Word Embeddings in Large-Scale Deep Architecture Learning

Mohammad Golam Sohrab　Makoto Miwa　Yutaka Sasaki
Totoya Technological Institute

{sohrab, makoto-miwa, yutaka.sasaki}@toyota-ti.ac.jp

1 Introduction

Statistical Natural Language Processing (NLP) is now facing various "Big Data" challenges. In machine learning (ML)-based text classification (TC), the current front-line of "Big Data" is millions of training and test documents with hundred of thousands, or even millions, of labels and features.

In recent years, in addition to supervised statistical learning approaches, many studies have been carried out with showing success in adopting unsupervised methods for learning continuous word embedding [2, 3], from unlabeled texts. Word embedding (WE) features have actively studied on word analogies, word similarity, chunking, and named entity recognition. WE vectors are also used in TC [4, 7], but there remains the task of investigating how WE features can be infused into existing statistical features in multi-label hierarchical classification. This paper gives a shed on this issue to infusing WE into the large-scale deep architecture learning.

We apply a vector space model (VSM) in the deep architecture called Semantically-Augmented Statistical VSM (SAS-VSM) [7] for information access systems, especially for hierarchical text classification (HTC) [5, 6].

2 SAS-VSM

The SAS-VSM [7] that merges together statistical VSM and word-co-occurrence-based continuous embedding vectors. The architecture of SAS-VSM for a document space can be represented as:

SAS-VSM = Statistical-VSM || Semantic-VSM,

where || denotes the concatenation of two different VSMs. An SAS-VSM feature vector $\vec{x}(d)$ for a document d is,

$$\vec{x}(d) = \left(\vec{x}^{Stat}(d), \vec{x}^{Sem}(d)\right),$$

where $\vec{x}^{Stat}(d)$ is a statistical feature vector and $\vec{x}^{Sem}(d)$ is a semantic feature vector. In the above formulation of SAS-VSM, where the Statistical-VSM denotes term weightings based on discrete weights of terms for a corresponding document d. A Statistical-VSM vector is an $\vec{x}^{Stat}(d) = \left(x_1^{Stat}(d), \dots, x_M^{Stat}(d)\right)$. For term t_i, $\vec{x}^{Stat}(d)$ is as:

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

where $f(t_i)$ is a term weighting function representing any weighting approach for term t_i. In contrast, a Semantic-SVM vector is an $\vec{x}^{Sem}(d) = (x_1^{Sem}(d), \dots, x_N^{Sem}(d))$ for document d. Using a WE matrix $\mathbf{V}$, $\vec{x}^{Sem}(d)$ is defined as:

$$\vec{x}^{Sem}(d) = \vec{x}^{Stat}(d)\mathbf{V}. \quad (2)$$

WE matrix $\mathbf{V}$ is an $M \times N$ matrix with the vocabulary size M and the dimensionality N of WE vector. To infuse continuous WE vectors into existing discrete weights by rewriting Eqn. 1 as:

$$\vec{x}^{Stat'}(d) = \vec{x}^{Stat}(d) \times \overline{SCE}(d), \quad (3)$$

where $\overline{SCE}(d)$ is centroid-means-embedding,

$$\overline{SCE}(d) = \frac{1}{N}\sum_{i=1}^{N} SCE_i(d). \quad (4)$$

The SCE weight of a certain term t_i for a given document d can be represented as:

$$SCE(d) = \vec{A}(d)\mathbf{V} = (SCE_1(d), \dots, SCE_N(d)) \quad (5)$$

$$\vec{A}_i(d) = \begin{cases} 1, & \text{if } t_i \in d \\ 0, & \text{otherwise} \end{cases}, \quad (6)$$

where $\vec{A}$ is an M-dimensional row vector. For single-label classification, SAS-SVM [7] introduced Gaussian distribution based scaling function to scale each new generated weight. The hyper parameter $\vec{\lambda}$ for document d,

$$\lambda_i(d) = \frac{1}{\sqrt{2\pi\sigma_d^2}} \exp\left(-\frac{\left(x_i^{Sem}(d) - \mu_d\right)^2}{2\sigma_d^2}\right), \quad (7)$$

where the mean μ_d and standard deviation σ_d. Therefore the new generated vector from Eqn. 2 as:

$$\vec{x}^{Sem'}(d) = \vec{\lambda}(d) \circ \left(\vec{x}^{Stat'}(d)\vec{V}\right), \quad (8)$$

where $\circ$ denotes the element-wise multiplication of two row vectors.

