Supplementary Manuscript for triUMPF - Leveraging Heterogeneous Network Embedding for Metabolic Pathway Prediction

Abdur Rahman M. A. Basher and Steven J. Hallam

February 20, 2020

Abstract

Metabolic pathway reconstruction from genomic sequence information is a key step in predicting regulatory and functional potential of cells at the individual, population and community levels of organization. Although the most common methods for metabolic pathway reconstruction are gene-centric e.g. mapping annotated proteins onto known pathways using a reference database, pathway-centric methods based on heuristics or machine learning to infer pathway presence provide a powerful engine for hypothesis generation in biological systems. Such methods rely on rule sets or rich feature information that may not be known or readily accessible. Here, we present pathway2vec, a software package consisting of six representational learning based modules used to automatically generate features for pathway inference. Specifically, we build a three layered network composed of compounds, enzymes, and pathways, where nodes within a layer manifest inter-interactions and nodes between layers manifest betweenness interactions. This layered architecture captures relevant relationships used to learn a neural embedding-based low-dimensional space of metabolic features. We benchmark pathway2vec performance based on node-clustering, embedding visualization and pathway prediction using MetaCyc as a trusted source. Remarkably, in the pathway prediction task, all the modules indicate that it is possible to leverage embeddings to improve pathway prediction than the alternative approaches.

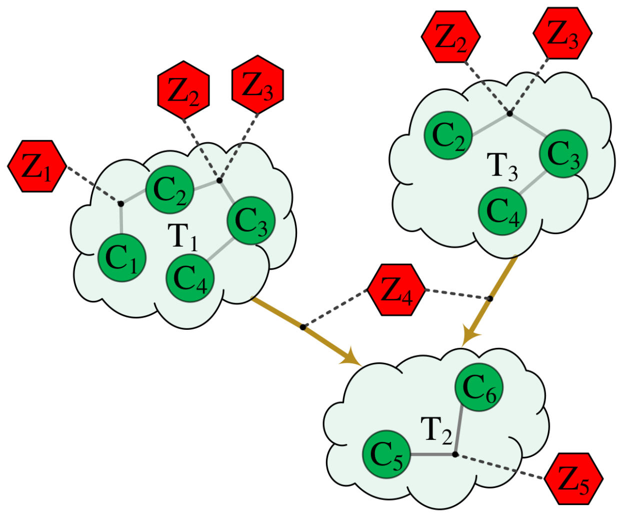
1 Introduction

Metabolic pathway reconstruction from genomic sequence information is a key step in predicting regulatory and functional potential of cells at the individual, population and community levels of organization. ([1]). Exponential advances in sequencing throughput continue to lower the cost of data generation with concomitant increases in data volume and complexity ([2]). Resulting data sets create new opportunities for metabolic reconstruction within biological systems that require the development of new computational tools and approaches that scale with data volume and complexity. Although the most common methods for metabolic pathway reconstruction are gene-centric e.g. mapping annotated proteins onto known pathways using a reference database based on sequence homology, heuristic or rule-based methods for pathway-centric inference including PathoLogic ([18]) and MinPath ([32]) have become increasingly used to generate hypotheses and build quantitative models. Pathologic generates pathway genome databases (PGDBs) that can be refined based on experimental validation e.g. EcoCyc ([19]) and stored in repositories e.g. BioCyc ([7]).

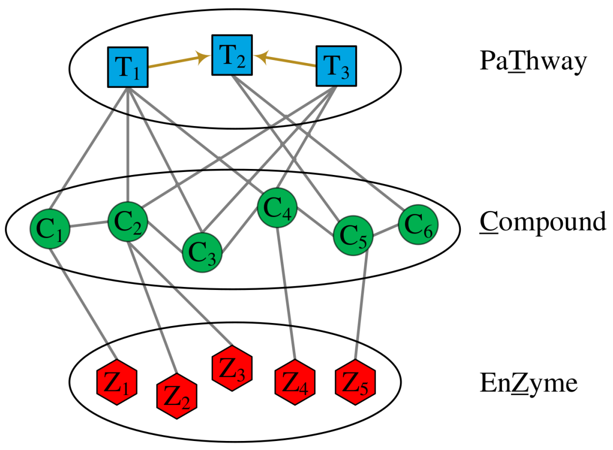
The development of accurate and flexible rule sets for pathway prediction remains a challenging enterprise informed by expert curators incorporating thermodynamic, kinetic, and structural information for validation ([30]). Updating these rule sets as new orgaisms or pathways are discovered and validated can be cumbersome and out of phase with current user needs. This has led to the consideration of machine learning (ML) approaches for pathway prediction based on rich feature information. Dale and colleagues conducted a seminal study comparing the performance of Pathologic to different types of supervised ML algorithms (naive Bayes, k nearest neighbors, decision trees and logistic regression), converting rules into features, defining new features, and evaluating on experimentally validated pathways from six highly curated organisms in the BioCyc collection randomly divided into training and test sets ([8]). Resulting performance metrics indicated that generic ML methods equaled the performance of Pathologic with the benefit of probability estimation for pathway presence and increased flexibility and transparency of use.

Despite the potential benefits of adopting ML methods for pathway prediction from genomic sequence information, Pathologic remains the primary inference engine of Pathway Tools ([18]), and alternative methods for pathway-centric inference expanding on the generic methods described above remain nascent. Several of these methods incorporate metabolite information to improve pathway inference and reaction rules to infer metabolic pathways ([5, 30, 29]). Other methods including BiomeNet ([26]) and MetaNetSim ([16]) dispense with pathways all together and model reaction networks based on enzyme abundance information. We recently implemented a multi-label classification approach to metabolic pathway inference using rich pathway feature information called mlLGPR ([4]). mlLGPR uses logistic regression and feature vectors based in part on the work of Dale and colleagues to predict metabolic pathways for individual genomes as well as more complex cellular communities e.g. microbiomes. One of the primary challenges encountered in developing mlLGPR relates to engineering reliable features representing heterogeneous and degenerate functions within cellular communities ([20]).

Advances in representational learning have led to the development of scalable methods for engineering features from graphical networks e.g. networks composed of multiple nodes including information systems or social networks ([13, 9, 25])). These approaches learn feature vectors for nodes in a network by solving an optimization problem in an unsupervised manner, using random walks followed by Skip-Gram extraction of low dimensional latent continuous features, known as *embeddings* ([23]). Here we present pathway2vec, a software package incorporating multiple algorithms for representational learning used to automatically generate feature representations of metabolic pathways, which are decomposed into three interacting layers: compounds, enzymes and pathways, where each layer consists of associated nodes. A Skip-Gram model is applied to extract embeddings for each node, encoding smooth decision boundaries between groups of nodes in that graph. Nodes within a layer manifest inter-interactions and nodes between layers manifest betweenness interactions resulting in a multi-layer heterogeneous information network ([27]). This layered architecture captures relevant relationships used to learn a neural embedding-based low-dimensional space of metabolic features (Fig 1).



(a)



(b)

Figure 1: Three interacting metabolic pathways (a), depicted as a cloud glyph, where each pathway is comprised of compounds (green) and enzymes (red). Interacting compound, enzyme and pathway components are transformed into a multi-layer heterogeneous information network (b).

In addition to implementing several published methods, we develop a novel random walk algorithm, RUST (unit-circle based jump and stay random walk), adopting a unit-circle equation to sample node pairs that generalize previous random walk methods ([13, 9, 15]). The modules in pathway2vec were benchmarked based on node-clustering, embedding visualization, and pathway prediction. In the case of pathway prediction, leveraging embedding, generated using modules from pathway2vec, provide a viable solution to overcome with hand-crafted features for the use of ML based metabolic pathway reconstruction from genomic sequence information. To the best of our knowledge, this is the first study that employs a heterogeneous graph to automatically recover latent structure and semantic features of metabolic pathways, and to incorporate the learned embeddings for metabolic pathway inference.

2 Definitions and Problem Statement

In this section, we formulate the problem of metabolic feature engineering using a heterogeneous information network. Throughout the paper, all vectors are column vectors denoted by boldface lowercase letters (e.g., $\mathbf{x}$) while matrices are represented by boldface uppercase letters (e.g., $\mathbf{X}$). The $\mathbf{X}_i$ matrix denotes the i-th row of $\mathbf{X}$ and $\mathbf{X}_{i,j}$ denotes the (i,j)-th element of $\mathbf{X}$. A subscript character to a vector, $\mathbf{x}_i$, denotes an i-th cell of $\mathbf{x}$. Occasional superscript, $\mathbf{X}^{(i)}$, suggests an index to a sample, position, or current epoch during learning period. We use calligraphic letters to represent sets (e.g., $\mathcal{E}$) while we use the notation $|.|$ to denote the cardinality of a given set. With these notations, we define a multi-label pathway dataset.

