Multi-Label Classification using Attention based Graph Neural Network

Ankit Pal[1], Muru Selvakumar[2], Malaikannan Sankarasubbu[3]
{[1]ankit.pal, [2]smurugan, [3]malaikannan.sankarasubbu}@saama.com

Saama AI Research
Chennai, India

Abstract—Multi-label text classification is a task of assigning a set of labels to one sentence. It is observed that in most of the multi-label classification tasks there could be a clear dependency or correlation among labels. Existing methods tend to ignore the correlations between labels. We propose a Graph attention network based model to capture the attentive dependency structure among the labels. The graph attention network uses a feature matrix and correlation matrix to capture and explore the important dependencies between the labels and generate classifiers for classification task, After generating classifiers we apply this classifier to sentence features vectors generated by another sub-net (LSTM) to enable end to end training.

Attention enables MAGENT to assign different importances to neighbor nodes per label, learning how labels interact (implicitly). We validate the benefits of MAGNET on five real-world MLTC datasets, covering a broad spectrum of input/output types and our proposed methods achieves similar, or better performance compared to the previous state of the art.

1 Introduction

Multi-label text classification (MLTC) is an important and challenging task in Natural language processing(NLP) field, Multi label text classification assign more than one labels to each samples in corpus. which means that one text can belong to more classes simultaneously.
we are given a set of labeled training data $\{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \epsilon \mathbb{R}^p$ are the input features for each data instances and $y_i \epsilon \{0,1\}^d$ The vector y_i has a one in the jth coordinate if the ith data point belongs to jth class. We wish to learn a mapping (prediction rule) between the features and the labels, such that, we can predict the class label vector y of a new data point x correctly.

It can be applied in many real-world scenarios, such as text categorization (Schapire and Singer 2000) [20], tag recommendation (Katakis, Tsoumakas, and Vlahavas 2008) [10] , information retrieval (Gopal and Yang 2010) [8], and so on. Before the deep learning era, MLTC task used to focuses on traditional machine learning algorithms. Different techniques have been proposed in the literature for treating multi-label classification problems. In some of them, single-label classifiers are combined to treat multi-label classification problems. Other techniques modify single-label classifiers, changing their algorithms to allow their use in multi-label problems.

Most famous and traditional method for solving multi label text classification is binary relevance (BR) [28] , binary relevance (BR) decomposes the MLTC task into independent binary classification problems. However, in many MLTC tasks there is a clear dependency structure or correlation among labels, which BR methods ignore [15] . Thats why it is commonly criticised in the literature because of its label independence assumption. This stimulates research for approaches to capture and explore the label correlations in various ways. Some approaches, based on probabilistic graph model or Recurrent

Neural Networks (RNNs) [1], are proposed to explicitly model label dependencies.Unfortunately, accurately modelling all combinatorial label interactions is an NP-hard problem. Many types of models, including a few deep neural network (DNN) based and probabilistic model , have been introduced to approximately model such interactions, thus boosting classification accuracy. Our main concern of this paper is how to capture the topology structure between labels.

Recently Graph based neural networks [26] e.g. Graph convolution network [12] , Graph attention networks [23] and Graph embeddings [2] has received considerable research attention. That is due to the fact that many real-world problems in complex systems, such as recommendation system [27] , social networks and biology networks [7] etc, can be modelled as machine learning tasks over large network. Graph convolutional network (GCN), as an extension of the CNN, was proposed to deal with graph structures. The GCN benefits from the advantage of the CNN architecture; it performs with a high accuracy but a relatively low computational cost by utilizing fewer parameters compared to a fully connected multi-layer perceptron (MLP) model. It can also capture important sentence features that determine node properties by analyzing relations between neighboring nodes. Despite the aforementioned advantages, we suspect that the GCN is still missing an important structural feature to capture better correlation or decencies between nodes.

To improve performance of the GCN, one possible approach is to add adaptive attention weights depending on feature matrix to graph convolutions. The so-called graph attention network was originally developed for a network analysis in computer science. To resolve and capture the correlation between labels we propose a Graph attention based novel deep learning architecture, the resulting neural network optimizes attention weights in a learning process of the correlation between labels.

Our proposed model with graph attention allows us to capture dependency structure among labels for multi label classification task. As a result, the correlation between labels can be automatically learned based on feature matrix. we propose to learn inter-dependent sentence classifiers from prior label representations (e.g. word embeddings) via an Attention based function. The key idea of our method is to rely on attention-based graph neural network entirely to draw dependencies from labels. We name the proposed method Multi label classification using attention based graph neural network (MAGNET). it uses multi-head attention mechanism to capture the correlation between labels for multi label classification task.

We use graph attention network to generate the classifiers for multi-label task. After generating classifiers we apply this classifier to sentence feature vectors generated by another sub-net (bi-directional LSTM) to enable end to end training.
Specifically, we make the following contributions:

- We systematically analyze the drawbacks of the current models for the multi-label text classification task.
- We propose a novel deep learning end-to-end trainable network for multi-label text classification task, which employs GAT to find the correlation between labels.
- Extensive experimental results show that our proposed methods achieves similar, or better performance compared to the previous state of the art across two MLTC metrics. We validate our model on five MLTC datasets.

the MLTC task can be modeled as finding an optimal label sequence y^* that maximizes the conditional probability $p(y \mid x)$, which is calculated as follows:

$$\mathbf{p(y \mid x) = \prod_{i=1}^{n} p(y_i \mid} y_1, y_2, .., y_{i-1}, x) \qquad (1)$$

For example In slashdot dataset a document might be about any of Entertainment, Developers, News, Games at the same time of none of these.

2 Learning Algorithms

Multi-label classification methods can be grouped in two categories:

1) Problem Transformation Methods. [3]
2) Algorithm Adaptation Methods. [21]

Problem transformation methods transform the multi-label classification problem either into one or more single-label classification or regression problems. There exist many simple problem transformation methods that transform multi-label Dataset

into single-label dataset so that existing single-label classifier can be applied to Multilabel dataset.

Algorithm adaptation, on the other hand, extends specific learning algorithms in order to handle multi-label data directly. Our proposed model comes under Algorithm Adaptation Methods category.

3 MAGNET ARCHITECTURE

3.1 Motivation

Multi-label text classification task assigns set of target labels to a single sentence. This can be thought as predicting properties of a data-point that are not mutually exclusive. We propose a novel architecture to capture the dependencies between target variables and use those dependencies to classify the data-points into different set of labels. we use graph attention network to capture the relation between labels, which is a flexible way to capture the topological structure in the label space. We use word2vec to obtain labels2vec embeddings, we use random vector for those labels which are not in pre-trained word embeddings. Distributed representations of target labels in a vector space helps learning algorithms to achieve better performance in natural language processing tasks by grouping similar labels. If two target words share the some context or semantic meaning, intuitively the weight of the network for this two target words will be close to each other and thus their matching vectors.

3.2 Graph representation of labels

A label graph $\mathcal{G}$ is consists of feature description $M \in \mathbb{R}^{n \times d}$ and the corresponding correlation matrix $A \in \mathbb{R}^{n \times n}$ (where n denotes the no of labels and d denote the no of categories) We represent each node (label) of the graph as word embedding of the label and from here we get label2vec which describe the different semantic meaning of words in multi-dimensional vector space.

We create co-occerence matrix by counting the occurrence of labels pairs in training set basis on conditional probability.Graph attention network compute attention coefficients e_{ij} across pairs of labels i,j basis on this co-relation matrix.

