MELISSA: Semi-Supervised Embedding for Protein Function Prediction Across Multiple Networks

Kaiyi Wu
kaiyi.wu@tufts.edu
Department of Mathematics
Tufts University
177 College Ave
Medford, MA 02155
USA

Di Zhou
di.zhou@tufts.edu
Department of Computer Science
Tufts University
177 College Ave
Medford, MA 02155
USA

Donna Slonim
slonim@cs.tufts.edu
Department of Computer Science
Tufts University
177 College Ave
Medford, MA 02155
USA

Xiaozhe Hu
xiaozhe.hu@tufts.edu
Department of Mathematics
Tufts University
177 College Ave
Medford, MA 02155
USA

Lenore Cowen*
cowen@cs.tufts.edu
Department of Computer Science
Tufts University
177 College Ave
Medford, MA 02155
USA

ABSTRACT

Several popular methods exist to predict function from multiple protein-protein association networks. For example, both the Mashup algorithm, introduced by Cho, Peng and Berger, and deepNF, introduced by Gligorijević, Barotand, and Bonneau, analyze the diffusion in each network first, to characterize the topological context of each node. In Mashup the high-dimensional topological patterns in individual networks are canonically represented using low-dimensional vectors, one per gene or protein, to yield the multi-network embedding. In deepNF, a multimodal autoencoder is trained to extract common network features across networks that yield a low-dimensional embedding. Neither embedding takes into account known functional labels; rather, these are then used by the machine learning methods applied after embedding. We introduce MELISSA (MultiNetwork Embedding with Label Integrated Semi-Supervised Augmentation) which incorporates functional labels in the embedding stage. The function labels induce sets of "must link" and "cannot link" constraints which guide a further semi-supervised dimension reduction to yield an embedding that captures both the network topology and the information contained in the annotations. We find that the MELISSA embedding improves on the Mashup embedding and outperforms the deepNF embedding in creating more functionally enriched neighborhoods for predicting GO labels for multiplex association networks in both yeast and humans.
Availability: MELISSA is available at
https://github.com/XiaozheHu/melissa

*To whom correspondence should be addressed

BCB '23, September 3-6, 2023, Houston, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9386-7/22/08.
https://doi.org/10.1145/3535508.3545542

ACM Reference Format:
Kaiyi Wu, Di Zhou, Donna Slonim, Xiaozhe Hu, and Lenore Cowen. 2023. MELISSA: Semi-Supervised Embedding for Protein Function Prediction Across Multiple Networks. In *14th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '23), September 3–6, 2023, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3535508.3545542

1 INTRODUCTION

In 2016, Cho et al. introduced their groundbreaking Mashup algorithm for function prediction by integrating information across multiplex biological networks [7]. Mashup consists of three steps: in the first step, the Random Walk with Restart (RWR) algorithm [23] is run separately for each node for each network; each node is represented by its diffusion state vector in each network. In the second step, a low-dimensional embedding is constructed to minimize the distance to all the individual network-specific vectors globally. In the last step, these global low-dimensional feature vectors are then passed to classifiers such as k-nearest neighbors [6] or support vector machines [3] in order to do the functional label prediction.

However, in the Mashup paradigm, we notice that biological knowledge, encoded in the form of the GO [8] functional labels, is only incorporated in the last step of this process. The embedding itself, constructed in the first and second steps, is entirely unsupervised and comes only from the topological structure of the networks. The same is true of deepNF [14], an alternative method that constructs the embedding using a multi-modal deep autoencoder. Therefore, the motivation of this paper is to incorporate the known GO labels in the low-dimensional embedding to improve its quality and, thus, improve on the state-of-the-art Mashup and deepNF algorithms.

One popular approach to include the biological label information is *semi-supervised* graph embedding methods [1, 2, 26, 28]. The biological information, in the form of functional labels, can give rise to a set of "must-link" (ML) constraints and a set of "cannot-link" (CL) constraints [1]. These constraints then guide the embedding procedure toward a result that encodes both the inherent structure of the data as well as the information contained in the functional annotations.

In the protein function domain, genes have multiple functional labels, both noisy and incomplete [10]. So it can be challenging to generate both types of constraints. The incompleteness is a challenging problem: "cannot link" constraints require proteins known *not* to be involved in some function. But in our setting, the fact that a gene lacks a particular functional annotation does not necessarily indicate that the gene does not play a role in that function. It may very well be that it simply has not been experimentally observed yet.

One possible way to handle the above challenges, on a gene-by-gene basis, would be to use some of the sets of specially curated negative GO annotations that the community has begun to construct [12, 25]. Our method, MELISSA, instead takes a different approach. We augment each network with a sparse set of artificial new nodes, which also are involved in the embedding step. Intuitively, the new artificial nodes represent the "centers" of a coarse clustering of functional labels. We place ML constraints between the original nodes and these new artificial nodes to bind genes in the training set to their cluster label and encourage them to cluster. At the same time, CL constraints are placed between the artificial nodes themselves to encourage the clusters to separate from each other (See Section 2.4). MELISSA uses a biclustering procedure [9] on the annotation matrix that maps genes to GO labels of appropriate specificity to generate the set of coarse cluster labels that will be assigned as artificial cluster nodes.

After augmentation, the resulting networks now contain positive and negative weights. The standard RWR approach to compute each node's diffusion state vectors cannot be directly applied. Thus, in MELISSA, we adopted a signed version of the graph Laplacian [13, 16, 18] and generalize the diffusion state representations of each node to the networks with both positive and negative weights. Then, the rest of the Mashup pipeline, from dimension reduction to function prediction based on the low-dimensional embedding, proceeds as before.

Once the embedding is formed, a variety of different classification methods can be applied to the embedded space. Because our focus is on improving the information content of the embedding, in this work we pair MELISSA with the simple kNN classifier, where comparing functional label prediction of competing embeddings gives a sense of the functional enrichment in local neighborhoods. As shown below, in this setting, MELISSA improves the overall performance of the functional label prediction task, compared to the original Mashup and the deepNF embeddings, demonstrating its ability. Thus MELISSA can be used to analyze multiple networks constructed by the guilty-by-association property and provide an accurate and scalable framework for network integration and analysis from different experiments.

