Distributional Semantics meets Multi-Label Learning

Vivek Gupta[1,3], **Rahul Wadbude**[2], **Nagarajan Natarajan**[3], **Harish Karnick**[2], **Prateek Jain**[3], **Piyush Rai**[2]
[1]School of Computing, University of Utah, [2]Computer Science Department, IIT Kanpur
[3]Microsoft Research Lab, Bangalore
vgupta@cs.utah.edu, rahulwadbude2@gmail.com, nagarajn@microsoft.com, hk@cse.iitk.ac.in
prajain@microsoft.com, piyush@cse.iitk.ac.in

Abstract

We present a label embedding based approach to large-scale multi-label learning, drawing inspiration from ideas rooted in distributional semantics, specifically the Skip Gram Negative Sampling (SGNS) approach, widely used to learn word embeddings. Besides leading to a highly scalable model for multi-label learning, our approach highlights interesting connections between label embedding methods commonly used for multi-label learning and paragraph embedding methods commonly used for learning representations of text data. The framework easily extends to incorporating auxiliary information such as label-label correlations; this is crucial especially when many training instances are only partially annotated. To facilitate end-to-end learning, we develop a joint learning algorithm that can learn the embeddings as well as a regression model that predicts these embeddings for the new input to be annotated, via efficient gradient based methods. We demonstrate the effectiveness of our approach through an extensive set of experiments on a variety of benchmark datasets, and show that the proposed models perform favorably as compared to state-of-the-art methods for large-scale multi-label learning. We have released the source code along with the paper[1].

Introduction

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 *annotate* each instance using a subset of labels from a potentially very large label vocabulary. Nowadays, multi-label learning problems can even have label vocabulary consisting of millions of labels, popularly known as extreme multi-label learning (Jain, Prabhu, and Varma 2016; Bhatia et al. 2015; Babbar and Schölkopf 2017; Prabhu and Varma 2014).

The key challenges in multi-label learning, especially extreme multi-label learning, include: a) training instances may have a large fraction of labels missing, and b) the labels are often heavy-tailed (Bhatia et al. 2015; Jain, Prabhu, and Varma 2016) and predicting labels in the tail becomes significantly hard due to the lack of training data. For these reasons, and the sheer scale of data, traditional multi-label classifiers are rendered impractical. State-of-the-art approaches to extreme multi-label learning fall broadly under two classes: 1) embedding based methods, e.g., LEML (Yu et al. 2014), WSABIE (Weston, Bengio, and Usunier 2010), SLEEC (Bhatia et al. 2015), PD-SPARSE (Yen et al. 2016), PPDSPARSE (Yen et al. 2017), and 2) tree-based methods, e.g., FASTXML (Prabhu and Varma 2014) , PFASTREXML (Jain, Prabhu, and Varma 2016), LEM (Tagami 2017b). The first class of approaches are generally scalable and work by embedding the high-dimensional label vectors to a lower-dimensional space and learning a regressor in that space. In most cases, these methods 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 (Bhatia et al. 2015; Jain, Prabhu, and Varma 2016), 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) for multi-label learning try to move away from rigid structural assumptions (Prabhu and Varma 2014; Jain, Prabhu, and Varma 2016), and have been demonstrated to work very well especially on the tail labels.

In this paper, we propose a label embedding based approach for multi-label learning, leveraging word embedding techniques (Mikolov et al. 2013), which have found resounding success in NLP tasks. We show that by learning rich `word2vec` style embedding for instances (and labels), we can: a) achieve competitive multi-label prediction accuracies which are comparable to state-of-the-art label embedding approaches, such as SLEEC (Bhatia et al. 2015); b) cope with missing labels, by incorporating auxiliary information in the form of label-label co-occurrences, c) can jointly learn embedding and perform regression, and d) learn faster than label embedding methods (SLEEC). We also extend our model to *joint* learning of label embeddings and input-to-embedding regressors, which performs comparatively to the state-of-the-art in multi-label learning algorithms like ANNEXML (Tagami 2017a), DISMEC (Babbar and Schölkopf 2017) and PPDSPARSE (Yen et al. 2017) on most of the datasets. Our learning algo-

[1] `https://bit.ly/31AXZOD`

rithms admit significantly faster implementation as compared to other embedding based approaches. The distinguishing aspect of our work is that it draws inspiration from distributional semantics approaches (Mikolov et al. 2013; Le and Mikolov 2014), widely used for learning non-linear representations of text data for NLP tasks, such as understanding word and document semantics, classifying documents, etc. In future, we can try to generalize the method with other NLP techniques used for representing text documents such as LSTMs (Hochreiter and Schmidhuber 1997), RNNs (Jain and Medsker 1999) and skip-thought vector (Kiros et al. 2015) for embedding instance label vector. Our main contributions are summarized below.

