MONET: DEBIASING GRAPH EMBEDDINGS VIA THE METADATA-ORTHOGONAL TRAINING UNIT

Anonymous authors
Paper under double-blind review

ABSTRACT

Are Graph Neural Networks (GNNs) fair? In many real world graphs, the formation of edges is related to certain node attributes (e.g. gender, community, reputation). In this case, standard GNNs using these edges will be biased by this information, as it is encoded in the structure of the adjacency matrix itself. In this paper, we show that when metadata is correlated with the formation of node neighborhoods, unsupervised node embedding dimensions learn this metadata. This bias implies an inability to control for important covariates in real-world applications, such as recommendation systems.

To solve these issues, we introduce the Metadata-Orthogonal Node Embedding Training (MONET) unit, a general model for debiasing embeddings of nodes in a graph. MONET achieves this by ensuring that the node embeddings are trained on a hyperplane orthogonal to that of the node metadata. This effectively organizes unstructured embedding dimensions into an interpretable topology-only, metadata-only division with no linear interactions. We illustrate the effectiveness of MONET though our experiments on a variety of real world graphs, which shows that our method can learn and remove the effect of arbitrary covariates in tasks such as preventing the leakage of political party affiliation in a blog network, and thwarting the gaming of embedding-based recommendation systems.

1 INTRODUCTION

Graph embeddings – continuous, low-dimensional vector representations of nodes – have been eminently useful in network visualization, node classification, link prediction, and many other graph learning tasks (10). While graph embeddings can be estimated directly by unsupervised algorithms using the graph’s structure (e.g. 22; 26; 13; 23), there is often additional (non-relational) information available for each node in the graph. This information, frequently referred to as node attributes or node metadata, can contain information that is useful for prediction tasks including demographic, geo-spatial, and/or textual features.

The interplay between a node’s metadata and edges is a rich and active area of research. Interestingly, in a number of cases, this metadata can be measurably related to a graph’s structure (19), and in some instances there may be a causal relationship (the node’s attributes influence the formation of edges). As such, metadata can enhance graph learning models (28; 18), and conversely, graphs can be used as regularizers in supervised and semi-supervised models of node features (29; 11). Furthermore, metadata are commonly used as evaluation data for graph embeddings (8). For example, node embeddings trained on a Flickr user graph were shown to predict user-specified Flickr “interests” (22). This is presumably because users (as nodes) in the Flickr graph tend to follow users with similar interests, which illustrates a potential causal connection between node topology and node metadata.

However, despite the usefulness and prevalence of metadata in graph learning, there are instances where it desirable to design a system to avoid the effects of a particular kind of sensitive data. For instance, the designers of a recommendation system may want to make recommendations independent of a user’s demographic information or location.

At first glance, this may seem like an artificial dilemma – surely one could just avoid the problem by not adding such sensitive attributes to the model. However, such an approach (ignoring a sensitive attribute) does not control for any existing correlations that may exist between the sensitive metadata
and the edges of a node. In other words, if the edges of the graph are correlated with sensitive metadata, then any algorithm which does not explicitly model and remove this correlation will be biased as a result of it. Surprisingly, almost all of the existing work in the area \cite{28,31} has ignored this important realization.\footnote{While preparing this manuscript, we have become aware of a recent independent result \cite{5} in this area. In contrast to that work, we use a substantially different methodology which offers guarantees about the debiasing process.}

In this work, we seek to refocus the discussion about graph learning with node metadata. To this end, we propose a novel, general technique for extending graph representations with metadata embedding dimensions while debiasing the remaining (topology) dimensions. Specifically, our contributions are the following:

1. The Metadata-Orthogonal Node Embedding Training (MONET) unit, a novel GNN algorithm which jointly embeds graph topology and graph metadata while enforcing independence between the two embedding spaces.
2. Analysis which proves that a naive approach (adding metadata embeddings without MONET) leaks metadata information into topology embeddings, and that the MONET unit does not.
3. Experimental results on real world graphs which show that MONET can successfully “debias” topology embeddings while relegating metadata information to separate metadata embeddings.