2 METHODS

MELISSA varies from Mashup by including a step of network augmentation to encode functional information before the embedding and the learning phase. This step requires augmenting the networks with auxiliary cluster nodes which we induce from the available gene annotations. A summary of the procedure is given in Figure 1.

2.1 Preliminaries and Notation

The datasets that we consider consist of a collection of networks $G^i = (V, E^i, w^i)$ with $i \in \{1, \dots N\}$ which share a set of nodes V but each has its own set of edges E^i and the set of edge weights $w^i > 0$. In our work, the nodes correspond to genes that appear in the union of all the networks. The edges E^i correspond to a different type of relationship between pairs of genes for each network. As in the experiment in the original Mashup paper, these relations range from experimental evidence of interaction or association between two genes, to co-expression, among other things (see [7]). Finally, the weights w^i indicate the confidence in the edges being correct.

The adjacency matrices of the graph G^i are denoted by $\boldsymbol{A}^i$, and their probability transition matrices are denoted by $\boldsymbol{T}^i$ where $\boldsymbol{T}^i_{uv}$ denotes the probability that a random walk on graph G^i at vertex v transitions to u in one step. The functional annotations of the genes are represented by the binary matrix $\boldsymbol{B} \in \{0, 1\}^{l \times n}$ where l is the number of distinct labels and $n = |V|$ is the number of the nodes. In $\boldsymbol{B}$, each column corresponds to the set of annotations a gene has been given, and each row corresponds to the set of genes with a given annotation.

2.2 Review of the Mashup Embedding

The original Mashup procedure consists of the following three core steps:

(1) **Diffusion.** On each G^i, a diffusion process is run which creates a RWR matrix representation $\boldsymbol{W}^i \in \mathbb{R}^{n \times n}$ of the network.
(2) **Embedding.** A shared embedding is created using the matrix representations generated in the diffusion step. This is achieved via a singular value decomposition or dictionary learning techniques. Ultimately this gives a d-dimensional vector representation of every node in the dataset.
(3) **Learning.** Once every node in the dataset is represented by a vector, existing function prediction methods can be applied using the embedding and the available annotations.

The original Mashup uses a support vector machine (SVM) for the final learning step. However, we can apply any function prediction method. Therefore, we consider the **Diffusion** and **Embedding** steps to be the main contribution of Mashup and briefly describe them in detail.

The **Diffusion** step generates a matrix representation of each network. There are many ways of generating node embeddings, such as Diffusion State Distance [5, 6], Node2vec [15], and spectral methods [19]. Mashup adopted a Random Walk with Restart (RWR) based approach [17], i.e., for each vertex $u \in G^i$, Mashup iteratively computes the t-step RWR distribution $\boldsymbol{s}^t_u \in \mathbb{R}^n$ as follows, let $\boldsymbol{s}^0_u = \boldsymbol{e}_u$,

$$\boldsymbol{s}^{t+1}_u = (1 - \alpha)\boldsymbol{T}^i \boldsymbol{s}^t_u + \alpha \boldsymbol{e}_u. \tag{1}$$

where $\boldsymbol{e}_u \in \mathbb{R}^n$ is the vector with entry 1 at the u-th index and 0 elsewhere and $0 \leq \alpha < 1$. Following [5–7], the node embedding of u is the *diffusion state* $\boldsymbol{s}^\infty_u$, the stationary distribution at the fixed point of the iteration (1). By stacking the diffusion states $\boldsymbol{s}^\infty_u$ as

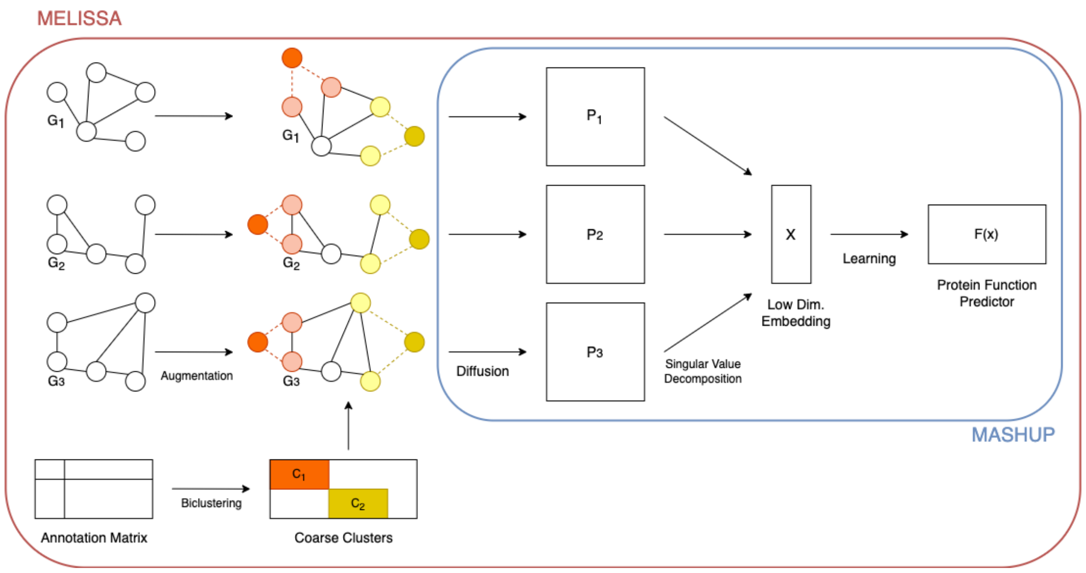

Figure 1: Workflow of MELISSA extending Mashup. Nodes and labels are first biclustered, and each network is augmented with one auxiliary node for each cluster. ML constraints (dashed edges) are added to pull the nodes in the same cluster closer to each other, while CL constraints (solid edges) are added to push the clusters away from each other in the embedding. Then we run the Mashup procedure as summarized in Section 2.2 on each of the augmented networks.

columns, we obtain the diffusion state matrix $\boldsymbol{W}^i$, which is the RWR representation matrix used in Mashup.