1. We establish a connection between multi-label learning using label embedding methods and the problem of learning distributional semantics in text data analysis. We leverage this connection to design a novel objective function for multi-label learning that can be solved efficiently using matrix factorization.
2. Unlike most existing multi-label learning methods, our method easily extends to leverage label co-occurrence information while learning the embeddings; this is especially appealing when many training instances might have incomplete annotations.
3. Our models have much faster training times as compared to state-of-art label embedding methods for extreme multi-label learning, while being competitive in terms of label prediction accuracies. We demonstrate this on several moderate-sized as well as very large-scale multi-label benchmark datasets.
4. We show improvement in performance by joint learning of embedding and regressors through a novel objective. Our jointly optimized objective is competitive with respect to state-of-the-art methods like ANNEXML (Tagami 2017a), DISMEC (Babbar and Schölkopf 2017) and PPDSPARSE (Yen et al. 2017).
5. Our joint objective is flexible to incorporate online parameter update using online stochastic gradient decent update algorithms. This can even help us in learning with limited labeled data. However, this is beyond the scope of this paper and is left as future work.

Problem Formulation and Background

We assume that we are given a set of training instances, e.g., documents in BoW/tf-idf representation, $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ and the associated label/tag vectors $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$, where $\mathbf{y}_i \in \{0,1\}^L$. In many cases, 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 external source such as the wikipedia corpus). The goal in multi-label learning is to learn a vector-valued function $\mathbf{f} : \mathbf{x} \mapsto \mathbf{s}$, where $\mathbf{s} \in \mathbb{R}^L$ scores the labels (which can be used to *rank* the labels according to their relevance).

Embedding-based approaches typically model $\mathbf{f}$ as a composite function $\mathbf{h}(\mathbf{g}(\mathbf{x}))$ where, $\mathbf{g} : \mathbb{R}^d \to \mathbb{R}^{d'}$ and $\mathbf{h} : \mathbb{R}^{d'} \to \mathbb{R}^L$. For example, assuming both $\mathbf{g}$ and $\mathbf{h}$ as linear transformations, one obtains the formulation proposed by (Yu et al. 2014). The functions $\mathbf{g}$ and $\mathbf{h}$ 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 $\mathbf{g}$ is a linear transformation, but $\mathbf{h}$ is non-linear, and in particular, based on k-nearest neighbors in the embedded feature space.

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

$$\mathbf{h}\left(\mathbf{z}; \{\mathbf{z}_i, \mathbf{y}_i\}_{i=1}^n\right) = \frac{1}{|\mathcal{N}_k|} \sum_{i \in \mathcal{N}_k} \mathbf{y}_i, \tag{1}$$

where $\mathbf{z}_i = \mathbf{g}(\mathbf{x}_i)$ and $\mathcal{N}_k$ denotes the $k-$nearest neighbor training instances of $\mathbf{z}$ 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 $\mathbf{x}_i$; for each cluster of instances Q^τ, a different linear embedding $\mathbf{g}_\tau$, denoted by V^τ, is learnt.

Algorithm 1 Prediction Algorithm

Input: Test point: $\mathbf{x}$, no. of nearest neighbors k, no. of desired labels p.
1. Q_τ : partition closest to $\mathbf{x}$.
2. $\mathbf{z} \leftarrow V^\tau \mathbf{x}$.
3. $\mathcal{N}_k \leftarrow k$ nearest neighbors of $\mathbf{z}$ in the embedded instances of Q_τ.
4. $\mathbf{s} = \mathbf{h}(\mathbf{z}; \{\mathbf{z}_i, \mathbf{y}_i\}_{i \in Q_\tau})$ where $\mathbf{h}$ is defined in 1
return top p scoring labels according to $\mathbf{s}$.

In this work, we focus on learning algorithms for the functions $\mathbf{g}$ and $\mathbf{h}$, inspired by their successes in natural language processing in the context of learning distributional semantics (Mikolov et al. 2013; Levy and Goldberg 2014). In particular, we use techniques for inferring word-vector embeddings for learning the function $\mathbf{h}$ using a) training label vectors $\mathbf{y}_i$, and b) label-label correlations $C \in \mathbb{R}^{L \times L}$.

Word embeddings are desired in natural language processing in order to understand semantic relationships between words, also 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 appearing in similar *contexts* (i.e. adjacent words in documents) 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 instances, using their label vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$. SLEEC also uses nearest neighbors

in the space of label vectors $\mathbf{y}_i$ in order to learn the embeddings. However, we show in experiments that `word2vec` based embeddings are richer and help improve the prediction performance significantly, especially when there is a lot of missing labels. In the subsequent section, we discuss our algorithms for learning the embeddings and the training phase of multi-label learning.

Learning Instance and Label Embeddings

There are multiple algorithms in the literature for learning word embeddings (Mikolov et al. 2013; 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:

$$P(\text{Observing } (w, w')) = \sigma(\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle) = \frac{1}{1 + \exp(\langle -\mathbf{z}_w, \mathbf{z}_{w'} \rangle)} \tag{2}$$

To promote dissimilar words to be further apart, negative sampling is used, wherein randomly sampled negative examples (w, w'') are used. Overall objective favors $\mathbf{z}_w, \mathbf{z}_{w'}, \mathbf{z}_{w''}$ that maximize the log likelihood of observing $(w.w')$, for $w' \in c$, and 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:

$$\arg\max_{\mathbf{z}} \sum_{w} \Bigg(\sum_{w':(w',w)} \log\left(\sigma(\langle \mathbf{z}_w, \mathbf{z}_{w'} \rangle)\right) + \frac{n_-}{\#w} \sum_{w''} \log\left(\sigma(-\langle \mathbf{z}_w, \mathbf{z}_{w''} \rangle)\right) \Bigg), \tag{3}$$

