Representation Learning for Classification in Heterogeneous Graphs with Application to Social Networks

LUDOVIC DOS SANTOS, BENJAMIN PIWOWARSKI, LUDOVIC DENOYER, and PATRICK GALLINARI, CNRS, Sorbonne Université, LIP6 UMR 7606

We address the task of node classification in heterogeneous networks, where the nodes are of different types, each type having its own set of labels, and the relations between nodes may also be of different types. A typical example is provided by social networks where node types may for example be users, content, or films, and relations *friendship*, *like*, *authorship*. Learning and performing inference on such heterogeneous networks is a recent task requiring new models and algorithms. We propose a model, *Labeling Heterogeneous Network (LaHNet)*, a transductive approach to classification that learns to project the different types of nodes into a common latent space. This embedding is learned so as to reflect different characteristics of the problem such as the correlation between node labels, as well as the graph topology. The application focus is on social graphs, but the algorithm is general and can be used for other domains. The model is evaluated on five datasets representative of different instances of social data.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Learning latent representations**; *Statistical relational learning*;

Additional Key Words and Phrases: Representation learning, node classification, relational data

ACM Reference format:
Ludovic Dos Santos, Benjamin Piwowarski, Ludovic Denoyer, and Patrick Gallinari. 2018. Representation Learning for Classification in Heterogeneous Graphs with Application to Social Networks. *ACM Trans. Knowl. Discov. Data.* 12, 5, Article 62 (July 2018), 33 pages.
https://doi.org/10.1145/3201603

1 INTRODUCTION

Social networks (Facebook, Google+, ...) and social media (Twitter, LastFM, ...) may be naturally represented as heterogeneous graphs, where different types of nodes (users, authors, films, ...) interact through different types of relations (friendship, like, authorship, ...). For example, the LastFM social network used in our experiments (Figure 1) links users, tracks, artists, and albums via seven different types of relations such as *friendship*, *most listened tracks*, *authorship*, and so on. Different generic data mining tasks can be conducted on heterogeneous networks such as classification [2], clustering [39], link prediction [11], as well as tasks more specific of social media, such as information diffusion [8] or influence analysis [20, 21].

This work was supported in part by a Grant from the Research Agency ANR (Agence Nationale de la Recherche) under the MLVIS project.
Authors' addresses: L. D. Santos, B. Piwowarski, L. Denoyer, and P. Gallinari, Sorbonne Universite, Univ Paris 06, CNRS, LIP6 UMR 7606, 4 Place Jussieu 75005 Paris, France; emails: {ludovic.dos-santos, benjamin.piwowarski, ludovic.denoyer, patrick.gallinari}@lip6.fr.

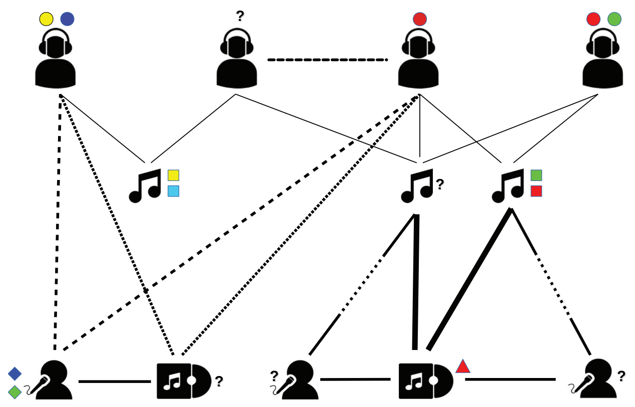


Fig. 1. Representation of the LastFM network with users, tracks, albums, and artists (top to bottom). The different relations are represented with different types of lines.

We consider here the generic task of multilabel node classification, in the general case of heterogeneous graphs with multiple node and relation types. Graph node classification has been investigated in machine learning since the early 2000s. Most work until now has focused on homogeneous graphs, where nodes are of the same type, share the same label set, and where relations reflect a unique type of dependency between nodes.

The main assumption in most homogeneous graph classification approaches is that *two connected nodes tend to have the same labels*, so that graph labeling can be performed using some form of label propagation from labeled nodes to their unlabeled neighbors [1, 6, 49]. Work in this domain can be grouped in two main families : transduction/diffusion models and induction (feature-based) models. Transduction models use all the nodes (labeled and unlabeled) to predict the labels of the entire graph, in contrast to induction models, where classifiers are trained only on labeled nodes.

Extending the ideas used for homogeneous graphs to heterogeneous ones is not trivial. For example, in transductive approaches, the label propagation paradigm becomes irrelevant when neighbors are of different types, and hence have different sets of labels. Different attempts have tried to reformulate the problem in order to use known homogeneous methods. The simplest idea developed by several authors is to project the heterogeneous graph onto a family of homogeneous ones (one node type, one relation type) and then perform classification independently for each projection. This basic approach suffers from several drawbacks (i) defining semantically meaningful projections is problem specific, (ii) the number of projections can rapidly become extremely large, and (iii) as shown in our experiments, correlations between different types of nodes may play an important role in the labeling task, and are lost after the projection. Another simple alternative consists in using restrictive assumptions so that homogeneous label propagation (HLP) can be used. The authors in [17, 19], for example, consider heterogeneous graphs where the label set is the same for all node types. Only some works have tackled the general heterogeneous node labeling problem with new algorithms, such as [2] in which the authors defined new forms of random walks on heterogeneous graphs specifically adapted to this problem.

While all the above methods have been designed for operating directly on the discrete graph structure, we follow here a different path by using a representation learning approach [5]. We propose to map the classification problem onto a continuous space of latent node representations. This representation space is common to all node types, hence allowing to handle heterogeneity. Node embedding in this space reflects the classification objective for each type of node and its associated label set, as well as the heterogeneous relational structure of the graph. Classification is

Table 1. Graph Notations

General	
$\#S$	Cardinality of a set S
Graph	
$\mathcal{N}$	Set of nodes of the graph
$\mathcal{E}$	Set of edges of the graph
n_i	Node of the graph
$\mathcal{N}_j$	Set of neighbors of n_j
$\mathcal{T}$	Set of node types
$\mathcal{R}$	Set of relation types
$t_i \in \mathcal{T}$	Type of node n_i
$r_{ij} \in \mathcal{R}$	Directed relation type between node n_i and n_j
Labels	
$\mathcal{N}_L$	Set of labeled nodes
$\mathcal{L}$	Set of labels
$\mathcal{L}_t$	Set of labels associated with nodes of type t
$y_{i\ell}$	$+1$ if n_i has label ℓ, -1 otherwise

then performed directly on the latent space. This idea has been introduced in a preliminary work [18]. The current article extends this work in several ways: we introduce prior parameters based on the graph characteristics, we enrich the model with *learned* hyperparameters weighting the relative importance of the different relation types, and, finally, we perform extensive experimental evaluations on social datasets representative of different situations.

Summarizing, our contributions are the following:

— We address the heterogeneous graph multilabel classification problem as a representation learning problem in a continuous latent space and consider a transductive approach to graph node classification.
— We propose new algorithms for learning this representation space able to handle heterogeneity for both nodes and relations.
— We develop an experimental analysis of the algorithm behavior.
— We perform extensive experimentations with five datasets, representative of different heterogeneous social graphs, and compare our model with state of the art models.

The article is organized as follows. Section 2 introduces the notations. Section 3 describes the model and the learning algorithms. Section 4 presents experimental results and comparisons with baselines on five datasets. Section 5 is a synthesis of related work on graph labeling models and graph representation learning. Section 6 concludes this work and gives some perspectives.

2 NOTATIONS

Let us first introduce the notations used throughout this article—they are summarized in Table 1.

An heterogeneous network is modeled as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{E}$ the set of edges. We denote n_i a node of this graph, $\mathcal{T}$ the set of node types, and $t_i \in \mathcal{T}$ the type of node n_i. We denote by $\mathcal{R}$ the set of possible relation types, and $\mathcal{E}^r$ the set of edges of type $r \in \mathcal{R}$. When there is an edge between n_i and n_j, we denote $r_{ij} \in \mathcal{R}$ the type of the relation between n_i and n_j. We consider that there is at most one relation between two nodes. Note that this is not restrictive, since the model can easily be used with multigraphs, but this simplifies the description and notations.

Table 2. Model Notations

d	Dimension of the latent space
$\mathcal{N}_C$	Set of labeled nodes used for learning the classifiers f_{θ_ℓ} and the labeled node representations z
$\mathcal{N}_W$	Set of labeled nodes used to learn the relation weights w_r
z_i	Latent representation ($z_i \in \mathbb{R}^d$) of node n_i
$\mathbf{z}$	Latent representation set for all nodes $\mathbf{z} = \{z_i \| i \in \mathcal{N}\}$
w_r	Weight of the relation type r
$w_{r_{ij}}$	Weight of the relation type r_{ij}, $w_{r_{ij}} \in \mathbb{R}$
$\mathbf{w}$	Set of weights $\mathbf{w} = \{w_r \| r \in \mathcal{R}\}$
ψ_i	Prior parameter reflecting the relative importance of node n_i, $\psi_i \in \mathbb{R}^+$
ϕ_{ij}	Prior parameter reflecting the relative importance of relation between nodes n_i and n_j, $\phi_{ij} \in \mathbb{R}^+$
θ_ℓ	Parameters for the classifier for label ℓ
$\mathbf{\Theta}$	All the classifiers parameters: $\mathbf{\Theta} = \{\theta_\ell \| \ell \in \mathcal{L}\}$

With respect to node labels, $\mathcal{L}_t$ denotes the set of categories associated with nodes of type t, and $\mathcal{N}_L \subset \mathcal{N}$ is the set of labeled nodes. For $i \in \mathcal{N}_L$, $y_{i\ell}$ is the class indicator associated with n_i and label ℓ: if node n_i is labeled by ℓ, $y_{i\ell} = 1$ and if not $y_{i\ell} = -1$.

Finally, we denote by $\#S$ the cardinality of a set S.

3 MODEL

3.1 Model Description

Our model, *Labeling Heterogeneous Network* (LaHNet), has been designed for transductive node classification in heterogeneous graphs. The objective is to learn node representations so that each type of node can be correctly classified while exploiting the correlations between the labels of different node types as well as the graph structure. These interdependencies between nodes of different types play an important role in several classification problems, and none of the current methods is able to correctly handle them. The variables to be learned are as follows:

(1) the latent representations $z_i \in \mathbb{R}^d$ (for each node n_i),
(2) the relation weights w_r (for each relation type r),
(3) and the classifiers parameters θ_ℓ, where ℓ indicates a given label.

The notations for the model description and parameters are summarized in Table 2.

3.1.1 Loss Function. The loss function takes the general form of a transductive regularized loss [19, 49], with a classification term L_C and a regularization term L_G:

$$L(\mathbf{z}, \mathbf{\Theta}) = \underbrace{\sum_{i \in \mathcal{N}_C} \sum_{\ell \in \mathcal{L}_{t_i}} \frac{\psi_i}{\#\mathcal{L}_{t_i}} \Delta_C(f(z_i;\theta_\ell), y_{i\ell})}_{L_C} + \lambda \underbrace{\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}_i} w_{r_{ij}} \phi_{ij} \Delta_G(z_i, z_j)}_{L_G}, \tag{1}$$

where

— Δ_C is the classification loss (for one node and one label), which is optimized on a subset of labeled nodes $\mathcal{N}_C \subset \mathcal{N}_L$;

— Δ_G is the graph regularization loss (for a pair of connected nodes), which is optimized over all the edges present in the graph;
— $f(.;\theta_\ell)$ is a parametric classifier with parameters θ_ℓ (there is one set of parameters for each $\ell \in \mathcal{L}$);
— ϕ_{ij}, ψ_i and $w_{r_{ij}} \geqslant 0$ are real parameters that will be discussed later (Section 3.2);
— $\lambda \in \mathbb{R}$ is the regularization weight.

Within a classical transductive homogeneous graph formulation [1, 19, 49, 49, 50], each z_i can be interpreted as a vector of label probabilities, and the regularization term operates as a diffusion equation for propagating labels from labeled nodes to their unlabeled neighbors. In our model, the z_is have a different role: they are latent node representations in $\mathbb{R}^d$. The second term L_G in Equation (1) forces neighbors to have close latent representations z whatever their type is. The terms L_C and L_G are optimized simultaneously w.r.t. parameters z_i for each node n_i, and w.r.t. θ_ℓ, for each node label $\ell \in \mathcal{L}$.

Let us now detail the components of the loss presented in Equation (1).

