Neural Message Passing for Multi-Label Classification

Jack Lanchantin[1]✉, Arshdeep Sekhon[1], and Yanjun Qi[1]

University of Virginia, Charlottesville VA 22903, USA
{jjl5sw,as5cu,yq2h}@virginia.edu

Abstract. Multi-label classification (MLC) is the task of assigning a set of target labels for a given sample. Modeling the combinatorial label interactions in MLC has been a long-haul challenge. We propose Label Message Passing (LaMP) Neural Networks to efficiently model the joint prediction of multiple labels. LaMP treats labels as nodes on a label-interaction graph and computes the hidden representation of each label node conditioned on the input using attention-based neural message passing. Attention enables LaMP to assign different importances to neighbor nodes per label, learning how labels interact (implicitly). The proposed models are simple, accurate, interpretable, structure-agnostic, and applicable for predicting dense labels since LaMP is incredibly parallelizable. We validate the benefits of LaMP on seven real-world MLC datasets, covering a broad spectrum of input/output types and outperforming the state-of-the-art results. Notably, LaMP enables intuitive interpretation of how classifying each label depends on the elements of a sample and at the same time rely on its interaction with other labels[1].

1 Introduction

Multi-label classification (MLC) is receiving increasing attention in areas such as natural language processing, computational biology, and image recognition. Accurate and scalable MLC methods are in urgent need for applications like assigning topics to web articles, or identifying binding proteins on DNA. The most common and straightforward MLC method is the binary relevance (BR) approach that considers multiple target labels independently [27]. However, in many MLC tasks there is a clear dependency structure among labels, which BR methods ignore. Unfortunately, accurately modelling all combinatorial label interactions is an NP-hard problem. Many types of models, including a few deep neural network (DNN) based, have been introduced to approximately model such interactions, thus boosting classification accuracy.

Our main concern of this paper is how to represent multiple labels jointly (and conditioned on the input features) in order to make accurate predictions. The most relevant literature addressing this concern falls roughly into three groups. The first group, probabilistic classifier chain (PCC) models, formulate the joint label

[1] We provide our code and datasets at https://github.com/QData/LaMP

dependencies using the chain rule and perform MLC in a sequential prediction manner [22, 32, 19]. Notably, [19] used a recurrent neural network (RNN) sequence to sequence (Seq2Seq) architecture [11] for MLC and achieved the state-of-the-art performance on multiple text-based datasets. However, these methods are inherently unfit for MLC tasks due to their incapacity to be parallelized, and inability to perform well in dense label settings, or when there are a large number of positive labels (since errors propagate in the sequential prediction). We refer the reader to the supplementary material for a full background and analysis of PCC methods (Appendix section 5). The second group learns a shared latent space representing both input features and output labels, and then upsamples from the space to reconstruct the target labels [33, 5]. The main drawback of this group is the interpretability issue with a learned low dimensional latent space, as many real-world applications prefer interpretable predictors. The third group models conditional label dependencies using a structured output or graphical model representation [17, 26]. However, these methods are often limited to only considering pair-wise dependencies due to computational constraints, or are forced to use some variation of approximate inference which has no clear representation of conditional dependencies.

Thus our main question is: *is it possible to have accurate, flexible and explainable MLC methods that are applicable to many dense labels?* This paper provides empirical results showing that this is possible through extending attention based Message Passing Neural Networks (MPNNs) to learn the joint representation of multiple labels conditioned on input features. MPNNs [10] are a class of methods that efficiently learn the joint representations of variables using neural message passing strategies. They provide a flexible framework for modeling multiple variables jointly which have no explicit ordering.

The key idea of our method is to rely on attention-based neural message passing entirely to draw global dependencies from labels to input features, and from labels to labels. To the best of our knowledge, this is the first extension of MPNNs to model a conditional joint representation of output labels, and additionally the first extension of MPNNs to model the interactions of variables where the exact structure is unknown. We name the proposed method Label Message Passing (LaMP) Networks since it performs neural message passing on an unknown, fully-connected label-to-label graph. Through intra-attention (aka self-attention), LaMP assigns different importance to different neighbor nodes per label, dynamically learning how labels interact conditioned on a specific input. We further extend LaMP to cases when a known label interaction graph is provided by modifying the intra-attention to only attend over a node's known neighbors. LaMP networks allow for parallelization in training and testing and can work with dense labels, overcoming the drawbacks of PCC methods.

LaMP most closely belongs to the third MLC category we mentioned above, however it trains a unified model to classify each label *and* to model the label to label dependencies at the same time, in an end-to-end fashion. The important aspect is that LaMP networks automatically learn the output label dependency

structure conditioned on a specific input using neural message passing. This in turn can easily be interpreted to understand the conditional structure.

The main contributions of this paper include: (1) **Accurate MLC**: Our model achieves similar, or better performance compared to the previous state-of-the-art across five MLC metrics. We validate our model on eight MLC datasets which cover a wide spectrum of input data structure: sequences (English text, DNA), tabular (binary word vectors), graph (drug molecules), and images, as well as output label structure: unknown and graph. (2) **Interpretable**: Although deep-learning based systems have widely been viewed as "black boxes", our attention based LaMP models allow for a straightforward way to extract three different types of model visualization: intermediate network predictions, label to feature dependencies, and label to label dependencies.

2 Method: LaMP Networks

Notations. We define the following notations, used throughout the paper. Let $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$ be the set of data samples with inputs $\boldsymbol{x} \in X$ and outputs $\boldsymbol{y} \in Y$. Inputs $\boldsymbol{x}$ are a (possibly ordered) set of S components $\{x_1, x_2, ..., x_S\}$, and outputs $\boldsymbol{y}$ are a set of L labels $\{y_1, y_2, ..., y_L\}$. MLC involves predicting the set of binary labels $\{y_1, y_2, ..., y_L\}, y_i \in \{0, 1\}$ given input $\boldsymbol{x}$.

In general we can assume to represent the input feature components as embedded vectors $\{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_S\}$, $\boldsymbol{z}_i \in \mathbb{R}^d$, using some learned embedding matrix $\mathbf{W}^x \in \mathbb{R}^{\delta \times d}$. Here d is the embedding size and, δ is the size of x_i. x_i can be any component of a particular input (for example, words in a sentence, patches of an image, nodes of a known graph, or one of the tabular features).

