WALAILAK JOURNAL
http://wjst.wu.ac.th Information Technology

Hierarchical Text Categorization Using Level Based Neural Networks of Word Embedding Sequences with Sharing Layer Information[†]

Mongkud KLUNGPORNKUN and Peerapon VATEEKUL[*]

Department of Computer Engineering, Chulalongkorn University, Bangkok 10330, Thailand

([*]Corresponding author's e-mail: peerapon.v@chula.ac.th)

Received: 16 July 2017, Revised: 23 September 2018, Accepted: 19 Octoberber 2018

Abstract

In text corpora, it is common to categorize each document to a predefined class hierarchy, which is usually a tree. One of the most widely-used approaches is a level-based strategy that induces a multiclass classifier for each class level independently. However, all prior attempts did not utilize information from its parent level and employed a bag of words rather than considered a sequence of words. In this paper, we present a novel level-based hierarchical text categorization with a strategy called "sharing layer information" For each class level, a neural network is constructed, where its input is a sequence of word embedding vectors generated from Convolutional Neural Networks (CNN). Also, a training strategy to avoid imbalance issues is proposed called "the balanced resampling with mini-batch training" Furthermore, a label correction strategy is proposed to conform the predicted results from all networks on different class levels. The experiment was conducted on 2 standard benchmarks: WIPO and Wiki comparing to a top-down based SVM framework with TF-IDF inputs called "HR-SVM." The results show that the proposed model can achieved the highest accuracy in terms of micro F1 and outperforms the baseline in the top levels in terms of macro F1.

Keywords: Text categorization, hierarchical multi-label classification, deep learning

Introduction

Deep Learning is a successful technique in various fields of machine learning such as computer vision, speech recognition, and natural language processing. The one property of deep learning is an unsupervised feature learning which will become a powerful method of dimensionality reduction and feature embedding. Currently, Word2Vec is a well-known word representation and it has been shown that this method is efficient without applying complex pre-processing [1].

Hierarchical Multi-label Classification (HMC) is a system of grouping things into a hierarchy in which generally classes and specific classes are determined along a relationship from top to bottom level. Most real world data are usually grouped into hierarchies and there are many applications for data manipulation across very different domains such as the international patents [2], a protein function [3], and an image annotation [4]. For the text corpora, Hierarchical Text Categorization (HTC) is a special case where word sparsity becomes a main issue even though the amount of data is huge. For example, DMOZ is a webpage directory that has over a million classes in descriptive hierarchies and Wikipedia is a multilingual encyclopedia which is also hierarchical data. Lots of work has been performed on the multi-label classification on the most appearance classes of the hierarchy and several techniques that use in

[†]Presented at the 14th International Joint Conference on Computer Science and Software Engineering: July 12th -14th, 2017

HTC ignore the word order in corpora to extract features such as Word Count and TF-IDF and they have achieved excellent results. However, there is potential to retrieve more information from sentences.

In this work, we aim to improve the classifier for HTC using Deep Learning to apply the words sequence on the Convolutional Neural Network (CNN) that was introduced by Yoon Kim [5] and we propose level-based neural networks with sharing layer information to retrieve more features about the hierarchy, then we compare with the baseline neural network models that are a fully connected neural network and multi-layer perceptron, and also a TF-IDF based model called HR-SVM [6] that is based on LIBSVM [7] and HEMKit [8]. This paper describes the neural network using sharing layer information from a hierarchy for HTC applications with label correcting approaches and training strategies for the neural network.

The remainder of this paper is organized as follows. Section II is related works about a hierarchical classification, the process of CNN for text feature extraction, and the original model architecture that we have improved. In Section III, the methods and all involved techniques for training and prediction by the model will be explained, and then the experiment setup and preprocessing of the datasets will be stated in Section IV. We discuss the experimental result of our novel method in Section V. Finally, we conclude the experiment results in Section VI.

