Deep Learning Approaches to the Multi-Instance Multi-Label (MIML) Learning Problem

by

Aviral Kumar
(140070031)

under the guidance of

Prof. Ganesh Ramakrishnan

and jointly supervised by

Prof. Purushottam Kar, IIT Kanpur
Dr. Prateek Jain, Microsoft Research India

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

April, 2017

Abstract

Multi-instance Multi-label learning (MIML) problem is that, given a bag of instances and a set of labels, the task is to assign labels to the bag, which are relevant to the bag as a whole. The problem of MIML finds its relevance in relation extraction, vision, machine learning and information extraction. In this piece of work, our aim is to develop a scalable deep learning approach towards the MIML problem, with applications in document information retrieval and document labeling. In the process, we also wish to directly optimize non-decomposable performance measures like F-measure, average precision, prec@k which are directly used for evaluation of the approach. The approach proposed by us not only achieves these goals but also, as a side application gives embeddings for paragraphs in a document which can be used as paragraph2vec vectors analogous to word2vec vectors for words.

Acknowledgment

This report is an outcome of my work under the guidance of Prof. Ganesh Ramakrishnan on: **Deep Learning Approaches to the Multi Instance Multi Label Learning Problem**. I would like to thank him for his guidance, motivation and constant support throughout the duration of the project. I would also like to thank my co-advisors, Prof. Purushottam Kar (IIT Kanpur) and Dr. Prateek Jain (MSR India) for their brilliant ideas and guidance on the project. My research in the MIML problem is being driven by their support.

I would also like to thank my colleagues working on this project: Ankith MS, Suhit Sinha and Suvadeep Hajra for healthy and useful discussions and all brainstorming on the problems.

Contents

Chapter 1

Introduction

Conventional supervised learning methods of classification assume that an object occurs as a single instance and is associated with a single label. There are many real-world problems which do not fit this framework well, where a real-world object may be associated with a number of instances and a number of labels simultaneously. For example, an image usually contains multiple patches each can be represented by an instance, while in image classification such an image can belong to several classes simultaneously, e.g. an image can belong to mountains as well as India. Another example is text categorization, where a document usually contains multiple sections each of which can be represented as an instance, and the document can be regarded as belonging to different categories if it was viewed from different aspects, e.g. a document can be categorized as scientific novel or a magazine. Web mining is a further example, where each of the links can be regarded as an instance while the web page itself can be recognized as news page, sports page, soccer page, etc.

Apart from this, the percentage of labeled data present around us is not too high. This has led to the development of interest in methods which rely on distant or weak supervision. The paradigm of Multi Instance Multi Label (MIML) learning has proved to be a good way of modeling situations under distant supervision and proves to be a good model in the above mentioned examples.

1.1 Multi Instance Multi Label Learning Problem

In traditional supervised learning, an object is represented by an instance (or feature vector) and associated with a class label. Formally, let X denote the instance space (or feature space) and Y the set of class labels. Then the task is to learn a function $f : X \to Y$ from a given data set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)\}$, where $x_i \in X$ is an instance and $y_i \in Y$ the known label of x_i.

MIML learning problem deals with data objects represented by a bag of instances and labeled jointly by a set of labels. The task is to learn to label the bag of instances with labels which are relevant to the bag as a whole.

We now formalize the MIML framework. A training example is described by multiple instances and associated with multiple class labels. Formally, let X denote the instance space and Y the set of class labels. Then the task is to learn a function $f_{MIML} : 2^X \rightarrow 2^Y$ from a given data set $\{(X_1, Y_1), (X_2, Y_2), \cdots, (X_m, Y_m)\}$, where $X_i \in X$ is a set of instances $\{x_1^{(i)}, x_2^{(i)}, ..., x_{n_i}^{(i)}\}, x_j^{(i)} \in X(j = 1, 2, ..., n_i)$, and $Y_i \in Y$ is a set of labels $\{y_1^{(i)}, y_2^{(i)}, ..., y_{l_i}^{(i)}\}, y_k^{(i)} \in Y(k = 1, 2, , l_i)$. Here n_i denotes the number of instances in X_i and l_i the number of labels in Y_i.

Example

Consider a bag of sentences about Mickey Mouse and Disney. Here each sentence is an instance. On the other side consider a set of labels which are the relations which relate the entities *(Mickey Mouse, Disney)*. It can be seen that one sentence (instance) expresses just 1 relation (or 1 label) (for example, mascot) and the bag expresses the set of labels/relations expressed by all the sentences in the bag, as these labels/relations relate the entities in some way or the other. For example, relations expressed are:

In this report the notation $r(e_1, e_2)$ will be used to denote a relation tuple where r is the relation name and e_1, e_2 are the related entities. For example, *Mascot(Mickey Mouse, Disney)*

1.2 Current Approaches for MIML Problem

Most of the current approaches in the MIML problem include conversion of Multi Instance to Single Instance problems, thereby getting a multi label classification problem which has now been researched upon for ages. The first few works of Zheng et.al. [11] on Multi Instance Multi Label learning framework used the strategy mentioned above. [Li et al] modelled how instances trigger some labels and others don't by considering shared patterns across relevant labels. Briggs et al. (2012) [2] proposed rank-loss support instance machines for MIML instance annotation. Briggs et al. (2013) [3] considered the problem of predicting instance labels while learning from data labeled only at the bag level by using a new regularized rank-loss objective. Huang et al. (2014) [5] proposed a fast MIML algorithm by exploiting label relations with shared space and discovering sub-concepts for complicated labels. Pham et al.(2015) used a discriminative probabilistic model to discover novel classes in MIML learning.

[Zeng et al. 2015] applied Piecewise Convolutional Neural Networks (PCNNs) to the MIML problem. These models were trained by optimizing performance measures like conditional log-

likelihood (Surdeanu et al. 2012), error rate, or bag level entropy (Zeng et al. 2015) that are not directly related to the measures actually used for evaluation such as F-measure and average precision.

