MULTI-VECTOR EMBEDDING ON NETWORKS WITH TAXONOMIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Networks serve as efficient tools to describe close relationships among nodes. Taxonomies consist of labels organized into hierarchical structures and are often employed to describe rich attributes of the network nodes. Existing methods that co-embed nodes and labels in a low-dimensional space all encounter an obstacle called under-fitting, which occurs when the vector of a node is obliged to fit all its labels and neighbor nodes. In this paper, we propose HIerarchical Multi-vector Embedding (HIME), which allows multiple vectors of a node to fit different sets of its labels in a Poincaré ball, where the label hierarchy is well preserved. Experiments show that HIME has comprehensive advantages over existing network embedding methods in preserving both node-node and node-label relationships.

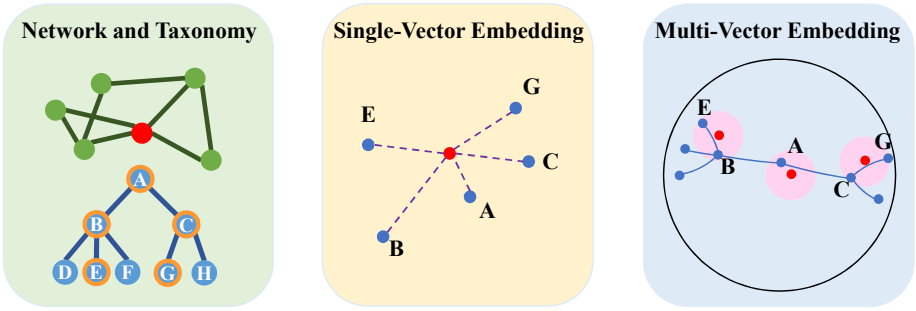
1 INTRODUCTION

A network is usually applied to depict the proximities among nodes, with extra node properties described by labels organized as a taxonomy. For instance, in a co-authorship network with a research taxonomy, a network edge between two researchers reveals their close academic relationship, while the hierarchical labels 'artificial intelligence', 'natural language processing' and 'sentiment analysis' formed into a label-path in the taxonomy reveal a researcher's areas of interest. Generally, a node can have multiple label-paths in the taxonomy telling different properties, as shown in figure 1 (a).

Heterogeneous network embedding serves as an option for co-embedding nodes and labels. It learns a low-dimensional vector for each entity, so that the distances among the embedding vectors indicate the topographic distances among entities in the network with the taxonomy. However, the single embedding vector of a node can be overloaded to preserve both the node-node and the node-label relationships, which causes the under-fitting problem Yang et al. (2020). As illustrated in figure 1 (b), single-vector embedding methods will possibly embed a node in the middle of all its labels in order to minimize the overall node-label distances, resulting in the node not being close to any of its labels. The under-fitting problem seriously worsens the performance on the distance-related tasks, such as label retrieval for a certain node according to the node-label distances.

To solve the under-fitting problems, one way is to reform the embedding position of the labels so as to minimize the chances of under-fitting, which can be achieved by using hyperbolic spaces. A hyperbolic space is a non-Euclidean space, and it grows exponentially with the increase of the space radius. It can be viewed as a continuous version of trees, which naturally caters to the hierarchical label structure Nickel & Kiela (2017). Labels in a label-path are embedded in the same direction in the hyperbolic space, so that a node vector is less likely to under-fit the labels belonging to a single label-path. But simply using hyperbolic embedding technique does not necessarily tackle the under-fitting problem, since a node can have multiple label-paths in different directions. However, by allocating multiple vectors to a node, with each fitting a small set of labels as shown in figure 1 (c), the under-fitting problem can be well resolved.

Therefore, we propose HIerarchical Multi-vector Embedding (HIME), which co-embeds network nodes and taxonomy labels in a hyperbolic space. HIME gives each label a single embedding vector to preserve the hierarchical structure, while a node has multiple vectors, with one specialized in maintaining the node-node proximity, and the others overcoming the under-fitting problem in preserving the node-label relationships.



(a) (b) (c)

Figure 1: The difference between single-vector embedding and multi-vector embedding. Given a network node with five circled labels in a taxonomy (a), single-vector embedding learns a vector in the middle of all the five labels, which leads to under-fitting (b). However, by learning multiple vectors for a node, the under-fitting problem can be overcome in a Poincaré disk (c).

Our work makes the following contributions: (1) We solve the under-fitting problem by allowing multiple embedding vectors of a node to fit different sets of labels. (2) A Least Recently Used (LRU) based algorithm balances the loads on the multiple vectors of a node. (3) Hyperbolic spaces are applied to co-embed network nodes and hierarchical labels more effectively. (4) Experiments show that HIME has better capability of preserving both node-node and node-label relationships compared with existing network embedding methods.

2 Related Work

In this section, we first introduce some classic approaches of network embedding, followed by the taxonomy-related embedding methods most relevant to our background. Hyperbolic embedding methods will then be presented. Finally we will introduce the concept of multi-aspect embedding.

Network Embedding. Approaches of network embedding can be mainly categorized into matrix factorization, Skip-Gram with walk patterns Mikolov et al. (2013) and neural networks from the perspective of methodology. Algorithms based on matrix factorization Belkin & Niyogi (2003); Cao et al. (2015); Ou et al. (2016) obtain embedding vectors by factorizing either Laplacian eigenmaps or the node proximity matrix. Methods using Skip-Gram model with walk patterns are prevailed in both plain network embedding Perozzi et al. (2014); Grover & Leskovec (2016); Tang et al. (2015b), and heterogeneous network embedding Dong et al. (2017); Tang et al. (2015a); Hussein et al. (2018). The neural network based methods Kipf & Welling (2016); Hamilton et al. (2017); Wang et al. (2018; 2016); Cao et al. (2016); Fu et al. (2017) generate the embedding vectors by ways such as graph convolution and adversarial training Goodfellow et al. (2014). Though not customized for networks with taxonomies, these classic methods laid the foundation for later embedding methods varying for different tasks or data.

Taxonomy-Related Embedding. Several embedding methods on networks with taxonomies have been proposed in recent years. Onto2Vec Smaili et al. (2018) first learns the embedding vectors of the biomedical taxonomy, and then generates the vectors for biological entities based on the learned taxonomy vectors. The drawback of Onto2Vec is that it ignores the label co-occurrences when learning the vectors for labels. Tag2Vec Wang et al. (2019) applies meta-path patterns and co-embeds nodes and hierarchical labels, while it still suffers from the under-fitting problem when a node has multiple label-paths. Given a label in the taxonomy, TaxoGAN Yang et al. (2020) spares an individual embedding space for it, where its nodes and its child labels are embedded by adversarial training. Although TaxoGAN well preserves the node-label proximity in small sub-graphs, it fails to maintain the node-label relationships globally. The key issue of embedding lies in using globally functional vectors to overcome the under-fitting problem, which has not been well addressed by existing taxonomy-related methods.

Hyperbolic Embedding. Previous researches have shown that preserving hierarchical structures in Euclidean space requires large embedding dimension Nickel et al. (2014), while it can be easily achieved by using low-dimensional hyperbolic spaces Gromov (1987); Boguná et al. (2010). HGCN Chami et al. (2019) embeds nodes by using graph convolution operation defined in hyperbolic spaces. Poincaré Nickel & Kiela (2017) embeds words into a Poincaré ball so as to maintain the hypernym relationships, while HMLC Chen et al. (2020) performs hierarchical multi-label text classification by co-embedding words and labels into a hyperbolic space. The success of single-vector hyperbolic embedding on words mainly attributes to the fact that most words are univocal in the hierarchy. However, these methods are not capable of representing nodes with multiple label-paths.

Multi-Aspect Embedding. Contrast to existing single-vector embedding methods, several recent works have revealed the necessity of learning multiple vectors for each node in different aspects. PolyDW Liu et al. (2019) represents each facet of an item using an embedding vector; Splitter Epasto & Perozzi (2019) allows multiple node embedding vectors to encode different communities; Asp2Vec Park et al. (2020) employs random walks to dynamically assign aspects to each node according to its local context; JOIE Hao et al. (2019) presents a two-view embedding model for a knowledge graph. Though innovative, most multi-aspect methods preserve relationships among nodes in different aspects, while neglecting either node-aspect proximity or hierarchical structures. Differently, HIME preserve both node-node and node-label relationships under the label hierarchy.

