From Small-scale to Large-scale Text Classification

Kang-Min Kim
Korea University
Seoul, South Korea
kangmin89@korea.ac.kr

Yeachan Kim
Korea University
Seoul, South Korea
yeachan@korea.ac.kr

Jungho Lee
Korea University
Seoul, South Korea
leejungho@korea.ac.kr

Ji-Min Lee
Korea University
Seoul, South Korea
jm94318@korea.ac.kr

SangKeun Lee
Korea University
Seoul, South Korea
yalphy@korea.ac.kr

ABSTRACT

Neural network models have achieved impressive results in the field of text classification. However, existing approaches often suffer from insufficient training data in a large-scale text classification involving a large number of categories (e.g., several thousands of categories). Several neural network models have utilized multi-task learning to overcome the limited amount of training data. However, these approaches are also limited to small-scale text classification. In this paper, we propose a novel neural network-based multi-task learning framework for large-scale text classification. To this end, we first treat the different scales of text classification (i.e., large and small numbers of categories) as multiple, related tasks. Then, we train the proposed neural network, which learns small- and large-scale text classification tasks simultaneously. In particular, we further enhance this multi-task learning architecture by using a gate mechanism, which controls the flow of features between the small- and large-scale text classification tasks. Experimental results clearly show that our proposed model improves the performance of the large-scale text classification task with the help of the small-scale text classification task. The proposed scheme exhibits significant improvements of as much as 14% and 5% in terms of micro-averaging and macro-averaging F1-score, respectively, over state-of-the-art techniques.

CCS CONCEPTS

• **Information systems → Clustering and classification**; • **Computing methodologies → Multi-task learning**; **Neural networks**.

KEYWORDS

Large-scale Text Classification; Multi-task Learning; Deep Neural Networks

ACM Reference Format:
Kang-Min Kim, Yeachan Kim, Jungho Lee, Ji-Min Lee, and SangKeun Lee. 2019. From Small-scale to Large-scale Text Classification. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3308558.3313563

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313563

1 INTRODUCTION

Large-scale text classification seeks to classify arbitrary texts into a few semantically relevant classes or categories. There is a wide range of applications of large-scale text classification, including web search personalization [6], contextual advertising [5, 23], topical web search [4], recommender systems [2, 33], etc. The success of these applications is highly dependent on the quality of text classification. Usually, large-scale text classification requires a sufficiently large taxonomy of topical categories to capture various topics in arbitrary texts. In addition, it is necessary to collect a large amount of training data for each category in the taxonomy.

Although neural network models have shown promising results on text classification tasks [17, 21, 42], they perform poorly in large-scale text classification [16], which is often involved in a few or several thousands of categories. This is largely attributed to the fact that the training data for each category is relatively insufficient and distributed unevenly among classification categories.

To handle the large-scale text classification, several works [23, 34] have represented categories in large-scale taxonomy by using an explicit representation model [38] based on bag-of-words or bag-of-phrases. They exhibit a robust performance in the large-scale text classification. One advantage of the explicit representation model used in these studies [23, 34] is that it easily utilizes the taxonomy structure to share training data among categories with "is-a" relationships. However, it may be difficult to apply this method in a neural network that implicitly represents text. In addition, the performance of these approaches is limited to associated knowledge bases.

In the neural network-based text classification, several studies often utilize a multi-task learning framework with other related classification tasks to cope with the limited amount of training data [24–26, 39, 41]. Multi-task learning learns multiple related tasks simultaneously to extract common features and significantly improves the performance of a single task. One characteristic of these multi-task architectures is they share some lower layers to catch common features. Non-shared layers, the remaining layers are split into multiple specific tasks.

In this paper, we propose a multi-task learning framework based on the convolutional neural network (CNN) to handle the large-scale text classification. We apply multi-task learning by treating

Table 1: Statistics of taxonomies

	Original ODP taxonomy	Large-scale taxonomy	Small-scale taxonomy
No. categories	802,379	2,735	13
No. webpages	3,624,444	76,167	1,032,411
Avg. webpages	4.5	27.8	79,416
Min. webpages	1	2	18,021
Max. webpages	1,276	915	171,955

small- and large-scale text classification tasks as two related tasks. The multi-task framework converts the large-scale text classification task to a small-scale text classification task and jointly learns both tasks. This multi-task learning framework with different scales is motivated by our hypothesis that the useful features learned from the small-scale classification task are highly expected to be useful for the large-scale classification task. In particular, the large-scale classification task is further enhanced by task-dependent features from the small-scale classification task through a gate mechanism.