Similarly, labels can be first represented as embedded vectors $\{\boldsymbol{u}_1^{t=0}, \boldsymbol{u}_2^{t=0}, ..., \boldsymbol{u}_L^{t=0}\}$, $\boldsymbol{u}_i^t \in \mathbb{R}^d$, through a learned embedding matrix $\mathbf{W}^y \in \mathbb{R}^{L \times d}$, where L denotes the number of labels. Here we use t to represent the 'state' of the embedding after the t^{th} update step. This is because in LaMP networks, each label embedding is updated for t steps before the predictions are made. The key idea of LaMP networks is that labels are represented as nodes in a label-interaction graph G_{yy} denoting nodes as embedding vectors $\{\boldsymbol{u}_{1:L}^t\}$. LaMP networks use MPNN modules with attention to pass messages from input embeddings $\{\boldsymbol{z}_{1:S}\}$ to G_{yy}, and then within G_{yy} to model the joint prediction of labels.

2.1 Background: Message Passing Neural Networks

Message Passing Neural Networks (MPNNs) [10] are a generalization of graph neural networks (GNNs) [23]. MPNNs model variables as nodes on a graph G. Here $G = (V, E)$, where V describes the set of nodes (variables) and E denotes the set of edges (about how variables interact with other variables). In MPNNs, joint representations of nodes and edges are modelled using message passing rather than explicit probabilistic formulations, allowing for efficient inference. MPNNs model the joint dependencies using message function M^t and node update function U^t for T time steps, where t is the current time step. The

hidden state $\boldsymbol{v}_i^t \in \mathbb{R}^d$ of node $i \in G$ is updated based on messages $\boldsymbol{m}_i^t$ from its neighboring nodes $\{\boldsymbol{v}_{j \in \mathcal{N}(i)}^t\}$ defined by neighborhood $\mathcal{N}(i)$:

$$\boldsymbol{m}_i^t = \sum_{j \in \mathcal{N}(i)} M^t(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t), \tag{1}$$

$$\boldsymbol{v}_i^{t+1} = U^t(\boldsymbol{m}_i^t). \tag{2}$$

After T rounds of iterative updates to spread information to distant nodes, a readout function R is used on the updated node embeddings to make predictions like classifying nodes or classifying properties about the graph.

Many possibilities exist for functions M^t and U^t. We specifically choose to pass messages using intra-attention (also called as self-attention) neural message passing which enable nodes to attend over their neighborhoods differentially. This allows for the network to learn different importances for different nodes in a neighborhood, without depending on knowing the graph structure upfront (essentially learning the unknown graph structure) [30]. In this formulation, messages for node $\boldsymbol{v}_i^t$ are obtained by a weighted sum of all its neighboring nodes $\{\boldsymbol{v}_{j \in \mathcal{N}(i)}^t\}$ where the weights are calculated by attention representing the importance of each neighbor for a specific node [1]. In the rest of the paper, we use "graph attention" and "neural message passing" interchangeably.

Intra-attention neural message passing works as follows. We first calculate attention weights α_{ij}^t for pair of nodes $(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t)$ using attention function $a(\cdot)$:

$$\alpha_{ij}^t = \text{softmax}_j(e_{ij}^t) = \frac{\exp(e_{ij}^t)}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^t)} \tag{3}$$

$$e_{ij}^t = a(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t) \tag{4}$$

$$a(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t) = \frac{(\mathbf{W}^q \boldsymbol{v}_i^t)^\top (\mathbf{W}^u \boldsymbol{v}_j^t)}{\sqrt{d}} \tag{5}$$

where e_{ij}^t represents the importance of node j for node i, however un-normalized. e_{ij}^t are normalized across all neighboring nodes of node i using a softmax function (Eq 3) to get α_{ij}^t. For the attention function $a(\cdot)$, we used a scaled dot product with node-wise linear transformations $\mathbf{W}^q \in \mathbb{R}^{d \times d}$ on node $\boldsymbol{v}_i^t$ and $\mathbf{W}^u \in \mathbb{R}^{d \times d}$ on node $\boldsymbol{v}_j^t$. Scaling by $\sqrt{d}$ is used to mitigate training issues [29].

Then we use a so called attention message function M_{atn}^t to produce the message from node j to node i using the learned attention weights α_{ij}^t and another transformation matrix $\mathbf{W}^v \in \mathbb{R}^{d \times d}$:

$$M_{\text{atn}}(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t; \boldsymbol{W}) = \alpha_{ij}^t \mathbf{W}^v \boldsymbol{v}_j^t, \tag{6}$$

$$\boldsymbol{m}_i^t = \boldsymbol{v}_i^t + \sum_{j \in \mathcal{N}(i)} M_{\text{atn}}(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t; \boldsymbol{W}). \tag{7}$$

Eq 7 computes the full message $\boldsymbol{m}_i^t$ for node $\boldsymbol{v}_i^t$ by linearly combining messages from all neighbor nodes $j \in \mathcal{N}(i)$ with a residual connection on the current $\boldsymbol{v}_i^t$.

Lastly, node $\boldsymbol{v}_i^t$ is updated to next state $\boldsymbol{v}_i^{t+1}$ using message $\boldsymbol{m}_i^t$ by a multi-layer perceptron (MLP) update function U_{mlp}, plus a $\boldsymbol{m}_i^t$ residual connection:

$$U_{\text{mlp}}(\boldsymbol{m}_i^t; \boldsymbol{W}) = \text{ReLU}(\mathbf{W}^r \boldsymbol{m}_i^t + b_1)^\top \mathbf{W}^b + b_2 \tag{8}$$

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{m}_i^t + U_{\text{mlp}}(\boldsymbol{m}_i^t; \boldsymbol{W}). \tag{9}$$

Function U_{mlp} is parameterized with matrices $\{\mathbf{W}^r \in \mathbb{R}^{d\times d}, \mathbf{W}^b \in \mathbb{R}^{d\times d}\}$. It is important to note that $\mathbf{W}$ in Eq 9 are shared (i.e., separately applied) across all nodes. This can be viewed as 1-dimensional convolution operation with kernel and stride sizes of 1. Weight sharing across nodes is a key aspect of MPNNs, where node dependencies are learned in an order-invariant manner.

2.2 LaMP: Label Message Passing

Given the input embeddings $\{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_S\}$, the goal of Label Message Passing is to model the conditional dependencies between label embeddings $\{\boldsymbol{u}_1^t, \boldsymbol{u}_2^t, ..., \boldsymbol{u}_L^t\}$ using Message Passing Neural Networks. We assume that the label embeddings are nodes on a label-interaction graph called G_{yy}, where the initial state of the embeddings $\{\boldsymbol{u}_{1:L}^0\}$ at $t = 0$ are obtained using label embedding matrix $\mathbf{W}^y$.

