Semisupervised Learning on Heterogeneous Graphs and its Applications to Facebook News Feed

Cheng Ju
Facebook, Inc
University of California, Berkeley

James Li
Facebook, Inc

Bram Wasti
Facebook, Inc

Shengbo Guo
Facebook, Inc

ABSTRACT

Graph-based semi-supervised learning is a fundamental machine learning problem, and has been well studied. Most studies focus on homogeneous networks (e.g. citation network, friend network). In the present paper, we propose the Heterogeneous Embedding Label Propagation (HELP) algorithm, a graph-based semi-supervised deep learning algorithm, for graphs that are characterized by heterogeneous node types. Empirically, we demonstrate the effectiveness of this method in domain classification tasks with Facebook user-domain interaction graph, and compare the performance of the proposed HELP algorithm with the state of the art algorithms. We show that the HELP algorithm improves the predictive performance across multiple tasks, together with semantically meaningful embedding that are discriminative for downstream classification or regression tasks.

CCS CONCEPTS

•**Information systems** →*Data mining;*

KEYWORDS

Social Network; Semisupervised Learning; Neural Netorks

ACM Reference format:
Cheng Ju, James Li, Bram Wasti, and Shengbo Guo. 2017. Semisupervised Learning on Heterogeneous Graphs and its Applications to Facebook News Feed. In *Proceedings of ACM SIGKDD, London, UK, Aug 2018 (SIGKDD'18),* 9 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Graph-based semi-supervised learning is widely used in network analysis, for prediction/clustering tasks over nodes and edges. A class of commonly used approaches can be considered as a two-stage procedure: the first first step is node embedding, where each nodes are represented in a vector which contains the graph information; the second step simply apply these vectors are further for the conventional machine learning tasks. [23] proposed a spectral clustering method, which uses the eigenvectors of the normalized Laplacian matrix as node embedding, and applies k-means algorithm on the embedding vectors for unsupervised clustering. [22] proposed another clustering method, using the eigenvectors of the modularity matrix to find hidden community in networks . [1] generated several handcrafted local features (e.g. sum of neighbors) as embedding, and applied supervised learning on them to predict the probability that two node would be connected in the future, which is more flexible compared to proximity based link-prediction [17, 19]. [29, 31] further studied the embedding methods proposed by [22, 23] for supervised learning tasks, to predict the community label of the nodes in social network, which showed great success. [30] proposed a edge-centric clustering scheme, which learns a sparse social dimension for each node by clustering its edges. Recently, several deep learning based representation learning methods have shown great success in a wide range of tasks for network data. DEEPWALK [26] learns latent representations of vertices in a network based on truncated random walks and the SkipGram model. Node2vec [10] further extends DEEPWALK by two additional bias search parameters which controls the random walks, and thus control the representation on homophilic and structural pattern. Both of [26] and [10] are assessed by feeding the generated embedding into a supervised task on graph. Compared to previous embedding methods, these two methods are more flexible and scalable: the features could be learned by parallel training with stochastic gradient descent, and adding new nodes on the graph does not require recomputing the features for all the observations. With extra computational trick like negative sampling and hierarchical loss [21], the computation could be further reduced. To learn sparse features, [6] further proposed a deep learning based model for the latent representation learning of mixed categories of vertex. Large-scale information network embedding [28] computes the embedding by optimizing the objective function to preserve "first-order" and "second-order" graph proximity.

Another class of semi-supervised methods directly use the graph information during supervised training, instead of the two-stage embedding-learning procedure in the last paragraph. Label propagation [34] is an simple but effective algorithm, where the label information of labeled nodes are propagated on graph to unlabeled data. [32] presented a semi-supervised learning framework that learns graph embedding during the training of a supervised task. [32] further proposed both transductive and inductive version of their algorithm, and compared them with several widely used semi-supervised methods. The neural graph machine [5] extended idea of label propagation of regularizing on the final prediction to regularizing the hidden output of neural networks. Another class

SIGKDD'18, London, UK

DOI: 10.475/123_4

of algorithms build additional nuisance task to predict the graph context, in addition to the supervised label prediction.

Most work about semi-supervised learning on graph focused on homogeneous networks, where there exists only singular type of nodes and relationships. LSHM (Latent Space Heterogeneous Model) is proposed by [11], which creates a loop-up table for the embedding of each node in the graph. The model are trained by both the supervised loss, defined as classification loss from a logistic regression model on the top of the embedding, and an unsupervised loss, defined as the distance between two connected nodes. [6] further proposed the Heterogeneous Networks Embedding (HNE) algorithm based on deep neural networks, which in contrast is a purely unsupervised method. It uses each pair of node as input to predict their similarity, and define a hidden output as the embedding. It applies different network structure to process nodes with different type, while keeps the networks sharing the parameter for same type of node. Inspired by DeepWalk and Node2vec, [8] proposed a new meta-path-based random-walk strategy to build the sequences of nodes, and then feed them into SkipGram model to get a unsupervised embedding for each node.

