Leveraging Distributional Semantics for Multi-Label Learning

Rahul Wadbude  
IIT Kanpur  
warahul@iitk.ac.in

Vivek Gupta  
Microsoft Research  
t-vigu@microsoft.com

Piyush Rai  
IIT Kanpur  
piyush@iitk.ac.in

Nagarajan Natarajan  
Microsoft Research  
t-nanata@microsoft.com

Harish Karnick  
IIT Kanpur  
hk@iitk.ac.in

Abstract

We present a novel and scalable label embedding framework for large-scale multi-label learning a.k.a. ExMLDS (Extreme Multi-Label Learning using Distributional Semantics). Our approach draws inspiration from ideas rooted in distributional semantics, specifically the Skip Gram Negative Sampling (SGNS) approach, widely used to learn word embeddings for natural language processing tasks. Learning such embeddings can be reduced to a certain matrix factorization. Our approach is novel in that it highlights interesting connections between label embedding methods used for multi-label learning and paragraph/document embedding methods commonly used for learning representations of text data. The framework can also be easily extended to incorporate auxiliary information such as label-label correlations; this is crucial especially when there are a lot of missing labels in the training data. We demonstrate the effectiveness of our approach through an extensive set of experiments on a variety of benchmark datasets, and show that the proposed learning methods perform favorably compared to several baselines and state-of-the-art methods for large-scale multi-label learning.

Introduction

Modern data generated in various domains are increasingly "multi-label" in nature; images (e.g. Instagram) and documents (e.g. Wikipedia) are often identified with multiple tags, online advertisers often associate multiple search keywords with ads, and so on. Multi-label learning is the problem of learning to assign multiple labels to instances, and has received a great deal of attention over the last few years; especially so, in the context of learning with millions of labels, now popularly known as extreme multi-label learning \cite{Jain2016}. Multi-label learning fall broadly under two classes: 1) embedding based methods, e.g. LEML \cite{Yu2014}, WSABIE \cite{Weston2010}, SLEEC \cite{Bhattia2015}, PD-SPARSE \cite{Yen2016}, and 2) tree-based methods \cite{Prabhu2014}. The key challenges in multi-label learning, especially when there are millions of labels, include a) the data may have a large fraction of labels missing, and b) the labels are often heavy-tailed \cite{Bhattia2015} \cite{Jain2016} and predicting labels in the tail becomes significantly hard for lack of training data. For these reasons, and the sheer scale of data, traditional multi-label classifiers are rendered impracticable. State-of-the-art approaches to extreme multi-label learning fall broadly under two classes: 1) embedding based methods, e.g. LEML \cite{Yu2014}, WSABIE \cite{Weston2010}, SLEEC \cite{Bhattia2015}, PD-SPARSE \cite{Yen2016}, and 2) tree-based methods \cite{Prabhu2014}. The first class of approaches rely on a key assumption that the binary label matrix is low rank and consequently the label vectors can be embedded into a lower-dimensional space. At the time of prediction, a decompression matrix is used to retrieve the original label vector from the low-dimensional embeddings. As corroborated by recent empirical evidence \cite{Bhattia2015} \cite{Jain2016}, approaches based on standard structural assumptions such as low-rank label matrix fail and perform poorly on the tail. The second class of methods (tree-based) methods for multi-label learning try to move away from rigid structural assumptions \cite{Prabhu2014} \cite{Jain2016}, and have been demonstrated to work very well especially on the tail labels.

In this work, we propose an embedding based approach, closely following the framework of SLEEC \cite{Bhattia2015}, that leverages a word vector embedding technique \cite{Mikolov2013} which has found resounding success in natural language processing tasks. Unlike other embedding based methods, SLEEC has the ability to learn non-linear embeddings by aiming to preserve only local structures and example neighborhoods. We show that by learning rich word2vec style embedding for instances (and labels), we can a) achieve competitive multi-label prediction accuracies, and often improve over the performance of the state-of-the-art embedding approach SLEEC and b) cope with missing labels, by incorporating auxiliary information in the form of label-label co-occurrences, which most of the state-of-the-art methods can not. Furthermore, our learning algorithm admits significantly faster implementation compared to other embedding based approaches. The distinguishing aspect of our work is that it draws inspiration from dis-

Submitted to Thirty-Second Association for the Advancement of Artificial Intelligence, AAAI 2018 (www.aaai.org). Do not distribute.
tributational semantics approaches (Mikolov et al. 2013
Le and Mikolov 2014), widely used for learning non-linear
representations of text data for natural language processing
tasks such as understand word and document semantics,
classifying documents, etc.

Our main contributions are:

1. We leverage an interesting connection between the problem
   of learning distributional semantics in text data analysis
   and the multi-label learning problem. To the best of
   our knowledge, this is a novel application.

2. The proposed objectives for learning embeddings can be
   solved efficiently and scalably; the learning reduces to a
   certain matrix factorization problem.

3. Unlike existing multi-label learning methods, our method
   can also leverage label co-occurrence information while
   learning the embeddings; this is especially appealing
   when a large fraction of labels are missing in the label
   matrix.