Materials and methods

Hierarchical classification

Hierarchical classification is a special case of classification that categorizes a hierarchical relationship to form a structure like a Tree or Directed Acyclic Graph (DAG). The hierarchical relationships contain the transitive relation and the asymmetric relation, but the reflexive relation is not included. In the case of a Tree, each class only has one parent class. Let C be a finite set of all classes. All relationships are defined below.

1) Irreflexive: $\forall c_i \in C, c_i \not\prec c_i$

2) Asymmetric: $\forall c_i, c_j \in C, c_i \prec c_j \rightarrow c_j \not\prec c_i$

3) Transitive: $\forall c_i, c_j, c_k \in C, c_i \prec c_j \wedge c_j \prec c_i \rightarrow c_i \prec c_k$

It follows that, these definitions mean any data that are labeled in the subclass are also considered as all ancestor classes, so this problem typically is a multi-label classification.

According to [9] existing hierarchical classification methods, are interested in a local classification approach, which is the composition of classifiers that predicts some part of a hierarchy and they are combined with a hierarchical constraint, so the subclass could not be predicted if any super classes were negative. The advantages of this approach are preserving the natural constraint of hierarchical classes and the compatibility with any classifier, but it may suffer from a blocking problem.

Convolutional neural network for text feature extraction

To extract features from meaningful sequences of words, there is no common method in machine learning for this situation, because the meaning of word vectors is not interpreted in a time like time series, but it depends on the context, grammar, and meaning of each word. However, with deep learning, the convolution neural network is a useful feature extraction model for complicated features like images. The technique from [5] is described as follows.

The process of feature extraction from CNN in sentences is shown in **Figure 1**. From the sentences, Word2Vec generates vectors that represent the words with a vector size of V. In the representation, each word vector from the Word2Vec model are sequentially combined into the same word order as in the original sentences. To apply CNN, the continuous vectors are considered as pictures. In convolution, each filter has a size of the desired length × size of word vector ($2 \times V$ in **Figure 1**) and it convolutes word vectors with max-pooling to extract the data into features. We could interpret that they capture the same short sentence and sum up the words into a feature. Therefore, with this technique, we use this model on our HTC architecture to extract the features from sentences.

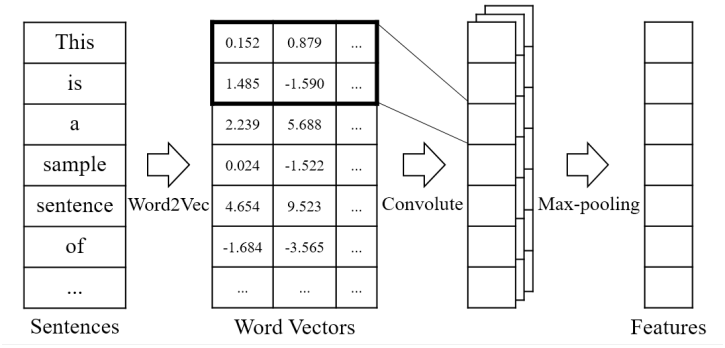

Figure 1 Feature extraction using CNN on word vectors.

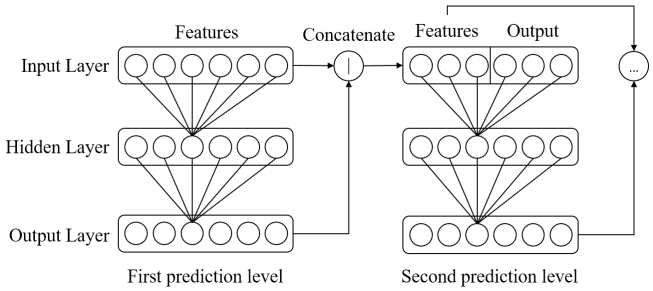

Figure 2 Architecture of the HMC-LMLP in the beginning of 2 layers.