In this work, we propose a new graph-based semi-supervised algorithm, HELP (heterogeneous embedding label propagation). It is an inductive algorithm that can utilize both the features and the graph where predictions can be made on instances unobserved in the graph seen at training time. It is also able to handle multiple heterogeneous nodes in the graph, and generate embedding for them. We call it "label propagation" as it also implicitly impose a "smooth constraint" based on the graph [5], which is similar to the label propagation algorithm [34]. We also demonstrated the effectiveness of our proposed approach with several node-classification tasks on a subset of the Facebook graph consisting of users and Web Domains, with focus in particular to identifying domains who repeatedly show content that are sensational [2] and/or otherwise low quality [18], or domains who repeatedly show content that are authentic and high quality [16].

2 MOTIVATION

There are multiple factors that influence the ranking of a story on a person's News Feed. A comprehensive look of the many factors involved can be found in [3]. For content that contains links to outside Web domains, one of the most important factor is the quality of the content from this domain. There are different dimensions under consideration for the overall quality of a domain (e.g. if its URLs always contain exaggerated headlines). For many of the important dimensions, we train classifiers to predict the likelihood a piece of content is of this dimension using content features. These classifier predictions are then used in conjunction with other signals (e.g. timeliness, interaction history) to assess the content rank on a person's News Feed.

We have following demands and expectations for the semi-supervised methods for our applications. First, as the data is large and predictions can get stale quickly, we must pay special attention to training time and warm-start issues. When an unseen domain appears, we need the score immediately, instead of retrain the model on the whole data. Second, as the number of nodes is huge, if the embedding is given by a look-up table for every nodes in the graph, the computation would be a bottleneck. Thus we plan to avoid embedding nodes based on IDs. Third, as we has clear classification tasks, we are looking for an end-to-end approach to take the graph information into supervised training simultaneously, instead of two-stage embedding-supervision procedure.

2.1 Notations

We use the notation u_i to denote the feature vector for an user. We use d_j to denote the feature vector for a domain. We use y_j to denote the label of domain d_j. We use the index $j = 1, \cdots, L$ to denote the index of the labeled domains. We further define a function concat$(\cdot, \cdot)$, which concatenates two row vectors into one. We use X^T to denote the transpose of a matrix X. We use θ to denote all the trainable model parameters for a neural network.

2.2 Related Works

In this section, we briefly review several inductive contextual graph-based semi-supervised deep learning methods, and show how they can be applied into our domain classification task. In general, graph-based semi-supervised learning methods relies on the assumption that connected nodes tend to have similar labels. By this assumption, [32] summarized that the loss function for graph-based semi-supervised learning can be decomposed into two part: the supervised loss part (fitting constraint) and the graph-based unsupervised regularization part (smoothness constraint). [32] systematically summarized most of the non-deep existed graph-based semi-supervised learning method, including Learning with local and global consistency [33] and Manifold regularization [4]. It then presented a semi-supervised learning framework called Planetoid (Predicting Labels And Neighbors with Embeddings Transductively Or Inductively from Data) that learns graph embedding during the training of a supervised task. Authors further proposed both the transductive and inductive version of their algorithm, and compared them with several widely used semi-supervised methods [32]. Figure 1 shows the inductive version of the Planetoid with an example our domain label prediction task, where the features are passed into a feed-forward neural network for both predict the domain label and the graph context. The transductive version is similar, except it trains a look-up table for each domain as embedding, instead of the intermediate output of a neural network (a parameterized function of input feature vectors). In out context, the supervised loss is the label prediction loss for each domain, and the unsupervised loss is defined as the prediction loss for the existence of each domain in its context, where the context is defined for the nodes share the same label, or the nodes appear close to each other in the random walk on the graph based on DEEPWALK [26].

To be more specific, the right-most network block in 1 used in [32] is a single-layer network with sigmoid activation and w_c is the row for node c in the weight matrix, which makes the loss function for Planetoid-I to be:

$$\begin{aligned} G_{Planetoid-I}(\theta) &= L_s + L_u \\ L_s &= -\frac{1}{L}\sum_{i=1}^{L} \log p(y_i|d_i) \\ L_u &= \lambda\, \mathbb{E}_{i,c,\gamma} \log \sigma(\gamma w_c^T h(d_i)) \end{aligned}$$

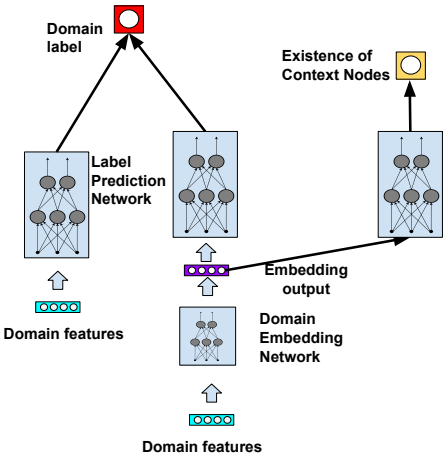

Figure 1: Network architecture for Planetoid-I.