4. We show improvement in training time as compared to
   state-of-the-art label embedding methods for extreme multi-
   label learning, while being competitive in terms of label
   prediction accuracies; we demonstrate scalability and pre-
   diction performance on several state-of-the-art moderate-
   to-large scale multi-label benchmark datasets.

The outline of the paper is as follows. We begin by setting
up notation, background and describing the problem formu-
lation in Section 1. In Section 1, we present our training al-
gorithms based on learning word embeddings for understand-
ing word and document semantics. Here we propose two
objectives, where we progressively incorporate auxiliary in-
formation viz. label correlations. We present comprehensive
experimental evaluation in Section 1, and conclude.

Problem Formulation and Background

In the standard multi-label learning formulation, the learn-
ing algorithm is given a set of training instances
\{x_1, x_2, \ldots, x_n\}, where x_i \in \mathbb{R}^d and the associated label
vectors \{y_1, y_2, \ldots, y_n\}, where y_i \in \{0, 1\}^L. In real-world
multi-label learning data sets, one does not usually observe
irrelevant labels; here \(y_{ij} = 1\) indicates that the \(j\)th label is
relevant for instance \(i\) but \(y_{ij} = 0\) indicates that the label is
missing or irrelevant. Let \(Y \in \{0, 1\}^{n \times L}\) denote the matrix
of label vectors. In addition, we may have access to label-
label co-occurrence information, denoted by \(C \in \mathbb{Z}^{L \times L}\)
(e.g., number of times a pair of labels co-occur in some exter-
nal source such as the Wikipedia corpus). The goal in multi-
label learning is to learn a vector-valued function \(f : x \mapsto s\),
where \(s \in \mathbb{R}^L\) scores the labels.

Embedding-based approaches typically model \(f\) as a com-
posite function \(h(g(x))\) where, \(g : \mathbb{R}^d \rightarrow \mathbb{R}^d\) and
\(h : \mathbb{R}^d \rightarrow \mathbb{R}^L\). For example, assuming both \(g\) and \(h\)
as linear transformations, one obtains the formulation pro-
posed by (Yu et al. 2014). The functions \(g\) and \(h\) can be
learnt using training instances or label vectors, or both.
More recently, non-linear embedding methods have been
shown to help improve multi-label prediction accuracies
significantly. In this work, we follow the framework of
(Bhatia et al. 2015), where \(g\) is a linear transformation, but
\(h\) is non-linear, and in particular, based on \(k\)-nearest neigh-
bor in the embedded feature space.

In SLEEC, the function \(g : \mathbb{R}^d \rightarrow \mathbb{R}^d\) is given by
\(g(x) = V x\) where \(V \in \mathbb{R}^{d \times d}\). The function \(h : \mathbb{R}^d \rightarrow \mathbb{R}^L\)
is defined as:

\[
h(z; \{z_i, y_i\}_{i=1}^n) = \frac{1}{|N_k|} \sum_{i \in N_k} y_i,
\]

where \(z_i = g(x_i)\) and \(N_k\) denotes the \(k\)-nearest neighbor
training instances of \(z\) in the embedded space. Our algorithm
for predicting the labels of a new instance is identical to that
of SLEEC and is presented for convenience in Algorithm 1.

Note that, for speeding up predictions, the algorithm relies
on clustering the training instances \(x_i\); for each cluster of
instances \(Q^\tau\), a different linear embedding \(g^\tau\), denoted by
\(V^\tau\), is learnt.

Algorithm 1 Prediction Algorithm

\begin{tabular}{l}
\textbf{Input}: Test point: \(x\), no. of nearest neighbors \(k\), no. of
desired labels \(p\).
1. \(Q^\tau\) : partition closest to \(x\).
2. \(z \leftarrow V^\tau x\).
3. \(N_k \leftarrow k\) nearest neighbors of \(z\) in the embedded in-
stances of \(Q^\tau\).
4. \(s = h(z; \{z_i, y_i\}_{i \in Q^\tau})\) where \(h\) is defined in (1).
\end{tabular}
\begin{tabular}{l}
\textbf{return} top \(p\) scoring labels according to \(s\).
\end{tabular}

In this work, we focus on learning algorithms for the func-
tions \(g\) and \(h\), inspired by their successes in natural language
processing in the context of learning distributional seman-
tics (Mikolov et al. 2013, Levy and Goldberg 2014). In par-
icular, we use techniques for inferring word-vector embed-
dings for learning the function \(h\) using a) training label vec-
tors \(y_i\), and b) label-label correlations \(C \in \mathbb{R}^{L \times L}\).

Word embeddings are desired in natural language pro-
cessing in order to understand semantic relationships be-
tween words, classifying text documents, etc. Given a text
corpus consisting of a collection of documents, the goal is
to embed each word in some space such that words ap-
pearing in similar contexts (i.e. adjacent words in docu-
ments) should be closer in the space, than those that do not.
In particular, we use the word2vec embedding approach
(Mikolov et al. 2013) to learn an embedding of in-
stances, using their label vectors \(y_1, y_2, \ldots, y_n\), SLEEC also
uses nearest neighbors in the space of label vectors \(y\) in or-
der to learn the embeddings. However, we show in experi-
ments that word2vec based embeddings are richer and
help improve the prediction performance significantly, espe-
cially when there is a lot of missing labels. In the subsequent
section, we discuss our algorithms for learning the embed-
dings and the training phase of multi-label learning.