Each step t in Label Message Passing consists of two parts in order to update the label embeddings: (a). Feature-to-Label Message Passing, where messages are passed from the input embeddings to the label embeddings, and (b). Label-to-Label Message Passing, where messages are passed between labels. An overview of our model is shown in Fig. 1. We explain these two parts in detail in the following subsections. LaMP Networks use T steps of attention-based neural message passing to update the label nodes before a readout function makes a prediction for each label i on its final state $\boldsymbol{u}_i^T$.

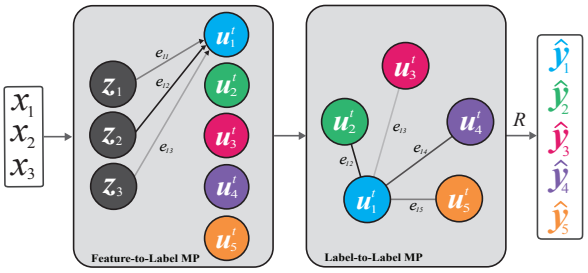

Fig. 1. LaMP Networks. Given input $\mathbf{x}$, we encode its components $\{x_1, x_2, x_3\}$ as embedded input nodes $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3\}$. We encode labels $\{y_1, y_2, ..., y_5\}$ as embedded label nodes $\{\boldsymbol{u}_1^t, \boldsymbol{u}_2^t, ..., \boldsymbol{u}_5^t\}$ of label-interaction graph G_{yy}. First, MPNN_{xy} is used to pass messages from the input nodes to the labels nodes and update the label nodes. Then, MPNN_{yy} is used to pass messages between the label nodes and update label nodes. Finally, readout function R performs node-level classification on label nodes to make binary label predictions $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_5\}$.

Updating Label Embeddings via Feature-to-Label Message Passing
Given a particular input $\boldsymbol{x}$ with embedded feature components $\{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_S\}$, the first step in LaMP is to update the label embeddings by passing messages from the input embeddings to the label embeddings, as shown in the "Feature-to-Label MP" block of Fig. 1. To do this, LaMP uses neural message passing module MPNN_{xy} to update the i^{th} label node's embedding $\boldsymbol{u}_i^t$ using the embeddings of all the components of an input.

That is, we update each $\boldsymbol{u}_i^t$ by using a weighted sum of all input embeddings $\{\boldsymbol{z}_{1:S}\}$, in which the weights represent how important an input component is to the i^{th} label node. The weights for the message are learned via Label-to-Feature attention (i.e., each label attends to each input embedding differently to compute the weights). In this step, messages are only passed from the input nodes to the label nodes, and not vice versa (i.e. Feature-to-Label message passing is directed).

More specifically, to update label embedding $\boldsymbol{u}_i^t$, MPNN_{xy} uses attention message function M_{atn}^t on all embeddings of the input $\{\boldsymbol{z}_{1:S}\}$ to produce messages $\boldsymbol{m}_i^t$, and MLP update function U_{mlp} to produce the updated intermediate embedding state $\boldsymbol{u}_i^{t'}$:

$$\boldsymbol{m}_i^t = \boldsymbol{u}_i^t + \sum_{j=1}^{S} M_{\text{atn}}(\boldsymbol{u}_i^t, \boldsymbol{z}_j; \boldsymbol{W}_{\text{xy}}), \tag{10}$$

$$\boldsymbol{u}_i^{t'} = \boldsymbol{m}_i^t + U_{\text{mlp}}(\boldsymbol{m}_i^t; \boldsymbol{W}_{\text{xy}}). \tag{11}$$

The key advantage of Feature-to-Label message passing with attention is that each label node can attend differently on input elements (e.g. different words in an input sentence).

Updating Label Embeddings via Label-to-Label Message Passing
At this point, an independent prediction can be made for each label conditioned on $\boldsymbol{x}$ using $\{\boldsymbol{u}_{1:L}^{t'}\}$. However, in order to consider label dependencies, we model interactions between the label nodes $\{\boldsymbol{u}_{1:L}^{t'}\}$ using Label-to-Label message passing and update them accordingly, as shown in the "Label-to-Label MP" block of Fig. 1. Given the exponentially large number of possible conditional dependencies, we use neural message passing as an efficient way to model such interactions, which has been shown to work well in practice for other tasks.

We assume there exist a label interaction graph $G_{\text{yy}} = (V_{\text{yy}}, E_{\text{yy}})$, $V_{\text{yy}} = \{y_{1:L}\}$, and E_{yy} includes all undirected pairwise edges connecting node $\boldsymbol{y}_i$ and node $\boldsymbol{y}_j$. At this stage, we use another message passing module, MPNN_{yy} to pass messages between labels and update them. The label embedding $\boldsymbol{u}_i^{t'}$ is updated by a weighted combination through attention of all its neighbor label nodes $\{\boldsymbol{u}_{j\in\mathcal{N}(i)}^{t'}\}$.

MPNN_{yy} uses attention message function $M_{\text{atn}}^{t'}$ on all neighbor label embeddings $\{\boldsymbol{u}_{j\in\mathcal{N}(i)}^{t'}\}$ to produce message $\boldsymbol{m}_i^t$, and MLP update function $U_{\text{mlp}}^{t'}$ to compute updated embedding $\boldsymbol{u}_i^{t+1}$:

$$\boldsymbol{m}_i^{t'} = \boldsymbol{u}_i^{t'} + \sum_{j\in\mathcal{N}(i)} M_{\text{atn}}(\boldsymbol{u}_i^{t'}, \boldsymbol{u}_j^{t'}; \boldsymbol{W}_{\text{yy}}), \tag{12}$$

$$\boldsymbol{u}_i^{t+1} = \boldsymbol{m}_i^{t'} + U_{\text{mlp}}(\boldsymbol{u}_i^{t'}, \boldsymbol{m}_i^{t'}; \boldsymbol{W}_{\text{yy}}). \tag{13}$$

If there exists a known label interaction graph G_{yy}, message $\boldsymbol{m}_i^t$ for node i is computed using its neighboring nodes $j \in \mathcal{N}(i)$, where the neighbors $\mathcal{N}(i)$ are defined by the graph. If there is no known G_{yy} graph, we assume a fully connected graph, which means $\mathcal{N}(i) = \{j \in V_{\text{yy}}\}$ (including i).