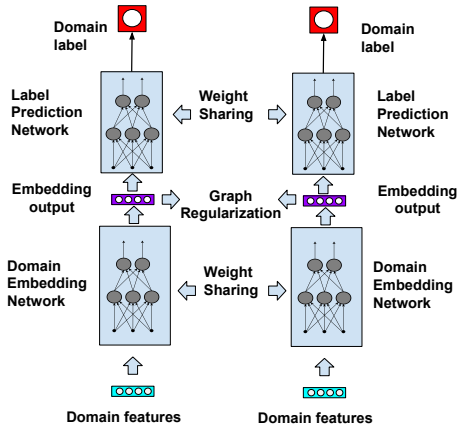

Figure 2: Network architecture for neural graphical machines

where γ is a binary random indicator determines if node with index c, i are similar or not; $p(y_i|d_i)$ is the output of the left three building blocks, representing the predicted probability of true label from the classification neural network. h represents the building block at the middle bottom, which generates the embedding for the node by applying a parametric function on the input feature. λ is the hyper-parameter that controls the trade-off for the fitting constraint and smoothing constraint.

The neural graph machine [5] is a deep learning based extension of label propagation, which imposes a non-linear smoothing constraint by regularizing the intermediate output of a hidden layer of neural networks. In out example, the supervised loss is still the predicting loss for the domain label, while the unsupervised smooth constraint is the average distance between connected domains.

$$
\begin{aligned}
G_{NGM}(\theta) &= L_s + L_u \\
L_s &= -\frac{1}{L}\sum_{i=1}^{L} \log p(y_i|d_i) \\
L_u &= \lambda_1 \sum_{i,j\in\mathcal{E}_{LL}} w_{d_i,d_j} d(h(d_i), h(d_j)) + \\
&\quad \lambda_2 \sum_{i,j\in\mathcal{E}_{LU}} w_{d_i,d_j} d(h(d_i), h(d_j)) \\
&\quad \lambda_3 \sum_{i,j\in\mathcal{E}_{UU}} w_{d_i,d_j} d(h(d_i), h(d_j))
\end{aligned}
$$

where $d(\cdot,\cdot)$ is a distance function for a pair of vector, and [5] suggests either $l1$ or $l2$. $p(y_i|d_i)$ has same meaning as for Planetoid-I, and $h(d_i)$ is the node embedding that defined as the intermediate output of the second laster layer. $\mathcal{E}_{LL}$, $\mathcal{E}_{LU}$ and $\mathcal{E}_{UU}$ defines the node pair that both labeled, only one labeled, and both unlabeled. $\lambda_1, \lambda_2, \lambda_3$ are hyper-parameters control the smoothing constraint for different label types.

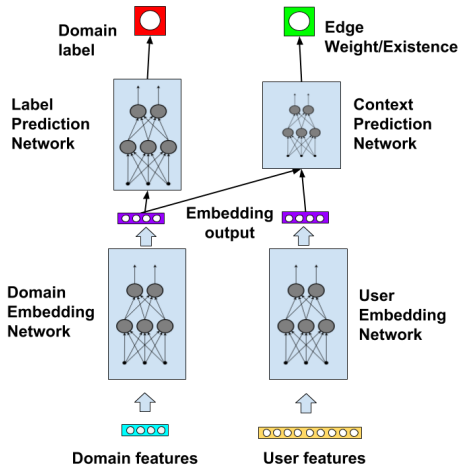

Figure 3: The network structure for the HELP.

3 THE HELP

3.1 Neural Network Structure

Figure 3 shows the network structure of the HELP for user-domain network. Inspired by the neural graphical machines [5], which impose a smoothing constraint on the intermediate output of a feed forward neural network, we propose a new network architecture with four building blocks that can handle two different type nodes. The two building blocks, h_d, h_u, at the bottom of figure 3 represents two feed forward neural network block, with the input as the contextual features of domain and user, and the output as the embedding for domain and user. Two "embedding" building blocks do not share any parameter, and there is no constraint on the input/output shape.

After the "embedding" building blocks, we define the other two building blocks. The first is the label prediction block for domain label prediction, which we defined as f. It takes the embedding $e_d = h_d(d_i)$ of the given domain as input, and output the probability

$f(e_d)$ that the given domain would be labeled 1 by human checker. The other is the "context" block g, which "predicts" the context of the graph. To be more specific, it is a block of feed forward neural network that computes the distance $g(e_u, e_u)$ of between the user and the domain, given the embedding of both of them from the embedding blocks.