Learning Instance and Label Embeddings

There are multiple algorithms in the literature for learn-
ing word embeddings (Mikolov et al. 2013)

\begin{tabular}{l}
\textbf{Input}: Test point: \(x\), no. of nearest neighbors \(k\), no. of
desired labels \(p\).
1. \(Q^\tau\) : partition closest to \(x\).
2. \(z \leftarrow V^\tau x\).
3. \(N_k \leftarrow k\) nearest neighbors of \(z\) in the embedded in-
stances of \(Q^\tau\).
4. \(s = h(z; \{z_i, y_i\}_{i \in Q^\tau})\) where \(h\) is defined in (1).
\end{tabular}
\begin{tabular}{l}
\textbf{return} top \(p\) scoring labels according to \(s\).
\end{tabular}
Pennington, Socher, and Manning 2014). In this work, we use the Skip Gram Negative Sampling (SGNS) technique, for two reasons a) it is shown to be competitive in natural language processing tasks, and more importantly b) it presents a unique advantage in terms of scalability, which we will address shortly after discussing the technique.

Skip Gram Negative Sampling. In SGNS, the goal is to learn an embedding \( \mathbf{z} \in \mathbb{R}^d \) for each word \( w \) in the vocabulary. To do so, words are considered in the contexts in which they occur; context \( c \) is typically defined as a fixed size window of words around an occurrence of the word. The goal is to learn \( \mathbf{z} \) such that the words in similar contexts are closer to each other in the embedded space. Let \( w' \in c \) denote a word in the context \( c \) of word \( w \). Then, the likelihood of observing the pair \((w, w')\) in the data is modeled as a sigmoid of their inner product similarity:

\[
P(\text{Observing } (w, w')) = \frac{1}{1 + \exp(-\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle)}.
\]

To promote dissimilar words to be further apart, negative sampling is used, wherein randomly sampled negative examples \((w, w')\) are used. Overall objective favors \( \mathbf{z}_w, \mathbf{z}_{w'} \) that maximize the log likelihood of observing \((w, w')\), for \( w' \in c \), and maximizing the log likelihood of \( P(\text{not observing } (w, w')) = 1 - P(\text{Observing } (w, w')) \) for randomly sampled negative instances. Typically, \( n_- \) negative examples are sampled per observed example, and the resulting SGNS objective is given by:

\[
\max_{\mathbf{z}} \sum_{w} \left( \sum_{w':(w', w)} \log(\sigma(\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle)) \right) + \\
\frac{n_-}{\# w} \sum_{w'} \log(\sigma(-\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle)),
\]

(2)

where \( \# w \) denotes the total number of words in the vocabulary, and the negative instances are sampled uniformly over the vocabulary.

Embedding label vectors

We now derive the analogous embedding technique for multi-label learning. A simple model is to treat each instance as a "word"; define the "context" as \( k \)-nearest neighbors of a given instance in the space formed by the training label vectors \( \mathbf{y}_i \), with cosine similarity as the metric. We then arrive at an objective identical to (2) for learning embeddings \( \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n \) for instances \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \) respectively:

\[
\max_{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n} \sum_{i=1}^{n} \left( \sum_{j, \mathbf{y}_i(j)} \log(\sigma(\langle \mathbf{z}_i, \mathbf{z}_j \rangle)) \right) + \\
\frac{n_-}{n} \sum_{i=1}^{n} \log(\sigma(-\langle \mathbf{z}_i, \mathbf{z}_j \rangle)).
\]

(3)

Note that \( \mathcal{N}_k(\mathbf{y}_i) \) denotes the \( k \)-nearest neighborhood of \( i \)th instance in the space of label vectors or instance embedding. After learning label embeddings \( \mathbf{z}_i \), we can learn the function \( g : \mathbf{x} \rightarrow \mathbf{z} \) by regressing \( \mathbf{x} \) onto \( \mathbf{z} \), as in SLEEC. Solving (3) for \( \mathbf{z}_i \) using standard word2vec implementations can be computationally expensive, as it requires training multiple-layer neural networks. Fortunately, the learning can be significantly sped up using the key observation by Levy and Goldberg 2014. Levy and Goldberg 2014 showed that solving SGNS objective is equivalent to matrix factorization of the shifted positive point-wise mutual information (SPPMI) matrix defined as follows. Let \( M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle \).

\[
P_{\text{SPPMI}}_{ij}(M) = \log \left( \frac{M_{ij} \cdot |M|}{\sum_k M_{i,k} + \sum_k M_{k,j}} \right)
\]

(4)

Here, \( P \) is the point-wise mutual information matrix of \( M \) and \( |M| \) denotes the sum of all elements in \( M \). Solving the problem (3) reduces to factorizing the shifted PPMI matrix \( M \).