Definition 2.1. Multi-label Pathway Dataset ([4]). A pathway dataset is characterized by $\mathcal{S} = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) : 1 < i \leqslant n\}$ consisting of n examples, where $\mathbf{x}^{(i)}$ is a vector indicating abundance information for each enzymatic reaction denoted by z, which is an element of a set $\mathcal{Z} = \{z_1, z_2, ..., z_r\}$, having r possible reactions. The abundance of an enzymatic reaction for a given example i, say $z_l^{(i)}$, is defined as

$a_l^{(i)} (\in \mathbb{R}_{\geq 0})$. The class label $\mathbf{y}^{(i)} = [y_1^{(i)}, ..., y_t^{(i)}] \subseteq \{-1, +1\}^t$ is a pathway label vector of size t representing the total number of pathways derived from a trusted source of experimentally validated metabolic pathways $\mathcal{Y}$. The matrix form of $\mathbf{x}^{(i)}$ and $\mathbf{y}^{(i)}$ are symbolized as $\mathbf{X}$ and $\mathbf{Y}$, respectively. ■

Both $\mathcal{E}$ and $\mathcal{Y}$ are derived from trusted sources, such as KEGG ([17]) or MetaCyc ([6]). We assume that there is a numerical representation behind every instance and label. We use $\mathcal{X} = \mathbb{R}^r$ to denote the r-dimensional feature vector (input space), encoding various information about an instance, and $\mathcal{U} = \mathbb{R}^d$ for the d-dimensional numerical label vector. Furthermore, each example in $\mathcal{S}$ is considered to be drawn independent, identically distributed (i.i.d) from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times 2^{|\mathcal{Y}|}$.

The pathway inference task can be formulated as retrieving a set of pathway labels for an example i given features learned according to a *heterogeneous information network* defined as:

Definition 2.2. Heterogeneous Information Network A heterogeneous information network is defined as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ and $\mathcal{E}$ denote to the set of nodes and edges (either directed or undirected), respectively ([28]). Each $v \in \mathcal{V}$ is associated with an object type mapping function $\phi(v) : \mathcal{V} \rightarrow \mathcal{O}$, where $\mathcal{O}$ represents a set of object types. Each edge $e \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ includes multiple types of links, and is associated with a link type mapping function $\phi(e) : \mathcal{E} \rightarrow \mathcal{R}$, where $\mathcal{R}$ represents a set of relation types. In particular, when $|\mathcal{O}| + |\mathcal{R}| > 2$, the graph is referred to as a heterogeneous information network. ■

In heterogeneous information networks, both object types and relationship types are explicitly segregated. For the undirected edges, notice that if a relation exists from a type $O_i (\in \mathcal{O})$ to a type $O_j (\in \mathcal{O})$, denoted as $O_i R O_j$ and $R \in \mathcal{R}$, the inverse relation R^{-1} holds naturally for $O_j R^{-1} O_i$. However, in many circumstances, R and its inverse R^{-1} are not equal, unless the two objects are in the same domain, and R is symmetric. In addition, the network may be weighted where each edge $e_{i,j}$, of nodes i and j, is associated with a weight of type R. The linkage type of an edge automatically defines the node types of it's end points. The graph articulated in this paper is considered directed and weighted (in some cases), but for simplification is converted to a undirected network by simply treating edges as symmetric links. Note that if $|\mathcal{O}| = |\mathcal{R}| = 1$, the network is *homogeneous*; otherwise, it is heterogeneous.

***Example* 2.2.1.** MetaCyc can be abstracted as a heterogeneous information network, in Fig 1(b), which contains 3 types of objects, namely compounds (C), enzymes (Z), and pathways (T). There exist different types of links between objects representing semantic relationships e.g. *"composed of"* and *"involved in"*, relationships between pathways and compounds or relations between enzymes and compounds e.g. *"transform"* and *"transformed by"*. An enzyme may be mapped to a numerical category, known as an enzyme commission number (EC) based on the chemical reaction it catalyzes.

Two objects within heterogeneous information networks describe meta-level relationships refereed to as *meta-paths*.

Definition 2.3. Meta-Path A meta-path $P \in \mathcal{P}$ is a path over $\mathcal{G}$ in the form of $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} O_i \xrightarrow{R_k} \ldots \xrightarrow{R_j} O_{j+1}$, which defines an aggregation of relationships $R = R_1 \circ R_2 \circ \ldots \circ R_j$ between type O_1 and O_{j+1}, where $\circ$ denotes the composition operator on relationships and $O_i \in \mathcal{O}$ and $R_k \in \mathcal{R}$ are object and relation type, respectively ([28]).

***Example* 2.3.1.** MetaCyc contains multiple meta-paths conveying different semantics. For example, a meta-path "ZCZ" represents the co-catalyst relationships on a compound (C) between two enzymatic reactions (Z), and "ZCTCZ" may indicate a meta-path that requires two enzymatic reactions (Z) transforming two compounds (C) within a pathway (T). Another important meta-path to consider is "CZC", which implies "C + Z $\Rightarrow$ C" transformation relationship.

Problem Statement 1. *Metabolic Pathway Prediction* *Given three inputs: i)-a heterogeneous information network $\mathcal{G}$, ii)- a dataset $\mathcal{S}$, and iii)- an optional set of meta-paths $\mathcal{P}$, the goal is to automatically resolve node embeddings such that leveraging the features will effectively improve pathway prediction for a hitherto unseeen instance $\mathbf{x}^*$.*

3 The pathway2vec Framework

The pathway2vec framework is a package composed of five modules: i)- node2vec ([13]), ii)- metapath2vec ([9]), iii)- metapath2vec++ ([9]), iv)- JUST ([15]), and v)- RUST (this work), where each module contains a random walk modeling and node representation step. A graphical representation of the pathway2vec framework is depicted in Fig 2.

C1. Random Walks. In this step, a sequence of random walks over an input graph (whether heterogeneous or homogeneous) is generated based on the selected model. (see Section 3.1).

C2. Learning Node Representation. Resulting walks are fed into the Skip-Gram model to learn node embeddings ([23, 12, 13, 9]). An embedding is a low dimensional latent continuous feature for each node in $\mathcal{G}$, which encodes smooth decision boundaries between groups or communities within a graph. Details are provided in Section 3.2.

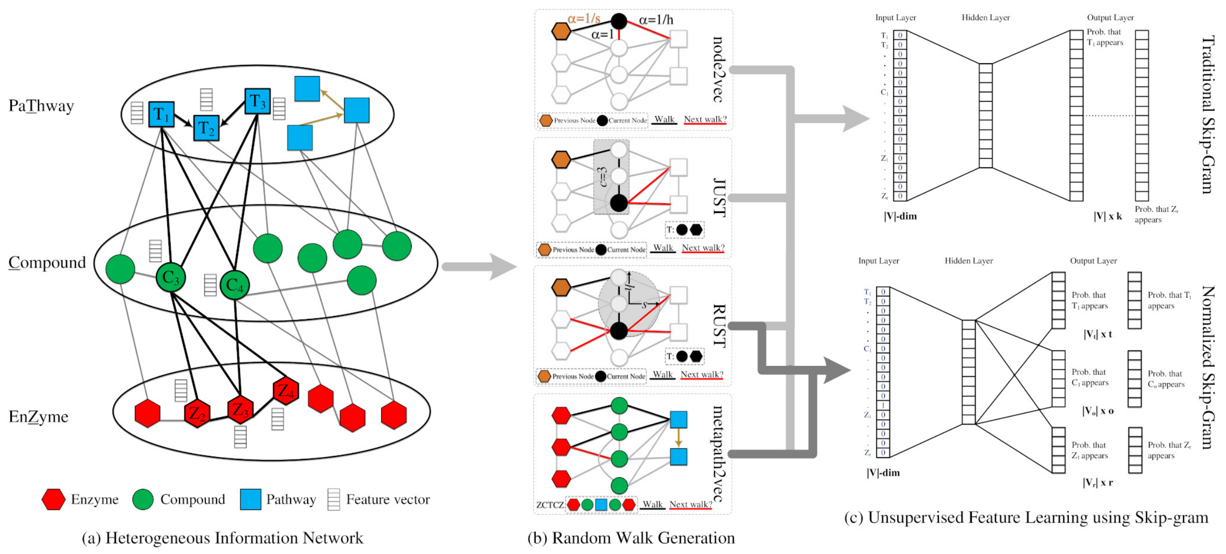


Figure 2: **Graphical representation of pathway2vec framework.** Main components: (a) a multi-layer heterogeneous information network of MetaCyc, showing meta-level interaction among compounds, enzymes, and pathways; (b) four random walks, and (c) two representational learning models: traditional Skip-Gram (top) and Skip-Gram by normalizing domain types (bottom). Highlighted network neighbors of T_1 (nitrifier denitrification) indicate this pathway interacts directly with T_2 (nitrogen fixation I (ferredoxin)) and indirectly to T_3 (nitrate reduction I (denitrification)) by second-order with relationships to several compounds, including nitric oxide (C_3) and nitrite (C_4) converted by enzymes represented by the EC numbers (Z_2: EC 1.7.2.6, Z_3: EC 1.7.2.1, and Z_4: EC 1.7.2.5).