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 show how an SGNS-like approach can be designed 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 (3) 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:

$$\arg\max_{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n} \sum_{i=1}^{n} \Bigg(\sum_{j:\mathcal{N}_k(\mathbf{y}_i)} \log\left(\sigma(\langle \mathbf{z}_i, \mathbf{z}_j \rangle)\right) + \frac{n_-}{n} \sum_{j'} \log\left(\sigma(-\langle \mathbf{z}_i, \mathbf{z}_{j'} \rangle)\right) \Bigg), \tag{4}$$

Note that $\mathcal{N}_k(\mathbf{y}_i)$ denotes the k-nearest neighborhood of i^{th}f instance in the space of *label vectors* [2] or instance embedding. After learning label embeddings $\mathbf{z}_i$, we can learn the function $\mathbf{g} : \mathbf{x} \to \mathbf{z}$ by regressing $\mathbf{x}$ onto $\mathbf{z}$, as in SLEEC. Solving (4) 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 speed up using the key observation by (Levy and Goldberg 2014).

Levy et. al. (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 $M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$.

$$\mathsf{PMI}_{ij}(M) = \log\left(\frac{M_{ij} * |M|}{\sum_k M_{(i,k)} * \sum_k M_{(k,j)}}\right)$$

$$\mathsf{SPPMI}_{ij}(M) = \max(\mathsf{PMI}_{ij}(M) - \log(k), 0) \tag{5}$$

Here, PMI is the point-wise mutual information matrix of M and $|M|$ denotes the sum of all elements in M. Solving the problem (4) 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 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 $\widehat{M} := \mathsf{SPPMI}(M)$ in (5), where $M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$.
2. Let $U, S, V = \text{svd}(\widehat{M})$, 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 4 using a neural network model (Mikolov et al. 2013); we refer to this `word2vec` method for learning embeddings in Algorithm 2 as EXMLDS2.

[2] Alternately, one can consider the neighborhood in the d-dimensional feature space $\mathbf{x}_i$; however, we perform clustering in this space for speed up, and therefore the label vectors are likely to preserve more discriminative information within clusters.

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 together 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 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. 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 matrix:

$$\arg\max_{z,\bar{z}} \mathbb{O}_{z,\bar{z}} = \mu_1 \mathbb{O}^1_{\bar{z}} + \mu_2 \mathbb{O}^2_{z} + \mu_3 \mathbb{O}^3{}_{\{z,\bar{z}\}} \tag{6}$$

$$\mathbb{O}^1_{\bar{z}} = \sum_{i=1}^{L} \Bigg(\sum_{j:\mathcal{N}_k(C(i,:))} \log\left(\sigma(\langle \bar{\mathbf{z}}_i, \bar{\mathbf{z}}_j \rangle)\right) + \frac{n^1_-}{L} \sum_{j'} \log\left(\sigma(-\langle \bar{\mathbf{z}}_i, \bar{\mathbf{z}}_{j'} \rangle)\right) \Bigg), \tag{7}$$

$$\mathbb{O}^2_{z} = \sum_{i=1}^{n} \Bigg(\sum_{j:\mathcal{N}_k(M(i,:))} \log\left(\sigma(\langle \mathbf{z}_i, \mathbf{z}_j \rangle)\right) + \frac{n^2_-}{n} \sum_{j'} \log\left(\sigma(-\langle \mathbf{z}_i, \mathbf{z}_{j'} \rangle)\right) \Bigg), \tag{8}$$

$$\mathbb{O}^3_{\{z,\bar{z}\}} = \sum_{i=1}^{L} \Bigg(\sum_{j:y_{ij}=1} \log\left(\sigma(\langle \mathbf{z}_i, \bar{\mathbf{z}}_j \rangle)\right) + \frac{n^3_-}{L} \sum_{j'} \log\left(\sigma(-\langle \mathbf{z}_i, \bar{\mathbf{z}}_{j'} \rangle)\right) \Bigg) \tag{9}$$

Here, $\mathbf{z}_i$, $i = 1, 2, \ldots, n$ denote embeddings of instances while $\bar{\mathbf{z}}_i$, $i = 1, 2, \ldots, L$ denote embeddings of labels. $\mathcal{N}_k(M(i,:))$ denotes the k-nearest neighborhood of i^{th} instance in the space of label vectors. $\mathcal{N}_k(C(i,:))$ denotes the k-nearest neighborhood of i^{th} label in the space of *labels*. Here, M defines instance-instance correlation i.e. $M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$ and C is the label-label correlation matrix. Clearly, (8) above is identical to (4). $\mathbb{O}^1_{\bar{\mathbf{z}}}$ tries to embed labels $\bar{\mathbf{z}}_i$ in a vector space, where correlated labels are closer; $\mathbb{O}^2_{\mathbf{z}}$ tries to embed instances $\mathbf{z}_i$ in such a vector space, where correlated instances are closer; and finally, $\mathbb{O}^3_{\{\mathbf{z},\bar{\mathbf{z}}\}}$ tries to embed labels and instances in a common space where labels occurring in the i^{th} instance are close to embedded instance.