2 Preliminaries

Early graph embedding methods involved dimensionality reduction techniques like multidimensional scaling and singular value decomposition \cite{8}. In this paper we use graph neural networks trained on random walks, similarly to DeepWalk \cite{22}. DeepWalk and many subsequent methods first generate a sequence of random walks from the graph, to create a “corpus” of node “sentences” which are then modeled via word embedding techniques (e.g. word2vec \cite{17} or GloVe \cite{21}) to learn low dimensional representations that preserve the observed co-occurrence similarity.

Let $W$ be a $d$-dimensional graph embedding matrix, $W \in \mathbb{R}^{n \times d}$, which aims to preserve the low-dimensional structure of a graph ($d << n$). Rows of $W$ correspond to nodes, and node pairs $i, j$ with large dot-products $W_i^T W_j$ should be structurally or topologically close in the graph. As a concrete example, in this paper we consider the debiasing of a recently proposed graph embedding using the GloVe model \cite{6}. Its training objective is:

$$\text{GloVe}(U, V, a, b | C) = \sum_{i,j \leq n} f_\alpha(C_{ij})(a_i + b_j + U_i^T V_j - \log(C_{ij}))^2, \quad (1)$$

where $U, V \in \mathbb{R}^{n \times d}$ are the “center” and “context” embeddings, $a, b \in \mathbb{R}^{n \times 1}$ are the biases, $C$ is the walk-distance-weighted context co-occurrences, and $f_\alpha$ is the loss smoothing function \cite{21}. We use the GloVe model throughout to demonstrate topology/metadata embeddings and metadata-orthogonal training, though the MONET unit we propose is broadly generalizable.

Notation. In this paper, given a matrix $A \in \mathbb{R}^{n \times d}$ and an index $i \in 1, \ldots, n$, $A_i$ denotes the $d \times 1$ $i$-th row vector of $A$. Column indices will not be used. $0_{n \times d}$ denotes the $n \times d$ zero matrix, and $\| \cdot \|_F$ denotes the Frobenius norm.

3 Metadata Embeddings and Orthogonal Training

In this section we present MONET, our proposed method for separating and controlling the effects of metadata on topology embeddings. First, we begin by outlining the straightforward extension of metadata to traditional embedding models in Section 3.1. Next, in Section 3.2 we prove that such a simple model will leak information from the metadata to the topology (structural) embeddings. Then, in Section 3.3 we present MONET, our proposed approach for training embeddings of a graph’s structure which are not correlated with metadata. Finally, we conclude with some analysis of MONET in Section 3.4.
3.1 Jointly Modeling Metadata & Topology

A natural first approach to modeling the effects of metadata on the graph is to explicitly include the node metadata as part of a node embedding model. For instance, to extend Eq. (1), in addition to $U$ and $V$ (the “topology embeddings”), we can consider the node metadata $M$ directly ($M \in \mathbb{R}^{n \times m}$, row vector $M_i$ is the metadata for node $u_i$). We then can define metadata embeddings $X = MT_1$, $Y = MT_2$, where $T_1, T_2$ are trainable transformations, and propose the concatenations $[U, X]$ and $[V, Y]$ as full-graph representations. The GloVe loss with metadata embeddings is:

$$\text{GloVe}_{\text{meta}}(U, V, T_1, T_2, a, b|C, M) = \frac{1}{2} \sum_{i,j \leq n} f_a(C_{ij})(a_i + b_j + U_i^T V_j + X_i^T Y_j - \log(C_{ij}))^2.$$  \hspace{1cm} (2)

While in this paper we demonstrate metadata embeddings within the GloVe model, they can be incorporated in any dot-product-based graph neural network. For instance, the well-known DeepWalk loss, which is based on word2vec (17), would incorporate metadata embeddings as follows:

$$\text{DeepWalk}_{\text{meta}}(U, V, T_1, T_2|W, M) = - \sum_{i,j \in W} \log(U_i^T V_j + X_i^T Y_j) - \sum_{k \in K_i} \log(-U_i^T V_k - X_i^T Y_k).$$  \hspace{1cm} (3)

Above, $W$ is the set of context pairs from random walks, and $K_i$ is a set of negative samples associated with node $i$. For GloVe, DeepWalk, and many other GNNs, this approach augments the overall graph representation by concatenating metadata-learned dimensions.