Once each set of N networks has been represented by its associated RWR matrix $\boldsymbol{W}^i$, Mashup then combines the matrix representations and constructs a low-dimensional embedding. The original Mashup framework proposed two embedding strategies, the first being a dictionary learning approach and the second being the singular value decomposition (SVD). For the sake of simplicity, we focus on the SVD approach, i.e., the low-dimensional embedding $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ are formed by the scaled largest d left singular vectors of the concatenated matrix $\log \boldsymbol{S}$ where $\boldsymbol{S} = [(\boldsymbol{W}^1)^T, \cdots, (\boldsymbol{W}^N)^T]^T$ and $\log(\cdot)$ denotes the element-wise logarithm. As suggested in [7], for the optimization purpose in the implementation, the SVD-based embedding can be computed by the eigenvalue decomposition of the $n\times n$ matrix $\boldsymbol{R} = \sum_{i=1}^{N} \log(\boldsymbol{W}^i)^T \log(\boldsymbol{W}^i)$. A small constant (e.g., reciprocal of the number of genes) was added to each entry of $\boldsymbol{W}^i$ to avoid taking the log of zero entries. Taking the top d eigenvalues $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \cdots, \lambda_d)$ and eigenvectors $\boldsymbol{U} = (\boldsymbol{u}_1, \cdots, \boldsymbol{u}_d) \in \mathbb{R}^{n\times d}$, the low-dimensional embedding is $\boldsymbol{X} = \boldsymbol{\Lambda}^{1/4}\boldsymbol{U}^T$.

Note that Mashup as described in [7] involves concatenating the diffusion state matrices from different networks vertically when performing the joint factorization; the Mashup code allows us to concatenate either horizontally or vertically. In our experiments, we found the vertical approach always performed better than the horizontal one.

Our key observation is that Mashup uses only network topology to construct its embedding. Our objective is to attain a more functionally meaningful low-dimensional embedding of the networks by augmenting the original networks so that the **Diffusion** and **Embedding** steps are aware of known functional annotations. We hypothesize that the improvements in the embedding transfer to the performance in the **Learning** step.

2.3 Review of the deepNF Embedding

deepNF [14], introduced in 2018, tries to learn a useful low-dimensional embedding of proteins with a multimodal deep autoencoder (MDA) [24] that preserves non-linear network structure across multiple networks characterized by diverse connectivity patterns. In [14] it is shown that deepNF preserves the non-linear network structure with its deep neural network (DNN) architecture in an efficient and scalable manner, and at the same time denoises the links in the networks.

The deepNF method involves the following three steps:

(1) **Pre-processing.** On each network G^i, a RWR procedure is run to create its matrix representation $\boldsymbol{W}^i \in \mathbb{R}^{n\times n}$. Then each RWR matrix is converted into a Positive Pointwise Mutual Information (PPMI) matrix $\boldsymbol{Q}^i \in \mathbb{R}^{n\times n}$ that captures the structural information of the network.
(2) **MDA Embedding.**
A MDA is trained that takes the PPMI matrices as input. A canonical d-dimensional feature representation across the networks is extracted from the middle layer of the MDA.
(3) **Learning.** The middle layer of the MDA which serves as the low-dimensional vector representation of every node in the networks is then fed into function prediction classifiers.

We elaborate on each step in a bit more detail next. In the **preprecessing** step, each RWR matrix $\boldsymbol{W}^i$ is generated in the same way as in the Mashup **Diffusion** step. After that, the PPMI matrix

$\boldsymbol{Q}^i$ for the i-th network is computed as,

$$\boldsymbol{Q}^i_{lm} = \max\left\{0, \log_2\left(\frac{\boldsymbol{W}^i_{lm}\sum_l\sum_m \boldsymbol{W}^i_{lm}}{\sum_l \boldsymbol{W}^i_{lm}\sum_m \boldsymbol{W}^i_{lm}}\right)\right\}.$$

Once each network has been transformed into its information-rich matrix representation, deepNF integrates the PPMI matrices with MDA to construct a low-dimensional feature representation that best approximates all networks. In particular, deepNF first trains a low-dimensional non-linear embedding for each biological network and then concatenates the network embeddings from the previous step into a single hidden layer, allowing the MDA to learn feature representations using all networks. The single bottleneck layer of the MDA is then extracted as the integrated low-dimensional feature representation. Mini-batch stochastic gradient descent with momentum is used to train the MDA.

2.4 Semi-Supervised Embedding via Graph Augmentation

Although the networks $G^1, \cdots, G^N$ have edges that encode interactions and can be used to reconstruct protein functions, we intend to fully utilize any known protein functions from the beginning, even before the **Diffusion** step starts. This can be done by augmenting the original networks using the ML and CL constraints and employing a semi-supervised embedding approach. However, directly adding the ML and CL constraints between the original nodes might outweigh the original edges and, as a result, destroy the original networks' clustering structure. Therefore, we first augment the networks with auxiliary nodes that encode functional information and then apply the constraints to gently enhance the clustering structure without polluting it.

To introduce the auxiliary nodes that encode functional information, we simultaneously bicluster the proteins and the function labels (see Figure 2). Within the resulting biclusters, each pair of proteins has similar function labels, and functional labels are rarely shared across clusters. Therefore, this suggests we introduce one auxiliary node for each cluster. There are several popular algorithms for producing biclusters [20]. MELISSA uses the method of [9], which produces more balanced biclusters for better practical performance. The number of biclusters is a parameter of MELISSA. Given the few labels at specific functional levels, we do not want to choose too many biclusters in our initial study. Thus, we mainly test relatively small numbers of biclusters, e.g., 2, 4, and 8. We also experiment with larger numbers of biclusters for some generic functional levels with many labels. A thorough study of tuning this parameter is a subject of future work.

The next step is to augment the graphs by introducing one auxiliary node for each cluster and adding the ML and CL constraints as suggested in [27]. More precisely, in each network, we link the auxiliary nodes to all nodes in the corresponding clusters and put positive weight on the added edges to pull the nodes in the same cluster close to each other. Those added edges are the ML constraints and we denote its set by E_{ML} with weight $w^+ > 0$. In addition, we add pairwise CL constraints between the auxiliary nodes to push the clusters away from each other in the embedding. The set of those added edges is denoted by E_{CL} with weight $w^- < 0$.

