ISSN 2007-9737

EDGE2VEC: Edge Representations for Large-Scale Scalable Hierarchical Learning

Mohammad Golam Sohrab[1], Toru Nakata[1], Makoto Miwa[2], Yutaka Sasaki[2]

[1] Artificial Intelligence Research Center,
National Institute of Advanced Industrial Science and Technology, Tokyo,
Japan

[2] Toyota Technological Institute, Nagoya,
Japan

{sohrab.mohammad; toru-nakata}@aist.go.jp,
{makoto-miwa; yutaka.sasaki}@toyota-ti.ac.jp

Abstract. In present front-line of Big Data, prediction tasks over the nodes and edges in complex deep architecture needs a careful representation of features by assigning hundreds of thousands, or even millions of labels and samples for information access system, especially for hierarchical extreme multi-label classification. We introduce $edge2vec$, an edge representations framework for learning discrete and continuous features of edges in deep architecture. In $edge2vec$, we learn a mapping of edges associated with nodes where random samples are augmented by statistical and semantic representations of words and documents. We argue that infusing semantic representations of features for edges by exploiting $word2vec$ and $para2vec$ is the key to learning richer representations for exploring target nodes or labels in the hierarchy. Moreover, we design and implement a balanced stochastic dual coordinate ascent (DCA)-based support vector machine for speeding up training. We introduce a global decision-based top-down walks instead of random walks to predict the most likelihood labels in the deep architecture. We judge the efficiency of $edge2vec$ over the existing state-of-the-art techniques on extreme multi-label hierarchical as well as flat classification tasks. The empirical results show that $edge2vec$ is very promising and computationally very efficient in fast learning and predicting tasks. In deep learning workbench, $edge2vec$ represents a new direction for statistical and semantic representations of features in task-independent networks.

Keywords. Hierarchical text classification, multi-label learning, indexing, extreme classification, tree-structured class hierarchy, DAG-structured class hierarchy, DG-structured class hierarchy.

1 Introduction

In machine learning (ML) platforms the current front-line of "Big Data" deals with millions of training and test documents as well as hundreds of thousands, or even millions of labels. Therefore, scalable learning and optimization in the deep architecture are the key to deal with such large-scale date-sets. Although strong ML methods such as Support Vector Machines (SVMs) [2, 4, 20] have been successfully applied to text classification (TC).

In general, ML-based TC can be categorized into two classification tasks: a flat classification (FC) [5, 13, 14, 17] by referring to standard binary or multi-class classification problems where parent-child relations are completely omitted. Second is the hierarchical classification (HC) [15, 16] – typically a tree, a directed acyclic graph (DAG), or a directed graph (DG) are incorporated, where the classes to be predicted are organized into a class hierarchy. A very large amount of research in TC, data mining (DM), and related researches have focused on FC problems. In contrast, many important real-world classification problems are naturally cast as HC problems. In

Big Data platform, the size of data is too large to implement suitable classifiers. Moreover, it is difficult to label new data into predefined categories since text and labels are growing exponentially. Therefore, it is still an open and more challenging problem to design and implement such a model that classify large-scale documents into large-scale hierarchically-structured categories accurately and efficiently.

To generate text to vector, statistical vector space model (VSM) is a common fashion for learning and prediction tasks. But in complex hierarchical domain where many categories require extremely large training sets to achieve higher accuracy. To build efficient scalable learning model where a mini-batch is considered by random sampling from large training sets of a certain node. In the prediction stage, many or completely missing features are appeared in vector space model (VSM) because of sparsity, that can not precisely predict the leaf categories for a candidate sample.

Here, semantic learning based on unsupervised technique to learn continuous feature representations may give a shed to predict the candidate samples. In the above directions, this paper presents edge representations learning ($edge2vec$) where each edge in the hierarchy a statistical feature vector and a semantic feature vector based on word and paragraph representations from unlabeled data are incorporated. In $edge2vec$, we first learn word representations based on word-word co-occurrence from unlabeled data to generate word vectors.

We then infuse continuous weights into features along with discrete weights in the VSM. In addition, We then learn paragraph vectors and infuse continuous weights of paragraph vectors into features. The new input vector for deep or hierarchical learning, we call hierarchical semantically augmented statistical vector space model (hSAS-VSM). This study makes the following major contributions with introducing the $edge2vec$ based hSAS-VSM approach to address large-scale classification task:

— The proposed edge representations learning ($edge2vec$) consists of discrete and continuous feature learning using word vectors ($word2vec$) and paragraph vectors ($para2vec$).

— The hSAS-VSM enriches the existing statistical-VSM using semantic knowledge for words and documents.

— The proposed $edge2vec$ follows the inductive learning and deductive classification for very large-scale dataset. The training and test speed for learning and classification are fast which makes the system scalable.

— Infusing embedding features are useful to enrich the categorical performance for large-scale dataset.