However, this naïve approach does not guarantee that the topology embeddings converge to be statistically independent of the metadata embeddings. Suppose that the metadata (like demographic information) are indeed associated with the formation of links in the graph. In this case, any algorithm which does not explicitly model and remove the association will be biased as a result of it. In the next section we formalize this concept, which we call metadata leakage.

3.2 Metadata Leakage in Graph Neural Networks

Here, we formally define metadata leakage for general topology and metadata embeddings, and show how it can occur even in embedding models with separate metadata embeddings. All proofs appear in the Appendix.

**Definition 1.** The metadata leakage of metadata embeddings $Z \in \mathbb{R}^{n \times d_z}$ into topology embeddings $W \in \mathbb{R}^{n \times d}$ is defined $\mathcal{ML}(Z, W) := \|Z^TW\|^2_F$. We say that there is no metadata leakage if and only if $\mathcal{ML}(Z, W) = 0$.

Without a more nuanced approach, metadata leakage can occur even in embedding models that explicitly include the metadata, like Eqs. (2) and (3). To demonstrate this, we consider for simplicity a reduced metadata-aware GloVe loss with $W := U = V \in \mathbb{R}^{n \times d}$ as the sole topology embedding and $T := T_1 = T_2 \in \mathbb{R}^{m \times d_z}$ as the sole metadata transformation parameter. With $\tilde{Z} := MT$, the reduced loss is:

$$\text{GloVe}_{\text{meta}}^*(W, T, a|C, M) = \frac{1}{2} \sum_{i,j \leq n} (a_i + a_j + W_i^T W_j + Z_i^T Z_j - \log(C_{ij}))^2$$  \hspace{1cm} (4)

We now show that under a random update of the GloVe$_{\text{meta}}^*$ model in Eq. (4), the expected metadata leakage is non-zero. Specifically, let $(i, j)$ be a node pair from $C$, and define $\delta_W(i, j)$ as the incurred Stochastic Gradient Descent update $W' \leftarrow W + \delta_W(i, j)$. Suppose there is a “ground-truth” metadata transformation $B \in \mathbb{R}^{m \times d_B}$, and define ground-truth metadata embeddings $\tilde{Z} := MB$, which represent the “true” dimensions of the metadata effect on the co-occurrences $C$. Define $\Sigma_B := BB^T$ and $\Sigma_T := TT^T$. With expectations taken with respect to the sampling of a pair $(i, j)$ for Stochastic Gradient Descent, define $\mu_W := E[W_i]$ and $\Sigma_W := E[W_iW_i^T]$. Define $\mu_M, \Sigma_M$ similarly. Then our main Theorem is as follows:

**Theorem 1.** Assume $\Sigma_W = \sigma_W I_d$ for $\sigma_W > 0$, $\mu_W = 0_{d \times 1}$, and $\mu_M = 0_{m \times 1}$. Suppose for some fixed $\theta \in \mathbb{R}$ we have $\log(C_{ij}) = \theta + Z_i^T \tilde{Z}_j$. Let $(i, j)$ be a randomly sampled co-occurrence pair and $W'$ the incurred update. Then if $E[M_iW_i^T] = \beta \in \mathbb{R}^{m \times d}$, we have

$$E[\mathcal{ML}(Z, W')] \geq 2\|T^T(\Sigma_M(\Sigma_B - \Sigma_T)) + (n - \sigma_W)I_m\| \beta^2_F.$$  \hspace{1cm} (5)
Here, we introduce the Metadata-Orthogonal Node Embedding Training (MONET) unit for training joint topology–metadata graph representations $[W,Z]$ without metadata leakage. MONET explicitly prevents the correlation between topology and metadata, by using the Singular Value Decomposition (SVD) of $Z$ to orthogonalize updates to $W$ during training.