3.3 Node updating Mechanism in Graph convolution

Nodes in graph structure can be updated by different types of node updating mechanism, The most general form introduced by Gilmer et al. adopts a message passing framework given by

$$\mathbf{H_i^{(l+1)}} = \mathbf{U}\left(\mathbf{H_i^{(l)}}, \mathbf{m_i^{(l+1)}}\right)$$

Here ith node state in graph is updated as a function of the previous node state, $H_i^{(l)}$ and a message state containing pair-wise interaction terms with its neighbours in graph $m_i^{(l+1)}$. The GCN is the basic version of message passing neural network. GCN updates each node state of the ith layer $H_i^{(l+1)}$

$$\mathbf{H^{(l+1)}} = \sigma\left(\mathbf{A H^{(l)} W^{(l)}}\right) \tag{2}$$

Where $\sigma(\cdot)$ denote an activation function, A is an adjacency matrix and $W^{(l)}$ is convolution weights of the lth layer. If one label or node; eg. label 2 has three adjacent labels 1, 3 and 4, In this case, equation (2) can be rewritten as

$$\begin{aligned}\mathbf{H_2^{(l+1)}} = \sigma\Big(&\mathbf{H_2^{(l)} W^{(l)}} + \mathbf{H_1^{(l)} W^{(l)}} \\ &+ H_3^{(l)} W^{(l)} + H_4^{(l)} W^{(l)}\end{aligned} \tag{3}$$

So in this case graph convolution network sum up all labels features with same convolution weights and then result is pass through one activation function to produce the update node feature output

3.4 Graph Attention networks for multi-label classification

Attention is one of the most influential ideas in the Deep Learning community. For example Attention mechanisms are being increasingly used to improve the performance of Neural Machine Translation (NMT) by selectively focusing on sub-parts of the sentence during translation. [26-28]

In GCNs, the neighborhoods of nodes are combined with equal or pre-defined weights. However, the influence of neighbors can vary greatly, Attention mechanism can be applied to the GCN to capture the relationship between adjacent labels basis on correlation feature matrix.

Node updating mechanism equation (3) can be written as a linear combination of neighbouring labels states with attention coefficients

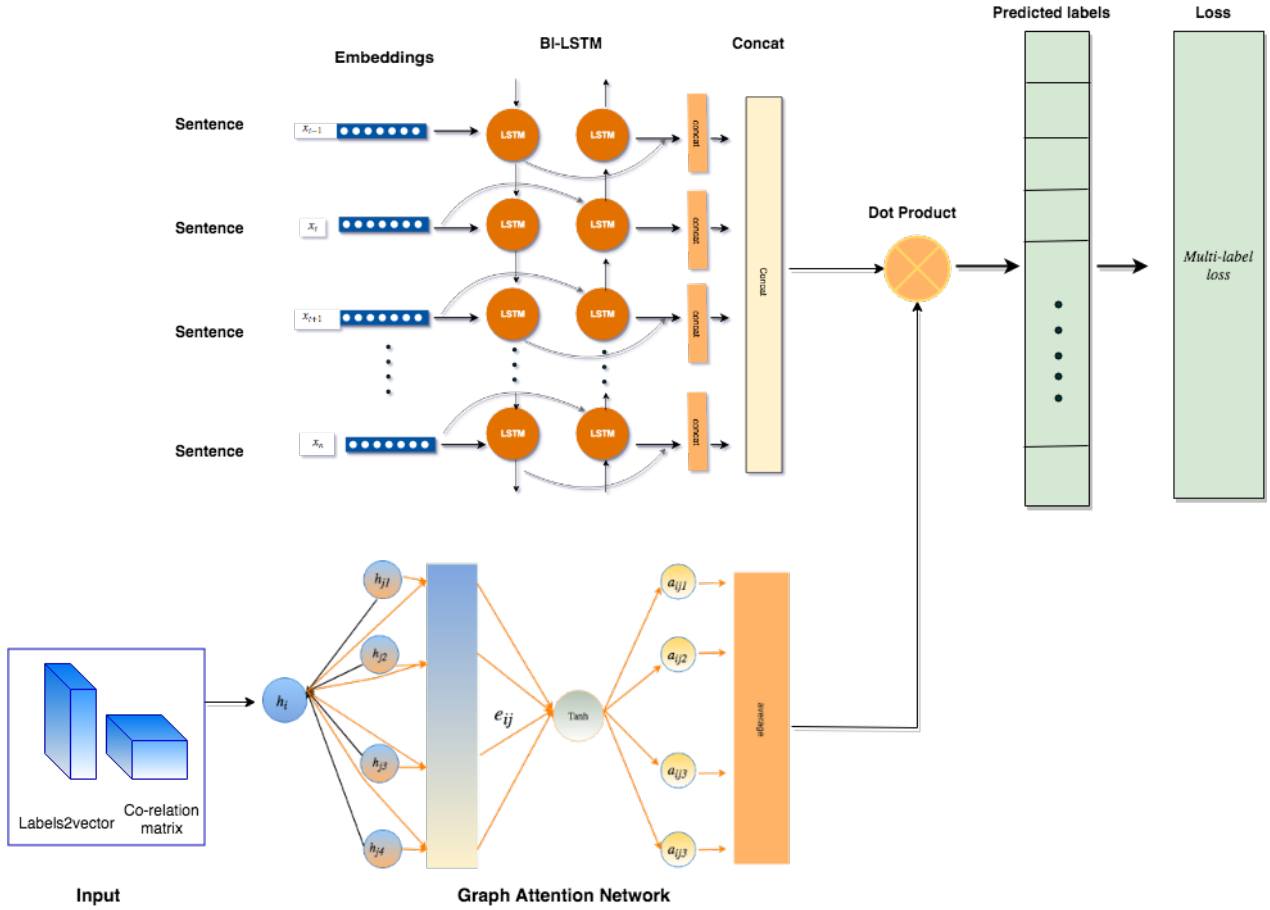

Figure 1: **Illustration of overall structure of MAGNET model for multi label text classification.** $\left(x^{(n)}, \mathbf{y}^{(n)}\right), n = 1, 2, \ldots, N$ **is input for bi-lstm to generate the feature vectors.** $x_{(n)}$ **are encoded using bert embedding. Input for Graph attention network is correlation matrix** $A \in \mathbb{R}^{n \times n}$ **and label2vec** $H^l \in \mathbb{R}^{n \times d}$ **. Label2vec obtain from encoding the labels using word2vec. Gat output is attentive classifiers which are applied on feature vectors obtained from bi-lstm for multi label text classification.**

$$\mathbf{H_2^{(l+1)}} = \sigma(\alpha_{\mathbf{22}}^{(\mathbf{l})}\mathbf{H_2^{(l)}}\mathbf{W^{(l)}} + \alpha_{\mathbf{21}}^{(\mathbf{l})}\mathbf{H_1^{(l)}}\mathbf{W^{(l)}} + \alpha_{23}^{(l)}H_3^{(l)}W^{(l)} + \alpha_{24}^{(l)}H_4^{(l)}W^{(l)}) \tag{4}$$

where $\alpha_{ij}^{(l)}$ is an attention coefficient which measures the importance of the jth node in updating the ith state of the lth hidden layer basis on correlation matrix. Basic expression of the attention coefficient can be written as

$$\alpha_{\mathbf{ij}}^{(\mathbf{l})} = \mathbf{f}\left(\mathbf{H_i^{(l)}}\mathbf{W^{(l)}}, \mathbf{H_j^{(l)}}\mathbf{W^{(l)}}\right) \tag{5}$$

First, GCN with the attention mechanism used a concatenation of node features to attention coefficient

$$\alpha_{\mathbf{ij}} = \frac{\mathbf{e_{ij}}}{\sum_{\mathbf{k \in N(i)}} \mathbf{e_{ik}}} = \frac{\sigma\left(MLP\left[H_iW, H_jW\right]\right)}{\sum_{k \in N(i)} \sigma\left(MLP\left[H_iW, H_kW\right]\right)} \tag{6}$$

where α_{ij} is the attention coefficient of label j to i, $N(i)$ represents the neighborhoods of label i in the graph.