— We introduce a balanced stochastic dual coordinate ascent for linear support vector machines for efficient learning and to adjust the positive-negative samples imbalance in a certain node in the hierarchy.

— The proposed edge-based learning is not only reduce the computational cost but also can significantly improve the classification score. Therefore, edge-based learning is a prominent approach for hierarchical classification.

2 Related Work

TC is a typical multi-class single- and multi-label classification problem. Platt [11] proposed a faster training of SVM using sequential minimal optimization (SMO) that breaking a very large quadratic programming optimization problem into a series of smallest possible problems as an inner loop in each outer iteration. The approach is generally 1200 and 15 times faster for linear and non-linear SVMs respectively.

Studies to solve multi-class multi-label classification have been summarized in [18], in three smaller data sets with maximum labels of 27 in compare to current front-line of multi-label classification task. Sohrab [14, 16] proposed a semantically augmented statistical vector space model (SAS-VSM) by introducing word embedding into feature for single- and multi-label text classification (TC). In this work, the SAS-VSM is introduced in FC and outperformed in compare to

VSM. There have been many studies that use local context in HTC [1, 7, 10]. Chakrabarti et al. [1] proposed a Naive-Bayes document classification system that follows hierarchy edges from the root node. Koller et al. [7] applied Bayesian networks to a hierarchical document classification. In LSHTC3, the arthur system [21] successfully applied meta-classifiers to the large-scale hierarchical text classification (LSHTC) task.

Meta-classifiers can also be regarded as a sort of pruning. The system employed Liblinear, Max Entropy classifier, and SVM^{light}. The meta-classifier with SVM^{light} achieved 43.81% on the aspect of accuracy; however relatively slow in compare to Liblinear and Max Entropy on the aspect of efficiency. Lee [8] proposed a Multi-Stage Rocchio classification (MSRC) based on similarity between test documents and label's centroids for large-scale datasets.

The system used greedy search algorithm in the predicted label set and then compare similarities between test documents and two centroid to check whether more labels are needed or not. The MSRC achieved 39.74%, 43.26%, and 67.83% in terms of accuracy, LBMiF, and HF respectively for Wikipedia medium dataset. On the aspect of efficiency the system is much faster than baseline such as K-Nearest Neighbor when the expected number of labels per document are less.

3 Our Approach: Edge Representations Learning

Edge representations learning ($edge2vec$) is formulated with different words or terms and document representations in natural language processing (NLP). It consists of discrete and continuous feature learning from words and documents using word and paragraph vectors respectively. Consider a hierarchy $H\,(N, E)$, where N is the set of nodes and E is the set of edges. A hierarchy H is a collection of superiors or parents and subordinates or children categories. In $edge2vec$, a certain edge e_i in the hierarchy H is augmented by hSAS-VSM. Each edge e_i is an $edge2vec$ optimized-based learning model by propagating a set of samples associated with a certain node n_i. The SAS-VSM is the inspiration which leads to generate $edge2vec$ for a certain edge in the hierarchy by infusing paragraph vectors along with word vectors into features for hSAS-VSM.

The sample augmentation process of $edge2vec$ consist of three sub-tasks. First is the VSM-based $edge2vec$ learning i.e. $edge2vec^{vsm}$, second is the hSAS-VSM-based learning using word vectors i.e. $edge2vec^{vsm,w2v}$. Finally we generate $edge2vec^{vsm,w2v,p2v}$ which incorporated with VSM, word and paragraph vectors in the sample augmentation process.

3.1 Hierarchical SAS-VSM: hSAS-VSM

Our approach for learning word vectors into features along with existing supervised VSM is inspired by SAS-VSM. The inspiration is that how to infuse continuous word and paragraph vectors along with discrete weights into features for hSAS-VSM in very large-scale extreme multi-label HTC. In the hSAS-VSM, an augmented document space $D = \{d_1, d_2, ...d_n\}$ can be denoted as: $\vec{x}(d) = \left(\vec{x}^{vsm}(d), \vec{x}^{w2v}(d), \vec{x}^{p2v}(d)\right)$, where $\vec{x}(d)$ is a hSAS-VSM augmented feature vector for a document d, $\vec{x}^{vsm}(d) = (x_1^{vsm}(d), \dots, x_M^{vsm}(d))$ is a statistical feature vector can be defined as:

$$\vec{x}^{vsm}(d) = \begin{cases} f(t_i), & \text{if } t_i \in d \\ 0, & \text{otherwise} \end{cases}, \qquad (1)$$

where $f(t_i)$ is a term weighting function representing any weighting approach for term t_i. To avoid excessive effects of large feature values in the $\vec{x}^{vsm}(d) = (x_1^{vsm}(d), \dots, x_M^{vsm}(d))$, we normalize the weight for document d as: $\bar{t}_i = \frac{t_i}{t_i+1}$. $\vec{x}^{w2v}(d)$ and $\vec{x}^{p2v}(d)$ are semantically augmented feature vectors using word and paragraph vectors respectively.