Message Passing for Multiple Time Steps
To learn more complex relations among nodes, we compute a total of T time steps of updates. This is essentially a stack of T MPNN layers. In our implementation, the label embeddings are updated by MPNN_{xy} and MPNN_{yy} for T time steps to produce $\{\boldsymbol{u}_1^T, \boldsymbol{u}_2^T, ..., \boldsymbol{u}_L^T\}$.

2.3 Readout Layer (MLC Predictions from the label embeddings)

After T updates to the label embeddings, the last module predicts each label $\{\hat{y}_1, ...\hat{y}_L\}$. A readout function R projects each of the L label embeddings $\boldsymbol{u}_i^T$ using projection matrix $\mathbf{W}^o \in \mathbb{R}^{L \times d}$, where row $\mathbf{W}_i^o \in \mathbb{R}^d$ is the learned output vector for label i. The calculated vector of size $L \times 1$ is then fed through an element-wise sigmoid function to produce probabilities of each label being positive:

$$\hat{y}_i = R(\boldsymbol{u}_i^T; \mathbf{W}^o) = \text{sigmoid}(\mathbf{W}_i^o \boldsymbol{u}_i^T). \tag{14}$$

2.4 Model Details

Multi-head Attention. In order to allow a particular node to attend to multiple other nodes (or multiple groups of nodes) at once, LaMP uses multiple attention heads. Inspired by [29], we use K independent attention heads for each $\mathbf{W}^{\cdot}$ matrix during the message computation, where each matrix column $\mathbf{W}_j^{\cdot,k}$ is of dimension d/K. The generated representations are concatenated (denoted by $\|$) and linearly transformed by matrix $\mathbf{W}^z \in \mathbb{R}^{d \times d}$. Multi-head attention changes message passing function M_{atn}, but update function U_{mlp} stays the same.

$$e_{ij}^{t,k} = (\mathbf{W}^{q,k} \boldsymbol{v}_i^t)^\top (\mathbf{W}^{u,k} \boldsymbol{v}_j^t) / \sqrt{d} \tag{15}$$

$$\alpha_{ij}^{t,k} = \frac{\exp(e_{ij}^{t,k})}{\sum_{j \in \mathcal{N}(i)} \exp(e_{ij}^{t,k})} \tag{16}$$

$$M_{\text{atn}}^k(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t; \boldsymbol{W}) = \alpha_{ij}^{t,k} \mathbf{W}^{v,k} \boldsymbol{v}_j^t, \tag{17}$$

$$\boldsymbol{m}_i^t = \boldsymbol{v}_i^t + \left(\Big\|_{k=1}^{K} \left[\sum_{j \in \mathcal{N}(i)} M_{\text{atn}}^k(\boldsymbol{v}_i^t, \boldsymbol{v}_j^t; \boldsymbol{W}) \right] \right) \mathbf{W}^c \tag{18}$$

Matrices $\mathbf{W}_{\cdot}^q, \mathbf{W}_{\cdot}^u, \mathbf{W}_{\cdot}^v, \mathbf{W}_{\cdot}^r, \mathbf{W}_{\cdot}^b, \mathbf{W}_{\cdot}^c$, are not shared across time steps (but are shared across nodes).

Label Embedding Weight Sharing. To enforce each label's input embedding to correspond to that particular label, the label embedding matrix weights $\mathbf{W}^y$

are shared with the readout projection matrix $\mathbf{W}^o$. In other words, $\mathbf{W}^y$ is used to produce the initial node vectors for G_{yy}, and then is used again to calculate the pre-sigmoid output values for each label, so $\mathbf{W}^o \equiv \mathbf{W}^y$. This was shown beneficial in Seq2Seq models for machine translation [21].

2.5 Loss Function

The final output of LaMP networks $\hat{\boldsymbol{y}}$ are trained using the mean binary cross entropy (BCE) over all outputs y_i. For one sample, given true binary label vector $\boldsymbol{y}$ and predicted labels $\hat{\boldsymbol{y}}$, the output loss $\mathcal{L}_{out}$ is:

$$\mathcal{L}_{out}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{L}\sum_{i=1}^{L} -(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{19}$$

The final outputs $\hat{y}_i$ are computed from the final label node states $\boldsymbol{u}_i^T$ (Eq. 14). However, since LaMP networks iteratively update the label nodes from $t = 0$ to T, we can "probe" the label nodes at each intermediate state from t=1 to T-1 and enforce an auxilary loss on those states. To do this, we use the same matrix W^o to extract the intermediate prediction $\hat{y}_i^t$ at state t: $\hat{y}_i^t = R(\boldsymbol{u}_i^t; \mathbf{W}^o)$. We use the same BCE loss on the these predictions to compute intermediate loss $\mathcal{L}_{int}$:

$$\mathcal{L}_{int}(\boldsymbol{y}, \hat{\boldsymbol{y}}^t) = \frac{1}{L}\sum_{i=1}^{L} -(y_i \log(\hat{y}_i^t) + (1 - y_i)\log(1 - \hat{y}_i^t)). \tag{20}$$

We note that the intermediate predictions $\hat{y}_i^t$ are computed for both $\boldsymbol{u}_i^t$ (after Label-to-Label message passing), as well as $\boldsymbol{u}_i^{t'}$ (after Feature-to-Label message passing). The final loss is a combination of both the original and intermediate, where the intermediate loss is weighted by λ:

$$\mathcal{L}_{LaMP} = \mathcal{L}_{out}(\boldsymbol{y}, \hat{\boldsymbol{y}}) + \lambda \sum_{t=1}^{T-1} \mathcal{L}_{int}(\boldsymbol{y}, \hat{\boldsymbol{y}}^t) \tag{21}$$

In LaMP networks, $p(y_i|\{y_{j\neq i}\}, \boldsymbol{z}_{1:S}; \mathbf{W})$ is approximated by jointly representing $\{y_{1:L}\}$ using message passing from $\{\boldsymbol{z}_{1:S}\}$ and from the embeddings of all neighboring labels $\{y_{j\in\mathcal{N}(i)}\}$.

2.6 LaMP Variation: Input Encoding with Feature Message Passing (FMP)

Thus far, we have assumed that we use the raw feature embeddings $\{\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_S\}$ to pass messages to the labels. However, we could also update the feature embeddings before they are passed to the label nodes by modelling the interactions between features.