3 WE in Deep Learning

In NLP, WE is the feature learning method where words from the vocabulary are mapped to continuous-valued vector of real numbers in low dimensional space. WE features can be useful as input to classification models or as additional features to enhance existing systems. The motivation for exploiting WE features in the deep architecture learning can be attributed to two main properties. First, in generating a more information-rich VSM, many documents in the training or test set for large-scale dataset may not share enough information to classify the test set properly. Second, there is a demand for document representation to integrate semantic VSM into statistical VSM for multi-label HTC.

3.1 SAS-VSM in HTC

We introduce a simple solution for SAS-VSM in large-scale HTC. In SAS-VSM in deep learning, we generate the Statistical- and Semantic-VSM based on Eqns. 1 and 2 respectively. In Eqn. 1, to avoid excessive effects of large feature values in the term weighting function $f(t_i)$, we normalized the feature value as:

$$\bar{t}_i = \frac{t_i}{t_i + 1}. \tag{9}$$

In Eqn. 2, we can see that new updated augmented features for document d are incorporated with discrete and continuous weights. It is also noticeable that, the new generated weight gets a larger weight than existing normalized statistical vectors. We therefore scale the semantic weight as:

$$x_i^{Sem}(d) = \frac{x_i^{Sem}(d)}{Q^{Stat}(d)}, \tag{10}$$

where for a certain document d, $Q^{Stat}(d) = \vec{x}^{Stat^T}\vec{x}^{Stat}$. The SAS-VSM in HTC is free from infusing continuous WE vectors into existing discrete weights as stated in Eqn. 3.

4 Edge-based Learning in Deep Architechture

4.1 Sampling: Bottom-up Manner

Since only leaf categories are assigned to data, we propagate training samples from the leaf level to the root in the class hierarchy. Fig. 1 illustrates propagation of documents in a hierarchy consisting of six categories A-F. In this figure sample x_1 is assigned to categories D and E as well as x_2 to D, and x_3 to F. Samples are propagated in a directed acyclic graph (DAG) in the bottom-up manner.

4.2 Learning in Deep with DCD-SVM: Top-down Manner

Based on the propagation of training samples in the hierarchy, we train classifiers for each edge of the hierarchy where each edge is coupled with a binary class classifier using the one-against-the-rest approach. In Fig. 2 at node B, during the bottom-up propagation where x_1 and x_2 are assigned to node B. Since edge-based learning is in concern, therefore model M_{BD} is trained in the hierarchy as to classify both x_1 and x_2 to D; whereas model M_{BE} is trained as to classify x_1 to E but not x_2 to E.

In large-scale hierarchical learning, each node is propagated with hundreds of thousands, or even millions of samples. Therefore, for efficient learning and to adjust the effect of positive-negative samples imbalance in a certain node in the hierarchy, we present a dual coordinate decent for linear support vector machines (DCD-SVM) [1] with L1-loss function. For randomly chosen $(\vec{x}_i, y_i)$, DCD-SVM updates the weight vector as,

$$\vec{w} \leftarrow \vec{w} + (\alpha_i - \alpha_i')y_i\vec{x}_i, \tag{11}$$

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

$$\alpha_i^{k,i+1} = min\left(max\left(\alpha_i^{k,i} - \frac{\nabla_i f\left(\vec{\alpha}^{k,i}\right)}{\vec{x}_i^T\vec{x}_i}, 0\right), C\right), \tag{12}$$

where $C > 0$ is a penalty parameter and set to 0.5 based on our previous results. $\nabla_i f$ is the ith component of the gradient ∇f. To evaluate $\nabla_i f\left(\vec{\alpha}^{k,i}\right)$,

$$\nabla_i f(\vec{\alpha}) = y_i\vec{w}^T\vec{x}_i - 1. \tag{13}$$

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

$$\nabla_i^P f(\vec{\alpha}) = \begin{cases} \nabla_i f(\vec{\alpha}) & \text{if } 0 < \alpha_i < C, \\ \min(0, \nabla_i f(\vec{\alpha})) & \text{if } \alpha_i = 0, \\ \max(0, \nabla_i f(\vec{\alpha})) & \text{if } \alpha_i = C. \end{cases} \tag{14}$$

In Eqn. 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 $(\vec{x}_i, y_i \in -1)$.