3.1.1 Word Vector in *h* SAS-VSM

For fast and accurate learning in deep architecture, first we introduce a simple solution for infusing word vectors into features in large-scale hierarchical text classification (LSHTC). Suppose that there are M words in matrix V and each word is mapped

to p-dimensions, then to compute $\vec{x}^{w2v}(d)$ based on word vectors as:

$$\vec{x}^{w2v}(d) = \vec{x}^{vsm}(d)V. \tag{2}$$

Word embedding vector V is defined as:

$$V = \begin{bmatrix} \vec{v}_1 \\ \vdots \\ \vec{v}_M \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ v_{M1} & v_{M2} & \cdots & v_{Mp} \end{bmatrix}.$$

Each row $\vec{v}_i$ represents the word embedding vector for word or term t_i. The new generated augmented features for document d are incorporated with discrete and continuous weights that get a larger weight than existing normalized statistical vectors. We therefore scale the $\vec{x}^{w2v}(d)$ as:

$$x_i^{w2v}(d) = \frac{x_i^{w2v}(d)}{Q^{vsm}(d)}, \tag{3}$$

where for a document d, $Q^{vsm}(d) = \vec{x}^T\vec{x}$ and can be computed from $\vec{x}^{vsm}(d)$.

3.1.2 Paragraph Vector in *h* SAS-VSM

In TC, document D consists of a sequence of documents $\{d_1, d_2, ...d_n\}$ in the corpus. In paragraph or document vectors, matrix $\vec{P}$ is an $N \times q$ matrix with N documents and each document is mapped to q-dimensions continuous-valued vector. In (4), we can then generate the paragraph vector-based $\vec{x}^{p2v}(d)$ for a certain document d as,

$$\vec{x}^{p2v}(d) = \vec{x}^{w2v}(d)P. \tag{4}$$

Paragraph vector P is defined as,

$$P = \begin{bmatrix} \vec{p}_1 \\ \vdots \\ \vec{p}_N \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1q} \\ \vdots & \vdots & \ddots & \vdots \\ p_{N1} & p_{N2} & \cdots & p_{Nq} \end{bmatrix}.$$

Each row $\vec{p}_i$ denotes the embedding vector for document d_i. In the $edge2vec$ process, two properties are of main concern: inductive learning and deductive classification.

3.2 *EDGE2VEC* in Inductive Learning

The inductive learning induces a set of observed instances or samples from specific bottom categories to general top categories in the category hierarchy. During inductive learning, the $edge2vec$ follows the sample augmentation using hSAS-VSM. We then perform the bottom-up propagation a sampling strategy that assign all the augmented samples from a specific or leaf category to more general top category in the hierarchy. Finally, we train each edge in the hierarchy based on top-down walks.

3.2.1 Bottom-up Propagation

Since only leaf categories are assigned to data, first we propagate training samples from the leaf level to the root in the category hierarchy. Fig. 1a illustrates toy example of document propagation in a hierarchy consisting of ten categories R-I. In this figure, sample x_1 is assigned to category F, x_2 to F and G, x_3 to H, x_4 and x_5 to I. Let us look at the case of x_2 assigned to G. x_2 of G is propagated to both categories C and D. Then, x_2 of C is propagated to A and then to R. When x_2 is propagated from D to A afterwards, to avoid redundant propagation, the propagation of x_2 (originally from G via D) terminates at A, even if A is a parent category. We employ a recursive algorithm to perform the bottom-up propagation of the samples.

3.2.2 Edge-based Learning: Top-down Walks

Based on the propagation of training samples, we train the classifiers for each edge in the hierarchy where each edge is coupled with a binary class classifier using the one-against-the-rest approach. In Fig. 1b at node C, during the bottom-up propagation where x_1 and x_2 are assigned. Since edge-based learning is in concern, therefore model W_{CF} is trained in the hierarchy as to classify x_1 and x_2 to F; whereas model W_{CG} is trained as to classify x_2 to G but not x_2 to F. In large-scale hierarchical learning, each node is propagated with hundreds of thousands, or even millions of samples, where positive-negative samples imbalance occur