**MONET.** The MONET unit is a two-step algorithm applied to the training of a topology embedding in a neural network, and is detailed in Algorithm 1. The input to a MONET unit is a metadata embedding $Z \in \mathbb{R}^{n \times d_z}$ and a target topology embedding $W \in \mathbb{R}^{n \times d}$ for debiasing. Then, let $Q_Z$ be the left-singular vectors of $Z$, and define the projection $P_Z := I_{n \times n} - Q_Z Q_Z^T$. In the forward pass procedure, debiased topology weights are obtained by using the projection $W^\perp = P_Z W$. Similarly, $W^\perp$ is used in place of $W$ in subsequent GNN layers. In the backward pass, MONET also debiases the backpropagation update to the topology embedding, $\delta_W$, using $\delta^\perp_W = P_Z \delta_W$. Figure 1 illustrates a geometric interpretation of the MONET algorithm.

Straightforward properties of the SVD show that MONET directly prevents metadata leakage:

**Theorem 2.** Using Algorithm 1, $\mathbb{E}[\mathcal{L}(Z, W)] = \Omega(n \|T^T \beta\|_F^2)$ as $n \to \infty$.

We note that in this work we have only considered linear metadata leakage; debiasing nonlinear topology/metadata associations is an area of future work.

**Implementation (MONET$_G$).** We demonstrate MONET in our experiments by applying Algorithm 1 to Eq. 2. We denote this model MONET$_G$, and create it as follows. We orthogonalize the input and output topology embeddings $U,V$ with the summed metadata embeddings $Z := X + Y$. By linearity, this implies $Z$-orthogonal training of the summed topology representation $W = U + V$. We note that working with the sums of center and context embeddings is the standard way to combine these matrices (21). Figure 2 shows an illustration of MONET$_G$.

3.4 ANALYSIS

Here we address some brief remarks about the algorithmic complexity of MONET, and the interpretation of its parameters.

**Algorithmic Complexity.** The bottleneck of MONET occurs in the SVD computation and orthogonalization. In our setting, the SVD is $O(n d_z^2)$ (27). The matrix $P_Z$ need not be computed to perform orthogonalization steps, as $P_Z W = W - Q_Z (Q_Z^T W)$, and the right-hand quantity is $O(n d_z)$ to compute. Hence the general complexity of the MONET unit is $O(n d_z \max\{d, d_z\})$.

**Metadata Parameter Interpretation.** The $i,j$ terms in the sum of the loss for GloVe models with metadata (GloVe$_{meta}$ and MONET$_G$) involve the dot product $X_i^T Y_j = M_i^T T_1 T_2^T M_j$. That expans-
Both prediction and training for $W$ occur on a hyperplane orthogonal to $Z$. In the forward pass, $W$ is projected onto the $Z$-orthogonal plane. When an update $\delta W$ is proposed, it too is projected, resulting in the best metadata-orthogonal update. This allows $W$ to explore the space of unknown latent structure without bias from $Z$.

$sion$ suggests that the matrix $\Sigma_T := T_1 T_2^T$ contains all pairwise metadata dimension relationships. In other words, $\Sigma_T$ gives the direction and magnitude of the raw metadata effect on log co-occurrence, and is therefore a way to measure the extent to which the model has captured metadata information. We will refer to this interpretation in the experiments that follow. An important experiment will show that applying the MONET algorithm increases the magnitude of $\Sigma_T$ entries.

4 Metadata De-Biasing Experiments

Here we empirically demonstrate Theorems 1 and 2 by confirming the following hypotheses:

1. H1. The MONET unit can remove leakage of metadata information from topology embeddings, so that the topology embeddings cannot predict the metadata.

2. H2. The MONET unit can make recommender systems more robust to abuse by removing malicious user directions from rating graphs.

For all embedding models, we use the center-context embedding sum of topology embeddings $W := U + V$ as the graph representation for task evaluation. Note that some standard baselines (e.g. DeepWalk) do not incorporate metadata and therefore only train topology embeddings. All GloVe-based models are trained with TensorFlow (1) using the AdaGrad optimizer (12) with initial learning rate 0.05. DeepWalk models were trained using the gensim software (24).