For a particular input $\boldsymbol{x}$, we first assume that the input features $\{x_{1:S}\}$ are nodes on a graph, G_{xx}. $G_{\text{xx}} = (V_{\text{xx}}, E_{\text{xx}})$, $V_{\text{xx}} = \{x_{1:S}\}$, and E includes all

undirected pairwise edges connecting node $\boldsymbol{x}_i$ and node $\boldsymbol{x}_j$. MPNN_{xx}, parameterized by $\mathbf{W}_{\text{xx}}$, is used to pass messages between the input embeddings in order to update their states. Nodes on G_{xx} are represented as embedding vectors $\{\boldsymbol{z}_1^t, \boldsymbol{z}_2^t, ..., \boldsymbol{z}_S^t\}$, where the initial states $\{\boldsymbol{z}_{1:S}^0\}$ are obtained using embedding matrix $\mathbf{W}^x$ on input components $\{x_1, x_2, ..., x_S\}$. The embeddings are then updated by MPNN_{xx} using message passing for T time steps to produce $\{\boldsymbol{z}_1^T, \boldsymbol{z}_2^T, ..., \boldsymbol{z}_S^T\}$.

To update input embedding $\boldsymbol{z}_i^t$, MPNN_{xx} uses attention message function M_{atn}^t (Eq. 6) on all neighboring input embeddings $\{\boldsymbol{z}_{j\in\mathcal{N}(i)}^t\}$ to produce messages $\boldsymbol{m}_i^t$, and MLP update function U_{mlp} (Eq. 9) to produce updated embedding $\boldsymbol{z}_i^{t+1}$:

$$\boldsymbol{m}_i^t = \boldsymbol{z}_i^t + \sum_{j\in\mathcal{N}(i)} M_{\text{atn}}(\boldsymbol{z}_i^t, \boldsymbol{z}_j^t; \boldsymbol{W}_{\text{xx}}), \tag{22}$$

$$\boldsymbol{z}_i^{t+1} = \boldsymbol{m}_i^t + U_{\text{mlp}}(\boldsymbol{m}_i^t; \boldsymbol{W}_{\text{xx}}). \tag{23}$$

If there exists a known G_{xx} graph, message $\boldsymbol{m}_i^t$ for node i is computed using its neighboring nodes $j \in \mathcal{N}(i)$, where the neighbors $\mathcal{N}(i)$ are defined by the graph. If there is no known graph, we assume a fully connected G_{xx} graph, which means $\mathcal{N}(i) = \{j \in V_{\text{xx}}\}$. Inputs with a sequential ordering can be modelled as a fully connected graph using positional embeddings [3].

In summary, MPNN_{xx} is used to update input feature nodes $\{\boldsymbol{z}_{1:S}^t\}$ by passing messages within the feature-interaction graph. MPNN_{xy}, is used to update output label nodes $\{\boldsymbol{u}_{1:L}^t\}$ by passing messages from the features to labels (from input nodes $\{\boldsymbol{z}_{1:S}^t\}$ to output nodes $\{\boldsymbol{u}_{1:L}^t\}$). MPNN_{yy}, is used to update output label nodes $\{\boldsymbol{u}_{1:L}^t\}$ by passing messages within the label-interaction graph (between label nodes). Once messages have been passed to update the feature and label nodes for T integrative updates, a readout function R is then used on the label nodes to make a binary classification prediction on each label, $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_L\}$. Figure 1 shows the LaMP network without the feature-interaction graph.

2.7 Advantages of LaMP Models

Efficiently Handling Dense Label Predictions. It is known that autoregressive models such as RNN Seq2Seq suffer from the propagation of errors over the sequential positive label predictions. This makes it difficult for these models to handle dense, or many positive label, samples. In addition, autoregressive models require a time consuming post-processing step such as beam search to obtain the optimal label set. Lastly, autoregressive models require a predefined label ordering for training the sequential prediction, which can lead to instabilities at testing time [31].

Motivated by the drawbacks of autoregressive models for MLC, the proposed LaMP model removes the reliance on sequential predictions, beam search, and a chosen label ordering, while still modelling the label dependencies. This is particularly beneficial when the number of positive output labels is large (i.e. dense). LaMP networks predict the output *set* of labels all at once, which is made possible by the fact that inference doesn't use a probabilistic chain, but there

is still a representation of label dependencies via label to label attention. As an additional benefit, as noted by [4], it may be useful to maintain 'soft' predictions for each label in MLC. This is a major drawback of the PCC models which make 'hard' predictions of the positive labels, defaulting all other labels to 0.

Structure Agnostic. Many input or output types are instances where the relational structure is not made explicit, and must be inferred or assumed [3]. LaMP networks allow for greater flexibility of both input structures (known structure such as sequence or graph, or unknown such as tabular), as well as output structures (e.g., known graph vs unknown structure). To the best of our knowledge, this is the first work to use MPNNs to *infer* the relational structure of the data by using attention mechanisms.

Interpretability. Our formulation of LaMP allows us to visualize predictions in several different ways. First, since predictions are made in an iterative manner via graph update steps, we can "probe" each label's state at each step to get intermediate predictions. Second, we can visualize the attention weights which automatically learn the relational structure. Combining these two visualization methods allows us to see how the predictions change from the initial predictions given only the input sequence to the final state where messages have been passed from other labels, leading us to better insights for specific MLC samples.

2.8 Connecting to Related Topics

Structured Output Predictions. The use of graph attention in LaMP models is closely connected to the literature of structured output prediction for MLC. [9] used conditional random fields (CRFs) [17] to model dependencies among labels and features for MLC by learning a distribution over pairs of labels to input features, but these are limited to pairwise dependencies.

To overcome the naive pairwise dependency constraint of CRFs, structured prediction energy networks (SPENS) [4] and related methods [28, 13] locally optimize an unconstrained structured output. In contrast to SPENs which use an iterative refinement of the output label predictions, our method is a simpler feed forward block to make predictions in one step, yet still models dependencies through attention on embeddings, which gives the added interpretability benefit.

Multi-label Classification By Modeling Label Interaction Graphs. [12] formulate MLC using a label graph and they introduced a conditional dependency SVM where they first trained separate classifiers for each label given the input and all other true labels and used Gibbs sampling to find the optimal label set. The main drawback is that this method does not scale to a large number of labels. [24] proposes a method to label the pairwise edges of randomly generated label graphs, and requires some chosen aggregation method over all random graphs. The authors introduce the idea that variation in the graph structure shifts the inductive bias of the base learners. Our fully connected label graph with attention on the neighboring nodes can be regarded as a form of graph ensemble learning [15]. [8] use graph neural networks for MLC, but focus on graph inputs and do not explicitly model label the label-to-label dependencies, thus resulting in a worse performance than LaMP.

