in subsection 4.1. We will explain how each of the architecture blocks relates to our defined principles. Then we will discuss the search mechanism based on the generated embedding vectors in subsection 4.2.

4.1 Indexing

Using representation vectors in the latent space helps acquire higher performance by several means. First, the common raw graph representations are sparse naturally, thus, it is hard to train graph neural models on them. Furthermore, using a transformer-based neural network generates contextualized embedding of nodes. This is especially important when searching over graphs with missing values because contextualization generates more specific and informative embedding vectors. Moreover, low-dimension embedded vectors increase the computation speeds during both the training and inference phases. Lastly, while it is possible to merge different pieces of information into one vector by concatenation, training an encoder on a mixture of that information yields a drastically better representation that can improve the downstream searching task performance. This is in response to our first principle (P_1). In response to the second and third principles (P_2 and P_3) we propose four representations to be captured in a graph. These embedding vectors create a multi-aspect process of a graph to retain as much key information as possible (P_2) and understand graph topology even with edges missing (P_3). In this subsection, we precisely define these representations, namely Global Semantics ($\mathcal{GS}$), Adjusted Semantics ($\mathcal{AS}$), Local Semantics ($\mathcal{LS}$) and Feature Embedding ($\mathcal{FE}$).

Global Semantics. This semantic is designed to be utilized as a representation of the target node compared to the entire graph. The goal is to reach an embedding vector that can be used to distinguish two nodes with similar neighbourhood characteristics but in different parts of a graph. To achieve this, we adopt the concept of anchor nodes for determining the global semantics of each node in the graph [65]. It has been shown that positioning nodes within the context of a particular set of anchor nodes can help capture and embody the broader context of the graph structure within node representations [34]. That is because even if two nodes reside in different parts of the graph with a similar neighbourhood, their semantic embeddings still will be different because of their position with respect to anchor nodes.

Anchor nodes become especially useful in *incomplete graphs* since missing edges can increase the likelihood of graph node representations losing their discriminative power. According to the Bourgain Theorem [8], selecting the proper set of anchor nodes can drastically affect the quality of representations learnt for graph nodes. Linial et al. [31] have demonstrated a practical approach for generating representations using a random set of anchor nodes. Based on the concept of anchor nodes, we define the Global Semantics ($\mathcal{GS}$) of a node as follows:

$$\mathcal{GS}(v_i) = \left(\frac{d\left(v_i, S_1\right)}{k}, \frac{d\left(v_i, S_2\right)}{k}, \ldots, \frac{d\left(v_i, S_j\right)}{k}\right) \tag{6}$$

where $S_j \subset V$ represents a set of anchor nodes selected based on the approach proposed by Linial et al. [31], and k is the number of anchor sets. The $d(v_i, S_j)$ is a distance metric for which we adopt the shortest-length path between node v_i and anchor node S_j. The distance can be defined as follows:

$$d\left(v_i, S_j\right) = \min_{u \in S_j} d(v, u) \tag{7}$$

Local Semantics. In order to learn the local semantics of each node within its neighbourhood, we utilize the Weisfeiler-Lehman (WL) algorithm [39, 47] which is widely used for distinguishing a pair of non-isomorphic graphs G, G'. The Weisfeiler-Lehman (WL) algorithm maintains a state for each node, which it progressively refines by aggregating the states of the node's neighbours. Often these states are represented by colours for better intuition. This iterative process results in an embedding for the graph, where each node's final state is represented in the form of an embedding that captures the structural context of the node within the entire graph [24].

Therefore, the WL algorithm can represent a node's position in its neighbourhood within a graph. This positioning system offers a local view of the graph for each given node. We incorporate the WL algorithm into a kernel function as suggested in prior research studies [60–62] and define an embedding function to produce the local semantics. More specifically, we define the Local Semantics ($\mathcal{LS}$) of a node as follows:

$$\mathcal{LS}(v_i) = \left[\sin\left(\frac{\mathrm{WL}(v_i)}{10000^{\frac{2l}{d_h}}}\right), \cos\left(\frac{\mathrm{WL}(v_i)}{10000^{\frac{2l+1}{d_h}}}\right) \right]_{l=0}^{\left\lfloor \frac{d_h}{2} \right\rfloor} \tag{8}$$

where $v_i \in V$, d_h is the embedding dimension and l is the context processed for generating embeddings. In Equation 8, $\mathcal{WL}$ denotes the Weisfeiler-Lehman algorithm. The detailed steps of the $\mathcal{WL}$ algorithm can be seen in Algorithm 1.

Algorithm 1: Weisfeiler-Lehman ($\mathcal{WL}$) algorithm

```
  Input: $G = (V, E, W)$
1 $c_v^0 \leftarrow \mathrm{hash}(W_v)$ for all $v \in V$
2 repeat
3     $c_v^\ell \leftarrow \mathrm{hash}\left(c_v^{\ell-1}, \left\{c_u^{\ell-1} : u \in \mathcal{N}_G(v)\right\}\right) \forall v \in V$
4 until $\left(c_v^\ell\right)_{v\in V} = \left(c_v^{\ell-1}\right)_{v\in V}$
5 return $\left\{c_v^\ell : v \in V\right\}$
```

In Algorithm 1, we consider graph $G = (V, E, W)$, where V is the set of vertices; $E \subseteq V \times V$ is the set of edges between vertices; and W_V is the set of node keywords: For all $v \in V, W_v \in \mathbb{R}^d$. The neighbors of a vertex v is $\mathcal{N}_G(v) = \{u : (v, u) \in E\}$. Each level of this iterative process is shown by ℓ. Thus, we can formulate the colour of nodes in level ℓ by c_v^ℓ.

Lines 2-4 in Algorithm 1, represent an iterative process where the colours are updated so that the colour of node $\sqsubseteq \in V$, becomes different from its neighbourhood $\mathcal{N}_G(v)$. This is done by using an injective hash function *hash* [24]:

$$\forall (v, u) \in V : \mathrm{hash}(W_v) = \mathrm{hash}(W_u) \iff W_v = W_u \tag{9}$$

Capturing local semantics is especially useful when facing missing edges. The local semantics and global semantics can play a complementary role for each other and cover each other's shortcomings in different situations. There can be a graph where several nodes have highly similar global semantics ($\mathcal{GS}$). This is because, due to missing edges, the chance of having nodes with identical distances to anchor nodes increases significantly. Therefore, the local semantics of nodes can help distinguish nodes from each other.

Adjusted Semantics. The objective of the adjusted semantics is to learn a balanced representation of a node that considers both immediate node neighbours as well as globally reachable nodes. This semantic aspect acts as a complimentary feature to fill the gap between global and neighbourhood positioning.

For this purpose, we utilize a k-hop strategy to traverse the graph starting from the node of interest, which will result in a sequence of k-visited nodes. This sequence can serve as a context for the node of interest and hence can be used to learn representations for the node based on this concept of Adjusted Semantics ($\mathcal{AS}$). Therefore, the formulation of the adjusted semantic representation contains a series of mentioned extracted contexts. To achieve this, we can use a similar equation to global semantics Equation 6. The difference is that instead of anchor nodes, k-hops will be used. Thus the $S_j \subset V$ represents the sequence of nodes traversed by the k-hop.

Feature Embedding. In the context of the keyword graph, each node is often associated with a set of keywords. However, the number of keywords associated with each node is quite insignificant compared to the total number of keywords in the entire graph, making keyword distribution quite sparse. In addition, with *missing keywords*, the keyword distribution becomes even more sparse.

In order to produce feature representations, we utilize a tokenizer to transform the sparse occurrence vector of keywords to a dense low-dimension feature embedding representation. This captures the interaction between keywords and the co-occurrence of them.

Let $W_{v_i} = \left(w_1^{v_i}, w_2^{v_i}, \ldots, w_{n_v}^{v_i}\right)$ represent the input sequence of words for node v_i, where $(w_i^{v_i})$ denotes the i_{th} word in the sequence, and n_{v_i} is the length of the sequence for node v_i.

Let T represent the vocabulary of tokens, where each token $t_j \in T$ is associated with a unique index j such that $j \in \{1, 2, \ldots, |T|\}$, and $|T|$ is the size of the vocabulary.

The tokenization process can be described by Θ function that maps the input word sequence W_{v_i} to a sequence of tokens $\{t_1, t_2, \ldots, t_{n_{v_i}}\}$:

$$\mathcal{FE}(v_i) = \Theta(\{w_1^{v_i}, w_2^{v_i}, \ldots, w_{n_v}^{v_i}\}) = \{t_1, t_2, \ldots, t_{n_{v_i}}\} \tag{10}$$

where each token t is an element of the vocabulary T.

4.1.1 *Input Encoding.*

The four views on node semantics from the prior sections provide rich information about graph nodes. We integrate the four different embedding vectors through an aggregation function, referred to as AGGREGATE(.). We formulate the final representative embedding vector for $v_i \in V$ as follows:

$$e_i^{in} = \text{AGGREGATE}\ (\mathcal{GS}(v_i), \mathcal{LS}(v_i), \mathcal{AS}(v_i), \mathcal{FE}(v_i)) \tag{11}$$

The AGGREGATE(.) function plays a central role in fusing the diverse dimensions of node semantics into a unified embedding vector. This aggregation enables a holistic representation of each node's characteristics, thereby facilitating stronger and deeper analyses and interpretations. Xu et al. [56] argue that $AGGREGATE(.)$ can be implemented through various strategies such as element-wise max-pooling or vector summation.

The keyword search task relies heavily on matching query keywords with node attributes. Thus, it is important to capture and retain as much keyword information as possible. By aggregating embeddings to reflect the collective attribute strength, the summation strategy as the aggregation method creates a broad feature profile for each node. This broad profile is advantageous in keyword search, as it increases the likelihood of matching a wide range of query keywords, thereby compensating for the information loss in incomplete graphs. This complies with our second principle (P_2).