3.1.2 Classifier. The mapping onto the latent space is learned so that the labels for each type of node can be predicted from the latent node representation. For that, for each label $\ell \in \mathcal{L}$, we use a classifier parameterized by θ_ℓ, denoted by $f(.;\theta_\ell)$. For a node n_i and a label $\ell \in \mathcal{L}_{t_i}$, this classification function takes as input the node representation z_i and outputs a classification score $f(z_i;\theta_\ell)$.

The classifiers parameters θ_ℓ, for $\ell \in \mathcal{L}$, are learned by minimizing the first term in Equation (1) for the labeled dataset $\mathcal{N}_C$, that is,

$$L_C = \sum_{i \in \mathcal{N}_C} \sum_{\ell \in \mathcal{L}_{t_i}} \frac{\psi_i}{\#\mathcal{L}_{t_i}} \Delta_C(f(z_i;\theta_\ell), y_{i\ell}), \tag{2}$$

where $f(z_i;\theta_\ell)$ and $y_{i\ell}$ are, respectively, the classification score and target associated with node n_i and label ℓ. $\Delta_C(f(z_i;\theta_\ell), y_{i\ell})$ is the loss of predicting a classification score $f(z_i;\theta_\ell)$, while the target is $y_{i\ell}$. The hyperparameter ψ_i represents the node i prior importance, and is defined in Section 3.2.

In the experiments, we used a standard hinge-loss function for Δ_C:

$$\Delta_C(f(z;\theta_\ell), y_\ell) = \max(0, 1 - y_\ell f(z;\theta_\ell)), \tag{3}$$

where y_ℓ is equal to 1 if n is labeled by ℓ and -1 otherwise, and $f(z;\theta_\ell)$ is the predicted score of label ℓ given that the node representation is z. In our experiments, f is a linear classifier, i.e.,

$$f(z;\theta) = z \cdot \theta.$$

Other classifiers, e.g., multilayer neural networks, could have been used. In our experiments, we have observed that using a simpler classifier led to easier convergence, as well as comparable or better results.

3.1.3 Transductive Graph Model. The second term in Equation (1) exploits the proximity of the nodes in the graph. More precisely, nodes linked in the graph are encouraged to have close representations in the latent space. We used an L2 term $\Delta_G(z_i, z_j) = ||z_i - z_j||^2$ for this loss:

$$L_G = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}_i} w_{r_{ij}} \phi_{ij} ||z_i - z_j||^2, \tag{4}$$

where $\mathcal{N}_i$ denotes the graph neighbors of node i, $w_{r_{ij}} \in \mathbb{R}^+$ is a learned weight for the relation of type r_{ij} (Section 3.2.2), and $\phi_{ij} \in \mathbb{R}$ represents a prior on the relative importance of the relation between n_i and n_j (Section 3.2.1).

Ensuring that connected nodes are close in the latent space, first implies that connected nodes of *the same type* will be classified similarly. This is a common hypothesis of all transductive models for node classification. Even if nodes are not directly connected, but are a few hops away, the graph regularization will push the representations, and hence the labels of indirectly connected nodes to be the same. This is beneficial for classification as shown in other models such as Graffiti [2].

Finally, ensuring that neighbors of different types are close in the representation space also allows one to exploit correlations between labels of nodes with different types and to better handle the classification problem in heterogeneous graphs.

To illustrate why this might be the case, suppose that we have two types of nodes, A and B, and that, when a node of type A is labeled ℓ_1, this is highly correlated with the fact that a neighbor of type B is labeled ℓ_2. Let us denote z_A and z_B the representations of two neighbors from types A and B.

The regularization term L_G will push z_A and z_B close one to the other. If their labels are strongly correlated, so is the classification term Δ_C. This will lead the corresponding classifiers $f(.;\theta_{\ell_1})$ and $f(.;\theta_{\ell_2})$ to be aligned (see also Section 4.7 where this is confirmed experimentally). Both terms in the loss function have the same action so that the two nodes (z_A and z_B) and classifiers representations (θ_{ℓ_1} and θ_{ℓ_2}) will be very close. An unlabeled node of type say B in the neighborhood of the node corresponding to z_A, will then be attracted toward z_A and then toward z_B thanks to the label correlation between A and B. Label correlation is hence captured by the model, and helps improving inference. If the labels of the two nodes are not correlated, the two components of the loss term may act in an opposite way on z_A and z_B, and nodes should not influence each other. This is also handled by the relation specific weight which is learned—see Sections 3.2.2 (theoretical development) and 4.6 (experimental observations).

3.2 Prior Parameters and Learning Algorithms

The loss function in Equation (1) makes use of three graph related parameters, namely ϕ_{ij}, ψ_i, and w_r. Introducing these three families of hyperparameters allowed us to significantly improve the model performance. The ϕ_{ij} and ψ_i are priors taking into account local graph characteristics, while the w_r are learned relation-specific weights reflecting the relative importance of the different relation types $r \in \mathcal{R}$ for the classification task. The definition and computation of the three hyperparameters is described as follows.

3.2.1 Prior Parameters. Let us suppose the ϕ_{ij}, the ψ_i, and the w_r parameters be set to 1 in loss function (1). When optimizing this loss, all nodes have an equal importance (term L_C) and similarly, all the edges have the same importance (term L_G). This might not be optimal, and, during preliminary experiments, we explored various settings for the hyperparameters ψ_i and ϕ_{ij}. We report here the most reliable ones.

First, wrong decision for important nodes should be over-penalized. In the regularization loss L_G, this is the role of the coefficient ψ_i associated with node i. For ψ_i, we found out that a node importance should be increased when (1) it has links with other nodes; (2) those links are of different types. Formally, we found that the following formula behaved well:

$$\psi_i = \frac{1}{\#\mathcal{R}} \sum_{r \in \mathcal{R}} \frac{\#\mathcal{N}_i^r}{\#\mathcal{E}^r},$$

with $\#\mathcal{N}_i^r$ the number of neighbors of i for relation r and $\#\mathcal{E}^r$ the total number of relations of type r in the graph. The coefficient ψ_i is high if node i has many relations of different types.

Besides node priors, some relation types between nodes might be important for the classification objective, but if they are under-represented in the graph, they will be ignored. We found out that it was important to weight them up, which is the role played by the prior coefficient ϕ_{ij}. More precisely, we found that the following formulation adapted from [2] was the most efficient among several heuristics:

$$\phi_{ij} = \frac{1}{\#\mathcal{R}\#\mathcal{N}_j^{r_{ij}}\#\mathcal{E}^{r_{ij}}},$$

with $\#\mathcal{N}_j^{r_{ij}}$ the number of neighbors of j for which the relationship is r_{ij}. The $\#\mathcal{E}^{r_{ij}}$ coefficient gives a similar importance to all relation types: it increases the weight of under-represented relation types so that in (4) their influence will be balanced with the one of more frequent relations. The $\#\mathcal{N}_j^{r_{ij}}$ coefficient gives more importance to neighbors j for which the relationship r_{ij} is less frequent.

3.2.2 Learned Relation Specific Parameters. The graph regularization coefficients w_r are relationship-specific. They reflect the importance of relation r for the inference task. For example, if inference consists in classifying an author research domain, then the authorship relation between authors and their published papers is probably more important than their affiliation relationship. For the model, this means that authors' representations should be close to their papers representations, whereas no such constraint operates on the affiliation representation. The w_r are hyperparameters that could be learned by grid search and cross-validation. Since there might be several relation types for a heterogeneous graph, and hence a high number of potential values to experiment with, this is not a relevant option here. We used instead the framework of continuous optimization of hyperparameters [4, 23].

This framework has been developed for learning regularization hyperparameters. Given a regularized loss such as (1), hyperparameters are learned along with the model parameters, by optimizing the unregularized loss on a distinct training set $\mathcal{N}_W$.

Contrarily to grid search, which also selects regularization hyperparameters using an unregularized loss on a validation set, by testing different preset hyperparameters values, here parameters and hyperparameters are learned simultaneously and their dynamics are intertwined. There is no formal proof that such procedures converge to an optimal choice of the hyperparameters, but they offer an approximate solution, which performs well in many cases (see [23] for a discussion).

There have been several instances of this general framework, and we derive below our own version for the specific problem handled here. Our inference problem is classification, and hence, following [4, 23], our loss function for hyperparameter training is the classification objective denoted L_W. This loss L_W is defined on $\mathcal{N}_W$, a set of labeled nodes distinct from the labeled set $\mathcal{N}_C$ used in Equation (1), i.e., we ensure that $\mathcal{N}_W \cap \mathcal{N}_C = \emptyset$.

The loss we optimize with respect to the weights $\mathbf{w} = \{w_r\}$ is similar to L_C but defined on a different set of labeled nodes:

$$L_W(\mathbf{w}) = \sum_{i \in \mathcal{N}_W} \sum_{\ell \in \mathcal{L}_{t_i}} \frac{\psi_i}{\#\mathcal{L}_{t_i}} \Delta_C(f(z_i(\mathbf{w}); \theta_\ell(\mathbf{w})), y_{i\ell}). \quad (5)$$

The proposed learning scheme uses an alternating optimization algorithm:

(1) learning the parameters $\{z_i\}$, $\{\theta_\ell\}$ by optimizing the loss $L_C + \lambda L_G$ of Equation (1), with the $\{w_r\}$ fixed;
(2) learning the hyperparameters $\{w_r\}$ by optimizing the loss of Equation (5).

At each iteration of this process, the values of Θ and $\mathbf{z}$ after step 1 are dependent on the current $\mathbf{w}$: more precisely, $\Theta(\mathbf{w})$ and $\mathbf{z}(\mathbf{w})$ are the parameters that minimize the loss of Equation (1). This dependency is emphasized in Equation (5) by the notations $z_i(\mathbf{w})$ and $\theta_\ell(\mathbf{w})$.

Minimizing (5) is performed by gradient descent. In Appendix A, we derive a closed form for $\frac{\partial L_W}{\partial w_r}$, that relies on a series of reasonable assumptions about how the classifier and embeddings change with respect to a change of the hyperparameter w_r. We show that the derivative of the unregularized loss L_W with respect to a weight w_r is approximated by

$$\frac{\partial L_W}{\partial w_r} \approx \sum_{i \in \mathcal{N}_W} \sum_{\ell \in \mathcal{L}_{t_i}} \underbrace{\psi_i y_{i\ell} \frac{\sum_{r' \neq r} w_{r'} A_i^{rr'} \theta_\ell \cdot \left(S_i^{r'} - S_i^r\right)}{\left(\sum_{r' \in \mathcal{R}} w_{r'} A_i^{rr'}\right)^2}}_{D_i^{r\ell}}, \tag{6}$$

with

$$S_i^r = \frac{\sum_{j \in \mathcal{N}_i^r} \phi_{ij} z_j}{\sum_{j \in \mathcal{N}_i^r} \phi_{ij}} \quad \text{and} \quad A_i^{rr'} = \frac{\sum_{j \in \mathcal{N}_i^{r'}} \phi_{ij}}{\sum_{j \in \mathcal{N}_i^r} \phi_{ij}}. \tag{7}$$

Equation (6) measures the variation of the empirical risk (5) when w_r changes. To develop some intuition of this equation, let us suppose that all the ϕ and ψ coefficients are equal to 1: they are all positive anyway, and the main change would be in the relative importance of the type of neighbors depending on the set hyperparameters ϕ_{ij} and ψ_i.

We consider the inner term in (6), i.e., term $D_i^{r\ell}$, and, more precisely, its sign that determines whether we should increase or decrease the weight w_r. Without loss of generality, let us consider the case of a target label $y_{i\ell} = 1$. The sign of $D_i^{r\ell}$ is then determined by

$$\sum_{r' \neq r} w_{r'} A_i^{rr'} \theta_\ell \cdot \left(S_i^{r'} - S_i^r\right),$$

the expression $w_{r'} A_i^{rr'}$ is positive, since both of its components are positive. The sign of $\frac{\partial L_W}{\partial w_r}$ hence depends on $\theta_\ell \cdot (S_i^{r'} - S_i^r)$. In this expression, S_i^r is the average of the z_j vectors for all j r-neighbors (neighbors according to relation r) of node i. Hence, $(\theta_\ell \cdot S_i^r)$ is the score for label ℓ of the center of mass of the r-neighbors of node i. Similarly, $\theta_\ell \cdot S_i^{r'}$ is the score for label ℓ of the center of mass of the r'-neighbors of node n_i.