Table 1. ebF1 Scores across all 8 datasets

	Reuters	Bibtex	Bookmarks	Delicious	RCV1	TFBS	SIDER	NUSWIDE
FastXML [20]	-	-	-	-	0.841	-	-	-
Madjarov [18]	-	0.434	0.257	0.343	-	-	-	-
SPEN [4]	-	0.422	0.344	0.375	-	-	-	-
RNN Seq2Seq [19]	0.894	0.393	0.362	0.320	**0.890**	0.249	0.356	0.329
Emb + MLP	0.854	0.363	0.368	0.371	0.865	0.167	**0.766**	0.371
Emb + LaMP$_{el}$	0.859	0.379	0.351	0.358	0.868	0.289	0.767	**0.376**
Emb + LaMP$_{fc}$	0.896	0.427	0.376	0.368	0.871	0.319	0.763	**0.376**
Emb + LaMP$_{pr}$	0.895	0.424	0.373	0.366	0.870	0.317	0.765	0.372
FMP + LaMP$_{el}$	0.883	0.435	0.375	0.369	0.887	0.310	**0.766**	-
FMP + LaMP$_{fc}$	**0.906**	0.445	**0.389**	**0.372**	0.889	**0.321**	0.764	-
FMP + LaMP$_{pr}$	0.902	**0.447**	0.386	**0.372**	0.887	0.321	**0.766**	-

Graph Neural Networks (GNNs). Passing embedding messages from node to neighbor nodes connects to a large body of literature on graph neural networks [3] and embedding models for structures [6]. The key idea is that instead of conducting probabilistic operations (e.g., product or re-normalization), the proposed models perform nonlinear function mappings in each step to learn feature representations of structured components. [10, 30, 2] all follow similar ideas to pass the embedding from node to neighbor nodes or neighbor edges.

There have been many recent works extending the basic GNN framework to update nodes using various message passing, update, and readout functions [16, 14, 2, 10]. We refer the readers to [3] for a survey. However, none of these have used GNNs for MLC. In addition, none of these have attempted to learn the graph structure by using neural attention on fully connected graphs.

3 Experiments

We validate our model on eight real world MLC datasets. These datasets vary in the number of samples, number of labels, input type (sequential, tabular, graph, vector), and output type (unknown, known label graph). They also cover a wide spectrum of input data types, including: raw English text (sequential form), binary word vector (tabular form), drug molecules (graph form), and images (vector form). Data statistics are in Table 5 and Section 8.1. Due to the space limit, we move the details of evaluation metrics to Section 8.2 and the hyper-parameters to Section 8.3. Details of previous results from the state-of-the-art baselines are in Section 8.4.

3.1 LaMP Variations

For LaMP models, we use two variations of input features, and three variations of Label-to-Label Message Passing. For input features, we use (1) **Emb**, which

is the raw learned feature embeddings of dimension d, and (2) **FMP**[2] which is the updated state of each feature embedding after 2 layers of Feature Message Passing, as explained in 2.6. For each of the two input feature variations, we use three variations of the label graph which Label-to-Label Message Passing uses to update the labels given the input features, explained as follows.

LaMP$_{el}$ uses an edgeless label graph and messages are not passed between labels, assuming no label dependencies.

LaMP$_{fc}$ uses a fully connected label graph where each label is able to attend to all other labels (including itself) in order to compute the messages.

LaMP$_{pr}$ uses a prior label graph where each label is able to attend to only other labels from the known label graph (including itself) in order to compute the messages. For RCV1, we use the known tree label structure, and for TFBS we use known protein-protein interactions (PPI) from [25]. For all other datasets, we create a graph where we place an edge on the adjacency matrix for all labels that co-occur in any sample for the training set. This is summarized in the last column of Appendix Table 5.

3.2 Performance Evaluation

ebF1. Table 1 shows the most commonly used evaluation, example-based F1 (ebF1) scores, for the seven datasets. LaMP outperforms the baseline MLP models which assume no label dependencies, as well as RNN Seq2Seq, which models label dependencies using a classifier chain. More importantly, we compare using an output graph with no edges (LaMP$_{el}$), which assumes no label dependencies vs. an output graph with edges (LaMP$_{fc}$). The two models have the same architecture and number of parameters, with the only thing varying being the message passing between label nodes. We can see that for most datasets, modelling label dependencies using LaMP$_{fc}$ does in fact help. We found that using a known prior label structure (LaMP$_{pr}$) did not improve the results significantly. LaMP$_{fc}$ predictions produced an average 1.8% ebF1 score increase over the independent LaMP$_{el}$ predictions. LaMP$_{pr}$ resulted in an average 1.7% ebF1 score increase over LaMP$_{el}$. When comparing to the MLP baseline, LaMP$_{fc}$ and LaMP$_{pr}$ produced an average 18.5% and 18.4% increase, respectively.

miF1. While high ebF1 scores indicate strong average F1 scores over all samples, the label-based Micro-averaged F1 (miF1) scores indicate strong results on the most frequent labels. Table 2 shows the miF1 scores, for the all datasets. LaMP$_{fc}$ produced an average 1.6% miF1 score increase over the independent LaMP$_{el}$. LaMP$_{pr}$ produced an average 1.8% miF1 score increase over LaMP$_{el}$. When comparing to the MLP baseline, LaMP$_{fc}$ and LaMP$_{pr}$ resulted in an average 20.2% and 20.5% increase, respectively.

maF1. Contrarily, high label-based Macro-averaged F1 (maF1) scores indicate strong results on less frequent labels. Table 2 shows maF1 scores, which show

[2] For NUS-WIDE, since we use the 128-dimensional cVLAD features as input to compare to [8], we cannot use the **FMP** method.