In addition, an incomplete graph may lack certain edges, leading to challenges in capturing the full relational context of nodes. It is important to save information echoed by existing edges. The summation strategy indirectly compensates for this by emphasizing the available nodal attributes over relational data, thus maintaining a high level of search effectiveness even when the graph

Algorithm 2: Unsupervised Fine-tuning of Transformer Model using Triplet Sampling

```
   Input: $G = (V, E, W)$, $\mathcal{GS}()$, $\mathcal{LS}()$, $\mathcal{AS}()$, $\mathcal{FE}()$
   Output: Fine-tuned model $\mathcal{T}$
1  Initialize transformer model $\mathcal{T}$
2  foreach $v_i \in V$ do
3      $e_i^{in}$ = AGGREGATE $(\mathcal{GS}(v_i), \mathcal{LS}(v_i), \mathcal{AS}(v_i), \mathcal{FE}(v_i))$ // Compute embedding $e_i^{in}$
4      $\psi(e_i^{in}) = \mathcal{T}(e_i^{in})$ // Compute initial representation
5  end
6  repeat
7      foreach $v_i \in V$ do
           // step 1 in sampling mechanism
8          Select as positive sample $e_{i_+}^{in}$ from the 1-hop neighbourhood
9          Select as negative sample $e_{i_-}^{in}$ if not in the 1-hop neighbourhood
           // step 2 in sampling mechanism
10         if $Hamming(e_i^{in}, e_{i_+}^{in}) > Hamming(e_i^{in}, e_{i_-}^{in})$ then
11             $\mathcal{L} = relu\left(\left(\left[\left\|\psi\left(e_i^{in}\right) - \psi\left(e_{i_+}^{in}\right)\right\|_2^2 - \left\|\psi\left(e_i^{in}\right) - \psi\left(e_{i_-}^{in}\right)\right\|_2^2\right]\right)\right)$ // Calculate
                 triplet loss for $i_{th}$ node
12             Update model $\mathcal{T}$ to minimize $\mathcal{L}$
13         end
14     end
15 until convergence
16 foreach $v_i \in V$ do
       // sample-and-aggregate neighbor representations
17     $\text{AGGREGATE}_k^{\text{pool}} = \max\left(\left\{\sigma\left(W_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\right\}\right)$
18 end
```

structure is partially unknown. This addressed our third principle (P_3). Methods that eliminate part of the information, such as pooling-based techniques, are not well-suited for our task.

Moreover, dimensionality is another factor to consider for aggregation. Hence, concatenation is not a practical strategy since four embedding vectors need to be concatenated. This complies with our first principle (P_1).

Altogether, considering our predefined principles and detailed analysis of available aggregation strategies above, we have adapted the vector summation method for the AGGREGATE(.) function. Its capacity to uniformly integrate available information, maximize the utility of partial data, and maintain simplicity and efficiency in analysis makes it particularly apt for navigating the challenges posed by incomplete graphs. This strategy ensures that despite the inherent data limitations, the process of keyword search remains robust, comprehensive, and effective.

4.1.2 *Training the Model*.

In this section, we discuss the training procedure. Our proposed method uses a transformer-based model to generate node representation. Let's represent the generated representation for the i^{th} node with $\psi(e_i^{in})$ and the transformer model with $\mathcal{T}$. Therefore:

$$\psi(e_i^{in}) = \mathcal{T}(e_i^{in}) \tag{12}$$

Given the nodes in the keyword graph are primarily associated with keywords, we adopt an unsupervised learning technique to fine-tune the transformer model given the generated semantics and feature embedding vectors. Using the triplet sampling technique [45, 54], each training record has a negative and positive sample. Thus, the loss function tries to maximize the similarity between the target node and the positive sample and minimize the negative sample. The positive and negative samples need to be specifically defined for the keyword search problem. The ideal loss function must take into effect both the semantics and keywords of nodes when comparing the similarity. We have adopted a customized sampling mechanism for the triplet function to consider both semantics and keyword similarity:

(1) To consider the semantics, we explore the 1-hop neighbourhood of the target node to find a positive sample. This is because nodes with the most similar semantics to the target node are its immediate neighbourhood. Thus, a positive sample must be selected from the target node's 1-hop neighbourhood. Likewise, the negative sample will be any node that is not within the 1-hop neighbourhood of the target node and consequently has no edge connecting to the target node. Positive samples are randomly selected from the target node's 1-hop neighborhood. Similarly, negative samples are randomly selected from non-neighbors within 1-hop from the target node.
(2) To consider the similarity of the keywords, we represent the keywords of each node with one-hot encoding and calculate the Hamming similarity between the target node and its positive and negative samples. Then, we only accept selected positive and negative samples in step 1 if the Hamming similarity between the target node and positive sample is greater than the target and negative sample.

Therefore, the loss function for fine-tuning the model can be formulated as:

$$\mathcal{L} = \frac{1}{N}\sum_{i}^{N} Max\left(\left[\left\|\psi\left(e_i^{in}\right) - \psi\left(e_{i_+}^{in}\right)\right\|_2^2 - \left\|\psi\left(e_i^{in}\right) - \psi\left(e_{i_-}^{in}\right)\right\|_2^2\right], 0\right) \tag{13}$$

where $e_{i_+}^{in}$ and $e_{i_-}^{in}$ represent positive and negative samples, respectively. Here, N is the total number of samples. The loss function in Equation 13 maximizes the similarity between the target node and positive sample and simultaneously minimizes the similarity between the target node and negative sample.

After generating representation vectors for each of the nodes in the keyword graph, we utilize a sample-and-aggregate strategy [21]:

$$\text{AGGREGATE}_k^{\text{pool}} = \max\left(\left\{\sigma\left(W_{\text{pool}}\,\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\right\}\right) \tag{14}$$

where σ is a non-linear activation function, W_{pool} denotes the weight matrices, h_{wi}^k is the aggregated neighborhood vectors related to the k^{th} layer of the model and b is the bias term. The set of neighbour nodes for node v is denoted by $\mathcal{N}(v)$. The benefit of the sample-and-aggregate strategy is that each node's final representation vector is mixed with a set of samples of its neighbours' representations. This will smooth the search space and propagate the information even further, which is a crucial factor in graphs with missing values. The final node representation vectors are used for searching over the keyword graph.

The procedure for fine-tuning the transformer model is detailed in Algorithm 2. This algorithm outlines the steps involved in initializing the model and computing input embeddings (lines 1-5), sampling positive and negative nodes (lines 8-10), computing loss value and updating the model parameters based on it (lines 11-12). By iterating through all samples (line 7), the model learns to distinguish between similar and dissimilar nodes, effectively improving its performance on the search task. We have used the $relu()$ function (line 11) to implement the $max(x, 0)$ in the loss function. Lastly, a sample and aggregate technique will be applied to all nodes in the graphs (lines 16-18).

4.2 Search

During the search, the input query is given as a set of keywords. We denote i_{th} query with q_i as defined in Equation 1. The pseudo-algorithm for search operation can be seen in Algorithm 3. Since the query can be treated as a new node (line 1), its input embedding $e^{in}_{v_{q_i}}$ can be computed using only its features, $\mathcal{FE}(v_{q_i})$ (line 2). This is because the query node will not have any semantic embedding as it is not part of the graph. Next, the final representation for the input query node is generated using the transformed model $\mathcal{T}$ (line 3). Lastly, we compute the similarity between all the node representations and query nodes (lines 4-6) and return the $top - k$ most similar nodes (line 7).

Algorithm 3: Keyword Search

```
  Input: $G = (V, E, W)$, $q_i$, $k$
  Output: Top-$k$ similar nodes to the input query $q_i$
  // Search using query keywords
1 Represent query $q_i$ as new node $v_{q_i}$
  // Use feature embedding to generate input embedding for $v_q$
2 $e^{in}_{v_{q_i}} \leftarrow \mathcal{FE}(v_{q_i})$
  // Generate representation using trained transformer-based model
3 $\psi(e^{in}_{v_{q_i}}) \leftarrow \mathcal{T}(e^{in}_{v_{q_i}})$
4 for each node $v_i \in V$ do
5     Compute similarity $S(\psi(e^{in}_{v_{q_i}}), \psi(e^{in}_{v_i}))$ between query node $v_{q_i}$ and node $v_i$
6 end
7 Return top-$k$ nodes $\{v_1, v_2, \ldots, v_k\}$ with the highest similarity
```

4.3 Time Complexity Analysis

The problem of keyword search in graphs has been shown to be **NP-hard** [26, 27, 58]. Since searching in incomplete graphs generalizes the problem of complete graph search, it remains NP-hard. To address this challenge, the proposed approach leverages *graph representation learning* to estimate the relevance of graph elements, thereby reducing the search space dimensionality.

Starting with semantics, Global Semantics require computing the shortest path length to anchor nodes. Using Breadth-First Search (BFS) to determine the shortest path between two nodes results in a time complexity of $O(V + E)$. In the worst case, where the graph is complete ($E = V^2$), this simplifies to $O(V^2)$ [12].

Adjusted Semantics utilize k-hop traversal. Since the number of hops is small, in a dense network, the time complexity is dominated by $O(d^k)$, where d is the average node degree and k is the number of hops [12]. Lastly, Local Semantics employ the Weisfeiler-Lehman (WL) graph coloring algorithm with a time complexity of $O(hE)$, when using hashing-based algorithm. Where h is the number of iterations and E is the number of edges [47]. Feature Embedding component employs a tokenizer with a time complexity of $O(n_q)$. Where n_q is size of tokens.

Our proposed method incorporates a transformer-based model for graph representation learning. The self-attention mechanism in transformers has a time complexity of:

$$O({n_q}^2) \tag{15}$$

where n_q represents the number of tokens [51].

Finally, a sample-and-aggregate mechanism is applied to generate final embeddings. For a single node v with its k-hop neighborhood, let:

- k be the number of layers,
- S be the sample size (number of neighbors sampled per node),
- F be the feature dimension.