Finally, we use ADMM (Boyd et al. 2011) to learn the regressors \( V \) over the embedding space formed by \( \mathbf{z}_i \). Overall training algorithm is presented in Algorithm 2.

Algorithm 2 Learning embeddings via SPPMI factorization (ExMLDS1).

**Input.** Training data \((\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n\).

1. Compute \( \tilde{M} := \text{SPPMI}(M) \) in (4), where \( M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle \).
2. Let \( U, S, V = \text{svd}(\tilde{M}) \), and preserve top \( d' \) singular values and singular vectors.
3. Compute the embedding matrix \( Z = US^{0.5} \), where \( Z \in \mathbb{R}^{n \times d'} \), where \( i \)th row gives \( \mathbf{z}_i \).
4. Learn \( V \) s.t. \( XV^T = Z \) using ADMM (Boyd et al. 2011), where \( X \) is the matrix with \( \mathbf{x}_i \) as rows.

**Return** \( V, Z \).

We refer to Algorithm 2 based on fast PPMI matrix factorization for learning label vector embeddings as ExMLDS1. We can also optimize the objective \( \min \) using a neural network model (Mikolov et al. 2013); we refer to this word2vec method for learning embeddings in Algorithm 2 as ExMLDS2.

Using label correlations

In various practical natural language processing applications, superior performance is obtained using joint models for learning embeddings of text documents as well as individual words in a corpus (Dai, Olah, and Le 2015). For example, in PV-DBoW (Dai, Olah, and Le 2015), the objective while learning embeddings is to maximize similarity between embedded documents and words that compose the documents. Negative sampling is also included, where the objective is to minimize the similarity between the document this space for speed up, and therefore the label vectors are likely to preserve more discriminative information within clusters.
embeddings and the embeddings of high frequency words. In multi-label learning, we want to learn the embeddings of labels as well as instances jointly. Here, we think of labels as individual words, whereas label vectors (or instances with the corresponding label vectors) as paragraphs or documents. As alluded to in the beginning of Section 3.3, in many real world problems, we may also have auxiliary label correlation information, such as label-label co-occurrence. We can easily incorporate such information in the joint modeling approach outlined above. To this end, we propose the following objective that incorporates information from both label vectors as well as label correlations: \[ \text{max}_{z,z'} \bar{O} \] \[ \bar{O}_z = \sum_{i=1}^{L} \left( \sum_{j \in N_k(i)} \log(\sigma(\langle z_i, z_j \rangle))) + \frac{n^1}{L} \sum_{j'} \log(\sigma(-\langle z_i, z_{j'} \rangle))) \right) \] \[ \bar{O}_z = \sum_{i=1}^{n} \left( \sum_{j \in N_k(M(i))} \log(\sigma(\langle z_i, z_j \rangle))) + \frac{n^2}{n} \sum_{j'} \log(\sigma(-\langle z_i, z_{j'} \rangle))) \right) \] \[ \bar{O}_{\{z,z'\}} = \sum_{i=1}^{L} \left( \sum_{j \in N_k(C(i))} \log(\sigma(\langle z_i, z_j \rangle))) + \frac{n^3}{L} \sum_{j'} \log(\sigma(-\langle z_i, z_{j'} \rangle))) \right) \] (5)

Here, \( z_i, i = 1, 2, \ldots, n \) denote embeddings of instances while \( z_i, i = 1, 2, \ldots, L \) denote embeddings of labels. \( N_k(i,:) \) denotes the \( k \)-nearest neighborhood of \( i \)-th instance in the space of label vectors. \( N_k(C(i,:)) \) denotes the \( k \)-nearest neighborhood of \( i \)-th label in the space of labels. Here, \( M \) defines instance-instance correlation i.e. \( M_{ij} = \langle y_i, y_j \rangle \) and \( C \) is the label-label correlation matrix. Clearly, (7) above is identical to (5). \( \bar{O}_z \) tries to embed labels \( z_i \) in a vector space, where correlated labels are closer; \( \bar{O}_z \) tries to embed instances \( z_i \) in such a vector space, where correlated instances are closer; and finally, \( \bar{O}_{\{z,z'\}} \) tries to embed labels and instances in a common space where labels occurring in the \( i \)-th instance are closer to the embedded instance.

Overall the combined objective \( \bar{O}_{\{z,z'\}} \) promotes learning a common embedding space where correlated labels, correlated instances and observed labels for a given instance occur closely. Here \( \mu_1, \mu_2 \) and \( \mu_3 \) are hyper-parameters to weight the contributions from each type of correlation. \( n^1 \) negative examples are sampled per observed label, \( n^2 \) negative examples are sampled per observed instance in context of labels and \( n^3 \) negative examples are sampled per observed instance in context of instances. Hence, the proposed objective efficiently utilizes label-label correlations to help improve embedding and, importantly, to cope with missing labels. The complete training procedure using SPPMI factorization is presented in Algorithm 3. Note that we can use the same arguments given by [Levy and Goldberg 2014] to show that the proposed combined objective (5) is solved by SPPMI factorization of the joint matrix \( A \) given in Step 1 of Algorithm 3.