Overall the combined objective $\mathbb{O}_{\{\mathbf{z},\bar{\mathbf{z}}\}}$ promotes learning a common embedding space where correlated labels, correlated instances and observed labels for a given instance occur closely. Here μ_1,μ_2 and μ_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 (6) 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 $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n$ and C (label-label correlation matrix) and objective weighting μ_1,μ_2 and μ_3.

1. Compute $\widehat{A} := \mathsf{SPPMI}(A)$ in (5); write

$$A = \begin{pmatrix} \mu_2 M & \mu_3 Y \\ \mu_3 Y^T & \mu_1 C \end{pmatrix},$$

$M_{ij} = \langle \mathbf{y}_i, \mathbf{y}_j \rangle$, Y: label matrix with $\mathbf{y}_i$ as rows.

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

3. Compute the embedding matrix $Z = US^{0.5}$; write

$$Z = \begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix},$$

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, where X is the matrix with $\mathbf{x}_i$ as rows.

return V, Z

Algorithm 4 Prediction Algorithm with Label Correlations (EXMLDS3 prediction).

Input: Test point: $\mathbf{x}$, no. of nearest neighbors k, no. of desired labels p, V, embeddings Z_1 and Z_2.

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

3. Get score $s_2 = Z_2 V x$

4. Get final score $s = \frac{s_1}{\|s_1\|} + \frac{s_2}{\|s_2\|}$.

return top p scoring labels according to $\mathbf{s}$.

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 $\mathbf{z}_i$ corresponds to the i^{th} row of Z_1, and $\bar{\mathbf{z}}_j$ corresponds to the j^{th} row of Z_2. We refer the Algorithm 3 based on the combined learning objective (6) as EXMLDS3.

Joint Embedding and Regression: We extended the SGNS objective to directly learn the regression matrix (V) through gradient descent, resulting in EXMLDS4. Compared to previous approaches, EXMLDS4 jointly learn the embeddings and regressors, by directly learning the matrix (V) in one step. For detail see Algorithm 5. Let,

$$K_{ij} = \langle \frac{\mathbf{z}_i}{\|\mathbf{z}_i\|} \frac{\mathbf{z}_j}{\|\mathbf{z}_j\|} \rangle = \frac{\langle \mathbf{z}_i^T \mathbf{z}_j \rangle}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|}$$

$$\mathbf{z}_i = V\mathbf{x}_i, \text{where V} \in \mathbb{R}^{d' \times d}$$

Objective 4 for i^{th} instance at step t:

$$\mathbb{O}_i = \sum_{j:\mathcal{N}_k(\mathbf{y}_i)} \log\big(\sigma(K_{ij})\big) + \frac{n_-}{n} \sum_{j'} \log\big(\sigma(-K_{ij'})\big), \tag{10}$$

Gradient of objective 4 w.r.t to V i.e. $\nabla_V \mathbb{O}_i$ is :

$$\nabla_V \mathbb{O}_i = \sum_{j:\mathcal{N}_k(\mathbf{y}_i)} \sigma(-K_{ij}) \nabla_V K_{ij} - \frac{n_-}{n} \sum_{j'} \sigma(K_{ij'}) \nabla_V K_{ij'} \tag{11}$$

where $\nabla_V K_{ij}$ is given by,

$$\nabla_V K_{ij} = -ab^3 c \mathbf{z}_i (\mathbf{x}_i)^T - abc^3 \mathbf{z}_j (\mathbf{x}_j)^T + bc(\mathbf{z}_i \mathbf{x}_j^T + \mathbf{z}_j \mathbf{x}_i^T) \tag{12}$$

where $a = \mathbf{z}_i^T \mathbf{z}_j, b = \frac{1}{\|\mathbf{z}_i\|}, c = \frac{1}{\|\mathbf{z}_j\|}$

Joint Learning in SLEEC SLEEC paper Section 2.1 (Optimization) stated that the joint objective function (Equation 3) is non-convex as well as non-differentiable and extremely challenging to optimize. Even in the disjoint simplified problem (Equation 4) is a low-rank matrix completion problem and is known to be NP-hard. The embedding learning (Equation 5) is non-smooth due to L1 penalty constraint. In EXMLDS4our SGNS objective is non-convex but easily differentiable, we don't have any L1 penalty constraint.

Experiments

We conduct experiments on commonly used benchmark datasets from the extreme multi-label classification repository provided by the authors of (Prabhu and Varma 2014; Bhatia et al. 2015) [3]; these datasets are pre-processed, and have prescribed train-test splits. 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. Source code will be made available to public later. We evaluate three models (a) EXMLDS1 i.e. Algorithm 2 based on fast PPMI matrix factorization for learning label embeddings as described earlier, (b) EXMLDS2 based on optimizing the objective (4) as described earlier, using neural network (Mikolov et al. 2013) (c) EXMLDS3 i.e. Algorithm 3 based on combined objective (6).

[3] Datasets and Benchmark :https://bit.ly/2IDtQbS

Algorithm 5 Learning joint instance embeddings and regression via gradient decent (EXMLDS4).