Then the final output features of each node can be obtained by (after applying a nonlinearity σ)

$$\mathbf{h}_i' = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{W}\mathbf{h}_j\right) \tag{7}$$

In our experiment we are using multi-head attention [22] that utilizes K different heads to describe

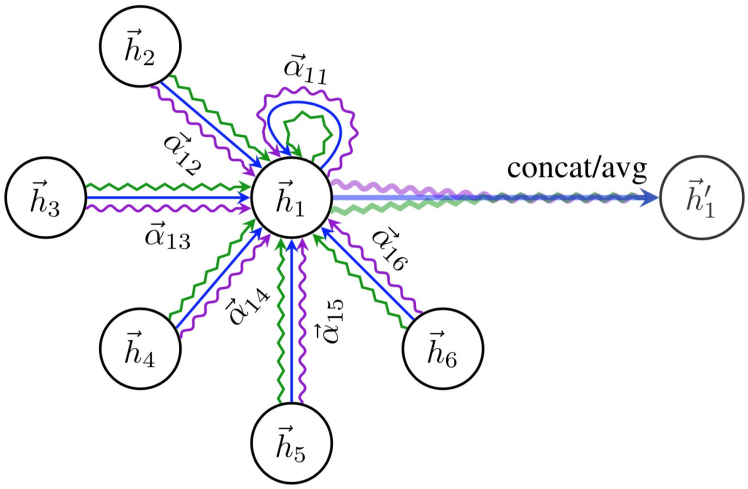

Figure 2: **illustration of multihead attention which is using k = 3 heads. Every neighbour i of node 1 sends its own vector of attentional coefficients $\vec{\alpha}_{1I}$ one per each attention head α^k_{1i}. These are used to compute K separate linear combinations of neighbours feature $\vec{h}_I$ which are then aggregated to get the next level features of node 1 $\vec{h}'_1$.**[23]

labels relationships, the operations of the layer are independently replicated K times (each replica with different parameters), and outputs are featurewise aggregated (typically by concatenating or adding).

$$H_i^{(l+1)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in N(i)}\alpha_{ij,k}^{(l)}H_j^{(l)}W^{(l)}\right) \tag{8}$$

If the feature description is $M \in \mathbb{R}^{n\times d}$ and the corresponding correlation matrix is $A \in \mathbb{R}^{n\times n}$ (where n denotes the no of labels and d denote the no of categories) then

$$\mathbf{C} = \mathbf{f_{gat}}(\mathbf{M}; \mathbf{A}; \theta_{\mathbf{gat}}) \tag{9}$$

where θ_{gat} indicates gat parameters, M indicates label2vec matrix and A indicates correlation matrix

Output from GAT is attentive dependency matrix $\mathbf{C} \in \mathbb{R}^{\mathbf{c}\times\mathbf{d}}$ which are applied to the textual features extracted from the encoded text data via a LSTM (long short term memory) for multi-label classification task.

3.5 Feature vector generation

For feature extraction from textual dataset we are using bi-directional long short term memory network (LSTM) with RNN [19, 24, 25]

A detailed description of the original long short-termmemory (LSTM) algorithm can be found in [9]. In LSTM networks, all weights were adapted using a simplified RealTime Recurrent Learning (RTRL) algorithm having $O(1)$ per time step and perweight. Formally, the formulas to update an LSTM unit at time t are:

$$\begin{aligned}
\mathbf{i}_t &= \sigma\left(\boldsymbol{W}_i\mathbf{h}_{t-1} + \boldsymbol{U}_i\mathbf{x}_t + \boldsymbol{b}_i\right)\\
\mathbf{f}_t &= \sigma\left(\boldsymbol{W}_f\mathbf{h}_{t-1} + \boldsymbol{U}_f\mathbf{x}_t + \boldsymbol{b}_f\right)\\
\tilde{\mathbf{c}}_t &= \tanh\left(\boldsymbol{W}_{c-1}\mathbf{h}_{t-1} + \boldsymbol{U}_c\mathbf{x}_t + \boldsymbol{b}_c\right)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t\\
\mathbf{o}_t &= \sigma\left(\boldsymbol{W}_o\mathbf{h}_{t-1} + \boldsymbol{U}_o\mathbf{x}_t + \boldsymbol{b}_o\right)\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t\right)
\end{aligned}$$

BLSTM For many sequence labeling tasks it is beneficial to have access to both past (left) and future (right) contexts. However, the LSTMs hidden state ht takes information only from past, knowing nothing about the future. An elegant solution (Dyer et al., 2015) is bi-directional LSTM (BLSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output.

We use Rnn and bi-directional lstm to obtain the feature vectors. We first encode the sentences using

Bert's feature vectors then we feed it to lstm for fine tuning for domain specific task .
We use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997) to read the text sequence x from both directions and compute the hidden states for each word,

$$\begin{aligned}\overrightarrow{\boldsymbol{h}}_i &= \overrightarrow{\mathrm{LSTM}}\left(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{x}_i\right) \\ \overleftarrow{\boldsymbol{h}}_i &= \overleftarrow{\mathrm{LSTM}}\left(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{x}_i\right)\end{aligned} \tag{10}$$

We obtain the final hidden representation of the i-th word by concatenating the hidden states from both directions, $\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}}_i\right]$, which embodies the information of the sequence centered around the I-th word.

If input sentence i is $B \times M \times D$ then we get $2 \times (B \times H)$ from bi-directional lstm Where B is batch size , M is max sentence length (padded sentences) D is the dimension of vector and H is the hidden size of rnn

$$\mathbf{F} = \mathbf{f_{rnn}}(\mathbf{f_{bert}}(\mathbf{I}; \theta_{\mathbf{bert}}); \theta_{\mathbf{rnn}}) \in \mathbb{R}^{\mathbf{D}} \tag{11}$$

where θ_{rnn} indicates rnn parameters , θ_{bert} indicates Berts parameter D = hidden dim of lstm later we use multiply attentive dependency matrix with feature vectors to get final prediction score as

$$\hat{\boldsymbol{y}} = \boldsymbol{C} \odot \boldsymbol{F} \tag{12}$$

Where C is $\mathbf{C} \in \mathbb{R}^{\mathbf{c} \times \mathbf{d}}$ and F is feature vectors obtained from LSTM model.