We demonstrate the efficacy of our model on a group of text classification tasks with different scales. Experimental results show that the joint learning of small- and large-scale classification yields significantly improved results in the large-scale text classification task with the help of the small-scale text classification task. In summary, our contributions are three-fold:

- We propose a convolutional neural network (CNN) based multi-task learning framework to handle the large-scale text classification, which shares useful features between small- and large-scale classification.
- We incorporate a gate mechanism into the proposed CNN-based multi-task learning, which helps to selectively utilize useful features of a small-scale classification task in large-scale classification task.
- We demonstrate the efficacy of the proposed methodology through extensive experiments. The performance evaluation clearly shows that our approach significantly outperforms strong baselines in terms of micro- and macro-averaging F1-score.

The remainder of this paper is organized as follows. We briefly describe the knowledge taxonomy and convolutional neural networks for text classification in Section 2. Section 3 describes the multi-task learning framework for large-scale text classification. We present the performance evaluation results and in-depth analysis in Sections 4 and 5, respectively. We discuss related work in Section 6 and conclude the paper in Section 7.

2 PRELIMINARY

2.1 Building Taxonomy

For the large-scale text classification, the first step of our methodology is to build the taxonomy of topics by utilizing the Open Directory Project (ODP) [1]. The ODP is a large-scale and tree-structured web directory with a maximum of 15 levels, where approximately 4 million webpages are classified into 0.8 million categories by volunteer editors. We utilize only subsets of ODP categories rather than entire sets of them to build the large-scale taxonomy, since we observe that there are many useless or too-specific categories. Specifically, we apply three heuristic rules suggested in the work [23] to remove those categories. To construct the small-scale taxonomy, we use only top-level categories from the large-scale taxonomy.

The large-scale taxonomy is well organized to classify arbitrary documents into various topics (e.g., *Top/Sports/Baseball/Major_League/Teams/Los_Angeles_Dodgers/News_and_Media*). However, the training data for each category in large-scale taxonomy is relatively insufficient and distributed unevenly among categories. On the other hand, the small-scale taxonomy represents highly abstracted topics (*Top/Sports*, *Top/Health*, *Top/Computers*, *Top/Business*, etc.), but has sufficient training data (i.e., containing the training data of all descendant categories from the top-level category to the leaf category) whose distribution is relatively balanced. Table 1 shows the statistics of taxonomies.

2.2 Convolutional Neural Network for Text Classification

For the text classification, many studies have utilized neural network models, such as recurrent neural networks [15] and convolutional neural networks [17, 37, 42]. In particular, long short-term memory network (LSTM) [12], which is a type of recurrent neural networks, has achieved impressive results on various natural language processing tasks [13, 28, 36]. However, due to the large number of parameters, LSTM usually requires sufficient training data. Therefore, LSTM may perform poorly in large-scale text classification where training data is insufficient. We adopt convolutional neural networks (CNN) which have been very successful on text classification tasks [17, 37, 42] and need relatively small parameters. The component of CNN-based text classifier usually consists of four layers (i.e., embedding layer, convolution layer, pooling layer and fully connected layer).

2.2.1 Embedding Layer. The embedding layer transforms the document into a vector of embeddings, denoted as $\mathbf{w}_{1:n} \in \mathbb{R}^{nk}$ as the input of the network, where n and k are the number of words and the dimension of word embedding, respectively. A document of length n is represented as follows:

$$\mathbf{w}_{1:n} = \mathbf{w}_1 \oplus \mathbf{w}_2 \oplus \ldots \oplus \mathbf{w}_n \tag{1}$$

where $\oplus$ is the concatenation operator. $\mathbf{w}_i \in \mathbb{R}^k$ is the word vector corresponding to the i-th word in the document. We insert zero-paddings to the both sides of these word representations to handle variable-sized documents and circumvent the border effect [8].

