

000 001 002 003 004 005 DIRAGNN: ATTENTION-ENHANCED ENTITY RANK- 006 ING FOR SPARSE GRAPH NETWORKS 007 008 009

010 **Anonymous authors**
 011

012 Paper under double-blind review
 013
 014
 015
 016
 017
 018
 019
 020
 021
 022
 023

ABSTRACT

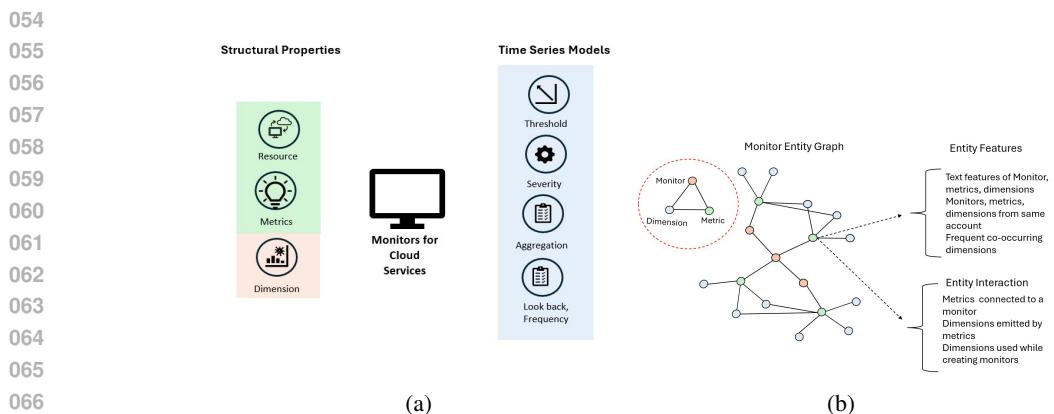
024 Sparsity in both the structural and engagement information presents a core chal-
 025 lenge in entity ranking problems for graph networks. The interaction dynamics of
 026 entities are often characterized by limited structural and engagement information
 027 which results in inferior performance of the state-of-the-art approaches. In this
 028 work, we present DiRaGNN, an attention-enhanced entity ranking model designed
 029 to address the problem of dimension recommendation and ranking for automated
 030 watchdogs in the cloud setting. DiRaGNN is inspired by transformer architec-
 031 tures and utilizes a multi-head attention mechanism to focus on heterogeneous
 032 neighbors and their attributes. Additionally, our model employs multi-faceted
 033 loss functions to optimize for relevant recommendations and reduce popularity
 034 bias. To manage computational complexity, we sample a local subgraph that in-
 035 cludes multiple hops of neighbors. Empirical evaluations demonstrate significant
 036 improvements over existing methods, with our model achieving a 39.7% increase
 037 in MRR.

1 INTRODUCTION

038 Graph neural networks (GNNs) have proven to be efficient representations for a wide range of
 039 real-world systems, including social media graphs (Kipf & Welling, 2016; Hamilton et al., 2017;
 040 Borisuk et al., 2024; Sankar et al., 2021; Zhang et al., 2018; Borisuk et al., 2024). Heteroge-
 041 neous graph neural networks (HGNN), on the other hand, offer the additional flexibility to encode
 042 both structured and unstructured information associated with various node types, such as explicit
 043 links between different nodes and unstructured features associated with nodes, such as texts and
 044 images Zhang et al. (2019). HGNNs use either message-passing to learn effective node representa-
 045 tions from local graph neighborhoods containing structural relations among nodes and unstructured
 046 content Zhang et al. (2019); Hong et al. (2020); Zhao et al. (2021); Hu et al. (2019; 2020), or
 047 metapath-based neighbors Wang et al. (2019); Fu et al. (2020); Yun et al. (2019).
 048

049 **Graph networks for latent entity representation:** Recently, HGNNs have demonstrated promis-
 050 ing results in several industrial systems designed for item recommendations in bipartite Ying et al.
 051 (2018) or multipartite Yang et al. (2020) user-to-item interaction graphs. GCN Kipf & Welling
 052 (2016), GraphSAGE Hamilton et al. (2017), and GAT Velickovic et al. (2017) employ various net-
 053 work architecture and self attention mechanism to aggregate the feature information from neighbor-
 054 ing nodes. Further, scalable extensions to these techniques were introduced in Zeng et al. (2019);
 055 Chiang et al. (2019); Huang et al. (2018). GRAFRank Sankar et al. (2021) extends GNNs for
 056 large-scale user-user social modeling applications and employs multi-modal neighbor aggregators
 057 and cross-modality attentions to learn user representations. Yet, entity ranking using heterogeneous
 058 graph networks with the structural and engagement is still a core challenge which reduces the quality
 059 of the recommended entities. In this work, we investigate this problem in the context of cloud setting
 060 and propose an improved framework for enhanced entity ranking. We focus on the message-passing
 061 approach as it avoids the need for domain experts to mine meta-paths.
 062

063 **Entity ranking in cloud setting:** Recommending attributes (dimensions) for aggregating time-
 064 series signals to create automated watchdogs that ensure continuous service availability is a complex
 065 problem in the cloud setting Surianarayanan & Chelliah (2019); Chen et al. (2020); Montes et al.
 066 (2013). Previous research has focused on recommending time-series signals to be associated with
 067 automated watchdogs Nair et al. (2015); Srinivas et al. (2024). Generating recommendations in a
 068



068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100
101
102
103
104
105
106
107

Figure 1: a) Monitors (automated watch dogs) in a cloud setting: Structural elements include resources, metrics, and dimensions which are utilized to raise alerts, b) Monitor entity ranking problem: Nodes represent the monitors, metrics, and dimensions in a cloud setting. Each node contains text features and interacts with their neighboring nodes using link communication.

cloud setting require leveraging domain knowledge and interaction between multiple entities. Therefore, HGNNs are a natural choice due to their effective message passing mechanisms and ability in capturing intricate relationships between entities with different types of relationship. However, applying HGNNs directly to cloud domain datasets is challenging. This is due to the limited structural information available from interactions and the sparse nature of the graph. Most attributes are not connected to all types of nodes, exhibit fewer degrees, and have fewer interactions. Additionally, different neighboring nodes and node attributes contribute differently to learning node presentations, ranking, and predictions.