The area of learning structured prediction models is also quite relevant in the field of MIML as the task in MIML to predict a structured array of labels. (respecting the order and the heirarchy among labels) Works like Structural SVMs and graphical models were applied to optimize non-decomposable performance measures in MIML by [Haffari et.al.] But in general, all these approaches were not highly scalable.

Deep learning approaches into the problem are quite recent. Some of the relevant papers include [10] where the label and the instance embeddings were jointly learned in case of multi-label image classification, Deep-MIML Network [4] where the authors introduce a 2D concept layer, which is used to map instances to labels and the entire bag is mapped to its labels using a 3D version of this layer.

1.3 Problem Statement

We are interested in achieving two primary goals. Firstly, we want to develop a scalable algorithm for Multi Instance Multi Label learning problem, especially focusing on areas of distant supervision such as wikipedia data. Secondly, we want to be able to learn to make instance level predictions along with bag level predictions. In most of the existing techniques, model parameters are learned by optimizing performance measures (e.g.: conditional log-likelihood, error rate) on each training samples. However, these are not directly related to evaluation measures, and the loss function such as F_β cannot be decomposed into a linear combination of loss functions over individual training samples. So, our third goal is to explore how to optimize non decomposable performance measures like F_β scores, *prec@k* loss, etc.

1.4 Our Work

In this report, we first discuss the initial few weeks of work done by our team, where we were trying to come up with a better initialization strategy for an existing algorithm $MIML^{perf}$ [1] proposed by [Kar et al, AAAI 2017]. Our goal was to use approaches like Positive Unlabeled learning in order to get faster convergence for the algorithm. In the remaining part of this report, we discuss the development of a proposal for our new deep learning model, we will start to experiment with now. We first discuss the motivation for a deep learning approach, and then move on to two models, which we propose and expect to work on in future. We also introduce a new way of integrating the prec@k loss into the deep learning setting, and it is called Deeptron@k.

Chapter 2

Optimizing Non Decomposable Performance Measures in MIML

In this chapter we present one of the recent approaches to MIML learning problem under Distant Supervision where the aim is to optimize non-decomposable performance measures along with providing a scalable solution to the problem.

The algorithm, $MIML^{perf}$ [1] tries to optimize performance measures like F-macro and F-micro. It can predict rare labels correctly. It is streaming in nature and so it does not need the ability to store the entire dataset and can run by making several passes over the data. The authors have shown that $MIML^{perf}$ 1) offers greater label extraction accuracies on RE than specialized methods for the problem, 2) it also outperforms state-of-the-art MIML approaches on text categorization and scene classification problems, and 3) it can offer orders of magnitude faster running times.

2.1 Problem Formulation

As mentioned above, this approach deals with predicting the instance wise labels and then combining the set of labels of the bag using a simple disjunction to produce the set of labels for the entire bag.

There are some differences in the notations for this chapter. The dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$ where $x_i = \{x_i^{(1)}, \cdots, x_i^{(n_i)}\}$ is the i^{th} bag containing n_i instances and $y_i = [y_{i,1}, y_{i,2}, \cdots, y_{i,L}] \in Y = 0, 1^L$ is a vector of labels associated with x_i. The labels of each instance are represented as $h_i \in \{1, .., L, nil\}^{n_i}$ to denote the vector of hidden labels for x_i. Again analogous to $y_{i,j}$, $h_{i,j}$ encodes if the j^{th} instance in the bag x_i expresses one of the labels $\{1, \cdots, L\}$ or not $\{nil\}$.

A function $f : \mathcal{X} \to \mathcal{Y}$ needs to be learned which predicts labels for an instance $x \in \mathcal{X}$ as a one-hot vector over the labels. The optimization measure is a function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. The optimization measures used in the paper are as follows: (Label Wise precision and recall)

$$PREC^j(f; X, Y) := \frac{\sum_{i=1}^{n} y_{i,j} \cdot f^j(x_i)}{\sum_{i=1}^{n} f^j(x_i)}$$

$$REC^j(f; X, Y) := \frac{\sum_{i=1}^{n} y_{i,j} \cdot f^j(x_i)}{\sum_{i=1}^{n} y_{i,j}}$$

and the combined precision and recall

$$PREC(f; X, Y) := \frac{\sum_j \sum_{i=1}^{n} y_{i,j} \cdot f^j(x_i)}{\sum_j \sum_{i=1}^{n} f^j(x_i)}, REC(f; X, Y) := \frac{\sum_j \sum_{i=1}^{n} y_{i,j} \cdot f^j(x_i)}{\sum_j \sum_{i=1}^{n} y_{i,j}}$$

Then using these 4 quantities F measures have been defined:

$$F_\beta^{macro}(f; X, Y) := \frac{1}{N} \sum_{1}^{L} F_\beta^j(f; X, Y)$$

$$F_\beta^{micro}(f; X, Y) := \Big(\frac{\beta}{PREC(f; X, Y)} + \frac{1 - \beta}{REC(f; X, Y)} \Big)^{-1}$$

2.2 MIML-perf Framework

2.2.1 Plug In Classifiers

The model described in the paper deals with plug-in classifiers which incur the main benefit that the same plug in classifier can be tuned according to various performance measures. Plug in classifiers allot each instance a Class Probability Estimate Score (CPE) score which is given by a function $g : \mathcal{X} \to \mathbb{R}$ such that for a binary classification setting, $g(x) \approx P[x = 1]$. Then classification is done based on $sign(g(x) - \eta)$, where η is a threshold parameter.

2.2.2 Training Algorithm

The training algorithm for $MIML^{perf}$ is a EM-like algorithm. $MIML^{perf}$ reduces the task of predicting labels for the entire bag as the task of predicting labels for each of the individual instances and then combining the labels at the end. In this approach, as the training data is available only at the bag level, the instance level labels are estimated using latent variables. At every time step the algorithm makes an estimate of the latent variables which are used as instance level supervision data, then trains the plug in classifiers (one for each label) based on these estimates, tunes the thresholds and then uses this to update the estimate of the hidden variables.