3.1 Random Walks

To capture meaningful graph relationships, existing techniques such as DeepWalk ([25]), design simple but effective algorithms based on random walks for representational learning of features. The following is the definition of a first-order random walk as implemented in DeepWalk, extending a nodes immediate neighbors to include nodes that are locally connected.

Definition 3.1. Random Walk([25]). A random walk W of length l, rooted at node v, is a stochastic process with random variables $v^{i+1}, v^{i+2}, ..., v^{l}, v^{l+1}$ such that v^{j+1} is a vertex sampled at random from the neighbors of vertex v^j for all $1 \leq j \leq l$ according to the following distribution:

$$p(v^{j+1}|v^j) = \begin{cases} \frac{\alpha \pi_{j,j+1}}{Q} & \text{if } (v^j, v^{j+1}) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

where $\pi_{j,j+1} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ is an unnormalized transition probability, indicating the probability of a random walker visiting a node v^{j+1} conditioned on the current node being v^j, Q is a normalizing term, and $\alpha \in [0, 1]$ is a prior probability.

The above definition does not address in-depth and in-breadth graph exploration. node2vec ([13]) was developed to traverse local and global graph structures based on the principles of: i)- homophily ([24, 11]) where interconnected nodes form a community of correlated attributes and ii)- structural equivalence ([14]) where nodes having similar structural roles in a graph should be close to one another. node2vec simulates a second-order random walk, where the next node is sampled conditioned on the previous and the current node in a walk. For this, two hyper-parameters are adjusted, $s \in \mathbb{R}_{>0}$ that extracts local information of a graph, and $h \in \mathbb{R}_{>0}$ that enables local and global traversals by moving deep in a graph or walking within the vicinity of the current node. This method is illustrated in Fig 2 (b) top. The process of node2vec random walks can be defined by manipulating α in Definition 3.1 according to:

$$\alpha_{s,h}(j-1, j+1) = \begin{cases} \frac{1}{s} & \text{if } \beta_{j-1,j+1} = 0 \\ 1 & \text{if } \beta_{j-1,j+1} = 1 \\ \frac{1}{h} & \text{if } \beta_{j-1,j+1} = 2 \end{cases} \tag{3.2}$$

where $\beta_{j-1,j+1} \in \{0, 1, 2\}$ denotes the distance between the previously visited node v^{j-1} and the next neighbor node v^{j+1}.

First-order and second-order random walks were initially proposed for homogeneous graphs, but can be readily extended to heterogeneous information networks. Sun and colleagues have observed that random walks can suffer from implicit bias due to initial node selection or the presence of a small set of dominant node types skewing results toward a subset of interconnected nodes [28]. metapath2vec was developed [9], to resolve implicit bias in graph traversal to characterize the semantical associations embodied between different types of nodes according to a certain path definition. This method is illustrated in Fig 2 (b) bottom.

Definition 3.2. Meta-Path based Random Walk ([9]). Given a meta-path $P \in \mathcal{P}$ and $\mathcal{G}$, a meta-path based random walk W of length l, rooted at node $v_k \in O_k$ as dictated by P, is a stochastic process with random variables $v_1^{i+1}, v_2^{i+2}, ..., v_k^l, v_{k+1}^{l+1}$ such that v_{k+1}^{j+1} is a vertex of type O_{k+1}, as suggested by P, sampled randomly from the neighbors of vertex $v_k^j \in O_k$ for all $1 \leq j \leq l$ according to the following distribution:

$$p(v^{j+1}|v_k^j, P) = \begin{cases} \frac{1}{|\mathcal{N}_{k+1}(v_k^j)|} & \text{if } (v_k^j, v^{j+1}) \in \mathcal{E}, \phi(v^{j+1}) = k+1 \\ 0 & \text{if } (v_k^j, v^{j+1}) \in \mathcal{E}, \phi(v^{j+1}) \neq k+1 \\ 0 & \text{if } (v_k^j, v^{j+1}) \notin \mathcal{E} \end{cases} \tag{3.3}$$

where $v_k^j \in \mathcal{O}_k$ and $\mathcal{N}_{k+1}(v_k^j)$ denotes the neighbors of v_k^j that are of type $\mathcal{O}_{k+1}$ as pre-specified by the meta-path scheme P.

The use of meta-paths requires either prior domain-specific knowledge to recover the semantical knowledge of HIN according to a certain path definition. As a result, groups of vertices with the heterogeneous information network may not be visited or revisited multiple times. This limitation was partially addressed by leveraging multiple path schemes ([12]) to guide random walks based on a meta-path length parameter. Hussein and colleagues developed the Jump and Stay (JUST) heterogeneous graph embedding method using random walks [15] as an alternative to meta-paths. JUST randomly selects the next node in a walk from either the same node type or from different node types using an exponential decay function and a tuning parameter. This method is illustrated in Fig 2 (b) second from top.

Definition 3.3. Jump and Stay based Random Walk (JUST)([15]). Given a set of domain types $\mathcal{O}$, a graph $\mathcal{G}$, a queue T of size m, and an initial stay probability $\alpha \in [0, 1]$, a JUST based random walk W of length l, rooted at node $v_k \in O_k$, is a stochastic process with random variables $v^{i+1}, v^{i+2}, ..., v^l, v^{l+1}$ such that v^{j+1} is selected according to the two following consecutive steps:

1. Predict the *stay probability* p^{stay} as:

$$p^{\mathbf{st}}(v^j) = \begin{cases} 0 & \text{if } S(v^j) = \varnothing \\ 1 & \text{if } J(v^j) = \varnothing \\ \alpha^c & \text{if otherwise} \end{cases} \tag{3.4}$$

   where c is the number of nodes consecutively visited in the same domain as v^j and the remaining terms $S(v^j)$ and $J(v^j)$ are:

$$S(v^j) = \{v^{j+1}|(v^j, v^{j+1}) \in \mathcal{E} \wedge \phi(v^j) = \phi(v^{j+1})\}$$
$$J(v^j) = \{v^{j+1}|(v^j, v^{j+1}) \in \mathcal{E} \wedge \phi(v^j) \neq \phi(v^{j+1})\}$$

2. Sample v^{j+1} either: i)- from the same domain as v^j or ii)- apply the equation below, iff $p^{\mathbf{st}}(v^j) = 0$ or $p^{\mathbf{st}}(v^j) = 1 - \alpha^c$:

$$H(v^j) = \begin{cases} \{k|k \in \mathcal{O} \wedge k \notin T, J(v^j) \neq \varnothing\} & \text{if not empty} \\ \{k|k \in \mathcal{O}, k \neq \phi(v^j), J(v^j) \neq \varnothing\} & \text{if otherwise} \end{cases} \tag{3.5}$$

   where T is a queue of size m that stores m previously visited types.

If the set $S(v^j)$ is empty, meaning no edges exist between v^j and any nodes in $\mathcal{V}$ that share the same domain type as v^j, then a node is sampled from different types based on Eq. 3.5, which suggests to select randomly any domains not included in T. If, however, the latter condition is not satisfied then simply choose one domain that is different than the current node type. If $J(v^j)$ is empty, i.e., no heterogeneous edges connected to v^j, then the random walker is forced to stay in the same domain. Finally, if both homogeneous and heterogeneous edges are connected to v^j then the walker may choose to either stay with α^c or jump with $1-\alpha^c$, where α value decays exponentially by c that stores the number of nodes sequentially visited in the same type of v^j.

To balance the node distribution over multiple node types, the number of memorized domains m to store in T must be set within the range of $[1, |\mathcal{O}| - 1] \in \mathbb{Z}_{>1}$. This can misrepresent graph structure in two ways: i)- explorations within domain because the last visited consecutive c nodes may enforce sampling from another domain, or ii) jumping deep towards nodes from other domains because T is constrained. To alleviate these problems we develop a novel random walk algorithm, RUST, adopting a unit-circle equation to sample node pairs that generalize previous representational learning methods

Definition 3.4. Unit-Circle based Jump and Stay Random Walk (RUST). Given a set of domain types $\mathcal{O}$, a graph $\mathcal{G}$, a queue T of size m, and two hyper-parameters s and h, a RUST based random walk W of length l, rooted at node $v_k \in O_k$, is a stochastic process with random variables $v^{i+1}, v^{i+2}, ..., v^l, v^{l+1}$ such that v^{j+1} is chosen in two steps:

Database	#EC	#Compound	#Pathway	$\|\mathcal{V}\|$	$\|\mathcal{E}\|$
MetaCyc	6378	13689	2526	22593	37631
MetaCyc (r)	3606	6469	2467	12542	37631
MetaCyc (uec)	6378	13689	2526	22593	33353
MetaCyc (uec + r)	3229	6469	2467	12165	33353

Table 1: Different configurations of compound, enzyme and (EC) and pathway objects extracted from the MetaCyc database: i)- full content (MetaCyc), ii)- reduced content based in trimming nodes below 2 links (MetaCycy r), iii)- links among enzymatic reactions removed (MetaCyc uec)), and iv)- combination of unconnected enzymatic reactions and trimmed nodes (MetaCyc uec + r).