Local multi-layer perceptron for HMC

Cerri *et al*. [3] introduced a Hierarchical Multi-label Classification using Local Multi-Layer Perceptron (HMC-LMLP) to predict the protein function that is a hierarchical classification problem in biology. Their concept is utilizing the output from upper level for prediction in the lower level. The model consists of many layers of Multi-Layer Perceptron (MLP) which are connected by an output layer concatenating with the input features as shown in **Figure 2**.

However, in HTC, we confront the huge hierarchy of categories. So, in the lower level, most input of MLP will come from upper level outputs that is a majority of input, in comparison with the real features. We would like to introduce the sharing layers which came from the idea of utilizing layers that with controllable size of shared inputs that can utilize well-trained layers from the upper prediction level. The improved version of this neural network is described in the Sharing Layer Models section.

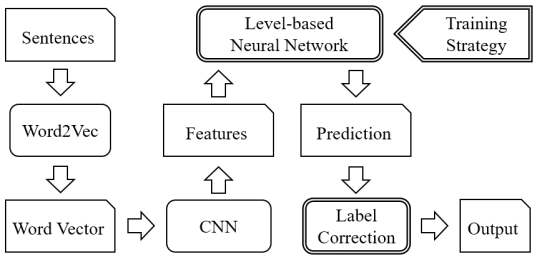

Figure 3 Work flow of the HTC model.

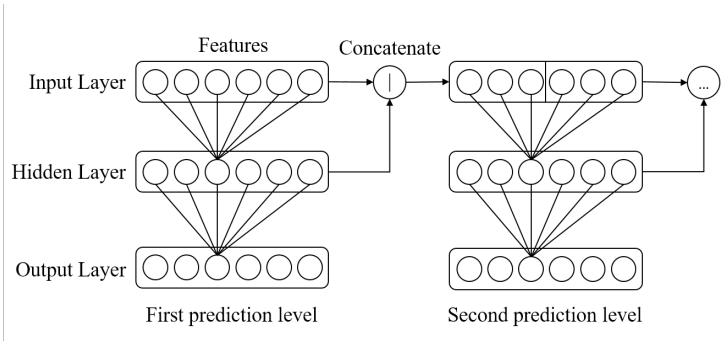

Figure 4 Architecture of Shared Hidden Layer model.

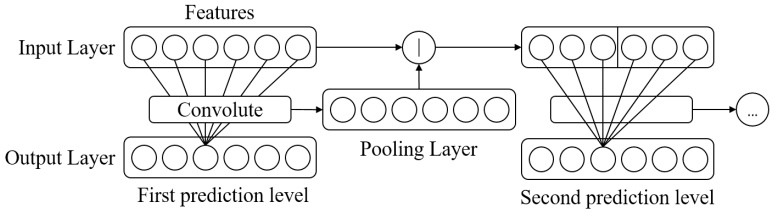

Figure 5 Architecture of Shared Pooling Layer model.

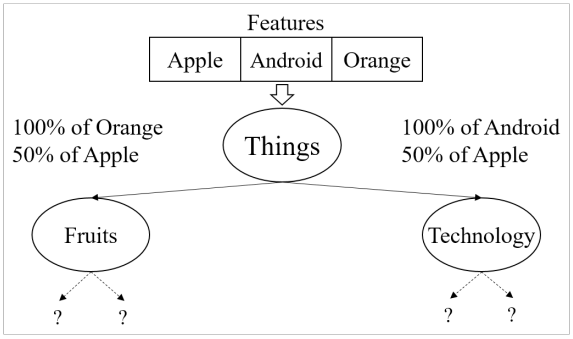

Figure 6 Example of the Hierarchical Classification.

Figure 3 displays the brief workflow of overall process from the sentences to the prediction. At first, we use Word2Vec model training from sentences to represent all words in the sentences into word vectors. Then, we applied the convolutional neural network for features extraction from word vectors which is sequentially combined to be the same as the word order. In our work, we propose a novel level-based neural network to perform the HTC in the prediction process with sharing layer information in topic A, namely Shared Hidden Layer and Shared Pooling Layer. Finally, we perform the label correction to fix the prediction as described in topic B and also apply the training strategy to enhance the accuracy of the NN in the last topic.