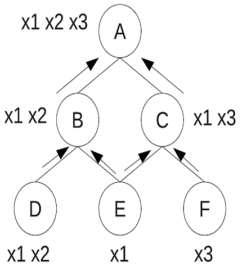

Figure 1: Bottom–up of training data.

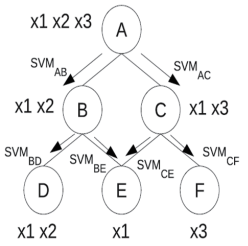

Figure 2: Top–down learning.

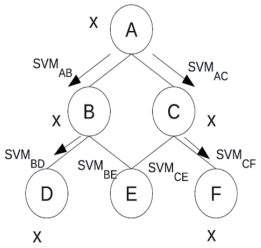

Figure 3: Top–down classification.

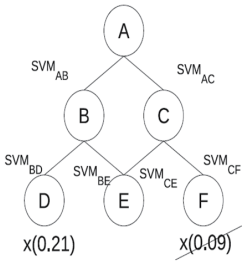

Figure 4: Global Pruning.

4.3 Classification: Top-down Manner

Fig. 3 illustrates top-down classification of test data $\vec{x}$. First, $\vec{x}$ is classified to B and C, based on the decision by $M_{AB}(\vec{x})$ and $M_{AC}(\vec{x})$, respectively. The decision is made by:

$$G_{pc}(\vec{x}) = \vec{w}_{pc}.\vec{x} + b_{pc}. \tag{15}$$

To adjust the effect of positive-negative sample imbalance, we set a bias β. When $G_{pc}(\vec{x}) > \beta$, $\vec{x}$ is classified from parent category p to child category c. When both $G_{AB}(\vec{x}) > \beta$ and $G_{AC}(\vec{x}) > \beta$ are satisfied, $\vec{x}$ is classified into both B and C. Note that the standard bias term b_{pc} is automatically tuned for each edge in the training stage.

Along with the HTC, we keep track of confidence scores of the classification. The output value of a classifier is converted to $[0, 1]$ range:

$$\sigma_\alpha(x) = \frac{1}{1 + \exp(-\alpha x)}. \tag{16}$$

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

$$c_\alpha(x, n) = \prod_{(n_1, n_2) \in E} \sigma_\alpha(G_{n_1 n_2}(x))), \tag{17}$$

where E is the set of edges that x has followed in the path from the root to leaf n. α is set to 2 from our previous study. After the classification of x throughout the hierarchy, we prune unlikely classes for x. We set the global threshold θ. When there are multiple nodes assigned to x, if $c(x, n) < \theta$, the assignment of x to n is removed. Figure 4 illustrates the global pruning.

5 Experiments

We target the standard HTC datasets prepared for the third edition of PASCAL challenge on Large-Scale Hierarchical Text Classification (LSHTC[1]) for Wikipedia medium dataset (WMD). We employ sofia-ml[2] package for the experiments with Pegasos, SGD-SVM, PA, ROMMA, logreg, and logreg-pegasos. We assessed the training and classification time using a single Xeon 3.0GHz core with 396GB memory. The gold standard labels for the test data of WMD are not publicly open. Therefore, we evaluate our systems based on LSHTC evaluation site[3]. Table 1 show the statistics of WMD. In this paper, to represent the Semantic-VSM, we consider a context prediction global vectors (GloVe) model [3] for learning WE. Table 2 shows the result of DCD-SVM with different dimensional embedding vectors (EV). we set $\beta = -0.5$ means that data classified into negative side to some extent are passed to the child node. This means that some incorrect assignments are kept in the candidate sets. However, most of the incorrect classification are removed afterwards during the pruning stage. When $\beta = -0.5$, $\theta = 0.39$, and EV=100, we obtained the best accuracy 44.92%. Table 3 shows the scores with several efficient ML algorithms with our edge-based classification approaches. Table 4 summarizes our result with compare to the top four systems using WMD. The result shows that our systems outperformed over the other systems that have been participated in the LSHTC3 challenge.