1. Estimate domain types transition probabilities given v^j:

$$\pi^{\mathbf{dom}}_{j,j+1} = \begin{cases} \frac{h.\beta_j \pi_{j-1,j}}{Q} & \text{if } S(v^j) = \varnothing \\ \frac{s.\beta_j \pi_{j-1,j}}{Q} & \text{if } J(v^j) = \varnothing \end{cases} \tag{3.6}$$

   where Q is a normalizing term, $\beta_j \in (0, 1]$ is a domain weight hyperparameter of v^j added to give more weights, if necessary, to some domains, and $\pi_{j-1,j}$ is an unnormalized transition probability from previous node v^{j-1} to the current node v^j. The remaining terms:

$$S(v^j) = \{v^{j+1} | (v^j, v^{j+1}) \in \mathcal{E} \wedge \phi(v^j) = \phi(v^{j+1})\}$$
$$J(v^j) = \{v^{j+1} | (v^j, v^{j+1}) \in \mathcal{E} \wedge \phi(v^j) \neq \phi(v^{j+1})\}$$

2. Sample a domain type k at random from $\pi^{\mathbf{dom}}_{j,j+1}$ according to:

$$H(v^j) = \{k | k \in \mathcal{O}, \alpha_k.\pi^k_{j,j+1}\} \tag{3.7}$$

   where $\alpha_k = 1/e^{c_k}$ and c_k is the number of nodes with type k that is stored in T. Finally, select randomly a node v^{j+1} based on $H(v^j)$.

The two hyper-parameters s and h constitute a unit circle, i.e., $h^2 + s^2 = 1$, where $h \in [0, 1]$ indicates how much exploration is needed within a domain while $s \in [0, 1]$ defines the in-depth search towards other domains such that $s > h$ encourages the walk to explore more domains and vice versa. Consequently, RUST blends both semantical knowledge and local/global structural information for generating walks. Note that the above definition does not place boundaries on the number of memorized domains m while the α_k serves as a function of node size having the type k as stored in T. Full details of RUST process to generate a set of walks $\mathcal{W}$ is illustrated in Algorithm 1. The method is illustrated in Fig 2 (b) second from bottom.

1. Estimate domain types transition probabilities given v^j:

$$\pi^{\mathbf{dom}}_{j,j+1} = \begin{cases} \frac{h.\beta_j \pi_{j-1,j}}{Q} & \text{if } S(v^j) = \varnothing \\ \frac{s.\beta_j \pi_{j-1,j}}{Q} & \text{if } J(v^j) = \varnothing \end{cases} \tag{3.8}$$

   where Q is a normalizing term, $\beta_j \in (0, 1]$ is a domain weight hyperparameter of v^j added to give more weights, if necessary, to some domains, and $\pi_{j-1,j}$ is unnormalized transition probability from previous node v^{j-1} to the current node v^j. The remaining terms:

$$S(v^j) = \{v^{j+1} | (v^j, v^{j+1}) \in \mathcal{E} \wedge \phi(v^j) = \phi(v^{j+1})\}$$
$$J(v^j) = \{v^{j+1} | (v^j, v^{j+1}) \in \mathcal{E} \wedge \phi(v^j) \neq \phi(v^{j+1})\}$$

2. Sample a domain type k at random from $\pi^{\mathbf{dom}}_{j,j+1}$ according to:

$$H(v^j) = \{k | k \in \mathcal{O}, \alpha_k.\pi^k_{j,j+1}\} \tag{3.9}$$

   where $\alpha_k = 1/e^{c_k}$ and c_k is the number of nodes with type k that is stored in T. Finally, select randomly a node v^{j+1} based on $H(v^j)$.

```
Inputs  : A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a prior probability $\alpha$, number of memorized domains $m$, explore hyperparameter
          $h$, in-out hyperparameter $s$, walk length $l$, number of random walks per node $K$
Outputs: A set of walks $\mathcal{W}$
Process :
1  Initialize a type transition probability $\pi^p$ over all nodes;
2  for $v \in \mathcal{V}$ do
3      for $i \leftarrow 1$ to $K$ do
4          walk=[$v$];
5          $T \leftarrow \varnothing$;
6          for $j \leftarrow 1$ to $l - 1$ do
7              $\pi^{\mathbf{dom}}_{j,j+1} \leftarrow$ by applying Eq. 3.8;
8              $H(v^j) \leftarrow$ by applying Eq. 3.9;
9              if $|T| = m$ then
10                 Update $\alpha_k = 1/e^{c_k}$;
11             Sample a $v^j$ from $H(v^j)$;
12             walk.append($v^j$)
13         Add walk to $\mathcal{W}$;
14 Return $\mathcal{W}$;
```

Algorithm 1: RUST based Random Walk

3.2 Learning Latent Embedding in Graph

Random walks $\mathcal{W}$ generated using node2vec, metapath2vec, JUST and RUST are fed into the Skip-Gram model to learn node embeddings ([23]). The Skip-Gram model exploits context information defined as a fixed number of nodes surrounding a target node. The model attempts to maximize co-occurrence probability among a pair of nodes identified within a given window of size q in $\mathcal{W}$ based on log-likelihood:

$$\sum_{l \in \mathcal{W}} \sum_{j \in l} \sum_{-q \leq k \leq q, j \neq 0} \log p(v^{j+k} | v^j) \tag{3.10}$$

where $v^{j-c}, ..., v^{j+c}$ are the context neighbor nodes of node v^j and $p(v^{j+i}|v^j)$ defines the conditional probability of having context nodes given the node v^j. The $p(v^{j+k}|v^j)$ is the commonly used softmax function, i.e, $= \frac{e^{\mathbf{D}_{v^{j+k}} \cdot \mathbf{D}_{v^j}}}{\sum_{i \in \mathcal{V}} e^{\mathbf{D}_{v^i} \cdot \mathbf{D}_{v^j}}}$, where $\mathbf{D} \in \mathbb{R}^{|V| \times d}$ stores the embeddings of all nodes and $\mathbf{D}_v$ is the v-th row corresponding to the embedding vector for node v. In practice, the vocabulary of nodes may be very large, which intensifies the computation of $p(v^{j+k}|v^j)$. The Skip-Gram model uses negative sampling, which randomly selects a small set of nodes N that are not in the context to reduce computational complexity. This idea, represented in updated Eq. 3.10 is implemented in node2vec, metapath2vec, JUST, and RUST according to:

$$\begin{aligned} &\sum_{l \in \mathcal{W}} \sum_{j \in l} \sum_{-q \leq k \leq q, j \neq 0} \Big(\log \sigma(\mathbf{D}_{v^{j+k}} . \mathbf{D}_{v^j}) \\ &\quad + \sum_{u \in N \wedge u \notin \mathcal{N}(j)} \mathbb{E}_{v^u} [\log p(v^u | v^j)] \Big) \end{aligned} \tag{3.11}$$

where $\sigma(v) = \frac{1}{1+e^{-v}}$ is the sigmoid function.

In addition to the equation above, Dong and colleagues proposed a normalized version of metapath2vec, called metapath2vec++, where the domain type of the context node is considered in calculating the probability $p(v^{j+k}|v^j)$, resulting in the following objective formula:

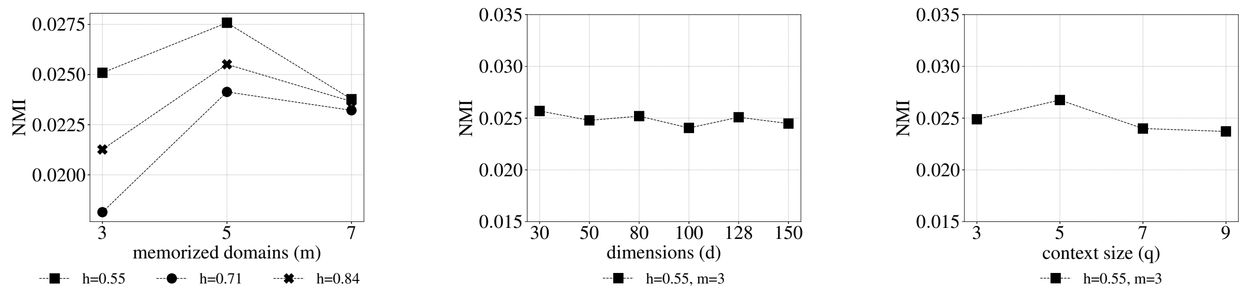
$$\begin{aligned} &\sum_{l \in \mathcal{W}} \sum_{j \in l} \sum_{-q \leq k \leq q, j \neq 0} \Big(\log \sigma(\mathbf{D}_{v^{j+k}} . \mathbf{D}_{v^j}) \\ &\quad + \sum_{u \in N \wedge u \notin \mathcal{N}(j) \wedge \phi(v^u) = \phi(v^{j+k})} \mathbb{E}_{v^u} [\log p(v^u | v^j)] \Big) \end{aligned} \tag{3.12}$$

where $\phi(v^u) = \phi(v^{j+k})$ suggests that the negative nodes are of the same type as the context node $\phi(v^{j+k})$. The above formula is also applied for RUST, and we refer it to RUST-norm. Through iterative update over all the context nodes, whether using Eq. 3.11 or Eq. 3.12, for each walk in $\mathcal{W}$, the learned features are expected to capture semantic and structural contents of a graph, thereby, can be made useful for pathway inference.