If the center of mass of the r'-neighbors has a higher score that the center of mass of the r-neighbors, the difference $\theta_\ell \cdot (S_i^{r'} - S_i^r)$ will be negative, leading to an increase of w_r—and a decrease if the contrary holds. Summarizing, the weight of relation r will be increased if the center of mass of the r-neighbors is better located (w.r.t. the classification task) than the center of mass of the other neighbors, i.e., if it reinforces the correct classification, and will be decreased otherwise.

3.3 Algorithm

Algorithm 1 describes the alternating optimization scheme for learning the θs, zs, and ws.

Step 1. Learning θ and z corresponds to lines 1–18. One samples a pair of connected nodes i and j, and then makes a gradient update of the parameters for loss (1). A node in $\mathcal{N}_C$ appears in the two terms of Equation (1). The update w.r.t. the first term is indicated lines 8–9 for node i and 13–14 for node j and consists in successively modifying the parameters of the classification function θ and of the latent representations z_i and z_j so as to minimize the classification loss. For all the nodes, be they in $\mathcal{N}_C$ or not, the model updates the parameters w.r.t the graph loss (second term of Equation (1))—lines 16–17. These different steps are repeated up to a fixed number of iterations.

Step 2. Using as training set $\mathcal{N}_W$, one then learns the ws by gradient descent on L_W. We compute the derivatives w.r.t. the relations by summing over all the nodes in $\mathcal{N}_W$—line

ALGORITHM 1: Alternating Optimization Algorithm

```
Input: $z$ node representations, $w$ weights, $\epsilon$ gradient step, $\lambda$ tradeoff, $\mathcal{N}_C$,$\mathcal{N}_W$  $\ell$ the number of
       iterations;
Output: Learned $z$ and $w$;
for $k$ iterations do
    for #$\mathcal{N}$ iterations do
        ▷ Learn the representations $\{z_i\}$ and the classifier parameters $\Theta$;
        Choose $r$ in $\mathcal{R}$ at random;
        Pick randomly an edge $i \overset{r}{\rightarrow} j$;
        if $i \in \mathcal{N}_C$ then
            for $\ell \in \mathcal{L}_i$ do
                $\theta \leftarrow \theta - \epsilon \nabla_\theta \Delta(f(z_i;\theta_\ell), y_{i\ell})$ ;
                $z_i \leftarrow z_i - \epsilon \nabla_{z_i} \Delta(f(z_i;\theta_\ell), y_{i\ell})$ ;
            end
        end
        if $j \in \mathcal{N}_C$ then
            for $\ell \in \mathcal{L}_j$ do
                $\theta \leftarrow \theta - \epsilon \nabla_\theta \Delta(f(z_j;\theta_\ell), y_{j\ell})$;
                $z_j \leftarrow z_j - \epsilon \nabla_{z_j} \Delta(f(z_j;\theta_\ell), y_{j\ell})$;
            end
        end
        $z_i \leftarrow z_i - \epsilon \frac{w_{ij}}{N_j^{r_{ij}}} \lambda \nabla_{z_i} ||z_i - z_j||^2$;
        $z_j \leftarrow z_j - \epsilon \frac{w_{ji}}{N_i^{r_{ji}}} \lambda \nabla_{z_j} ||z_i - z_j||^2$ ;
    end
    Let grad_w = $[0, \ldots, 0]$                          ▷ Learn the $w_r$;
    for $i \in \mathcal{N}_W$ do
        for $\ell \in \mathcal{L}_{t_i}$ do
            if $1 - y_{i\ell}(\theta_\ell | z_i^\star) > 0$ then
                for $r \in \mathcal{R}$ do
                    grad_w[r] +=$D_i^{r\ell}$            ▷ See Equation (6) ;
                end
            end
        end
    end
    for $r$ in $\mathcal{R}$ do
        $w_r \leftarrow w_r - \epsilon \times$ grad_w[r]
    end
    Normalize $w$ w.r.t. relations
end
```

26—according to Equation (6). Then, we update w_r—line 32. Finally, we normalize w_r—line 34—(Equation (13)). This alternating scheme is iterated for a fixed number of iterations.

For step 1, we have been using a stochastic gradient. For Step 2, because of normalization (13), we used a batch gradient.

Note that in Algorithm 1, ϕ_i and ψ_{ij} do not appear explicitly. They are implicitly computed through the following sampling procedure: we first sample uniformly the type of edge (line 4), then choose uniformly an edge given that type (line 5). This sampling corresponds to optimizing Equation (4) with hyperparameters

$$\phi_i = \frac{1}{\#\mathcal{R}} \sum_{r \in \mathcal{R}} \frac{\#\mathcal{N}_i^r}{\#\mathcal{E}^r},$$

for the classification term and

$$\psi_{ij} = \frac{1}{\#\mathcal{R}\#\mathcal{N}_j^{r_{ij}}\#\mathcal{E}^{r_{ij}}},$$

for the graph regularization one.

Regarding the numerical complexity, we can see that the main loop has a fixed number of iterations (k), while the inner loops are composed of (step 1) $\#\mathcal{N}$ iterations (one iteration per node), each of complexity d (since we have a linear classifier), and (step 2) $\#\mathcal{N}_W$ iterations (where $\mathcal{N}_W$ is the training set which is used to learn the hyperameters), where each computation is of complexity $O(\bar{N}d)$, with $\bar{N}$ the average number of neighbors for a node. Overall, the complexity is $O(kd(\#\mathcal{N} + \bar{N}))$. In practice, we were able to deal easily with huge datasets like IMDB (see Section 4)—with k being set to a maximum of 1,000 iterations (in practice, convergence happened much earlier).

While there is no formal proof for the algorithm convergence, we observed that node representations, classifiers, and weights w_r converged after a few hundred iterations (k) of the algorithm. Intuitively, we can see that a minimum of the loss L of Equation (1) corresponds to an equilibrium between the regularization loss L_G, that pushes all the nodes to a single point, and the classification loss L_C, that pushes the nodes toward the regions corresponding to the correct label classification.

In Algorithm 1, ϵ is the gradient step, and λ is the tradeoff between the classification and smoothness terms. Both parameters were set by grid search.

4 EXPERIMENTS

4.1 Datasets

Experiments have been performed on five datasets respectively extracted from DBLP, FlickR, LastFM, and IMDB. For all but the first dataset (DBLP), each node can have multiple labels. The datasets are described below. Statistics for the datasets are provided in Table 3.

The *DBLP* dataset [1] [39] is a bibliographic network composed of authors and papers. We consider here two different sets of labels: authors are labeled with their research domain (4 labels), while papers are labeled with the conference name they were published in (20 labels). Authors and papers are connected through an *authorship* relation. The network is then composed of two types of nodes and is bipartite, with one relation type. Classification is monolabel on papers and authors.

The *FlickR* corpus is composed of photos and users. The photo labels correspond to different possible tags while the user labels correspond to their subscribed groups. The classification problem is multilabel: images and users may belong to more than one category. Photos are related to users through an *authorship* relation, while users are related to other users through a *following* relation. We have kept the image tags that appear in at least 500 images, and user categories that also appear at least 500 times in the dataset resulting in 21 possible labels for photos and 42 for authors.

We used two different *LastFM* datasets denoted *LastFM1* and *LastFM2*, collected independently using the LastFM API.[2] Both datasets are social networks composed of users, tracks, albums, and

[1]The dataset is available at http://web.cs.ucla.edu/yzsun/data.

[2]To access the API go to http://www.lastfm.fr/api. The detailed procedure on how we collected these datasets is provided in Appendix B.

Table 3. Statistics for the Datasets

		Type	No. nodes	No.labeled nodes	No. labels
DBLP	Nodes	Paper	14,376	14,376	20
		Author	14,475	4,057	4
	Edges	Type	No. edges		
		Author↔Paper	41,794		
FlickR	Nodes	Photo	46,926	8,766	21
		User	4,760	3,476	42
	Edges	User↔User	175,779		
		User↔Photo	46,926		
LastFM1	Nodes	User	1,013	321	59
		Track	35,181	24,562	28
		Album	32,118	15,966	47
		Artist	17,138	11,564	47
	Edges	User↔User	1,109		
		User↔Album	47,541		
		User↔Artist	47,812		
		User↔Track	47,807		
		Track↔Album	29,647		
		Track↔Artist	35,181		
		Album↔Artist	32,118		
LastFM2	Nodes	User	20,004	5,000	48
		Track	340,356	177,297	32
		Album	247,681	59,226	52
		Artist	103,951	60,388	59
	Edges	User↔User	23,573		
		User↔Album	951,274		
		User↔Artist	960,227		
		User↔Track	970,381		
		Track↔Album	162,364		
		Track↔Artist	340,356		
		Album↔Artist	247,681		
IMdB	Nodes	Film	2,724,246	1,050,323	28
		Actor	3,146,496	0	0
		Director	362,470	0	0
	Edges	Film↔Actor	14,308,748		
		Film↔Actress	8,526,214		
		Film↔Director	2,055,234		

artists. The task is multilabel classification and all the node types have their own set of labels. Users are labeled with the type of music they like (59/48 labels, respectively, for *LastFM1* and *LastFM2*, e.g., *female vocalists*, *ambient*, ...), tracks with the kind of music they belong to (28/32 labels, e.g., *rock*, *indie*, ...), albums with their type (47/52 labels, e.g., *various artists*, *live*, ...), and artists with the kind of music they play the most (47/59 labels, e.g., *folk*, *singer songwriter*, ...). Users are linked to users (*friendship*), tracks (*favorite tracks*), albums (*favorite albums*), and artists (*favorite artists*). Tracks are linked to albums (*belong to*) and artists (*singer*). Finally, albums are linked to artists (*sing in*). Note that both a track and an album may be linked to several artists. The *LastFM* graph is schematically represented in Figure 1. Some labels may overlap between different types of nodes but we suppose that in this case, they are distinct, e.g., *pop* is not the same for an artist or a track.

Finally, the *IMDB* dataset[3] is a movie description dataset composed of actors, actresses, films, and directors. Only films are labeled with 28 possible labels corresponding to their type (Documentary, Drama, Comedy, ...). Films are related to actors and actresses using a *casting* relationship and to directors using a *film maker* relationship.

4.2 Evidence for Heterogeneous Node Reciprocal Influence

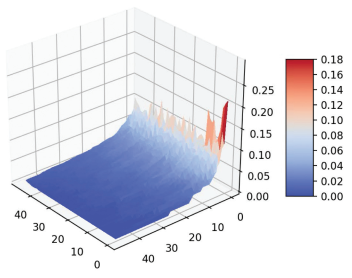
Throughout this work, we make the assumption that correlations exist between the labels of different types of nodes, and that modeling this influence is useful for the heterogeneous classification task. We exhibit below some statistics and characteristics of the datasets to support this hypothesis.

In the following, we denote L_x the random variable corresponding to the label of a node of type x. We consider the conditional probability $P(L_y = l_y | L_x = l_x)$ that a node of type y has label l_y if a neighbor of type x has label l_x. To simplify the analysis, we make the hypothesis that $P(L_x = x)$ was a uniform probability distribution over the labels.

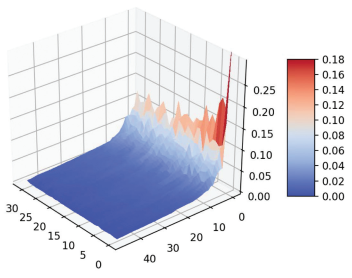
In Figure 2, we have plotted some typical conditional distributions between couples of variables. In order to make the graphics more readable, we have ordered labels l_x by increasing conditional entropy[4] $H(L_y | L_x = l_x)$ along the x-axis, for all the possible values l_x. The conditional entropy quantifies the amount of information needed to describe the outcome of L_y given L_x, so that the most informative L_x values are near the x-axis origin. For each value l_x of L_x, i.e., for each x coordinate, we further order the L_y values according to decreasing values of $P(L_y|L_x)$ for all the possible values of L_Y, i.e., for each L_X value, the strongest dependencies are near the origin of the y-axis (the red part of the plots). Figure 2 shows four pairwise dependencies as measured by $P(L_y|L_x)$ on the LastFM2 dataset. All of them show clear peaks along the 0-y ordinate, and the peak values decrease when x increases. For all these examples, there is a clear dependency between the two variables. If we take a closer look, we can see (first row of the Figure) that $P(L_{user}|L_{track})$ is more peaked than $P(L_{user}|L_{user})$, meaning that the music a user listens to is more correlated to the user labels than to his friends' labels. The second row of the figure shows a similar phenomenon: $P(L_{artist}|L_{user}) < P(L_{artist}|L_{track})$, meaning that the labels for an artist are more correlated to his musical production than to the labels of his followers. We also show under each figure the conditional entropy $H(L_y|L_x)$ for each couple of variables. The most peaky distributions have an entropy, which is two orders of magnitude lower than the flatter distributions. Table 4 summarizes the conditional entropies computed for the different relations on the LastFM2 dataset. From those values, we could conclude that we should learn more from the tracks a user listen to than from the music his friends listen to. This matches the intuition of the semantics of such relations. We show in Section 4.6 that this property is properly captured when learning the relation specific

[3] The dataset is available at http://www.imdb.com/interfaces.