The time complexity of this process is:

$$O(S^k \times F) \tag{16}$$

The overall time complexity of the proposed method is:

$$O(\text{proposed method}) = O(V^2) + O(d^k) + O(hE) + O(n_q) + O(n_q^2) + O(S^k \times F) \tag{17}$$

4.3.1 Search Mechanism Complexity. The search mechanism involves *embedding-based keyword matching*, reducing the brute-force complexity from:

$$O(V \times n_q) \tag{18}$$

The efficiency of the search is influenced by the number of query keywords (n_q) and the number of graph nodes (V) that must be compared.

Complexity Dominance Considerations: The dominant complexity term depends on various factors:

- In **dense graphs** ($E = V^2$), the term $O(V^2)$ dominates.
- When using **high-hop traversal** (k is large), the exponential term $O(d^k)$ can become a bottleneck.
- If **sampling neighborhoods** (S^k) is extensive, the complexity $O(S^k \times F)$ may grow significantly.
- The **transformer model** scales quadratically with the number of tokens, meaning $O(n_q^2)$ is crucial for large queries.

In typical scenarios where the number of hops (k) and query tokens (n_q) remain relatively small, the term $O(V^2)$ is the dominant factor in the time complexity from Equation 17. Therefore, it provides a reasonable approximation for the overall time complexity in common use cases.

5 Experiments

5.1 Dataset

In our study of keyword search within incomplete graphs, we selected a set of diverse datasets known for their broad application in graph neural network research. This selection mirrors the state-of-the-art approach utilized by KS-GNN [22], ensuring consistency in evaluation criteria and facilitating direct comparison of results. The datasets include:

- **CiteSeer:** A digital library focusing on literature in computer and information science, known for its citation graph, which links documents through citations [44]. In this dataset, nodes represent published papers, and the edges between the nodes indicate citation relationships. Each paper within the graph is associated with a set of keywords.

- **DBLP:** This is a comprehensive bibliographic database covering major journals and proceedings in the field of computer science, presenting a rich source of academic relationships [49]. DBLP is analyzed as a co-authorship graph where each node represents an author, and edges signify co-authorship. Each author within the graph is associated with a set of keywords. Keywords associated with each node are extracted from the abstracts of the authors' published work.
- **Amazon Toys:** A subset of the Amazon co-purchase graph, where nodes represent products (in this case, toys), and edges indicate frequently co-purchased items, highlighting consumer behavior patterns [38].
- **Amazon Video:** Similar to the Amazon Toys dataset but focuses on movies, providing insights into co-purchasing trends among movie products on Amazon [38].

Table 2. Stats of experimental datasets.

Datasets	Nodes	Edges	Keys	Avg Deg	Closeness	Cluster	Cosine	Neigh Sim	Non-Neigh Sim	N/N Ratio	Avg Keys/Node
CiteSeer [44]	3,327	9,104	3,703	5.55	0.05	0.14	0.19	0.21	0.04	4.80	31.61
DBLP [49]	32,361	69,448	4,094	4.29	0.02	0.21	0.31	0.35	0.03	12.10	11.48
Amazon Video [38]	20,882	66,003	11,514	6.23	0.05	0.06	0.004	0.005	0.001	3.31	22.00
Amazon Toy [38]	20,682	224,603	4,114	20.78	0.10	0.40	0.10	0.11	0.002	57.16	3.01

These datasets were chosen due to their relevance in demonstrating the efficacy of chosen baselines in processing incomplete information. The statistical characteristics of datasets are shown in Table 2. The table reports the number of nodes and edges in each dataset, reflecting graph size and connectivity. The keys column denotes the number of unique keywords associated with nodes, which serve as search attributes. Average degree (Avg Deg) represents the mean number of edges per node, indicating graph sparsity or density. Closeness centrality (Closeness) captures how easily a node can reach others, while clustering coefficient (Cluster) measures the tendency of nodes to form tightly connected groups.

To quantify feature similarity, we report mean cosine similarity (Cosine) between node feature vectors, along with neighbor similarity (Neigh Sim) and non-neighbor similarity (Non-Neigh Sim), which indicate the average similarity among connected and unconnected nodes, respectively. The neighbor/non-neighbor similarity ratio (N/N Ratio) further highlights the extent to which connected nodes share higher feature similarity than unconnected ones. Lastly, the average keys per node (Avg Keys/Node) column represents the mean number of keywords assigned to each node, providing insight into attribute density within the dataset.

As seen in Table 2, we have intentionally selected datasets that have differing graph characteristics. For instance, the CiteSeer dataset, with 3,327 nodes and 9,104 edges, focuses on a more compact graph structure yet contains a substantial number of keywords (3,703), represents a relatively compact graph while maintaining a substantial number of keywords (3,703). This results in a high average keys per node value of 31.61, indicating that each node carries a significant amount of textual information, making it well-suited for evaluating search algorithms on feature-rich networks. Additionally, CiteSeer exhibits a moderate neighbor/non-neighbor similarity ratio (N/N Ratio) of 4.80, suggesting that connected nodes tend to have moderately higher semantic similarity than unconnected ones. In contrast, while the DBLP dataset also focuses on academic publications, it features the highest node count (32,361), along with 69,448 edges and 4,094 keywords, forming a large yet relatively sparse network. The average degree (4.29) is lower compared to the other datasets, reflecting a less connected structure. However, the neighbor similarity (0.35) remains significantly higher than the non-neighbor similarity (0.03), leading to a strong N/N Ratio of 12.10. Despite having a comparable number of keywords to CiteSeer, DBLP's average keys per node

(11.48) is considerably lower, implying that individual nodes contain fewer descriptive attributes. This makes DBLP a suitable benchmark for analyzing search effectiveness in large-scale academic graphs with sparse feature distributions.

We have also chosen two other datasets from a completely different domain, that is, electronic commerce. The Amazon Video and Amazon Toy datasets highlight different aspects of graph complexity. The Video dataset, with 20,882 nodes and 66,003 edges, has the highest number of keywords (11,514), leading to a moderately high average keys per node of 22.00. However, it has the lowest cosine similarity (0.004) and neighbor similarity (0.005), indicating that feature-based relationships are weaker in this dataset, likely due to the diverse nature of video content. Meanwhile, the Toy dataset, despite having a similar number of nodes (20,682), is significantly more interconnected, with 224,603 edges and a much higher average degree (20.78). The clustering coefficient (0.40) indicates strong local grouping tendencies, while its exceptionally high N/N Ratio (57.16) suggests that connected nodes are far more semantically aligned than unconnected ones. Interestingly, it has the lowest average keys per node (3.01), implying that while nodes in this dataset are densely interconnected, they contain fewer distinguishing textual features.

These variations across datasets, spanning differences in graph size, connectivity, feature similarity, and attribute density, provide a rich experimental ground for evaluating graph-based search and learning algorithms. The inclusion of datasets with *highly interconnected structures* (Amazon Toy), *sparse academic graphs* (DBLP), *feature-dense but structurally moderate networks* (CiteSeer), and *semantically diverse but weakly connected content* (Amazon Video) ensures a thorough assessment of our model's performance across different real-world scenarios.

5.2 Experimental Setup

In our study, we evaluate our suggested approach against standard techniques for keyword search tasks within two types of networks as suggested by [11, 22, 57]: (1) networks containing only absent keywords, and (2) networks missing both keywords and connections. For each dataset, to mimic real-world conditions and accurately adjust the levels of missing information, we modify the original data in two phases:

(1) Concealing the keywords for a specified portion (marked as r_w) of randomly chosen nodes in the network;
(2) Randomly hide a certain percentage (notated as r_e) of the network's edges.

We denote that the random selection functions independently, uniformly and without any tie to graph characteristics. Total number of words in a query q as $n_q = |q|$ and randomly select 100 queries for testing for each n_q value that ranges from 3 to 9, increasing by increments of 2 (i.e. [3,5,7,9]). Therefore, the size of keywords in the results tables indicates the size of the query.

The ground truth is generated by the BLINK [23] method. Similar to KS-GNN [22], the original graph is processed by BLINK and the top-K retrieved answers are used as the gold standard.

Furthermore, in each incomplete network, we create a validation set comprising 100 randomly chosen queries with known correct answers. We adjust the parameters of the methods under comparison using the grid search technique on the validation set. This approach allows for a comprehensive comparison with baseline methods under varying conditions of information incompleteness.

To ensure the reliability and stability of our findings, each method in our study is executed 20 times. This repetition allows us to mitigate any variability caused by random factors in the data or the algorithms. The final results we present are the averages of these runs, providing a more accurate reflection of each method's performance under the experimental conditions. This approach helps in understanding the expected behaviour of these methods in real-world scenarios. For each evaluation setup, the best results are indicated in bold

5.3 Code

The implementation of our proposed method and other baselines along with the dataset is publicly available on GitHub[1].

5.4 Baselines

In this study, we introduce a novel approach to keyword search on incomplete graphs, benchmarking against the contemporary state-of-the-art methodologies:

- **KS-GNN [22]:** A versatile method distinguished by its use of a three-term loss function and an encoder-decoder mechanism, enhanced with aggregation for embedding vector creation. This approach is adaptable, allowing for various graph representation learning techniques to be applied, thereby tailoring the search process to the specific characteristics of the graph data.
- **D^{2}PT** [32] is a dual-channel GNN that enhances learning in graphs with weak information by employing a diffused propagation then transformation (DPT) backbone. It utilizes a diffusion-based propagation mechanism to compensate for weak connectivity and missing information. Then, D^{2}PT constructs an auxiliary global graph based on the nearest-neighbor strategy to propagate information even for disconnected nodes. D^{2}PT is fundamentally a supervised approach, requiring labeled data for effective training. Therefore, we were only able to evaluate on Citesser and DBLP datasets.
- **GNN-Based Methods:**
  - **T2-GNN** [25] is a teacher-student distillation framework designed to improve robustness on incomplete graphs. It employs two separate teacher models: one for feature imputation and another for structure reconstruction, ensuring that missing information does not interfere with learning. A dual-distillation mechanism then transfers this knowledge to a student GNN. While T2-GNN achieves strong results on standard benchmarks, it relies on a fully supervised setting with labeled training data. Thus, we were only able to evaluate on Citesser and DBLP datasets.
  - **Graph Convolutional Networks (GCN) [29]:** Leverages the spectral properties of graphs to implement convolutional neural networks directly on graph-structured data. This method is pivotal for capturing local node features and their topological relationships, making it suitable for tasks that require an understanding of graph structure.
  - **Graph Attention Networks (GAT) [52]:** Introduces an attention mechanism into the graph neural network framework, allowing nodes to weigh the importance of their neighbours' features dynamically. This adaptability enables the model to focus on the most relevant parts of the graph for the task at hand.
  - **Chebyshev Convolution (ChebConv) [14]:** Utilizes Chebyshev polynomials to approximate the graph Laplacian's spectral filter, enabling the model to learn graph features over various scales. This method is efficient for large graphs where capturing global graph properties is essential.
  - **GraphSAGE [21]:** Employs a sampling-based approach to efficiently generate node embeddings by aggregating features from a node's local neighbourhood. This methodology is designed to handle large graphs by learning a function that generates embeddings by sampling and aggregating features from a node's neighbourhood.
- **BLINK+SAT [11, 23]:** A two-stage process where the SAT model initially predicts missing links/keywords, followed by the application of the BLINK method for keyword search. This

[1]GitHub code: https://github.com/radinhamidi/A-Robust-Neural-Approach-for-Searching-over-Incomplete-Graphs

sequential approach aims to enhance the accuracy of keyword searches on incomplete graphs by mitigating the impact of missing information.