3 PRELIMINARIES

In this part, we will first formally define our problem, and introduce the notations used throughout the paper. Then we will briefly describe the Poincaré ball where nodes and labels are embedded.

3.1 PROBLEM STATEMENT

HIME takes the following data as inputs:

- a network $N = \{V, E\}$, where $V = \{v_i\}, i = 1, 2, .., n$ is the set of nodes, and $E = \{e_{ij}\}, i, j = 1, 2, .., n, i \neq j$ is the set of edges;
- a label taxonomy $T = \{L, S\}$, where $L = \{l_i\}, i = 1, 2, ..., m$ is the set of labels, and $S = \{s_{ij}\}, i, j = 1, 2, ..., m, i \neq j$ is the set of parent-child edges;
- a label assignment $A = \{A_i\}, i = 1, 2, ..., n$, where A_i is the set of labels that node v_i has. Once a label appears in A_i, all its ancestors in the taxonomy will also appear in A_i.

Given the inputs, HIME learns a single vector for each label and multiple vectors for each node. Specifically, every node has a single 'root vector' to preserve node proximity, and several 'branch vectors' to fit its labels. Formally, given dimension d and branch vector number k, HIME produces:

- The root vectors $R^{n\times d}$ of the nodes, with R_i referring to the root vector of v_i;
- The branch vectors $B^{n\times k\times d}$ of the nodes, with $B_{ij}, i = 1, 2, .., n, j = 1, 2, ..., k$ referring to the j-th branch vector of v_i;
- The label vectors $Q^{m\times d}$, with Q_i referring to the representation vector of l_i.

In this way, R preserves the node proximity, Q preserves the hierarchical taxonomy, while B and Q together preserve the node-label relationships. All vectors function in a Poincaré Ball as follows.

3.2 THE POINCARÉ BALL MODEL

Let $\|\cdot\|$ denotes the Euclidean norm. A Poincaré ball with dimension d and radius 1 can be defined as: $\mathbb{D}^{d,1} := \{x \in \mathbb{R}^d : \|x\|^2 < 1\}, g_x = \lambda_x^2 I_d$, where g_x is the Riemannian metric tensor, $\lambda_x = \frac{2}{1-\|x\|^2}$ and I_d is the identity matrix. The distance between two points $x, y \in \mathbb{D}^{d,1}$ can be computed as:

$$\text{dist}(x, y) = \text{arcosh}\left(1 + 2\frac{\|x - y\|^2}{(1 - \|x\|^2)(1 - \|y\|^2)}\right). \tag{1}$$

Though having limited radius, the Poincaré ball has infinite space volume. A hierarchical structure can be embedded in the ball by placing the root near the center while leaves near the open surface.

4 METHOD

In this section, we will first introduce multi-vector embedding used for preserving the node-label relationships. Then a load balancing strategy that optimizes multi-vector embedding will be explained. Finally we will detail the whole learning procedure and discuss the complexities of HIME.

4.1 MULTI-VECTOR EMBEDDING

Traditional single-vector embedding methods generate a single embedding vector for each node. When a node has numerous labels, the single vector is often overloaded to preserve all the node-label relationships, and therefore stuck in an under-fitting position. However, by giving multiple branch vectors to a node, the under-fitting problem will be significantly alleviated.

To implement multi-vector embedding, we should first define the distance $Dist_{nl}(\cdot,\cdot)$ between a node and a label. Since the label will be fit by the node if only one of the node's branch vectors is close to it, we define the node-label distance as the shortest distance from all the node's branch vectors to the label vector. With the distance function $\mathrm{dist}(\cdot,\cdot)$ defined by equation 1, formally,

$$Dist_{nl}(v_i, l_j) = \min_{a=1,2,...k} \mathrm{dist}(B_{ia}, Q_j). \tag{2}$$

We optimize the branch vectors and the label vectors by Skip-Gram with negative sampling Mikolov et al. (2013). Specifically, we view all the ground-truth node-label pairs as the positive set R. Given the negative sampling number p, for every node v_i, we sample $p|A_i|$ labels not in set A_i, and form the negative node-label pairs with v_i. We gather all the negative node-label pairs as a set R_{neg}. Given the node-label distance defined by equation 2, the loss function of node-label relationships can be written as:

$$Loss_{nl} = -\sum_{(v_i,l_j)\in R} \log\sigma(-Dist_{nl}(v_i, l_j)) - \sum_{(v_i,l_j)\in R_{neg}} \log\sigma(Dist_{nl}(v_i, l_j)). \tag{3}$$

where $\sigma(x) = (\frac{1}{1+e^{-x}})$. We first randomly initialize the vectors of B and Q near the origin of the Poincaré ball. By minimizing $Loss_{nl}$, the negative samples serve as a strong force to push all the vectors away from the origin, while the positive samples decrease the distances of the ground-truth node-label pairs. Finally, the node-label distances of the positive node-label pairs will become relatively smaller, while those of the negative ones will become larger.

Here we show how B and Q are optimized by minimizing $Loss_{nl}$. Given a node v_i and a label l_j, $Loss_{nl}$ first computes their node-label distance by equation 2. Suppose B_{i1} is the branch vector closest to Q_j among all v_i's branch vectors. In this way, $Dist_{nl}(v_i, l_j) = \mathrm{dist}(B_{i1}, Q_j)$. Therefore, during the back propagation process, only Q_j and B_{i1} are updated by pair (v_i, l_j), with the gradients of v_i's other branch vectors being unchanged. Particularly, when (v_i, l_j) serves as a positive sample, the label l_j will be assigned to the closest branch vector of v_i, and both the label vector and the branch vector will be updated closer. The branch vectors and label vectors can be compared to centroids and points in K-means clustering algorithm, where a point is always assigned to its closest centroid, while a centroid fits a cluster of points.

4.2 LOAD BALANCING STRATEGY

Here, we will first discuss the problem of inactive branch vectors that penalizes multi-vector embedding, then an LRU-based policy will be presented to settle this problem.

4.2.1 THE INACTIVE BRANCH VECTORS

As mentioned above, given a positive node-label sample $(v_i, l_j), l_j \in A_i$, l_j is always assigned to the branch vector closest to it during the optimization. We call a node's branch vector 'active' if at least one of the node's labels is assigned to it. Relatively, an 'inactive' branch vector means that none of the node's labels is assigned to it, also indicating that it fits no labels. In order to balance the loads of fitting labels on all branch vectors of a node, we aim at maximizing the number of the active branch vectors.

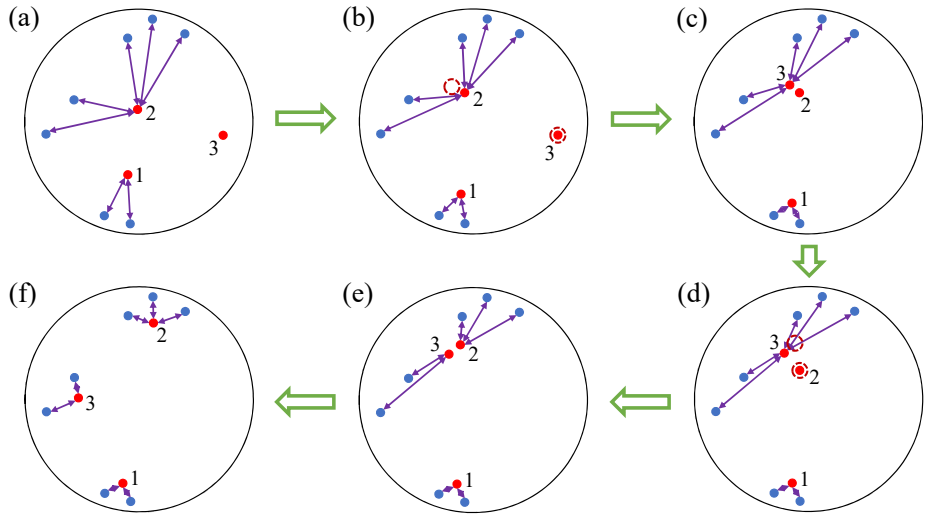


Figure 2: An illustration of the LRU replacement policy. Three red branch vectors are allocated to fit the blue labels, with one continuously being lazy (a). After two LRU periods (b)-(c) and (d)-(e), finally each branch vector fits a small set of labels (f). In real cases, the label vectors are also updated, and the geodesic line between any two points in a 2D Poincaré disk is actually a curve.