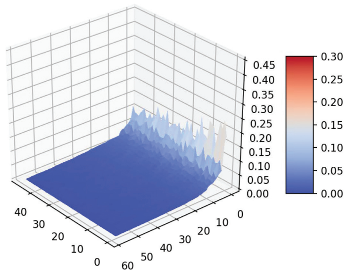
[4] $H(L_y | L_x = l_x) = \sum_{l_y \in \mathcal{L}_y} p(L_x = l_x, L_y = l_y) \log \frac{p(L_x = l_x)}{p(L_x = l_x, L_y = l_y)}$.



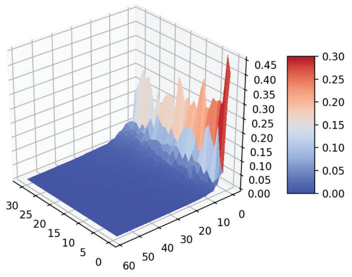
(a) $P(L_{user}|L_{user})$
Conditional Entropy 3.2e-3



(b) $P(L_{user}|L_{track})$
Conditional Entropy 8.7e-5



(c) $P(L_{artist}|L_{user})$
Conditional Entropy 1.2e-3



(d) $P(L_{artist}|L_{track})$
Conditional Entropy 4.4e-5

Fig. 2. Plots illustrating the interdependencies between labels of two node types for the LastFM2 dataset. Each plot shows $P(L_y = l_y|L_x = l_x)$ for a specific variable couple (l_x, l_y). The values l_x and l_y have been reordered for clarity (see text). The x-axis corresponding to variable L_x is on the bottom left of each plot, and the y-axis corresponding to the L_y is on the bottom right.

Table 4. Conditional Entropies $H(L_Y|L_X)$ for the LastFM2 Corpus, X and Y Being Neighbors in the LastFM Graph

Y \ X	User	Track	Album	Artist
User	$3.2 \cdot 10^{-3}$	$8.7 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$
Track	$1.1 \cdot 10^{-3}$		$3.0 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$
Album	$1.2 \cdot 10^{-3}$	$1.0 \cdot 10^{-4}$		$2.4 \cdot 10^{-4}$
Artist	$1.2 \cdot 10^{-3}$	$4.4 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	

Note: Lower values mean higher dependency between the two variables.

parameters w_r. Additional figures exhibiting similar behaviors for other relations and corpora are available in Appendices C and D (Figures 6, 7, 8 and Tables 15, 16, 17).

Further insights into this dependency are provided by Figure 3. It shows the cumulative values of $P(L_y|L_x)$ for different relations w.r.t. the percentage of the number of couples considered on the LastFM2 dataset. We start with the highest $P(L_y|L_x)$ for any relation, the value of which is plotted

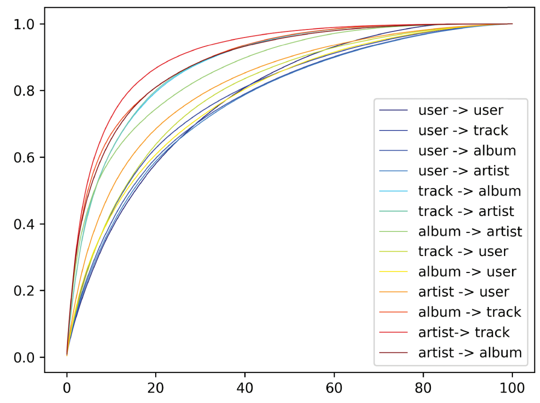


Fig. 3. Plot of the average of the cumulative sum of conditional probabilities $P(L_y|L_x)$ for all relation types on the LastFM2 dataset. The $P(L_y|L_x)$s are in decreasing order on the x-axis. $x = 40\%$ means that 40% of the conditional probability values have been considered and the corresponding cumulative value is plotted on the y-axis.

at $x = 0\%$, and we accumulate the successive values $P(L_y|L_x)$ taken in decreasing order. The figure shows first that all the relations between the pairs of variables exhibit some dependency: at $x = 20\%$, one already has a cumulated value of $P(L_y|L_x)$ between 58% and 87% regardless of the relation type. Second, some relations are clearly more informative than others, e.g., for $x = 10\%$ the relation *user* $\rightarrow$ *user* has cumulated 38% of conditional probability, whereas the relation *artist* $\rightarrow$ *track* has cumulated 72%. Additional figures exhibiting similar behaviors for other corpora are available in Appendix E (Figure 9).

These statistics clearly show that there exists strong interdependencies between labels of different node types, which could be exploited by an appropriate model, and this confirms our base hypothesis.

4.3 Comparison with Other Models

We compare our approach with four baselines described below. The first one, *LINE* [41] is representative of unsupervised learning methods for graph embeddings. Based on [26], LINE learns distributed representations by maximizing the probability of a node knowing its neighbors. We then performed a logistic regression with the learned representations as inputs. The second one, HLP [53], provides a comparison with a representative model of label propagation, typical of semisupervised transductive learning on homogeneous graphs. For HLP, there are two main ways to construct an homogeneous graph from an heterogeneous one, each considers a separate classification problem for any node type:

— For each node type, use the whole graph for propagating the labels of this given type: only the nodes from this type will be considered for classification, and all the nodes are considered in the propagation loss. Then, repeat the propagation for all types of nodes.
— Construct a projection of the heterogeneous graph for each type of nodes, thus building an homogeneous graph, and then propagate the labels on each homogeneous graphs.

The second option (projection) requires defining projections of the heterogeneous graph onto homogeneous ones. There are many possible choices for that and the results heavily depend on this choice. We rather considered the first option. The third baseline, *Graffiti*, is a state of the art model for the task of classification in heterogeneous graphs [2]. It is a modified random walk that captures the influence among nodes of the same type by considering their common neighbors. For example, given a node, it can jump to another node of the same type by either following a direct link or using a 2-hop jump via a common neighbor of another type. Graffiti has been shown to be superior to the following four models on different datasets (FlickR and LibraryThing): a naive Bayesian classifier, TopicalPR [31], a hybrid classifier [28], and an iterative relaxation labeling classifier [3]. Although it allows to develop random walks on heterogeneous graphs, it does not take advantage of the correlations between labels as our model does. Finally, we experimented with a transductive approach, which is closer to our work, namely *Max-Margin Deep Walk* (MMDW) proposed[5] in [42]. Like LaHNet, this approach uses a classification loss, which is a Support Vector Machine (SVM), as well as a regularization loss, which is based on Deep Walk [33]. The main differences with our approach are that (1) there is no overall regularized loss; (2) the Deep Walk criterion is different from ours and imposes more constraints on the latent space since linked nodes should have a high inner product, and hence the same direction in space; (3) MMDW does not differentiate the importance of relations or nodes like LaHNET. We show in the experimental section that these three differences have an important impact on the performance.

4.4 Evaluation Measures and Protocol

4.4.1 Evaluation Measures. We have considered two different evaluation measures: *Precision at 1 (P@1)* measures the percentage of nodes for which the category with the highest predicted score is among the observed labels for this node. For monolabel classification, this should be the target label, while for multilabel classification, this could be any of the target labels. *Precision at k (P@k)* is the proportion of correct labels in the set of k labels with the highest predicted scores. For the monolabel dataset DBLP, we only make use of *Precision at 1 (P@1)*. For the multilabels dataset, P@k will denote an average over all the node types, with k set to the number of categories a node belongs to. We optimized the different models with regard to microaverage, but we report both microaverage and macroaverage precision P@•.

4.4.2 Hyperparameters. LINE has four hyperparameters: the mini-batch size, the learning rate, the dimensionality of the unsupervised learned representation, and the number of negative samples. *HLP* has one hyperparameter: the number of times the model is iterated. *Graffiti* has four hyperparameters: three characterize the random walk and one the number of iterations. *MMDW* has three hyperparameters: the regularization term λ, the SVM margin C, the SVM bias α. Given the algorithmic complexity of MMDW, we verified that the hyperparameters chosen by the authors [42] were the best on DBLP, FlickR, and LastFM1, which was the case, and used them for IMDB and LastFM2.

Finally, our model has seven hyperparameters described in Table 5. The hyperparameters for all the models were optimized by grid search on a validation set: we tested all possible settings, and selected those that maximized the precision at k measure on the validation set. The performance was then measured on the test set.

4.4.3 Protocol. Since we are in a transductive setting, all the nodes are used for training—but not all the labels. For all the methods, we partition the labeled nodes into three datasets: train, validation, and test. Model parameters are trained as described in Algorithm 1.[6] Model selection

[5]We adapted the code from the authors https://github.com/thunlp/MMDW. To deal with our datasets, which are much bigger than those the authors experimented with, we (1) optimized the code to leverage optimized linear algebra libraries; (2) used gradient descent instead of second order methods for two of our biggest datasets, namely IMDB and LastFM2.
[6]We split the training set into two distinct training sets for optimizing, respectively, $\mathcal{N}_C$ and $\mathcal{N}_W$.

Table 5. Hyperparameters of the Model

Hyperparameter	Meaning
d	Dimension of the latent vector space: [50; 200]
$\mathtt{GS}_z$	Step size of the gradient descent when updating the vector representations: $[10^{-4}; 10^{-3}]$
$\mathtt{ITER}_z$	Number of iterations of the algorithm: [100; 1000]
$\mathtt{V}$	Variance for the initialization of z representations: $[10^{-4}; 10^{-3}]$
λ	Tradeoff between classification and smoothness: [10; 100]
$\mathtt{GS}_w$	Step size of the gradient descent when updating the w_r: $[10^{-1}; 1]$
$\mathtt{ITER}_w$	Number of iterations when to stop learning of w_r: [10; 100]

(i.e., hyperparameter selection) is performed on the validation set using a *Micro P@k* measure, which corresponds to the mean of P@k over all nodes. Performance is then evaluated on the test set.

Experiments were performed with different training + validation set sizes: 10%, 30%, 50% of the total size of the labeled dataset was used for training and validation. For a training + validation size of 10% of the labeled nodes, we used a 50–50 partition (training-validation) of the labeled nodes. We used a 80–20% partition for the other training + validation sizes. The training nodes are selected at random. Summarizing, this gives us three scenarios:

10%: 5% for training, 5% for validation, and 90% for testing.
30%: 24% for training, 6% for validation, and 70% for testing.
50%: 40% for training, 10% for validation, and 50% for testing.

For example, in the LastFM2 dataset, for the user node type, a training + validation set of 10% means that 500 users out of 5,000 labeled users have been selected; among those 500 users, 250 (50%) are used during the parameters training phase and the remaining 250 are used for the validation. For a training + validation set of 30%, we have 1,500 users that are divided into 1,200 users (80%) for training and remaining 300 users for validation. Note that for this dataset, only 5,000 users out of 20,004 are labeled. Unlabeled nodes only appear in the graph regularization term of Equation (1).

Since, in our model, we need to train the hyperparameters, we further split for this model the training set into two distinct subsets. The first one (denoted $\mathcal{N}_C$) is used to learn the representations and the classifiers and corresponds to 90% of this training set. The second one (denoted $\mathcal{N}_W$) is used to learn the hyperparameters w_r and corresponds to 10% of this training set.

Finally, experiments are performed on five random splits. The hyperparameters are selected for each split using the validation set. We then average 10 runs over each split to account for the random initialization of parameters.

4.5 Results

In this section, we compare our model to the baselines on the five datasets described in Table 3. For each dataset, we report the averaged P@k measures (microaverage and macroaverage on the node types) and the P@k for each node type. The best performance on the test set is in bold. Results are presented for the different datasets in Tables 6–10.

The main conclusions are as follows:

— Supervised models have better performance than the representative unsupervised LINE, which proceeds in two steps: unsupervised representation learning followed by classifier learning.