Input. Training data $(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n$. no. of nearest neighbors k, Gaussian initialize regression matrix V
while $t = i \neq n$ **do**
1. Compute $\mathbb{O}_i$ (10) and gradient $\nabla_V \mathbb{O}_i$ (11);
2. Perform SGD update for V,

$$\mathbb{V} \leftarrow \mathbb{V} + \eta \nabla_V \mathbb{O}_i$$

3. Update η using Adam.
end while
return V

Compared methods We compare our algorithms with the following baselines. 1. SLEEC (Bhatia et al. 2015), which was shown to outperform all other embedding baselines on the benchmark datasets. 2. LEML (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 LEML which uses item. features[4]. We refer to the latter method that uses label correlations as LEML-IMC. 3. FASTXML (Prabhu and Varma 2014), a tree-based method. 4. PD-SPARSE (Yen et al. 2016), recently proposed embedding based method. 5. PPDSPARSE (Yen et al. 2017), parallel, and distributed fast version of PD-SPARSE (Yen et al. 2016). 6. PFASTREXML (Jain, Prabhu, and Varma 2016) is an extension of FASTXML; it was shown to outperform all other tree-based baselines on benchmark datasets. 7. DISMEC (Babbar and Schölkopf 2017) is recently proposed scalable implementation of the ONE-VS-ALL method. 8. DXML (Zhang et al. 2017) is a recent deep learning solution for multi-label learning. 9. XML-CNN (Liu et al. 2017) is a recent deep learning solution for multi-label learning, specifically for text classification. 10 .ANNEXML (Tagami 2017a) is recent learning solution for multi-label learning, which uses DSSM (Yih et al. 2011) as objective. 11. PLT (Jasinska et al. 2016) is recent is a tree-based classifier that directly maximizes the F-measure 12. ONE-VS-ALL (Rifkin and Klautau 2004) is traditional one vs all multi-label classifier. We report all baseline results from the extreme classification repository [5], where they have been curated; note that all the baseline use the same train-test split for benchmarking.

[4] LEML IMC: https://goo.gl/jdGbDP1 [5] Baseline: https://bit.ly/2IDtQbS

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 EXMLDS1and 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, 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 hyper-parameters 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 4; 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. We obtain slightly poor performance than SLEEC on some datasets because we used unweighted SVD to factorize the PPMI matrix instead of a weighted matrix factorization as suggested by (Levy and Goldberg 2014) to obtain embedding from PPMI matrix. We can further improve our performance by using a weighted SVD algorithm, however it might increase training time of algorithm significantly.

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 $\mathbf{y}_i$. We use the Google `word2vec` code[6] 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 1. 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 SLEECon Bibtex dataset. In the large dataset, Delicious-200K, EXMLDS1 trains 5x faster than SLEEC.

Table 1: Comparing training times (in seconds) of different methods

Method -	Bibtex -	Delicious	Eurlex	Media mill	Deli 200K
EXMLDS1	**23**	**259**	**580.9**	**1200**	**1937**
EXMLDS2	143.19	781.94	880.64	12000	13000
SLEEC	313	1351	4660	8912	10000

[6] `https://goo.gl/D8aEgF`

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. Our method EXMLDS3 can also 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 learning 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 μ_1 to $\mathbb{O}^1$ during training in Algorithm 3. For prediction, We use Algorithm 4 which takes missing labels into account. We compare the performance of EXMLDS3with SLEEC, LEML and LEML-IMCin Table 4. Note that while SLEEC and LEML methods do not incorporate such auxiliary information, LEML-IMC does. In particular, we use the spectral embedding based features i.e. SVD of YY^T and take all the singular vectors corresponding to non-zero singular values as label features. It can be observed that on all three datasets, EXMLDS3 performs significantly better by huge margins. In particular, the lift over LEML-IMC is significant, even though both the methods use the same information. This serves to demonstrate the strength of our approach.

Joint Learning: Our method EXMLDS4 can jointly learn the embedding and regression. To implement joint learning, we modified the existing code of state-of-the-art embedding based extreme classification approach AnnexML (Tagami 2017a) [7], by replacing the `DSSM` [8] training objective by `word2vec` objective, while keeping cosine similarity, partitioning algorithm, and approximate nearest prediction algorithm same. For efficient training of rare label, we keep the coefficient ratio of negative to positive samples as *20:1*, while training for all datasets. We used the same hyper-parameters i.e. embedding size as 50, number of learner for each cluster as 15, number of nearest neighbor as 10, number of embedding and partitioning iteration both 100, gamma as 1, label normalization as true, number of threads as 32. We obtain state-of-the-art result i.e. similar or better in comparison to DISMEC (Babbar and Schölkopf 2017), PPDSPARSE (Yen et al. 2017) and ANNEXML (Tagami 2017b) and SLEEC (Bhatia et al. 2015) on all large datasets, see table 3 for details results. Our training and prediction time for EXMLDS4 was similar to that of ANNEXML.