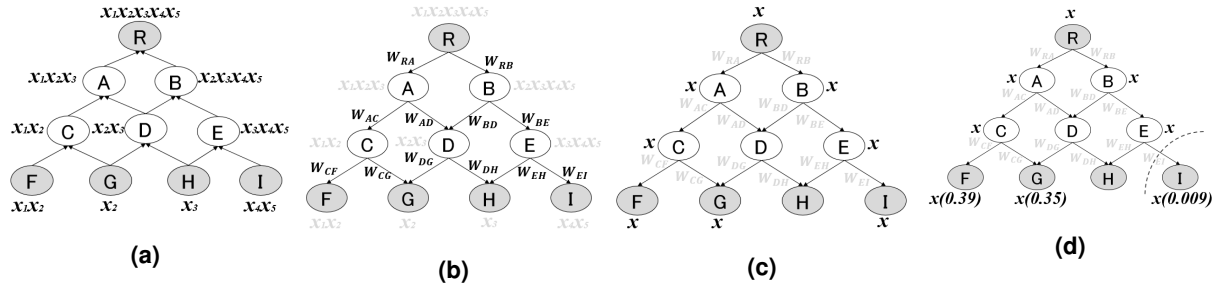

Fig. 1. Hierarchy: (a) Bottom–up propagation (b)Edge–based Training (c) Top–down Classification (d) Global Pruning

repeatedly. For efficient learning and to adjust the effect of positive-negative samples imbalance in a certain node in the hierarchy, we present a balanced stochastic dual coordinate ascent for linear support vector machines (BS-DCASVM) with L1-loss function. For randomly chosen $(\vec{x}_i, y_i)$, BS-DCASVM updates the weight vector as:

$$\vec{w}_{pc} \leftarrow \vec{w}_{pc} + (\alpha_i - \alpha_i')y_i\vec{x}_i, \tag{5}$$

where $\vec{w_{pc}}$ is a weight vector of certain edge e_i in the hierarchy. The optimization process starts from an initial point $\alpha \in \mathbb{R}^l$ and generates a sequence of vectors $\{\alpha^k\}_k^\infty$. We refer to the process from α^k to α^{k+1} as an outer iteration. In each outer iteration we have l inner iterations, so that sequentially $\alpha_1, \alpha_2, ..., \alpha_l$ are updated. Each outer iteration thus generates vectors $\alpha^{k,i} \in \mathbb{R}^l$. For updating $\alpha^{k,i}$ to $\alpha^{k,i+1}$, must find the optimal solution as:

$$\alpha_i^{k,i+1} = min\left(max\left(\alpha_i^{k,i} - \frac{\nabla_i f\left(\vec{\alpha}^{k,i}\right)}{\vec{x}_i^T\vec{x}_i}, 0\right), C\right), \tag{6}$$

where $C > 0$ is a regularization parameter. $\nabla_i f$ is the ith component of the gradient $\nabla_i f$. To evaluate $\nabla_i f\left(\alpha^{k,i}\right)$:

$$\nabla_i f\left(\vec{\alpha}\right) = y_i\vec{w}_{pc}^T\vec{x}_i - 1. \tag{7}$$

In (6), we move to index $i+1$ with updating $\alpha_i^{k,i}$, if and only if the projected gradient $\nabla_i^P f\left(\alpha^{k,i}\right) \neq 0$ and satisfy the following conditions:

$$\nabla_i^P f\left(\vec{\alpha}\right) = \begin{cases} \nabla_i f\left(\alpha\right) & \text{if } 0 < \alpha_i < C, \\ \min\left(0, \nabla_i f\left(\alpha\right)\right) & \text{if } \alpha_i = 0, \\ \max\left(0, \nabla_i f\left(\alpha\right)\right) & \text{if } \alpha_i = C. \end{cases} \tag{8}$$

In (5), α_i' is the current value and α_i is the value after the updating. In the inner iterations of a certain node, in each iteration we maintain the updates of a weight vector $\vec{w}$ in a balanced stochastic way, by randomly chosen one from positive samples $(\vec{x}_i, y_i \in +1)$ and in next iteration the other from negative samples $(\vec{x}_i, y_i \in -1)$.

3.3 *EDGE2VEC* in Deductive Classification

The deductive classification deduces a set of unlabeled samples from general top categories to more specific bottom categories in the hierarchy. In deductive classification, $edge2vec$ follows the unlabeled samples augmentation as stated in 2.1, decision-based top-down walks for classification with global adjustments, and global pruning.

3.3.1 Global Decision-based Top-down Walks

Fig. 1c illustrates top-down classification of test data $\vec{x}$. First, $\vec{x}$ is classified to A and B, based on the decision by $W_{RA}(\vec{x})$ and $W_{RB}(\vec{x})$, respectively. The decision is made by:

$$W_{pc}(\vec{x}) = \vec{w}_{pc}.\vec{x} + b_{pc}. \tag{9}$$

To adjust the effect of positive-negative samples imbalance, we set a bias β. When $W_{pc}(\vec{x}) > \beta$, $\vec{x}$ is classified from parent category p to child category c. When both $W_{RA}(\vec{x}) > \beta$ and $W_{RB}(\vec{x}) > \beta$ are satisfied, $\vec{x}$ is classified into both A and B. Note that the standard bias term b_{pc} is automatically tuned for each edge in the training stage. After the classification, we prune unlikely classes for query sample x. We define a confidence score and set the global threshold θ for it. When x reaches a leaf

node n, the confidence score $c_\alpha(x, n)$ is calculated as follows:

$$c_\alpha(x, n) = \prod_{(n_1,n_2)\in E} \sigma_\alpha(G_{n_1 n_2}(x))), \qquad (10)$$

where E is a set of edges that x has followed in the path from the root to the leaf n. The output value of a classifier is converted to $[0, 1]$ range by $\sigma_\alpha(x) = \frac{1}{1+\exp(-\alpha x)}$. α is set to 2 from the cross validation stage. When multiple nodes are assigned to x, if $c(x, n) < \theta$, the assignment of x to n is removed. Fig. 1d illustrates the global pruning.