4 Predicting Pathways

For pathway inference, the learned EC embedding vectors are concatenated into each example i according to:

$$\tilde{\mathbf{x}}^{(i)} = \mathbf{x}^{(i)} \oplus \frac{1}{r} \mathbf{x}^{(i)} \mathbf{D}_{v: v \in \mathcal{Z}} \tag{4.1}$$



(a) Number of memorized domains vs different h value

(b) Number of dimensions

(c) Neighborhood size

Figure 3: Parameter sensitivity of RUST based on NMI metric.

where $\oplus$ denotes the vector concatenation operation, $\mathbf{D} \in \mathbb{R}^{|V| \times d}$ stores the embeddings of all nodes and $\mathbf{D}_{v:v \in \mathcal{Z}}$ indicates feature vectors for r enzymatic reactions. By incorporating enzymatic reaction features into $\mathbf{x}^{(i)}$, the dimension size is extended to $r + d$, where r is the enzyme vector size while d corresponds the embeddings size. This modified version of $\mathbf{x}^{(i)}$ is denoted by $\tilde{\mathbf{x}}^{(i)}$, which then can be used by an appropriate machine learning algorithm, such as mlLGPR ([4]), to train and infer a set of metabolic pathways from genomic sequence information.

5 Experimental Setup and Results

In this section, we first investigate the impact of hyperparameters involved in RUST (Section 5.1), then we benchmark the four random walk algorithms, jointly with the two learning methods, based on node-clustering (Section 5.1), embedding visualization (Section 5.3) and pathway prediction (Section 5.4) using MetaCyc version 21 ([6]). With regard to the sensitivity analysis for the other random walk methods, we refer the readers to the previously established works in ([13, 9, 15]). We apply different configurations of MetaCyc, as summarized in Table 1, to examine the effects of various graph type in the quality of generated walks and embeddings. The package pathway2vec is entirely implemented in Python and trained using tensorflow version 1.10. For training, we randomly initialize model parameters with a truncated Gaussian distribution, and set the learning rate to 0.01, the batch size to 100, and the number of epochs to 10. Unless otherwise indicated, for each module, the number of sampled path instances is $K = 100$, the walk length is $l = 100$, the embedding dimension size is $d = 128$, the neighborhood size is 5, the size of negative samples is 5, and the number of memorized domain m for JUST and RUST are 2 and 3, respectively. The explore and the in-out hyperparameters for node2vec and RUST are $h = 0.7$ (or $h = 0.55$) and $s = 0.7$ (or $s = 0.84$), respectively, using the uec configuration. For metapath2vec and metapath2vec++, we applied the meta-path scheme "ZCTCZ" to guide random walks. All the experiments are conducted on a Linux server using 10 Intel Xeon CPU E5-2650 cores. For brevity, we denote node2vec, metapath2vec, metapath2vec++, JUST, RUST, and RUST-norm as n2v, m2v, cm2v, jt, and rt, crt, respectively.

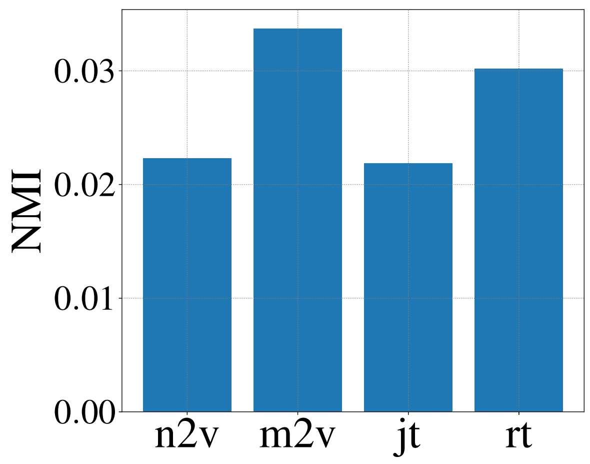
5.1 Parameter Sensitivity of RUST

Experimental setup. In this section, we study the effect of different hyperparameter settings in RUST on the quality of learned nodes embeddings. Since the hyperparameter space, involved in RUST, is infinite, exhaustive searching for the best settings is intractable. Hence, we follow the procedures that are previously reported and sub-select the settings under which we examine RUST's performances. Specifically, we select the influences of the dimensions $d \in \{30, 50, 80, 100, 128, 150\}$, the neighborhood size $q \in \{3, 5, 7, 9\}$, the memorized domains $m \in \{3, 5, 7\}$, and the two hyperparameters s and h ($\in \{0.55, 0.71, 0.84\}$) according to the Normalized Mutual Information (NMI) scores, after 10 trials. The NMI produces scores between 0, indicating no mutual information exists for grouping (or clustering) nodes, and 1, suggesting the features are well characterized into groups that are based on the class information: enzyme, compound, and pathway. The clustering is performed using the k-means algorithm ([3]), similar to works in ([9, 15]), to group data based on the learned representations from RUST. Using the aforementioned variation in RUST settings, random walks $\mathcal{W}$ are generated using MetaCyc with uec option.
Experimental results. From Fig 3a, we observe that performance tends to saturate when the memorized domains are concentrated around $m = 5$ and $h = 0.55$, indicating the preference of RUST to explore more domain types, as reported by ([15]). By fixing $m = 3$ and $h = 0.55$, we observe that the optimal results of NMI score w.r.t. the number of embedding dimensionality is achieved at $d = 128$, as depicted in Fig 3b. The performance of RUST, in general, improves with the increase of dimensions until 128, suggesting that RUST is able to capture the complex network structure/semantics when $d = 128$, after which the performance

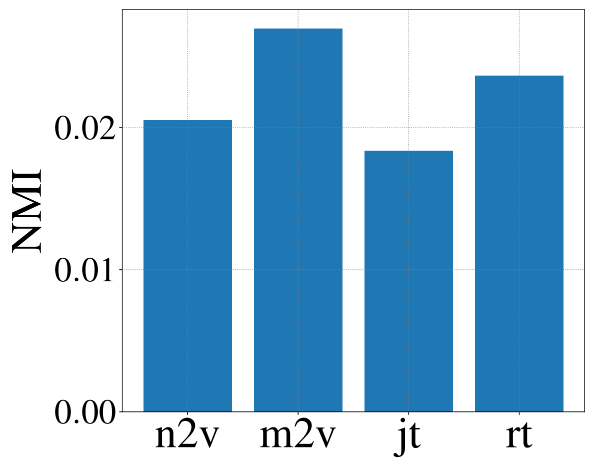
starts to degrade. A similar trend is also observed in Fig 3c, when the context neighborhood size increases beyond $q > 5$. Taking together, we suggest that the following settings $m = 3$, $h = 0.55$, $d = 128$, and $q = 5$ provide the RUST's promising clustering outcome on MetaCyc with uec option while not degrading the overall sampling budget.

5.2 Node Clustering

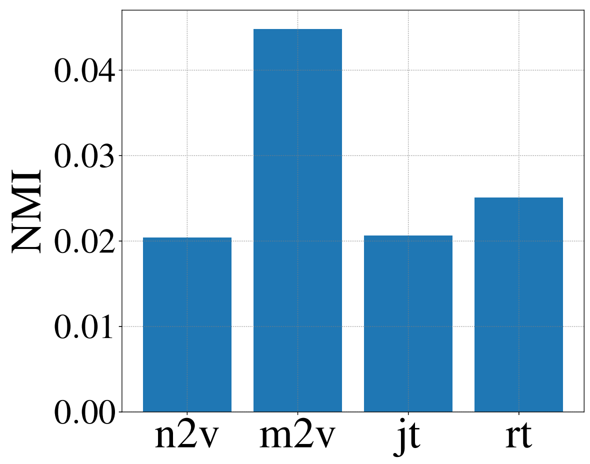
Experimental setup. The objective of this experiment is to evaluate the performance of random walk methods based on the clustering quality using NMI, as a performance indicator, on all MetaCyc graph types in Table 1. For clustering, we apply the k-means algorithm, as before, to group homogeneous nodes based on the embeddings learned by each methods. The NMI scores are reported based on 10 repeated trials using hyperparameters introduced in the beginning of Section 5.