4.1 Quantitative Experiment: Political Blogs Network

To address H1, illustrating Theorem 1 and the effect of MONET de-biasing, we embed the effect of political ideology on a blogger network (3). The political blog network has has 1,107 nodes corresponding to blog websites, 19,034 hyperlink edges between the blogs (after converting the graph to be undirected), and two clearly defined, equally sized communities of liberal and conservative bloggers.

Design and Methods. In this experiment all graph neural network models were trained on 5 iterations across 80 random walks per node of length 40 with context window size 10. Topology embeddings had dimension 16, and metadata embeddings had dimension 2. We measure embedding bias by the Macro-F1 score of a linear SVM predicting political party from the embeddings, using a LIBLINEAR

\footnote{Available within the Graph-Tool software (20).}
Table 1: Macro-F1 scores from political blog network classifications using graph topology embeddings only. MONET is successful in removing all metadata information from the topology embeddings – the links in the graph are no longer an effective predictor of political party. Comparison of the metadata transformation product $\Sigma_T$ between GloVe-meta and MONET shows MONET allows for considerably more metadata information learning. Finally, only MONET removes metadata leakage to precision error (recall $M\mathcal{L}$ is a Frobenius norm).

<table>
<thead>
<tr>
<th>Model</th>
<th>F1 (mean ± std)</th>
<th>$\Sigma_T = T_1T_2^T$ (mean ± std)</th>
<th>$M\mathcal{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>95.59% ± 0.07%</td>
<td>N/A</td>
<td>2743.9 ± 36.7</td>
</tr>
<tr>
<td>GloVe</td>
<td>95.94% ± 0.07%</td>
<td>N/A</td>
<td>6598.0 ± 200.1</td>
</tr>
<tr>
<td>GloVe-meta</td>
<td>88.33% ± 0.60%</td>
<td>$\begin{pmatrix} 0.108 \pm 0.006 &amp; -0.106 \pm 0.004 \ -0.108 \pm 0.009 &amp; 0.106 \pm 0.006 \end{pmatrix}$</td>
<td>1827.6 ± 289.7</td>
</tr>
<tr>
<td>MONET$_G$</td>
<td><strong>49.30%</strong> ± 0.60%</td>
<td>$\begin{pmatrix} 0.180 \pm 0.006 &amp; -0.178 \pm 0.006 \ -0.181 \pm 0.008 &amp; 0.179 \pm 0.006 \end{pmatrix}$</td>
<td><strong>0.018</strong> ± 0.002</td>
</tr>
</tbody>
</table>

Figure 3: PCA of political blog graph embeddings. (a): Party separation clearly visible on standard GloVe embeddings. (b): Party separation reduces when GloVe-meta captures some metadata information. (c): Party separation disappears with MONET$_G$ orthogonalized training.

For each embedding set, we compute the mean and standard deviation Macro-F1 over 10 independent classification repetitions, each trained using half of the node labels sampled at random. To assess metadata information leakage, we also track the metadata dimension importance matrix $\Sigma_T := T_1T_2^T$, recalling its interpretation from Section 3.

Results. Table 1 shows that the baselines DeepWalk and GloVe are highly effective at predicting political party, and therefore biased. This is unsurprising, as these methods are trained without metadata information, and were originally intended to encode low-dimensional structure like that present in this data set. The bias in DeepWalk and GloVe embeddings is further seen in their metadata leakage values, computed using political party one-hot vectors as metadata embeddings.