Present work: In this work, we begin by discussing the cloud setting and introducing the different entities involved in the recommendation problem. We then introduce the monitor entity graph and formulate the problem. Next, we discuss the specific nature of the graph, including its long-tailed distribution and the sparsity of interactions. To overcome the challenges of structural and interaction sparsity, we propose leveraging the available set of node attributes and multi-type interactions, and present an attention-enhanced entity ranking framework. Specifically, we make the following contributions.

Contributions: We propose, Diverse Ranking for GNN (DiRaGNN), a representation learning framework enhanced with transformer-style graph convolutions. Inspired by transformer architectures, the convolutions incorporate multi-head attention mechanisms, enabling the model to capture complex, long-range dependencies in the graph structure. The edge-aware message passing ensures that the message is sensitive to specific relationship types. Furthermore, the framework incorporates a diversity-aware loss function, which aggregates and attends to various information from different nodes and their interaction patterns. Unlike baseline approaches, DiRaGNN employs a multi-faceted loss function to prioritize relevant recommendations and effectively attend to different node contexts. The diversity loss penalizes similar attention patterns across different heads and learns a comprehensive graph structure, while the ranking loss improves the model’s ability to distinguish between more and less relevant entities. Finally, experiments on the monitor-entity dataset show significant improvements in the hit-rate, mean reciprocal rank, and recall over baselines.

2 PRELIMINARIES

In this section, we introduce the problem of recommending dimensions for creating monitors for cloud services. We formally define the problem and provide preliminary notations.

2.1 PROBLEM FORMULATION

Ensuring continuous availability of services is essential for cloud service providers. Cloud services continuously record information about their health in the form of run-time telemetry. This telemetry

108 serves as signals to be analyzed for detecting anomalies. Therefore, cloud services are equipped with
 109 automated watchdogs, also known as monitors, that monitor service health. Each service emits multiple
 110 metrics (time-series data) along different dimensions, which are the topological components of
 111 the service. Examples of these dimensions include indicators such as the success of an operation, file
 112 path, environment of service deployment, and the identifier of a compute node. Monitors aggregate
 113 the metrics emitted along various dimensions, and the alerting conditions operate on the aggregated
 114 signal to create an alert. Based on the nature of the monitor, only a subset of the dimensions used
 115 for aggregating the signal.

116 Figure 1a provides an overview of the cloud monitoring framework, including the structural prop-
 117 erties of the monitor. These properties encompass the resource upon which the monitor is created
 118 (e.g., CPU), the metric being monitored (e.g., processor time), the time series data used to raise
 119 alerts, and dimensions such as region and environment of service deployment.

120 There are several approaches to predict metrics to be monitored for a given cloud service Srinivas
 121 et al. (2024). However, given the metric, m_i and the set of dimensions along which the metric is
 122 already being emitted, the problem of ranking dimensions along which the metric needs to be aggre-
 123 gated to raise an alert has not yet been explored. Furthermore, the monitor entity graph containing
 124 monitors, metrics, and dimensions is a heavy-tailed sparse graph, with limited interaction between
 125 most of the monitors, metrics, and dimensions. The sparsity is further exacerbated by the fact that
 126 most of the dimensions along which a metric is emitted are not used during monitor creation. As
 127 a result, our work focuses on developing a solution to the challenging problem of recommending
 128 dimensions for monitor creation in the context of this heavy-tailed sparse graph. We start by mod-
 129 eling the monitor entities and their various interactions, as well as their textual node attributes, as a
 130 heterogeneous multi-type interaction graph. From this graph, we generate effective node represen-
 131 tations, which are then used for dimension ranking. Next, we define the monitor entity graph and
 132 related attributes.
 133

Definition 1. (Monitor Entity Graph): We represent the data as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
 134 where $\mathcal{V} = \{\mathcal{V}_m \cup \mathcal{V}_d \cup \mathcal{V}_k\}$ represents the set of nodes with \mathcal{V}_m , \mathcal{V}_d , \mathcal{V}_k denoting monitors, dimen-
 135 sions, and metrics, respectively and $\mathcal{E} = \{\mathcal{E}_{md} \cup \mathcal{E}_{kd} \cup \mathcal{E}_{mk}\}$ represents the set of edges, capturing
 136 three types of relationships: 1) \mathcal{E}_{md} : “monitor associated with dimension”, 2) \mathcal{E}_{kd} : “metric has
 137 dimension”, and 3) \mathcal{E}_{mk} : “monitor emits metric”.