**Algorithm 3** Learning joint label and instance embeddings via SPPMI factorization (ExMLDS3).

**Input.** Training data \((x_i, y_i), i = 1, 2, \ldots, n \) and \( C \) (label-label correlation matrix) and objective weighting \( \mu_1, \mu_2 \) and \( \mu_3 \).

1. Compute \( \bar{A} := \text{SPPMI}(A) \) in (4); write \[ A = \left( \begin{array}{c} \mu_2 M \\ \mu_3 Y \\ \mu_1 C \end{array} \right), \]

   \[ M_{ij} = \langle y_i, y_j \rangle, \ Y \text{ is label matrix with } y_i \text{ as rows.} \]

2. Let \( U, S, V = \text{svd}(\bar{A}) \), and preserve top \( d' \) singular values and singular vectors.

3. Compute the embedding matrix \( Z = US^{0.5} \); write \[ Z = \left( \begin{array}{c} Z_1 \\ Z_2 \end{array} \right), \]

   where rows of \( Z_1 \in \mathbb{R}^{n \times d'} \) give instance embedding and rows of \( Z_2 \in \mathbb{R}^{L \times d'} \) give label embedding.

4. Learn \( V \) s.t. \( XV^T = Z_1 \) using ADMM (Boyd et al. 2011), where \( X \) is the matrix with \( x_i \) as rows.

**return** \( V, Z \).

**Algorithm 4** Prediction Algorithm with Label Correlations (ExMLDS3 prediction).

**Input:** Test point: \( x \), no. of nearest neighbors \( k \), no. of desired labels \( p \), \( V \), embeddings \( Z_1 \) and \( Z_2 \).

1. Use Algorithm 4 (Step 3) with input \( Z_1, k, p \) to get score \( s_1 \).

2. Get score \( s_2 = Z_2 \langle x \rangle \).

3. Get final score \( s = \frac{s_1}{\|s_1\|} + \frac{s_2}{\|s_2\|} \).

**return** top \( p \) scoring labels according to \( s \).

At test time, given a new data point we could use the Algorithm 1 to get top \( p \) labels. Alternately, we propose to use Algorithm 4 that also incorporates similarity with label embeddings \( Z_2 \) along with \( Z_1 \) during prediction, especially when there are very few training labels to learn from. In practice, we find this prediction approach useful. Note the \( z_i \) corresponds to the \( i \)-th row of \( Z_1 \), and \( z_i \) corresponds to the \( i \)-th row of \( Z_2 \). We refer the Algorithm 3 based on the combined learning objective (5) as ExMLDS3.

**Experiments**

We conduct experiments on commonly used benchmark datasets from the extreme multi-label classification repos-
2. DXML (Zhang et al. 2017) is a recent deep learning solution. We use the standard, practically relevant, precision at $k$ (denoted by Prec@$k$) as the evaluation metric of the prediction performance. Prec@$k$ denotes the number of correct labels in the top $k$ predictions. We run our code and all other baselines on a Linux machine with 40 cores and 128 GB RAM. We implemented our prediction Algorithms 1 and 4 in MATLAB. Learning Algorithms 2 and 3 are implemented partly in Python and partly in MATLAB. The source code will be made available later.

We evaluate three models (a) EXMLDS1 i.e. Algorithm 2 based on fast PPMI matrix factorization for learning label embeddings as described in Section 4, (b) EXMLDS2 based on optimizing the objective 3 as described in section 5, using neural network (Mikolov et al. 2013) (c) EXMLDS3 i.e. Algorithm 5 based on combined learning objective 5.

**Compared methods.** We compare our algorithms with the following baselines.

1. SLSC (Bhatia et al. 2015), which was shown to outperform all other embedding baselines on the benchmark datasets.
2. LEM (Yu et al. 2014), an embedding based method. This method also facilitates incorporating label information (though not proposed in the original paper); we use the code given by the authors of LEM which uses item feature 4. We refer to the latter method that uses label correlations as LEM-IMC.
3. FASTXML (Prabhu and Varma 2014), a tree-based method.
4. PD-Sparse (Yen et al. 2016), recently proposed embedding based method.
5. PFASTXML (Jain, Prabhu, and Varma 2016) is an extension of FASTXML; it was shown to outperform all other tree-based baselines on benchmark datasets.
6. DiSMEC (Babbar and Schölkopf 2017) is recently proposed scalable implementation of the ONE-VS-ALL method.
7. DXML (Zhang et al. 2017) is a recent deep learning solution for multi-label learning.
8. ONE-VS-ALL (Zhang et al. 2017) is traditional one vs all multi-label classifier.

We report all baseline results from the the extreme classification repository, where they have been curated; note that all the relevant research work use the same train-test split for benchmarking.