During the training stage, the inputs are the pairs of the user-domain. Inspired by [5], our proposed objective function function can be also decomposed into a neural network cost (supervised) and the label propagation cost (unsupervised) as follows:

$$G_{HELP}(\theta) = \sum_{j=1}^{L} L_s(f(h_d(d_j)); \theta) + \lambda \sum_{i,j} L_u(w_{u_i,d_j}, h_d(d_i), h_u(u_i))$$

The first part, the supervised loss, is the cross-entropy for the binary label of domains:

$$L_s(f(h_d(d_j)) = y_j \log(f(h_d(d_j)) + y_j \log(1 - f(h_d(d_j))$$

The second part, the graph regularization loss, is defined as:

$$L_u(w_{u_i,d_j}, h_d(d_i), h_u(u_i)) = w_{u_i,d_j} \cdot d^2_{u_i,d_j} + (1 - w_{u_i,d_j}) \cdot \max(0, m - d_{u_i,d_j})^2$$

where $d_{u_i,d_j} = \sqrt{1 - g(\text{concat}(h_d(d_i), h_u(u_i)))}$, and m is a tunable, fixed margin parameter. Having a margin indicates that unconnected pairs that have the distance beyond this margin will not contribute to the loss. This loss is used in Siamese network, to distinguish a given pair of images [14]. Instead of using L2 distance of the output of an embedding network/feature extractor, we use a separate neural network block to generate "similarity score" for each pair, and use one minus such score as the distance metric.

In out experiment, the input contextual features are numerical vector, thus we only consider the fully-connected neural networks. f is a 2-layer fully connected neural network with output shape $(16, 1)$; h_d and h_u are 3-layer fully connected neural networks with output shape $(96, 64, 32)$ (note they do not share parameters); g is a 2-layer fully connected neural network with output shape $(16, 1)$.

During the training stage, in each epoch, all the labeled domain are passed, and user-domain pairs are sub-sampled due to the huge number of pairs. In each iterations in the epoch, the total loss is computed, and the gradient based on the total loss is back-propagated to the whole network, including f, g, h_u, and h_d, simultaneously. During the domain classification (predicting) stage, it requires no extra re-training: only the domain feature is used.

Notice here the network structure is for illustration, and designed for user-domain bipartite graph. It can be adapted to multiple type of nodes, with multiple smoothing constraints for more than one edge type.

4 EXPERIMENTS

4.1 Labels of Domains

The labels used in the experiments are generated manually according to some internal guideline. We consider three different "dimensions": each dimension stands for a certain type of domain. Table 1 shows the summary statistics of each label.

Table 1: Summary Statistics for Labeled Domains

Label Type	Total Size	# of Positive
dimension1	5498	1094
dimension2	6399	748
dimension3	1781	477

4.2 Metric

In the experiments, we considered a binary classification problem, thus following metrics are considered. The first metric is the area under Receiver Operating Characteristic curve (AUROC). The curve is plotted with the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The AUROC is defined as the area below the ROC curve. It can be explained as the expectation that a uniformly drawn random positive is ranked before a uniformly drawn random negative.

The second metric is the area under the Precision-Recall curve (AUPRC). The curve is plotted with the precision (true positives over the sum of true positives and false positives) against the recall (true positives over the sum of true positives and false negatives) at various threshold settings. Actually we are more in favor of AUPRC in comparison to PRAUC due to the following reasons. First, the classes for all the three label types are imbalanced. It has been shown that in the imbalanced data set, PR curve is more informative [27]. To be more specific, as there are much more negative samples than positive ones, the true negative examples will overwhelm the comparison in ROC, while will not influence PRC. The second reason is we mainly focus on finding the positive (the domains labeled as 1). The PRC mainly reflect the quality of retrieval of the positives and its value is not invariant when we change the baseline, while the AUC does not.

4.3 Features

For domains, we collected 29 features, which include multiple base summary statistics (e.g. number of likes), and some score generated from other model. For users, we collected 129 features, which mainly are user activity statistics in the past .We do not disclose the details of features as it does not influence understanding the proposed algorithm and the following experiments.

We sub-sampled 2.4 million English-speaking users at Facebook for this offline experiment, with the domains that have at least one interaction with the sampled users in last 7 days. The bipartite graph contains 14.46 Million user-domain edges.

4.4 The User-Domain Graph

Figure 4 visualize a user-domain graph. Each edge is considered as undirected, containing two information: the interaction type, and

Table 2: Sample size for user, domain, and their interactions (edges).

type	size
Domain	241, 205
User	2, 433, 581
Edge	14, 460, 336

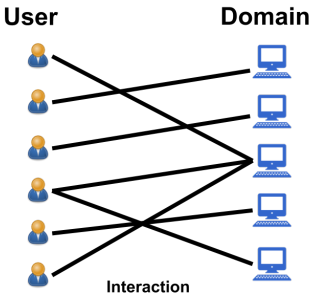

Figure 4: An illustration of user-domain interaction graph.

the count of such interaction in last 7 days. In this study, we only focus on the Resharing as the interaction type. Thus the weight of each edge represents the number of reshares for the given user for the URLs from the given domain.