Sharing layer models

1) Shared Hidden Layer Neural Network (SHL-NN): The architecture is shown in **Figure 4**. This model was developed from HMC-LMLP. For each prediction level, we use multi-layer perceptron with

inputs from CNN to produces outputs as a prediction of each level. The hidden layers are shared to the next layer by concatenating with the features. It is different from HMC-LMLP that uses an output layer as the shared layer. With this adaptation, we can control the number of nodes in the neural network when we are confronted with a large hierarchy and we can utilize trained hidden layers from higher levels which connect to other hidden layers and produce a more complex MLP.

2) Shared Pooling Layer Neural Network (SPL-NN): From **Figure 5**, the architecture is similar to SHL-NN. The concept came from CNN for text categorization using filters to capture the continuous words as features. In hierarchical classification, we can get more information from the hierarchy. For example, the 3 words-"Apple", "Android", and "Orange" as input features, and the hierarchy is shown in **Figure 6**. We could group "Apple" with "Android" into "Technologies" and "Apple" with "Orange" into "Fruits". From this separation, we could know that the lower classification level might include 2 classes of fruits and technologies which are the numbers of classified features.

So, we use this concept to capture the combination of grouping using CNN on this information. Technically, weights of a single perceptron can be interpreted as the strength of features that influence it over each group. We can use filters to extract from how we classify into the hierarchy, and then these features from hierarchy can be reanalyzed by the next levels. So, SHL-NN was built from many single layer perceptrons of each prediction levels, which share the hierarchy features together.

Let N outputs of the CNN model be input features (X) for hierarchical categorization and the number of outputs being equal to the number of classes (C) of each level. From a single layer perceptron, the outputs (O) are derived from (1) where x is a bias and W is a weight matrix with size $C \times N$.

$$O = f(WX + \beta) \tag{1}$$

The weight matrices were randomly initialized from -0.001 to 0.001 and an activation function (f) is a sigmoid function. Then, we produce hierarchy features (H) from (2) where $w_{i,j}$ is an element of a weight and x_i is a feature from the input.

$$H = f\left(\begin{bmatrix} w_{1,1}x_1 & \cdots & w_{1,n}x_n \\ \vdots & \ddots & \vdots \\ w_{C,1}x_1 & \cdots & w_{C,n}x_n \end{bmatrix} + \beta\right) \tag{2}$$

After that, we applied the convolution operation with a window of 1 × n and then we used max pooling over the results. A shared pooling layer (S) is a concatenated vector of many filters by convolution matrices (M) from (3) where r is a rectifier and hi is an element of the hierarchy features.

$$s_i = max(r(M \times h_1), r(M \times h_2), \ldots, r(M \times h_n)) \tag{3}$$

This shared pooling layer will be combined with other input layers. So, the hierarchy features in the next level will be computed from this shared information and the input features then are sent to the sharing layer in the next level and so on.

Label correction

To correct inconsistencies in the prediction, we applied a label correction for every model before evaluations. There are 2 approaches:

1) Parent-Based Correction: Parent-based correction gives a priority to parent classes over child classes. Any positive labels that its parent class is negative will be removed.

2) Child-Based Correction: Child-based correction is an inverted method. All ancestor classes of positive labels are fixed to positive predictions.

Training strategies for deep learning

To train the neural network, we calculate the error of the model and slowly change many variables until the model converges. The gradient descent is a method for calculation of the steps that we should take to find a minimum error of the model. However, we cannot compute the gradient of errors on the entire data. So, we tested on 4 iterative training strategies to find the most effective ways.

1) Stochastic gradient descent (SGD): SGD is an iterative optimization method using a single sample for each an error calculation and performs an update by sweeping through the training set [10].

2) Mini-batch stochastic gradient descent: Mini-batch is an adaptive strategy which increases training samples to reduce the variance of the parameter updates and it can lead to stable convergence.