- **PCA-Based Methods:**
  - **PCA (Principal Component Analysis) [19]:** PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. In the context of graph search, PCA can reduce the dimensionality of the graph's feature space, retaining those characteristics most significant for the search while minimizing information loss. This process simplifies the graph's complexity, facilitating a more efficient search operation.
  - **Conv-PCA [22]:** This method employs a convolutional graph neural network layer to aggregate and refine node representations from the graph's data, capturing both local and global structural patterns effectively. After processing through this convolutional layer, the refined node representations are then subjected to PCA for dimensionality reduction. This sequence ensures that the intricate patterns discerned by the convolutional layer are preserved, albeit in a reduced-dimensional space, enhancing the keyword search's efficiency and accuracy by focusing on the most impactful features.
  - **KS-PCA [22]:** This approach represents a hybrid approach that merges the strengths of KS-GNN's graph neural network framework with PCA's dimensionality reduction capabilities. In this method, the data is processed by PCA transformation instead of KS-GNN encoder-decoder architecture. This combination aims to harness PCA's efficiency in simplifying data representation and KS-GNN's robustness in handling graph-based data, especially in scenarios characterized by incomplete information.

This comparative list of baselines aims to underscore our proposed method's robustness and versatility in handling keyword searches on incomplete graphs. By evaluating against these diverse methodologies, our goal is to highlight the superior performance and efficiency of our approach, thereby making a significant contribution to the ongoing research in graph search algorithms.

5.5 Ablation Study

An ablation study was conducted to assess the individual and combined effects of the components of our proposed method on its overall performance. We evaluated our proposed method and all component combination variations under a challenging scenario, with half missing edges ($r_e = 50\%$) and half missing keywords ($r_w = 50\%$). This setup allowed us to rigorously investigate the role and impact of each component under conditions of significant data incompleteness. The results are shown in Table 3, from which several key insights emerge:

(1) **Single-component Efficacy:** Performance of the single components depends on the dataset characteristic. In a small dataset, i.e., CiteSeer, $\mathcal{AS}$ demonstrated slightly better performance relative to its counterparts. However, this changes when the dataset's size starts to grow. This growth can be in terms of the number of nodes (i.e. DBLP), number of total keywords (i.e. Amazon Video) or number of edges (i.e. Amazon Toy). It can be seen that $\mathcal{GS}$ and $\mathcal{LS}$ components perform slightly better than $\mathcal{AS}$. This can be justified by the fact that by increasing the size of the graph, $\mathcal{AS}$ semantic becomes less efficient in representing either the close neighbourhood or global position of a target node.

(2) **Double-component Synergy:** The combination of any two components invariably resulted in performance enhancement beyond that achievable by the components in isolation. Notably, not all the synergies contribute equally to the performance. CiteSeer and DBLP benefit greatly from combinations $\mathcal{GS} + \mathcal{LS}$ due to the need to capture both broad global

Table 3. Performance results of the ablation Study for CiteSeer, DBLP, Amazon Toy and Video datasets.

Datasets	r_e / r_w / n_q	50% Missing Edges, 50% Missing Keywords: 3	5	7	9
CiteSeer	$\mathcal{AS}$	25.48	30.74	33.24	34.17
	$\mathcal{GS}$	25.27	30.41	32.95	33.83
	$\mathcal{LS}$	25.35	30.05	33.79	34.01
	$\mathcal{GS}+\mathcal{AS}$	25.95	31.65	34.11	34.75
	$\mathcal{LS}+\mathcal{AS}$	26.21	32.71	34.17	35.32
	$\mathcal{GS}+\mathcal{LS}$	26.19	32.80	34.94	35.36
	Proposed Method (All)	**27.3**	**34.57**	**35.48**	**36.04**
Video	$\mathcal{AS}$	11.21	14.37	16.97	18.77
	$\mathcal{GS}$	11.43	14.89	17.08	19.58
	$\mathcal{LS}$	11.84	14.90	17.10	19.71
	$\mathcal{GS}+\mathcal{AS}$	12.31	15.35	18.56	20.19
	$\mathcal{LS}+\mathcal{AS}$	12.49	15.58	18.63	20.34
	$\mathcal{GS}+\mathcal{LS}$	12.09	15.12	18.07	20.11
	Proposed Method (All)	**13.71**	**16.63**	**19.09**	**21.45**
Toy	$\mathcal{AS}$	17.39	23.42	24.74	25.19
	$\mathcal{GS}$	17.60	23.90	24.83	25.67
	$\mathcal{LS}$	17.95	23.98	24.89	25.79
	$\mathcal{GS}+\mathcal{AS}$	18.07	24.07	25.41	26.14
	$\mathcal{LS}+\mathcal{AS}$	18.31	24.44	25.39	26.85
	$\mathcal{GS}+\mathcal{LS}$	18.60	24.82	25.58	26.92
	Proposed Method (All)	**19.9**	**25.54**	**27.03**	**28.34**
DBLP	$\mathcal{AS}$	16.06	23.20	27.59	28.33
	$\mathcal{GS}$	16.47	23.72	27.62	28.40
	$\mathcal{LS}$	16.19	23.93	27.48	28.67
	$\mathcal{GS}+\mathcal{AS}$	18.38	24.37	28.69	29.14
	$\mathcal{LS}+\mathcal{AS}$	17.84	24.02	28.13	29.01
	$\mathcal{GS}+\mathcal{LS}$	18.29	24.49	28.72	29.21
	Proposed Method (All)	**18.86**	**25.02**	**29.33**	**30.65**

relationships and detailed local structures, which is especially important in sparser graphs like DBLP where the combination of the relationships between nodes is often either very local or requires a broad, global perspective. Amazon Video and Amazon Toy tend to benefit more from $\mathcal{LS}+\mathcal{AS}$ combination, as these can better manage the dense local structures and large number of keywords. This underscores the importance of component interaction in optimizing performance.

(3) **Integrated Performance:** The holistic integration of all four components significantly outperforms any individual or combined subset thereof, across various configurations and scenarios. This demonstrates the synergistic effect of the components, each contributing uniquely to the framework's robustness and adaptability.

These observations not only highlight the strengths and limitations of the individual components but also affirm the efficacy of the integrated approach in enhancing search performance in incomplete graphs. Future work may explore further optimizations and the potential for additional components to address identified weaknesses, thereby extending the methodology's applicability and effectiveness.

5.6 Comparison With Baselines

Starting with the first scenario where only keywords are missed, we make several observations based on the result from Table 4:

Table 4. Performance for our proposed method and baselines by Hits@100 (%) with Missing Keywords and No Missing Edges. T2-GNN and D^{2}PT employ supervised approaches and are evaluated on labeled datasets, i.e., Citeseer and DBLP.