Table 2. miF1 Scores across all 8 datasets

	Reuters	Bibtex	Bookmarks	Delicious	RCV1	TFBS	SIDER	NUSWIDE
FastXML [20]	-	-	-	-	0.847	-	-	-
SVM [7]	0.787	-	-	-	-	-	-	-
GAML [8]	-	-	0.333	-	-	-	-	0.398
Madjarov [18]	-	0.462	0.268	0.339	-	-	-	-
RNN Seq2Seq [19]	0.858	0.384	0.329	0.329	**0.884**	0.311	0.389	0.418
Emb + MLP	0.835	0.389	0.349	0.385	0.855	0.218	0.795	0.465
Emb + LaMP$_{el}$	0.842	0.413	0.334	0.372	0.858	0.401	0.797	**0.472**
Emb + LaMP$_{fc}$	0.871	0.458	0.363	0.379	0.859	0.449	0.797	0.470
Emb + LaMP$_{pr}$	0.877	0.462	0.363	0.380	0.859	0.448	**0.798**	0.468
FMP + LaMP$_{el}$	0.870	0.455	0.355	0.381	0.877	0.445	0.795	-
FMP + LaMP$_{fc}$	0.886	0.465	**0.373**	0.384	0.877	**0.450**	0.795	-
FMP + LaMP$_{pr}$	**0.889**	**0.473**	0.371	**0.386**	0.877	0.449	0.797	-

the strongest improvement of LaMP$_{fc}$ and LaMP$_{pr}$ variation over independent predictions. LaMP$_{fc}$ resulted in an average 2.4% maF1 score increase over the independent LaMP$_{el}$. LaMP$_{pr}$ produced an average 2.1% maF1 score increase over LaMP$_{el}$. This indicates that Label-to-Label message passing can help boost the accuracy of rare label predictions. When comparing to Emb + MLP, LaMP$_{fc}$ and LaMP$_{pr}$ produced an average 57.0% and 56.7% increase, respectively.

Other Metrics. Due to space constraints, we report subset accuracy in Appendix (supplementary) Table 7. RNN Seq2Seq models mostly perform all other models for this metric since they are trained to maximize it[19]. However, for all other metrics, RNN Seq2Seq does not perform as well, concluding that for most applications, PCC models aren't necessary. We also report Hamming Accuracy in Appendix Table 8, and we note that LaMP networks outperform or perform similarly to baseline methods, but we observe that this metric is mostly unhelpful.

Metrics Performance Summary. While LaMP does not explicitly model label dependencies as autoregressive or structured prediction models do, the attention weights do learn some dependencies among labels (Section 3.3). This is indicated by the fact that LaMP, which uses Label-to-Label attention, mostly outperforms the ones which don't, indicating that it is learning label dependencies.

Speed. LaMP results in a mean of 1.7x and 5.0x training and testing speedups, respectively, over the previous state-of-the-art probabilistic MLC method, RNN Seq2Seq. Speedups over RNN Seq2Seq model are shown in Table 3.3.

3.3 Interpretability Evaluation

The structure of LaMP networks allows for three different types of visualization methods to understand how the network predicts each label. We explain the three types here and show the results for a sample from the Bookmarks dataset using the FMP + LaMP$_{fc}$ model.

Table 3. maF1 Scores across all 8 datasets

	Reuters	Bibtex	Bookmarks	Delicious	RCV1	TFBS	SIDER	NUSWIDE
SVM [7]	0.468	-	-	-	-	-	-	-
FastXML [20]	-	-	-	-	0.592	-	-	-
GAML [8]	-	-	0.217	-	-	-	-	0.114
Madjarov [18]	-	0.316	0.119	0.142	-	-	-	-
RNN Seq2Seq [19]	0.457	0.282	0.237	0.166	0.741	0.210	0.207	0.143
Emb + MLP	0.366	0.275	0.248	0.180	0.667	0.094	0.665	0.173
Emb + LaMP$_{el}$	0.476	0.308	0.229	0.176	0.680	0.326	0.666	0.198
Emb + LaMP$_{fc}$	0.547	0.366	0.271	0.192	0.691	0.362	0.663	**0.203**
Emb + LaMP$_{pr}$	**0.560**	0.372	0.267	0.192	0.698	0.365	0.663	0.196
FMP + LaMP$_{el}$	0.508	0.353	0.266	0.192	0.742	**0.368**	0.664	-
FMP + LaMP$_{fc}$	0.520	0.371	**0.286**	0.195	**0.743**	0.364	**0.668**	-
FMP + LaMP$_{pr}$	0.517	**0.376**	0.280	**0.196**	0.740	0.364	0.664	-

Intermediate Output Prediction. One advantage of the multi step formulation of label embedding updates is that it gives us the ability to probe the state of each label at intermediate steps and view the model's predictions at those steps. To do this, we use the readout function R on each intermediate label embeddings state $\boldsymbol{u}_i^t$ to find the probability that the label embedding would predict a positive label. In other words, this is the post-sigmoid output of the readout function of each embedding $R(\boldsymbol{u}_i^t; \mathbf{W^o})$ at each step $t = 1, ..., T$. We note that each step contains two stages: $t.1$ is the output after the Feature-to-Label message passing, and $t.2$ is output after the Label-to-Label message passing. The output after the second stage of the final step (i.e. $T.2$) is the model's final output.

Figure 2 (a.) shows the intermediate prediction outputs from the $T = 2$ step model. On the horizontal axis are a selected subset of all possible labels, with the red colored axis labels being all true positive labels. On the vertical axis, each row represents one of the label embedding states in the $T = 2$ step model. Each cell represents the readout function's prediction for each label embedding's state. The brighter the grid cell, the more likely that label is positive at the current stage. Starting from the bottom, the first row (1.1) shows the prediction of each label after the first Feature-to-Label message passing. The second row (1.2) shows the prediction of each label after the first Label-to-Label message passing. This is then repeated in (2.1) and (2.2) for the second layer's output states, where the final output, 2.2 is the network's final output predictions. The most important aspect of this figure is that we can see the labels "design", "html", and "web design", all change from weakly positive to strongly positive after the first Label-to-Label message passing step (row 1.2). In other words, this indicates that these labels change to a strongly positive prediction by passing messages between each other.

Label-to-Feature Attention. While the iterative prediction visualization shows how the model updates its prediction of each label, it doesn't explicitly show how or why. To understand why each label changes its predictions, we first look at the Feature-to-Label attention, which tells us the input nodes

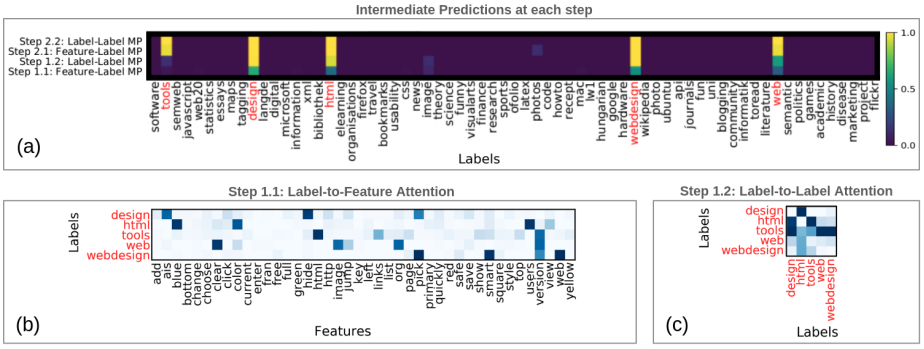

Fig. 2. (a) Visualization of Model Predictions and Attention Weights Intermediate Predictions: this shows the readout function predictions for each intermediate state in the two update steps. **(b) Label-to-Feature Attention Weights** for the first step of Feature-to-Label message passing (1.1). **(c) Label-to-Label Attention Weights** for the first step of Label-to-Label message passing (1.2).