3) Stratified sampling with mini-batch: Stratified sampling is a sampling technique that divides the data into separate groups with equal class distribution. There are several advantages over random sampling [11].

4) Balanced resampling with mini-batch: Balanced resampling is our strategy to train the model with more balanced mini-batch. For each batch, we sample with replacement of the low appearance classes to keep the equality of each class. So, the neural networks are trained without a bias from majority classes and the model will pay more attention to a few example classes.

The sampling methods for mini-batch were only applied to the second level of hierarchy but not for all classes because keeping the class distributions in different levels is impossible in the hierarchy structure.

Table 1 Data statistics.

Dataset	Instances	Depth	Class[a]	$\|W\|$[b]
WIPO-D	1710	4	1650	91
WIPO-C	16245	4	5666	77
Wiki-small	60821	4	5162	66

a. Number of classes of all level.

b. $|W|$ refers to average number of words per example in that dataset.

Datasets

We tested our models on 3 datasets on 5-folds stratified cross-validation, which were separated by the stratification method [12] on moderate appearance classes (the middle level of hierarchy), because the lower subclasses are very low in quantities and have a very high numbers of classes. The data statistics are shown in **Table 1**.

1) WIPO-alpha: The national collection of public patent documents for research into automated categorization. This dataset contains 8 sections (from A to H) and has a hierarchical Tree structure. We selected D and C-sections, which are the smallest and the biggest sections for independent testing on main group classification. All instances have only one main group in which the class is a leaf node. We only used title and abstract from documents as input [13].

2) Wiki-small: An online encyclopedia created by the open community where anyone can edit articles. This dataset was selected from the subtree of Wikipedia medium-sized dataset that is a part of the Large-Scale Hierarchical Text Classification Challenge (LSHTC3) [14]. This dataset has a hierarchy in DAG structure with a depth of 4. We used all contents from each page as input.

Models setup

For all datasets, we preprocessed the words using the Lancaster stemmer and removed stop words, then we built word vectors from scratch for each dataset without any external sources, the vectors were trained using the continuous skip-gram model with a dimensionality of 50. Each input sentence is cut by

average length of inputs ($|W|$ in **Table 1**) and it will be expanded with zero-padding to the same length when they are smaller than others.

The inputs of models have applied a dropout with a drop rate of 20 %. After that, we used CNN with filter windows of 3, 4 and 5, with each size producing 100 features. Then, all 300 features were assigned to hierarchical classification levels.

Table 2 Evaluation metric.

	Classification	**Multi-Label Classification**	
		Macro-Averaging	***Micro-Averaging***
Precision	$Pr = TP \Big/ (TP + FP)$	$MaPr = \frac{1}{\lvert C \rvert} \sum_{i=1}^{\lvert C \rvert} Pr_i$	$MiPr = \sum_{i=1}^{\lvert C \rvert} TP_i \Big/ \sum_{i=1}^{\lvert C \rvert} (TP_i + FP_i)$
Recall	$Re = TP \Big/ (TP + FN)$	$MaRe = \frac{1}{\lvert C \rvert} \sum_{i=1}^{\lvert C \rvert} Re_i$	$MiRe = \sum_{i=1}^{\lvert C \rvert} TP_i \Big/ \sum_{i=1}^{\lvert C \rvert} (TP_i + FN_i)$
$\mathbf{F^{\beta}}$	$F_\beta = \frac{(\beta^2 + 1) \times Pr \times Re}{\beta^2 \times Pr + Re}$	$MaF_\beta = \frac{1}{\lvert C \rvert} \sum_{i=1}^{\lvert C \rvert} F_{\beta,i}$	$MiF_\beta = \frac{(\beta^2 + 1) \times MiPr \times MiRe}{\beta^2 \times MiPr + MiRe}$

TP is True Positive, *FP* is False Positive, *FN* is False Negative

Table 3 F_1 of Label Correction Approaches.