138 The graph structure \mathcal{G} captures the intricate relationships in the data that are crucial for making
 139 informed recommendations. By explicitly modeling different entity types and their relationships,
 140 we enable the model to learn domain-specific patterns. Each node in the graph, denoted as $v \in \mathcal{V}$,
 141 has a unique initial representation given by $\mathbf{x}_v \in \mathbb{R}^d$, where d is the dimension of the embedding
 142 space. The vector \mathbf{x}_v is the concatenation of two types of features: intrinsic features, which are
 143 domain-specific attributes of the entity (e.g., metric name, dimension name, monitor name, related
 144 service), and learned embeddings, which are trainable embeddings that capture the entity’s role in
 145 the graph structure (e.g., co-occurrence with another node). The monitor entity graph is shown in
 146 Figure 1b. It is to be noted that we assume a static graph.
 147

We define the problem of ranking dimensions for monitor creation in the monitor-entity graph, \mathcal{G}
 148 with different types of node features and link features as follows:

Problem (Dimension Ranking using Heterogeneous Interactions). Leverage entity features $\{\mathbf{x}_v : v \in \mathcal{V}\}$, link features $\{e_{v_1 v_2} \in \mathcal{E}\}$ and monitor entity graph, \mathcal{G} , to generate entity representations
 151 that facilitate relevant recommendations.

153 2.2 DEFINITION OF HETEROGENEOUS GRAPH AND NOTATIONS

155 Graph Neural Networks (GNNs) learn node representations by propagating features from local graph
 156 neighborhoods via trainable neighbor aggregators. In this context, we introduce the basic notations
 157 and formulations that are useful during message passing in GNN frameworks.
 158

159 GNNs use multiple layers to learn node representations. At each layer $l > 0$ (where $l = 0$ is the
 160 input layer), GNNs compute a representation for node v_1 by aggregating features from its neighbor-
 161 hood through a learnable aggregator $F_{\theta, l}$ per layer Hamilton et al. (2017); Kipf & Welling (2016);
 162 Velickovic et al. (2017). The embedding for node v_1 at the l -th layer is given by:

162

$$\mathbf{h}_{v_1,l} = F_{\theta,l}(\mathbf{h}_{v_1,l-1}, \{\mathbf{h}_{v_2,l-1}\}), v_2 \in N(v_1) \quad (1)$$

164

The embedding $\mathbf{h}_{v_1,l}$ at the l -th layer is a non-linear aggregation of its embedding $\mathbf{h}_{v_1,l-1}$ from layer $l - 1$ and the embedding of its immediate neighbors $v_1 \in N(v)$. The function $F_{\theta,l}$ defines the message-passing function at layer l and can be instantiated using a variety of aggregators Hamilton et al. (2017); Kipf & Welling (2016); Velickovic et al. (2017). The node representation for v_1 at the input layer is $\mathbf{h}_{v_1,0}$, where $\mathbf{h}_{v_1,0} = \mathbf{x}_v \in \mathbb{R}^D$.

170

3 ATTENTION ENHANCED ENTITY RANKING FOR SPARSE GNNs

172

In this section, we begin with the analysis of the monitor entity graph and discuss insights into the structure of the network, as well as the features that impact node relationships. We then use these insights to inform the framework design for dimension recommendations and ranking.

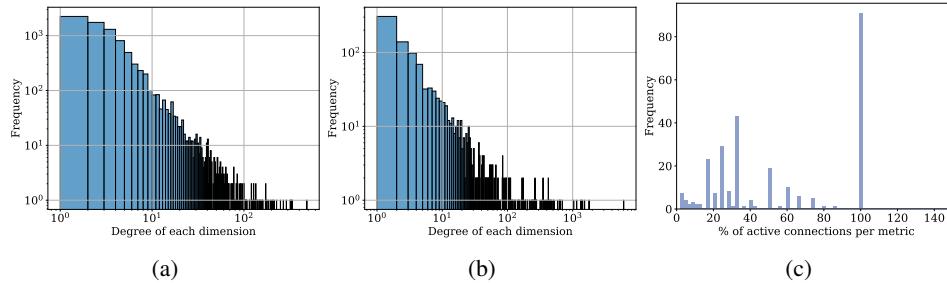
176

3.1 MONITOR ENTITY GRAPH: QUALITATIVE ANALYSIS

178

Figure 2 illustrates the characteristics of the monitor entity graph. Figure 2a shows the distribution of degree associated with dimensions based on the metric-to-dimension links (i.e., the number of metrics to which each dimension is connected). Similarly, Figure 2b shows the distribution of dimension degree based on the monitor-to-dimension interactions from the monitor entity graph. The likelihood test ratio indicates a resemblance to long-tailed distributions. Next, we analyze the distribution of the percentage of dimensions selected from the set of all dimensions along which the metric is emitted. As seen in Figure 2c, the distribution is skewed to the left, indicating that the majority of monitors do not need to aggregate the metrics along all dimensions along which they are emitted.

187



195

Figure 2: Characteristics of the Monitor Entity Graph: a) Distribution of degree associated with dimensions based on the metric-to-dimension links, b) Distribution of dimension degree based on the monitor-to-dimension interactions from the monitor entity graph, and c) Distribution of the percentage of dimensions selected from the set of all dimensions along which the metric is emitted.

201

Observation 1. The “monitor entity” graph faces activity sparsity. Although many dimensions are associated with metrics, only a subset of them is used to aggregate the metric while raising an alert.

205

Next, we analyze the features associated with the nodes to understand its impact on recommendations. We start with the text features associated with the monitor entity graph and its effect on dimensions associated with the monitors. Figure 3a shows the distribution of Jaccard similarity between sets of dimensions associated with monitors that exhibit high cosine similarity (> 0.8) between different text features. We consider the text similarity in the monitor names, metric names, and the service account associated with the monitor. The feature embeddings are generated using the “E5” embedding model, a general-purpose model trained through contrastive learning (Wang et al., 2022). The Jaccard similarity of dimensions from monitors exhibits different trends with respect to the similarity in the metric, monitor names, and that from the same service account. Furthermore, the similarity in dimensions with similar monitor names shows higher variance.