Analysis and Discussion

Although SLEEC performs slightly better on some datasets, our EXMLDS1 model is much faster than SLEEC in training on large datasets as shown in Table 1. The performance of our model EXMLDS3 in Table 4 when a significant fraction of labels missing is considerably better than SLEEC and other

[7] Code: `https://bit.ly/2wB4nLu` [8] DSSM `https://bit.ly/2Ih8duw`

Table 2: 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 1. Also note the performance of our model in Table 4 when a significant fraction of labels are missing is considerably better than SLEEC

Dataset	Prec@k	Propose	Embedding Based			Sparsity	Tree Based XML		Others	
		EXMLDS1	DXML	SLEEC	LEML	PD-SPARSE	PFASTRE	FAST	1-vs-All	DISMEC
Bibtex	P@1	63.38	63.69	**65.29**	62.54	61.29	63.46	63.42	62.62	-
	P@3	38.00	37.63	**39.60**	38.41	35.82	39.22	39.23	39.09	-
	P@5	27.64	27.71	**28.63**	28.21	25.74	29.14	28.86	28.79	-
Delicious	P@1	67.94	67.57	**68.10**	65.67	51.82	67.13	69.61	65.01	-
	P@3	61.35	61.15	61.78	60.55	44.18	62.33	**64.12**	58.88	-
	P@5	56.3	56.7	57.34	56.08	38.95	58.62	**59.27**	53.28	-
Eurlex	P@1	77.55	77.13	79.52	63.40	76.43	75.45	71.36	79.89	**82.40**
	P@3	64.18	64.21	64.27	50.35	60.37	62.70	59.90	66.01	**68.50**
	P@5	52.51	52.31	52.32	41.28	49.72	52.51	50.39	53.80	**57.70**
Mediamill	P@1	87.49	**88.71**	87.37	84.01	81.86	83.98	84.22	83.57	-
	P@3	**72.62**	71.65	72.6	67.20	62.52	67.37	67.33	65.60	-
	P@5	**58.46**	56.81	58.39	52.80	45.11	53.02	53.04	48.57	-
Delicious-200K	P@1	46.07	44.13	**47.50**	40.73	34.37	41.72	43.07	-	45.50
	P@3	41.15	39.88	**42.00**	37.71	29.48	37.83	38.66	-	38.70
	P@5	38.57	37.20	**39.20**	35.84	27.04	35.58	36.19	-	35.50

Table 3: Comparing prediction performance of state-of-the-art embedding based methods (− mean unavailable results) on large dataset with joint learning

Dataset	Prec@k	Proposed	Embedding Based			Sparsity Based	
		EXMLDS4	ANNEXML	SLEEC	XML-CNN	PD-SPARSE	PPDSPARSE
AmazonCat-13K	P@1	93.05	93.55	90.53	**95.06**	89.31	92.72
	P@3	79.18	78.38	76.33	**79.86**	74.03	78.14
	P@5	64.54	63.32	61.52	**63.91**	60.11	63.41
Wiki10K-31K	P@1	**86.82**	86.50	85.88	84.06	77.71	-
	P@3	**74.30**	74.28	72.98	73.96	65.73	-
	P@5	63.68	**64.19**	62.70	64.11	55.39	-
Delicious-200K	P@1	47.70	46.66	**47.85**	-	34.37	45.05
	P@3	41.22	40.79	**42.21**	-	29.48	38.34
	P@5	37.98	37.64	**39.43**	-	27.04	34.90
WikiLSHTC-325K	P@1	62.15	63.36	54.83	-	61.26	**64.13**
	P@3	39.58	40.66	33.42	-	39.48	**42.10**
	P@5	29.10	29.79	23.85	-	28.79	**31.14**
Wikipedia-500K	P@1	62.27	**63.86**	58.39	59.85	-	-
	P@3	41.43	**42.69**	37.88	39.28	-	-
	P@5	31.42	**32.37**	28.21	29.31	-	-
Amazon-670K	P@1	41.47	42.08	35.05	-	**44.70**	43.04
	P@3	36.35	36.65	31.25	-	**39.70**	38.24
	P@5	32.43	32.76	28.56	-	**36.10**	34.94

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

Data	P@k	Exmlds3	SLEEC	LEML	LEML-IMC
Bibtex	P@1	**48.51**	30.5	35.98	41.23
	P@3	**28.43**	14.9	21.02	25.25
	P@5	**20.7**	9.81	15.50	18.56
Eurlex	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
rcv1v2	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

label embedding based baselines such as LEMLand LEML-IMC(uses auxiliary information). Without joint objective our models EXMLDS1 and EXMLDS2 have poor performance on few large datasets. However, with the joint objective learning our model EXMLDS4 obtained comparable and most closest performance to the best known state-of-the-art performance obtain by different multi-label baseline on different datasets. In Table 3, it should be noted that both EXMLDS4 andANNEXML is closest to the best performance and outperform most of the other compared baselines on all dataset simultaneously.

Conclusions and Future Work

The this paper we establish a connection between `word2vec` in NLP with multi-label learning in XML. The benefit leap by the connection is efficient and fast training, easy handling of the missing label using external auxiliary label-label correlation information and easily perform joint embedding learning and regression. Our proposed objective can be optimized efficiently by SPPMI matrix factorization and can also incorporates side information, which is effective in handling missing labels. 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 accuracies. Joint training model learning learn the regression V which has a large number of entries $d \times L$, as L is the number of labels (millions). One can use the sparsity in X (we used VX), to design better training over the joint model.

References

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*, 721–729. ACM.