Level	**Parent-based correction**	**Child-based correction**
1	0.5677	**0.5785**
2	0.4228	**0.4442**
3	0.0505	**0.0523**
4	0.0055	**0.0075**
Macro-Average	0.0289	**0.0314**

Boldface result is a winner on that prediction level.

The baseline models are Fully Connected Neural Network (FC-NN) and Multi-Layer Perceptron (MLP). For MLP-NN, SHL-NN, and SPL-NN models, a hidden layer size was 2 × (number of classes of its level) with a maximum size at 300 for each level. In shared layer models, the hidden layers were shared to the next 2 levels and each level was fitted to its prediction level using the RMSprop [15] optimizer with a learning rate of 0.001 and a momentum of 0.9 on the cross-entropy function.

For predictions in HMC, we use a threshold at 0.5 for cutting off the outputs into positive and negative predictions and we performed further experiments about the label correction.

Evaluations

The performance criteria are defined by the F-measures with macro- and micro-averaging for the multi-label purpose. To avoid domination of the majority classes, it is appropriate to use macro-averaging, which is an unweighted mean over micro-averaging, which globally counts. The measurements are defined in **Table 2**.

Results and discussion

Firstly, we performed 2 experiments to find out the best method of label correction that is post-processing of prediction outputs and training strategy for testing in further experiments. After that, we compared our model, existing methods, and baseline models with each other. The details of the experiments are explained below.

Label correction approaches

To compare the label correction approaches, we examine a FC-NN model with WIPO-D dataset on parent-based and child-based label correction. **Table 3** shows that the child-based approach is better than the parent-based approach because the parent-based correction leads to the blocking problem from top levels prediction while the child-based approach could correct more labels in top levels of hierarchy.

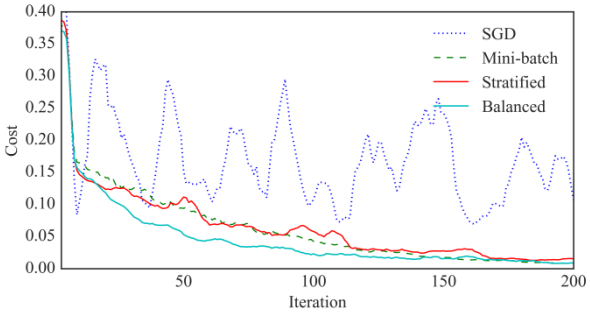

Figure 7 Moving average costs of the training strategies.

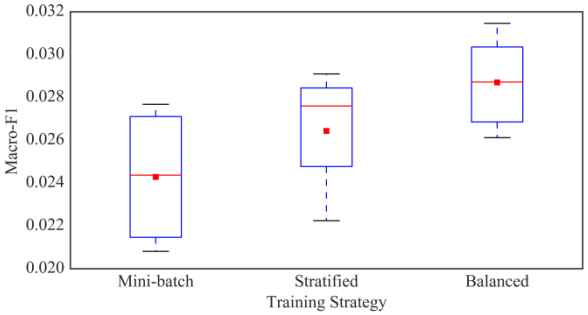

Figure 8 Boxplot of 5-fold stratified cross-validation results.

Training approaches

To find out the best strategy, we test with a FC-NN model using all strategies on the WIPO-D dataset. From a single training set, moving average costs of each training strategy are illustrated in **Figure 7**. With SGD, the cost makes large oscillations due to the gradients being changed every epoch of optimization and it tends to decrease quietly until the final iteration while the other methods quickly converge and slightly reach the bottom. We performed further experiments on Mini-batch, Stratified sampling, and Balanced resampling to find out the best training approach with 5-fold Stratified CV. **Figure 8** shows a boxplot of the results. The balanced resampling mini-batch performed better than the

others because the balanced resampling provides low appearance examples into the training set more than one time per iteration while the stratified mini-batch gives only one time per iteration. We move on to the next experiments with the balanced resampling method.