Datasets	r_w	30% Missing Keywords				50% Missing Keywords				70% Missing Keywords			
	n_q	3	5	7	9	3	5	7	9	3	5	7	9
CiteSeer	GCN	15.72	16.05	16.33	13.59	12.56	14.01	20.18	19.86	12.79	16.57	15.46	15.12
	GAT	19.02	17.06	22.7	22.46	15.51	22.53	21.53	18.62	16.82	14.9	19.66	23.04
	ChebConv	17.74	20.24	23.04	13.98	15.59	23.2	20.63	20.38	16.92	22.7	18.99	23.24
	GraphSAGE	0.74	0.59	1.23	1.16	3.39	2.36	2.53	1.87	10.45	7.14	8.57	7.74
	BLINK+SAT	8.81	7.23	5.65	4.42	9.14	5.97	5.47	4.88	9.82	8.34	8.75	5.49
	PCA	11.2	8.42	7.05	6.05	9.87	7.33	6.8	5.66	8.4	6.83	6.07	5.46
	Conv-PCA	11.56	11.11	9.08	10.32	8.14	10.05	10.76	9.25	12.69	11.06	10.83	7.3
	KS-PCA	25.7	30.06	35.01	36.38	26.08	27.56	30.84	33.82	26.08	28.52	32.08	36
	D^{2}PT	4.54	5.35	5.57	4.84	3.05	6.17	4.82	6.59	5.93	4.46	5.01	5.37
	T2-GNN	24.75	31.07	36.42	38.55	25.01	32.15	36.22	37.74	22.56	29.82	35.15	37.11
	KS-GNN	29.52	36.37	37.92	39.00	30.07	36.41	38.85	39.00	27.08	37.48	37.66	36.02
	Proposed Method	**34.84**	**43.3**	**45.05**	**46.49**	**34.67**	**42.68**	**44.95**	**46.94**	**34.79**	**43.85**	**45.73**	**47.03**
DBLP	GCN	2.75	5.27	3.05	8.04	7.27	2.44	7.17	7.6	5.86	3.85	9.12	11.31
	GAT	4.41	4.77	10.09	2.35	5.93	5.22	10.74	3.68	5.26	12.97	8.72	9.63
	ChebConv	4.56	9.41	5.68	8.96	6.4	4.17	8.5	8.68	7.54	11.03	8.63	17.36
	GraphSAGE	0.42	0.49	0.36	0.82	0.29	0.43	0.53	0.05	0.07	0.01	0.01	0.00
	BLINK+SAT	3.26	6.09	4.01	6.65	3.49	1.66	5.42	3.95	4.25	2.42	3.12	4.24
	PCA	3.78	2.57	2.25	2.38	3.06	2.55	2.21	2.15	2.97	2.51	2.02	1.88
	Conv-PCA	9.00	13.24	13.29	16.87	5.93	6.56	7.64	10.62	7.00	10.52	10.03	15.68
	KS-PCA	15.28	21.41	25.61	31.64	14.98	20.73	23.21	31.72	12.49	19.49	21.23	28.63
	D^{2}PT	5.72	5.83	6.22	5.64	5.11	5.98	5.61	5.91	5.03	4.89	5.55	5.28
	T2-GNN	15.42	23.85	27.31	30.67	15.88	23.14	27.18	30.03	14.75	22.78	25.79	30.12
	KS-GNN	16.21	24.94	29.55	33.51	16.52	22.73	26.85	30.69	15.57	24.15	27.12	29.06
	Proposed Method	**23.11**	**32.29**	**39.38**	**40.54**	**24.01**	**33.46**	**38.01**	**42.13**	**23.33**	**32.46**	**39.16**	**40.86**
Video	GCN	13.66	16.06	19.56	25.28	16.14	19.46	19.99	23.85	6.49	6.79	11.31	12.06
	GAT	10.47	13.4	9.15	11.35	11.29	12.16	14.5	19.12	8.84	12.26	13.46	13.84
	ChebConv	4.45	4.3	4.57	8.49	8.19	12.91	8.4	7.05	5.77	5.96	9.44	13.92
	GraphSAGE	0.49	0.30	0.06	0.03	0.44	0.14	0.00	0.05	0.34	0.35	0.21	0.16
	BLINK+SAT	10.21	9.86	10.99	14.87	8.55	6.92	8.63	5.82	1.18	1.15	4.38	3.35
	PCA	1.54	0.91	0.55	0.61	1.71	0.72	0.71	0.57	1.66	0.95	0.66	0.55
	Conv-PCA	1.81	2.46	1.58	2.38	2.43	1.49	1.66	2.54	2.42	2.37	2.80	3.37
	KS-PCA	10.19	12.23	16.15	21.37	11.26	15.62	19.51	23.87	10.66	15.57	19.17	25.36
	KS-GNN	21.43	23.36	22.92	26.79	22.54	22.57	30.41	33.41	21.01	16.48	22.01	28.47
	Proposed Method	**23.33**	**30.13**	**34.89**	**38.71**	**22.76**	**29.56**	**34.27**	**38.11**	**21.6**	**27.71**	**32.73**	**36.18**
Toy	GCN	18.97	21.27	19.13	21.44	18.47	20.26	19.07	22.34	18.03	19.40	18.91	20.73
	GAT	20.84	22.41	22.6	19.65	21.39	27.43	29.82	31.09	19.16	20.87	21.36	21.80
	ChebConv	15.5	18.08	22.63	12.0	14.94	21.01	24.27	19.85	16.3	17.48	19.97	12.35
	GraphSAGE	0.74	0.88	5.11	4.21	4.24	12.49	6.50	6.10	1.45	4.39	6.71	1.09
	BLINK+SAT	6.47	8.17	8.12	10.54	6.79	9.59	10.99	11.93	6.44	8.56	11.55	8.15
	PCA	1.15	0.68	0.63	0.51	1.01	0.69	0.51	0.44	0.67	0.47	0.44	0.30
	Conv-PCA	21.34	21.99	23.76	25.40	19.17	19.72	20.21	25.22	16.99	21.61	23.95	24.90
	KS-PCA	27.23	27.73	31.58	33.79	25.78	28.94	31.04	32.82	18.35	22.03	25.50	26.25
	KS-GNN	28.56	29.85	29.55	34.28	24.65	29.16	31.27	33.25	21.78	27.41	25.55	30.17
	Proposed Method	**31.82**	**40.5**	**43.87**	**45.31**	**32.19**	**41.41**	**43.49**	**45.18**	**24.09**	**36.31**	**40.75**	**41.92**

(1) Our proposed method has been able to show consistently better performance compared to the baselines on all query sizes and for a different rate of missing keywords. Even when compared to KS-GNN, the state-of-the-art baseline for keyword search on incomplete graphs. We observe a consistent and significant improvement under various conditions.
(2) It can be seen that a change in the size of keywords or the missing rate can cause a change in performance for some methods while performance remains in the same range for the rest. More specifically, GCN, GAT, ChebConv, PCA, T2-GNN and KS-GNN are the methods that suffer performance drop by increasing the missing keyword ratio. On the other hand, the

Table 5. Performance for our proposed method and baselines by Hits@100 (%) with Missing Keywords and 30% Missing Edges. T2-GNN and D^{2}PT employ supervised approaches and are evaluated on labeled datasets, i.e., Citeseer and DBLP.

Datasets	r_w	30% Missing Keywords				50% Missing Keywords				70% Missing Keywords			
	n_q	3	5	7	9	3	5	7	9	3	5	7	9
CiteSeer	GCN	19.00	17.86	21.19	22.44	13.7	15.18	15.82	11.46	11.27	19.34	22.43	21.33
	GAT	10.58	8.88	9.9	12.44	12.2	18.52	14.36	21.24	12.96	15.98	14.83	12.41
	ChebConv	18.33	28.27	21.06	17.54	17.89	16.54	28.33	26.67	18.99	22.42	29.54	28.77
	GraphSAGE	1.25	1.03	0.75	1.03	3.26	2.45	3.34	1.62	9.44	7.86	7.59	5.45
	BLINK+SAT	9.19	8.14	5.68	3.71	8.26	6.79	6.04	8.57	7.88	8.33	7.69	8.3
	PCA	11.2	8.42	7.05	6.05	9.87	7.33	6.8	5.66	8.4	6.83	6.07	5.46
	Conv-PCA	11.6	10.97	9.05	10.56	9.77	10.61	12.73	9.31	12.92	11.83	12.25	8.45
	KS-PCA	26.08	30.06	32.26	34.65	23.8	26.45	29.11	31.81	26.06	26.57	31.42	34.53
	D^{2}PT	4.07	3.48	5.78	3.04	5.34	3.45	5.47	2.88	2.93	1.76	2.86	4.73
	T2-GNN	24.79	31.16	34.29	38.61	23.83	30.69	33.72	36.98	22.35	28.52	32.86	35.17
	KS-GNN	28.96	35.75	39.44	39.4	26.48	34.92	36.31	37.3	24.2	27.84	28.19	33.41
	Proposed Method	**32.79**	**41.72**	**42.45**	**43.63**	**33.28**	**40.74**	**41.59**	**42.48**	**33.17**	**41.72**	**41.72**	**43.16**
DBLP	GCN	4.85	6.71	5.89	3.62	2.91	10.47	9.53	8.5	2.21	2.79	1.41	4.56
	GAT	3.27	3.74	7.37	7.11	3.71	4.49	5.55	8.04	5.63	7.73	8.34	7.63
	ChebConv	5.51	8.51	12.55	4.24	7.66	12.24	11.94	13.09	6.3	6.44	10.21	14.2
	GraphSAGE	0.51	0.20	0.42	0.34	0.67	0.25	0.40	0.90	0.40	0.05	0.04	0.02
	BLINK+SAT	4.78	3.45	6.74	4.58	3.94	3.66	5.05	3.97	2.96	2.75	2.83	7.29
	PCA	3.78	2.57	2.25	2.38	3.06	2.55	2.21	2.15	2.97	2.51	2.02	1.88
	Conv-PCA	8.35	11.8	12.51	14.51	5.25	6.77	7.03	9.91	6.46	9.59	9.47	14.33
	KS-PCA	15.05	19.99	22.61	28.57	13.68	19.19	21.59	28.96	11.67	17.37	18.38	24.51
	D^{2}PT	4.18	5.63	6.05	5.72	4.51	5.03	5.67	5.96	3.32	4.08	4.41	4.82
	T2-GNN	14.83	19.95	22.25	27.72	12.21	18.78	21.35	25.90	10.90	16.20	18.89	24.13
	KS-GNN	15.91	20.49	25.09	29.04	15.79	19.86	22.15	29.71	12.07	17.18	19.23	24.19
	Proposed Method	**22.29**	**29.88**	**35.22**	**37.36**	**19.22**	**25.86**	**29.46**	**30.90**	**19.79**	**26.36**	**29.65**	**31.81**
Video	GCN	2.83	2.95	4.5	4.69	1.62	2.8	4.14	2.94	2.19	2.19	3.01	3.04
	GAT	3.01	2.13	3.01	2.52	2.51	3.99	3.76	3.84	4.27	3.0	4.31	3.64
	ChebConv	3.35	4.08	4.85	4.76	2.65	3.03	4.46	4.54	1.96	2.55	5.67	2.9
	GraphSAGE	0.09	0.11	0.02	0.00	0.26	0.05	0.00	0.04	1.65	1.65	2.22	1.29
	BLINK+SAT	1.67	1.85	2.48	1.44	0.08	0.99	4.96	2.97	2.19	1.77	0.78	1.21
	PCA	1.54	0.91	0.55	0.61	1.71	0.72	0.71	0.57	1.66	0.95	0.66	0.55
	Conv-PCA	1.05	1.59	0.83	2.01	1.43	0.81	0.87	1.23	1.25	1.25	1.31	1.38
	KS-PCA	3.82	4.13	4.88	7.13	3.96	4.52	5.33	6.11	3.64	4.58	5.17	6.55
	KS-GNN	8.08	8.34	12.88	11.82	6.84	7.68	4.18	11.12	6.37	10.31	13.92	10.07
	Proposed Method	**15.31**	**18.74**	**21.8**	**24.95**	**14.74**	**18.98**	**22.09**	**25.03**	**13.87**	**17.17**	**20.02**	**22.39**
Toy	GCN	2.64	2.37	1.78	2.99	2.45	2.55	2.04	2.93	2.18	2.21	2.11	2.30
	GAT	2.22	1.45	2.77	0.92	1.01	0.87	1.11	0.64	1.16	0.93	0.74	0.88
	ChebConv	6.93	6.15	11.69	7.82	7.25	8.25	9.03	8.91	3.09	3.83	4.46	5.17
	GraphSAGE	0.04	0.02	0.01	0.00	0.28	0.00	0.01	0.02	0.02	0.18	0.29	0.23
	BLINK+SAT	3.69	3.03	4.85	5.66	2.56	1.34	1.86	5.46	1.76	1.39	4.73	4.45
	PCA	1.15	0.68	0.63	0.51	1.01	0.69	0.51	0.44	0.67	0.47	0.44	0.30
	Conv-PCA	11.64	10.46	11.23	12.29	9.31	9.00	9.08	11.61	6.74	8.87	8.99	9.56
	KS-PCA	13.80	13.03	13.55	15.67	11.35	11.03	11.09	13.02	6.84	8.37	9.50	10.43
	KS-GNN	13.82	13.28	14.38	15.22	12.51	12.54	12.68	12.59	9.41	8.99	12.83	11.15
	Proposed Method	**22.72**	**28.62**	**30.26**	**31.13**	**22.09**	**28.57**	**30.19**	**31.85**	**17.32**	**22.96**	**23.49**	**24.03**