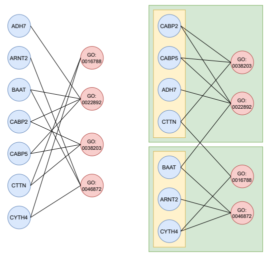

Figure 2: Left: A fragment of the bipartite graph with gene nodes connected to their labeled GO terms, with nodes ordered lexicographically. Nodes on the right are: GO:002678 hydrolase activity; GO:0022892: transmembrane transporter activity; GO:0038203 TORC2 signalling; and GO:0046872 metal ion binding. Right: The same graph with nodes and labels grouped by biclustering, where the transmembrane transporter activity and metal ion binding labels are placed in the same bicluster, with a separate bicluster containing the TORC2 signaling label and the hydrolase activity. The biclusters are highlighted in green and the sets of nodes sharing an auxiliary coarse label are highlighted in yellow.

Note, for the sake of simplicity, we drop the superscript i in this section.

Since the augmented networks become *signed* graphs due to the negatively weighted edges introduced by the CL constraints, we need to look at the so-called *signed* graph Laplacians [13, 16, 18] to understand the effects of the added ML and CL edges. Let $\overline{\delta}_u = \sum_{v\in\mathcal{N}_u}|w_{uv}|$ be the *signed* weighted degree of node u, where $\mathcal{N}_u$ denotes the neighboring nodes of node u, and $\overline{\boldsymbol{D}} = \text{diag}(\overline{\delta}_1, \cdots, \overline{\delta}_n)$ be the *signed degree matrix*. Then the *signed graph Laplacian* is defined by $\overline{\boldsymbol{L}} = \overline{\boldsymbol{D}} - \boldsymbol{A}$. Let $Y \in \mathbb{R}^{d\times n}$ be the d-dimensional embedding of the nodes, as usual, the energy function $\mathcal{E}(Y)$ is defined as

$$\mathcal{E}(Y) = \text{Trace}(Y\overline{\boldsymbol{L}}Y) = \sum_{\ell=1}^{d}\left[\sum_{e(u,v)\in E} w_{uv}(Y_{\ell u} - Y_{\ell v})^2 \right.$$
$$\left. + \sum_{e(u,v)\in E_{\text{ML}}} w^+(Y_{\ell u} - Y_{\ell v})^2 + \sum_{e(u,v)\in E_{\text{CL}}} |w^-|(Y_{\ell u} + Y_{\ell v})^2\right].$$

To minimize the energy $\mathcal{E}(Y)$, for $e(u,v) \in E_{\text{ML}}$, we should have $Y_{\ell u} \approx Y_{\ell v}$ which pulls the nodes in one cluster closer to the corresponding auxiliary node and, therefore, the nodes within the same cluster will be placed closer to each other indirectly. On the other hand, for $e(u,v) \in E_{\text{CL}}$, we expect $Y_{\ell u} \approx -Y_{\ell v}$, which pushes the auxiliary nodes away from each other and, therefore, places the nodes belong to different clusters apart indirectly.

As we can see, the edge weights w^+ and w^- for the ML and CL constraints are parameters of MELISSA. In our initial study, since the edge weights on the original networks are confidence scores ranging between $(0, 1]$, we simply set $w^+ = 1$ and $w^- = 0$ or -1 in our numerical tests to demonstrate the idea of MELISSA. We plan to optimize those weights in our future work.

Once the augmented graphs are constructed, we still want to use the RWR matrix to represent the graph. In MELISSA, we still use the diffusion state matrix. Thus, we need to generalize the diffusion state matrix $\boldsymbol{W}$ to signed graphs. For graphs with only positively weighted edges, from (1), by direct calculation, we have, for $\alpha \in (0, 1)$,

$$\boldsymbol{W} = \alpha \boldsymbol{D} \left(\boldsymbol{D} - (1-\alpha)\boldsymbol{A}\right)^{-1},$$

where $\boldsymbol{D} = \mathrm{diag}(\delta_1, \cdots, \delta_n)$ is the weighted degree matrix with $\delta_u = \sum_{v \in \mathcal{N}_u} \omega_{uv}$ being the usual weighted degree for node u. Thus, a natural generalization to a signed graph is obtained by replacing the weighted degree matrix $\boldsymbol{D}$ with the signed weighted degree matrix $\overline{\boldsymbol{D}}$ and defining the diffusion state matrix for signed graphs as follows,

$$\overline{\boldsymbol{W}} = \alpha \overline{\boldsymbol{D}} \left(\overline{\boldsymbol{D}} - (1-\alpha)\boldsymbol{A}\right)^{-1}. \tag{2}$$

We use this approach to construct $\overline{\boldsymbol{W}}^i$, $i = 1, \cdots, N$, for each augmented graph and then follow Mashup's step to combine those matrix representations to obtain a low-dimensional embedding. Due to the presence of the negatively weighted CL links, the entries of $\overline{\boldsymbol{W}}^i$ might be negative and, therefore, we increase the small constant used in Mashup to avoid taking the log of negative entries $\overline{\boldsymbol{W}}^i$. In our experiments, we set this smoothing constant to be the reciprocal of the number of genes plus $|\min_{u,v} \overline{\boldsymbol{W}}_{uv}|$. Finally, the low-dimensional embedding of MELISSA is constructed the same as the **Embedding** step of Mashup.

2.5 MELISSA

Our proposed MELISSA procedure consists of five core steps as follows, where steps 3-5 duplicate Mashup with augmented networks.

(1) **Biclustering.** Bicluster the annotation matrix $\boldsymbol{B}$ to obtain coarse groupings of the nodes. The number of biclusters is the parameter Nc.
(2) **Graph Augmentation** Add new auxiliary nodes corresponding to the biclusters. We create two types of new edges involving these auxiliary nodes. First, we add "must-link" edges of weight w^+ that connect each original node to the bicluster to which it belongs. Then, we add "cannot link" edges of weight -1 between all pairs of the auxiliary nodes.
(3) **Diffusion.** On each augemented graph $\overline{G}^i$ a diffusion process (2) is run which creates a matrix representation $\overline{\boldsymbol{W}}^i$ of the augmented network.
(4) **Embedding** A shared low-dimensional embedding is created using the matrix representations $\overline{\boldsymbol{W}}^i$. This is achieved via a singular value decomposition or dictionary learning techniques. Ultimately this results in a d-dimensional vector representation of every node in the dataset.
(5) **Learning.** Once every node in the dataset is represented by a vector, existing function prediction methods can be applied. For example, an SVM approach, like Mashup or DeepNF uses, or a (weighted) majority voting classifier by k-nearest neighbors (kNN) (similar to [6]) can be trained using the embedding and the available annotations.