215

In addition, Figure 3b shows the density of pairwise correlation between dimensions connected to a monitor. The correlation plot shows two peaks signifying the presence of distinct groups with

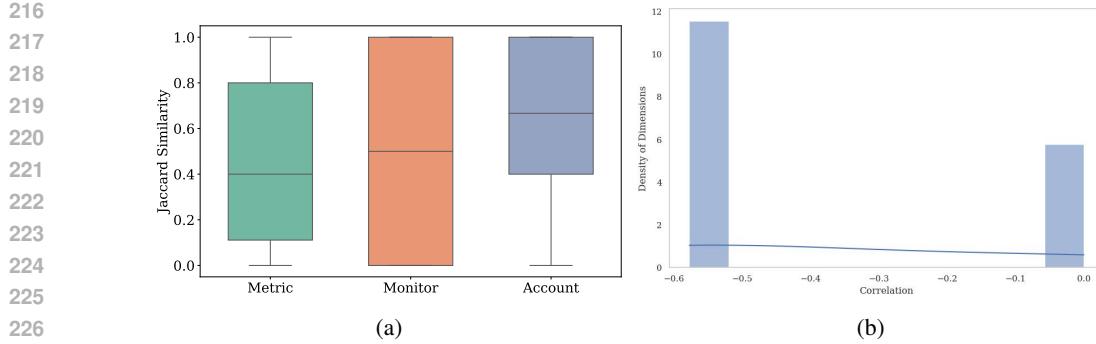


Figure 3: a) Variation in jaccard similarity of set of dimensions associated with monitors with similar metric, monitor names, and same service account, and b) Distribution of pairwise correlation between dimensions.

different trends. As seen in the figure, the majority of dimension pairs are negatively correlated, indicating the absence of a specific dimension in the presence of another.

Observation 2. We observe a significant overlap in the dimensions used by monitors associated with the same service account, those with similar metrics, and those with similar names. However, the extent of similarity varies across the features. Further, the correlation between dimension pairs shows the presence of two distinct groups, where some dimensions are negatively correlated, while the other group is not correlated. Specifically, the framework for representing entities of a node should consider the varying degrees of similarity across different features and the distinct correlation patterns among dimension pairs.

3.2 DiRaGNN FOR DIVERSE RANKING OF RECOMMENDATIONS

In this section, we present the framework to encode the multifaceted nature of cloud monitoring data and generate node representation for different entities. The graph representation, \mathcal{G} captures both the inherent properties of each entity and its context within the graph structure. The domain-specific attributes provide an inductive bias based on domain knowledge, while the learned embeddings allow the model to discover latent relationships and characteristics. The use of domain-specific attributes enable the model to generalize to new entities not seen during training which is useful in dynamic cloud environments. We begin by discussing the messaging passing mechanism.

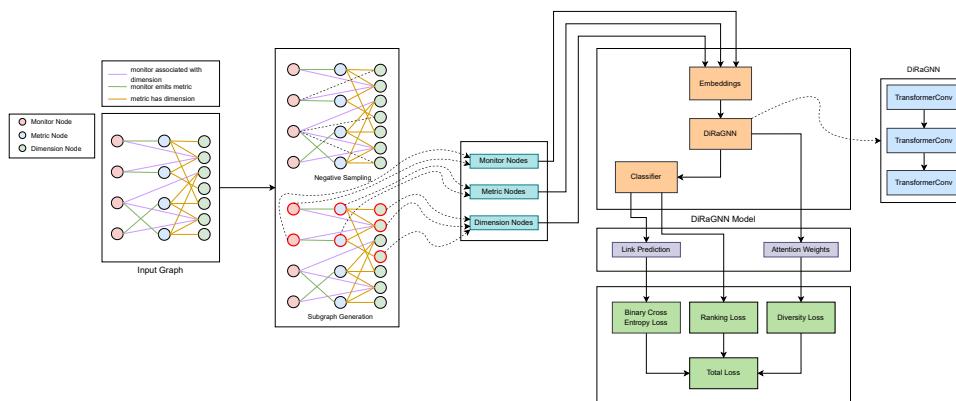


Figure 4: The overall architecture of DiRaGNN framework. The framework uses an enhanced transformer-style graph convolution that incorporates multi-head attention mechanisms, enabling the model to capture complex, long-range dependencies in the graph structure. Additionally, the custom loss function includes diversity loss, ranking loss, and binary cross-entropy loss.

270 3.2.1 MESSAGE PASSING MECHANISM
 271

272 The framework employs an effective message passing mechanism designed to capture complex re-
 273 lationships and contextual information. The multi-head attention enhanced message passing ap-
 274 proach leverages edge-specific transformations and heterogeneous neighborhood aggregation. The
 275 key components of our message passing mechanism are:

276 **Multi-Head Attention.** A multi-head attention mechanism to allow the model to focus on different
 277 aspects of node relationships simultaneously. This is important for distinguishing the relevance of
 278 different neighbors and relationship types in our graph. The attention weights α are computed as
 279 follows: $\alpha_{ij} = \frac{(q_i \cdot k_j)}{\sqrt{d_h d_o}}$, where q_i is the query vector for node i , k_j is the key vector for node j , d_h is
 280 the number of attention heads, and d_o is the output dimension per head. The attention weights are
 281 then normalized using softmax: $\alpha_{ij} = \frac{\exp(\alpha_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(\alpha_{ik})}$ where $\mathcal{N}(i)$ is the set of neighbors of node
 282 i .
 283