3.6 Co-relation matrix generation

To generate the correlation matrix of labels, firstly, we count the the occurrence of label pairs in the training set from here we obtain the matrix $D \in \mathbb{R}^{c \times c}$. Where c is the number of categories and D_{ij} shows the concurring times of L_i and L_j. After we get conditional probability matrix using this correlation matrix by

$$\boldsymbol{P}_{\mathbf{i}} = \boldsymbol{D}_{\mathbf{i}} / \mathbf{N}_{\mathbf{i}} \tag{13}$$

N_i denote the occurrence of times of label L_i in dataset and

$$\mathbf{P}_{\mathbf{ij}} = \mathbf{P}\left(\mathbf{L}_{\mathbf{j}} | \mathbf{L}_{\mathbf{i}}\right) \tag{14}$$

means the probability of label L_j when label L_i appears.

3.7 Loss function

For multi-label classification, we can minimize the Cross-Entropy over all labeled node between the ground-truth and the prediction, If the ground truth label of a data point is $Y \in \mathbb{R}^c$, where $y_i = \{0, 1\}$

$$\mathcal{L} = \sum_{\mathbf{c=1}}^{\mathbf{C}} \mathbf{y^c} \log\left(\sigma\left(\hat{\mathbf{y}}^{\mathbf{c}}\right)\right) + \left(\mathbf{1} - \mathbf{y^c}\right) \log\left(\mathbf{1} - \sigma\left(\hat{\mathbf{y}}^{\mathbf{c}}\right)\right) \tag{15}$$

Where σ is sigmoid activation function

4 Experiment

In this section, we evaluate our proposed methods on five datasets. We first introduce the datasets, evaluation metrics, experimental details, and all baselines. Then, we compare our methods with the baselines. Finally, we provide the analysis and discussions of experimental results.

we report the result on benchmark multi label text datasets Reuters-21578 (90 classes, 11K records), the RCV117 (103 classes, 800K records), Arxiv Academic Paper Dataset (AAPD), Slashdot (Read et al., 2011) and toxic comment dataset(7 classes, 159571 records)

4.1 Datasets

Reuters-21578 : The documents in this dataset were collected from the Reuters newswire in 1987. The Reuters-21578 test collection, together with its earlier variants, has been such a standard benchmark for the text categorization (TC). [4] This was once a popular dataset for researchers who worked in text categorization since 1996 (now superseded by RCV1). Adapted from this preview Reuters-22173 version.

RCV1-V2 : This dataset is provided by Lewis et al. (2004) [13]. It consists of over 800,000 manually categorized newswire stories made available by Reuters Ltd for research purposes. Multiple topics can be assigned to each newswire story and there are 103 topics in total.

Arxiv Academic Paper Dataset (AAPD) : This dataset is provided by Yang et al. (2018). The dataset consists of the abstract and corresponding subjects of 55,840 academic papers, each paper can have multiple subjects. The target is to predict subjects of an academic paper according to the content of the

abstract.

Slashdot: The slashdot dataset was collected from the slashdot website and consists of articles blurbs labelled with the subject categories. Slashdot dataset contains 24,072 samples.

Toxic comment dataset : The dataset we are using for toxic comment classification is taken from Kaggle competition which can be found at Kaggle. Dataset has a large number of comments from Wikipedia talk page edits. They have been labeled by human raters for toxic behavior.

4.2 Performance Evaluation

miF1 micro-average will aggregate the contributions of all classes to compute the average metric. Table 1 shows the miF1 scores, for the all datasets. In Micro-average method, you sum up the individual true positives, false positives, and false negatives of the system for different sets and the apply them to get the statistics. It's the weighted average of F1 score of each class.

$$Precision_{micro} = \frac{\sum_{\mathbf{l=1}}^{\mathbf{L}} \mathbf{TP_l}}{\sum_{\mathbf{l=1}}^{\mathbf{L}} \mathbf{TP_l} + \mathbf{FP_l}} \tag{16}$$

$$Recall_{micro} = \frac{\sum_{\mathbf{l=1}}^{\mathbf{L}} \mathbf{TP_l}}{\sum_{\mathbf{l=1}}^{\mathbf{L}} \mathbf{TP_l} + \mathbf{FN_l}} \tag{17}$$

$$F1 - Score_{micro} = \frac{\sum_{\mathbf{l=1}}^{\mathbf{L}} \mathbf{2TP_l}}{\sum_{\mathbf{l=1}}^{\mathbf{L}} \mathbf{2TP_l} + \mathbf{FP_l} + \mathbf{FN_l}} \tag{18}$$

Hamming loss Hamming Loss (HL): Hamming Loss reports how many times on average, the relevance of an example to a class label is incorrectly predicted [5]. Therefore, hamming loss takes into account the prediction error (an incorrect label is predicted) and the missing error (a relevant label not predicted), normalized over total number of classes and total number of examples.

$$\frac{\mathbf{1}}{|\mathbf{N}| \cdot |\mathbf{L}|} \sum_{\mathbf{i=1}}^{|\mathbf{N}|} \sum_{\mathbf{j=1}}^{|\mathbf{L}|} \text{xor}\left(\mathbf{y_{i,j}}, \mathbf{z_{i,j}}\right) \tag{19}$$

where $y_{i,j}$ is the target and $z_{i,j}$ is the prediction. Ideally, we would expect hamming loss, $HL = 0$, which would imply no error; practically the smaller the value of *hamming loss*, the better the performance of the learning algorithm.

4.3 Baseline Models

We compare our proposed methods with the following baseline Models:

- **Binary Relevance (BR)** transforms the MLC task into multiple single-label classification problems by ignoring the correlations between labels, Independently training one binary classifier for each label.
- **Classifier Chains (CC)** transforms the MLC task into a chain of binary classification problems and takes high-order label correlations into consideration.[18]
- **CNN (Kim, 2014)** uses multiple convolution kernels to extract text feature, which is then input to the linear transformation layer followed by a sigmoid function to output the probability distribution over the label space. [31]
- **CNN-RNN (Chen et al., 2017)** using CNN and RNN to capture both global and local textual semantics and model label correlations. [11]
- **Seq2Seq (Yang et al. 2018)** applies the sequence-tosequence model (Bahdanau, Cho, and Bengio 2014) to perform multi-label text classification. [14]

5 Result

For the purpose of simplicity, we denote the proposed graph model as MAGNET. We report the evaluation results of our methods and all baselines on the test sets. The experimental results of our methods and the baselines on dataset RCV1-V2 are shown in Table 1 Results show that our proposed methods give the best performance in the main evaluation metrics.

Our proposed MAGNET model using LSTM for feature vectors achieves an improvement of 2% micro-F1 score on reuters dataset over the most commonly used baseline BR. Besides, our methods outperform other traditional deep-learning models.

We test various algorithms over the mentioned datasets, including Binary relevance (BR), Classifier chain (CC), CNN, CNN-RNN, Seq2Seq model. BR is used to represent multi-label algorithms without taking into account the correlation between labels.