Considering the embedding models with metadata embeddings, we find that, interestingly, GloVe-meta’s topology embeddings are still able to predict political party with 88.3% Macro-F1. Also, as predicted by Corollary 1, GloVe-meta’s metadata leakage remains $O(n)$. This shows that simply concatenating metadata embeddings is not sufficient to isolate the metadata effect. In contrast, MONET$_G$ achieves random Macro-F1 and no metadata leakage (under machine precision), demonstrating that on this data, the MONET unit is necessary to de-bias the blog embeddings from political party. This contrast is seen in two other ways. First, there is a noticeable increase in $\Sigma_T$ magnitude when MONET is used, implying that GloVe-meta metadata embeddings are not capturing all possible metadata information. Second, as seen in Fig. 3, the 2-dimensional PCA plots of the GloVe-meta embeddings still show political party separation, whereas the MONET$_G$ PCA dimensions reveal strong mixing.
Table 2: Results from the shilling attack experiment. Attackers attempt to insert 10 items in the top-20 recommendations of a target video. The results show that MONET can best mitigate the effect of an attack under incomplete information. We note that there is an implicit trade-off between de-biasing and maintaining correlation with the original (biased) embeddings.

### 4.2 EXPERIMENT 2: THWARTING ATTACKS ON GRAPH-BASED RECOMMENDATION SYSTEMS

In this experiment we address H2, investigating the effectiveness of MONET to defend against a shilling attack (9) against graph-embedding based recommender systems (30). In a shilling attack, a number of users act together to artificially increase the likelihood that a particular influenced item will be recommended for a particular target item.

**Data.** In a single repetition of this experiment, we inject an artificial shilling attack into the MovieLens 100k dataset. The raw data is represented as a bipartite graph with 943 users, 1682 items, and a total of 100,000 ratings (edges). Each user has rated at least 20 items. At random, we sample 10 items into an influence set $S_I$, and a target item $i_t$ to be attacked. We take a random sample of 5% of the existing users to be the set of attackers, $S_A$. We then create a new graph, $G_{\text{attacked}}$ which in addition to all the existing ratings, contains new ratings from each attacker $\in S_A$ to each item $\in S_I$ as well as the target video. (Note that this corresponds to several varieties of behavior including both incentivizing formerly good users, and account takeover.)

**Design and Methods.** For each embedding method, we perform random walks through the new bipartite graph $G_{\text{attacked}}$. As we wish to study item recommendation, in the random walks, we simply remove user nodes each time they are visited (so the walks contain only pairwise co-occurrence information over items). With any given network embedding, we measure its bias by the number of influence items in $S_I$ in top-20 embedding-nearest-neighbor list of $i_t$. As metadata, we allow MONET to know the per-movie attacker rating count for each attacked movie. However, to better demonstrate real-world performance, we only allow 50% (randomly sampled) attackers from the original 5% sample to be “known” when constructing these metadata. As non-debiasing baselines, we compare against DeepWalk and Glove. As de-biasing baselines, we applied a generalized correlation removal framework developed for removing word embedding bias (25; 4). Specifically, we tried two approaches to “debias” the GloVe embedding of the MovieLens graph – as the “gender” embedding direction, we tried both (a) the most attacked movie vector and (b) the sum of attacked movie vectors. All methods use 128 dimensional topology embeddings and are trained on 100 random walks per node, each walk of length 5.

**Results.** As seen in Table 2, the topology embeddings from MONET are the least biased by a large margin, letting on average only 1.2 influence items in the top-20 neighbors of $i_t$. Interestingly, we note that this behavior occurs even though the majority of observed co-occurrences for the algorithm had nothing to do with the attack in question, and only the known 50% of attackers were used to construct the metadata. In contrast, all baselines (including those that explicitly model the attacker metadata) left at least around half of the attacked items in the top-20 list. To measure the extent to which de-biased embeddings retain the original recommendation signal, we compute the pair-wise embedding distances of each method, and compute their Pearson correlation to the standard GloVe embeddings. We find that MONET embedding distances achieve high correlation (0.83) to the original distances, showing that with MONET it is possible to nearly nullify a shilling attack while preserving most of the signal from the true, un-attacked ratings. We note that the max-attack-embedding baseline higher embedding distance correlation, but this method let many more attacked items (on average) into higher ranks. This reveals a trade-off between embedding de-biasing and prediction efficacy which has also been observed in other contexts (30).