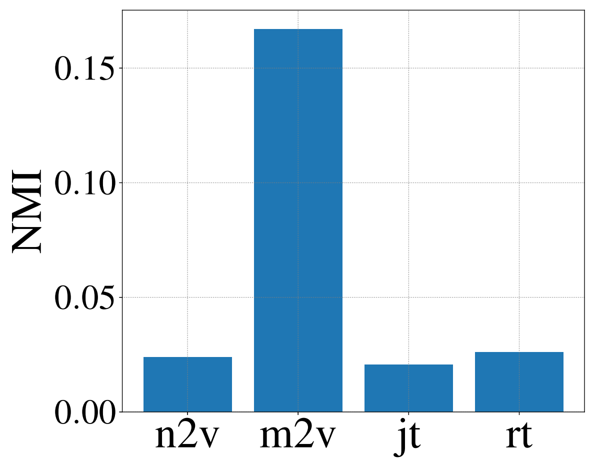
(a) MetaCyc



(b) MetaCycy r



(c) MetaCyc uec



(d) MetaCyc uec + r

Figure 4: Node clustering results based on NMI metric using MetaCyc data.

Experimental results. Fig 4 indicates clustering results using the configurations described in Table 1. Overall, node2vec, JUST, and RUST exhibit robust performances across all configurations, indicating these methods are less likely to extract semantic knowledge, characterizing node domains, from MetaCyc. However, the RUST method performing optimally better than node2vec and JUST in learning representations.

For metapath2vec, the random walk follows a predefined meta-path scheme, hence, capturing the necessary relational knowledge for defining node types. For example, *nitrogenase* (EC-1.18.6.1), which reduces *nitrogen* gas into *ammonium*, is exclusively linked to the *nitrogen fixation I (ferredoxin)* pathway ([10]). And without a predefined relation, a walker may explore more local/global structure of $\mathcal{G}$, hence, become less efficient in exploiting relations between those two nodes. Among the four walks, only metapath2vec is able to accurately group those nodes, according to their classes. However, despite the advantages of metapath2vec in those cases, it is biased to a scheme, as elucidated in [15], which is explicitly observed for the case of "uec+r" (Fig 4d). In such circumstances, both isolated nodes and links among ECs are discarded, resulting in a reduced number of nodes. Consequently, it is easier for meta-bath based walker to traverse and exploit specific relations.

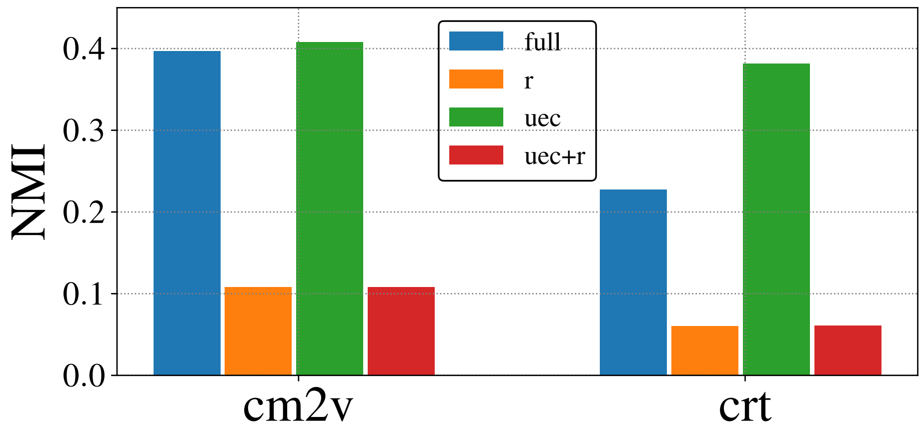


Figure 5: Node clustering results based on NMI metric using MetaCyc data.

With regard to metapath2vec++ method, it exhibits similar trends as metapath2vec because they share the same walks, however, the former method is trained using normalized Skip-Gram, thereby, it is expected to achieve good NMI scores, yielding over 0.41 on uec+full content in Fig. 5, which is also similar to RUST-norm NMI score (~ 0.38). This is interesting because RUST-norm employs RUST based walks but the embeddings are learned using normalized Skip-Gram, as metapath2vec++, suggesting that RUST is capable to exploit the structural and semantics (with a lesser extent as discussed previously). In fact, the NMI score is observed to be similar to metapath2vec, ~ 0.039 (for uec), for clustering embeddings obtained from RUST using $m = 2$ and $h < 0.1$.

To summarize, these tests indicate that node2vec, JUST, and RUST based walks are good for analyzing graph structure while metapath2vec can learn good embeddings. However, RUST strike a balance between the two proprieties through proper adjustments of m and the two unit-circle hyperparameters. Regarding the MetaCyc type, we recommend "uec" because the associations among ECs are captured at the pathway level. Also, the trimmed graph is not suggested, as it will eliminate many isolated, and important, pathways and ECs.

5.3 Manifold Visualization

Experimental setup. In this section, we visualize the learned high dimensional embeddings by projecting them onto a two-dimensional space under two case studies. The first case examines the quality of learned nodes embeddings according to the generated random walks, where we speculate that a good representation learning method creates clear distinct boundaries for nodes of the same type. This approach is commonly sought in most graph learning embedding techniques ([13, 31]). While any group of nodes can be employed in such case study, but for illustration purposes, we pick nodes corresponding the nitrogen metabolism. In the second case study, we investigate the impact of meta-path based random walks, extending our discussions in Section 5.2. For this, we solely focus at the pathway layer in Fig 2a and analysis the representation of pathways having no enzymatic reactions. For visualization, we use UMAP, a.k.a. uniform manifold approximation and projection ([21]) that is implemented in python ([22]) using 1000 epochs while the remaining setting are fixed to their default values.
Experimental results. Fig 6 illustrates the visualization outputs of 185 nodes, corresponding the nitrogen metabolism extracted from MetaCyc. Each point denotes a node in HIN and each color indicates the type of the node. Clearly, the methods node2vec (Fig 6a), JUST (Fig. 6c), and RUST (Fig. 6d) are not appealing in extracting walks that preserve three layers relational knowledge, where we see that nodes belonging to different types forming unclear boundaries and diffuse clusters. For the metapath2vec (Fig. 6b), metapath2vec++ (Fig. 6f), and RUST-norm (Fig. 6f), we observe the nodes of the same color are decently portrayed. Although this suggests that the meta-path based walks is suitable to exploit semantics, nonetheless, it does not highlight some limitations associated with the generated walks.

To address the above concern, we conducted the second case study, where we identified 80 pathways, having no enzymatic reactions, with their 109 pathway neighbors, as shown in Fig. 7a. From Fig. 7, we observe that, in contrast to node2vec, JUST, RUST, and RUST-norm, pathway nodes are skewed in both metapath2vec++ and metapath2vec (with lesser degree), which are incorrect as 80 pathway nodes are immediate neighbors to 109 pathways. This demonstrates the rigidness of meta-path based methods where they strictly follow a certain schemes, thereby, unable to adsorb local structure in learning embeddings.