Table 6. P@1 DBLP

Train + Val. size	Model	Train	Val	Test			
		Micro		Micro	Macro	Author	Paper
10%	LINE	25.1	18.9	19.5	23.0	29.1	16.8
	HLP	100	24.7	24.1	27.2	32.6	21.8
	MMDW	89.7	25.7	25.5	29.9	37.7	22.0
	Graffiti	100	32.4	30.9	38.1	50.8	25.3
	LaHNet	99.8	33.8	**32.1**	**40.0**	**53.9**	**26.0**
30%	LINE	24.0	21.5	21.9	24.8	30.1	19.5
	HLP	100	35.8	36.0	41.9	52.4	31.4
	MMDW	86.4	36.0	35.5	41.9	53.4	30.4
	Graffiti	100	39.6	38.5	46.6	61.1	**32.1**
	LaHNet	99.7	43.0	**41.2**	**52.9**	**73.8**	31.9
50%	LINE	24.2	21.1	22.3	25.0	29.8	20.2
	HLP	100	39.7	39.4	46.5	59.3	33.7
	MMDW	82.8	38.6	38.0	44.8	57.0	32.7
	Graffiti	100	41.5	41.2	49.4	64.1	**34.8**
	LaHNet	99.9	45.5	**44.4**	**56.8**	**79.2**	34.5

Table 7. P@k FlickR

Train + Val. size	Model	Train	Val	Test			
		Micro		Micro	Macro	User	Photo
10%	LINE	24.4	19.4	20.7	23.2	29.1	17.3
	HLP	100	26.0	26.3	27.8	31.3	24.3
	MMDW	62.4	21.5	21.0	21.0	20.9	21.0
	Graffiti	100	24.3	24.5	27.0	**32.7**	21.2
	LaHNet	99.3	31.8	**31.9**	**31.9**	31.8	**32.0**
30%	LINE	23.0	21.6	21.5	24.2	30.6	17.9
	HLP	100	47.6	47.7	43.7	34.5	53.0
	MMDW	54.4	32.4	31.5	28.0	20.0	36.0
	Graffiti	100	47.5	47.0	43.7	**36.1**	51.3
	LaHNet	100	50.1	**49.0**	**44.3**	33.3	**55.3**
50%	LINE	23.2	21.8	21.8	24.6	31.0	18.2
	HLP	100	54.2	54.1	48.6	35.8	61.4
	MMDW	54.9	37.5	36.7	31.9	20.9	42.9
	Graffiti	100	54.4	54.0	**48.8**	**36.9**	60.8
	LaHNet	99.9	55.8	**54.3**	48.2	33.9	**62.4**

—On all datasets *HLP* is below *Graffiti* and *LaHNet*. This clearly shows that modeling the heterogeneity of the graph is essential.
—On all datasets *MMDW* is below *Graffiti* and *LaHNet*. We discuss the reasons below.
—*LaHNet* outperforms the baselines on four out of five datasets.

Let us first analyze the performance of the different models. Based on the microaverage P@k measure, our model outperforms all the other models on all the data sets except IMDB. *LaHNet*

Table 8. P@k LastFM1

Train + Val. size	Model	Train Micro	Val Micro	Test Micro	Test Macro	Test User	Test Track	Test Album	Test Artist
10%	LINE	20.8	20.6	20.4	15.9	5.6	26.0	14.5	17.4
	HLP	98.7	38.1	38.4	30.0	9.9	47.8	27.2	35.1
	MMDW	58.4	25.1	25.3	19.6	7.4	32.9	17.4	20.7
	Graffiti	100	40.1	**40.0**	**31.4**	**10.6**	49.0	28.1	**38.1**
	LaHNet	100	39.5	39.0	29.2	9.5	**50.7**	**30.8**	25.9
30%	LINE	20.5	20.9	20.5	17.0	10.1	25.9	14.4	17.5
	HLP	98.9	50.2	49.7	40.0	**17.2**	60.5	37.7	44.8
	MMDW	60.0	32.6	32.4	25.9	12.9	42.1	22.8	25.6
	Graffiti	100	50.8	50.3	40.4	**17.2**	61.7	36.2	46.5
	LaHNet	100	56.6	**55.6**	**43.6**	15.3	**67.1**	**44.7**	**47.3**
50%	LINE	20.5	20.5	20.5	17.0	10.3	26.0	14.4	17.5
	HLP	98.8	51.9	52.1	42.3	**19.4**	63.1	40.2	46.4
	MMDW	60.5	34.2	34.2	27.6	15.1	44.6	24.1	26.6
	Graffiti	100	53.2	53.5	43.2	19.1	65.4	39.5	48.7
	LaHNet	100	59.2	**59.3**	**46.9**	16.8	**70.4**	**47.8**	**52.7**

Table 9. P@k LastFM2

Train + Val. size	Model	Train Micro	Val Micro	Test Micro	Test Macro	Test User	Test Track	Test Album	Test Artist
10%	LINE	17.5	17.4	17.5	15.3	13.8	20.1	12.2	15.1
	HLP	98.2	43.5	43.4	35.6	**26.6**	51.7	29.7	34.6
	MMDW	31.6	15.9	15.8	12.6	11.4	20.3	10.4	8.4
	Graffiti	100	44.4	44.3	**35.8**	25.9	53.1	29.3	**35.0**
	LaHNet	100	44.5	**44.5**	30.9	11.6	**56.7**	**30.9**	24.5
30%	LINE	17.5	17.5	17.5	15.5	14.3	20.2	12.2	15.1
	HLP	98.2	50.0	50.0	41.8	31.5	58.4	37.9	39.3
	MMDW	36.0	23.0	23.1	18.5	15.1	28.7	16.8	13.4
	Graffiti	100	53.9	54.0	43.7	**31.9**	64.5	38.2	40.2
	LaHNet	99.8	57.3	**56.9**	**45.4**	27.6	**66.8**	**45.0**	**42.1**
50%	LINE	17.5	17.6	17.5	15.4	14.2	20.2	12.2	15.1
	HLP	98.2	51.7	51.9	43.6	33.2	60.4	39.9	41.0
	MMDW	35.3	25.8	26.0	20.6	14.9	31.7	19.5	16.4
	Graffiti	100	56.8	57.0	46.5	**34.4**	67.7	41.8	42.4
	LaHNet	100	59.6	**59.5**	**47.8**	29.0	**69.3**	**47.7**	**45.1**

outperforms (or is close to for IMDB) *Graffiti*, *HLP*, and *MMDW* with on average over all the settings, +2.4 points for DBLP (see Table 6), +3.0 for FlickR (see Table 7), +3.6 for LastFM1 (see Table 8), +1.9 for LastFM2 (see Table 9) and −0.8 for IMDB (see Table 10) compared to the best competitor. The behavior of the models is slightly different according to the datasets. On DBLP, *LaHNet* is superior to all the other models for all the settings (10%, 30%, 50% of labeled data used for training). For FlickR, *LaHNet* is better at (10%, 30%) and is similar to *Graffiti* and *HLP* for 50%. On the LastFM datasets, *LaHNet* is better at 30% and 50% while all the supervised

Table 10. P@k IMDB

Train + Val. size	Model	Train	Val	Test
10%	LINE	38.7	38.6	38.6
	HLP	100	39.6	39.7
	MMDW	32.9	31.9	31.8
	Graffiti	100	44.9	**44.9**
	LaHNet	94.1	44.9	44.7
30%	LINE	38.6	38.5	38.6
	HLP	100	47.7	47.7
	MMDW	32.9	32.4	32.5
	Graffiti	100	49.6	**49.6**
	LaHNet	99.4	49.0	48.8
50%	LINE	38.6	38.6	38.6
	HLP	100	50.4	50.4
	MMDW	32.5	32.3	32.4
	Graffiti	100	51.3	**51.3**
	LaHNet	99.5	50.3	50.1

Table 11. Comparison of the Evolution of P@k Performance between LaHNet (LHN) and the Best Baseline When Increasing the Training + Validation Size (TVS)

Dataset	DBLP		LastFM1		LastFM2		FlickR		IMDB	
Model	LHN	Graf.	LHN	Graf.	LHN	Graf.	LHN	HLP	LHN	Graf.
TVS 10%	32.1	30.9	39.0	40.0	55.6	50.3	31.9	26.3	44.7	44.9
TVS 10 to 30%	+9.1	+7.6	+16.6	+10.3	+12.4	+9.7	+17.1	+22.5	+4.1	+4.7
TVS 30 to 50%	+3.2	+2.7	+3.7	+3.2	+2.5	+3.0	+5.3	+7.0	+1.3	+1.7

models are similar at 10%. On IMDB, *Graffiti* and *LaHNet* have similar performance with a slight advantage to *Graffiti*. The IMDB dataset is specific since only movies have labels, hence LaHNet has no advantage over homogeneous models. Graffiti shows a slight advantage over HLP since it explicitly models the 2-hop relations occurring in this dataset.

MMDW did not perform well for most datasets; since this model has some similarities with ours, it is interesting to understand why. The algorithm was initially evaluated on small datasets (around 2,000 nodes) and does not scale well in our experiments on large datasets like LastFM2 and IMDB with millions of nodes. It has been designed for monolabel classification and the extension to multilabel does not perform well. Finally, in DBLP, *MMDW* did work better when the amount of training data was important; however, even in that case the performance did not reach the one of *Graffiti* or *LaHNet*. The difference may be explained by the following facts: (1) We have a different way to learn from the graph structure, using distance rather than inner product; (2) we learn to associate a weight with each relationship; and (3) our optimization scheme is simpler since [42] bias their learning by using the SVM learned weights, rather than learning with a single objective.

Table 11 shows the performance evolution at different training and validation set sizes (TVS) for *LaHNET* and the most competitive baseline.

With respect to the datasets, we can distinguish two groups: the three datasets DBLP, LastFM1, and LastFM2 and the two datasets FlickR and IMDB. For the first group, the performance increase for LaHNet is higher than for *Graffiti* even if *LaHNet* already outperforms *Graffiti*. This shows that

Table 12. Effect of Learning the Relation-Specific Weights w_r for FlickR

Train size	Model	Train	Val	Test			
		Micro		Micro	Macro	User	Photo
10%	Without	99.6	29.6	29.3	29.1	28.6	29.5
	With	99.3	31.8	**31.9**	**31.9**	**31.8**	**32.0**
30%	Without	99.9	49.2	48.4	43.6	32.5	54.7
	With	100	50.1	**49.0**	**44.3**	**33.3**	**55.3**
50%	Without	99.4	55.1	54.0	47.9	33.7	62.0
	With	99.9	55.8	**54.3**	**48.2**	**33.9**	**62.3**

Note: Performance is expressed as microprecisions and macroprecisions as indicated, performance for individual relations is microprecision—see text for further explanation.

in this case, for low training set size, *LaHNet* takes greater benefits of additional training data compared to *Graffiti*. The reverse holds for the second group: for IMDB the increases are comparable and *LaHNet* remains slightly below *Graffiti*, for FlickR, *LaHNet* outperforms HLP for all training set sizes, but the gap between HLP and our model decreases when the training data increases.

While the models are optimized for microaverage P@k, macroaverage P@k gives a complementary indication. Macroprecision is below microprecision for all the models, but globally the models rank similarly for both measures. Macroprecision shows that underrepresented types, e.g., users on the LastFM datasets, have low performance. Note that *LaHNet* is below *Graffiti* and HLP for these node types, probably because the number of parameters per class is higher for *LaHNet*.

4.6 Importance of the Relations' Weights

Let us now analyze the role of the relation-specific weights w_r in the model. When there is only one type of relation like DBLP, there is nothing to learn. For the other datasets, learning the w_r significantly improves the performance. We discuss below each dataset in turn, and we denote $w_{A \to B}$ the weight of the relation from node type A to node type B.

4.6.1 FlickR. After convergence the weights are almost equal on average: $w_{user \to photo} = 0.53$ and $w_{user \to user} = 0.47$ for the two relation types involving users. Actually, one can expect close weights as users follow users who publish photos they like, which might be on average the same type as their own photos. The results of the two models are reported in Table 12. Even with such a small average difference between the two types of weights, learning the w_r improves the results. It has a significant impact at small training set size (10%) with an increase of +2.6 compared to a model with no weights, and the difference tends to vanish when the training set is increased.

4.6.2 LastFM. The LastFM datasets are the most interesting: there are four types of nodes with several relations, the number of labels is larger than for the other datasets. Let us focus on the relations involving users since this is the richer and most diverse group of relations with four relation types. The results of the two models for LastFM1 and LastFM2 are reported respectively in Table 13 and Table 14.