**Hyperparameters.** We use the same embedding dimensionality, preserve the same number of nearest neighbors for learning embeddings as well as at prediction time, and the same number of data partitions used in SLEEC (Bhatia et al. 2015) for our method EXMLDS1 and EXMLDS2. For small datasets, we fix negative sample size to 15 and number of iterations to 35 during neural network training, tuned based on a separate validation set. For large datasets (4 and 5 in Table 1), we fix negative sample size to 2 and number of iterations to 5, tuned on a validation set. In EXMLDS3, the parameters (negative sampling) are set identical to EXMLDS1. For baselines, we either report results from the respective publications or used the best hyperparameters reported by the authors in our experiments, as needed.

**Performance evaluation.** The performance of the compared methods are reported in Table 2. Performances of the proposed methods EXMLDS1 and EXMLDS2 are found to be similar in our experiments, as they optimize the same objective 3 so we include only the results of EXMLDS1 in the Table. We see that the proposed methods achieve competitive prediction performance among the state-of-the-art embedding and tree-based approaches. In particular, note that on Medialmill and Delicious-200K datasets our method achieves the best performance.

**Training time.** Objective 4 can be trained using a neural network, as described in (Mikolov et al. 2013). For training the neural network model, we give as input the $k$-nearest neighbor instance pairs for each training instance $i$, where the neighborhood is computed in the space of the label vectors $y_i$. We use the Google word2vec code 3 for training. We parallelize the training on 40 cores Linux machine for speed-up. Recall that we call this method EXMLDS2. We compare the training time with our method EXMLDS1, which uses a fast matrix factorization approach for learning embeddings. Algorithm 2 involves a single SVD as opposed to iterative SVP used by SLEEC and therefore it is significantly faster. We present training time measurements in Table 2. As anticipated, we observe that EXMLDS2 which uses neural networks is slower than EXMLDS1 (with 40 cores). Also, among the smaller datasets, EXMLDS1 trains 14x faster compared to SLEEC on Bibtex dataset. In the large dataset, Delicious-200K, EXMLDS1 trains 5x faster than SLEEC.

**Coping with missing labels.** In many real-world scenarios, data is plagued with lots of missing labels. A desirable property of multi-label learning methods is to cope with missing labels, and yield good prediction performance with very few training labels. In the dearth of training labels, auxiliary information such as label correlations can come in handy. As described in Section 4, our method EXMLDS3 can learn from additional information. The benchmark datasets, however, do not come with auxiliary information. To simulate this setting, we hide 80% non-zero entries of the training label matrix, and reveal the 20% training labels to learn algorithms. As a proxy for label correlations matrix $C$, we simply use the label-label co-occurrence from the 100% training data, i.e. $C = Y^T Y$ where $Y$ denotes the full training matrix. We give higher weight $\mu_1$ to $\varnothing$ during training in Algorithm 5. For prediction, We use Algorithm 4 which...
Table 1: Dataset statistics

| Dataset         | Feature | Label | Train | Test  |
|-----------------|---------|-------|-------|-------|
| rcv1v2          | 47236   | 101   | 3000  | 3000  |
| Deliciou-200K   | 782585  | 205443| 196606| 100095|
| Mediamill       | 120     | 101   | 30993 | 12914 |
| Bibtex (Katakis, Tsoumakas, and Vlahavas 2008) | 1836 | 159 | 4880 | 2515 |
| Delicious (Tsoumakas, Katikis, and Vlahavas 2008) | 500 | 983 | 12920 | 3185 |
| EURlex-4K (Loza Mencia and Furnkranz 2008) | 5000 | 3993 | 15539 | 3809 |

Table 2: Comparing training times (in seconds) of different methods

| Method          | ExM LDS1 | Bibtex | Delicious | EURlex | Mediamill | Delicious-20K |
|-----------------|----------|--------|-----------|--------|------------|---------------|
| ExM LDS1        | 23       | 259    | 880.9     | 1200   | 1937       | 782585        |
| ExM LDS2        | 143.19   | 781.94 | 880.64    | 12000  | 13000      | 8912          |

Table 3: Comparing prediction performance of different methods (− mean unavailable results). Note that although SLEEC performs slightly better, our model is much faster as shown in the results in Table 2. Also note the performance of our model in Table 4 when a significant fraction of labels are missing is considerably better than SLEEC.

Table 4: Evaluating competitive methods in the setting where 80% of the training labels are hidden

| Dataset         | Prec@k | ExM LDS3 | SLEEC | LEML | LEML-IMC |
|-----------------|--------|---------|-------|------|----------|
| P@1             | 48.51  | 30.5    | 35.98 | 41.23|
| P@3             | 28.43  | 14.9    | 20.1  | 25.25|
| P@5             | 20.7   | 9.81    | 15.50 | 18.56|
| P@1             | 60.28  | 51.4    | 26.22 | 39.24|
| P@3             | 44.87  | 37.64   | 22.94 | 32.66|
| P@5             | 35.31  | 29.62   | 19.02 | 26.54|
| P@1             | 81.67  | 41.8    | 64.83 | 73.68|
| P@3             | 52.82  | 17.48   | 42.56 | 48.56|
| P@5             | 37.74  | 10.63   | 31.68 | 34.82|

Conclusions and Future Work

We proposed a novel objective for learning label embeddings for multi-label classification, that leverages word2vec embedding technique; furthermore, the proposed formulation can be optimized efficiently by SPPMI matrix factorization. Through comprehensive experiments, we showed that the proposed method is competitive compared to state-of-the-art multi-label learning methods in terms of prediction accuracy. We proposed a novel objective that incorporates side information, that is particularly effective in handling missing labels. One promising extension of our objective is to do joint learning of embeddings $Z$ and regressor $Y$ using gradient descent as shown in the Supplementary Material.