The experimental data is generated on 10/27/2017, which means the graph is based on the user-domain interaction statistics from 10/20/2017 to 10/27/2017.

5 BENCHMARKS

We consider following algorithms as benchmarks for HELP:

- Label Propagation algorithm (LP) by [34], which only use the graph information. It is not surprising to see it has much worse performance compared other methods use the more informative contextual features. We report this only to show demonstrate much information contains in the graph.
- Multi-layer Perceptron (MLP), which is a fully connected feed-forward neural network using only the feature information.
- Planetoid-I (Predicting Labels And Neighbors with Embeddings Transductively Or Inductively from Data, Inductive Version) by [32], with domain-domain graph compressed from user-domain graph.
- Neural Graph Machine (NGM) by [5], with domain-domain graph compressed from user-domain graph.

As we don't have explicit domain-domain graph, we construct it by compressing the user-domain graph. we construct the domain-domain graph by:

(1) For domain d_i and domain d_j, find the set of users U have edges for both domains.

(2) For $u_k \in U$, define $sim_k^{d_i,d_j} = \min(e_{u_k,d_i}, e_{u_k,d_j})$.

(3) Finally define the edge between d_i, d_j as

$$e_{d_i,d_j} = \sum_{u_k \in U} sim_k^{d_i,d_j}.$$

There are multiple way to compress the user-domain graph to domain-domain graph. We have experimented multiple strategies, but does not show significant difference. As this is not the main focus of this study, we only choose the most straightforward one.

5.1 Optimization

All the neural network models are trained by Adam optimizer [13], with initial learning rate 0.001, and decayed with ratio 0.1 for every 20 epochs. We set the weight decay as 10^{-5}. We train each model 60 epochs. We train each network 10 times and report the average of each performance metric. as this can stabilize the results by reducing the impact of randomness in initialization and training [12].

We also experimented warm-start reported in [32]. However, this does not improve the performance. So the supervised and unsupervised part are trained simultaneously.

6 CLASSIFICATION PERFORMANCE

6.1 Experiment Results

Though two metric are reported, we mainly focus on the PRAUC, as we mainly want to improve the quality of retrieval for positive samples. See detailed discuss ion section 4.2.

Table 3: The predictive performance on testing set for dimension1 domain label. All the values are in 10^{-2} scale.

Model	AUROC	PRAUC
LP	85.7	71.0
MLP	95.1	83.3
PLANETOID-I	95.1	83.8
NGM L1	**95.3**	83.5
NGM L2	95.1	82.9
HELP	95.2	**84.2**

Table 3 shows the predictive performance when predicting if a domain should be labeled as a dimension1 domain. The AUCROC does not have noticeable difference for all deep learning based algorithms. For PRAUC, Planetoid-I and NGM with L1 regularization slightly improved the performance, and HELP achieved the best performance.

Table 3 shows the predictive performance when predicting if a domain should be labeled as a dimension1 domain. Similar to previous experiment, the AUCROC does not have noticeable difference, which may due to the severe imbalance of the positive/negative samples. For PRAUC, the HELP significantly improved the benchmark MLP by 1.3% absolute increment. The Planetoid-I have small improvement compared to MLP, while other semi-supervised method does not show any noticeable improvement.

Table 5 shows the predictive performance when predicting if a domain should be labeled as a dimension3 domain. Different

Table 4: The predictive performance on testing set for dimension2 domain label. All the values are in 10^{-2} scale.

Model	AUROC	PRAUC
LP	87.1	67.7
MLP	95.6	81.6
PLANETOID-I	95.6	81.9
NGM L1	95.6	80.9
NGM L2	95.7	81.5
HELP	**96.3**	**82.9**

Table 5: The predictive performance on testing set for dimension3 domain label. All the values are in 10^{-2} scale.

Model	AUROC	PRAUC
LP	71.9	50.1
MLP	82.2	58.1
PLANETOID-I	82.2	60.2
NGM L1	82.6	63.3
NGM L2	82.2	62.9
HELP	**82.6**	**64.9**

from previous two labels, all the semi-supervised learning methods significantly improve the PRAUC, with at least 2% absolute improvement. One of the most convincing reason is the dimension3 data is much smaller than dimension1/dimension2 dataset, which is usually considered as the case that in favor of the semi-supervised method than purely supervised methods. The HELP model achieved best performance for both AUROC (0.4% absolute improvement) and PRAUC (6.8% absolute improvement).

6.2 Comparison of Unsupervised Loss

There are many loss functions can be applied for "context prediction" in the graph-based neural networks. In this section, we investigated the performance for different several variations of the HELP with different semi-supervised loss function.

6.2.1 Weighted Graph. Then we first consider commonly used supervised loss functions for edge prediction as the graph regularization.