For a node having a small number q of labels where $q < k$, unavoidably, at least $k - q$ inactive branch vectors exist at any time. However, there is no guarantee that all the k branch vectors will be active if $q > k$. Consider situation (a) shown in figure 2. A node is given $k = 3$ red branch vectors to fit its $q = 7$ blue labels, with the third branch vector being inactive since it is not the closest to any labels. Given the node-label loss function in equation 3 and the node-label distance defined by equation 2, in the following iterations, the presently active branch vectors will be continually updated by the positive node-label pairs of the node, with the inactive branch vector never being optimized positively. Therefore, the second branch vector is continually pulled by five labels and stuck in an under-fitting position, with contrast to the third branch vector always being lazy.

It should be pointed out that a node is allowed to have less than k active branch vectors, which is similar to K-means clustering algorithm having less than K active centroids. But in order to prevent the unbalanced loads shown in figure 2 (a), we aim at making the most of the k branch vectors. We present a solution to the unbalanced loads based on the LRU algorithm.

4.2.2 The Vector Replacement Policy

The least recently used (LRU) algorithm originally describes a cache replacement policy that swaps out the least recently used cache pages to make room for frequently used pages. Here, we apply the LRU algorithm to balance the loads among different branch vectors. The main idea is to move an inactive branch vector to a place near the most active branch vector so as to share the burden of it.

Figure 2 illustrates the optimization process using the LRU policy. In (b), a vector replacement is performed, and branch vector 3 is replaced by a new vector close to branch vector 2, with their margin being $10^{-3} * \epsilon$, where ϵ is a random normal distribution vector. Extreme cases exist that the new vector attracts all the labels of the original vector, as shown in (c), causing the previously active branch vector 2 becomes inactive. However, iterations of the LRU policy will finally achieve a correct vector replacement. As illustrated in (d), after an LRU period, the inactive branch vector 2 is relocated so that branch vectors 2 and 3 each attracts a set of labels, which is shown in (e). Finally each of the three branch vectors is optimized to fit a small group of labels, as shown in (f).

In the implementation, we maintain a hit matrix $H^{n\times k}$ to support the LRU algorithm. Specifically, H_{ij} record the hit number of B_{ij}. A hit of B_{ij} means that B_{ij} fits a label of node v_i. H can be simply maintained by sending a Boolean variable to the function 2 that calculates node-label distances. The Boolean variable indicates whether the node-label pairs are positive samples. Specifically, if a node-

Algorithm 1 The whole learning process.

Input: network N, label taxonomy T, label assignment A, dimension d, the branch vector number k, negative sampling number p, epoch number g, LRU period t
Output: root vectors R, branch vectors B, label vectors Q

initialize $H = \mathbf{0}$; randomly initialize B, R and Q near the origin of the Poincaré Ball
for epoch in $\{1, 2, ..., g\}$ **do**
  negatively sample label-label, node-label and node-node pairs
  compute Loss_{ll} and update Q by batches
  compute Loss_{nl} and update Q and B by batches meanwhile maintaining H
  compute Loss_{nn} and update R by batches
  if epoch mod $t == 0$ **then**
    update B by the LRU algorithm
    $H = \mathbf{0}$
  end if
end for
output R, B, Q and H

label pair is a positive sample, the function will remember the index a of the branch vector that fits the label, and then plus H_{ia} by 1 to record a hit. After a period of optimization, for every node with inactive branch vectors, we replace one of its inactive vectors with a new vector initialized close to the active branch vector with the highest hit value. We then clear H to $\mathbf{0}$ for the next LRU period.

4.3 The Whole Learning Process of HIME

There are three kinds of relationships that need to be preserved: the node-node relationships presented by the network N, the label-label relationships given by the taxonomy T, and the node-label relationships revealed by the label assignment A. We have mainly discussed how to preserve node-label relationships by multi-vector embedding. Here we briefly present the loss functions of the node-node and label-label relationships.

Node-Node Relationships. Node-node relationships are reflected by the edges of the network N. Each node v_i has a single root vector R_i, and we use the root vectors to preserve the node-node relationships. Given the edge set E of the network N, let $\overline{E}$ denote the set of edges not being in E. For every edge e_{ij} in E, we negatively sample p edges from $\overline{E}$ either connecting node v_i or node v_j. We combine these edges into a set of negative samples E_{neg}. With equation 1 calculating the distance between two vectors, we aim at minimizing the following loss function:

$$Loss_{nn} = -\sum_{e_{ij} \in E} \log \sigma(-\operatorname{dist}(R_i, R_j)) - \sum_{e_{ij} \in E_{neg}} \log \sigma(\operatorname{dist}(R_i, R_j)), \tag{4}$$

Label-Label Relationships. The label-label relationships are given by the label hierarchy $T = \{L, S\}$, where $S = \{s_{ij}\}, i, j = 1, 2, ..., m$ contains the parent-child relationships between label pairs. Here, we view edges in S as un-directed. Similarly, we take the edges in S as positive samples, and for every edge s_{ij} in S, we negatively sample p edges from $\overline{S}$ connecting either l_i or l_j, where $\overline{S}$ is the complementary set of S. We gather these negative samples as set S_{neg}. The objective function of label-label relationships is:

$$Loss_{ll} = -\sum_{s_{ij} \in S} \log \sigma(-\operatorname{dist}(Q_i, Q_j)) - \sum_{s_{ij} \in S_{neg}} \log \sigma(\operatorname{dist}(Q_i, Q_j)). \tag{5}$$

Given the three Loss functions 3, 4 and 5, the learning process aims at minimizing the following loss function:

$$Loss_{total} = Loss_{nn} + Loss_{ll} + Loss_{nl}.$$

During the back propagation, we use Riemannian Stochastic Gradient Descent (RSGD) Bonnabel (2013) to update R, B and Q in the Poincaré ball.

Algorithm 1 shows the whole learning process of HIME. The space complexity of HIME is $O((1 + k)|V|d + |L|d)$, which is the space occupied by R, B and Q. For simplicity, we use $|A|$ to refer to the

total number of ground-truth node-label pairs. Given the predetermined negative sampling number p, during each epoch, $(1+p)|S|$ label-label pairs, $(1+p)|A|$ node-label pairs and $(1+p)|E|$ node-node pairs are passed to the model. The extra cost of our method lies in the minimization computation defined by equation 2, which is $O(k)$. Therefore the total time complexity of learning for g epochs is $O(gp|S| + gkp|A| + gp|E|)$, which grows linearly with the increase of the branch vector number k.

5 EXPERIMENTS

We conduct the experiments on datasets from different domains to evaluate HIME's performance in preserving node-label and node-node relationships compared with existing methods.

DBLP. Yang et al. combines a DBLP co-authorship network with ACM research taxonomy. The nodes are authors, and the labels are hierarchical key words. A label will be assigned to an author if the paper of the author contains that key word. Based on their dataset, we extract a dense sub-network containing 12379 nodes and 268 labels organized into a 4-level hierarchy. Our DBLP dataset has 12164 un-directed node-node links and 50402 node-label links, therefore a node has 1.97 neighbor nodes and 4.07 labels on average.

STRING-GO. We use the protein-protein interaction network of humans from STRING database Szklarczyk et al. (2021) ,with the Cellular Components domain of Gene Ontology (GO) Ashburner et al. (2000); GO2 (2021) being the taxonomy. The GO annotation of human proteins is given by GOA database Camon et al. (2004); Huntley et al. (2015). The STRING-GO dataset consists of 13840 nodes and 4184 labels organized into a 4-level DAG, and has 348658 un-directed node-node links and 205629 node-label links. On average, a node has 50.38 neighbor nodes and 14.86 labels.