284 **Edge-Aware Message Transformation.** The message passing considers the type of relationship
 285 between the nodes. The transformed message incorporates both the node features and the computed
 286 attention weights, ensuring that the message is sensitive to the specific relationship type. The trans-
 287 formed message m_{ij} from node j to node i is computed as $m_{ij} = \alpha_{ij} v_j$, where v_j is the value
 288 vector of node j . The final aggregated message for node i is $m_i = \sum_{j \in \mathcal{N}(i)} m_{ij}$. The node features
 289 are updated as: $x_i^{(l+1)} = \sigma(W^{(l)} \cdot \text{CONCAT}(x_i^{(l)}, m_i^{(l)}))$, where $x_i^{(l)}$ is the feature vector of node
 290 i at layer l , $W^{(l)}$ is the learnable weight matrix for layer l , σ is the ReLU activation function and
 291 CONCAT is the concatenation operation.
 292

293 By using attention mechanisms, our approach can dynamically assign importance to different types
 294 of relationships and neighbors, important for our problem. The multi-head attention allows the
 295 model to leverage limited interactions more effectively, mitigating the impact of sparse data. After
 296 multiple rounds of message passing, each monitor node has an enriched representation incorporating
 297 information from relevant dimensions and metrics. These representations are used to score potential
 298 monitor-dimension pairs.

299 3.2.2 TRAINING OBJECTIVE
 300

301 In recommendations systems, the choice of training objective directly influences the model’s ability
 302 to make accurate, well-ranked, and diverse recommendations. We define a composite loss function
 303 that addresses multiple aspects of the recommendation task simultaneously. Our loss function com-
 304 bines three key components: 1) Binary Cross-Entropy Loss, 2) TOP1-max Ranking Loss, and 3)
 305 Diversity Loss.

306 **Binary Cross-Entropy Loss:** The binary cross-entropy (BCE) loss is fundamental for our basic
 307 prediction accuracy which is defined as:
 308

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (2)$$

311 where y_i is the true label and \hat{y}_i is the predicted probability. The BCE loss ensures basic prediction
 312 accuracy.
 313

314 **TOP1-max Ranking Loss:** To optimize the order of recommendations, we use the TOP1-max
 315 ranking loss proposed in Hidasi & Karatzoglou (2018). This loss aims to push the target score above
 316 the scores of negative samples while also acting as a regularizer. It’s defined as:

$$\mathcal{L}_{\text{TOP1-max}} = \sum_{j=1}^N s_j [\sigma(r_j - r_i) + \sigma(r_j^2)] \quad (3)$$

320 where r_i is the score for the positive sample, r_j are scores for negative samples, $s_j = \text{softmax}(r_j)$,
 321 and σ is the sigmoid function. The first term of the TOP1-max ranking loss encourages the positive
 322 score to be higher than the maximum negative score and the second term acts as a regularizer,
 323 pushing negative scores towards zero. This loss optimizes the order of recommendations, crucial for
 top-k recommendation scenarios.

Table 1: Dataset used in this work

Dataset	Node	Edge
Cloud Monitoring System	# monitors: 18291 # metrics: 4623 # dimensions: 8356	# monitor, associated_with, dimension: 52148 # metric, has, dimension: 109213 # monitor, emits, metric: 52148

Diversity Loss: To encourage the model to capture diverse aspects of the graph structure, we introduce a diversity loss component. The diversity loss \mathcal{L}_{div} is computed as:

$$\mathcal{L}_{\text{div}} = \lambda_{\text{div}} \cdot \frac{1}{L} \sum_{l=1}^L \text{MSE}(\alpha_l, \bar{\alpha}_l) \quad (4)$$

where α_l are the attention weights in layer l , $\bar{\alpha}_l$ is their mean, and λ_{div} is the diversity strength. This diversity loss penalizes similar attention patterns across different heads, promoting a more comprehensive representation of the graph structure. It prevents the model from over-focusing on a few popular items, particularly important in sparse data settings.

Our final loss function combines all the above components.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{div}} + \mathcal{L}_{\text{TOP1}} \quad (5)$$

Dynamic Loss Balancing To ensure optimal contribution from each loss component, we use a dynamic loss balancing mechanism which adjusts the weights of different loss components during training, allowing for adaptive optimization of our multi-objective function.

Neighbourhood Sampling and Subgraph Generation. In large-scale heterogeneous graphs, processing the entire graph for each recommendation task can be computationally expensive. Moreover, distant nodes in the graph may introduce noise rather than providing useful information. We address these challenges by focusing on the most relevant parts of the graph and employ a multi-stage approach to neighborhood sampling and subgraph generation:

1. We utilize a carefully designed edge splitting strategy for training, validation, and testing, balancing between information propagation and model supervision. During training, negative edges are generated on-the-fly which helps in efficient learning of edge distinction.
2. To capture relevant neighborhood context, we utilize a multi-hop sub-graph sampling method. This approach allows the model to consider both immediate and more distant relationships, crucial for understanding the complex interactions in cloud monitoring systems.

The multi-hop sampling strategy, combined with dynamic negative sampling during training, allows the model to explore a broader range of graph structures while focusing on the most informative negative examples. This also enables the model to effectively learn from sparse interaction graphs, capturing complex relationship between monitors, dimensions, and metrics, while maintaining computational feasibility. Figure 4 summarizes the proposed framework.