Learning the w_r has an important impact on the user nodes performance. For LastFM1, there is an improvement of +0.5 (train + validation size 10%), +5.6 (train + validation size 30%), and +5.8 (train + validation size 50%) when learning the w_r. The same holds for LastFM2, where an improvement of +2.2 (train + validation size 10%), +6.3 (train + validation size30%), and +6.2 (train + validation size 50%) can be observed. Concerning user scores, the more training data, the bigger the improvement.

Table 13. Effect of Learning the Relation-Specific Weights w_r for LastFM1

Train size	Model	Train	Val	Test					
		Micro		Micro	Macro	User	Track	Album	Artist
10%	Without	99.9	36.4	36.3	27.2	9.0	48.4	26.2	25.3
	With	100	39.5	**39.0**	**29.2**	**9.5**	**50.7**	**30.8**	**25.9**
30%	Without	99.8	54.2	53.3	40.3	9.7	65.8	42.7	42.9
	With	100	56.6	**55.6**	**43.6**	**15.3**	**67.1**	**44.7**	**47.3**
50%	Without	99.7	56.6	56.7	43.2	11.0	68.8	45.6	47.6
	With	100	59.2	**59.3**	**46.9**	**16.8**	**70.4**	**47.8**	**52.7**

Note: Performance is expressed as microprecision and macroprecision as indicated, performance for individual relations is microprecision—see text for further explanation.

Table 14. Effect of Learning the Relation-Specific Weights w_r for LastFM2

Train size	Model	Train	Val	Test					
		Micro		Micro	Macro	User	Track	Album	Artist
10%	Without	99.8	38.1	38.0	26.1	9.4	48.9	22.1	24.1
	With	100	44.5	**44.5**	**30.9**	**11.6**	**56.7**	**30.9**	**24.5**
30%	Without	99.9	53.7	53.4	40.4	21.3	64.9	40.1	35.3
	With	99.8	57.3	**56.9**	**45.4**	**27.6**	**66.8**	**45.0**	**42.1**
50%	Without	99.8	55.9	55.9	42.8	22.8	67.1	42.9	38.5
	With	100	59.6	**59.5**	**47.8**	**29.0**	**69.3**	**47.7**	**45.1**

Note: Performance is expressed as microprecision and macroprecision as indicated, performance for individual relations is microprecision—see text for further explanation.

The role of w_r for the other relations is less important, however, it still makes a difference. On the LastFM1 dataset there is a global improvement of +2.5 on average for all the train + validation sizes. The same holds for the LastFM2 dataset, with an improvement of +3.5/+3.6 for train + validation sizes 30% and 50%, and a greater improvement for a train size of 10% (+6.5).

If we make the parallel between the learned w_r and the conditional entropy (see Figure 2 and Table 4) one observes that, for relations of type $user \to \bullet$, the lower the entropy of a user label L_{user} conditioned on the label of its neighbor $L_\bullet$ is, the higher the coefficient of the corresponding relation type is. The model then captures this dependency. For example, since the value (0.22) of the w_r coefficient for $user \to user$ is rather low, one can conclude that this relation is less relevant than, for example, the relation $user \to track$ (0.28). One can explain this result by the fact that the $user \to user$ relationship is more related to friendship and less to tastes compared to $user \to track/album/artist$. Looking at the music a user listens to will be more informative to infer what kind of music a user likes than looking at what his friends listen to.

The evolution of $w_{user \to \bullet}$ during the learning phase is reported on Figure 4. It clearly shows that the proposed procedure is able to differentiate the weights according to their importance for the classification task.

In conclusion, learning the relation specific weights clearly improves the classification results. Comparing FlickR and LastFM, one can see that learning the relation weights w_r allows capturing the most important relation types for the inference task.

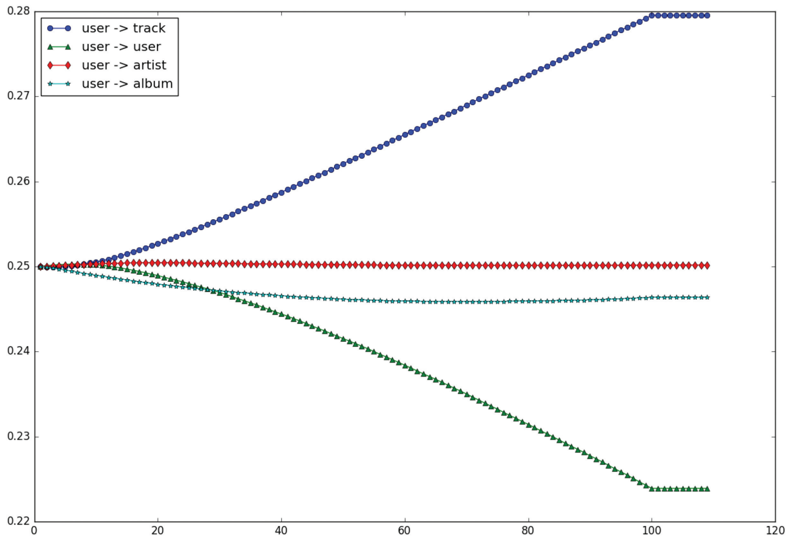


Fig. 4. Evolution of w_r values over training steps for all relations r involving users for LastFM2. At convergence $w_{\text{user}\to\text{user}} = 0.22$, $w_{\text{user}\to\text{track}} = 0.28$, $w_{\text{user}\to\text{album}} = 0.25$, $w_{\text{user}\to\text{artist}} = 0.25$.

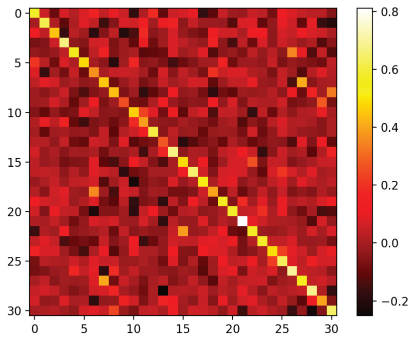


Fig. 5. LastFM2 corpus: Graph of the cosine between the 30 classifiers having the same string-wise label (e.g., “pop”) for artists and tracks. For example, line 15 corresponds to *pop_artist* and column 15 to *pop_track*. The lighter the square, the higher the scalar product.

4.7 Label Correlation on the LastFM2 Dataset

As mentioned in Section 4.1, labels for different types of nodes are supposed to be distinct even when they have the same name (e.g., the label *pop* is not considered to be the same label for a track and an artist). It is interesting to examine the relations between the representations of nodes from different types with the same labels. Since these labels are strongly correlated, intuitively the corresponding nodes should be in the same subspaces of the representation space. Since in our experiments the classifier for each label is linear, measuring the similarity of the two separating hyperplanes gives an indication on this correlation. We have plotted in Figure 5 the cosine of the linear classifier weights learned for labels with the same name for two different node types of the LastFM2 corpus. On Figure 5, the track label on the ith column corresponds to its artist counterpart on the ith row. We first notice that the cosine product is nonzero in most cases, showing existing correlations between the different classifiers, and further emphasizing the importance of learning a common latent representation space for nodes of different types.

Second, the diagonal values are all close to 1 for this example, so that the model has learned these correlations and the latent representations of tracks labeled *pop* are in the same region of the latent space as artists labeled *pop*, the same goes for the other labels. Other examples for the LastFM2 dataset are provided in Appendix F (Figure 10) for different pairs of node types.

5 RELATED WORK

The need for graph node classification stems from several application domains, like web data mining or biology. For example, web page classification may be formulated as a homogeneous graph node classification task where nodes are web pages, edges between web pages are hyperlinks and node labels are the web page topic.

Work in this domain first focused on homogeneous graphs, i.e., with only one type of node and one type of relation. The analysis of more complex data, e.g., coming from social sources or knowledge bases, where nodes may be of different types and share different and sometimes multiple relations, requires new models and techniques. Heterogeneous graph nodes classification is a recent trend and an open problem. We review below the main directions for graph nodes classification.

Most of the models share the same basic intuition [15]: neighboring nodes have similar properties, e.g., they tend to be classified similarly. In the following, we distinguish works (i) that associate with each node a classification score for each possible label, and propagate this information to neighboring nodes (ICA, random walks and regularized models); (ii) representation-based models that associate a vector in $\mathbb{R}^d$ with each node of the graph.

Iterative classification algorithms (ICA) [38] are extensions of classical inductive classification schemes to relational data. They consist in iteratively building a local classifier at each node, using as inputs both node characteristics and statistics on the node neighbors current labels. Standard models, like Bayesian classifiers [30] or logistic regression [22], are used to perform the classification task. Several variations and extensions of these ideas have been proposed. For instance, global label distribution statistics can be taken into account through a maximum entropy constraint [35], a label regularization loss [24], or KL-divergence [25]. As ICA methods can be heavily influenced by the absence of links, Gallagher et al. [14] proposed a way to predict new links by connecting unlabeled nodes to labeled ones to circumvents the potential graph sparsity. ICA methods have been extended to multirelationship graphs, by estimating how labels propagate through each type of relationship or type of nodes [34, 45], but dealing with multiple node and relation types is still an issue for this family of methods.

Random walks have been used for graph nodes classification. Labels are propagated from labeled to unlabeled nodes using the graph structure: each label has a given probability to be propagated to the neighboring nodes, and the stationary distribution corresponds to the scores given to the different labels. The most typical work, for homogeneous graphs, is that of Zhu et al. who proposed *HLP* [53] which iterates until convergence the walk $Y \leftarrow T \cdot Y$, where Y is the matrix of predicted labels, and T characterizes the transition matrix of the graph. This branch of research has motivated a large number of extensions, as for example taking into consideration more specific information, such as graph communities [12] or label distribution [29]. For heterogeneous graphs, there are two approaches that amount at modifying the random walk by taking into account different types of nodes and relationships. One defines specific random walks [52] based on input features and graph statistics. The specific definition of the random walk has to be done manually, which is a clear limit of this type of approach. General models, like *Graffiti* [2], are based on simple extensions of the random walk process: *Graffiti* is based on two intertwined random walks, the first one operating between nodes of the same type which are directly connected in the graph while the other one is defined between nodes of the same type, which are connected through another node type. While simple, *Graffiti* is very competitive: according to our experiments with different models, it represents the state of the art in the domain and is thus one of our baselines.

Regularized models, while related to random walks, are different since the objective is formulated as loss optimization when random walks do not make use of an explicit loss. A diffusion equation appearing as a regularization term in this loss propagates through connected nodes. For instance, the authors in [50] use a regularization inspired by random walks, and the authors in [37, 44] minimize the graph-based distance between the nodes using graph Laplacian matrices. Recently, the authors in [48] tackled the homogeneous multirelational classification case, by assuming that relations have varying levels of informativeness, and, associated with them, weights that are learned. Models for heterogeneous graphs have been proposed in the case where the label set is the same for all node types [17, 19]. In that case, simply ignoring the node type allows one to use frameworks developed for the homogeneous case [17]. The authors in [19] have nevertheless made a step toward heterogeneous networks by introducing weights on relations between node types, and thus differentiating in some way the different types of nodes. Our model also incorporates a regularization term in the loss function, but contrarily to the models cited previously, our regularization does not directly involve classification scores but learned representations.

Representation learning and deep learning. Representation learning [5] encompasses different families of methods aimed at automatically building relevant representations from data. This type of approach, popularized with deep learning, is now widespread in several domains. For classification, it has been used in a supervised setting in many different contexts such as text [40], image [51], video, time series [10, 46] classification and extreme classification [7].

In the context of graphs, these approaches amount at learning a representation in $\mathbb{R}^d$ of each node of the graph. This representation can then be used as an input for any appropriate classifier. Inspired from deep learning techniques, the authors in [13, 27] tackle the homogeneous graph node classification using Recurrent Neural Networks (RNN). They transform each set of neighbors for a given node into a sequence of their attributes and use a RNN to predict the label of the last (test) node of the sequence. The authors in [36] use a neural network, where the hidden representations are computed based on the input graph. Contrary to our model, these methods require node characteristics (inductive learning).

Several works focused on learning a node representation *without* supervision. Inspired by work on word representation learning [26], the authors in [33, 41] learn distributed representations of graph nodes using a random walk. In *LINE*, [41], the joint probability of neighbor nodes is a function of the inner product of their representations. In DeepWalk [33], the authors propose to sample *paths* within the graph, and then use them as sentences fed to SkipGram [26] to learn the representations of the nodes.