rest of the methods, GraphSAGE, BLINK+SAT, Conv-PCA, KS-PCA, D^{2}PT and our proposed method, show minor or no performance drop when the missing keyword ratio increases. This can be interpreted as a result of a robust learning technique that properly captures the structural and semantical characteristics of graphs.

(3) While the performance for the mentioned group remains stable, the overall performance varies across methods. For example, the GraphSAGE method shows a significantly poor performance compared to the rest. This is because GraphSAGE only utilizes the message passing and max-pooling operator to gather information from neighbour nodes. This causes

the model to be unable to distinguish the source of each unique keyword. Smoothing the neighbourhood information combined with missing values makes the representation generated by GraphSAGE almost non-functional for the purpose of keyword search in incomplete graphs. Furthermore, we observe that the D^2PT method also performs poorly in keyword search overall. This is because the learned representations may not directly optimize for keyword retrieval. Moreover, over-propagation in diffusion-based propagation mechanism can lead to over-smoothing, where node representations become too similar across the graph.

(4) In the PCA-based group of methods, we can see that PCA and KS-PCA suffer a performance drop compared to Conv-PCA when increasing the missing keyword ratio. This is because PCA generates embedding per node, and loss of information directly affects the performance. On the contrary, although the Conv-PCA does not perform well, it remains robust against missing information. This is because Conv-PCA uses a graph convolutional neural layer to aggregate and refine node representations from the graph's data, capturing both local and global structural patterns effectively. In addition, Conv-PCA uses a decoder unit; therefore, the reconstruction loss is also taken into account.

(5) GCN, GAT, and ChebConv models are showing weak performance overall. These models use graph neural networks along with different techniques to generate representations. GCN uses a convolutional layer to learn the node representations that incorporate information from their neighbourhoods. ChebConv is based on the Chebyshev Spectral Graph Convolution framework, which utilizes Chebyshev polynomials of the Laplacian matrix to efficiently compute graph convolutions. GAT dynamically determines the importance of each neighbour's features through a self-attention mechanism. While each of these models uses a unique approach to capture graph information, they all fail to achieve high performance when it comes to incomplete graphs. This shows that missing information in graphs needs a special mechanism to focus on the structure and features of nodes in the graph. Our proposed method uses a multi-aspect graph representation method as a solution for this.

(6) BLINK+SAT uses SAT [11], a GNN-based graph completion model, to reconstruct a graph without missing information and then uses BLINK [23] to index the graph and process keyword search. The BLINK+SAT performance is not significant because the graph completion process is a sensitive and unreliable task. The model also loses more performance when the missing information ratio increases. This is because graph completion tasks become more and more challenging when the number of missing information increases.

(7) We also observe that baselines that adopt a decoder in their architecture, i.e. KS-GNN, KS-PCA, Conv-PCA and our proposed method, are achieving higher performance in comparison to those without a decoder. This shows that decoders, in general, help models achieve better performance by adding reconstruction loss to their training loss function. To better assess this fact, the Conv-PCA performance should be compared to the ordinary PCA method.

(8) The proposed method shows consistent good performance when the size of the input query increases. This shows that the method can learn additional contextual information from the presence of additional keywords in the query. This pattern is also observed for the KS-GNN, KS-PCA and Conv-PCA methods; however, the degree of improvement and the performance measures are lower than our proposed approach.

Moving to the second scenario, where graph experiences both missing keywords and missing edges, we make several observations based on the result from Table 5:

(1) The overall performance for the majority of the methods drops by introducing the missing edge challenge to the problem. GCN, GAT, ChenConv, GraphSage, BLINK+SAT, PCA and ConvPCA are all subject to this pattern. This shows that the mentioned methods are not robust against missing edges, and the structural properties of the graph are not captured efficiently for search in incomplete graphs. While this was expected, we observe that for GCN and ChebConv methods, the pattern is the opposite for CiteSeer dataset particularly. This can be due to CiteSeer being a relatively small graph, and alteration cannot hurt the structure significantly.
(2) BLINK+SAT shows an exceptionally reduced performance when facing missing edges. BLINK+SAT uses a graph completion technique for missing information recovery. This proves that this technique may not be a good solution for search in incomplete graphs.
(3) For our proposed method and the baselines, including KS-GNN, the state-of-the-art model, the missing edges impact differently on each dataset. Our proposed method is more robust on CiteSeer and DBLP datasets. While it outperforms the other methods on the Toy and Video dataset, the performance is more impacted by the missing edge. It can be concluded that graph attributes, such as the number of nodes, edges and keywords, can contribute to the model performance and tolerance against missing values.
(4) We can conclude that our proposed method performs better than all baselines, even for cases when both missing keywords and edges exist. It can be seen that despite the missing values in both edges and keywords, our proposed method consistently outperformed others. This is because the proposed method crafts graph representations from four different perspectives, providing complementary information to deal with missing values.

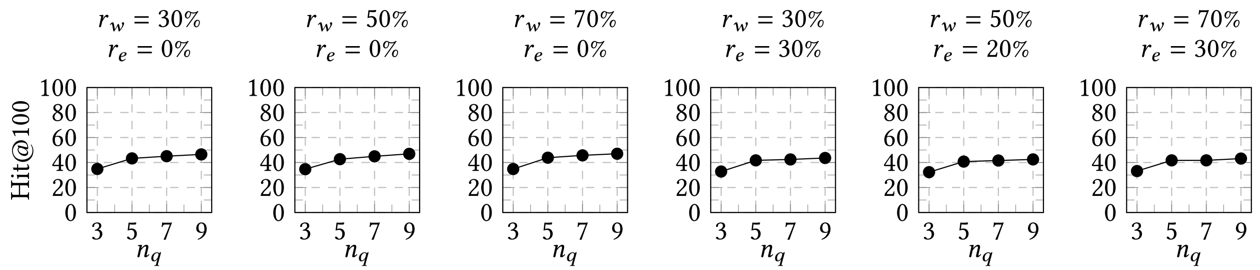
5.7 Sensitivity Analysis

This sensitivity analysis investigates the impact of keyword size on the performance of our proposed search method, focusing on its responsiveness and robustness to keyword size variations. We examine the behaviour of our model with query keyword sizes of $n_q = \{3, 5, 7, 9\}$. The analysis aims to demonstrate the extent to which keyword size influences the efficacy of our proposed method. The results can be seen in Figure 4. All other setup parameters remain the same during the study. We studied performance with several rates of missing keywords (r_w) and edges (r_e) to ensure that observed performance trends could be attributed solely to changes in keyword size.

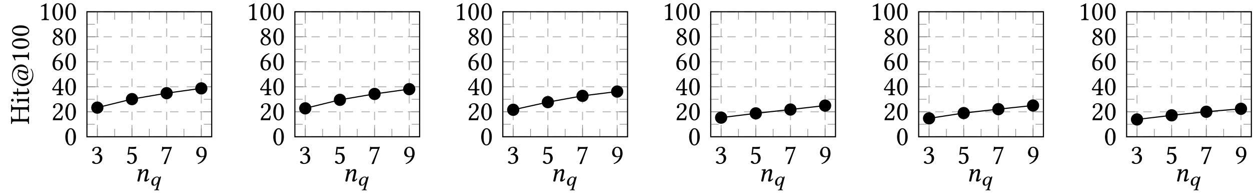
The results reveal distinct trends in the performance of our proposed search method as the keyword size increases. Notably, our method exhibited a better Hit@k score compared to smaller keyword sizes (3 and 5) across all four datasets. This indicates a positive correlation between keyword size and search performance. Our model maintained a competitive edge as the keyword size increased to 7 and 9, suggesting its robustness in capturing relevant information even with a larger set of keywords, however the rate of performance increase slowed down and near to flat in Citeseer and DBLP datasets. This shows that having a large number of unique keywords (i.e. Amazon Video dataset) or a large number of edges (i.e. Amazon Toy dataset) plays a role in determining the ideal minimum number of keywords per node.

Our proposed method's notable performance with smaller keyword sizes can be attributed to its sophisticated mechanism for handling missing information, allowing for more precise and relevant search results. As keyword size expands, the complexity of the search increases, yet our method demonstrates commendable adaptability and effectiveness. This demonstrates that the method is capable of capturing a greater range of semantics as the information density of the graph increases.