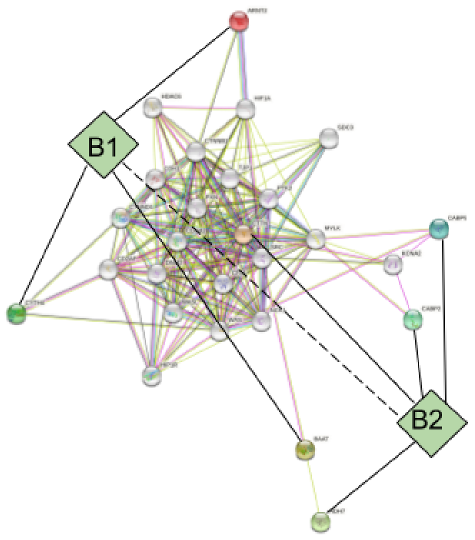

Figure 3: A subset of the augmented STRING network with the two new artificial bicluster nodes from the Figure 2 example added. In particular, B1 represents the bicluster containing ARNT1, BAAT, and CYTH4, and B2 represents the bicluster containing the other four nodes in the Figure 2 example. Solid lines represent the added must-link constraints with positive weight, and the dashed line represents the "cannot link" constraint that attempts to pull the biclusters apart. These constraints help tighten the clustering for the labeled data, but simultaneously pull the labeled data away from unlabeled data in the embedding, making the strength of the edges to the artificial nodes crucial to determining the success of the approach.

3 EXPERIMENTAL SETUP

3.1 Datasets

Our experimental setup matches the setup in the original Mashup paper as much as possible. The set of networks we consider in this paper are the ones used in the original Mashup paper. These are from the STRING database v9.1 [11], excluding links derived from text-mining. In particular, we consider six heterogeneous networks over 6,400 genes with the number of edges varying from 1,361 to 314,013 for yeast, and 18,362 genes with the number of edges varying from 3,717 to 1,544,348 for human. MELISSA, like Mashup, incorporates STRING confidence weights on the edges when computing the diffusion matrix.

The original Mashup paper used GO annotations for human networks and the now-depreciated MIPS annotations for yeast networks. To compare to the original Mashup, we used the same human annotations. For yeast, however, we decided not to use MIPS and instead also consider the GO, so our results are not directly comparable to the results reported in the original Mashup paper for yeast. (Our yeast GO annotations are from the Gene Ontology Consortium [22] (downloaded from FuncAssociate3.0 [4] on 02/12/19)). The GO functional labels are grouped into three distinct functional hierarchies: Biological Process(BP), Molecular

Function(MF), and Cellular Component(CC), where we again mimic Mashup to filter the GO terms to only retain those of intermediate specificity, labeling more than 10 and fewer than 301 genes among the nodes. This label set can be further divided into levels of varying specificity, each containing labels that annotate 11-30, 31-100, and 101-300 genes, respectively [21], filtering on Jaccard index to remove too similar GO labels, paralleling the experiments in the original Mashup paper on human (see below). We consider 9 different functional annotation experiments, parameterized by one of the 3 hierarchies (BP, MF, and CC) times one of the three levels of specificity of GO terms (11-30, 31-100, and 101-300), and construct MELISSA embeddings for each, see below.

3.2 Evaluation

We compare the performance of the original Mashup and deepNF embeddings with MELISSA in predicting GO functional labels in each of the three hierarchies (BP, MF, and CC) on both the human and yeast multi-network collections in 5-fold cross-validation. In the human network, we chose the same set of GO terms as was chosen by MASHUP, and evaluated on the same sets of GO term range specificities (note that some of the GO terms are obsolete in the current version of the GO). In the yeast network, we switched to the GO (the original MASHUP paper used MIPS), but used a similar procedure as MASHUP used in the human network, to remove highly overlapping GO terms. Because our focus is on the embedding part, where Mashup and MELISSA vary, we decided to use the computationally less expensive learning method, kNN, for function prediction as in [6]. We note that it is possible to achieve better performance than kNN with a more sophisticated SVM method [7, 14] (see Discussion section, below).

After obtaining the low dimensional representation of the data, for each node in the test set, we find its k nearest neighbors and then cast a majority vote weighted by the reciprocal of the pairwise distance. If none of the k nearest neighboring genes has a label, we find the single most closely labeled gene and have it vote. The label with the most votes is assigned to the gene as its predicted function, and this is considered correct if it matches at least one of its known annotations. The percentage accuracy is the percentage of nodes given correct annotations. We measure this, F1 score, and area under the precision-recall curve (AUPRC) broken down by both hierarchy (Molecular Function (MF), Biological Process (BP) or Cellular component (CC)) and specificity of GO terms (GO terms that label 11-30, 31-100 and 101-300 nodes in the dataset), for both the yeast and human dataset, matching the experimental setup in [7].

4 RESULTS

We need to set several parameters for the Melissa framework. They are the number of biclusters, embedding dimension, the strength of the ML and CL constraints, and the number of nearest neighbors for majority voting. We set these parameters by comparing Melissa to the original Mashup. We fixed the number of neighbors for the majority voting at 10, as recommended by [6], but tested different numbers of biclusters (in powers of 2) and embedding dimensions (ranging from 25 to 1000 for yeast and 25 to 1200 for human). We primarily looked at weighting ML and CL constraints equally but also considered a version where only ML constraints were considered (and CL constraints were given zero weights. See the Supplement.). Complete results of 5-fold cross-validation across all the parameters we tested between Mashup and MELISSA can be found in the Supplement. For the human trials, we included Mashup trials with the same setup as presented in the experiments in [7], namely concatenating the diffusion state matrices vertically in the **Embedding** step. For the yeast trials, we included both versions of Mashup, i.e., concatenating the diffusion state matrices vertically (denoted as Mashup $W'W$) and horizontally (denoted as Mashup WW'), and observed that vertical concatenation always performed better than the horizontal one.