Algorithm 1 MIML^{perf}: Training Routine

Input: Data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, expression rate κ, perf. measure Δ

For all (i, j) such that $y_{i,j} = 1$, set random $\kappa \cdot n_i$ entries of the vector $\mathbf{z}^{(i,j)}$ to 1 *// Initialize hidden labels randomly*

while not converged **do**

Step 1: Fix hidden variables, update plug-in classifiers

  for every label $j \in [L]$ **do**

    $\mathcal{D}^j \leftarrow \{\{(\mathbf{x}_i^{(k)}, z_k^{(i,j)})\}_{k=1}^{n_i}\}_{i=1}^{n}$ *// Prepare datasets*

    $g^j \leftarrow$ CPE-train$(\mathcal{D}^j)$ *// Train CPE models*

  end for

  $\boldsymbol{\eta} \leftarrow$ Tune-thresholds$((\mathbf{X}, \mathbf{Y}), \mathbf{p}; \Delta)$ *// Optimize* Δ

Step 2: Fix plug-in classifiers, update hidden variables

  for $(i, j) \in [N] \times [L]$ such that $y_{i,j} = 1$ **do**

    $\mathbf{z}^{(i,j)} \leftarrow \mathbf{0}^{n_i}$ *// Reset hidden labels*

    $c^{(i,j)} \leftarrow \sum_{k=1}^{n_i} \mathbb{I}\{g^j(\mathbf{x}_i^{(k)}) \geq \eta_j\}$

    $S^{(i,j)} \leftarrow$ Sample $c^{(i,j)}$ entries of $\mathbf{z}^{(i,j)}$ according to g^j

    $z_k^{(i,j)} \leftarrow 1$ for all $k \in S^{(i,j)}$ *// Reestimate hidden labels*

  end for

end while

Figure 2.1: *$MIML^{perf}$ training algorithm*

Hidden Variables

For a set of N data points $\{(xi, yi)\}_{i=1}^{N}$, the binary variable $z_{(i,j)}^{k} \in 0, 1$ indicates whether the $k-th$ instance in the $i-th$ bag expresses the $j-th$ label or not. The $i-th$ data point has n_i instances and the vector $z_{(i,j)} \in \{0, 1\}^{n_i}$ encodes which of the instances in the bag express the label j (one-hot representation). If a bag doesn't have a label then no instance can express it. As most bags in case of distant supervision have less number of labels, therefore, the data in this case is sparse, and this is one of reasons behind the scalability of the approach.

Step 1: Training the plug-in classifiers

In the first step of the algorithm the latent variables $z^(k)_{(i,j)}$ are estimated and the plug in classifiers are trained using these values. The thresholds are tuned based on the CPE scores.

Step 2: Re-estimating the latent variables

The CPE models trained in Step 1 are used to generate scores for each of the instances. These scores are quite noisy because of random initializations of the latent variables. But still high scores are to some extent indicative of the presence of the label in the bag. So, they sample new values of hidden/latent variables partially from the last estimated CPE scores and partially at random.(to allow for reaching to the global minimum in case of a local minimum)

Algorithm 2 MIML^{perf}: Testing Routine

Input: Test point $\mathbf{x} = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(n_t)}\}$ with n_t instances, CPE models $g^1, \ldots, g^L$, thresholds $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_L]$.

1: **for** $k = 1, 2, \ldots, n_t$ **do**
2: $\quad h_k \leftarrow \{j : g^j(\mathbf{x}^{(k)}) \geq \eta_j\}$ *// Discover hidden labels*
3: **end for**
4: **for** $j = 1, 2, \ldots, L$ **do**
5: $\quad \hat{y}_j \leftarrow \bigvee_{k=1}^{n_t} \mathbb{I}\{j \in h_k\}$ *// Aggregation step*
6: **end for**
7: **return** $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_L]$

Figure 2.2: *$MIML^{perf}$ training algorithm*

2.2.3 Testing Algorithm

The testing algorithm for $MIML^{perf}$ simply uses the training routine to estimate the hidden variables and then simply combines the labels expressed by each instance to generate the labels expressed by the bag.

2.3 Results and Improvements

The $MIML^{perf}$ algorithm although scales well and gave good results, better than the existing approaches at the time, both at the F^{macro} level and at the F^{micro} level and also good performance on rare labels.

We believe that the random initialization component of the algorithm might be a contributing factor towards its slower convergence and if we could do better in the initialization part, then the overall algorithm would work great. In this pursuit, we explored another training paradigm, **Positive Unlabeled Learning** which was to be used as an initializer for the $MIML^{perf}$ algorithm.

2.4 Positive Unlabeled Learning

The main aim of positive unlabeled learning is to solve the following problem: Given a set data points or objects of class P and a large set of mixed objects that contains objects from class P and other types of objects $\neg P$, label the objects as P or $\neg P$. The main crux of the situation is that a reasonably well model needs to be trained so that it is able to classify negative instances as well, although it has never seen a negative instance during training. This makes the use of conventional methods for classification not that helpful here.

2.4.1 Algorithm for PU learning

Bing Liu et.al. [8] [9] proposed an algorithm for generalised PU learning for binary classification. Let the set of positively labeled instances be called P, and the remaining set $Q = N - P$ is the set of yet unlabeled instances. As $|P|$ is considerably less than $|Q|$ (this is the assumption made), classifiers trained based on assuming the entire set Q as negative and then classifying are quite bad in this case. So, the idea is to identify a set of "reliably negative" instances, and then learn a classifier depending on the set of positive and reliably negative instances.

Step 1: Identifying the reliably negative examples:

In this step, the ”spy technique” is used. What is done is that, some fraction of the instances from P, say (s%) are labeled a spy and are sent to the unlabeled set Q. So, now the training model knows some positively labeled examples from the unlabeled set Q too.

An initial EM algorithm is applied in order to learn a Naive-Bayes classifier for separating the positive and negative components in the set $Q \cup S$, where S is the spy set. As, it is already known that S contains examples which are positively labeled, thresholds can be generated from the spy set, which is given by $\min_{x_i \in S} P[x_i \in P|x_i]$ across all the members of the spy set and then use this value as a threshold to partition unlabeled(U) into unlabeled(U') and negative class(N) instances. The choice of the negative instance class N, is such that it can be relied upon with high belief that the examples belonging to this assumed negative class are reliably negative.