4 EXPERIMENTS

To evaluate the effectiveness of our proposed framework, we conducted a series of experiments on the task of dimension recommendation for monitors in cloud environments. This section details our experimental setup, evaluation metrics, baseline comparisons, and results analysis.

4.1 EXPERIMENT SETUP

4.1.1 DATASETS

Our experiments utilize a heterogeneous graph dataset representing a complex cloud monitoring system. The dataset comprises of three types of entities (nodes) and three types of relationships (edges).

378 It captures the intricate interactions between monitors, dimensions, and metrics in a cloud environment.
 379 The main statistics of the dataset are summarized in Table 1. This graph structure effectively
 380 represents the complex relationships in cloud monitoring systems, where monitors are associated
 381 with specific dimensions and emit various metrics, while metrics themselves are characterized by
 382 multiple dimensions.

383 **Feature Representation:** To capture the semantic information of monitors and dimensions, we
 384 employ feature embeddings generated using a state-of-the-art language model: 1) Monitor Features:
 385 Represented by the embeddings of the metric names associated with each monitor. (each monitor is
 386 associated with a single metric) and 2) Dimension Features: Represented by the embedding of the
 387 dimensions names.

388 Both feature embeddings are generated using the “E5” embedding model, a general-purpose model
 389 trained through contrastive learning (Wang et al., 2022). This approach allows us to capture rich
 390 semantic information from the textual descriptions of metrics and dimensions, enabling our model to
 391 understand and utilize the contextual relationships between different entities in the cloud monitoring
 392 system.

394 4.1.2 TRAINING DETAILS

395 We train DiRaGNN using $L = 3$ message passing layers with hidden channels size of 256 and output
 396 channels size of 128. We use the Adam optimizer with a learning rate of 0.001 and weight decay
 397 of 10^{-5} . To adapt the learning rate during training, we employed a learning rate scheduler which
 398 reduces the learning rate by half if the validation loss doesn’t improve for 5 consecutive epochs.
 399 Our training ran for a maximum of 100 epochs, with early stopping implemented to prevent over-
 400 fitting. We used a patience of 10 epochs for early stopping. The edge set was divided into training
 401 (80%), validation (10%), and test (10%) sets. Within the training set, 70% of edges were used for
 402 message passing, and 30% for supervision. For evaluation, we generated fixed negative edges with
 403 a ratio of 2 : 1 (negative to positive). During training, negative edges were dynamically generated
 404 using on-the-fly negative sampling. We sampled multiple hops from both ends of a link to create
 405 subgraphs. We used a negative sampling ratio of 2.0 during training. We employed a batch size of
 406 128 for training.

407 4.1.3 EVALUATION METRICS

408 We used the following metrics to evaluate the performance of our model and baselines:

- 411 1. **Hit Ratio (HR@k):** Measures the percentage of test cases where the correct dimension is in
 412 the top k recommendations.
- 413 2. **Mean Reciprocal Rank (MRR):** The average of the reciprocal rank of the first correct dimen-
 414 sion in the recommendations.
- 415 3. **Normalized Discounted Cumulative Gain (NDCG@k):** Measures the ranking quality of the
 416 recommendations, with k set to the number of true dimensions for each monitor.
- 417 4. **Recall@k:** The proportion of true dimensions that are successfully retrieved in the top k rec-
 418 ommendations.

420 4.2 BASELINES

421 We compared our model against the following baselines:

- 424 • **SAGEConv + Mean:** GraphSAGE Hamilton et al. (2017) with mean aggregation. Element-
 425 wise mean pooling for neighbor aggregation and self-embedding concatenation at each layer.
- 426 • **SAGEConv + Max:** GraphSAGE with max pooling aggregation. Element-wise max pooling
 427 for neighbor aggregation and self-embedding concatenation at each layer.
- 428 • **TranformerConv:** The graph transformer operator Shi et al. (2020). A multi-layer Graph
 429 Transformer network takes the input to perform attentive information propagation between
 430 nodes. For message aggregation, the framework concatenates information across all the heads.

431 Our proposed model was evaluated in two configurations:

432
 433 Table 2: Proposed framework outperforms SAGEConv using mean and max aggregation and Trans-
 434 formerConv. The proposed framework demonstrates relative gains of 49.6%, 60.1%, and 29.06% in
 435 Hit-Rate@1, NDCG@k, and Recall@5 respectively with respect to the best baseline.

Metric	HR@1	HR@3	HR@5	MRR	N@k	R@1	R@3	R@5
SAGEConv + Mean	0.383	0.186	0.127	0.499	0.328	0.218	0.379	0.474
SAGEConv + Max	0.291	0.154	0.111	0.414	0.262	0.165	0.323	0.398
TransformerConv	0.331	0.188	0.134	0.481	0.306	0.178	0.399	0.523
TransformerConv + Diversity Loss	0.547	0.238	0.157	0.650	0.500	0.21	0.561	0.655
DiRaGNN	0.573	0.246	0.159	0.672	0.525	0.342	0.592	0.675

- **Diversity Loss:** Our model featuring a custom TransformerConv with multi-head attention and integrated layer-wise diversity loss.
- **DiRaGNN: Diverity Loss + Ranking Loss:** Our complete model featuring a custom TransformerConv with integrated layer-wise diversity loss and ranking optimization.