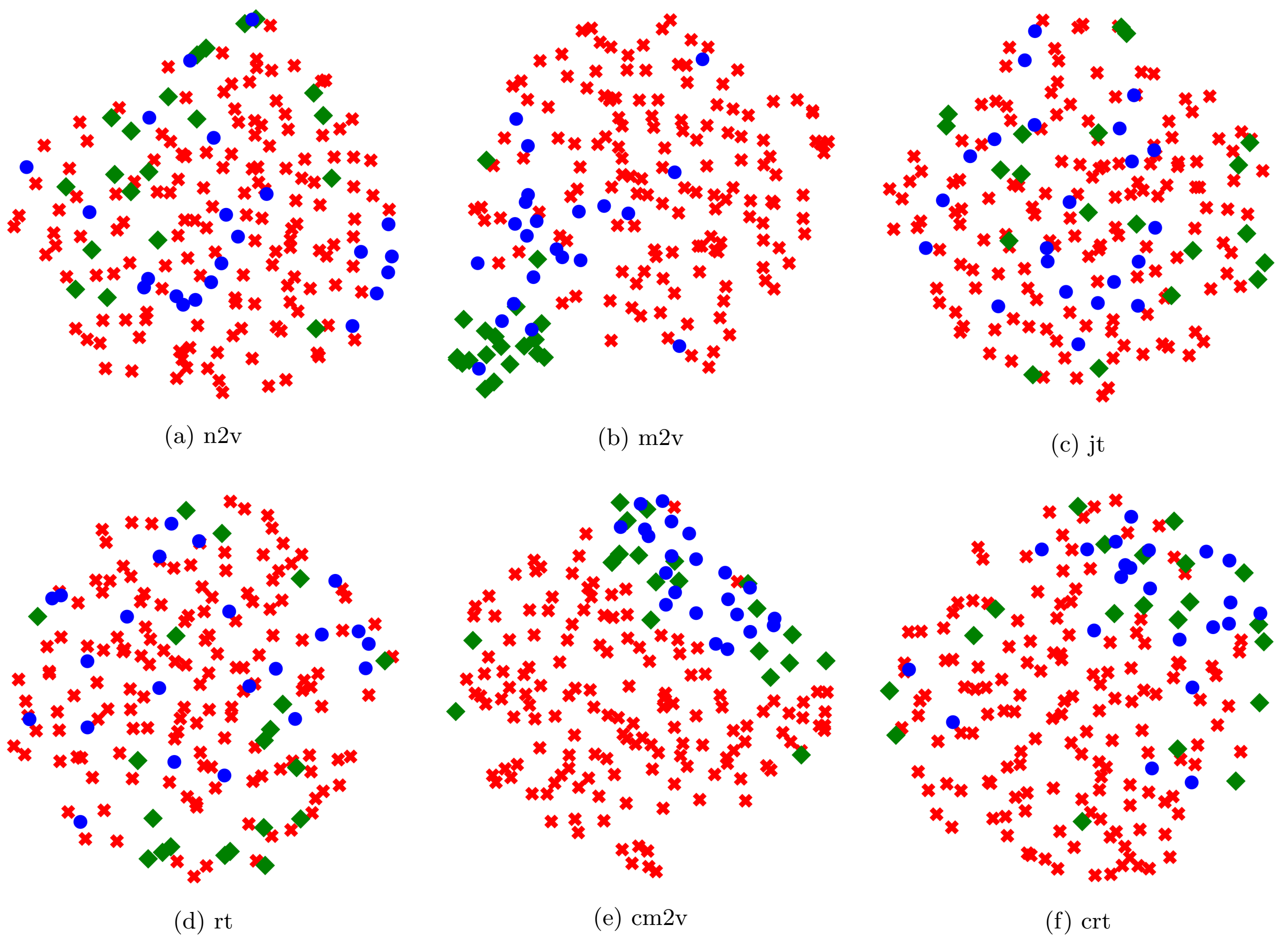


Figure 6: 2D UMAP projections of the 128 dimension embeddings, trained under uec+full setting, of selected 185 nodes, corresponding nitrogen compound metabolism. Color of a node indicates the category of the node type, where the red indicates the enzymatic reactions, the green indicates the compounds, and the blue is reserved for the metabolic pathways.

Interestingly, RUST-norm, based on RUST walks, is the only method that combines the structural knowledge, as demonstrated in Fig. 7g, with the semantic information, in Fig. 6f. Therefore, we suggest RUST based walks with training using Eq. 3.12 for obtaining efficient embeddings, which is ironically, the same conclusion in Section 5.2.

5.4 Metabolic Pathway Prediction

Experimental setup. The objective of this test is to examine the effectiveness of the learned embeddings from pathway2vec modules for the pathway inference task. Diverting from the previous approaches in multi-label classification ([25, 13, 15]), where the goal is to predict the most probable label set for nodes, we leverage the learned vectors to the multi-label dataset, according to Eq. 4.1. We use mlLGPR (elastic net) ([4]), using the default hyperparameter settings, after concatenating features from each learning method, to train on BioCyc (v20.5 tier 2 & 3) ([7]) that consists of 9255 PGDBs (Pathway/Genome Databases) with 1463 distinct pathway labels (see Supplementary Material). Then, we report results on tier 1 benchmark datasets, comprising of EcoCyc, HumanCyc, AraCyc, YeastCyc, LeishCyc, and TrypanoCyc. Four evaluation metrics are used to report the final scores after 3 repeated trials: *Hamming loss*, *micro precision*, *micro recall*, and *micro F1 score*. We contrast the performance of each learning method against three pathway prediction algorithms, namely PathoLogic v21 ([18]), MinPath v1.2 ([32]), and mlLGPR (elastic net with enzymatic reaction and pathway evidence features) ([4]).
Experimental results. Table 2 shows micro F1 scores of the prediction algorithms. Numbers in boldface represent the best performance score in each column while the underlined text indicates the best performance among the embedding methods. From the results, it is obvious that all variation of embedding methods performs consistently better than MinPath on four datasets. In fact, on EcoCyc, the mlLGPR+crt achieved a considerable improvement over all predictors (a micro F1-score of 0.7682. In other cases, the performances of embeddings are less competitive to PathoLogic and mlLGPR. This is because the mlLGPR with embeddings are trained on less than 1470 pathways, hence, obscuring the actual benefits of the learned features. With regard to the modules of pathway2vec, they perform on par to one another, albeit mlLGPR+rt and

Methods	Hamming Loss ↓					
	EcoCyc	HumanCyc	AraCyc	YeastCyc	LeishCyc	TrypanoCyc
PathoLogic	0.0610	**0.0633**	0.1188	**0.0424**	**0.0368**	**0.0424**
MinPath	0.2257	0.2530	0.3266	0.2482	0.1615	0.2561
mlLGPR	0.0804	**0.0633**	**0.1069**	0.0550	0.0380	0.0590
mlLGPR+n2v	0.0558	0.1021	0.1706	0.0768	0.0424	0.0883
mlLGPR+m2v	0.0558	0.0998	0.1742	0.0740	0.0412	0.0926
mlLGPR+cm2v	0.0586	0.1041	0.1742	0.0744	0.0420	0.0867
mlLGPR+jt	0.0550	0.1041	0.1738	0.0724	0.0459	0.0895
mlLGPR+rt	0.0554	0.0990	0.1746	0.0752	0.0428	0.0855
mlLGPR+crt	**0.0542**	0.1017	0.1615	0.0760	0.0439	0.0855

Methods	Micro Precision Score ↑					
	EcoCyc	HumanCyc	AraCyc	YeastCyc	LeishCyc	TrypanoCyc
PathoLogic	**0.7230**	**0.6695**	0.7011	**0.7194**	**0.4803**	**0.5480**
MinPath	0.3490	0.3004	0.3806	0.2675	0.1758	0.2129
mlLGPR	0.6187	0.6686	0.7372	0.6480	0.4731	0.5455
mlLGPR+n2v	0.7923	0.5745	0.6965	0.6446	0.4153	0.3974
mlLGPR+m2v	0.7862	0.6015	0.6786	0.6750	0.4261	0.3745
mlLGPR+cm2v	0.7770	0.5556	0.6620	0.6723	0.4159	0.4076
mlLGPR+jt	0.7979	0.5556	0.6732	0.6949	0.3840	0.3924
mlLGPR+rt	0.7889	0.6014	0.6635	0.6560	0.4146	0.4113
mlLGPR+crt	**0.7993**	0.5873	**0.7898**	0.6581	0.3983	0.4105

Methods	Micro Recall Score ↑					
	EcoCyc	HumanCyc	AraCyc	YeastCyc	LeishCyc	TrypanoCyc
PathoLogic	0.8078	0.8423	0.7176	0.8734	0.8391	0.7829
MinPath	**0.9902**	**0.9713**	**0.9843**	**1.0000**	**1.0000**	**1.0000**
mlLGPR	0.8827	0.8459	0.7314	0.8603	0.9080	0.8914
mlLGPR+n2v	0.7329	0.2903	0.2745	0.3406	0.5632	0.5314
mlLGPR+m2v	0.7427	0.2867	0.2608	0.3537	0.5632	0.5029
mlLGPR+cm2v	0.7264	0.2867	0.2804	0.3493	0.5402	0.5543
mlLGPR+jt	0.7329	0.2867	0.2706	0.3581	0.5517	0.5314
mlLGPR+rt	0.7427	0.3082	0.2745	0.3581	0.5862	0.5429
mlLGPR+crt	0.7394	0.2652	0.2725	0.3362	0.5402	0.5371

Methods	Micro F1 Score ↑					
	EcoCyc	HumanCyc	AraCyc	YeastCyc	LeishCyc	TrypanoCyc
PathoLogic	0.7631	0.7460	0.7093	**0.7890**	0.6109	0.6447
MinPath	0.5161	0.4589	0.5489	0.4221	0.2990	0.3511
mlLGPR	0.7275	**0.7468**	**0.7343**	0.7392	**0.6220**	**0.6768**
mlLGPR+n2v	0.7614	0.3857	0.3938	0.4457	0.4780	0.4548
mlLGPR+m2v	0.7638	0.3883	0.3768	0.4642	0.4851	0.4293
mlLGPR+cm2v	0.7508	0.3783	0.3939	0.4598	0.4700	0.4697
mlLGPR+jt	0.7640	0.3783	0.3860	0.4726	0.4528	0.4515
mlLGPR+rt	0.7651	0.4076	0.3883	0.4633	0.4857	0.4680
mlLGPR+crt	**0.7682**	0.3654	0.4052	0.4451	0.4585	0.4653

Table 2: **Predictive performance of each comparing algorithm on 6 benchmark datasets.** For each performance metric, '↓' indicates the smaller score is better while '↑' indicates the higher score is better.