Based on this work, the authors in [9] learn node representations that incorporate global and local structural information. These representations can then be used as inputs to a classifier. Similarly, the authors in [16] proposed an unsupervised learning model that maximizes the likelihood of preserving a network neighborhood of nodes. In practice, this neighborhood is defined through a biased random walk. Deep learning architectures have been used in recent works with convolutional neural networks [32] or auto-encoders [43]. In the experimental section, we compared our model to LINE [41], which is a representative instance of these unsupervised models, and we have shown that for classification, unsupervised representations did not lead to good performance.

Finally, our model belongs to the class of semisupervised transductive representation-based models. Closely related to our model are the works of [42, 47], both based on DeepWalk [33]. Yang et al. [47] have recently proposed a model that learns two different representations for a node, one from features associated with the node (e.g., the text of a web page), and another one from the graph alone. They try to predict the neighboring node representation, similarly to the SkipGram [26] model for text. The main focus of this model was to take into account external information, while in our work the focus is on handling various types of relationships between nodes without considering node characteristics. Tu et al. [42] proposed a Maximum-Margin Deep Walk (MMDW) procedure that couples DeepWalk with an SVM. There are several differences between our model and MMDW: their regularization term is based on an inner product and not on a distance like ours, our model uses a simpler optimization procedure and learns to weight the relationship. We compared with this model in our experiments, and have shown that our choices lead to improved performance in classification.

6 CONCLUSION

We have introduced a representation-based model for the challenging task of heterogeneous graph node classification. This model, LaHNet, learns, for each node and label, a latent representation in a vector space. It also learns relation specific weights that determine how informative the relations between different types of nodes are for the classification task. We introduced a continuous optimization hyperparameter scheme for simultaneously learning the different parameters of the model.

Experimental results show that LaHNet was able to learn correlations between labels of different types of nodes, thus notably improving the class label inference. We performed extensive experiments and showed that LaHNet works well compared to competing methods [3, 41, 53] on five different real-word datasets (DBLP, FlickR, LastFM, and IMDB).

APPENDIXES

A COMPUTING THE DERIVATIVE OF THE LOSS WITH RESPECT TO w_r

In this section, we show how the loss of Equation (5) can be derived with respect to a hyperparameter w_r. Let us derive a closed form for $\frac{\partial L_W}{\partial w_r}$. First, using the chain rule, the derivative of $L_W(\mathbf{w})$ with respect to a given weight w_r is

$$\begin{aligned}\frac{\partial L_W}{\partial w_r} &= \frac{\partial L_W}{\partial \Theta}\frac{\partial \Theta}{\partial w_r} + \frac{\partial L_W}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial w_r} \\ &= \sum_{\ell \in \mathcal{L}}\sum_{k=1}^{d}\frac{\partial L_W}{\partial \Theta_{\ell k}}\frac{\partial \Theta_{\ell k}}{\partial w_r} + \sum_{\ell \in \mathcal{L}}\sum_{k=1}^{d}\frac{\partial L_W}{\partial \mathbf{z}}\frac{\partial \mathbf{z}}{\partial w_r},\end{aligned} \tag{8}$$

where $\frac{\partial \Theta}{\partial w_r}$ quantifies how the classifier parameters change when w_r changes, and $\frac{\partial \mathbf{z}}{\partial w_r}$ how the node representation changes when w_r changes. In the following, we first make a hypothesis to simplify the above equation by discarding the first term, and then show how $\frac{\partial \mathbf{z}}{\partial w_r}$ can be computed.

We first suppose that the parameters of the classifiers will be much less affected by a change of w_r than the node representations. This is a reasonable assumption since θ is only indirectly impacted by w_r while $\mathbf{z}$ directly depends on w_r. This was, moreover, confirmed experimentally. Formally,

$$\left|\frac{\partial \Theta}{\partial w_r}\right| \ll \left|\frac{\partial \mathbf{z}}{\partial w_r}\right|.$$

We then suppose that $\frac{\partial L_W}{\partial \Theta}$ and $\frac{\partial L_W}{\partial \mathbf{z}}$ have roughly the same order of magnitude. The rationale is that both Θ and $\mathbf{z}$ will have a strong impact on classification error rate: the first by moving the hyperplane, the second by moving all the nodes in the latent space. Under these two assumptions, we can approximate the partial derivative of Equation (8) as

$$\frac{\partial L_W}{\partial w_r} \approx \frac{\partial L_W}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial w_r}. \tag{9}$$

By definition, $z_i(\mathbf{w})$ is the representation of node n_i when the regularized loss of Equation (1) is minimized. In that case, since the nodes used for training the parameters and the hyperparameters are disjoint, i.e., $\mathcal{N}_W \cap \mathcal{N}_C = \emptyset$, we can see that nodes in $\mathcal{N}_W$ are only present in the regularization term L_G. Based on this, by zeroing the gradient $\nabla_{z_i} L$ of the regularized loss L with respect to a node $i \in \mathcal{N}_W$, we can express $z_i(\mathbf{w})$ for $i \in \mathcal{N}_W$ as an explicit function of w:

$$z_i(\mathbf{w}) = \frac{\sum_{r\in\mathcal{R}} w_r \sum_{j\in\mathcal{N}_i^r} z_j(\mathbf{w})\phi_{ij}}{\sum_{r\in\mathcal{R}} w_r \sum_{j\in\mathcal{N}_i^r} \phi_{ij}}. \tag{10}$$

Since in the above equation the $z_j(\mathbf{w})$ depend on the weights $\mathbf{w}$, we would not be able to compute explicitly $\frac{\partial \mathbf{z}}{\partial w_r}$. Our last assumption is that the nodes in $\mathcal{N}_C$ will not change their position for a small variation of w_r. While this is not true in general, this is sensible since (1) most of the nodes are not in $\mathcal{N}_C$, and (2) the classification cost could prevent them from moving. In that case, $z_i(\mathbf{w})$ only depends on $\mathbf{w}$, and we can derive a closed form for $\frac{\partial \mathbf{z}}{\partial w_r}$.

Under these hypotheses, by plugging Equation (10) into Equation (9), the derivative of the unregularized loss with respect to a weight w_r can then be written as

$$\frac{\partial L_W}{\partial w_r} \approx \sum_{i\in\mathcal{N}_W} \sum_{\ell\in\mathcal{L}_{t_i}} \underbrace{\psi_i y_{i\ell} \frac{\sum_{r'\neq r} w_{r'} A_i^{rr'} \left(\theta_\ell \cdot S_i^{r'} - \theta_\ell \cdot S_i^{r}\right),}{\left(\sum_{r'\in\mathcal{R}} w_{r'} A_i^{rr'}\right)^2}}_{D_i^{r\ell}} \tag{11}$$

with

$$S_i^r = \frac{\sum_{j\in\mathcal{N}_i^r} z_j \phi_{ij}}{\sum_{j\in\mathcal{N}_i^r} \phi_{ij}} \quad \text{and} \quad A_i^{rr'} = \frac{\sum_{j\in\mathcal{N}_i^{r'}} \phi_{ij}}{\sum_{j\in\mathcal{N}_i^r} \phi_{ij}}. \tag{12}$$

Finally, $z_i(\mathbf{w})$ in Equation (10) is invariant to a global scaling of w_r, for $r \in \mathcal{R}$, leading to an infinite number of possible solutions. In order to avoid numerical problems, we normalize the w_r to force a unique solution. We used the following normalization:

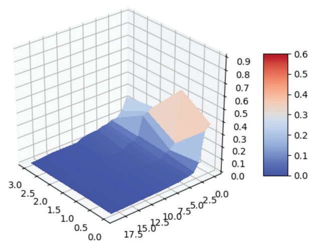
$$\text{for any node type } t, \sum_{r:t\to\bullet} w_r = 1, \tag{13}$$

where $r : t \to \bullet$ indicates all the relation types r from a node of type t to any other type of node $\bullet$.

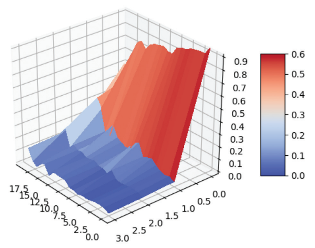
B CONSTRUCTION OF THE LASTFM DATASETS

The LastFM datasets were extracted for this work, so we describe succinctly the procedure using the methods of the LastFM API. We selected randomly a set of 10 users, and collected user friends using the `User.getFriends` method. We stopped when reaching a given limit (1,000 users for LastFM1 and 20,000 users for LastFM2). Then, for each user of our graph we fetched his top tracks, albums and artists (methods `User.getTopAlbums` `User.getTopArtists` and `User.getTopTracks`). At last, we labeled tracks, albums and artists according to the top tags LastFM users have given to them using the methods `Track.getTopTags`, `Artist.getTopTags`, and `Album.getTopTags`. For the users, we chose the top tags they added with the method `User.getTopTags`.

C PLOTS OF INTERDEPENDANCIES

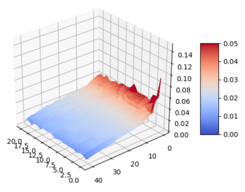


(a) $P(Y_{paper}|X_{author})$

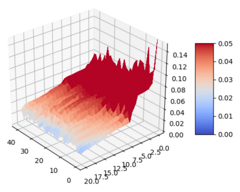


(b) $P(Y_{author}|X_{paper})$

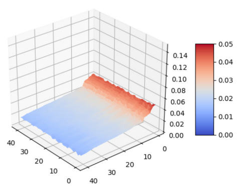
Fig. 6. Plots illustrating the interdependencies between labels on the DBLP corpus. A gray square means that there is no relation between the corresponding node types.



(a) $P(L_{user}|L_{photo})$

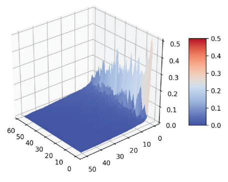


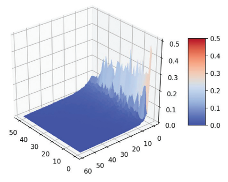
(b) $P(L_{photo}|L_{user})$

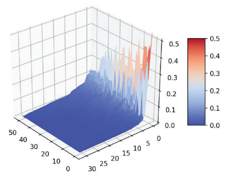


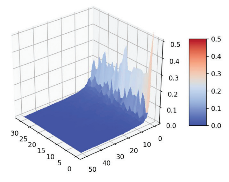
(c) $P(L_{user}|L_{user})$

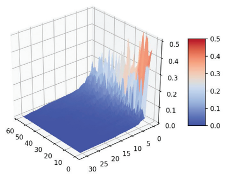
Fig. 7. Plots illustrating the interdependencies between labels on the FlickR corpus.

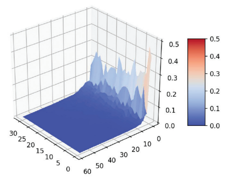

(a) $P(L_{album}|L_{artist})$

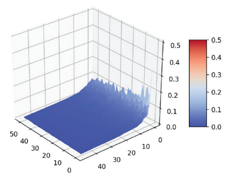

(b) $P(L_{artist}|L_{album})$

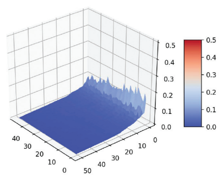

(c) $P(L_{track}|L_{album})$

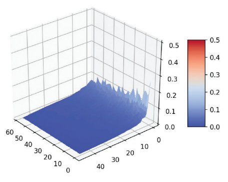

(d) $P(L_{album}|L_{track})$

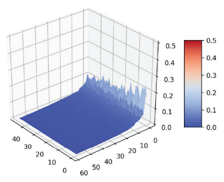

(e) $P(L_{track}|L_{artist})$

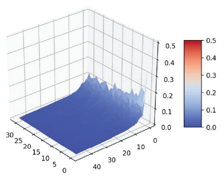

(f) $P(L_{artist}|L_{track})$

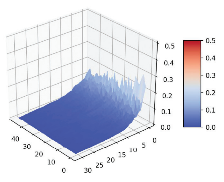

(g) $P(L_{user}|L_{album})$


(h) $P(L_{album}|L_{user})$


(i) $P(L_{user}|L_{artist})$


(j) $P(L_{artist}|L_{user})$


(k) $P(L_{user}|L_{track})$


(l) $P(L_{track}|L_{user})$

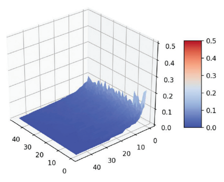

(m) $P(L_{user}|L_{user})$

Fig. 8. Plots illustrating the interdependencies between labels on the LastFM2 corpus.