References

[Babbar and Schölkopf 2017] Babbar, R., and Schölkopf, B. 2017. Dismec: Distributed sparse machines for extreme multi-label classification. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM ’17, 721–729. New York, NY, USA: ACM.

[Bhatia et al. 2015] Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse local embeddings for extreme multi-label classification. In Advances in Neural Information Processing Systems, 730–738.

[Boyd et al. 2011] Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.;
and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3(1):1–122.

[Dai, Olah, and Le 2015] Dai, A. M.; Olah, C.; and Le, Q. V. 2015. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*.

[Jain, Prabhu, and Varma 2016] Jain, H.; Prabhu, Y.; and Varma, M. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 935–944. ACM.

[Katakis, Tsoumakas, and Vlahavas 2008] Katakis, I.; Tsoumakas, G.; and Vlahavas, I. 2008. Multilabel text classification for automated tag suggestion. *ECML PKDD discovery challenge 75*.

[Le and Mikolov 2014] Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*, volume 14, 1188–1196.

[Levy and Goldberg 2014] Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, 2177–2185.

[Lewis et al. 2004] Lewis, D. D.; Yang, Y.; Rose, T. G.; and Li, F. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research* 5(Apr):361–397.

[Loza Mencía and Fürnkranz 2008] Loza Mencía, E., and Fürnkranz, J. 2008. Efficient pairwise multilabel classification for large-scale problems in the legal domain. *Machine Learning and Knowledge Discovery in Databases* 50–65.

[Mikolov et al. 2013] Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, 1532–1543.

[Prabhu and Varma 2014] Prabhu, Y., and Varma, M. 2014. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 263–272. ACM.

[Snoek et al. 2006] Snoek, C. G.; Worrning, M.; Van Gemert, J. C.; Geusebroek, J.-M.; and Smeulders, A. W. 2006. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th ACM international conference on Multimedia*, 421–430. ACM.

[Tsoumakas, Katakis, and Vlahavas 2008] Tsoumakas, G.; Katakis, I.; and Vlahavas, I. 2008. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD’08)*, 30–44.

[Weston, Bengio, and Usunier 2010] Weston, J.; Bengio, S.; and Usunier, N. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning* 81(1):21–35.

[Yen et al. 2016] Yen, I. E.-H.; Huang, X.; Ravikumar, P.; Zhong, K.; and Dhillon, I. 2016. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International Conference on Machine Learning*, 3069–3077.

[Yu et al. 2014] Yu, H.-F.; Jain, P.; Kar, P.; and Dhillon, I. 2014. Large-scale multi-label learning with missing labels. In *International Conference on Machine Learning*, 593–601.

[Zhang et al. 2017] Zhang, W.; Wang, L.; Yan, J.; Wang, X.; and Zha, H. 2017. Deep extreme multi-label learning. *arXiv preprint arXiv:1704.03718*.
Leveraging Distributional Semantics for Multi-Label Learning

A1. Joint Embedding and Regression

\[ O_i^{t+1} = O_i^t + \eta \nabla_V O_i \]

Gradient of objective w.r.t to \( V \) i.e. \( \nabla_V O_i \) is describe in detail below:

Given,

\[ K_{ij} = \langle z_i, z_j \rangle = \langle z_i^T, z_j \rangle \]

\[ z_i = Vx_i, \text{ where } V \in \mathbb{R}^{d \times d} \]

Objective:

\[
\max_{z_1, z_2, \ldots, z_n} \sum_{i=1}^{n} \left( \sum_{j \in \mathcal{N}_i(y_i)} \log \left( \sigma\left( (Vx_i, Vx_j) \right) \right) + \frac{n - n}{n} \sum_{j'} \log \left( \sigma\left( -\langle z_i, z_j' \rangle \right) \right) \right)
\]

rewriting with s.t.t \( V \) and \( K_{ij} \), we obtained

\[
\max_V \sum_{i=1}^{n} \left( \sum_{j \in \mathcal{N}_i(y_i)} \log \left( \sigma\left( (Vx_i, Vx_j) \right) \right) + \frac{n - n}{n} \sum_{j'} \log \left( \sigma\left( -\langle z_i, z_j' \rangle \right) \right) \right),
\]

rewriting for only \( i^{th} \) instance, we have

\[
\nabla_V O_i = \sum_{j \in \mathcal{N}_i(y_i)} \sigma(-K_{ij}) \nabla_V K_{ij} - \frac{n - n}{n} \sum_{j'} \sigma(K_{ij'}) \nabla_V K_{ij'}
\]
here, \( \nabla_V K_{ij} \) can be obtain through,

\[
\nabla_V K_{ij} = V(x_i x_j^T + x_j x_i^T) = z_i z_j^T + z_j z_i^T
\]