Methods for Comparison. We include 7 existing network embedding methods either classic or state-of-the-art for comparison. Node2Vec Grover & Leskovec (2016), LINE Tang et al. (2015b) and GraphGAN Wang et al. (2018) deal with plain networks, and we run these methods by viewing labels as plain nodes and label-related edges as plain edges. GraphSAGE Hamilton et al. (2017), PTE Tang et al. (2015a) and TaxoGAN Yang et al. (2020) are embedding methods on heterogenous or attributed networks, among which TaxoGAN is the state-of-the-art method on networks with taxonomies. We also include Poincaré Nickel & Kiela (2017), which embeds nodes and labels in the Poincaré Ball to better preserve the hierarchical structures. In the experiments, we set the embedding dimension of all methods to 256. We test HIME with k being 2, 4 and 8, and therefore the dimension of branch vectors are 128, 64, 32 respectively so as to ensure that HIME has the total dimension of 256. All methods are trained for 50 epochs.

5.1 NODE-LABEL RELATIONSHIPS

We design three tasks to evaluate the performance of all methods in preserving node-label relationships, namely label retrieval, label-path retrieval and node retrieval. For better describing the tasks, we first define the proximity score of two objects as the inner product of their representation vectors for methods in Euclidean spaces. For HIME and Poincaré, the score is the negative hyperbolic distance between the two vectors.

Label Retrieval. In the Label-retrieval task, given a node, the method should recall all the ground-truth labels of the node by ranking the scores of all labels with regard to that node. We use the Mean Rank $\mathbf{MR}$ of the ground-truth labels to evaluate the label-retrieval performance on a node. We average the $\mathbf{MR}$ over all nodes to obtain $\overline{\mathbf{MR}}$.

Label-Path Retrieval. The label-path retrieval task evaluates a method's capability of retrieving a correct label-path for a node. Given the node, the method first predicts the label with the highest score at the top level of the taxonomy. If the predicted label falls into one of the node's label-path, then the prediction will move to the sub-tree rooted at the predicted label. The prediction will continue until either a false prediction occurs or the whole label-path is predicted correctly. The accuracy $\mathbf{Acc}$ is computed as the ratio of the number of accurately predicted labels to the label-path length. We use $\overline{\mathbf{Acc}}$ averaged over all nodes to evaluate the label-path retrieval performance.

Node Retrieval. Given a label, the node-retrieval task aims at ranking the nodes of that label ahead of those not belonging to that label. The method ranks the scores of all nodes with regard to the

Table 1: The experiments on the two datasets with the embedding dimensions of all methods being 256. The dimensions of vectors of HIME_2, HIME_4, HIME_8 are 128, 64 and 32 respectively, satisfying the constraint that all methods have 256 dimensions.

DBLP	**node-label**			**node-node**		**label-label**
	$\overline{\mathbf{MR}}$	$\overline{\mathbf{Acc}}$	$\overline{\mathbf{AUPRC}}$	$\mathbf{AUPRC}$	$\mathbf{AUROC}$	$\overline{\mathbf{MR}}$
TaxoGAN	124.61	54.60	4.53	99.53	99.92	136.81
Node2Vec	111.20	33.59	11.23	94.58	97.59	68.67
LINE	85.57	29.83	27.85	82.79	92.74	80.40
GraphSAGE	58.27	66.46	27.56	65.39	87.62	81.51
GraphGAN	90.74	25.53	35.00	**99.88**	**99.98**	76.95
PTE	172.48	10.80	5.04	81.84	92.85	144.71
Poincare	23.72	52.39	45.69	96.12	98.92	81.20
HIME_2	7.72	87.39	91.26	98.47	99.13	50.81
HIME_4	7.36	**87.44**	**96.51**	98.38	99.10	**49.75**
HIME_8	**6.69**	68.95	76.61	97.25	98.42	68.44
STRING-GO	**node-label**			**node-node**		**label-label**
	$\overline{\mathbf{MR}}$	$\overline{\mathbf{Acc}}$	$\overline{\mathbf{AUPRC}}$	$\mathbf{AUPRC}$	$\mathbf{AUROC}$	$\overline{\mathbf{MR}}$
TaxoGAN	2356.12	**73.71**	4.69	91.09	96.77	2053.35
Node2Vec	617.73	12.80	19.10	69.49	87.36	611.01
LINE	1110.57	14.98	27.61	78.06	91.09	1072.03
GraphSAGE	1256.21	9.90	20.24	43.33	77.38	1165.21
GraphGAN	1231.63	12.45	25.74	**92.98**	**97.84**	1325.69
PTE	2891.33	10.98	5.07	79.68	92.35	1644.75
Poincare	110.39	18.12	22.10	85.90	93.10	710.47
HIME_2	91.33	26.00	53.64	86.59	94.84	596.30
HIME_4	**86.44**	17.93	70.16	86.64	94.91	**562.49**
HIME_8	90.30	19.13	**78.85**	86.70	94.98	584.19

label, and we use the Area Under the Precision-Recall Curve $\mathbf{AUPRC}$ to evaluate the ranking. We then average the $\mathbf{AUPRC}$ over all labels to obtain $\overline{\mathbf{AUPRC}}$ evaluating the overall performance.

The three 'node-label' columns of table 1 correspond to the three tasks. In the label retrieval task, HIME and Poincaré are the best two methods on both datasets, mainly because the Poincaré Ball has more embedding capacity compared to its Euclidean counterparts with the same dimension. Moreover, HIME performs slightly better than Poincaré, in that HIME minimizes the distances between a node and its labels by using multiple branch vectors, causing the ground-truth labels ranked further ahead.

In the label-path retrieval task, HIME and TaxoGAN are the two methods having the overall best performances on both DBLP and STRING-GO, mainly because they both allow multiple vectors of a node in essence. On DBLP, HIME obtains higher $\overline{\mathbf{Acc}}$ than TaxoGAN, While on STRING-GO, TaxoGAN outperforms all other algorithms with a considerable margin. This is because TaxoGAN is specialized in retrieving a node's ground-truth label from the label's siblings by embedding them into a small sub-graph. Compared to DBLP which has only 268 labels, STRING-GO has 4184 labels, which makes the effect of decomposing the taxonomy into different sub-graphs more remarkable. However, the drawback of decomposition is also obvious since the embedding vectors of the nodes only function locally in different sub-graphs, and this side-effect will be revealed by the task of node retrieval as follows.

By allowing multiple globally functional vectors of a node, HIME significantly outperforms all other methods on the task of node retrieval. HIME_4 and HIME_8 are the best on DBLP and STRING-GO respectively. By contrast, TaxoGAN performs poorly on the node-retrieval task. This is because in TaxoGAN, a local embedding vector of a node is not able to reflect the node's relationship with a label outside of the sub-graph. Therefore, the scores between most node-label pairs are almost random, which accounts for the poor ranking performance of TaxoGAN.

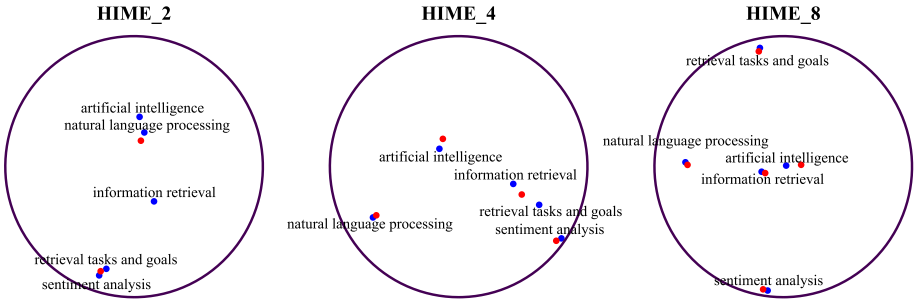


Figure 3: The research areas of an author in a 2D Poincaré disk. The blue points are the hierarchical labels, with the red points being the author's active branch vectors. 'Information retrieval' and 'retrieval tasks and goals' are embedded apart in HIME_8.

5.2 Node-Node Relationships

We apply the task of node-pair retrieval to evaluate the performance in preserving node-node relationships. Specifically, we randomly sample positive and negative node-node pairs, with their ratio being 1:4. The methods calculate the score of every node-node pair, and generate a ranking for all node-node pairs. We use AUPRC and AUROC to evaluate the node-node pair retrieval performance.

The columns of 'node-node' in table 1 display the results of node-pair retrieval. GraphGAN has the highest AUPRC and AUROC values compared to other methods, followed by TaxoGAN, HIME and Poincaré. The power of adversarial training can be inferred from the good performance of TaxoGAN and GraphGAN. Though not the best on the node-node pair retrieval task, HIME performs better than other Euclidean and hyperbolic methods. The success of HIME in preserving node-node relationships can be partly attributed to the branch vectors, which liberate the root vectors from the burden of fitting labels.