Hierarchical text categorization

In this experiment, we examine our novel models and compare with baseline models. The results are shown in **Table 4**. With WIPO-D dataset, the best model is SHL-NN while SPL-NN could not work well. For WIPO-C dataset, both our models are better than the baseline models with a 7.3 % gain compared to SPL-NN and 9.1 % gain compared to SHL-NN. These results suggest that large data are good for sharing layer models and the sharing of information can be utilized across the layers. For the Wiki-small dataset, MLP-NN performs remarkably well with a 21 % gain over FC-NN and a 9.9 % gain on SHL-NN. However, SHL-NN shows a 10 % gain from the FC-NN and SPL-NN did not perform well in this dataset. This was probably caused by the DAG hierarchy that has class relationships between itself and more than one parent classes.

Table 4 Results of hierarchical text categorization.

Dataset	Model	Macro-Averaging			Micro-Averaging		
		Precision	***Recall***	***F1***	***Precision***	***Recall***	***F1***
WIPO-D	FC-NN	0.0412	0.0246	0.0287	0.6776	0.3303	0.4440
	MLP-NN	0.0379	0.0271	0.0295	0.5548	0.3225	0.4073
	SHL-NN	0.0399	0.0285	**0.0309**	0.5529	0.3222	0.4070
	SPL-NN	0.0409	0.0223	0.0266	0.7030	0.3110	0.4311
WIPO-C	FC-NN	0.0121	0.0116	0.0110	0.3740	0.3200	0.3376
	MLP-NN	0.0167	0.0090	0.0109	0.5816	0.3227	0.4151
	SHL-NN	0.0184	0.0099	**0.0120**	0.5589	0.3219	0.4084
	SPL-NN	0.0142	0.0113	0.0118	0.4545	0.3281	0.3809
Wiki-small	FC-NN	0.1208	0.1036	0.1035	0.5276	0.5436	0.5352
	MLP-NN	0.1623	0.1157	**0.1256**	0.5820	0.5567	0.5690
	SHL-NN	0.1599	0.1008	0.1143	0.6093	0.5374	0.5709
	SPL-NN	0.1187	0.1037	0.0997	0.4812	0.5558	0.5155

Boldface result is the winner on that dataset.

Table 5 Comparison with existing methods.

Dataset	SHL-NN		HRSVM	
	Macro-F1	***Micro-F1***	***Macro-F1***	***Micro-F1***
WIPO-D	0.0309	**0.4070**	**0.0569**	0.2081
WIPO-C	0.0120	**0.4084**	**0.0316**	0.1074
Wiki-small	0.0620	**0.5099**	**0.2254**	0.2126

Boldface result is the winner on that averaging score.

Table 6 Detailed results in each level.

Dataset	Model	Level	Precision	Recall	F1
WIPO-D	SHL-NN	1	0.615	0.450	**0.487**
		2	0.465	0.391	**0.395**
		3	0.247	0.063	0.069
		4	0.277	0.005	0.006
	HR-SVM	1	0.143	1.000	0.227
		2	0.272	0.792	0.346
		3	0.261	0.207	**0.133**
		4	0.305	0.037	**0.028**
WIPO-C	SHL-NN	1	0.558	0.470	**0.496**
		2	0.401	0.259	**0.297**
		3	0.200	0.021	0.029
		4	0.271	0.004	0.005
	HR-SVM	1	0.059	1.000	0.100
		2	0.129	0.802	0.196
		3	0.123	0.200	**0.090**
		4	0.123	0.033	**0.019**
Wiki-small	SHL-NN	1	0.559	0.560	**0.556**
		2	0.393	0.194	0.204
		3	0.387	0.119	0.133
		4	0.403	0.073	0.087
	HR-SVM	1	0.086	1.000	0.149
		2	0.253	0.549	**0.266**
		3	0.296	0.421	**0.257**
		4	0.299	0.307	**0.207**

Boldface result is the winner on that prediction level.