Sometime cosine similarity perform better then dot product because of scale invariant, in that case the gradient would modify to:

\[
K_{ij} = \frac{\langle z_i, z_j \rangle}{\|z_i\| \|z_j\|} = \frac{\langle z_i^T z_j \rangle}{\|z_i\| \|z_j\|}
\]

\[
\nabla_V \langle z_i, z_j \rangle = \nabla_V \langle Vx_i, z_j \rangle + \nabla_V \langle Vx_i, z_i \rangle = \langle z_i x_j^T + z_j x_i^T \rangle
\]

A2. SGNS Objective as Implicit SPPMI factorization

The SGNS \cite{Mikolov2013} objective is as follows:

\[
O_i = \sum_{j \in S_i} \log(\sigma(K_{ij})) + \sum_{k \sim p_D} \mathbb{E}_{k \sim p_D} [\log(\sigma(-K_{ik}))]
\]

where, \( p_D = (\#k)^{0.75}/D \), \( D \) is collection of all word-context pairs and \( K_{ij} \) represent dot-product similarity between the embeddings of a given word (i) and context (j). Here, \#k represent total number of word-context pairs with context (k).

\[
O_{\{i,j\}} = \log(\sigma(K_{ij})) + \sum_{k \sim p_D} \mathbb{E}_{k \sim p_D} [\log(\sigma(-K_{ik}))]
\]

\[
\mathbb{E}_{k \sim p_D} [\log(\sigma(-K_{ik}))] = \sum_{k \sim p_D} \frac{\#k}{D} \log(\sigma(-K_{ik})) + \sum_{k \sim p_D, k \neq j} \frac{\#k}{D} \log(\sigma(-K_{ik}))
\]

Therefore,

\[
\mathbb{E}_{j \sim p_D} [\log(\sigma(-K_{ij}))] = \frac{\#j}{D} \log(\sigma(-K_{ij}))
\]
\[ O_{\{i,j\}} = \log(\sigma(K_{ij})) + \frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D} \log(\sigma(-K_{ij})) \]

Let \(\gamma K_{ij} = x\), then

\[ \nabla_x O_{\{i,j\}} = \sigma(-x) - \frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D} \sigma(x) \]

equating \(\nabla_x O_{\{i,j\}}\) to 0, we get:

\[ e^{2x} - \left( \frac{1}{\frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D}} - 1 \right) e^x - \left( \frac{1}{\frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D}} \right) = 0 \]

If we define \(y = e^x\), this equation becomes a quadratic equation of \(y\), which has two solutions, \(y = -1\) (which is invalid given the definition of \(y\)) and \(y = \frac{1}{\frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D}} \frac{|S|}{M \times (\#j)^{0.75}}\).

Substituting \(y\) with \(e^x\) and \(x\) with \(K_{ij}\) reveals:

\[ K_{ij} = \log \left( \frac{\#D \times |S|}{M \times (\#j)^{0.75}} \right) \]

Here \(|S| = \#(i, j)\) and \(M = \mu \#(i)\) i.e. \(\mu\) proportion of total number of times label vector \((i)\) appear with others.

\[ K_{ij} = \log \left( \frac{\#(i, j)(\#D)}{\#(i)(\#j)^{0.75}} \right) - \log(\mu) \]

\[ K_{ij} = \log \left( \frac{P(i, j)}{P(i)P(j)} \right) - \log(\mu) \]

Here \(P(i,j), P(i)\) and \(P(j)\) represent probability of co-occurrences of \(\{i, j\}\), occurrence of \(i\) and occurrence of \(j\) respectively.

Therefore,

\[ K_{ij} = PMI_{ij} - \log(\mu) = \log(P(i|j)) - \log(\mu) \]

Note that \(PMI^+\) is inconsistent, therefore we used the sparse and consistent positive \(PMI(PPMI)\) metric, in which all negative values and nan are replaced by 0:

\[ PPMI_{ij} = \max(\text{PMI}_{ij}, 0) \]

Here, \(PMI\) is point wise mutual information and \(PPMI\) is positive point wise mutual information. Similarity of two \(\{i, j\}\) is more influenced by the positive neighbor they share than by the negative neighbor they share as \textbf{uninformative} i.e. 0 value. Hence, SGNS objective can be cast into a weighted matrix factorization problem, seeking the optimal lower \(d\)-dimensional factorization of the matrix \(SPPMI\) under a metric which pays more for deviations on frequent \(\#(i, j)\) pairs than deviations on infrequent ones.

Using a similar derivation, it can be shown that noise-contrastive estimation (NCE) which is alternative to (SGNS) can be cast as factorization of (shifted) log-conditional-probability matrix

\[ K_{ij} = \log \left( \frac{\#(i,j)}{\#j} \right) - \log(\mu) \]