In [7] the suggested embedding dimension is 500 for yeast and 800 for human, and we tested a range of embedding dimensions at an increment of 25 for both yeast and human in MELISSA trials. Similar to the Mashup case, the performance of MELISSA initially increases as we increase the embedding dimension. After that, the prediction accuracy stops increasing or even decreases as we include the higher embedding dimension, which is possibly affected by noise (see the Supplement). In Table 1 and Table 2, we set the embedding dimension to be 400 for both human and yeast. We note that while setting the MELISSA embedding dimension between $300 - 500$ is usually robust enough, the ideal embedding dimension of MELISSA seems to vary for different species and function labels (see the Supplement for additional comparison of function prediction performance where the best embedding dimension is learned for Mashup and MELISSA in each trial). When we look at the performance in terms of the number of biclusters, in most cases, a smaller number of biclusters (8 or fewer) usually seems to perform better. We also find that for a small number of biclusters, including CL constraints generally improves performance, whereas when the number of biclusters becomes very large, including CL constraints starts to hurt performance seriously. Figure 4 and Figure 5, give two examples of how the function prediction performance of MELISSA with different numbers of biclusters and including CL or not varies for Yeast trials with MF $11 - 30$ functional label and Human trials with BP $101 - 300$ functional labels, respectively. Similar results for other parameter choices appear in the Supplement.

For deepNF embedding training, we adopted the suggested parameters in [14]. The MDA architectures are $[6 \times n, 6 \times 2000, 600, 6 \times 2000, 6 \times n]$, and $[6 \times n, 6 \times 2500, 9000, 1200, 9000, 6 \times 2500, 6 \times n]$ for yeast and human networks respectively. The stochastic gradient descent optimizer uses batch size of 128 for yeast networks and 256 for human networks, with momentum of 0.95 and learning rate of 0.2 for both yeast and human. The middle layers of the MDA are extracted as embeddings and are then fed into the kNN function prediction classifier described above. We ran deepNF with its recommended parameters.

In our experimental setup, where we chose to mimic the experiments run in the original Mashup paper as closely as possible, creating 2 biclusters (and allowing both ML and CL constraints) seems to do quite well across the different hierarchies, GO-term ranges, and yeast and human species. Results with these parameters appear in Table 1 and Table 2. Interestingly, this version of MELISSA consistently outperforms both Mashup and deepNF when measured in accuracy, F1 score, or AUPRC score, with the only exception of

Yeast CC $101 - 300$ trials. However, we note that this experimental setting is not the best to get insight into different choices of biclustering parameters, because Mashup filtered GO terms at each level, removing terms that had Jaccard similarity greater than 0.1 with another category in the same level in order to avoid statistical artifacts arising from overlapping functional categories (See the Supplement for the number and percentage of GO labels that got removed in this step). Thus we are already looking at a sparser subset of GO terms that are too distinct to form good biclusterings.

Additionally, we observe that in this setting, MELISSA is more robust to parameter choices on Yeast networks than on Human networks (see the Supplement). We suspect this is due to the more abundant availability of functional label information for yeast. As noted above, we observe that using a smaller number of clusters, 2 or 4, usually leads to better MELISSA embedding and function prediction results. The optimal set of parameters varies for different GO hierarchies and functional specificity. In the future, we would like to investigate how the functional label specificity or network structure affects the choice of these parameters. Note that we can apply any function prediction method besides weighted majority voting, and a more sophisticated predictor will get more out of the embedding, which at the same time comes with a more expensive computational cost. We noticed that some of the advantages of MELISSA disappear when we use the SVM predictor instead, but again, an SVM predictor will be advantaged in an experimental setting where substantially overlapping GO labels have been removed from the test set.

5 DISCUSSION

The results in section 3 demonstrate that incorporating annotation information at an earlier stage can lead to more informative network embeddings. Augmenting the networks with cluster nodes and introducing ML and CL constraints in the embedding procedure introduces significant structures that are otherwise missed. This creates a more valuable and information-rich node embedding which yields performance gains in the functional enrichment of the local neighborhoods, as evidenced by the improvements in kNN-based functional classification. However, we note that Mashup improves performance by using more sophisticated downstream learning: namely, learning SVMs to discriminate functional classes. The SVM is expressive enough to overcome the generic embedding and performs better on our function prediction task than the Mashup or the Melissa embedding paired with kNN. However, the story is far from complete, and there is much room for exploring new techniques, especially given the wealth of work in semi-supervised methods.

One future direction is to push the annotations into the diffusion process for complete end-to-end utilization of all the information provided from the start. Many methods could be explored by modifying the topology of the networks using constraints. For example, changing the diffusion process to avoid transitioning directly between pairs of proteins with different functional labels. Other methods that could sparsify the networks or introduce new edges could also be investigated.