<table>
<thead>
<tr>
<th>Model</th>
<th>Manipulated Items in Top-20 (mean ± std dev)</th>
<th>Embedding Distance Correlation w/GloVe</th>
</tr>
</thead>
<tbody>
<tr>
<td>DeepWalk</td>
<td>9.9 ± 0.40</td>
<td>0.385 ± 0.005</td>
</tr>
<tr>
<td>GloVe</td>
<td>9.8 ± 0.76</td>
<td>1.000 ± 0.000</td>
</tr>
<tr>
<td>GloVe$_{meta}$</td>
<td>9.8 ± 0.476</td>
<td>0.852 ± 0.002</td>
</tr>
<tr>
<td>NLP Debiasing (sum)</td>
<td>8.1 ± 1.7</td>
<td>0.566 ± 0.004</td>
</tr>
<tr>
<td>NLP Debiasing (max)</td>
<td>4.9 ± 3.4</td>
<td><strong>0.990 ± 0.003</strong></td>
</tr>
<tr>
<td>MONET$_G$</td>
<td><strong>1.2 ± 1.7</strong></td>
<td>0.831 ± 0.004</td>
</tr>
</tbody>
</table>

Available: [http://files.grouplens.org/datasets/movielens/ml-100k/](http://files.grouplens.org/datasets/movielens/ml-100k/)
5 Related Work

Though graph learning is an immense field, a minority of unsupervised graph embedding techniques involve graph metadata. To our knowledge, none of these techniques involve either metadata orthogonalization or the capacity to learn arbitrary metadata transformations. (32) is a matrix factorization approach which uses a shared node embedding matrix to factor both the graph adjacencies and the raw metadata in a joint loss, with a tunable parameter to control the influence of the metadata loss. Similarly, (16) pre-computes a metadata similarity matrix and trains shared center-context embedding matrices on the metadata similarities and random walk similarities. In contrast, we learn the direction and effect of metadata as neural network parameters, and we separate those parameters into unique embedding dimensions. (28) and (31) are matrix factorization approaches which factor an approximation to the co-occurrence matrix into equally-sized metadata and topology embeddings, and were built mainly for text metadata. Their approaches enforce metric space similarity and dimensional homogeneity between metadata and topology representations, restrictions that we do not rely on and are ill-suited to the setting with multiple types of arbitrarily-sized metadata. (14) constructs random walks that traverse between the original graph and the metadata freely, an approach which runs counter to our ability to separate out the effects of metadata on graph adjacencies. (18) introduce a version of the stochastic block model with metadata priors, and show that the estimated posteriors yield insight into the influence of metadata on the graph. However, this model estimates a community partition and in/out-community probabilities - it does not yield embeddings either of the node topology or the node metadata. There has been work in Natural Language Processing on removing gender bias from word embeddings (e.g. 4), but these methods operate with pre-computed embeddings and rely on identification of gendered terminology.

Additionally, there has been a wealth of work studying semi-supervised learning with graphs (e.g. 29) and graph convolutional networks (e.g. 21511), which use graph metadata as features. While most semi-supervised and supervised neural networks for graphs indirectly produce embeddings that in some cases can be identified with feature and topology dimensions, they are trained as part of prediction or label propagation tasks. Therefore, the topology embeddings are free to correlate with features to the extent that this serves the loss function - there is no explicit separation of topology and metadata dimensions. In this paper, we have studied the benefits of metadata orthogonalization in the unsupervised setting, and we leave the exploration of our techniques in the semi-supervised and supervised settings to future work.

6 Conclusion

In this work, we have shown that unsupervised training of graph embeddings induces bias from important graph metadata. We proposed a novel solution to address this problem – the Metadata-Orthogonal Node Embedding Training (MONET) unit. The MONET unit is the first graph learning technique for training-time debiasing of embeddings, using orthogonalization. Our experimental results using real datasets showed that MONET is able to encode the effect of graph metadata in isolated embedding dimensions, and simultaneously remove the effect from other dimensions. This has immediate practical applications, which we illustrate by mitigating a simulated shilling attack on a real dataset of movie ratings.