Bhatia, K.; Jain, H.; Kar, P.; Varma, M.; and Jain, P. 2015. Sparse local embeddings for extreme multi-label classification. In *NIPS*, 730–738.

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

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Jain, L. C., and Medsker, L. R. 1999. *Recurrent Neural Networks: Design and Applications*. Boca Raton, FL, USA: CRC Press, Inc., 1st edition.

Jain, H.; Prabhu, Y.; and Varma, M. 2016. Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications. In *22nd ACM SIGKDD*, 935–944. ACM.

Jasinska, K.; Dembczynski, K.; Busa-Fekete, R.; Pfannschmidt, K.; Klerx, T.; and Hullermeier, E. 2016. Extreme f-measure maximization using sparse probability estimates. In *ICML*, 1435–1444.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *NIPS*, 3294–3302.

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

Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*, 2177–2185.

Liu, J.; Chang, W.-C.; Wu, Y.; and Yang, Y. 2017. Deep learning for extreme multi-label text classification. In *40th International ACM SIGIR Conference*, SIGIR '17, 115–124. New York, NY, USA: ACM.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

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

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*, 263–272. ACM.

Rifkin, R., and Klautau, A. 2004. In defense of one-vs-all classification. *Journal of machine learning research* 5(Jan):101–141.

Tagami, Y. 2017a. Annexml: Approximate nearest neighbor search for extreme multi-label classification. In *Proceedings of the 23rd ACM SIGKDD*, KDD '17, 455–464. New York, NY, USA: ACM.

Tagami, Y. 2017b. Learning extreme multi-label tree-classifier via nearest neighbor graph partitioning. In *26th International Conference on World Wide Web Companion*, WWW '17 Companion, 845–846.

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, 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 *ICML*, 3069–3077.

Yen, I. E.; Huang, X.; Dai, W.; Ravikumar, P.; Dhillon, I.; and Xing, E. 2017. Ppdsparse: A parallel primal-dual sparse method for extreme classification. In *Proceedings of the 23rd ACM SIGKDD*, KDD '17, 545–553. New York, NY, USA: ACM.

Yih, W.-t.; Toutanova, K.; Platt, J. C.; and Meek, C. 2011. Learning discriminative projections for text similarity measures. In *Fifteenth Conference on Computational Natural Language Learning*, 247–256. Association for Computational Linguistics.

Yu, H.-F.; Jain, P.; Kar, P.; and Dhillon, I. 2014. Large-scale multi-label learning with missing labels. In *ICML*, 593–601.

Zhang, W.; Wang, L.; Yan, J.; Wang, X.; and Zha, H. 2017. Deep extreme multi-label learning. *arXiv preprint arXiv:1704.03718*.

Appendix: Distributional Semantics meets Multi-Label Learning

SGNS as Implicit SPPMI factorization

The SGNS (Mikolov et al. 2013) objective is as follows:

$$\mathbb{O}_i = \sum_{j \in S_i} \log(\sigma(K_{ij})) + \sum_{k \sim P_D}^{M} \mathbb{E}_{k \sim P_D}[\log(\sigma(-K_{ik}))] \tag{13}$$

where, $P_D = \frac{(\#k)^{0.75}}{\#D}$, 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).

$$\mathbb{O}_{\{i,j\}} = \log(\sigma(K_{ij})) + \sum_{k \sim P_D}^{\frac{M}{|S|}} \mathbb{E}_{k \sim P_D}[\log(\sigma(-K_{ik}))] \tag{14}$$

$$\mathbb{E}_{k \sim P_D}[\log(\sigma(-K_{ik}))] = \sum_{k \sim P_D} \frac{(\#k)^{0.75}}{\#D} \log(\sigma(-K_{ik})) \tag{15}$$

$$\begin{aligned} \mathbb{E}_{k \sim P_D}[\log(\sigma(-K_{ik}))] = & \frac{(\#j)^{0.75}}{\#D} \log(\sigma(-K_{ij})) \\ & + \sum_{k \sim P_D \& k \neq j} \frac{(\#k)^{0.75}}{\#D} \log(\sigma(-K_{ik})) \end{aligned} \tag{16}$$

Therefore,

$$\mathbb{E}_{j \sim P_D}[\log(\sigma(-K_{ij}))] = \frac{(\#j)^{0.75}}{\#D} \log(\sigma(-K_{ij})) \tag{17}$$

$$\mathbb{O}_{\{i,j\}} = \log(\sigma(K_{ij})) + \frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D} \log(\sigma(-K_{ij})) \tag{18}$$

Let γK_{ij} = x, then

$$\nabla_x \mathbb{O}_{\{i,j\}} = \sigma(-x) - \frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D} \sigma(x) \tag{19}$$

equating $\nabla_x \mathbb{J}_{\{i,j\}}$ to 0, we get :

$$e^{2x} - \left(\frac{1}{\frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D}} - 1 \right) e^x - \left(\frac{1}{\frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D}} \right) = 0 \tag{20}$$

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

$$y = \frac{1}{\frac{M}{|S|} \frac{(\#j)^{0.75}}{\#D}} = \frac{\#D * |S|}{M * (\#j)^{0.75}} \tag{21}$$