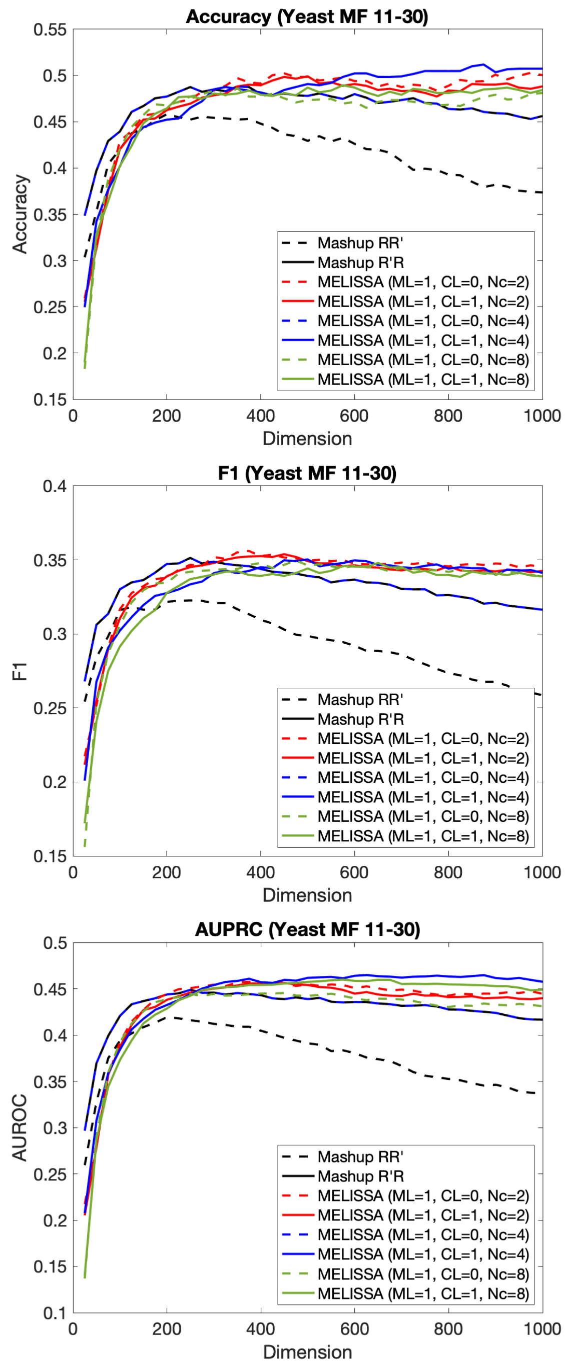

Figure 4: Function prediction comparison with Mashup and MELISSA for Yeast data. The functional labels used are GO labels in the Molecular Function(MF) hierarchy that annotate $10 - 30$ genes in the biological networks. The x-axis is the embedding dimension.

ACKNOWLEDGEMENTS

Thanks to the T-Tripods graduate fellows and the T-Tripods summer reading group on graphs and networks members at Tufts. Thanks

GO HR.	Term Range	Accuracy			F1			AUPRC		
		Mashup	deepNF	MELISSA	Mashup	deepNF	MELISSA	Mashup	deepNF	MELISSA
BP	11-30	46.60%	37.52%	**46.76%**	33.48%	28.82%	**34.48%**	34.47%	26.09%	**35.43%**
BP	31-100	54.13%	46.58%	**55.41%**	37.93%	34.12%	**39.05%**	45.31%	37.58%	**47.04%**
BP	101-300	65.08%	59.00%	**67.34%**	45.81%	43.70%	**46.47%**	64.60%	59.67%	**65.76%**
MF	11-30	47.99%	42.10%	**48.94%**	34.39%	30.15%	**35.26%**	44.31%	37.02%	**45.58%**
MF	31-100	48.93%	43.04%	**50.96%**	35.85%	32.67%	**37.17%**	45.08%	37.58%	**46.58%**
MF	101-300	57.98%	51.38%	**61.13%**	42.73%	39.64%	**43.94%**	58.55%	52.23%	**61.57%**
CC	11-30	70.47%	64.16%	**70.94%**	42.29%	41.07%	**42.75%**	73.74%	66.74%	**74.68%**
CC	31-100	71.73%	65.45%	**72.25%**	44.61%	42.84%	**45.28%**	71.18%	65.74%	**72.13%**
CC	101-300	**78.02%**	75.70%	76.27%	**48.00%**	47.98%	47.83%	**83.40%**	81.90%	81.29%

Table 1: Function prediction performance on yeast PPI network with GO functional labels from the BP, MF, and CC hierarchies based on MELISSA, Mashup, and deepNF data features by majority voting with 10 nearest neighbors in 5-folds cross-validation. The embedding dimension is 400 for Mashup and MELISSA and the MELISSA parameters are: $w^+ = 1$, $w^- = -1$, and 2 biclusters. The deepNF embedding dimension is 600.

GO HR.	Term Range	Accuracy			F1			AUPRC		
		Mashup	deepNF	MELISSA	Mashup	deepNF	MELISSA	Mashup	deepNF	MELISSA
BP	11-30	29.84%	21.78%	**30.44%**	20.76%	16.45%	**21.55%**	15.64%	10.12%	**16.19%**
BP	31-100	33.47%	26.07%	**34.37%**	23.40%	19.18%	**23.91%**	19.03%	13.32%	**20.00%**
BP	101-300	41.86%	34.72%	**43.81%**	31.36%	26.28%	**32.27%**	32.09%	25.64%	**34.23%**
MF	11-30	36.78%	29.97%	**37.67%**	26.60%	22.29%	**27.53%**	28.06%	20.10%	**28.42%**
MF	31-100	38.33%	29.60%	**38.77%**	28.69%	23.66%	**29.11%**	27.70%	19.39%	**28.40%**
MF	101-300	46.51%	33.52%	**48.75%**	35.89%	29.05%	**37.60%**	40.76%	28.10%	**43.31%**
CC	11-30	50.45%	44.32%	**50.70%**	32.02%	29.58%	**32.48%**	46.72%	39.67%	**48.41%**
CC	31-100	49.22%	44.96%	**52.32%**	33.17%	29.69%	**35.10%**	45.41%	40.14%	**47.52%**
CC	101-300	51.91%	45.69%	**52.82%**	35.94%	32.48%	**36.24%**	47.90%	40.74%	**47.93%**

Table 2: Function prediction performance on human PPI network with GO functional labels from the BP, MF, and CC hierarchies based on MELISSA, Mashup, and deepNF data features by majority voting with 10 nearest neighbors in 5-fold cross-validation. The embedding dimension is 400 and the MELISSA parameters are: $w^+ = 1$, CL $w^- = 1$, and 2 biclusters. The deepNF embedding dimension is 1200.

especially to T-Tripods graduate fellow Matthew Werenski who made important suggestions in the early stages of this project. This research was supported by NSF grants DMS-1812503 (to L.C. and X.H.) and the Tufts T-Tripods Institute (NSF grant CCF-1934553) and NSF CC* grant 2018149.