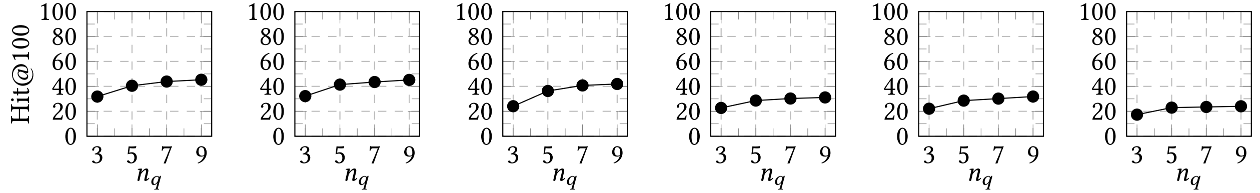
Overall, our method shows to be *robust* to missing keywords from the graph. It can be observed in Figure 4 that the performance of the method at different conditions for $n_q = 9$ remains comparable regardless of the percentage of missing keywords. However, according to Tables 4 and 5, this is not



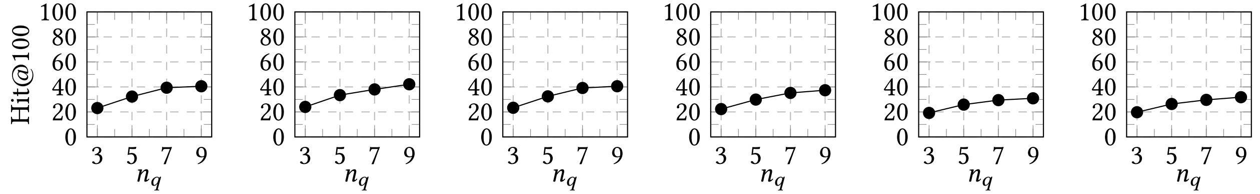
(a) Citeseer Dataset [44].



(b) Amazon Video Dataset [38].



(c) Amazon Toy Dataset [38].



(d) DBLP Dataset [49].

Fig. 4. Sensitivity Analysis of our proposed method for Citeseer, Video, Toy and DBLP datasets. These figures illustrate the performance impact of varying keyword sizes (n_q) on the effectiveness of keyword searches. Each subplot represents the search sensitivity under different keyword missing rates (r_w) and edge missing rates (r_e).

applicable to the best baseline KS-GNN method. This is due to our proposed method's enhanced ability to capture more structural information from the graph, surpassing that of KS-GNN.

5.8 Discussion on Satisfying Design Objectives

Our work in this paper has been motivated by three objectives that were earlier discussed in Section 3: Reduce the dimension of the output to make search space feasible (O_1), retain as much key information as possible despite missing Keywords and edge information (O_2) and capture the structural properties of the graph to the greatest extent to overcome the challenge of missing edges (O_3).

The results obtained from our experiments show that each of the objectives played a fundamental role in our proposed method. Methods that do not practice representation learning techniques, such as PCA, show no improvement by increasing the size of query keywords (n_q). PCA, Conv-PCA and KS-PCA also experience more drop in performance when the ratio of the missing keyword (r_w) increases and the size of the graphs increases. On the other hand, our proposed method and other graph representation learning methods maintain consistent performance over all datasets. This is due to the integration of **objective 1** (O_1) in our design.

Furthermore, ChebConv and ConvPCA methods perform significantly lower compared to our proposed approach on the video dataset, which has an extreme number of keywords. This shows that these methods are not storing keyword information efficiently. Moreover, the GCN and GAT methods perform poorly on the DBLP dataset. The DBLP dataset has a larger number of nodes with a relatively low number of keywords, which makes it sparse compared to the Video dataset. This shows that while GCN and GAT methods are robust against the density of keywords, they may not be able to capture key information efficiently in sparse graphs. Our proposed method demonstrates a stable performance despite keyword size and graph size thanks to addressing **objective 2** (O_2).

Moreover, when comparing the no missing edges scenario and 30% missing edges results, we notice a significant performance drop in a certain group of methods. A bold difference can be seen specifically in the Toy dataset, which has a large number of edges. GCN, GAT, ChebConv and BLINK+SAT are the methods with the most reduction. This means that these methods are not efficient in capturing the structural properties of the graphs, which was expressed in **objective 3**. Although missing edges affect the performance of our proposed method, this is shown to be less concerning in our experiments. This is due to the fact our model uses different semantics to capture graph structural characteristics to a greater extent as a result of addressing **objective 3** (O_3).

6 Concluding Remarks and Future Work

Our proposed method for searching over incomplete graphs is based on the idea of graph representations learning from multiple complementary perspectives. The effectiveness of our proposed approach is due to its reliance on multiple perspectives when learning graph representations, which makes it robust to *missing information*. This advantage helps in preserving graph structure and captures both the semantic and structural information of the graph to a great extent. To the best of our knowledge, our work is among the first that proposed a custom-tailored loss function to generate such representation in incomplete graphs. In our experiments, we compared our proposed method with various baselines and state-of-the-art approaches. We show through various experiments that our method outperforms the state of the art when missing information is present. We find that our approach shows superior performance compared to the other methods.

We propose three potential topics to study as future directions:

(1) **Uncertain Graphs:** Real-world graphs often include a degree of uncertainty. Nodes and edges in such graphs may not always exist definitively; instead, they might be present with

a certain probability. This notion leads us to the concept of uncertain graphs, where each entity (node or edge) is associated with a probability denoting its likelihood of existence. As future work, we can study techniques that must go beyond capturing the structural and semantic properties of nodes and edges, incorporating the probabilistic information that defines the likelihood of their existence.

(2) **Graph Completion:** Focusing on the idea of leveraging advanced methodologies for the graph completion task, this concept revolves around the principle of utilizing predictive modeling to infer and reconstruct missing nodes and edges, thereby approximating a more complete structure of the graph. The trained model can then be used to predict missing elements in incomplete graphs by identifying structural gaps or inconsistencies and filling them in with the most probable nodes or connections. This allows for a broader exploration of the graph during keyword searches.

(3) **Multi-view Optimization:** One potential limitation lies in the use of multi-view enhancements. While they are commonly employed to enrich feature representations, redundant information across different views may lead to the learning of duplicate feature representations [46]. As a future work can explore adaptive view selection mechanisms that dynamically weight the contribution of different views based on their unique information content.

References

[1] Alireza Abbasi, Liaquat Hossain, and Rolf T. Wigand. 2015. Evolutionary Dynamics of Complex Networks: Theory, Methods and Applications. *CoRR* abs/1503.06652 (2015). arXiv:1503.06652 http://arxiv.org/abs/1503.06652

[2] Réka Albert, Hawoong Jeong, and Albert-László Barabási. 2000. Error and attack tolerance of complex networks. *nature* 406, 6794 (2000), 378–382.

[3] R.E. Bellman. 2003. *Dynamic Programming.* Dover Publications. https://books.google.ca/books?id=fyVtp3EMxasC

[4] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. Generalized Denoising Auto-Encoders as Generative Models. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 899–907. https://proceedings.neurips.cc/paper/2013/hash/559cb990c9dffd8675f6bc2186971dc2-Abstract.html

[5] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, and S. Sudarshan. 2002. Keyword Searching and Browsing in Databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, Rakesh Agrawal and Klaus R. Dittrich (Eds.). IEEE Computer Society, 431–440. https://doi.org/10.1109/ICDE.2002.994756

[6] Ranran Bian, Yun Sing Koh, Gillian Dobbie, and Anna Divoli. 2019. Network Embedding and Change Modeling in Dynamic Heterogeneous Networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019*, Benjamin Piwowarski, Max Chevalier, Éric Gaussier, Yoelle Maarek, Jian-Yun Nie, and Falk Scholer (Eds.). ACM, 861–864. https://doi.org/10.1145/3331184.3331273

[7] Alex Bigelow, Katy Williams, and Katherine E. Isaacs. 2021. Guidelines For Pursuing and Revealing Data Abstractions. *IEEE Trans. Vis. Comput. Graph.* 27, 2 (2021), 1503–1513. https://doi.org/10.1109/TVCG.2020.3030355

[8] Jean Bourgain. 1985. On Lipschitz embedding of finite metric spaces in Hilbert space. *Israel Journal of Mathematics* 52, 1 (1985), 46–52.

[9] Daqing Chen. 2015. Online Retail. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5BW33.

[10] Min Chen, Shiwen Mao, and Yunhao Liu. 2014. Big data: A survey. *Mobile networks and applications* 19 (2014), 171–209. https://doi.org/10.1007/s11036-013-0489-0

[11] Xu Chen, Siheng Chen, Jiangchao Yao, Huangjie Zheng, Ya Zhang, and Ivor W. Tsang. 2022. Learning on Attribute-Missing Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 2 (2022), 740–757. https://doi.org/10.1109/TPAMI.2020.3032189

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, 3rd Edition.* MIT Press. http://mitpress.mit.edu/books/introduction-algorithms

[13] Bhavana Bharat Dalvi, Meghana Kshirsagar, and S. Sudarshan. 2008. Keyword search on external memory data graphs. *Proc. VLDB Endow.* 1, 1 (2008), 1189–1204. https://doi.org/10.14778/1453856.1453982

[14] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). 3837–3845. https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html

[15] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. Turl: Table understanding through representation learning. *ACM SIGMOD Record* 51, 1 (2022), 33–40.

[16] Xin Luna Dong. 2019. Building a broad knowledge graph for products. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 25–25.

[17] Tlamelo Emmanuel, Thabiso M. Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago, and Oteng Tabona. 2021. A survey on missing data in machine learning. *J. Big Data* 8, 1 (2021), 140. https://doi.org/10.1186/s40537-021-00516-9

[18] Chao Gao and Jiming Liu. 2011. Modeling and predicting the dynamics of mobile virus spread affected by human behavior. In *12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WOWMOM 2011, Lucca, Italy, 20-24 June, 2011*. IEEE Computer Society, 1–9. https://doi.org/10.1109/WOWMOM.2011.5986383

[19] Jan J. Gerbrands. 1981. On the relationships between SVD, KLT and PCA. *Pattern Recognit.* 14, 1-6 (1981), 375–381. https://doi.org/10.1016/0031-3203(81)90082-0

[20] Gang Gou and Rada Chirkova. 2008. Efficient algorithms for exact ranked twig-pattern matching over graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, Jason Tsong-Li Wang (Ed.). ACM, 581–594. https://doi.org/10.1145/1376616.1376676

[21] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[22] Yu Hao, Xin Cao, Yufan Sheng, Yixiang Fang, and Wei Wang. 2021. KS-GNN: Keywords Search over Incomplete Graphs via Graphs Neural Network. *Advances in Neural Information Processing Systems* 34 (2021).