After generates the embedding for an user e_u and a domain e_d, we concatenate two embedding into one:

$$e_{\text{concat}} = \text{concat}(e_u, e_d)$$

and directly feed it into a feed-forward neural network g to predict the edge for this user-domain pair:

$$\hat{w}_{u,d} = g(e_{\text{concat}})$$

In this setting, the label is the weight of the edge (i.e. number of reshares in the past week). We considered the following loss functions:

- L1 (least absolute deviations regression):

$$L(\vec{w}, \hat{\vec{w}}) = ||\vec{w} - \hat{\vec{w}}||_1$$

- L2 (least squares regression):

$$L(\vec{w}, \hat{\vec{w}}) = ||\vec{w} - \hat{\vec{w}}||_2^2$$

- SmoothL1: L1 loss is not strongly convex thus the solution is less stable compared to L2 loss, while L2 loss is sensitive for the outliers and vulnerable to exploding gradients[9, 15]. SmoothL1 loss, also known as the Huber loss, is a combination of L1 and L2 loss which enjoys the advantages from both of them [9]. It is implemented in PyTorch [24]:

$$L(\vec{w}, \hat{\vec{w}}) = \begin{cases} 0.5(\vec{w} - \hat{\vec{w}})^2, & ||\vec{w} - \hat{\vec{w}}||_1 < 1 \\ ||\vec{w} - \hat{\vec{w}}||_1, & ||\vec{w} - \hat{\vec{w}}||_1 >= 1 \end{cases}$$

6.2.2 Unweighted Graph. We also considered the unweighted graph. The only difference from 6.2.1 is, instead of predict the weight of the edge, we dichotomized the weighted edge into a unweighted binary edge. For instance, we defined there is an edge between user u_i and domain d_j, is the user reshare some link from domain d_j more than twice a week. For simplicity, we assume the target $w_{u,d}$ is a binary variable, and the output from the neural network is bounded in $[0, 1]$, which can be interpreted as the probability of the existence of an edge within this user-domain pair. As the target in this setting is binary, we considered the following loss functions:

- CrossEntropy: this is one of the most common loss in classification:

$$L(\vec{w}, \hat{\vec{w}}) = (\vec{1} - \vec{W}) \log(1 - \hat{\vec{w}}) + \vec{W} \log(\hat{\vec{w}})$$

We also consider the embedding distance based loss functions. These functions does not inputing the embedding into a new block of neural network. Instead, it only relies on the distance between the user and the domain embedding e_u, e_d, and **binary indicator** of the existence of the edge $w_{u,d}$.

- Contrastive: this is the loss decreases the energy of like pairs and increase the energy of unlike pairs [7, 14]. Here we define the energy as one minus the output of the graph regularization building block. Recall that the output of the graph regularization building block represents the predicted existence of the edge between the given user-domain pair. We simply set the margin m to be 0.2.

$$\begin{aligned} d &= \sqrt{1 - \hat{\vec{w}}} \\ L(w, \hat{\vec{w}}) &= wd^2 + (1 - w)\max(0, m - d)^2 \end{aligned}$$

- CosineEmbed: we consider the cosine embedding loss implemented in PyTorch [24]:

$$L(w_{u_i,d_j}, e_{u_i}, e_{d_j}) = \begin{cases} 1 - \cos(e_{u_i}, e_{d_j}), & w_{u_i,d_j} = 1 \\ \cos(e_{u_i}, e_{d_j}) & w_{u_i,d_j} = 0 \end{cases}$$

- L1Embed: we also consider the L1 and L2 distance metric used in neural graphical machines [5]:

$$L(w_{u_i,d_j}, e_{u_i}, e_{d_j}) = w_i ||e_{u_i}, e_{d_j}||_1$$

- L2Embed:

$$L(w_{u_i,d_j}, e_{u_i}, e_{d_j}) = w_i ||e_{u_i}, e_{d_j}||_2^2$$

For easier comparison, we cluster these loss function into 4 categories:

Table 6: The predictive performance for HELP with different unsupervised loss on testing set for dimension2 domain label. All the values are in 10^{-2} scale.

Loss	AUROC	PRAUC
Contrastive	96.3	82.9
CosineEmbed	95.6	82.1
L1Embed	95.2	81.6
L2Embed	95.3	81.4
L1	96.0	82.7
L2	95.8	82.1
SmoothL1	95.9	82.5
CrossEntropy	96.1	82.8
MLP	95.6	81.6

Table 7: The performance for different loss function when considering dimension2 label.

Table 6 shows the performance of the HELP model with different unsupervised loss. Among all the loss choices, the HELP with contrastive loss achieves both the best performance for AUROC and PRAUC. The other three embedding based loss, CosineEmbed, L1Embed and L2Embed, achieves worse performance. This may be explained by the flexible distance evaluation. For contrastive loss we used here, we generate the distance from a feed forward neural network with the embedding from both user and domain as input, instead of a fixed commonly used distance metric like cosine distance. This makes the distance selection more flexible.