Algorithm 1: The overall process of MAGNET

Input: Multi-label dataset :
$\left(x^{(n)}, \mathbf{y}^{(n)}\right), n = 1, 2, \ldots, N$;
A corelation matrix $A \in \mathbb{R}^{n \times n}$;
A feature matrix $M \in \mathbb{R}^{n \times d}$

Output: Predicted label set $\widehat{\mathbf{y}}$

Feature vector from bi-directional LSTM;
for *each epoch* **do**
  Feature vector from bi-directional LSTM;
  for *each batch* **do**
    1) Compute *x* using equation (13)
    $F = f_{rnn}(f_{bert}(I; \theta_{bert}); \theta_{rnn}) \in \mathbb{R}^D$
    Compute forward pass for lstm
    $\overrightarrow{\boldsymbol{h}}_i = \overrightarrow{\mathrm{LSTM}}\left(\overrightarrow{\boldsymbol{h}}_{i-1}, \boldsymbol{x}_i\right)$
    Compute backward pass for lstm
    $\overleftarrow{\boldsymbol{h}}_i = \overleftarrow{\mathrm{LSTM}}\left(\overleftarrow{\boldsymbol{h}}_{i+1}, \boldsymbol{x}_i\right)$
    concatenating the hidden states from both directions
    $\boldsymbol{h}_i = \left[\overrightarrow{\boldsymbol{h}_i}; \overleftarrow{\boldsymbol{h}}_i\right]$
    2) Compute *W* using equation (10)
    $\mathbf{C} = \mathbf{f_{gat}}(\mathbf{M}; \mathbf{A}; \theta_{\mathbf{gat}})$
    compute mulit-head attention using equation (9)
    $\mathbf{H_i^{(l+1)}} = \sigma\left(\frac{\mathbf{1}}{\mathbf{K}} \sum_{\mathbf{k=1}}^{\mathbf{K}} \sum_{\mathbf{j} \in \mathbf{N(i)}} \alpha_{\mathbf{ij,k}}^{\mathbf{(l)}} \mathbf{H_j^{(l)}} \mathbf{W^{(l)}}\right)$
    Matrix multiplication
    $\widehat{\mathbf{y}} = F \odot C$
    Compute cross entropy
    $\mathcal{L} = \sum_{\mathbf{c=1}}^{\mathbf{C}} \mathbf{y^c} \log\left(\sigma\left(\hat{\mathbf{y}}^{\mathbf{c}}\right)\right) +$
    $(1 - y^c) \log\left(1 - \sigma\left(\hat{y}^c\right)\right)$
    loss = reduce (cross entropy)
    update the parameters basis on loss using back propagation
    $\theta_{\mathbf{t+1}} = \theta_{\mathbf{t}} - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_{\mathbf{t}} + \epsilon}} \hat{\mathbf{m}}_{\mathbf{t}}$
  end
end

5.1 Comparisons with State-of-the-Arts

Results on Reuters-21578 We first present the results of the MAGNET on Reuters-21578. Reuters-21578 dataset has been used by many researchers to test the performance of the multi label text classification models. we report the benchmark performance on Reuters-21578 dataset, it contains 10, 788 documents which are divided into 8, 630 documents for training and 2, 158 for testing with total of 90 classes. results are reported in Table 1. We compare with state-of-the-art methods, including Binary Relevance (BR), Classifier Chains (CC), CNN, CNN-RNN, Seq2Seq.

Comparing with state-of-the-art methods, our approach with the proposed scheme consistently performs better under almost all metrics, which shows the effectiveness of our proposed model MAGNET, which achieve better than, or close to the same accuracy as previous methods across four datasets.

Results on AAPD AAPD is another dataset which is widely used benchmark for multi label text classification task. AAPD contains 55,840 documents which are divided into 44,672 documents for training and 11,168 for testing with total of 54 classes. Performance of AAPD dataset is shown in Table 1.

MAGNET yield higher accuracy than previous methods on AAPD dataset, We also noticed that the correlation matrix have a strong influence on the results. We can witness the importance of a graph satisfying the data assumptions by comparing its performance without correlation matrix.

Result on RCV1-V2 and Slashdot We tested our proposed model on two other benchmark text classification datasets, RCV1-V2 and Slashdot. RCV1-V2 contains 8,04,414 documents which are divided into 6,43,531 documents for training and 1,60,883 for testing with total of 103 classes where Slashdot contains19,258 samples for training and 4,814 samples for testing with total of 291 classes. Performance on both dataset is shown in Table 1

The proposed method increased the accuracy of text classification . CNN-RNN has the lowest accuracy and RNN-seq2seq has the highest accuracy among all five methods.The results show that the accuracy of RCV1-V2 and Slashdot were increased by 1% and 2% respectively. All the models were implemented in Google TensorFlow and trained using the Adam stochastic optimization algorithm with learning rate 102 Results in Table 1 reflect that our model performs the slightly better in comparison with models with the other types of architecture.

Results testifies that the graph network is capable of helping the model to capture the correlation between the labels more accurately. However, if we

can grasp better embedding vectors for label embeddings, the MAGNET model will perform better.

Results on Toxic comment dataset We tested our model on toxic comment classification dataset which is taken from Kaggle competition. our model performed very well on toxic dataset. Table 1 shows the accuracy (F1-score) of Toxic comment dataset.

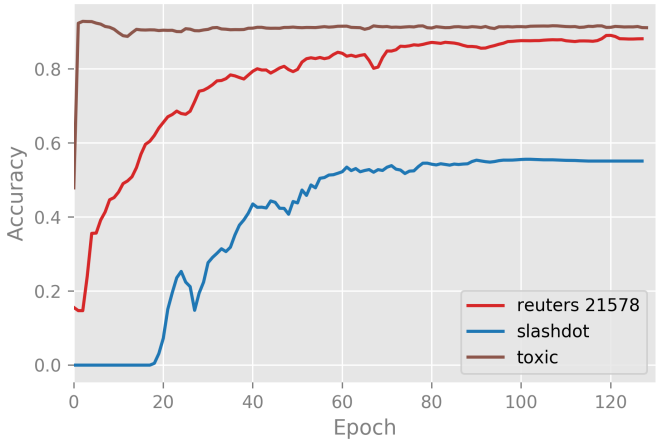

Figure 3: F1-accuracy of proposed model on Reuters, Slashdot and Toxic dataset. The x-axis refers to the number of epoch and the y-axis refers to the micro-F1 score.

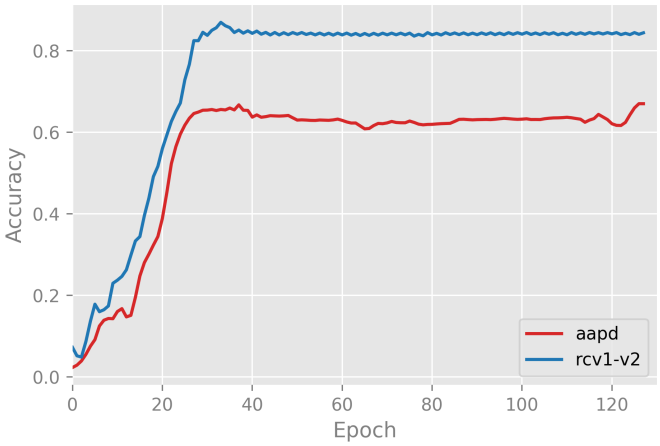

Figure 4: F1-accuracy of proposed model on Rcv1-v2 and Aapd dataset. The x-axis refers to the number of epoch and the y-axis refers to the micro-F1 score.

results demonstrating the significance of being able to assign different weights to different neighbors.

5.2 Analysis and Discussion

Here we perform and discuss further analysis on the model and experimental results. We report the evaluation results in terms of hamming loss and micro-F1 score. We are using moving average to draw the plots.

5.2.1 Exploring the Impact of the Correlation matrix

Co-relation matrix can significantly improve the performance of the model, because MAGNET works by propagating and assign attention to nodes based on the correlation matrix, graph attention networks employ attention mechanisms which assign larger weights to the more important nodes, walks, or models. The attention weight is learned together with neural network , parameters within an end-to-end framework.