451 4.3 RESULTS

452 Both variants of our proposed model significantly outperform the baselines across all metrics. The
 453 full model with both diversity and ranking losses shows the best overall performance. Our full
 454 model achieves HR@1 of 0.573, which is a 49.6% improvement over the best baseline (SAGEConv
 455 + Mean at 0.383), indicates a substantial enhancement in the ability to recommend the most relevant
 456 dimension as the top choice. The NDCG@k score of 0.525 for our full model, compared to 0.328
 457 for the best baseline, represents a 60.1% improvement. The results suggests that our model not
 458 only recommends relevant dimensions but also ranks them more effectively. Our model shows
 459 significant improvements in Recall@k, particularly at higher k values. The Recall@5 of 0.675
 460 for our full model compared to 0.523 for TransformerConv indicates a 29.06% improvement in
 461 retrieving relevant dimensions within the top 5 recommendations. The introduction of diversity
 462 loss alone leads to substantial improvements across all metrics compared to the baselines. This
 463 underscores the importance of encouraging diverse attention patterns in the model. The addition
 464 of ranking loss to diversity loss results in further improvements, particularly in HR@1 and MRR.
 465 This highlights the effectiveness of our multi-faceted loss function in optimizing both accuracy and
 466 ranking quality.

467 Next, we discuss the intuition behind the improved performance of the proposed framework;

- 468 1. DiRaGNN uses a multi-head attention mechanism, which allows it to capture the relative im-
 469 portance of different neighbor types more effectively than “SAGEConv”, which relies on mean
 470 or max aggregation. On the other hand, while “TransformerConv” uses attention, it may strug-
 471 gle to distinguish between different types of relationships as effectively as DiRaGNN.
- 472 2. The diversity loss in framework encourages it to capture varied aspects of the data, whereas the
 473 baselines do not have an explicit mechanism to encourage diversity in their representations.
- 474 3. The ranking loss from DiRaGNN directly optimizes for ranking quality. In contrast, the base-
 475 lines are typically trained with binary classification objectives, which may not directly optimize
 476 for ranking quality.
- 477 4. Baselines may struggle with sparsity more than the proposed framework, as they rely more
 478 heavily on dense connection patterns.

482 4.4 ABLATION STUDY

483 To understand the individual contributions of our model’s key components, we conducted an ablation
 484 study focusing on the diversity loss and ranking loss. We compared three model variants: 1) Base
 485 model (without diversity and ranking loss), 2) Model with only ranking loss, and 3) Model with

only diversity loss. We aggregate the changes in the ranks of the recommended dimensions across different monitors and the performance of these variants using rank stability plots, which visualize change in the relevance of top ranked dimensions across different model configurations.

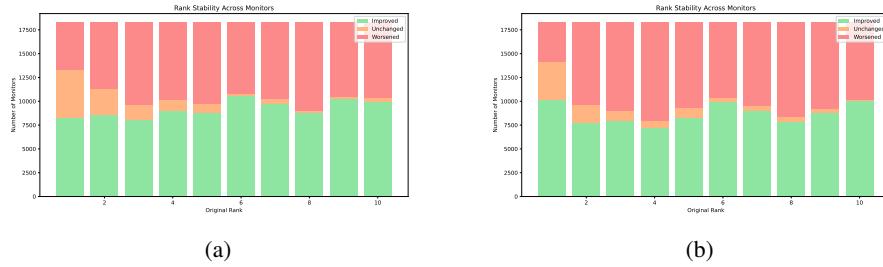


Figure 5: Impact of Ranking and Diversity Loss on Model Performance: a) Impact of Ranking Loss on Model Performance, and b) Impact of Diversity Loss on Model Performance. Overall quality of recommended dimensions improved with the addition of the ranking loss and diversity loss.

Impact of Ranking Loss. Figure 5a illustrates the effect of incorporating only the ranking loss into our base model. The results show a significant improvement in ranking performance across all positions: 1) Top-ranked items (Rank 1) demonstrate high stability, with a large proportion remaining unchanged, 2) Lower-ranked items (Ranks 2-10) show substantial improvements, with the proportion of improved rankings increasing for initially lower-ranked items, and 3) Very few rankings worsen, especially in the lower ranks, indicating that the ranking loss effectively optimizes the order of recommendations. The ranking loss appears to be particularly effective at improving the position of dimensions initially ranked lower (Ranks 8-10), suggesting that it helps in surfacing relevant but previously undervalued dimensions.

Impact of Diversity Loss. Figure 5b illustrates the effect of incorporating only the diversity loss into our base model. While also improving overall performance, it exhibits a different pattern compared to the ranking loss: 1) As compared to the impact of the ranking loss, improvements are more uniformly distributed, indicating a more uniform impact on the entire ranking, and 2) There is slightly less stability for top-ranked items compared to the ranking loss model.

The diversity loss appears to encourage a broader exploration of the dimensional space, promoting a wider range of dimensions across different rank positions. Both losses contribute positively to the model's performance, but with distinct characteristics. The ranking loss is crucial for optimizing the order of recommendations, particularly beneficial for surfacing relevant dimensions from lower ranks. The diversity loss contributes to a more balanced improvement across all ranks, likely enhancing the model's ability to recommend a varied set of relevant items. Both losses combined provide a synergistic effect: the ranking loss optimizes the overall order, while the diversity loss ensures a broader range of relevant dimensions to be considered.

These findings support our decision to incorporate both ranking and diversity losses in the final model. The ranking loss ensures high-quality ordered recommendations, while the diversity loss helps prevent over-focusing on a narrow set of popular items - a crucial factor in sparse interaction graphs typical in cloud monitoring systems.