4 Evaluation

In this section, we provide empirical evidence for the effectiveness of our proposed $edge2vec$ in deep architecture. We employ official LSHTC evaluation metrics [19] and evaluate our systems on LSHTC evaluation site[1] because the gold standard labels for the test data is not publicly available. Given documents D, correct labels Y_i, and predicted labels Z_i, the metrics are as follows:

— Accuracy(Acc):
$1/|D| \sum_{i\in D} |Y_i \cap Z_i| / (|Y_i \cup Z_i|)$.

— Example-based F1 measure (EBF):
$1/|D| \sum_{i\in D} 2|Y_i \cap Z_i| / (|Y_i| + |Z_i|)$.

— Label-based Macro-average F1 (LBMaF):
Standard multi-label Macro-F1.

— Label-based Micro-average F1 (LBMiF):
Standard multi-label Micro-F1.

— Hierarchical F1 measure (HF):
The example-based F1-measure counting ancestors of true and predicted categories.

[1] http://lshtc.iit.demokritos.gr/

4.1 Base Algorithms

We employ sofia-ml[2] for the experiments with Pagasos, SGD-SVM, Passive Aggressive (PA) [3], Relaxed Online Margin Algorithm (ROMMA) [9], and Logistic regression (logreg). The term frequency (TF), TF.IDF [13, 17], and TF.IDF.ICS$_\delta$F [13, 17] are defined as:

$$f_{TF}(t_i, d) = tf_{(t_i,d)}, \qquad (11)$$

$$f_{TF.IDF}(t_i, d) = tf_{(t_i,d)} \times \left(1 + \log \frac{D}{\#t_i}\right), \qquad (12)$$

$$f_{TF.IDF.ICS_\delta F}(t_i, d, c_k) = tf_{(t_i,d)} \times \left(1 + \log \frac{D}{\#t_i}\right) \times \left(1 + \log \frac{C}{CS_\delta(t_i)}\right), \qquad (13)$$

where in (11)-(13), $tf(t_i, d)$ is the number of occurrences of term t_i in document d, D denotes the total number of documents in the training corpus, $\#t_i$ is the number of documents in the training corpus in which term t_i occurs at least once, $D/\#t_i$ is the inverse document frequency (IDF) of term t_i, C denotes the total number of predefined categories in the training corpus, $c(t_i)$ is the number of categories in the training corpus in which term t_i occurs at least once, and $\frac{C}{CS_\delta(t_i)}$ is the inverse class space density frequency (ICS$_\delta$F) of term t_i. Please refer to [13, 14] for more details.

4.1.1 Word and Paragraph Vectors

In unsupervised learning, the statistics of word co-occurrences in a corpus is the primary source for learning word representations. The Word Vector ($word2vec$) [6] gives a shed on how meaning is generated and how the resulting word vectors might represent that meaning from the global corpus statistics. In this work, We consider the $word2vec$[3] for learning word representations from unlabeled data to generate word vectors. We construct a matrix of word-word co-occurrences count from unlabeled corpora. We set a context window, and use a context ten words to the left and ten words to the right.

[2] http://code.google.com/p/sofia-ml/
[3] http://code.google.com/p/word2vec/

Table 1. Flat (FC) vs Hierarchical Classification (HC) with BS-DCASVM

Name	C	α	β	θ	Acc	EBF	LBMaF	LBMiF	HF	CPD
	0.5	2	–0.5	0.0	13.69	21.85	15.12	18.01	38.21	12.30
FC	0.5	2	–0.5	0.70	40.11	45.64	20.22	45.95	68.15	1.16
	0.5	2	–0.5	0.0	26.91	35.98	24.18	30.43	56.59	5.96
HC	0.5	2	–0.5	0.39	44.47	49.63	26.58	49.75	70.83	1.54

***Note**: CPD refers to categories per document*

Table 2. Comparison of efficient learning algorithms (LA)