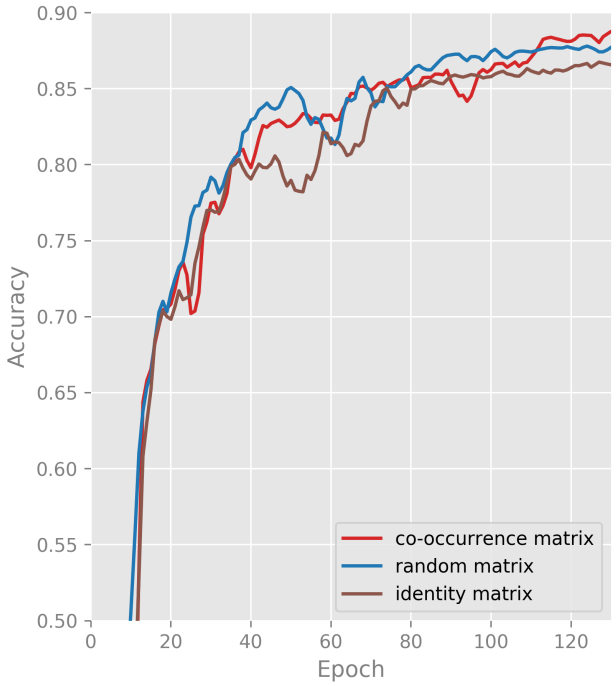

Figure 5: Performance of different types of correlation matrix. Here x-axis refers to the number of epoch and the y-axis refers to the micro-F1 score.

By default, we use co-occurence conditonal probability matrix as co-relation matrix, which serves as the inputs of the stacked GAT for learning the text classifiers. In this part, we evaluate the performance of MATNET under other types of co-relation matrix. Specifically, we investigate three dif-

F1-accuracy					
Methods	**Reuters-21578**	**AAPD**	**Rcv1-v2**	**Slashdot**	**Toxic**
BR	0.878	0.648	0.858	0.486	-
CC	0.879	0.654	0.857	0.480	0.893
CNN	0.863	0.664	0.855	-	0.775
CNN-RNN	0.855	0.664	0.856	-	-
seq2seq-RNN	0.858	0.691	0.868	0.528	-
MAGNET	**0.899**	**0.686**	**0.873**	**0.565**	**0.930**

Table 1: Comparisons of micro F1-score for various models on five benchmark datasets. **BR** : Binary Relevance. **CC** : Classifier chains. **CNN** : Convolutional neural network **CNN-RNN** : Convolutional neural network with Recurrent neural network. **seq2seq-RNN** : sequence to sequence recurrent neural network

ferent correlation matrix, including identity matrix, random matrix, Co-occurence matrix.

Figure (2) shows the Accuracy comparisons with different types of co-relation matrix. An empirical study over these parameters are performed by using the Reuters-21578 dataset with Modified Apte (ModApte) Split. We find that Co-relation matrix perform better than all other co-occurance matrix. This is reasonable because the co-relation matrix contains richer information, leading to the improvement in the performance of the model.

In addition, the proposed models are trained using the Graph attention network which requires predefined co-occurance matrix , which, however, is not provided in any standard multi-label text datasets. In this paper, we build this correlation matrix by mining their co-occurrence patterns within the dataset. This shows that the performance of the model can be improved by using the correlation matrix.

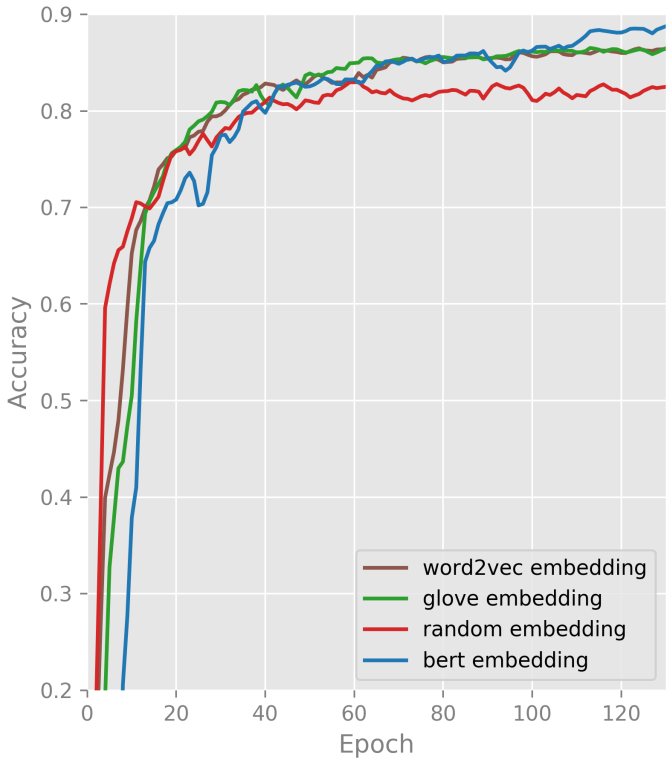

Figure 6: Performance of proposed model on different types of word embeddings. x-axis is the number of epoch and the y-axis refers to the micro-F1 score.

5.2.2 Results on different types of word embeddings

Our proposed default model is using Bert embeddings for encoding the sentences. In this section, we investigate the impact of the four different word embeddings on our proposed architecture, namely the Word2Vec embeddings[16], Glove embeddings [17], Random embeddings, Bert embeddings [6].

Fig. 3 shows the f1 score of all four different word embeddings on the (unseen) test dataset of Reuters-21578.

Accordingly, we make the following observations:

- Bert embeddings outperforms other embeddings in this experiment. Therefore, using Bert feature embeddings increase the accuracy and performance of our architecture.
- Random embeddings perform worse than other embeddings. Pre-trained word embeddings have proven to be highly useful in our proposed architecture compare to the random embeddings.
- Glove and word2vec embeddings producing quite smilier result.

We encode the text using bert embeddings,

Hamming-loss					
Methods	**Rcv1-v2**	**AAPD**	**Reuters-21578**	**Slashdot**	**Toxic**
BR	0.0086	0.0316	0.0032	-	-
CC	0.0087	0.0306	0.0031	-	-
CNN	0.0089	0.0256	0.855	-	-
CNN-RNN	0.0085	0.0278	-	-	-
seq2seq-RNN	0.091	0.0254	-	-	-
MAGNET	**0.0079**	**0.028**	**0.0029**	**0.039**	**0.022**

Table 2: Comparisons of hamming loss for various models on five benchmark datasets. the smaller the value, the better the multi-label classifier is.

which serves as the inputs of the stacked GATs for learning the correlation between the labels data points.

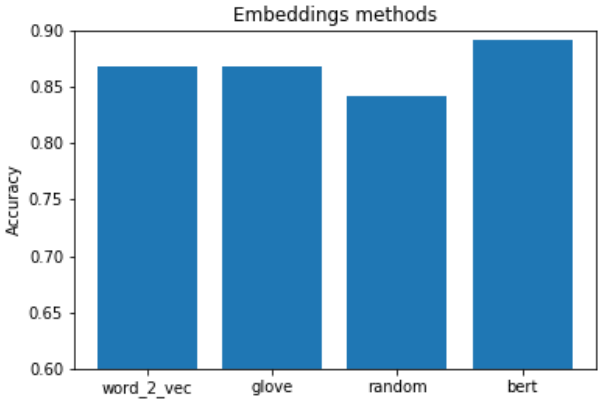

Figure 7: Different types of word embeddings performance on MAGNET x axis refer to the different types of word embeddings and y axis refer to the Accuracy (F1-score)

Fig (2) shows that using different variants of word embeddings on proposed architecture, the accuracy will not be affected significantly. In addition, the observations justify that the accuracy improvements achieved by our method do not absolutely come from the semantic meanings derived from word embeddings. Furthermore, using powerful word embeddings could lead to better performance.