Step 2: Learning a classifier:

At the end of step 1, we end up having three sets: positive(P), reliably negative (N) and unlabeled (U'). For the first two sets, $P[+|instance] = 1$ and $P[-|instance] = 1$ respectively. Now, the task is to learn to classify the unlabeled set(U'). So, again an EM algorithm is applied in order to learn a Naive Bayes classifier for this setting, with a modification that for the + and - examples, the probability values don’t change. We end up with a good classifier separating the positive and negative classes at the end of this step, and use it to classify the set U. The algorithm is summarized in the figure.

2.5 Our Work on Initialization Techniques

In the pursuit of good initialization techniques which could prove to be a good starting point for the EM based algorithm, we tried some techniques inspired by ideas from the PU learning domain. These are listed below.

Algorithm 1 Positive Unlabeled Learning: Identifying Reliably Negative Examples

procedure IDENTIFY-RELIABLY-NEGATIVE-EXAMPLES()
  $N = U = \phi$
  Sample S = $sample(P, s\%)$
  $MS = M \cup S$
  $P = P - S$
  Assign every object $d_i \in P$, a label $+$ and every $d_i \in N$, label $-$
  Build an initial Naive Bayes classifier (NBC) using sets MS and P
  while classifiers parameter change **do**
    Compute $P[c_1|d_j]usingcurrentNBC$
    Update $P[w_t|c_1]$ and $P[c_1]$ given the probabilistically assigned class for d_j ($P[c_1|d_j]$)
  Classify each document in MS;
  Determine the probability threshold t using S
  for each document d_j in M **do**
    if $P[c_1|d_j] < t$
      $N = N \cup \{d_j\}$
    else $U = U \cup \{d_j\}$

2.5.1 MIML posed as Negative Unlabeled Learning Problem

The MIML problem can be posed as a negative-unlabeled(NU) learning problem, which strives to learn good classifiers given initial data is negatively labeled and the remaining data is unlabeled. Given the bag level data, it is easy to generate the instance level supervision for a NU-learning problem because the labels not expressed by a bag are surely not expressed by any instance in the bag. Hence, we know correct instance to label mappings which are not expressed, or which belong to the negative class. In this manner, we could get a good starting point for the EM algorithm.

Disadvantages of this approach:

- Due to the sparsity of label occurrences in the MIML dataset, the number of negatively labeled data points that can be generated from the dataset is too large, and hence it does not exactly qualify towards the NU learning approach based on the theory of PU learning.
- Moreover, most of the MIML Relational Extraction Data like, Wikipedia dataset, exhibits noisy labels and a lot of labels are missing. By this I mean that the document training data is not labeled with some of the labels(relations) which are expressed by the instances(paragraphs). Learning latent variables by using NU learning might hence give a very poor estimate of the hidden variables and might affect the main algorithm adverserially.

Chapter 3

Deep Learning Approach to MIML

With the current surge of neural networks and the growing popularity of deep learning approaches in today's machine learning research community, researchers round the world have started exploring plausible deep learning solutions to the multi instance multi label problem. This is also the area which we are currently working in to try and find a good scalable approach to MIML using deep neural net (DNN) architectures.

Motivation for Deep Learning

Our preliminary investigation into deep learning approaches was a sequence-to-sequence encoder-decoder model where simply supplied the instances of a bag one-by-one into the LSTM/RNN cell and at the end started predicting the labels one by one through the LSTM cells of the decoder. This model is as sample as it can be. Using this model, with 50 epochs of training on a small dataset we were able to achieve F^{macro} scores close to 87% [On Reuters Dataset] which is quite close to the optimal results obtained by $MIML^{perf}$.

Table 3.1: *Comparison of F-Micro Scores in $MIML^{perf}$ and a plain simple RNN*

Dataset	$MIML^{perf}$	RNN
Reuters	0.89	0.87
Scene	0.62	0.51

3.1 Related Work

In recent years, deep learning has been extensively used in learning representations from raw data. In particular, deep covolutional neural networks such as (Krizhevsky, Sutskever, and Hinton 2012; Simonyan and Zisserman 2014) have been superior in their performance on image classification tasks. LSTM (Hochreiter and Schmidhuber 1997) based models have been successfully applied to text data; for instance, a pre-trained skip-thought model (Kiros et al. 2015; Zhu et al. 2015) can be used as an encoder to produce high quality sentence representations. People have worked on incorporating relationships between instances and labels. Deep MIML Network, and other deep learning paradigms related to representation learning. Quite a few works on the multi label problems such as [Wei et al, 2014, Wang et al 2016, Zeng et al 2015] were quite specific to some applications and were not necessarily in the exact same setup as the MIML problem formulation. Some papers like Deep MIML Network [4] which have focused on a wider class of the MIML problem, have used methods like representation learning, and then instance label relation discovery. On the other hand, papers like CNN-RNN [10] present a slightly different approach which falls under the paradigm of metric learning, which is to learn representations jointly in order to provide enough correlation and meaning connecting the label embeddings and the instance embeddings. People have also applied simple paradigms like generating the representations of each of the instances and then cross-sentence pooling over it to get the compact representation of the bag and then using this layer to predict the labels. One of these works [RE -MIML-CNN] has used such an approach.

3.2 Our Approach

In this section, I will describe the two major approaches proposed by us and their pros and cons. We are currently working on implementing both of these to figure out eventually which one of them is better.

3.2.1 Problem Formulation

Consider a document to be represented by the notation D. The paragraphs of a document are represented by the vector P_i (represents the i^{th} paragraph of a document. Hence, the overall document can be represented as $D(P_1, P_2, ..., P_N)$, assuming N to be the total number of paragraphs in the document. We intend to generate the following outputs from the data: (1) **paragraph2vec vectors** (analogous to word2vec vectors): we are interested in generating embeddings for paragraphs in a document which correspond to similarity in paragraphs in some space. (2) instance level labellings (and then getting bag level labellings by combining them.