LA	C	α	β	θ	Acc	EBF	LBMaF	LBMiF	HF	CPD
BS-DCASVM	0.5	2	–0.5	0.39	44.47	49.68	26.58	49.71	70.83	1.52
Pegasos	0.5	2	–0.5	0.32	44.23	49.48	26.69	49.66	70.76	1.50
SGD-SVM	0.5	2	–0.5	0.32	44.19	49.38	26.41	49.57	70.72	1.51
PA	0.5	2	–0.5	0.49	40.05	45.12	25.50	45.27	66.73	1.50
ROMMA	0.5	2	–0.5	0.15	38.27	43.24	22.96	43.62	56.10	1.50
logreg	0.5	2	–0.5	0.14	36.90	42.35	15.44	42.71	66.88	1.52

***Note**: CPD refers to categories per document*

Table 3. Parameter sensitivity of $edge2vec$ with BS-DCASVM

C	α	β	θ	Acc	EBF	LBMaF	LBMiF	HF	CPD
1	2	0.0	0.39	44.41	49.54	27.69	49.80	70.68	1.56
1	2	0.0	0.40	44.40	49.47	27.49	49.75	70.67	1.53
0.5	2	–0.5	0.40	45.09	50.17	26.94	50.40	71.27	1.48
0.5	2	–0.5	0.41	45.11	50.28	27.34	50.50	71.27	1.46

Table 4. $edge2vec$ with embedding vectors (EVs) over the BS-DCASVM

Learning Approach	Term Weighting	EVs	θ	Acc	EBF	LBMaF	LBMiF	HF
$edge2vec^{(vsm)}$	TF	-	0.39	44.47	49.68	26.58	49.71	70.83
$edge2vec^{(vsm)}$	TF.IDF	-	0.42	42.84	47.64	25.37	48.00	69.43
$edge2vec^{(vsm)}$	TF.IDF.ICS$_\delta$F	-	0.42	42.21	46.97	24.81	47.35	68.99
$edge2vec^{(vsm,w2v)}$		50	0.38	44.65	49.85	26.85	49.98	70.97
$edge2vec^{(vsm,w2v)}$	TF	100	0.39	44.89	50.06	26.95	50.25	71.13
$edge2vec^{(vsm,w2v)}$		200	0.38	44.72	49.92	26.81	49.99	70.99
$edge2vec^{(vsm,w2v,p2v)}$		50	0.39	44.76	49.82	26.29	49.95	71.02
$edge2vec^{(vsm,w2v,p2v)}$	TF	100	0.39	44.86	50.00	26.75	50.15	71.08
$edge2vec^{(vsm,w2v,p2v)}$		200	0.41	45.11	50.28	27.34	50.50	71.27

***Note**: In all cases, we use $C = 0.5$, $\alpha = 2$, and $\beta = -0.5$*

The unsupervised paragraph vector [12] is capable to learn continuous distributed vector representations of input sentences at any length: sentences, paragraphs, and documents. The paragraph vectors ($para2vec$) are generated from the available source code[4]. We run 25 iterations for all dimensional word and paragraph vectors. All parameters were left at default values in $word2vec$ and $para2vec$.

4.2 Experimental Environments

We evaluate the $edge2vec$ on standard multi-label HC for leaf nodes prediction through hidden or intermediate nodes in the hierarchy. We assess the training and classification time using a single Xeon 3.0GHz core with 396GB memory.

4.2.1 Dataset

To evaluate the performance of our proposed $edge2vec$, we compare our results with Wikipedia medium dataset (WMD) which considering as a benchmark for large-scale hierarchical classification.The WMD[5] consists of 456,866 training documents with 346,299 distinct features and 81,262 test documents with 132,296 distinct features. It contains 36,504 leaf categories and 50,312 categories in the hierarchy with maximum depth 12. The number of edges in the hierarchy are 65,333. The category hierarchies of WMD is in the form of DAG.

We learn the word and paragraph vectors using 456,866 and 2,365,436 training documents from WMD and Wikipedia large dataset[6] respectively. The vector representations of word and paragraph vectors are unsupervised learning that predicts the surrounding words in the paragraph. It is worth to mention that to learn paragraph vectors, a certain document is considered as one paragraph.

[4] https://github.com/hassyGo/paragraph-vector

[5] http://lshtc.iit.demokritos.gr/LSHTC3_DATASETS

[6] http://lshtc.iit.demokritos.gr/LSHTC4_GUIDELINES

4.3 Experimental Results

Table 1 shows the result with BS-DCASVM on flat vs hierarchical classification. For flat classification we achieve the best results in terms of accuracy while a set of parameters are set to $C = 0.5$, $\alpha = 2$, $\beta = -0.5$, and $\theta = 0.70$. In contrast, we achieve the best results while $C = 0.5$, $\alpha = 2$, $\beta = -0.5$, and $\theta = 0.39$ are set and the results show that the HC is outperformed in compare to FC.

In Table 2, we compare the efficient learning algorithms of edge-based approach in the hierarchical architecture. From the results, we can see that edge-based system with the BS-DCASVM, Pegasos, and SGD-SVM are performing better in compare to PA, ROMMA, and logreg. Since BS-DCASVM is performing best among the learners and for space limitation therefore rest of the experiments are conducted with BS-DCASVM for infusing word and paragraph vectors into features. Table 3 shows the effect of different parameters and how it improves the performance.