5.2.3 Comparison between Two different Graph neural networks

In this section, We compare the performance of two different Graph neural networks including Graph attention network and Graph convolution network.

We first use a graph attention network (GAT)[14] which incorporates the attention mechanism into the propagation step. It computes the hidden states of each node by attending over its neighbors, following a self-attention strategy. Then we tried Graph convolution network. The key difference between GAT and GCN is how the information from the one-hop neighborhood is aggregated.

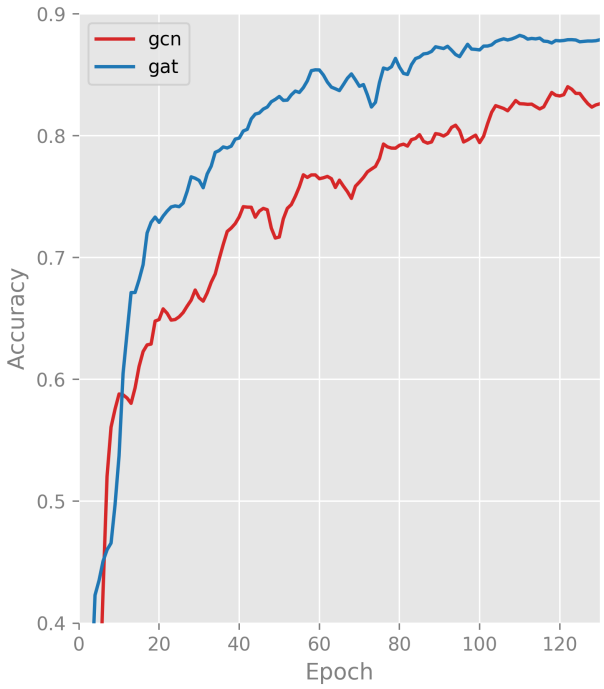

Figure 8: Performance of proposed model on different types of graph neural network. The x-axis refers to the number of epoch and the y-axis refers to the micro-F1 score.

In our case graph attention model performed better than graph convolution model. Fig 5 shows the accuracy of both neural network on Reuters-21578 dataset.

Both graph neural network takes feature description $\boldsymbol{M} \in \mathbb{R}^{n \times d}$ and the corresponding correlation matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ as input. Graph attention network is a spatial-based graph convolution network where the attention mechanism is involved in determining the weights of a nodes neighbors when aggregating feature information. Here feature information is correlation matrix and input is encoded label2vec.

GAT produced an average 4% miF1 score increase over the GCN model. It shows that the Graph attention network (GAT) model outperforms the vanilla GCN in supervised learning of various labels properties. The attention mechanism can identify label importance in dependency graph by considering the importance of their neighbor labels.

5.2.4 More layers More accuracy?

We investigate the performance of proposed model with different numbers of Graph attention layers.

We observed that after adding more than two layers the performance drops on dataset. By default we use two graph attention layers for all the datasets. Input for graph attention layer is label2vec and correlation matrix. It aggregating feature information from neighbourhood nodes basis on the attention mechanism. Fig(8) shows the impact of using different no of layers on proposed architecture.

5.2.5 Freezing vs fine turning the word embedding layer

Our next series of experiments examines the impact of freezing the word embedding layer vs training the word embedding layer.

We performed two experiments, in first experiment we freeze the embedding layer of model since the embedding is already learned, we want to examine if the model imporves while keeping the word embeddings fixed we tried the pertained weight of embeddings. Using pre-trained embedding prevent the weights from being updated during training so gradients wont propagate backwards past the frozen layer.

In second experiment, we train and fine tune the embedding vectors for domain specific task. Fine-tuning is one of the methods for transfer learning (Pan and Yang, 2009). There are also much work about multi-task learning, which is another approach often used for transfer learning for neural networks (Aguilar et al., 2017) (von Daniken and Cieliebak , 2017)

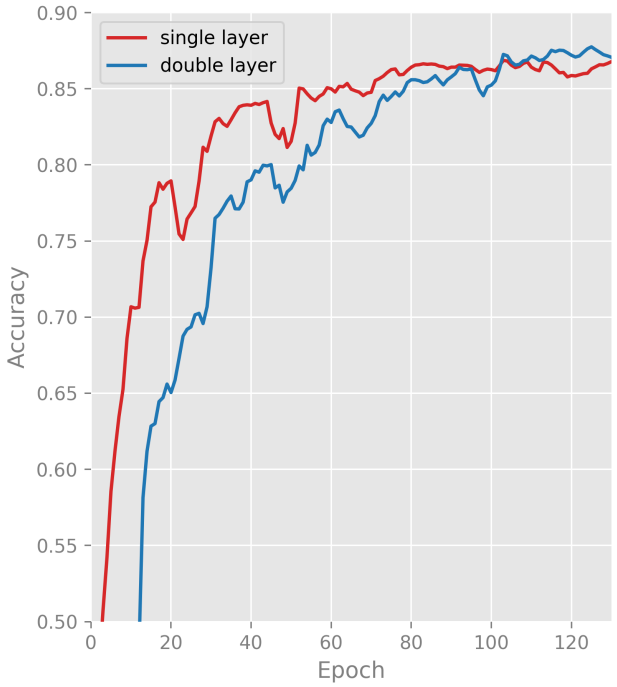

Figure 9: Performance of MAGNET model on different types of graph neural network. The x-axis refers to the number of epoch and the y-axis refers to the micro-F1 score.

In our case freezing the embedding layer performed better than fine tuning the embedding layer Fig (9) shows the impact of different embedding layer on both experiment including training and fine tuning the embedding layer. Accordingly, keeping the original weights of pre-trained embedding leads to the better performance on proposed architecture.

5.3 Conclusion

We proposed a deep learning model for multi-label classification, which consists of a graph attention network and a bidirectional LSTM. recently there has been a growing interest in the utilisation of graph convolutional networks (GCNs) and Graph attention networks(GATs) due to the fact they have been shown to provide great improvements in many tasks such a network embedding.

In order to model and capture the dependency between labels, we proposed a GAT based model.

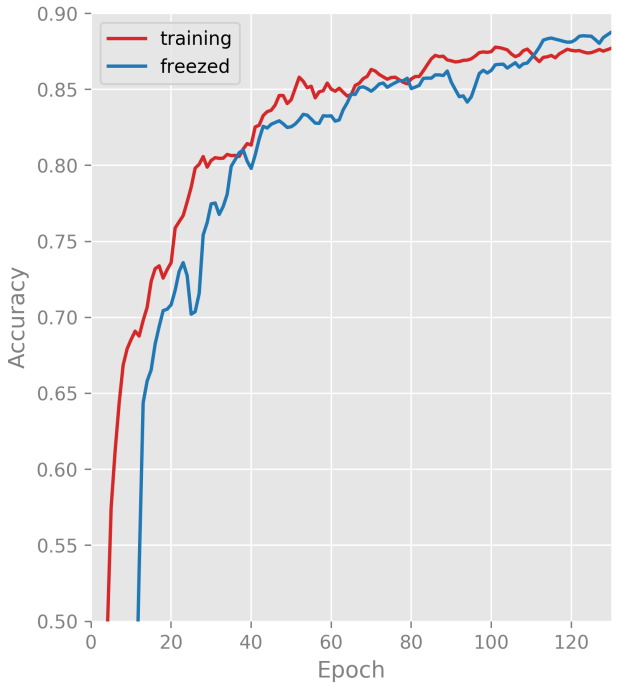

Figure 10: Different no of layer's performance on MAGNET. The x-axis refers to the number of epoch and the y-axis refers to the micro-F1 score.