3.2.2 Encoder-Decoder like approach

To reiterate, consider a document to be represented by the notation D. The paragraphs of a document are represented by the vector P_i (represents the i^{th} paragraph of a document. Hence, the overall document can be represented as $D(P_1, P_2, ..., P_N)$, assuming N to be the total number of paragraphs in the document.

3.2.3 Encoder Model

Now, in order to build the model based on the intuition described in the section above, we need to get embeddings at both the paragraph and document level, which can be parametrized by using a CNN (as done commonly in the metric learning papers, where learning a good metric is achieved through a CNN, and also drawing inspirations from handing images using CNNs, where a compact feature representation of an image is generated using a convolutional network).

- **Paragraph-Level Embeddings:** So, N CNNs (labeled CNN_{PL}), numbered 1 to N with shared parameters are applied to the set of N paragraphs in the document, in a one-to-one mapping (i^{th} CNN producing an embedding of the i^{th} paragraph). Let us call these embeddings: $f(P_i) \in \mathcal{R}^d, i \in [1, N]$, where d is a tunable hyper-parameter.
- **Document-Level Embeddings:** The output embeddings of these N CNN , as described in the previous point are applied as input to a new CNN (labeled CNN_{DL}). The input to CNN_{DL} is a $N.d$ vector $f(P_1).f(P_2)....f(P_n)$. The output of this CNN is an embedding of the whole document, which summarizes its properties to some extent, and acts like a suitable document-level representation of the entire document. We call this $g(D) = g(f(P_1), f(P_2), ..., f(P_N)) \in \mathcal{R}^m$, where m is again a tunable hyper-parameter. A good candidate for pooling in this network is average pooling, so that we propagate a summary of the entire document to the network ahead. [The inspiration of this idea comes from the following paper: Show Attend and Tell, Bengio et al, 2016]

3.2.4 Decoder Model

The network until this point performs the job of an encoder, and now using some mechanism we need to output labels one by one, using an attention mechanism. The idea is based on the intuition that now by learning a suitable transformation from the document-level representation to the paragraph-level space, we can focus on a few important paragraphs and then by means of a FC layer, produce/output the label for that particular set, and parallely, update the document-level representation to now remove the component that corresponds to this label output, and hence focus on the remaining labels from the next time step.

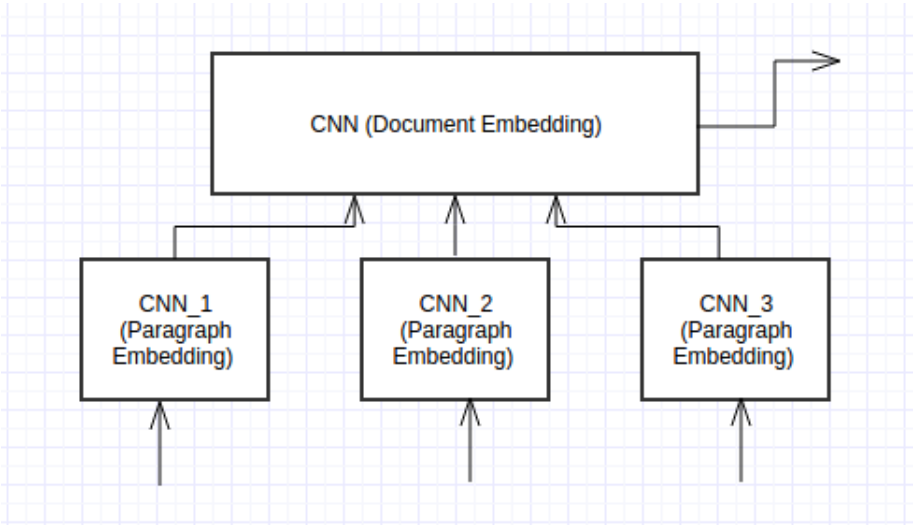

Figure 3.1: *Encoder Module for the model*

- **Attention Model:** We propose two kinds of attention models in this setting. These are described as follows:
  - **Attention Mechanism 1:** This model is one of the most popular attention models used in literature. The attention module generates a context vector $\overline{z}_c \in \mathcal{R}^d$, which is given by the following:
    $$\overline{z}_c(t) = \sum_{i \in [1,N]} \alpha_i(h_t) f(P_i)$$
    , where $\alpha_i(x) = \frac{exp(\beta x^T \mathbf{W} f(P_i))}{\sum_i exp(\beta x^T \mathbf{W} f(P_i))}$. Here the weight/ transformation matrix $\mathbf{W}$ is learned and $x^T \mathbf{W} f(P_i)$ symbolizes and denotes the cosine similarity between the transformation of x^T into the paragraph embedding space and the paragraph embedding of the i^{th} paragraph, β is a sharpening factor.
  - **Attention Mechanism 2:** This model generates a context vector directly, with the set of all paragraph embeddings and the bag representation. Consider $P = (f(P_1), f(P_2), ..., f(P_N))^T$ be the matrix containing $f(P_i)$ in the i^{th} row. Now, the attention model is composed of the matrix $W \in \mathcal{R}^{d \times m}$ which defines the similarity between P and B, the context vector is simply a function $F(B, P) = P \cdot W \cdot B$
- **Decoder LSTM Network:** At any time instant t, the hidden state vector h_t of the LSTM, (which represents the document representation at that time instant) is given to the attention model, and a context vector is generated as in the equations below. This context vector $\overline{z}_c(t)$ is passed to the next LSTM cell, which outputs the label y_t (in some format; probably as a softmax over the labels) at that time step, or more complex networks can be used which