5.3 Label-Label Relationships

We use the retrieval of parent-child edges to evaluate the preservation of label taxonomy. Given a label, the methods rank the scores of all other labels with respect to the label. We use the Mean Rank $\mathbf{MR}$ of the its immediate child labels to evaluate the preservation of hierarchy. We average the $\mathbf{MR}$ over all labels to obtain $\overline{\mathbf{MR}}$.

The 'label-label' column in table 1 shows the results. HIME, Poincaré and Node2Vec are the three best methods, with HIME_4 being the best on both DBLP an STRING-GO. It can be seen from figure 3 and table 1 that the hierarchical label structure of HIME_8 is not well preserved compared to that of HIME_2 and HIME_4. This is because the label vectors are mainly optimized by $Loss_{nl}$. When k increases, the branch vectors of nodes are given more degrees of freedom than the label vectors. The decrease of $Loss_{nl}$ is mainly contributed to the branch vectors instead of the label vectors, causing the label vectors being optimized insufficiently. Therefore, we recommend setting k less than 5 so as to gain a significant increase in performance compared to single-vector embedding methods, meanwhile keeping a low time complexity.

6 Conclusions

We propose a method called HIME, which endows a node with multiple vectors to alleviate the effect of under-fitting in the hyperbolic embedding space. Our multi-vector embedding can be easily extended to Euclidean spaces by simply changing the distance function, and can also be applied in plain networks where a node belongs to multiple communities. We point out that though multiple vectors can help preserving the node's relationships with labels, they may also distort the label hierarchy by over-fitting. Therefore, we suggest setting a small multi-vector number so as to achieve a notable improvement in performance with a small time cost. But how to strike an embedding balance between the node-label relationships and the label hierarchy awaits future researches.

REFERENCES

The gene ontology resource: enriching a gold mine. *Nucleic Acids Research*, 49(D1):D325–D334, 2021.

Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Marián Boguná, Fragkiskos Papadopoulos, and Dmitri Krioukov. Sustaining the internet with hyperbolic mapping. *Nature communications*, 1(1):1–8, 2010.

Silvere Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

Evelyn Camon, Michele Magrane, Daniel Barrell, Vivian Lee, Emily Dimmer, John Maslen, David Binns, Nicola Harte, Rodrigo Lopez, and Rolf Apweiler. The gene ontology annotation (goa) database: sharing knowledge in uniprot with gene ontology. *Nucleic Acids Research*, 32(suppl_1): D262–D266, 2004.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pp. 891–900, 2015.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in Neural Information Processing Systems*, 32:4868–4879, 2019.

Boli Chen, Xin Huang, Lin Xiao, Zixin Cai, and Liping Jing. Hyperbolic interaction model for hierarchical multi-label classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7496–7503, 2020.

Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 135–144, 2017.

Alessandro Epasto and Bryan Perozzi. Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference*, pp. 394–404. Association for Computing Machinery, 2019.

Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 26th ACM International Conference on Information and Knowledge Management*, pp. 1797–1806, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Mikhael Gromov. Hyperbolic groups. In *Essays in group theory*, pp. 75–263. Springer, 1987.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864, 2016.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, 2017.

Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. Universal representation learning of knowledge bases by jointly embedding instances and ontological concepts. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1709–1719, 2019.

Rachael P Huntley, Tony Sawford, Prudence Mutowo-Meullenet, Aleksandra Shypitsyna, Carlos Bonilla, Maria J Martin, and Claire O'Donovan. The goa database: gene ontology annotation updates for 2015. *Nucleic Acids Research*, 43(D1):D1057–D1063, 2015.

Rana Hussein, Dingqi Yang, and Philippe Cudré-Mauroux. Are meta-paths necessary? revisiting heterogeneous graph embeddings. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 437–446, 2018.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Ninghao Liu, Qiaoyu Tan, Yuening Li, Hongxia Yang, Jingren Zhou, and Xia Hu. Is a single vector enough? exploring node polysemy for network embedding. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 932–940, 2019.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30, 2017.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1105–1114, 2016.

Chanyoung Park, Carl Yang, Qi Zhu, Donghyun Kim, Hwanjo Yu, and Jiawei Han. Unsupervised differentiable multi-aspect network embedding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1435–1445, 2020.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, 2014.

Fatima Zohra Smaili, Xin Gao, and Robert Hoehndorf. Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics*, 34(13):i52–i60, 2018.

Damian Szklarczyk, Annika L Gable, Katerina C Nastou, David Lyon, Rebecca Kirsch, Sampo Pyysalo, Nadezhda T Doncheva, Marc Legeay, Tao Fang, Peer Bork, et al. The string database in 2021: customizable protein–protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic Acids Research*, 49(D1):D605–D612, 2021.

Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1165–1174, 2015a.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077, 2015b.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, 2016.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Junshan Wang, Zhicong Lu, Guojie Song, Yue Fan, Lun Du, and Wei Lin. Tag2vec: Learning tag representations in tag networks. In *The World Wide Web Conference*, pp. 3314–3320, 2019.

Carl Yang, Jieyu Zhang, and Jiawei Han. Co-embedding network nodes and hierarchical labels with taxonomy based generative adversarial networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 721–730. IEEE, 2020.

A APPENDIX

A.1 RIEMANNIAN STOCHASTIC GRADIENT DESCENT

Riemannian Stochastic Gradient Descent (RSGD) Bonnabel (2013) is a technique that updates parameters by back-propagation in hyperbolic spaces. Here we briefly introduce the RSGD process applied by Nickel & Kiela (2017) in a Poincaré ball.

Suppose we are given a set $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_i\}_{i=1}^n$ of parameters in a Poincaré ball of radius 1: $\forall \boldsymbol{\theta}_i \in \boldsymbol{\Theta}, \|\boldsymbol{\theta}_i\| < 1$. Given a loss function $L(\boldsymbol{\Theta})$, We aim at optimizing $\boldsymbol{\Theta}$ by minimizing $L(\boldsymbol{\Theta})$:

$$\boldsymbol{\Theta}^{'} \leftarrow \arg\min_{\boldsymbol{\Theta}} L(\boldsymbol{\Theta})$$

We should first calculate the Riemannian gradients for every parameters, and then update the parameters in the Poincaré ball. given the learning rate η, the back propagation in the Poincaré ball can be defined as:

$$\boldsymbol{\theta}_{t+1} \leftarrow P_{\boldsymbol{\theta}_t}(-\eta \nabla_R L(\boldsymbol{\theta}_t)),$$

where P is a retraction function, ∇_R is the Riemannian gradient. The retraction is related to both $\eta \nabla_R L(\boldsymbol{\theta}_t)$ and $\boldsymbol{\theta}$'s position at time t, and its function is similar to the optimization step in Euclidean space simply achieved by:

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta \nabla_E L(\boldsymbol{\theta}_t).$$

The Riemannian gradient is not difficult to calculate. Given the Riemannian metric tensor: $g_{\boldsymbol{\theta}_t} = \lambda_{\boldsymbol{\theta}_t}^2 I$ where $\lambda_{\boldsymbol{\theta}_t} = \frac{2}{1-\|\boldsymbol{\theta}_t\|^2}$ and I being the identity matrix, the Riemannian gradient can be calculated based on the Euclidean gradient ∇_E:

$$\nabla_R L(\boldsymbol{\theta}_t) = g_{\boldsymbol{\theta}_t}^{-1} \nabla_E L(\boldsymbol{\theta}_t).$$

Therefore, given the loss function L, we can first calculate the Euclidean gradient ∇_E of the parameter $\boldsymbol{\theta}$ at time t by traditional back-propagation in Euclidean space, and then divided it by $g_{\boldsymbol{\theta}_t}$ to obtain the Riemannian gradient of $\boldsymbol{\theta}_t$.