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

$$K_{ij} = \log\left(\frac{\#D * |S|}{M * (\#j)^{0.75}} \right) \tag{22}$$

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

$$K_{ij} = \log\left(\frac{\#(i,j)(\#D)}{\#(i)(\#j)^{0.75}} \right) - log(\mu) \tag{23}$$

$$K_{ij} = \log\left(\frac{P(i,j)}{P(i)P(j)} \right) - log(\mu) \tag{24}$$

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,

$$K_{ij} = \mathsf{PMI}_{ij} - \log(\mu) = \log(P(i|j)) - log(\mu) \tag{25}$$

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:

$$\mathsf{PPMI}_{ij} = \max(\mathsf{PMI}_{ij}, 0) \tag{26}$$

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 *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

$$K_{ij} = \log\left(\frac{\#(i,j)}{(\#j)} \right) - \log(\mu) \tag{27}$$

Gradient Computation

Gradient of objective 4 w.r.t to V i.e. $\nabla_V \mathbb{O}_i$ is :

$$\begin{aligned} \nabla_V \mathbb{O}_i = & \sum_{j : \mathcal{N}_k(\mathbf{y}_i)} \sigma(-K_{ij}) \nabla_V K_{ij} \\ & - \frac{n_-}{n} \sum_{j'} \sigma(K_{ij'}) \nabla_V K_{ij'} \end{aligned} \tag{28}$$

Table 5: Dataset Statistics

Dataset	Feature	Label	Train	Test
Bibtex (**?**; Prabhu and Varma 2014)	1836	159	4880	2515
Delicious (**?**; Prabhu and Varma 2014)	500	983	12920	3185
EURLex-4K (**?**; Prabhu and Varma 2014)	5000	3993	15539	3809
rcv1v2 (**?**; Prabhu and Varma 2014)	47236	101	3000	3000
Delicious-200K (**?**; Bhatia et al. 2015)	782585	205443	196606	100095
MediaMill (**?**; Bhatia et al. 2015)	120	101	30993	12914
Wiki10-31K (Bhatia et al. 2015; **?**)	101938	30938	14146	6616
AmazonCat-13K (**?**)	203882	13330	1186239	306782
WikiLSHTC-325 (Prabhu and Varma 2014; Bhatia et al. 2015)	1617899	325056	1778351	587084
Wikipedia-500K	2381304	501070	1813391	783743
Amazon-670K (**?**; Bhatia et al. 2015)	135909	670091	490449	153025

$$\begin{aligned}\nabla_V \langle \mathbf{z}_i \mathbf{z}_j \rangle &= \nabla_V (V\mathbf{x}_i)\mathbf{z}_j + \nabla_V (V\mathbf{x}_j)\mathbf{z}_i \\ &= \langle \mathbf{z}_i \mathbf{x}_j^T \rangle + \langle \mathbf{z}_j \mathbf{x}_i^T \rangle = V(\langle \mathbf{x}_i \mathbf{x}_j^T + \mathbf{x}_j \mathbf{x}_i^T \rangle)\end{aligned} \tag{29}$$

$$\begin{aligned}\nabla_V \frac{1}{\|\mathbf{z}_i\|} = \nabla_V \mathbf{z}_i \mathbf{z}_i^T{}^{\frac{-1}{2}} &= \frac{-1}{2}\mathbf{z}_i \mathbf{z}_i^T{}^{\frac{-3}{2}} \nabla_V \mathbf{z}_i \mathbf{z}_i^T \\ &= \frac{-1}{2}\mathbf{z}_i \mathbf{z}_i^T{}^{\frac{-3}{2}} \mathbf{x}_i \mathbf{z}_i^T\end{aligned} \tag{30}$$

$$\begin{aligned}\nabla_V \frac{1}{\|\mathbf{z}_j\|} = \nabla_V \mathbf{z}_j \mathbf{z}_j^T{}^{\frac{-1}{2}} &= \frac{-1}{2}\mathbf{z}_j \mathbf{z}_j^T{}^{\frac{-3}{2}} \nabla_V \mathbf{z}_j \mathbf{z}_j^T \\ &= \frac{-1}{2}\mathbf{z}_j \mathbf{z}_j^T{}^{\frac{-3}{2}} \mathbf{x}_j \mathbf{z}_j^T\end{aligned} \tag{31}$$

Let,

$$a = \mathbf{z}_i^T \mathbf{z}_j, b = \frac{1}{\|\mathbf{z}_i\|}, c = \frac{1}{\|\mathbf{z}_j\|} \tag{32}$$

Thus, we have,

$$\begin{aligned}\nabla_V K_{ij} = -ab^3 c\mathbf{z}_i(\mathbf{x}_i)^T - abc^3 \mathbf{z}_j(\mathbf{x}_j)^T \\ + bc(\mathbf{z}_i \mathbf{x}_j^T + \mathbf{z}_j \mathbf{x}_i^T)\end{aligned} \tag{33}$$

Similarity to graph embedding:

Graph embedding algorithm Grarep(**?**) and DNGR(**?**), which aim to learn embeddings for each node of graph are based on similar idea of weighted matrix factorization of shifted PPMImatrix of input adjacency matrix. These node embedding outperforms has outperform previous state-of-the-art in the task of community classification/detection.

Dataset Statistics:

We have provided the details datasets statistics in Table 5.