In addition, we observe the L1Embed and L2Embed is noticeably worse than CosineEmbed and Contrastive, and they does not show any improvement compared to simple MLP. This might due to the L1/L2 losses only "pull" the connected pair closer, while both CosineEmbed and Contrastive loss not only "pull" the connected pair closer, but also "push" the unconnected pair farther away, and therefore improves the learning of the embedding.

For the classification based loss (L1, L2, SmoothL1, and CrossEntropy), we observed all of them has improvement compared to the benchmark MLP. The L2 loss has slightly worse performance compared th L1 and SmoothL1, a combination of L1 and L2 loss. This might due to some extreme weight in the edge, which make too strong impact when training the network. Furthermore, when edges are treated unweighted by thresholding weighted edge, the performance is slightly improved. Similar to previous explanation, we believe such discretization improve the performance by avoid the outliers in the edge weights. A potential solution of it would be truncate the loss for unweighted edge, and we leave it for future work.

7 UNSUPERVISED LEARNING

As discussed above, we do not have explicit label for each users. However, we define some ad-hoc labels for each user to assess the effectiveness of the user embedding, a side-produce in the HELP model.

7.1 Visualization of Embedding

We visualize the embedding for users, which is the side-product of the HELP model.

To avoid information leakage/over-fitting during the training, we generate the graph with the interactions 1 week after the training data. In other word the graph is generated by the interactions between user and domain from 10/27/2017 to 11/03/2017. In addition, the user features/domain labels in our visualization are also collected one week after the collecting date of the experiment data.

We investigate and visualize the users that might be "vulnerable" to dimension2 domains, which we defined as the active users with frequent interaction with some dimension2 domains. To be more specific:

- For each type of interaction (e.g. clicking the link), we first select the users that have more than 5 such interactions during the whole evaluating week as active users.
- Among such users, if the user is more than 5 such interaction with domains that labeled as dimension2 domain, we define this user as a vulnerable (positive) user.
- In visualization, we use the red (positive) nodes to represent the vulnerable users, while using blue (negative) nodes for the remaining active users.
- As there are much less positive samples, we down sampled the negative samples to relative same size as positive samples.

In this section, we studied the five different interaction types, including:

- Click: clicking of the link.
- Reshare: resharing the link.
- Wow: Clicking the Wow button for the link.
- Angry: Clicking the Angry button for the link.

We compared the user embedding generated from the HELP, and the raw features. We use t-SNE to reduce the dimension to 2, while maintaining the Euclidean distance between nodes for both raw features [20] and the generated embedding from the HELP. We simply used the t-SNE function with default parameter in sklearn [25]. Then we plot each nodes on 2-D space, with color represents if the node is a vulnerable use or not.

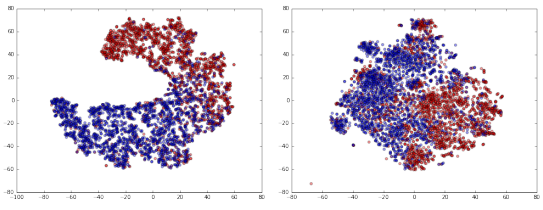

Figure 5: Click. The left figure is for the embedding from the HELP; the right figure is for the raw features.

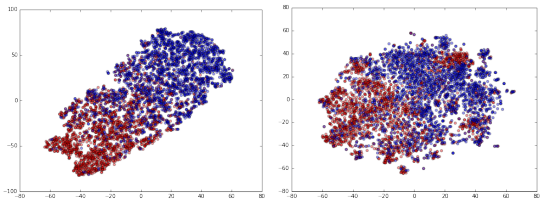

Figure 6: Reshare. The left figure is for the embedding from the HELP; the right figure is for the raw features.

Figure 5 and 6 shows the visualization comparison for Click and Reshare. For both Click and Reshare, we can observe a passable pattern for the separation of blue/red nodes even for the raw features. Though most of the blue nodes are on the one side, there are still many regions that blue and red nodes are mixed. However, the embedding from the HELP further pulled the users of different type further away. We can observe very clear separation boundary for two type of users.

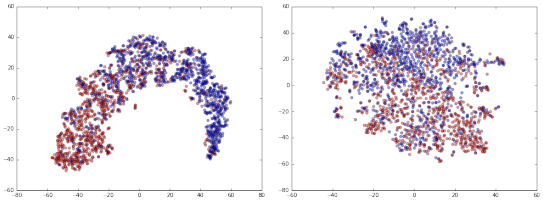

Figure 7: Wow. The left figure is for the embedding from the HELP; the right figure is for the raw features.

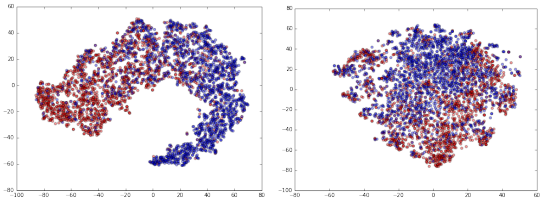

Figure 8: Angry. The left figure is for the embedding from the HELP; the right figure is for the raw features.