Given the Riemannian gradient $\nabla_R L(\boldsymbol{\theta}_t)$, Nickel & Kiela define their updating process as:

$$\boldsymbol{\theta}_{t+1} \leftarrow Q(\boldsymbol{\theta}_t - \eta \nabla_R L(\boldsymbol{\theta}_t)),$$

where

$$Q(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} / \|\boldsymbol{x}\| - \epsilon & \|\boldsymbol{x}\| \geq 1 \\ \boldsymbol{x} & \|\boldsymbol{x}\| < 1 \end{cases},$$

and ϵ serve as a small vector. In this way, if $\boldsymbol{\theta}_{t+1}$ falls out of the Poincaré ball with radius 1, ϵ will pull it back into the ball, therefore $\boldsymbol{\theta}_t$ always satisfies $\|\boldsymbol{\theta}_t\| < 1$.

The drawback of this linear retraction is that it neglects the characteristics of the hyperbolic spaces. By contrast, We use exponential map to perform the retraction. Exponential and logarithmic maps serve as transformation tools between a Poincaré Ball $\mathbb{D}^{d,1}$ and an Euclidean tangent space $\mathbb{E}^d$. For any point $\boldsymbol{x} \in \mathbb{D}^{d,1}$, the exponential map and the logarithm map for $\boldsymbol{v} \neq \boldsymbol{0}$ and $\boldsymbol{y} \neq \boldsymbol{x}$ are:

$$\exp_{\boldsymbol{x}}(\boldsymbol{v}) = \boldsymbol{x} \oplus (\tanh(\frac{\lambda_{\boldsymbol{x}} \|\boldsymbol{v}\|}{2}) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}),$$

$$\log_{\boldsymbol{x}}(\boldsymbol{y}) = \frac{2}{\lambda_{\boldsymbol{x}}} \tanh^{-1}(\|-\boldsymbol{x} \oplus \boldsymbol{y}\|) \frac{-\boldsymbol{x} \oplus \boldsymbol{y}}{\|-\boldsymbol{x} \oplus \boldsymbol{y}\|},$$

where

$$\boldsymbol{x} \oplus \boldsymbol{y} = \frac{(1 + 2\langle \boldsymbol{x}, \boldsymbol{y} \rangle + \|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 - \|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 + 2\langle \boldsymbol{x}, \boldsymbol{y} \rangle + \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2}.$$

The Riemannian gradient $\nabla_R L(\boldsymbol{\theta}_t)$ can be viewed as a vector in the tangent space of $\boldsymbol{\theta}_t$. Therefore, we can perform the optimization step on $\boldsymbol{\theta}_t$ by mapping the Riemannian gradient from $\boldsymbol{\theta}_t$'s tangent space to the Poincaré ball to obtain $\boldsymbol{\theta}_{t+1}$. Formally, we finally gets $\boldsymbol{\theta}_{t+1}$ by:

$$\boldsymbol{\theta}_{t+1} \leftarrow \exp_{\boldsymbol{\theta}_t}(-\nabla_R L(\boldsymbol{\theta}_t)).$$

Table 2: The experiments on preserving node-label relationships. HIME_k refers to HIME using k branch vectors. All results are shown in percentage.

DBLP	$1-\overline{\mathbf{MR}}/\lvert\mathbf{L}\rvert$			$\overline{\mathbf{Acc}}$			$\overline{\mathbf{AUPRC}}$		
Dimension	**32**	**8**	**2**	**32**	**8**	**2**	**32**	**8**	**2**
TaxoGAN	79.14	79.47	78.99	65.27	65.22	63.31	15.41	13.97	8.24
Node2Vec	86.20	80.32	62.56	60.14	45.81	24.25	56.55	34.37	10.39
LINE	84.56	82.40	70.84	58.34	46.82	42.71	49.34	33.98	15.55
GraphSAGE	87.31	85.83	81.02	62.94	47.60	41.69	54.64	38.39	13.10
GraphGAN	85.01	69.58	62.86	52.62	31.18	24.23	53.75	32.46	12.71
PTE	86.17	83.12	77.04	61.96	52.53	46.38	58.13	36.01	18.40
Poincaré	96.60	96.28	94.64	62.28	59.85	55.57	50.62	41.36	36.62
HIME_2	98.30	98.43	97.97	94.78	90.46	82.81	87.97	82.92	71.99
HIME_4	98.58	98.72	**98.63**	94.56	92.08	**90.82**	93.78	90.96	86.50
HIME_8	**98.92**	**98.90**	98.28	**97.82**	**95.04**	72.94	**97.42**	**95.22**	**87.36**
STRING-GO	$1-\overline{\mathbf{MR}}/\lvert\mathbf{L}\rvert$			$\overline{\mathbf{Acc}}$			$\overline{\mathbf{AUPRC}}$		
Dimension	**32**	**8**	**2**	**32**	**8**	**2**	**32**	**8**	**2**
TaxoGAN	82.48	80.37	78.11	**53.09**	**49.12**	**43.34**	5.03	4.92	4.70
Node2Vec	75.50	70.65	58.85	23.38	13.10	6.08	27.58	16.02	7.19
LINE	72.03	68.54	54.78	19.10	16.55	8.96	19.05	14.06	6.65
GraphSAGE	76.35	72.93	62.22	22.06	17.61	9.62	24.93	16.28	8.25
GraphGAN	76.88	69.79	60.07	25.90	19.58	8.64	28.82	17.83	7.77
PTE	80.94	73.67	66.28	26.57	18.42	9.32	29.10	18.80	10.12
Poincaré	97.02	96.56	90.96	24.89	18.61	14.31	25.46	21.26	15.59
HIME_2	98.01	97.65	97.75	30.67	23.70	22.93	38.66	35.84	25.11
HIME_4	98.35	98.31	98.17	33.77	29.06	25.90	57.76	55.56	38.76
HIME_8	**98.66**	**98.72**	**98.37**	40.18	32.63	29.69	**69.67**	**68.30**	**43.66**

A.2 Experiments

A.2.1 Dataset Construction

DBLP. We sample a subset of the DBLP network with the research taxonomy, where each author has at least one co-author in the network. Every author in the network has at least one label in the research taxonomy.

STRING-GO. We use the protein-protein interaction network (PPI) of humans provided by STRING Szklarczyk et al. (2021). We set a threshold of 800 to sample the edges in the original data to form our PPI network. We then delete the proteins not appearing in GOA Camon et al. (2004); Huntley et al. (2015), so that each protein has its GO-terms in the GO taxonomy.

A.2.2 Parameter Settings

On both datasets, we set the learning rate of the root vectors R and the label vectors Q to 0.01, while for the branch vectors B we set their learning rate to 0.02. The negative sampling number is set to 5, and the batch size is 1000. The LRU period is set to 5 epochs. We train HIME on both DBLP and STRING-GO for 100 epochs.

A.2.3 Results

Table 2 and figure 4 reveal the performance of all methods on the node-label preservation under different dimensions. Besides, table 3 and figure 5 tell detailed results of node-node preservation.