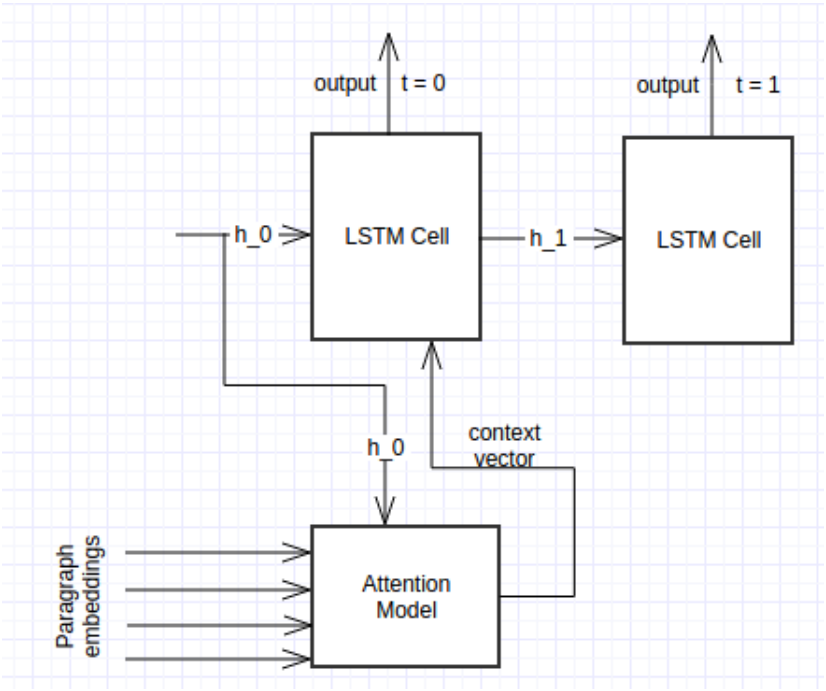

Figure 3.2: *Decoder Module for the model*

utilize the output of the LSTM y_t.

$$h_t, y_t = RNN(h_{t-1}, \bar{z}_c(t), l_t)$$
$$\bar{z}_c(t) = \sum_{i \in [1,N]} \alpha_i(h_{t-1}) f(P_i)$$
$$\alpha_i(x) = \frac{exp(\beta x^T W f(P_i))}{\sum_i exp(\beta x^T W f(P_i))}$$

3.2.5 Disadvantage of the model

- The model above predicts the labels in a sequential manner. In case of caption generation or natural language processing, an order is justified as English sentences exhibit a high probability $P(x_i|x_{1:i-1})$. On the other hand, labels don't exhibit any such order. Once way could be to enforce an order into the label sequence, as in rank the labels or number them in some way (hierarchical labeling or by prior distribution in the train data), but this way of artificially enforcing an order which is not a derivative of the structure is not usually a good way to do it.

3.3 Learning Joint Label and Instance Embeddings

The main drawback of the encoder-decoder approach is the enforcement of a sequential order on the output labels. Instead in the current approach we introduce representing the instances in the same space as labels which facilitates the idea of **paragraph2vec**, as now the embeddings of two similar paragraphs will be close and those of dissimilar paragraphs will be far from each other.

3.3.1 Joint Label-Instance Learning on Images

In this piece of work, Wang et.al. [10] apply joint multi-label learning to the task of multi label image classification. It consists of 2 parts: CNN part and RNN part. The CNN part generates semantic representations from images, and the RNN part models label dependency and label/image relationship.

Multi label prediction is converted into a ordered prediction path. Probability of a prediction path is computed by the RNN. The image, label, and recurrent representations are projected to the same low-dimensional space to model the image-text relationship and label redundancy. RNN model has been used as a compact and powerful representation of the label co-occurrence dependency in this low dimensional space. It takes the embedding of the predicted label at each time step and maintains a hidden state to model the label co-occurrence information.

Recurrent layer takes the label embedding of the previously predicted label, and models co-occurrence dependencies in its hidden recurrent states by learning nonlinear functions:

$$o(t) = h_o(r(t-1), w_k(t)), \quad r(t) = h_r(r(t-1), w_k(t))$$

where $r(t)$ and $o(t)$ are the hidden states and outputs of the RNN at the time step t, respectively, $w_k(t)$ is the label embedding of the t-th label in the prediction path.

Output of RNN and the image representation are projected into the same low-dimensional space as the label embedding. $x_t = h(U_o^x o(t) + U_I^x I)$

The CNN-RNN framework described above also learns a joint label/image embedding. Labels are highly semantic to its nearest neighbor labels. See Figure 3.3 It was observed that in the joint embedding space, the image and its labels are semantically relevant. So, joint embeddings efficiently signify the relevance of label-image relationship.

The label scores can be calculated by computing distance between x_t and each label embedding.

During the time of inference, a prediction path $(l_1, l_2, \cdots, l_n)$ needs to be chosen such that it maximizes $P(l_1, \cdots, l_n|I)$ which is the same as maximizing $P(l_k|I, l_1, \cdots, l_{k-1})$. Greedy approach is problematic because the if the first label predicted is wrong, the entire sequence can't be predicted correctly. Hence, beam search is used to predict the sequence of labels.

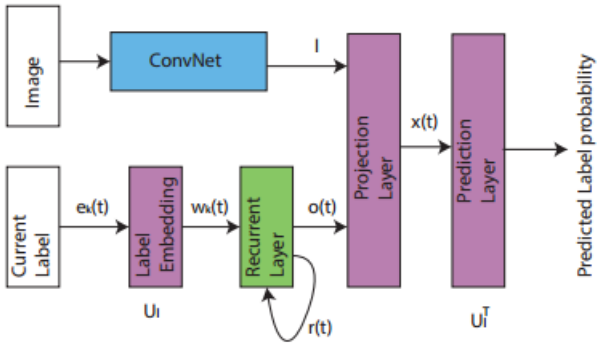

Figure 3.3: *Schematic of the CNN-RNN network*

3.3.2 Attention + Joint Label learning for MIML

In the MIML learning problem, it seems quite intuitive that an attention-mechanism is required in order to be able to focus on some specific instances/paragraphs in a bag/document and give out the label which highly correlates with these instances. Moreover, joint learning of label embeddings and instance embeddings helps in ensuring that we have meaningful paragraph embeddings which can act as **paragraph2vec** vectors. We, therefore, integrate an attention mechanism in a model similar to the joint learner in images described above, combining label correlation, joint embeddings and attention model.