that each label node attends to in order to update its state (and thus producing the predictions in Figure 2 (a.)). Figure 2 (b.) shows us which input nodes (i.e. features) each of the positive label attends to in order to make its first update step 1.1. The colors represent the post-softmax attention weight (summed over the 4 attention heads), with the darker cells representing high attention. In this example, we can see that the "web design" label attends to the "pick", "smart", and "version" features, but as we can see from the first row of Figure 2 (a.), prediction for the current state of the "web design" label isn't very strong yet.

Label-to-Label Attention. Label-to-Feature attention shows us the input nodes that each label node attends to in order to make its first update, but the second step of the label graph update is the Label-to-Label message passing step where labels are updated according to the states of all other nodes after the first Feature-to-Label message passing. Figure 2 (c.) shows us the first Label-to-Label attention stage 1.2 where each label node can attend to the other label nodes in order to update its state. Here we show only the Label-to-Label attention for the positive labels in this example. We then look at the second row of Figure 2 (a.) which shows the model's prediction of each label node after the Label-to-Label attention. The interesting thing to note is we can see many of the true positive labels change their state to positive after the positive labels attend to each other during the Label-to-Label attention step, indicating that dependencies are learned.

Attention weights for the second step $t = 2$ are not as interpretable since they model higher order interactions. We have added these plots in Appendix Fig. 3.

Dataset	Training	Testing
Reuters	0.788 (1.5x)	0.116 (2.1x)
Bibtex	0.376 (2.1x)	0.080 (2.1x)
Delicious	3.172 (1.1x)	0.473 (3.2x)
Bookmarks	9.664 (1.2x)	1.849 (1.3x)
RCV1	98.346 (1.2x)	1.003 (1.7x)
TFBS	187.14 (2.5x)	13.04 (4.2x)
NUS-WIDE	3.201 (1.2x)	0.921 (8.0x)
SIDER	0.027 (2.5x)	0.003 (21x)

Table 4. Speed. Each column shows training or testing speed for LaMP in minutes per epoch. Speedups over RNN Seq2Seq are in parentheses. Since LaMP does not depend on sequential prediction, we see a drastic speedup, especially during testing where RNN Seq2Seq requires beam search.

4 Conclusion

In this work we present Label Message Passing (LaMP) Networks which achieve better than, or close to the same accuracy as previous methods across five metrics and seven datasets. In addition, the iterative label embedding updates with attention of LaMP provide a straightforward way to shed light on the model's predictions and allow us to extract three forms of visualizations, including conditional label dependencies which influence MLC classifications.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Battaglia, P., Pascanu, R., Lai, M., Rezende, D.J., et al.: Interaction networks for learning about objects, relations and physics. In: Advances in neural information processing systems. pp. 4502–4510 (2016)
3. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks. arXiv:1806.01261 (2018)
4. Belanger, D., McCallum, A.: Structured prediction energy networks. In: International Conference on Machine Learning. pp. 983–992 (2016)
5. Bhatia, K., Jain, H., Kar, P., Varma, M., Jain, P.: Sparse local embeddings for extreme multi-label classification. In: Neural Information Processing Systems
6. Dai, H., Dai, B., Song, L.: Discriminative embeddings of latent variable models for structured data. In: International Conference on Machine Learning
7. Debole, F., Sebastiani, F.: An analysis of the relative hardness of reuters-21578. American Society for Information Science and Technology **56**(6), 584–596 (2005)
8. Do, K., Tran, T., Nguyen, T., Venkatesh, S.: Attentional multilabel learning over graphs-a message passing approach. arXiv preprint arXiv:1804.00293 (2018)
9. Ghamrawi, N., McCallum, A.: Collective multi-label classification. In: 14th ACM international conference on Information and knowledge management
10. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212 (2017)

11. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
12. Guo, Y., Gu, S.: Multi-label classification using conditional dependency networks. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence
13. Gygli, M., Norouzi, M., Angelova, A.: Deep value networks learn to evaluate and iteratively refine structured outputs. arXiv preprint arXiv:1703.04363 (2017)
14. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 (2017)
15. Hara, K., Saitoh, D., Shouno, H.: Analysis of dropout learning regarded as ensemble learning. In: International Conference on Artificial Neural Networks
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
17. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data (2001)
18. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern recognition **45**(9) (2012)
19. Nam, J., Mencía, E.L., Kim, H.J., Fürnkranz, J.: Maximizing subset accuracy with recurrent neural networks in multi-label classification. In: Advances in Neural Information Processing Systems. pp. 5419–5429 (2017)
20. Prabhu, Y., Varma, M.: Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 263–272. ACM (2014)
21. Press, O., Wolf, L.: Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859 (2016)
22. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning and Knowledge Discovery in Databases
23. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80 (2009)
24. Su, H., Rousu, J.: Multilabel classification through random graph ensembles. In: Asian Conference on Machine Learning. pp. 404–418 (2013)
25. Szklarczyk, D., Morris, J.H., Cook, H., Kuhn, M., Wyder, S., Simonovic, M., Santos, A., et al.: The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible. Nucleic acids research p. gkw937 (2016)
26. Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Large margin methods for structured and interdependent output variables. JMLR **6**(Sep), 1453–1484 (2005)
27. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. International Journal of Data Warehousing and Mining **3**(3) (2006)
28. Tu, L., Gimpel, K.: Learning approximate inference networks for structured prediction. arXiv preprint arXiv:1803.03376 (2018)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 6000–6010 (2017)
30. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
31. Vinyals, O., Bengio, S., Kudlur, M.: Order matters: Sequence to sequence for sets. arXiv preprint arXiv:1511.06391 (2015)
32. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: Cnn-rnn: A unified framework for multi-label image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2285–2294 (2016)
33. Yeh, C.K., Wu, W.C., Ko, W.J., Wang, Y.C.F.: Learning deep latent space for multi-label classification. In: AAAI. pp. 2838–2844 (2017)