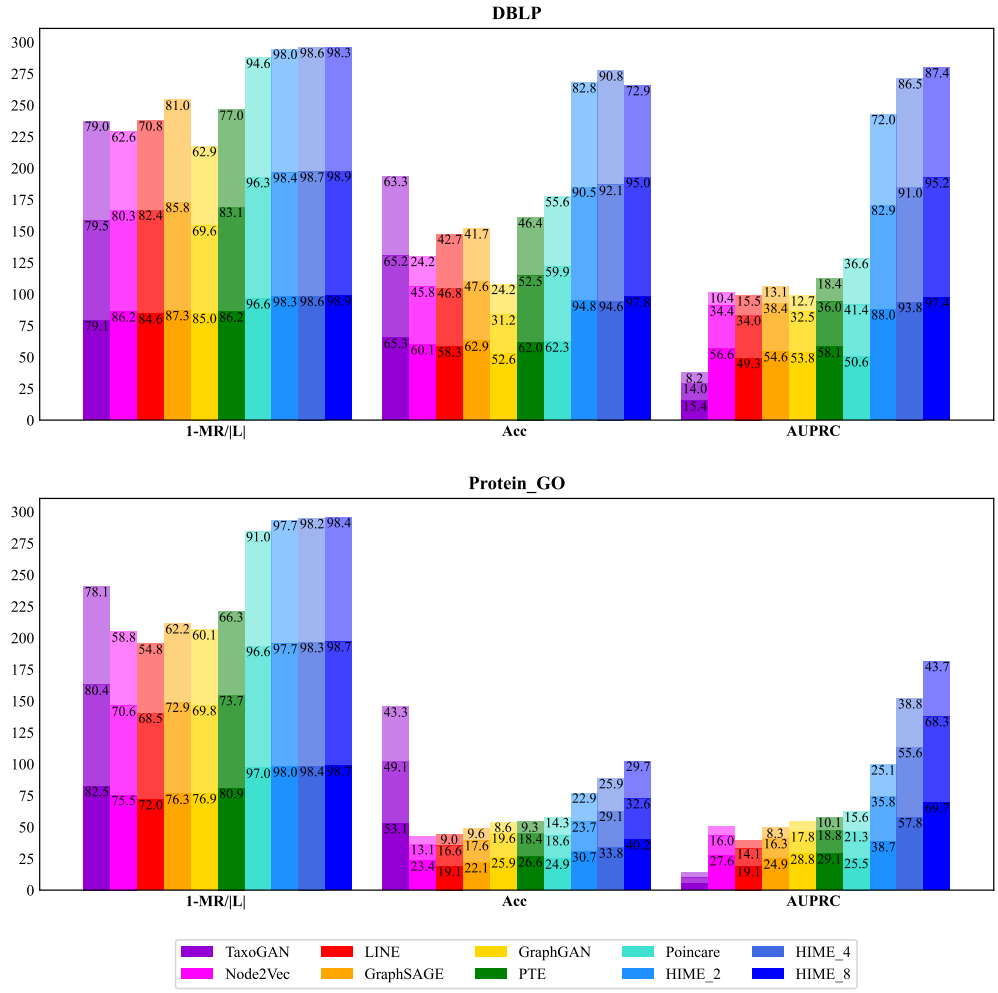


Figure 4: The experiments on preserving node-label relationships.

A.2.4 The Distortion of the Hierarchy

As shown in figure 6, 7, and 8, with the increase of the branch vector number k, the hierarchy structure of the labels is less preserved. Here we mainly present our understanding of hierarchy distortion caused by multi-vector embedding. Given the total loss function:

$$Loss_{total} = Loss_{nn} + Loss_{ll} + Loss_{nl},$$

We should first figure out which part of the loss function contributes to the preservation of the label hierarchy most, and then find the reason of the hierarchy distortion.

At first glance, one may first guess that $Loss_{ll}$ helps preserving the label hierarchy most, since it depicts the parent-child relationships among labels. However, by using the taxonomy information T alone, the machine are not likely to generate a correct tree structure in the Poincaré ball, with the root label embedded close to the origin, while the leaf labels embedded near the border. This can be explained by the fact that each label in a taxonomy tree with un-directed links can be equally viewed as the root. The un-directed parent-child set S gives no extra information of the true root labels to the machine.

Since $Loss_{nn}$ is specialized in preserving node-node relationships, $Loss_{nl}$ becomes the only answer. Labels in a higher hierarchy level will be frequently updated by the positive node-label

Table 3: The experiments on preserving node-node relationships.

Dataset	**DBLP**					
Metric	**AUPRC**			**AUROC**		
Dimension	**32**	**8**	**2**	**32**	**8**	**2**
TaxoGAN	99.12	98.18	64.34	99.70	99.29	90.67
Node2Vec	88.62	73.81	29.16	96.27	90.42	65.18
LINE	83.46	54.30	26.14	93.22	78.67	53.22
GraphSAGE	81.25	59.06	23.43	90.94	79.11	55.67
GraphGAN	99.38	96.43	71.15	**99.95**	99.10	90.75
PTE	88.38	85.45	47.02	96.04	95.07	73.89
Poincare	94.42	91.95	81.23	97.52	96.04	89.32
HIME	**99.56**	**99.46**	**98.03**	99.93	**99.91**	**99.27**
Dataset	**STRING-GO**					
Metric	**AUPRC**			**AUROC**		
Dimension	**32**	**8**	**2**	**32**	**8**	**2**
TaxoGAN	93.10	85.61	41.83	97.47	93.73	74.90
Node2Vec	87.84	79.10	43.10	96.02	89.11	71.01
LINE	75.98	58.94	38.89	90.52	79.98	64.96
GraphSAGE	88.65	77.26	34.03	96.62	89.12	62.87
GraphGAN	93.85	86.49	55.00	97.89	95.15	82.03
PTE	89.29	75.49	41.40	97.09	88.78	73.57
Poincare	90.51	82.12	69.25	97.01	91.69	85.25
HIME	**94.04**	**88.90**	**85.05**	**98.09**	**95.95**	**94.13**

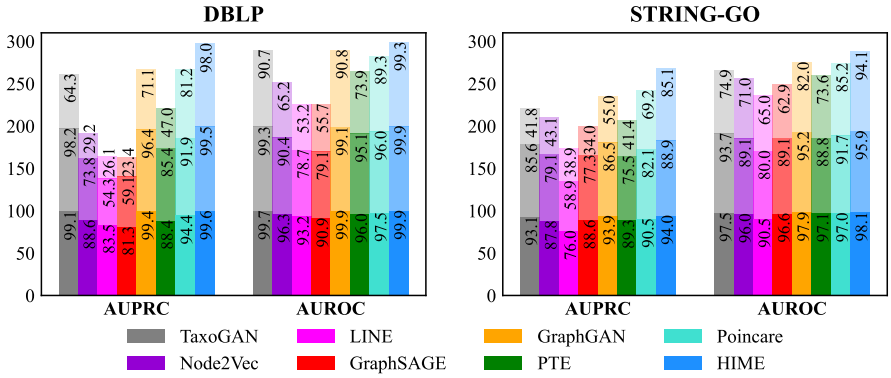


Figure 5: The experiments on preserving node-node relationships. The different shades of the colors represent the results under dimension 32, 8 and 2 from bottom to top.

pairs, therefore single-vector Poincaré embedding will place high-level labels near the center of the Poincaré ball so as to reduce the overall node-label distances. While low-level labels are pushed away from the center by negative samples. But when excessive branch vectors are allowed, the node-label distances call be decreased by updating multiple branch vectors, with labels being lazy and stuck in the middle, as shown in figure 8. Therefore, the hierarchy of the labels will be distorted. To prevent the hierarchy distortion, a small branch vector number is suggested.

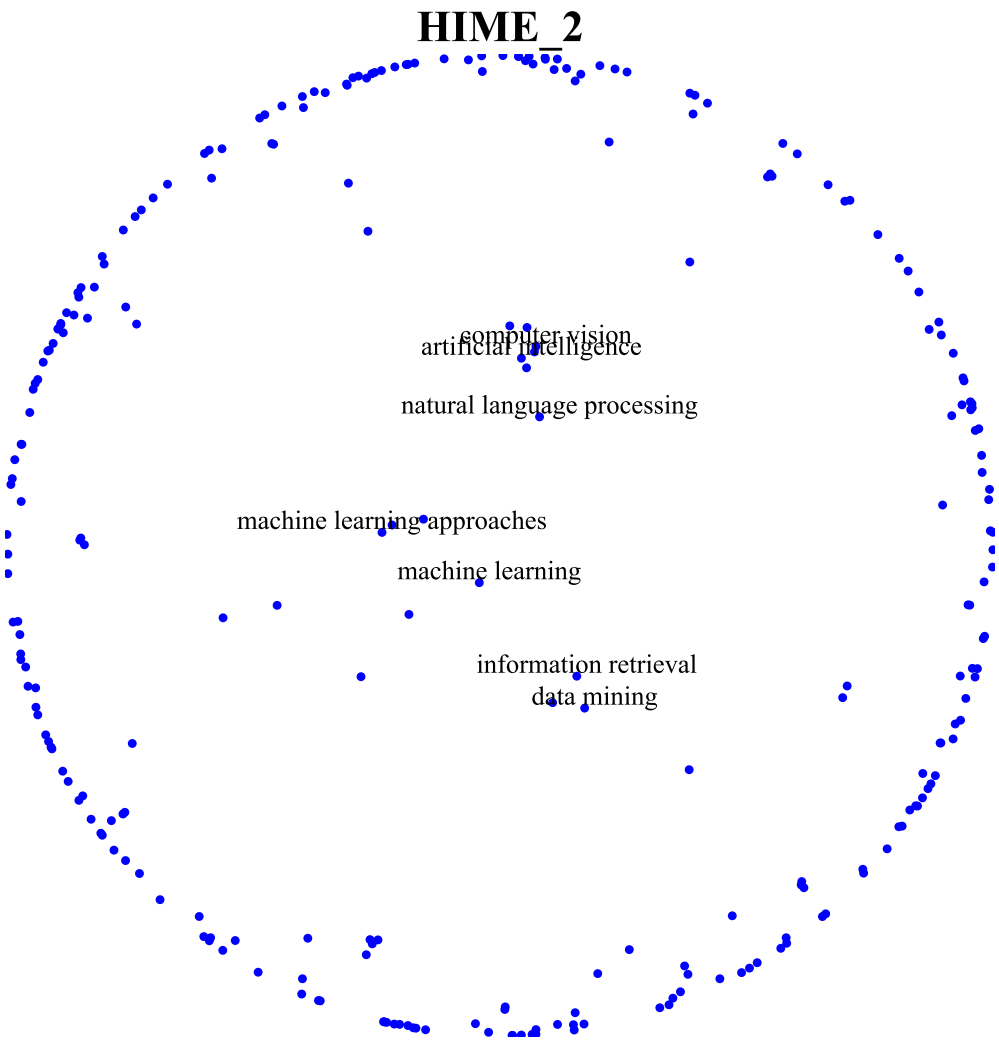


Figure 6: The embedding vectors of the research taxonomy produced by HIME_2.