Table 4 shows how the added different dimensional embedding vectors (EV) allow $edge2vec$ to achieve high performances. In this table, the $edge2vec$ (word2vec, EV=100) denotes a document or sample is augmented by infusing 100 word vectors into features with existing statistical-VSM. The $edge2vec$ (word2vec, para2vec, EV=100) represents a sample is augmented by infusing each 100 word and paragraph vectors into features incorporated with existing statistical-VSM.

When $\beta = -0.5$ and θ = 0.41, we obtain the best performances 45.11%, 50.28%, 27.34%, 50.50% and 71.27% for Acc, EBF, LBMaF, LBMiF, and HF respectively. We summarize the results with compare to the top four systems participated in the LSHTC3 challenge in Table 5. Here $edge2vec$ consistently outperforms the top system and achieves significant improvement over the other systems. We achieve a gain of 1.29%, 0.91%, 0.60%, 1.11%, and 0.35% for Acc, EBF, LBMaF, LBMiF, and HF respectively over the top system.

Table 5. Comparison with top four LSHTC3 systems on WMD

Name	Acc	EBF	LBMaF	LBMiF	HF
$edge2vec^{(vsm)}$	44.47	49.68	26.58	49.71	70.83
$edge2vec^{(vsm,w2v)}$	44.89	50.06	26.95	50.25	71.13
$edge2vec^{(vsm,w2v,p2v)}$	45.11	50.28	27.34	50.50	71.27
arthur (Ist)	43.82	49.37	26.74	49.39	70.92
coolvegpuff (2nd)	42.91	48.24	25.07	47.79	68.92
TTI (3rd)	42.00	47.71	28.35	47.25	69.22
chrishan (4th)	41.17	47.68	24.54	41.87	67.66

Table 6. Efficiency with $edge2vec$ Approach

Learning Approach	Learning Algorithm	Learning CPU Time	Test CPU Time
$edge2vec^{(vsm)}$	BS-DCASVM	2400.47s (40.01m)	491.26s (8.19m)
$edge2vec^{(vsm,w2v)}$	BS-DCASVM	6864.43s (114.41m)	638.26s (10.64m)
$edge2vec^{(vsm,w2v,p2v)}$	BS-DCASVM	7139.61s (118.99m)	686.57s (11.44m)
$edge2vec^{(vsm)}$	Pegasos	6355.79s (105.92m)	512.69s (8.54m)
$edge2vec^{(vsm,w2v)}$	Pegasos	14831.50s (247.18m)	758.19s (12.64m)
$edge2vec^{(vsm,w2v)}$	Pegasos	19344.09s (322.40m)	963.39s (16.05m)

4.3.1 Parameter Sensitivity

The edge representations learning ($edge2vec$) involves a set of parameters in Table 3. We examine how different choices of parameters affect the performance of $edge2vec$ over the WMD. We measure the Acc, EBF, LBMaF, LBMiF, and HF score as a function of parameters C, α, β, and θ.

The performance of $edge2vec$ improves by changing the hyper-parameter C, β, and θ. We show the results with $\beta \in (0.0, -0.5)$ and varied $\theta \in (0.38, 0.39, 0.40, 0.41)$. $\beta = -0.5$ allows the data classifies into negative side that means some incorrect assignments are kept for candidate sets. However, most of the incorrect classifications are removed after-ward in the global pruning stage.

4.3.2 Scalability and Complexity

We learn $egde2vec$ for 65,333 edges in the hierarchy. Table 6 shows the training and test efficiency with different learning algorithms of $edge2vec$ approach. The total training time takes less than one hour including sample augmentation of 456,866 training samples, sampling, optimization, and writing 65,333 models. The optimization phase for a certain edge is made the learning more efficient using negative sampling.

For each outer iteration, we randomly select positive and negative samples in a balanced stochastic way from a mini-batch. Therefore learning in deep is compatible to handle any size of large-scale data efficiently and accurately. The normalize feature vectors are very effective in large-scale dataset by avoiding excessive effects of large feature value during learning and classification stage in the deep architecture.

In the test phase, global decision-based walks allow us to reach leaf categories efficiently. The total test time takes less than ten minutes for assigning 81,262 test data into 36,506 leaf categories through 50,312 intermediate or hidden categories using BS-DCASVM.

The complexity is to decide a category as the assignment for a query sample will be $O(\log n)$ with n leaf categories. Besides learning the best settings of parameter also reduce the additional training and test cost.

5 Discussion and Conclusion

We described edge representations that learns word and paragraph vectors for a certain edge to build a classification model. In $edge2vec$ with additional optimized features help to improve the prediction task. The good performance demonstrates the merits of $edge2vec$ in capturing the semantics of word and paragraph vectors. To achieve the best result in Table 1 for flat classification, where the threshold θ is set to 0.70. In multi-label classification, the higher threshold value of θ indicates many leaf categories are assigned for a candidate sample during the one-vs-rest approach which increase the computational cost.