REFERENCES

[1] E. Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.
[2] M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(11), 2006.
[3] A. Ben-Hur, C. S. Ong, S. Sonnenburg, B. Schölkopf, and G. Rätsch. Support vector machines and kernels for computational biology. *PLoS computational biology*, 4(10):e1000173, 2008.
[4] G. Berriz, O. King, B. Bryant, C. Sander, and F. Roth. Characterizing gene sets with FuncAssociate. *Bioinformatics*, 19(18):2502–2504, 2003.
[5] M. Cao, C. M. Pietras, et al. New directions for diffusion-based prediction of protein function: incorporating pathways with confidence. *Bioinformatics*, 30:i219–i227, 2014.
[6] M. Cao, H. Zhang, J. Park, N. M. Daniels, M. E. Crovella, L. J. Cowen, and B. Hescott. Going the distance for protein function prediction: a new distance metric for protein interaction networks. *PloS one*, 8(10):e76339, 2013.
[7] H. Cho, B. Berger, and J. Peng. Compact integration of multi-network topology for functional analysis of genes. *Cell systems*, 3(6):540–548, 2016.
[8] G. O. Consortium. The gene ontology resource: 20 years and still GOing strong. *Nucleic acids research*, 47(D1):D330–D338, 2019.
[9] I. S. Dhillon, S. Mallela, and D. S. Modha. Information-theoretic co-clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 89–98, New York, NY, USA, 2003. Association for Computing Machinery.
[10] A. M. Edwards, B. Kus, R. Jansen, D. Greenbaum, J. Greenblatt, and M. Gerstein. Bridging structural biology and genomics: assessing protein interaction data with known complexes. *TRENDS in Genetics*, 18(10):529–536, 2002.
[11] A. Franceschini, D. Szklarczyk, S. Frankild, M. Kuhn, M. Simonovic, A. Roth, J. Lin, P. Minguez, P. Bork, C. vonMering, and L. Jensen. STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res.*, 41:D808–815, 2013.
[12] G. Fu, J. Wang, B. Yang, and G. Yu. NegGOA: negative GO annotations selection using ontology structure. *Bioinformatics*, 32(19):2996–3004, 2016.
[13] J. Gallier. Spectral theory of unsigned and signed graphs. applications to graph clustering: a survey. *arXiv preprint arXiv:1601.04692*, 2016.
[14] V. Gligorijević, M. Barot, and R. Bonneau. deepnf: deep network fusion for protein function prediction. *Bioinformatics*, 34(22):3873–3881, 2018.

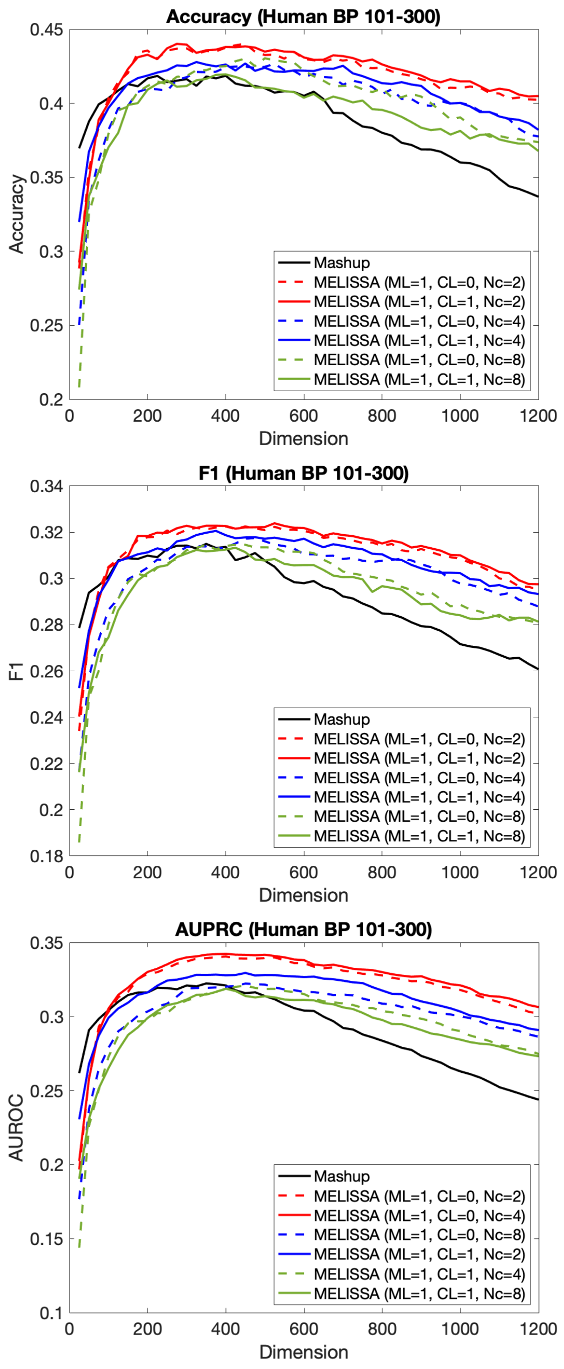

Figure 5: Function prediction comparison with Mashup and MELISSA for Human data. The functional labels used are GO labels in the Biological Process(BP) hierarchy that annotate $101 - 300$ genes in the biological networks. The x-axis is the embedding dimension.

[15] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proc. 22nd ACM SIGKDD*, pages 855–864. ACM, 2016.

[16] Y. P. Hou. Bounds for the least Laplacian eigenvalue of a signed graph. *Acta Mathematica Sinica*, 21(4):955–960, 2005.

[17] S. Kohler, S. Bauer, D. Horn, and P. N. Robinson. Walking the interactome for prioritization of candidate disease genes. *Am J Hum Genet.*, 82:949–958, 2008.

[18] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak. Spectral analysis of signed graphs for clustering, prediction and visualization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 559–570. SIAM, 2010.

[19] C.-S. Liao, K. Lu, M. Baym, R. Singh, and B. Berger. IsoRankN: spectral methods for global alignment of multiple protein networks. *Bioinformatics*, 25(12):i253–i258, 2009.

[20] V. A. Padilha and R. J. Campello. A systematic comparative evaluation of biclustering techniques. *BMC bioinformatics*, 18(1):1–25, 2017.

[21] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Guldener, G. Mannhaupt, M. Munsterkotter, and H. W. Mewes. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Res*, 32:5529–5545, 2004.

[22] the Gene Ontology Consortium. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000. http://www.geneontology.org.

[23] H. Tong, C. Faloutsos, and J. Pan. Fast random walk with restart and its applications. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 613–622, 2006.

[24] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, and L. Bottou. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(12), 2010.

[25] A. Warwick Vesztrocy and C. Dessimoz. Benchmarking gene ontology function predictions using negative annotations. *Bioinformatics*, 36(Supplement_1):i210–i218, 2020.

[26] J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on Machine learning*, pages 1168–1175, 2008.

[27] D. Zhang, Z.-H. Zhou, and S. Chen. Semi-supervised dimensionality reduction. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 629–634. SIAM, 2007.

[28] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.