This work was meant to introduce the basic principles underlying the need for the MONET technique, and show its utility in a shallow graph neural network (GloVe). While we used a shallow network for instructional purposes, we note that MONET is generalizable, and MONET units can be used to debias any set of embeddings from another set during training. Subsequent research can explore the use of MONET in deeper networks and potentially semi-supervised models or graph convolutional networks. As MONET’s SVD calculation can be expensive with large graphs and large embedding dimensions, future research could be in assessing the effect of SVD approximations, or training algorithms that utilize caching of previous metadata embedding SVDs to speed up training.

References


**A APPENDIX**

**Proposition 1.** Under the assumptions of Theorem 1, we have

\[
E[Z^T \delta_W(i,j)] = 2 \left[ \Sigma_M (\Sigma_B - \Sigma_T) + \sigma W I_m \right] \beta. \tag{6}
\]

**Proof.** Derivatives of GloVe*meta* yield that the i-th row of \(\delta_W(i,j)\) is \(d_{ij} W_j^T\), where

\[
d_{ij} = \log(C_{ij}) - Z_i^T Z_j - W_i^T W_j - a_i - a_j \\
= \theta + \tilde{Z}_i^T \tilde{Z}_j - Z_i^T Z_j - W_i^T W_j - a_i - a_j. \tag{7}
\]
Similarly the $j$-th row is $d_{ij}W_j^T$, and all other rows are zero vectors. Hence
\[\mathbb{E}[Z^T \delta_W(i,j)] = \mathbb{E}[Z_id_{ij}W_j^T] + \mathbb{E}[Z_jd_{ij}W_i^T]. \quad (8)\]
We derive the second term on the right-hand side of Equation (8); the first term follows by symmetry.
Note first that $\mathbb{E}[Z_i(\theta - a_i - b_j)W_j^T] = 0$ by independence and centering assumptions. Second:
\[\mathbb{E}[Z_iW_i^TW_jW_j^T] = \mathbb{E}[M_iM_i^TTT] = \mathbb{E}[M_iM_i^T] = \mathbb{T} \Sigma_M \Sigma_T \beta \]
by independence. Third:
\[\mathbb{E}[Z_iZ_i^TW_jW_j^T] = \mathbb{E}[M_iM_i^TTT] = \mathbb{E}[M_iM_i^T] = \mathbb{T} \Sigma_M \Sigma_T \beta \]
by independence, and similarly $\mathbb{E}[Z_iZ_i^TW_jW_j^T] = \mathbb{T} \Sigma_M \Sigma_B \beta$. Combining these with Equation (7),
we have
\[\mathbb{E}[Z_iW_i^T] = \mathbb{T} \Sigma_M (\Sigma_B - \Sigma_T) - \sigma_W I_d \beta. \quad (9)\]
Applying symmetry to the second term in Equation (8) completes the proof.

A.1 PROOF OF THEOREM 1

Proof. Proposition 1 gives $\mathbb{E}[Z^T \delta_W(i,j)] = 2 \mathbb{T} \Sigma_M (\Sigma_B - \Sigma_T) + \sigma_W I_m \beta$. Second, note that $\mathbb{E}[M_iW_i^T] = \beta \Rightarrow M^TW = n\beta$ and thus $Z^TW = T^T M^TW = nT^T \beta$. Recalling that $W' = W + \delta_W(i,j)$ we have
\[\mathbb{E}[Z^TW'] = 2 \Sigma_M (\Sigma_B - \Sigma_T) + (n - \sigma_W) I_m \beta. \]
Applying Jensen’s Inequality completes the proof.

A.2 PROOF OF THEOREM 2

Proof. Consider metadata embeddings $Z \in \mathbb{R}^{n \times d_Z}$ and, as in the MONET algorithm, define
the projection $P_Z = I_{d_Z} - Q_ZQ_Z^T$, where $Q_Z$ are the left-singular vectors of $Z$. By properties of the
SVD, $Z^T Q_ZQ_Z^T = Z_T$, and hence $Z^TP_Z = 0_{n \times d_Z}$. This means that $Z^T W^\perp = Z^T \delta_W^\perp = 0_{d_Z \times d_Z}$,
which completes the proof by definition of metadata leakage.