It is noticeable that $edge2vec$-based HC is very efficient for learning and prediction tasks. It decreases the computational cost as well as increase the system performances. It is also noticeable that $edge2vec$ outperformed top-group systems in LSHTC3, w.r.t the most of evaluation metrics. We believe that, to handle extreme multi-label LSHTC problems, the results will make an useful contribution as an useful performance reference.

Although this work focus on large-scale hierarchical classification task, but the edge representations approach can be applied to link prediction, opinion mining, sentiment analysis, or related works in deep architecture. Continuous feature representations are the key of many deep learning algorithms, it would be interesting how $edge2vec$ can further contributes in deep learning workbenches. Our future work includes the development of much more efficient algorithms for large-scale datasets.

Acknowledgement

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

1. **Chakrabarti, S., Dom, B., Agrawal, R., & Raghavan, P. (1998).** Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *International Journal on Very Large data Bases*, Vol. 7, No. 3, pp. 163–178.
2. **Cortes, C. & Vapnik, V. (1995).** Support vector networks. *Journal of Machine Learning*, Vol. 20, pp. 273–297.
3. **Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006).** Online passive-aggressive algorithm. *Journal of Machine Learning Research*, Vol. 7, pp. 551–585.
4. **Dumais, S., Platt, J., & Heckerman, D. (1998).** Inductive learning algorithms and representations for text categorization. *Proceedings of the CIKM*, pp. 148–155.
5. **Fattah, M. A. & Sohrab, M. G. (2016).** Combined term weighting scheme using ffnn, ga, mr, sum, and average for text classification. *International Journal of Scientific and Engineering Research*, Vol. 7, No. 8, pp. 2031–2040.
6. **Jeffrey, P., Richard, S., & Christopher, D. M. (2014).** Glove: Global vectors for word representation. *Proceedings of the EMNLP*, Qatar, pp. 1532–1543.
7. **Koller, D. & Sahami, M. (1997).** Hierarchically classifying documents using very few words. *Proceedings of the ICML*, pp. 170–178.
8. **Lee, D. H. (2012).** Multi-stage rocchio classification for large-scale multi-labeled text data. *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*, Bristol.
9. **Li, Y. & Long, P. (2002).** The relaxed online maximum margin algorithm. *Journal of Machine Learning*, Vol. 46, pp. 1–3.
10. **McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. (1998).** Improving text classification by shrinkage in a hierarchy of classes. *Proceedings of the ICML*, pp. 359–367.
11. **Platt, J. (1998).** Fast training of support vector machines using sequential minimal optimization. *Advances in Kernel Methods: Support Vector Learning*, MIT.

12. **Quoc, L. & Mikolov, T. (2014).** Distributed representations of sentences and documents. *Proceedings of the ICML*, pp. 1188–1196.
13. **Ren, F. & Sohrab, M. G. (2013).** Class-indexing-based term weighting for automatic text classification. *Information Sciences*, Vol. 236, pp. 109–125.
14. **Sohrab, M. G., Miwa, M., & Sasaki, Y. (2015).** Centroid-means-embedding: An approach to infusing word embeddings into features for text classification. *Proceedings of the PAKDD, LNCS*, volume 9077, pp. 289–300.
15. **Sohrab, M. G., Miwa, M., & Sasaki, Y. (2016).** In-deductive and dag-tree approaches for large-scale extreme multi-label hierarchical text classification. *Journal of Polibits*, Vol. 54, pp. 61–70.
16. **Sohrab, M. G., Miwa, M., & Sasaki, Y. (2016).** Word embeddings in large-scale deep architecture learning. *The Association for Natural Language Processing*, volume 9077, pp. 625–628.
17. **Sohrab, M. G. & Ren, F. (2012).** The effectiveness of class-space-density in high and low-dimensional vector space for text classification. *Proceedings of the IEEE International Conference of CCIS*, China, pp. 2034–2042.
18. **Tsoumakes, G. & Katakis, I. (2007).** Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, Vol. 3, No. 3, pp. 1–13.
19. **Tsoumakes, G., Katakis, I., & Vlahavas, I. (2010).** Random k-labelsets for multi-label classification. *Proceeding of the Knowledge Discovery and Data Engineering*.
20. **Vapnik, V. (1995).** *The Nature of Statistical learning Theory*. Springer.
21. **Wang, K. L., Zhao, H., & Lu, B. L. (2014).** A meta-top down method for large scale hierarchical classification. *IEEE Transactions on Knowledge and Data Engineering*, volume 26, pp. 500–513.

Article received on 17/12/2016; accepted on 22/02/2017.
Corresponding author is Mohammad Golam Sohrab.