#Training	456,866
#Test	81,262
#Distinct features	346,299
#Categories(Hierarchy)	50,312
#Leaf categories	36,504
#Edges	65,333

Table 1: Statistics of Wikipedia medium data

[1] http://lshtc.iit.demokritos.gr/LSHTC3_CALL

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

[3] For convenience, we call the official evaluation matrices Accuracy(Acc), Example-based F1 measure (EBF), Label-based Macro-average F1 measure (LBMaF), Label-based Micro-average F1 measure (LBMiF), and Hierarchical F1 measure (HF).

Learning	C	β	θ	Acc	EBF	LBMaF	LBMiF	HF
DCD-SVM + EV=0	0.5	–0.5	0.39	44.52	49.68	26.64	49.78	70.86
DCD-SVM + EV=50	0.5	–0.5	0.38	44.65	49.85	26.85	49.98	70.97
DCD-SVM + EV=100	0.5	–0.5	0.39	**44.92**	**50.08**	**26.99**	**50.26**	**71.13**
DCD-SVM + EV=200	0.5	–0.5	0.38	44.72	49.92	26.81	49.99	71.00

Table 2: Results of DCD-SVM with different dimensional embedding vectors

Learning	C	β	θ	Acc	EBF	LBMaF	LBMiF	HF
DCD-SVM + EV=100	0.5	–0.5	0.39	**44.92**	**50.08**	**26.99**	**50.26**	**71.13**
Pegasos	0.5	–0.5	0.32	44.23	49.48	26.69	49.66	70.76
SGD-SVM	0.5	–0.5	0.32	44.19	49.38	26.41	49.57	70.72
PA	0.5	–0.5	0.49	40.05	45.12	25.50	45.27	66.73
ROMMA	0.5	–0.5	0.15	38.27	43.24	22.96	43.62	56.10
logreg	0.5	–0.3	0.14	36.90	42.35	15.44	42.71	66.88
logreg-pegasos	0.5	–0.5	0.14	36.89	42.55	16.44	42.96	66.82

Table 3: Comparison of efficient ML methods

Name	Acc	EBF	LBMaF	LBMiF	HF
DCD-SVM + EV=100	**44.92**	**50.08**	26.99	**50.26**	**71.13**
arthur (1st)	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 4: Comparison with top four LSHTC3 participants

6 Conclusion

We investigated the effectiveness of exploiting WE in the deep architecture learning for multi-label classification problem. The DCD-SVM and SAS-VSM as well as with different embedding vector sizes can significantly enhance the HTC task. In every cases w.r.t the evaluation metrics, the DCD-SVM with SAS-VSM outperformed LSHTC3's top-group systems. Possible ideas for future work would be to conduct experiments with SAS-VSM on very large scale multi-label HTC for Wikipedia large datasets[4].

Acknowledgments

This work has been partially supported by JSPS KAKENHI Grant Number 25330271.

References

[1] C. Hsieh, K. Chang, C. Lin, S. S. Keerthi, and S. Sundarara-jan: A Dual Coordinate Descent Method for Large-Scale Linear SVM, in Proc. of ICML-08, pp. 408415, 2008.

[2] E. H. Huang, R. Socher, D. M. Christopher, Y. N. Andrew: Improving Word Representations via Global Context and Multiple Word Prototypes. In: 50th Annual meeting of the ACL, pp. 873–882. Korea (2012).

[3] P. Jeffrey, S. Richard, D. M Christopher: Glove: Global Vectors for Word representation. In: 2014 Conference on EMNLP, pp. 1532–1543. Qatar (2014).

[4] L. Quoc, T. Mikolov: Distributed Representation of Sentences and Documents, In: 31th ICML, pp. 1188–1196 (2014).

[5] Y. Sasaki and D. Weissenbacher: TTIS System for the LSHTC3 Challenge, ECML/PKDD-2012 Discovery Challenge Workshop on LSHTC, Bristol, 2012.

[6] Y. Sasaki, M. G. Sohrab, M. Miwa: High-Performance Large-Scale Hierarchical Text Classification with DCASVM, In: 21st annual meeting on the ANLP, D3-4, pp.485-488, Kyoto, 2015 (In Japanese).

[7] M. G. Sohrab, M. Miwa, and Y. Sasaki: Centroid-Means-Embedding: An approach to Infusing word Embeddings into Features for Text Classification, PAKDD-2015, Vol. 9077, pp. 289-300 (2015).

[4] Available at `http://lshtc.iit.demokritos.gr`