[23] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. 2007. BLINKS: ranked keyword searches on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou (Eds.). ACM, 305–316. https://doi.org/10.1145/1247480.1247516

[24] Ningyuan Teresa Huang and Soledad Villar. 2021. A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 8533–8537. https://doi.org/10.1109/ICASSP39728.2021.9413523

[25] Cuiying Huo, Di Jin, Yawen Li, Dongxiao He, Yu-Bin Yang, and Lingfei Wu. 2023. T2-GNN: Graph Neural Networks for Graphs with Incomplete Features and Structure via Teacher-Student Distillation. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 4339–4346. https://doi.org/10.1609/AAAI.V37I4.25553

[26] Mehdi Kargar, Lukasz Golab, Divesh Srivastava, Jaroslaw Szlichta, and Morteza Zihayat. 2022. Effective Keyword Search Over Weighted Graphs. *IEEE Trans. Knowl. Data Eng.* 34, 2 (2022), 601–616. https://doi.org/10.1109/TKDE.2020.2985376

[27] Mehdi Kargar, Lukasz Golab, and Jaroslaw Szlichta. 2016. eGraphSearch: Effective Keyword Search in Graphs. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM, 2461–2464. https://doi.org/10.1145/2983323.2983333

[28] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1312.6114

[29] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=SJU4ayYgl

[30] Wangchao Le, Feifei Li, Anastasios Kementsietsidis, and Songyun Duan. 2014. Scalable Keyword Search on Large RDF Data. *IEEE Trans. Knowl. Data Eng.* 26, 11 (2014), 2774–2788. https://doi.org/10.1109/TKDE.2014.2302294

[31] Nathan Linial, Eran London, and Yuri Rabinovich. 1995. The Geometry of Graphs and Some of its Algorithmic Applications. *Comb.* 15, 2 (1995), 215–245. https://doi.org/10.1007/BF01200757

[32] Yixin Liu, Kaize Ding, Jianling Wang, Vincent C. S. Lee, Huan Liu, and Shirui Pan. 2023. Learning Strong Graph Neural Networks with Weak Information. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.). ACM, 1559–1571. https://doi.org/10.1145/3580305.3599410

[33] Julian J. McAuley and Jure Leskovec. 2012. Learning to Discover Social Circles in Ego Networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger (Eds.). 548–556. https://proceedings.neurips.cc/paper/2012/hash/7a614fd06c325499f1680b9896beedeb-Abstract.html

[34] Yujie Mo, Liang Peng, Jie Xu, Xiaoshuang Shi, and Xiaofeng Zhu. 2022. Simple unsupervised graph representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7797–7805.

[35] Felix Naumann, Johann Christoph Freytag, and Ulf Leser. 2004. Completeness of integrated information sources. *Inf. Syst.* 29, 7 (2004), 583–615. https://doi.org/10.1016/J.IS.2003.12.005

[36] Hoang Nguyen, Radin Hamidi Rad, and Ebrahim Bagheri. 2022. PyDHNet: A Python Library for Dynamic Heterogeneous Network Representation Learning and Evaluation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 4936–4940. https://doi.org/10.1145/3511808.3557181

[37] Hoang Nguyen, Radin Hamidi Rad, Fattane Zarrinkalam, and Ebrahim Bagheri. 2023. DyHNet: Learning dynamic heterogeneous network representations. *Inf. Sci.* 646 (2023), 119371. https://doi.org/10.1016/J.INS.2023.119371

[38] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, 188–197. https://doi.org/10.18653/V1/D19-1018

[39] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning Convolutional Neural Networks for Graphs. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.). JMLR.org, 2014–2023. http://proceedings.mlr.press/v48/niepert16.html

[40] Radin Hamidi Rad, Silviu Cucerzan, Nirupama Chandrasekaran, and Michael Gamon. 2024. Interactive Topic Tagging in Community Question Answering Platforms. In *Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part III (Lecture Notes in Computer Science, Vol. 14610)*, Nazli Goharian, Nicola Tonellotto, Yulan He, Aldo Lipani, Graham McDonald, Craig Macdonald, and Iadh Ounis (Eds.). Springer, 195–209. https://doi.org/10.1007/978-3-031-56063-7_13

[41] Radin Hamidi Rad, Aabid Mitha, Hossein Fani, Mehdi Kargar, Jaroslaw Szlichta, and Ebrahim Bagheri. 2021. PyTFL: A Python-based Neural Team Formation Toolkit. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*, Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.). ACM, 4716–4720. https://doi.org/10.1145/3459637.3481992

[42] Radin Hamidi Rad, Hoang Nguyen, Feras N. Al-Obeidat, Ebrahim Bagheri, Mehdi Kargar, Divesh Srivastava, Jaroslaw Szlichta, and Fattane Zarrinkalam. 2023. Learning heterogeneous subgraph representations for team discovery. *Inf. Retr. J.* 26, 1 (2023), 8. https://doi.org/10.1007/S10791-023-09421-6

[43] Nils Reimers and Iryna Gurevych. 2021. The Curse of Dense Low-Dimensional Information Retrieval for Large Index Sizes. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 2: Short Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (Eds.). Association for Computational Linguistics, 605–611. https://doi.org/10.18653/V1/2021.ACL-SHORT.77

[44] Ryan A. Rossi and Nesreen K. Ahmed. 2015. The Network Data Repository with Interactive Graph Analytics and Visualization. In *AAAI*. https://networkrepository.com

[45] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 815–823. https://doi.org/10.1109/CVPR.2015.7298682

[46] Mingchao Shang, Cheng Liang, Jiawei Luo, and Huaxiang Zhang. 2023. Incomplete multi-view clustering by simultaneously learning robust representations and optimal graph structures. *Inf. Sci.* 640 (2023), 119038. https://doi.org/10.1016/J.INS.2023.119038

[47] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.* 12 (2011), 2539–2561. http://dl.acm.org/citation.cfm?id=2078187

[48] Qi Song, Yinghui Wu, Peng Lin, Luna Xin Dong, and Hui Sun. 2018. Mining summaries for knowledge graph search. *IEEE Transactions on Knowledge and Data Engineering* 30, 10 (2018), 1887–1900.

[49] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Ying Li, Bing Liu, and Sunita Sarawagi (Eds.). ACM, 990–998. https://doi.org/10.1145/1401890.1402008

[50] Thanh Tran, Haofen Wang, Sebastian Rudolph, and Philipp Cimiano. 2009. Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In *Proceedings of the 25th International Conference on Data Engineering, ICDE 2009, March 29 2009 - April 2 2009, Shanghai, China*, Yannis E. Ioannidis, Dik Lun Lee, and Raymond T. Ng (Eds.). IEEE Computer Society, 405–416. https://doi.org/10.1109/ICDE.2009.119

[51] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[52] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJXMpikCZ

[53] Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro A. Szekely. 2021. Retrieving Complex Tables with Multi-Granular Graph Representation Learning. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 1472–1482. https://doi.org/10.1145/3404835.3462909

[54] Kilian Q. Weinberger and Lawrence K. Saul. 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *J. Mach. Learn. Res.* 10 (2009), 207–244. https://dl.acm.org/citation.cfm?id=1577078

[55] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*. 1271–1279.

[56] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[57] Dejian Yang, Senzhang Wang, Chaozhuo Li, Xiaoming Zhang, and Zhoujun Li. 2017. From Properties to Links: Deep Network Embedding on Incomplete Graphs. In *CIKM*. ACM, 367–376.

[58] Jianye Yang, Wu Yao, and Wenjie Zhang. 2021. Keyword Search on Large Graphs: A Survey. *Data Sci. Eng.* 6, 2 (2021), 142–162. https://doi.org/10.1007/S41019-021-00154-4

[59] Dayu Yuan and Prasenjit Mitra. 2013. Lindex: a lattice-based index for graph databases. *VLDB J.* 22, 2 (2013), 229–252. https://doi.org/10.1007/S00778-012-0284-8

[60] Haopeng Zhang and Jiawei Zhang. 2020. Text Graph Transformer for Document Classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 8322–8327. https://doi.org/10.18653/V1/2020.EMNLP-MAIN.668

[61] Jiawei Zhang. 2020. Segmented Graph-Bert for Graph Instance Modeling. *CoRR* abs/2002.03283 (2020). arXiv:2002.03283 https://arxiv.org/abs/2002.03283

[62] Jiawei Zhang, Haopeng Zhang, Congying Xia, and Li Sun. 2020. Graph-Bert: Only Attention is Needed for Learning Graph Representations. *CoRR* abs/2001.05140 (2020). arXiv:2001.05140 https://arxiv.org/abs/2001.05140

[63] Yan Zhang and Mingli Jing. 2023. Keyword Search of Incomplete Graph Based on Kernel Principal Component Analysis. In *2023 5th International Conference on Intelligent Control, Measurement and Signal Processing (ICMSP)*. IEEE, 447–450.

[64] Bin Zhou and Jian Pei. 2008. Preserving Privacy in Social Networks Against Neighborhood Attacks. In *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, Gustavo Alonso, José A. Blakeley, and Arbee L. P. Chen (Eds.). IEEE Computer Society, 506–515. https://doi.org/10.1109/ICDE.2008.4497459

[65] Jingya Zhou, Ling Liu, Wenqi Wei, and Jianxi Fan. 2022. Network Representation Learning: From Preprocessing, Feature Extraction to Node Embedding. *ACM Computing Surveys (CSUR)* 55, 2 (2022), 1–35.