Overall Idea Like in the previous section, we use a CNN to get the paragraph embeddings, and a RNN to generate label embeddings, after which these vectors are passed into a projection layer and then into the prediction layer. In the previous case, the entire image labels were available as a part of the dataset, and therefore, simply the generation of an image embedding was followed by the matching of the label and the paragraph embeddings. On the other hand, the main crux here is that, in case of MIML, label supervision is available at the bag level (document labels) but we wish to learn the paragraph and the label embeddings jointly. Also, one way of learning joint embeddings is by summarizing the document by the taking an average of the paragraphs' content and then using this bag embedding to correlate with the label embedding. But we feel that a better approach is to instead generate a context vector (weighted average of the paragraph representations) and use this to correlate with the labels, this should help get better embeddings which are robust to the presence of other labels in the training data.

Paragraph Embeddings Paragraph embeddings are generated in the exact same manner as the method described in the Encoder-Decoder Approach. Paragraph embedding of paragraph P_i is

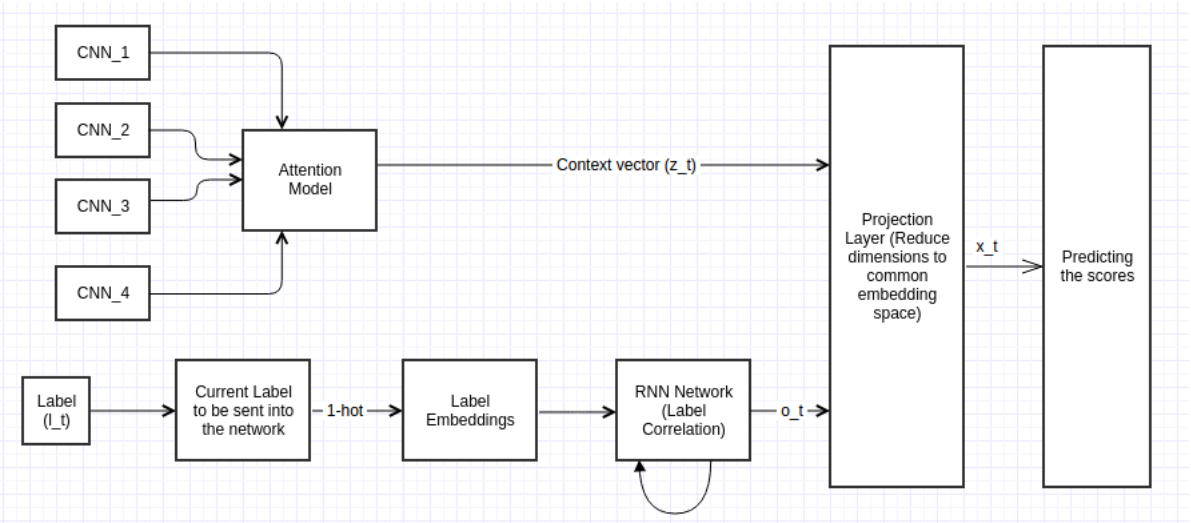

Figure 3.4: *Proposed Joint Embedding Learning Model for MIML*

given by $f(P_i)$, where $f(.)$ is the embedding function and is parametrized by a CNN.

$$f(P_i) = CNN(P_i)$$

Label Embeddings A label l_t is represented as a one hot vector $\in \mathbb{R}^d$. At every time step the previous predicted output label is passed into the RNN cell. Another possible idea is to pass the correct labels one by one into the RNN cell during training and asking the network to learn the label embeddings. Here, h_t represents the RNN hidden state vector, and o_t is the output of the RNN cell at every time step.

$$h_t, o_t = RNN(h_{t-1}, \bar{l}_t)$$

Generating a context vector A context vector is a vector $\in \mathbb{R}^d$, and is a weighted sum of the paragraph embedding vectors. The weights given to each of the individual paragraph embeddings can be varied to generate different contexts, for the purpose of generation of good joint embeddings. The context vector is generated using an attention model, so that focus on some specific areas of the paragraph-embedding state space could be maintained, so as to be able to learn better joint distributions over the paragraph and label embeddings. (Attention can enhance the occurrence of a particular label in the context vector and then correlate the label embeddings of that label with

this context vector and both should lie close in the joint representation space.)

$$z_c(t) = \sum_i \alpha_i(l_t) f(P_i)$$

$$\alpha_i(l_t) = \frac{exp(\beta l_t^T W f(P_i))}{\sum_k exp(\beta l_t^T W f(P_i))}$$

Projection and Prediction Layers The aim of the projection layer is to reduce the dimensionality of the embeddings and then the prediction layer gives a score based on the similarity between the label and the paragraph embeddings. The output of the projection layer is a vector y_t (with reduced dimensions) This vector y_t is passed on to a fully connected layer to output the next label.

$$y_t = \psi(g_t(D), 0_t) \ (reduction)$$

Use y_t to prediction the $(t+1)$-th label.

3.4 Objective Functions

A proper choice of objective functions is one of the most important factors affecting the performance for a deep neural network model. We plan to use two main loss functions for the purpose of training the network. These are squared loss, and a ranking loss called precision@k. We describe the details of these loss functions below.

Squared Loss Some of the previous works have shown that squared loss function is a good objective for multi instance learning. Therefore, we plan to use this function as one of the objectives for this problem.

Deeptron@k Deeptron@k is a loss strategy derived by modifications of perceptron@k strategy from Kar et.al 2015 [6] . The metric optimized by perceptron@k is the precision@k loss. In an MIML problem, often we want to give priorities to certain labels and this naturally enforces some ranking between the labels, as in some labels are more relevant than others. Precision@k loss aims to minimize the prediction error in the top k-labels being predicted (Labels are ranked according to a score). The mathematical definition of precision@k is given below.