D CONDITIONAL ENTROPIES

Table 15. Conditional Entropies $H(L_y|L_x)$ for the DBLP Corpus

X / Y	Author	Paper
Author		$7.1 \cdot 10^{-4}$
Paper	$2.5 \cdot 10^{-3}$	

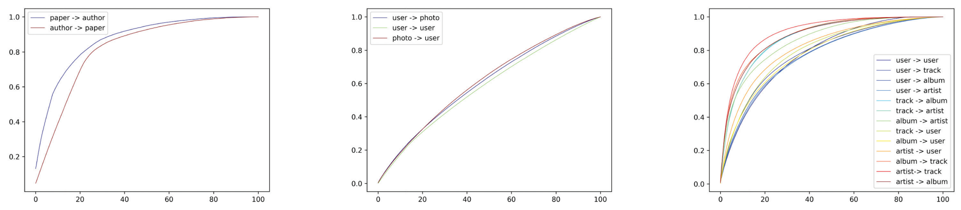
Table 16. Conditional Entropies $H(L_y|L_x)$ for the FlickR Corpus

X / Y	User	Photo
User	$4.7 \cdot 10^{-4}$	$8.9 \cdot 10^{-4}$
Photo	$8.1 \cdot 10^{-4}$	

Table 17. Conditional Entropies $H(L_y|L_x)$ for the Lastfm2 Corpus

X / Y	User	Track	Album	Artist
User	$3.2 \cdot 10^{-3}$	$8.7 \cdot 10^{-5}$	$2.4 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$
Track	$1.1 \cdot 10^{-3}$		$3.0 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$
Album	$1.2 \cdot 10^{-3}$	$1.0 \cdot 10^{-4}$		$2.4 \cdot 10^{-4}$
Artist	$1.2 \cdot 10^{-3}$	$4.4 \cdot 10^{-5}$	$1.4 \cdot 10^{-4}$	

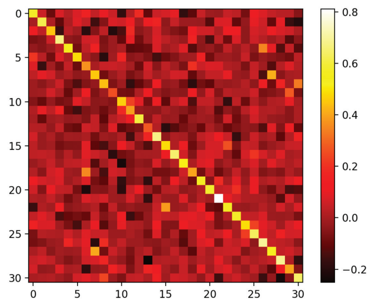
E CUMULATIVE SUM OF CONDITIONAL PROBABILITIES



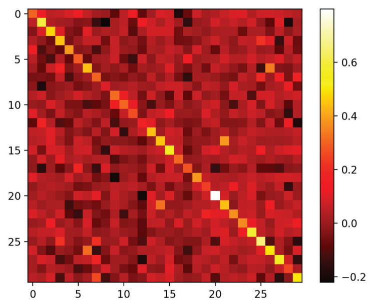
(a) DBLP dataset (b) FlickR dataset (c) LastFM2 dataset

Fig. 9. Plots of the average of the cumulative sum of conditional probabilities $P(L_y|L_x)$ (with cumulative percentages on the x-axis) for all relation types on all datasets.

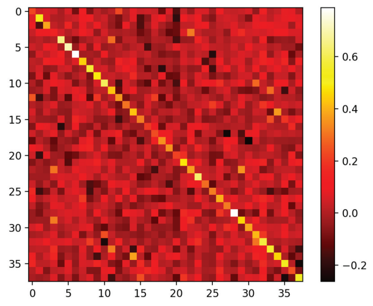
F GRAPHS OF CLASSIFIER INNER PRODUCT FOR THE LASTFM2 DATASET



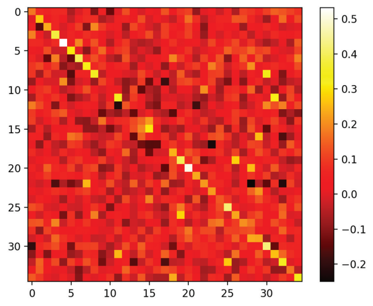
(a) Artists/Tracks labels.



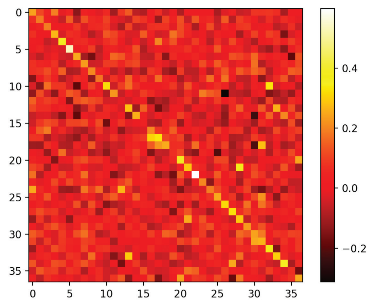
(b) Albums/Tracks labels.



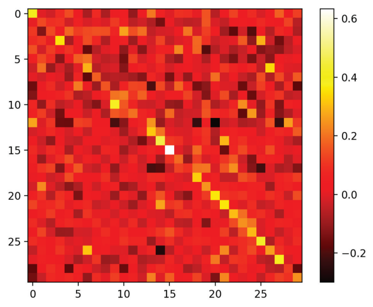
(c) Artists/Albums labels.



(d) Albums/Users labels.



(e) Artists/Users labels.



(f) Users/Tracks labels.

Fig. 10. LastFM2 corpus: Graph of the cosine between the classifiers having the same string-wise label (e.g., "pop") for two different node types.

REFERENCES

[1] Jacob Abernethy, Olivier Chapelle, and Carlos Castillo. 2008. WITCH: A new approach to web spam detection. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web (AIRWeb'08)*.

[2] Ralitsa Angelova, Gjergji Kasneci, and Gerhard Weikum. 2012. Graffiti: Graph-based classification in heterogeneous networks. *World Wide Web* 15, 2 (2012), 139–170.

[3] Ralitsa Angelova and Gerhard Weikum. 2006. Graph-based text classification: Learn from your neighbors. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 485–492.

[4] Yoshua Bengio. 2000. Continuous optimization of hyper-parameters. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)*, Vol. 1. IEEE, 305–310.

[5] Yoshua Bengio, Aaron Courville, and Pierre Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[6] Smriti Bhagat, Graham Cormode, and S. Muthukrishnan. 2011. Node classification in social networks. In *Social Network Data Analytics*. Springer, 115–148.

[7] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. 2015. Sparse local embeddings for extreme multi-label classification. In *Proceedings of Advances in Neural Information Processing Systems*. 730–738.

[8] Simon Bourigault, Cedric Lagnier, Sylvain Lamprier, Ludovic Denoyer, and Patrick Gallinari. 2014. Learning social network embeddings for predicting information diffusion. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. 393–402.

[9] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. ACM, 891–900.

[10] Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. 2015. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 507–516.

[11] Darcy Davis, Ryan Lichtenwalter, and N. V. Chawla. 2011. Multi-relational link prediction in heterogeneous information networks. In *ASONAM*. http://www.nd.edu/~dial/papers/ASONAM11b.pdf.

[12] Robin Devooght, Amin Mantrach, Ilkka Kivimäki, Hugues Bersini, Alejandro Jaimes, and Marco Saerens. 2014. Random walks based modularity: Application to semi-supervised learning. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 213–224.

[13] Shuangfei Fan and Bert Huang. 2017. Recurrent collective classification. arXiv:1703.06514 (2017).

[14] Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. 2008. Using ghost edges for classification in sparsely labeled networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 256–264.

[15] Lise Getoor. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

[16] Aditya Grover and Jure Leskovec. 2016. Node2Vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[17] Taehyun Hwang and Rui Kuang. 2010. A heterogeneous label propagation algorithm for disease gene discovery. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM'10)*. 12. http://www.siam.org/proceedings/datamining/2010/dm10_051_hwangt.pdf

[18] Yann Jacob, Ludovic Denoyer, and Patrick Gallinari. 2014. Learning latent representations of nodes for classifying in heterogeneous social networks. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. ACM, 373–382.

[19] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. 2010. Graph regularized transductive classification on heterogeneous information networks. In *Proceedings of the ECML PKDD*, Vol. 0053. Springer, 570–586. http://www.springerlink.com/index/E2700PJ82469276W.pdf.

[20] Sylvain Lamprier, Simon Bourigault, and Patrick Gallinari. 2016. Influence learning for cascade diffusion models: Focus on partial orders of infections. *Social Network Analysis and Mining* 6, 1 (2016), 93.

[21] Lu Liu, Jie Tang, Jiawei Han, and Meng Jiang. 2010. Mining topic-level influence in heterogeneous networks. In *Proceedings of the CIKM*. http://portal.acm.org/citation.cfm?id=1871467.

[22] Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proceedings of the ICML*, Vol. 3. 496–503.

[23] Jelena Luketina, Tapani Raiko, Mathias Berglund, and Klaus Greff. 2016. Scalable gradient-based tuning of continuous regularization hyperparameters. In *Proceedings of the 33nd International Conference on Machine Learning (ICML'16)*. New York City, NY, June 19–24, 2016, 2952–2960.

[24] Gideon S. Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research* 11 (2010), 955–984.

[25] Luke McDowell and David Aha. 2012. Semi-supervised collective classification via hybrid label regularization. In *Proceedings of the 29th International Conference on Machine Learning (ICML'12)*. John Langford and Joelle Pineau (Eds.). Omnipress, New York, NY, 975–982.

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems*. 3111–3119.

[27] John Moore and Jennifer Neville. 2017. Deep collective inference. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence.*

[28] Sung Hyon Myaeng and Mann-ho Lee. 2000. A practical hypertext categorization method using links and incrementally available class information. In *Proceedings of the 23rd ACM International Conference on Research and Development in Information Retrieval (SIGIR'00).* Athens, GR, Citeseer.

[29] Sharad Nandanwar and M. N. Murty. 2016. Structural neighborhood based classification of nodes in a network. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16).* ACM, New York, NY, 1085–1094. DOI: http://dx.doi.org/10.1145/2939672.2939782

[30] Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *Proceedings of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data.* 13–20.

[31] Lan Nie, Brian D. Davison, and Xiaoguang Qi. 2006. Topical link analysis for web search. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 91–98.

[32] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *Proceedings of ICML.*

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

[34] Stephane Peters, Ludovic Denoyer, and Patrick Gallinari. 2010. Iterative annotation of multi-relational social networks. In *Proceedings of 2010 International Conference on Advances in Social Networks Analysis and Mining (ASONAM'10).* IEEE, 96–103.

[35] Joseph J. Pfeiffer III, Jennifer Neville, and Paul N. Bennett. 2015. Overcoming relational learning biases to accurately predict preferences in large scale networks. In *Proceedings of the 24th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 853–863.

[36] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2017. Column networks for collective classification. *AAAI.* 2485–2491.

[37] Rakesh Pimplikar, Dinesh Garg, Deepesh Bharani, and Gyana Parija. 2014. Learning to propagate rare labels. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management.* ACM, 201–210.

[38] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Magazine* 29, 3 (2008), 93.

[39] Yizhou Sun, Yintao Yu, and Jiawei Han. 2009. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of KDD.* 797–806.

[40] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. 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.* ACM, 1165–1174.

[41] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 1067–1077.

[42] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-margin DeepWalk: Discriminative learning of network representation. In *Proceedings of IJCAI.* 3889–3895.

[43] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16).* 1225–1234.

[44] Jun Wang, Tony Jebara, and Shih-Fu Chang. 2008. Graph transduction via alternating minimization. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08).* ACM, New York, NY, 1144–1151. DOI: http://dx.doi.org/10.1145/1390156.1390300

[45] Xi Wang and Gita Sukthankar. 2013. Multi-label relational neighbor classification using social context features. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 464–472.

[46] Zuxuan Wu, Xi Wang, Yu-Gang Jiang, Hao Ye, and Xiangyang Xue. 2015. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference.* ACM, 461–470.

[47] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *Proceedings of ICML.*

[48] Junting Ye and Leman Akoglu. 2018. Robust semi-supervised learning on multiple networks with noise. *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 196–208.

[49] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16.* Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf (Eds.). MIT Press, Cambridge, MA.

[50] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2005. Learning from labeled and unlabeled data on a directed graph. In *Proceedings of the 22nd International Conference on Machine Learning (ICML'05)*. ACM, New York, NY, 1036–1043. DOI: http://dx.doi.org/10.1145/1102351.1102482

[51] Sijun Zhou, Shizhou Zhang, and Jinjun Wang. 2015. Deep sparse coding network for image classification. In *Proceedings of the 7th International Conference on Internet Multimedia Computing and Service*. ACM, 24.

[52] Yang Zhou and Ling Liu. 2014. Activity-edge centric multi-label classification for mining heterogeneous information networks. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1276–1285.

[53] Xiaojin Zhu and Zoubin Ghahramani. 2002. *Learning from Labeled and Unlabeled Data with Label Propagation*. Technical Report. Citeseer.

Received June 2017; revised March 2018; accepted March 2018