Figure 7 and 8 shows the visualization comparison for Wow and Angry. For these interaction types, the raw features did a bad job in separating two different type of users. However, the embedding from the HELP still achieves satisfactory performance in separating two type of users.

In conclusion, the HELP generates embedding for users as a side-product. Our visualization results suggest such user-level embedding can help other tasks, like user-level clustering.

8 DISCUSSION

In this work, we propose HELP, a graph-based semi-supervised deep learning method for graphs with heterogeneous type of node. We demonstrated its performance with several domain classification tasks at News Feed at Facebook. One potential future direction is multi-tasks prediction to predict different type of label simultaneously. The most promising and important direction is, we can extend the network architecture by stacking a multiple-output prediction layer on the second last layer, which output a vector of probability for multiple labels. This can be done by extending the supervised loss with multiple label type. It has following benefits: first the model size can be compressed as we only need to train one model for multi-labels. Second, the embedding generated in this network contains information for different label type, thus is more informative and can be potentially used as a general "reputation embedding" for a domain.

Another interesting direction is allowing different type of edge between nodes. In our experiments, we only consider the "resharing interaction" edges. Different type of edge can be included to further improve the performance of the semi-supervised approach. In addition, we may use weighted combination of multiple interaction types as the weight in graph.

We directly concatenated two embedding and then feed it into the network block to estimate the similarity for each pair. Instead of concatenating, several different approaches can be applied to combine the embedding of the domain-user pair, which may further improve the performance of the HELP. For example we may consider the element-wise product/difference of two embedding vectors.

There are also several minor changes may further improve the performanceof the HELP. We set margin $m = 0.2$ in an ad-hoc manner for the contrastive loss, which can be further investigated. We can also extend the EmbedL1/EmbedL2 loss by imitating the contrastive loss that including penalization for the unconnected pair with close distance. Due to the limited space, we leave this as our future work.

9 ACKNOWLEDGEMENTS

Authors would like to thank the Facebook News Feed team for the help during the project and the insightful feedback.

REFERENCES

[1] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, and Mohammed Zaki. 2006. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*.

[2] A. Babu, Liu A., and J Zhang. 2017. News Feed FYI: New Updates to Reduce Clickbait Headlines. https://newsroom.fb.com/news/2017/05/news-feed-fyi-new-updates-to-reduce-clickbait-headlines/. (2017).

[3] Lars Backstrom. 2016. Serving a Billion Personalized News Feeds. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*. 469. DOI: http://dx.doi.org/10.1145/2835776.2835848

[4] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research* 7, Nov (2006), 2399–2434.

[5] Thang D Bui, Sujith Ravi, and Vivek Ramavajjala. 2017. Neural Graph Machines: Learning Neural Networks Using Graphs. *arXiv preprint arXiv:1703.04818* (2017).

[6] Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C Aggarwal, and Thomas S Huang. 2015. Heterogeneous network embedding via deep architectures. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 119–128.

[7] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 539–546.

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

[9] Ross Girshick. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*. 1440–1448.

[10] 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*. ACM, 855–864.

[11] 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.

[12] Cheng Ju, Aurélien Bibaut, and Mark J van der Laan. 2017. The Relative Performance of Ensemble Methods with Deep Convolutional Neural Networks for Image Classification. *arXiv preprint arXiv:1704.01664* (2017).

[13] Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[14] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, Vol. 2.

[15] Roger Koenker and Kevin F Hallock. 2001. Quantile regression. *Journal of economic perspectives* 15, 4 (2001), 143–156.

[16] A. Lada, J. Li, and S. Ding. 2017. News Feed FYI: New Signals to Show You More Authentic and Timely Stories. https://newsroom.fb.com/news/2017/01/news-feed-fyi-new-signals-to-show-you-more-authentic-and-timely-stories/. (2017).

[17] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.

[18] J. Lin and Guo S. 2017. News Feed FYI: Reducing Links to Low-Quality Web Page Experiences. https://newsroom.fb.com/news/2017/05/reducing-links-to-low-quality-web-page-experiences/. (2017).

[19] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications* 390, 6 (2011), 1150–1170.

[20] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

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

[22] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.

[23] Andrew Y Ng, Michael I Jordan, Yair Weiss, and others. 2002. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems* 2 (2002), 849–856.

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[26] 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.

[27] Takaya Saito and Marc Rehmsmeier. 2015. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one* 10, 3 (2015), e0118432.

[28] 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.

[29] Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 817–826.

[30] Lei Tang and Huan Liu. 2009. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 1107–1116.

[31] Lei Tang and Huan Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23, 3 (2011), 447–478.

[32] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting Semi-Supervised Learning with Graph Embeddings. In *International Conference on Machine Learning*. 40–48.

[33] Denny Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.

[34] Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. (2002).