Existing method comparison

HR-SVM is an adaptive support vector machine for HMC. To evaluate the model, we tested using the same training and testing sets with TF-IDF features. **Table 5** shows that our proposed model does not perform well for overall classes prediction. In contrast, our model results are the best of all micro-averaging F1. So, we further examine the results of each level. **Table 6** shows that, in all datasets, the proposed model performs remarkably well in the top level and the performance drops quickly in the lower level. It shows that the neural network method still faces the blocking problem and could be improved in Macro-average results if we use more models or features like bypass techniques for lower hierarchies in further experiments.

Conclusions

In the present work, we describe a series of experiments on the hierarchical text categorization problem with a neural network using sharing layer information built on top of word vectors. From experiments, the child-based label correction is the approach that prevents the blocking problem and could correct more top levels prediction. Also, the balanced resampling with mini-batch training is proper for avoiding the imbalanced data situation for HTC. The proposed methods can achieve the highest micro-averaging F1. For SHL-NN, it is achieved in the both Tree and DAG hierarchy dataset and for SPL-NN, the shared pooling layer could improve overall performance when there are enough data except for DAG hierarchy. However, the neural network still has a limitation to prediction in most specific

classes at the bottom levels of hierarchy in comparison with HR-SVM. However, the prediction level result indicates that the neural network models have the potential to be improved for overall performance.

Acknowledgements

The authors gratefully acknowledge financial support from the Thailand Research Fund (TRF) under TRF Grant for New Researcher Scholar, Contact No. TRG5780220 and the Commission of Higher Education, Ministry of Education, Thailand.

References

[1] T Mikolov, K Chen, G Corrado and J Dean. Efficient Estimation of Word Representations in Vector Space. Available at: https://arxiv.org/abs/1301.3781, accessed January 2017.
[2] J Rousu, C Saunders, S Szedmak and J Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *J. Mach. Learn. Res*. 2006; **7**, 1601-26.
[3] R Cerri, RC Barros, AC de Carvalho and Y Jin. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinform.* 2016; **17**, 373.
[4] J Fan, Y Gao and H Luo. Hierarchical classification for automatic image annotation. *In*: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, Netherlands, 2007, p. 111-8.
[5] Y Kim. Convolutional neural networks for sentence classification. *In*: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 2014, p. 1746-51.
[6] P Vateekul, M Kubat and K Sarinnapakorn. 2012, Top-down optimized SVMs for hierarchical multi-label classification: A case study in gene function prediction. Ph. D. Dissertation. University of Miami, Florida, USA.
[7] CC Chang and CJ Lin. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Tech*. 2011; **2**, 27.
[8] A Kosmopoulos, I Partalas, E Gaussier, G Paliouras and I Androutsopoulos. Evaluation measures for hierarchical classification: A unified view and novel approaches. *Data Min. Knowl. Discov*. 2015; **29**, 820-65.
[9] CN Silla and AA Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov*. 2011; **22**, 31-72.
[10] T Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. *In*: Proceedings of the 21st International Conference on Machine Learning, Alberta, Canada, 2004, p. 116.
[11] P Zhao and T Zhang. Accelerating Minibatch Stochastic Gradient Descent using Stratified Sampling. Available at: https://arxiv.org/abs/1405.3080, accessed January 2017.
[12] K Sechidis, G Tsoumakas and I Vlahavas. On the stratification of multi-label data. *Lect. Notes Comput. Sci.* 2011; **6913**, 145-58.
[13] D Tikk, G Biró and JD Yang. Experiment with a hierarchical text categorization method on WIPO patent collections. *Int. Intell. Tech.* 2005; **20**, 283-302.
[14] I Partalas, A Kosmopoulos, N Baskiotis, T Artieres, G Paliouras, E Gaussier, I Androutsopoulos, MR Amini and P Galinari. LSHTC: A Benchmark for Large-scale Text Classification. Available at: https://arxiv.org/abs/1503.08581, accessed January 2017.
[15] G Hinton, N Srivastava and K Swersky. Lecture 6a Overview of Mini-batch Gradient Descent. Availiable at: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf, accessed September 2018.