Given n labeled data points $z_1, \cdots, z_n$ where $z_i = (x_i, y_i)$ and $y_i \in \{0, 1\}$ (positive points labeled 1, and negative points labeled 0) and a scoring function $s : X \to R$, let $\sigma_s \in S_n$ be the permutation that sorts points according to the scores given by s i.e. $s(x_{\sigma_{s(i)}}) \geq s(x_{\sigma_{s(j)}})$ for $i \leq j$.

The precision@k measure for this scoring function can be expressed as:

$$prec@k\ (s;\ z_1, \cdots, z_n) = \sum_{i=1}^{k} (1 - \mathbf{y}_{s(i)})$$

Here $\mathbf{y}$ is the label vector for the set of n points ($\mathbf{y} \in \{0,1\}^n$). In simple words, *prec@k* loss simply counts the number of irrelevant (negatively labeled points) that turn up in the top k points when ranked according to the scoring function $s(.)$. Few more notations are defined below, in order to be used in subsequent paragraphs.

Given two label vectors $\mathbf{y}', \mathbf{y}'' \in \{0,1\}^n$ define

$$\Delta_(\mathbf{y}', \mathbf{y}'') = \sum_{i=1}^{n} (1 - \mathbf{y}'_i)\mathbf{y}''_i \quad \Big| \quad K(\mathbf{y}', \mathbf{y}'') = \mathbf{y}'_i \mathbf{y}''_i$$

The algorithm given below presents the Deeptron@k optimization strategy which is a modified version of the Perceptron@k-max algorithm from [Kar, Jain 2015]. Deeptron@k returns the loss value over the entire mini-batch or batch of the training examples sent to it. The gradient descent step can then be performed using this value of loss. In the algorithm mentioned, $F(.)$ is the function which is parametrized by the neural network we are looking at and $loss(x_i)$ refers to the loss value caused due to the misclassification of the point x_i by the network.

Algorithm 2 Deeptron@K Algorithm

$W^{(0)} \leftarrow 0$, $t \leftarrow 0$, $L \leftarrow 0$
while batch stream not exhausted **do**
  $t \leftarrow t + 1$
  Receive one data point $x_t \in \mathbb{R}^d$ and its label $\mathbf{y}_t \in \{0,1\}^L$
  Calculate the scores $s = F(x_t) \in \{0,1\}^L$; Init $\hat{\mathbf{y}}_t \leftarrow 0 \in \{0,1\}^L$
  Set $\hat{\mathbf{y}}_t(j) \leftarrow 1$, if rank of label j according to s_t, is $\leq k$
  Compute precision@k loss: $\Delta_t \leftarrow \sum_{j=1}^{L} (1 - \mathbf{y}_j)\hat{\mathbf{y}}_t$
  if $\Delta_t = 0$ **then**
    $W^t \leftarrow W^{t-1}$
  else
    Compute $FN = \{j | \mathbf{y}_j = 1, \hat{\mathbf{y}}_j = 0, rank(j) \leq \Delta_t + k\}$
    Compute $FP = \{j | \mathbf{y}_j = 0, \hat{\mathbf{y}}_j = 1\}$
    Compute $Loss_{FP} = \sum_{i=1}^{|FP|} loss(x_i)$, $x_i \in FP$
    Compute $Loss_{FN} = \sum_{i=1}^{|FN|} loss(x_i)$, $x_i \in FN$
    $L \leftarrow L + Loss_{FP} + Loss_{FN}$
  end if
return L

Because of the nature of the instance of the MIML problem we are dealing with, the Deeptron@k loss seems to be more apt as compared to the squared error loss. For every instance/paragraph,

we are concerned with the few labels relevant to each. Hence, any error in prediction after a certain value of k is not considered an error. Therefore, the *prec@k* loss well suits the need of this problem.

Chapter 4

Conclusion and Future Work

In this report, we mainly described a deep learning approach for the Multi Instance Multi Label Learning Problem and showed how we came up to it and reasoned about the correctness of the approach mentioned. As earlier mentioned, the main motivation behind applying deep learning models on the problem is because a simple experiment of running a LSTM encoder decoder without any optimization gave a F score which was quite close to some of the best values obtained across the non deep learning papers on MIML. We also compared the approach with the existing approaches currently. The second approach of ours, where we jointly learn the label and instance embeddings seems to be promising especially taking into account the results of the CNN-RNN [10] paper.

As a part of the future work, we would first like to immediately implement the current model proposed and test how it performs with the prec@k loss function, Deeptron@k. We then want to explore more possibilities of improving the model, newer attention mechanisms. As a part of our model, we also want to introduce a new notion of paragraph2vec vector embeddings for paragraphs in a document. We believe that the approach we propose should work well with the scaling issues too.

Bibliography

[1] Apoorv Aggarwal, Sandip Ghoshal, Ankith M. S. Shetty, Suhit Sinha, Ganesh Ramakrishnan, Purushottam Kar, and Prateek Jain. Scalable optimization of multivariate performance measures in multi-instance multi-label learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1698–1704, 2017.

[2] Forrest Briggs, Xiaoli Z. Fern, and Raviv Raich. Rank-loss support instance machines for MIML instance annotation. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 534–542, 2012.

[3] Forrest Briggs, Xiaoli Z. Fern, Raviv Raich, and Qi Lou. Instance annotation for multi-instance multi-label learning. *ACM Trans. Knowl. Discov. Data*, 7(3):14:1–14:30, September 2013.

[4] Ji Feng and Zhi-Hua Zhou. Deep MIML network. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 1884–1890, 2017.

[5] Sheng-Jun Huang and Zhi-Hua Zhou. Fast multi-instance multi-label learning. *CoRR*, abs/1310.2049, 2013.

[6] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 189–198, 2015.

[7] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 587–592, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[8] Xiaoli Li, Philip S. Yu, Bing Liu, and See-Kiong Ng. Positive unlabeled learning for data stream classification. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA*, pages 259–270, 2009.

[9] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. Partially supervised classification of text documents. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pages 387–394, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[10] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. CNN-RNN: A unified framework for multi-label image classification. *CoRR*, abs/1604.04573, 2016.

[11] Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. MIML: A framework for learning with ambiguous objects. *CoRR*, abs/0808.3231, 2008.