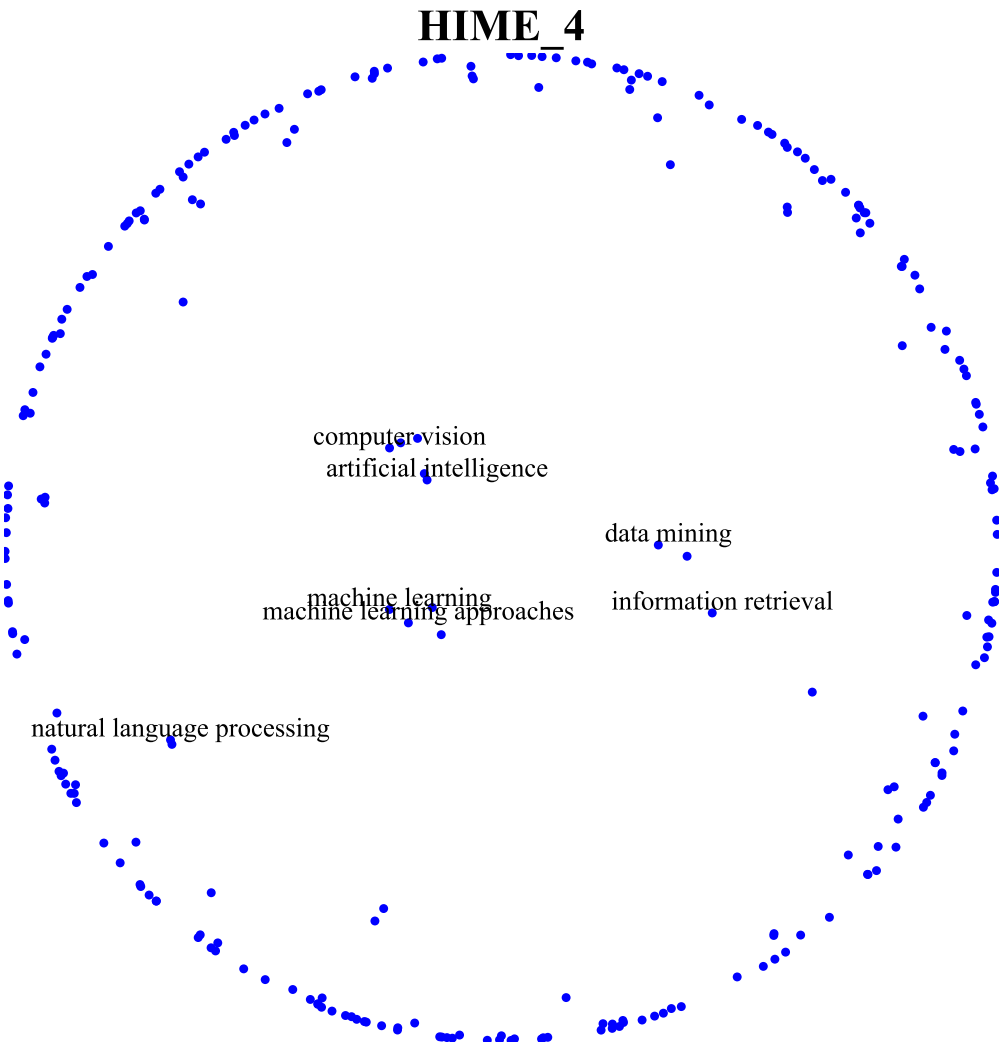


Figure 7: The embedding vectors of the research taxonomy produced by HIME_4.

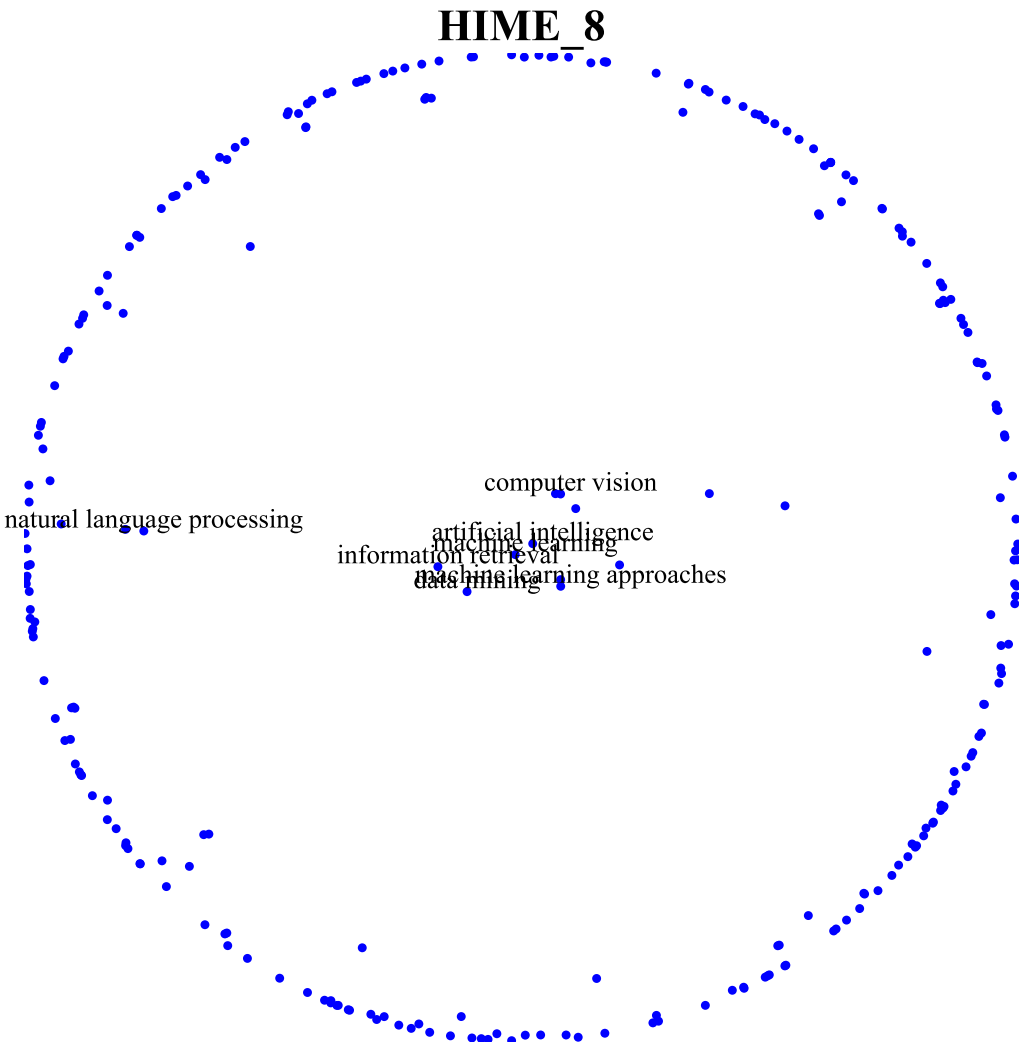


Figure 8: The embedding vectors of the research taxonomy produced by HIME_8.