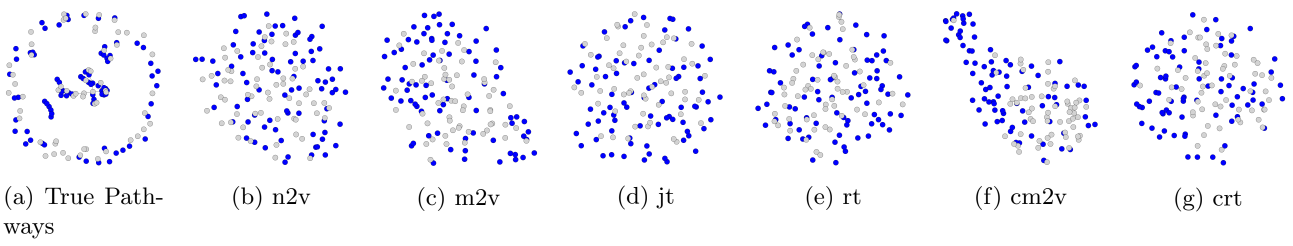


Figure 7: 2D UMAP projections of 80 pathways that have no enzymatic reactions, indicated by the blue color, with their 109 pathway neighbors, represented by the grey color.

mlLGPR+crt have slight edges in the performance gain. Regardless, this experiment showcases that the embeddings are a strong contender to the pathway and reaction evidence features, as employed in ([4]).

6 Conclusion

Automatically generating features (i.e., non hand-crafted) enables us to discover insightful latent patterns from the sheer volume of data, without requiring any prior knowledge. In this paper, we presented the pathway2vec package for learning features, for the first time, after manipulating the metabolic pathway, obtained from MetaCyc, as a multi-layer heterogeneous network. The package contains four modules with two learning strategy, each of which serves different purposes and applications. Three extensive empirical studies were conducted, which shown that the learned representations has potential to be a better replacement for hand-engineered rules and well suited for many tasks, especially, the pathway prediction task, thereby, allowing us to consider this approach in the future work. We also presented our novel method RUST that comes with unit-circle and domain size hyperparameters, making it flexible in capturing local/global structure while absorbing semantics from a given homogeneous/heterogeneous graph, as demonstrated by empirical studies.

There are several directions for the future study. We are currently in the process to build a more robust model, taking into the consideration graph techniques while leveraging embeddings learned from any modules in pathway2vec, in particular RUST-norm (or metapath2vec++), to enhance pathway prediction. Another important application is to extend the pathway prediction using embeddings to the community level of organization, akin to the procedure followed in ([4]), to make it useful for a wide range of dataset types.

Acknowledgments

We would like to thank Connor Morgan-Lang, Julia Glinos, Kishori Konwar and Aria Hahn for lucid discussions on the function of the pathway2vec framework, Ryan MacLaughlin for his participation in preliminary performance evaluations and all members of the Hallam Lab for helpful comments along the way.
Funding: This work was performed under the auspices Genome Canada, Genome British Columbia, the Natural Science and Engineering Research Council (NSERC) of Canada, and Compute/Calcul Canada). ARB was supported by a UBC four-year doctoral fellowship (4YF) administered through the UBC Graduate Program in Bioinformatics.
Conflict of Interest: none declared.

References

[1] Sahar Abubucker, Nicola Segata, Johannes Goll, Alyxandria M Schubert, Jacques Izard, Brandi L Cantarel, Beltran Rodriguez-Mueller, Jeremy Zucker, Mathangi Thiagarajan, Bernard Henrissat, et al. Metabolic reconstruction for metagenomic data and its application to the human microbiome. *PLoS Comput Biol*, 8(6):e1002358, 2012.

[2] Wilhelm J Ansorge. Next-generation dna sequencing techniques. *New biotechnology*, 25(4):195–203, 2009.

[3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] Abdur Rahman MA Basher, Ryan J McLaughlin, and Steven J Hallam. Metabolic pathway inference using multi-label classification with rich pathway features. *bioRxiv*, 2020.

[5] Pablo Carbonell, Jerry Wong, Neil Swainston, Eriko Takano, Nicholas J Turner, Nigel S Scrutton, Douglas B Kell, Rainer Breitling, and Jean-Loup Faulon. Selenzyme: Enzyme selection tool for pathway design. *Bioinformatics*, 34(12):2153–2154, 2018.

[6] Ron Caspi, Richard Billington, Luciana Ferrer, Hartmut Foerster, Carol A. Fulcher, Ingrid M. Keseler, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Lukas A. Mueller, Quang Ong, Suzanne Paley, Pallavi Subhraveti, Daniel S. Weaver, and Peter D. Karp. The metacyc database of metabolic pathways and enzymes and the biocyc collection of pathway/genome databases. *Nucleic Acids Research*, 44(D1):D471–D480, 2016.

[7] Ron Caspi, Richard Billington, Hartmut Foerster, Carol A Fulcher, Ingrid Keseler, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Lukas A Mueller, Quang Ong, et al. Biocyc: Online resource for genome and metabolic pathway analysis. *The FASEB Journal*, 30(1 Supplement):lb192–lb192, 2016.

[8] Joseph M Dale, Liviu Popescu, and Peter D Karp. Machine learning methods for metabolic pathway prediction. *BMC bioinformatics*, 11(1):1, 2010.

[9] 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. ACM, 2017.

[10] Robert R Eady. Structure- function relationships of alternative nitrogenases. *Chemical reviews*, 96(7):3013–3030, 1996.

[11] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

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

[13] 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. ACM, 2016.

[14] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1231–1239. ACM, 2012.

[15] 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. ACM, 2018.

[16] Dazhi Jiao, Yuzhen Ye, and Haixu Tang. Probabilistic inference of biochemical reactions in microbial communities from metagenomic sequences. *PLoS Comput Biol*, 9(3):e1002981, 2013.

[17] Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Kanae Morishima. Kegg: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, 45(D1):D353–D361, 2017.

[18] Peter D Karp, Mario Latendresse, Suzanne M Paley, Markus Krummenacker, Quang D Ong, Richard Billington, Anamika Kothari, Daniel Weaver, Thomas Lee, Pallavi Subhraveti, et al. Pathway tools version 19.0 update: software for pathway/genome informatics and systems biology. *Briefings in bioinformatics*, 17(5):877–890, 2016.

[19] Peter D Karp, Wai Kit Ong, Suzanne Paley, Richard Billington, Ron Caspi, Carol Fulcher, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Peter E Midford, et al. The ecocyc database. *EcoSal Plus*, 8(1), 2018.

[20] Christopher E Lawson, William R Harcombe, Roland Hatzenpichler, Stephen R Lindemann, Frank E Löffler, Michelle A O'Malley, Héctor García Martín, Brian F Pfleger, Lutgarde Raskin, Ophelia S Venturelli, et al. Common principles and best practices for engineering microbiomes. *Nature Reviews Microbiology*, pp. 1–17, 2019.

[21] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[22] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[24] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[25] 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. ACM, 2014.

[26] Mahdi Shafiei, Katherine A Dunn, Hugh Chipman, Hong Gu, and Joseph P Bielawski. Biomenet: A bayesian model for inference of metabolic divergence among microbial communities. *PLoS Comput Biol*, 10(11):e1003918, 2014.

[27] Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2017.

[28] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11), 2011.

[29] Yasuo Tabei, Yoshihiro Yamanishi, and Masaaki Kotera. Simultaneous prediction of enzyme orthologs from chemical transformation patterns for de novo metabolic pathway reconstruction. *Bioinformatics*, 32(12):i278–i287, 2016.

[30] David Toubiana, Rami Puzis, Lingling Wen, Noga Sikron, Assylay Kurmanbayeva, Aigerim Soltabayeva, Maria del Mar Rubio Wilhelmi, Nir Sade, Aaron Fait, Moshe Sagi, et al. Combined network analysis and machine learning allows the prediction of metabolic pathways from tomato metabolomics data. *Communications Biology*, 2(1):214, 2019.

[31] 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. ACM, 2016.

[32] Yuzhen Ye and Thomas G Doak. A parsimony approach to biological pathway reconstruction/inference for genomes and metagenomes. *PLoS Comput Biol*, 5(8):e1000465, 2009.