To explicitly model the label dependencies, we designed a novel re-weighted scheme to construct the correlation matrix and use the attention mechanism for GAT by balancing the weights and between a node and its neighborhood for node feature update. In our experiments, we provided quantitative results to support the effectiveness of our method.

Our evaluation on datasets obtained from the Reuters-21578 (90 classes, 11K records), the RCV117 (103 classes, 800K records), Arxiv Academic Paper Dataset (AAPD), Slashdot (Read et al., 2011), toxic comment dataset shows that combinations of Graph attention network with bi-directional LSTM, has consistently higher accuracy than those obtained by conventional approaches. These results show that deep learning methods can provide improvements for multi label classification and that they provide flexibility to classify datasets by capturing the dependencies between labels.

Even our proposed model perform very well but still there are some limitations, when a dataset contains large number of labels , how to predict all these true labels accurately is an intractable problem and training the graph over this huge labels matrix is hard. Our proposed methods alleviate this problem to some extent, but more effective solutions need to be further explored in the future.

The proposed approach has the ability to improve accuracy and efficiency of models and can be use across a wide range of data types and applications.

REFERENCES

[1] Wei Bi and James Kwok. *Multilabel Classification with Label Correlations and Missing Labels*. 2014. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8385.

[2] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. "A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications". In: *CoRR* abs/1709.07604 (2017). arXiv: 1709.07604. URL: http://arxiv.org/abs/1709.07604.

[3] Everton Alvares Cherman, Maria Carolina Monard, and Jean Metz. "Multi-label Problem Transformation Methods: a Case Study". In: *CLEI Electron. J.* 14.1 (2011). DOI: 10.19153/cleiej.14.1.4. URL: https://doi.org/10.19153/cleiej.14.1.4.

[4] Franca Debole and Fabrizio Sebastiani. "An Analysis of the Relative Hardness of Reuters-21578 Subsets: Research Articles". In: *J. Am. Soc. Inf. Sci. Technol.* 56.6 (Apr. 2005), pp. 584–596. ISSN: 1532-2882. DOI: 10.1002/asi.v56:6. URL: http://dx.doi.org/10.1002/asi.v56:6.

[5] Sébastien Destercke. "Multilabel Prediction with Probability Sets: The Hamming Loss Case". In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems - 15th International Conference, IPMU 2014, Montpellier, France, July 15-19, 2014, Proceedings, Part II*. 2014, pp. 496–505. DOI: 10.1007/978-3-319-08855-6_50. URL: https://doi.org/10.1007/978-3-319-08855-6%5C_50.

[6] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA,*

June 2-7, 2019, Volume 1 (Long and Short Papers). 2019, pp. 4171–4186. URL: https://aclweb.org/anthology/papers/N/N19/N19-1423/.

[7] Alex Fout et al. "Protein Interface Prediction Using Graph Convolutional Networks". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6533–6542. ISBN: 978-1-5108-6096-4. URL: http://dl.acm.org/citation.cfm?id=3295222.3295399.

[8] Siddharth Gopal and Yiming Yang. "Multilabel classification with meta-level features". In: 2010, pp. 315–322. DOI: 10.1145/1835449.1835503. URL: https://doi.org/10.1145/1835449.1835503.

[9] Sepp Hochreiter and Jfffdfffdrgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[10] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. "Multilabel Text Classification for Automated Tag Suggestion". In: *Proceedings of the ECML/PKDD 2008 Discovery Challenge*. Antwerp, Belgium, 2008.

[11] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1746–1751. URL: http://aclweb.org/anthology/D/D14/D14-1181.pdf.

[12] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *CoRR* abs/1609.02907 (2016). arXiv: 1609.02907. URL: http://arxiv.org/abs/1609.02907.

[13] David D. Lewis et al. "RCV1: A New Benchmark Collection for Text Categorization Research". In: *J. Mach. Learn. Res.* 5 (Dec. 2004), pp. 361–397. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=1005332.1005345.

[14] Junyang Lin et al. "Semantic-Unit-Based Dilated Convolution for Multi-Label Text Classification". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. 2018, pp. 4554–4564. URL: https://aclanthology.info/papers/D18-1485/d18-1485.

[15] Oscar Luaces et al. "Binary relevance efficacy for multilabel classification". In: *Progress in Artificial Intelligence* 1.4 (Dec. 2012), pp. 303–313. ISSN: 2192-6360. DOI: 10.1007/s13748-012-0030-x. URL: https://doi.org/10.1007/s13748-012-0030-x.

[16] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*. 2013, pp. 3111–3119. URL: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.

[17] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*. 2014, pp. 1532–1543. URL: http://aclweb.org/anthology/D/D14/D14-1162.pdf.

[18] Jesse Read et al. "Classifier chains for multi-label classification". In: *Machine Learning* 85.3 (2011), pp. 333–359. DOI: 10.1007/s10994-011-5256-5. URL: https://doi.org/10.1007/s10994-011-5256-5.

[19] A. J. Robinson and Frank Fallside. *The Utility Driven Dynamic Error Propagation Network*. Tech. rep. CUED/F-INFENG/TR.1. Cambridge, UK: Engineering Department, Cambridge University, 1987.

[20] Robert E. Schapire and Yoram Singer. "BoosTexter: A Boosting-based System for Text Categorization". In: *Machine Learning* 39.2/3 (2000), pp. 135–168. DOI: 10.1023/A:1007649029923. URL: https://doi.org/10.1023/A:1007649029923.

[21] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. “Effective and Efficient Multilabel Classification in Domains with Large Number of Labels”. In: *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD’08)*. Antwerp, Belgium, 2008.

[22] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[23] Petar Velickovic et al. “Graph Attention Networks”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018. URL: https://openreview.net/forum?id=rJXMpikCZ.

[24] Paul J. Werbos. “Generalization of backpropagation with application to a recurrent gas market model”. In: *Neural Networks* 1.4 (1988), pp. 339–356. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(88)90007-X. URL: http://www.sciencedirect.com/science/article/pii/089360808890007X.

[25] Ronald J. Williams and David Zipser. *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*. 1989.

[26] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks”. In: *CoRR* abs/1901.00596 (2019). arXiv: 1901.00596. URL: http://arxiv.org/abs/1901.00596.

[27] Rex Ying et al. “Graph Convolutional Neural Networks for Web-Scale Recommender Systems”. In: *CoRR* abs/1806.01973 (2018). arXiv: 1806.01973. URL: http://arxiv.org/abs/1806.01973.

[28] Min-Ling Zhang et al. “Binary relevance for multi-label learning: an overview”. In: *Frontiers of Computer Science* 12.2 (Apr. 2018), pp. 191–202. ISSN: 2095-2236. DOI: 10.1007/s11704-017-7031-7. URL: https://doi.org/10.1007/s11704-017-7031-7.