5 CONCLUSION

We propose a novel recommendation framework, DiRaGNN, which leverages the available set of node attributes and multi-type interactions to overcome the challenges of structural and interaction sparsity in the monitor entity graph. DiRaGNN incorporates a diversity-aware loss function, edge-aware message passing, and multi-head attention mechanisms inspired by transformer architectures. Experiments on the monitor-entity dataset demonstrate significant improvements in hit-rate, mean reciprocal rank, and recall over baseline approaches. The proposed framework presents a promising approach for addressing the recommendation problem in cloud settings with sparse and diverse interaction data.

540 REFERENCES
541

- 542 Fedor Borisuk, Shihai He, Yunbo Ouyang, Morteza Ramezani, Peng Du, Xiaochen Hou, Cheng-
543 ming Jiang, Nitin Pasumarthy, Priya Bannur, Birjodh Tiwana, et al. Lignn: Graph neural networks
544 at linkedin. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and*
545 *Data Mining*, pp. 4793–4803, 2024.
- 546 Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang,
547 Jeffrey Sun, Zhangwei Xu, et al. Towards intelligent incident management: why we need it
548 and how we make it. In *Proceedings of the 28th ACM Joint Meeting on European Software*
549 *Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1487–
550 1497, 2020.
- 551 Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-gcn:
552 An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings*
553 *of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp.
554 257–266, 2019.
- 555 Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural
556 network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, pp.
557 2331–2341, 2020.
- 559 Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs.
560 *Advances in neural information processing systems*, 30, 2017.
- 562 Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-
563 based recommendations. In *Proceedings of the 27th ACM international conference on information*
564 *and knowledge management*, pp. 843–852, 2018.
- 565 Huiting Hong, Hantao Guo, Yucheng Lin, Xiaoqing Yang, Zang Li, and Jieping Ye. An attention-
566 based graph neural network for heterogeneous structural learning. In *Proceedings of the AAAI*
567 *conference on artificial intelligence*, volume 34, pp. 4132–4139, 2020.
- 569 Binbin Hu, Zhiqiang Zhang, Chuan Shi, Jun Zhou, Xiaolong Li, and Yuan Qi. Cash-out user detec-
570 tion based on attributed heterogeneous information network with a hierarchical attention mecha-
571 nism. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 946–953,
572 2019.
- 573 Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In
574 *Proceedings of the web conference 2020*, pp. 2704–2710, 2020.
- 576 Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive sampling towards fast graph
577 representation learning. *Advances in neural information processing systems*, 31, 2018.
- 578 Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional net-
579 works. *arXiv preprint arXiv:1609.02907*, 2016.
- 581 Jesús Montes, Alberto Sánchez, Benjamin Memishi, María S Pérez, and Gabriel Antoniu. Gmone:
582 A complete approach to cloud monitoring. *Future Generation Computer Systems*, 29(8):2026–
583 2040, 2013.
- 584 Vinod Nair, Ameya Raul, Shwetabh Khanduja, Vikas Bahirwani, Qihong Shao, Sundararajan Sel-
585 lamanickam, Sathiya Keerthi, Steve Herbert, and Sudheer Dhulipalla. Learning a hierarchical
586 monitoring system for detecting and diagnosing service issues. In *Proceedings of the 21th ACM*
587 *SIGKDD international conference on knowledge discovery and data mining*, pp. 2029–2038,
588 2015.
- 589 Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. Graph neural networks for friend ranking in
590 large-scale social platforms. In *Proceedings of the Web Conference 2021*, pp. 2535–2546, 2021.
- 591 Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label
592 prediction: Unified message passing model for semi-supervised classification. *arXiv preprint*
593 *arXiv:2009.03509*, 2020.

- 594 Pooja Srinivas, Fiza Husain, Anjaly Parayil, Ayush Choure, Chetan Bansal, and Saravan Rajmohan.
 595 Intelligent monitoring framework for cloud services: A data-driven approach. In *Proceedings of*
 596 *the 46th International Conference on Software Engineering: Software Engineering in Practice*,
 597 pp. 381–391, 2024.
- 598 Chellammal Surianarayanan and Pethuru Raj Chelliah. Essentials of cloud computing. *Cham: Springer International Publishing*, 2019.
- 601 Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Ben-
 602 gio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.
- 603 Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Dixin Jiang, Rangan Ma-
 604 jumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *arXiv*
 605 *preprint arXiv:2212.03533*, 2022.
- 606 Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous
 607 graph attention network. In *The world wide web conference*, pp. 2022–2032, 2019.
- 608 Carl Yang, Aditya Pal, Andrew Zhai, Nikil Pancha, Jiawei Han, Charles Rosenberg, and Jure
 609 Leskovec. Multisage: Empowering gcn with contextualized multi-embeddings on web-scale
 610 multipartite networks. In *Proceedings of the 26th ACM SIGKDD international conference on*
 611 *knowledge discovery & data mining*, pp. 2434–2443, 2020.
- 612 Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec.
 613 Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the*
 614 *24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 974–
 615 983, 2018.
- 616 Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph trans-
 617 former networks. *Advances in neural information processing systems*, 32, 2019.
- 618 Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-
 619 saint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.
- 620 Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V Chawla. Heteroge-
 621 neous graph neural network. In *Proceedings of the 25th ACM SIGKDD international conference*
 622 *on knowledge discovery & data mining*, pp. 793–803, 2019.
- 623 Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning
 624 architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelli-
 625 gence*, volume 32, 2018.
- 626 Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. Heterogeneous
 627 graph structure learning for graph neural networks. In *Proceedings of the AAAI conference on*
 628 *artificial intelligence*, volume 35, pp. 4697–4705, 2021.
- 629
 630
 631
 632
 633
 634
 635
 636
 637
 638
 639
 640
 641
 642
 643
 644
 645
 646
 647