[1] https://curlie.org

2.2.2 Convolution Layer. The function of the convolution layer is to extract features from the input. Using a filter $\mathbf{f} \in \mathbb{R}^{hk}$, a convolution operation on h consecutive word vectors starting from i-th produces a feature. For example, a feature c_i is generated from a window of words $\mathbf{w}_{i:i+h-1}$ as follows:

$$c_i = \mathbf{ReLU}(\mathbf{f} \cdot \mathbf{w}_{i:i+h-1} + \mathbf{b}) \tag{2}$$

where $\mathbf{ReLU}(\cdot)$ is the element-wise rectified linear unit function [31] and $\mathbf{b} \in \mathbb{R}$ is a bias term. Repeating the convolution operations for each possible window of words in the document $\mathbf{w}_{1:h}, \mathbf{w}_{2:h+1}, ..., \mathbf{w}_{n-h+1:n}$, we obtain a feature map $\mathbf{c}$ as follows:

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{3}$$

2.2.3 Pooling Layer. The function of the pooling layer is to further abstract the features $\mathbf{c} \in \mathbb{R}^{n-h+1}$ by aggregating the scores for each filter. There are two choices for the pooling operation, i.e., average and max. In this paper, we apply a max-over-time pooling operation [8] over the feature map and take the maximum value $\hat{\mathbf{c}} = max\{\mathbf{c}\}$ as the feature corresponding to this particular filter. The idea behind max pooling is to capture the most important feature, which has the highest value, for each feature map. With pooling layers, we induce a fixed-length vector from feature maps. To obtain multiple features, we use multiple filters with different window sizes. For m different filters, we simply concatenate the results of each filter and obtain the output vector of the pooling layer $\mathbf{z} \in \mathbb{R}^m$ as follows:

$$\mathbf{z} = [\hat{\mathbf{c}}_1, \hat{\mathbf{c}}_2, ..., \hat{\mathbf{c}}_\mathbf{m}] \tag{4}$$

2.2.4 Fully Connected Layer. The features from the pooling layers are passed to a fully connected softmax layer whose output is the probability distribution over labels. We apply dropout [35] as a mean of regularization by randomly setting to zero a proportion of elements of the feature vector. The parameters of the network are trained to minimize the cross-entropy of the predicted and true distributions.

$$L(\hat{y}, y) = -\sum_{i=1}^{N}\sum_{j=1}^{M} y_i^j log(\hat{y}_i^j), \tag{5}$$

where y_i^j and $\hat{y}_i^j$ are the ground-truth label and prediction probabilities, respectively, while N and M denotes the number of training samples and the class number, respectively.

3 METHODOLOGY

3.1 Multi-task Learning Framework for Large-scale Text Classification

Most neural network methods often suffer from large-scale text classification [16], since training documents for each category are insufficient. Motivated by the success of multi-task learning on text classification [25, 26, 39, 41], we propose a new CNN-based multi-task learning framework for the large-scale text classification. We apply multi-task learning by treating small- and large-scale classification tasks as two related tasks. The proposed framework leverages supervised data from the small-scale classification task, which has relatively sufficient training data whose distribution is balanced. Figure 1 shows our proposed model.

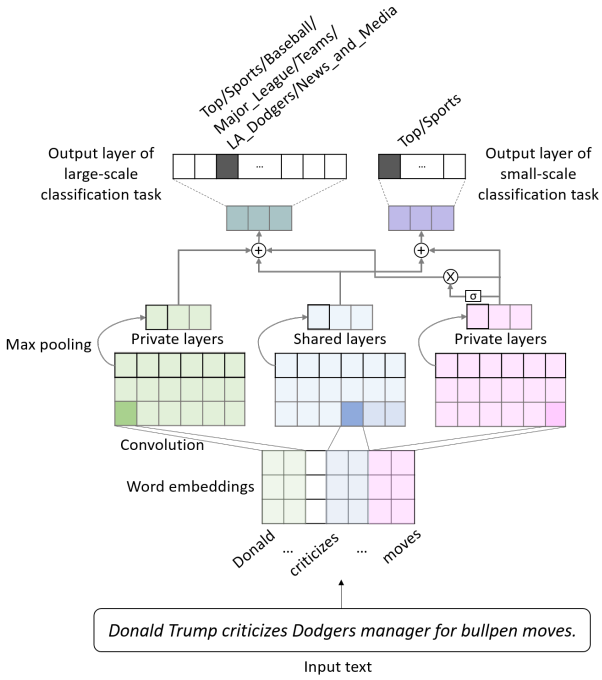

Figure 1: Illustration of architecture for multi-task CNN

In the proposed CNN architecture, the small- and large-scale classification tasks share the same embedding, convolution, and pooling layer. Each task, however, keeps its own private convolution, pooling and fully connected layer as well. Thus, the proposed model maintains two feature spaces: private (i.e., task-dependent) and shared (i.e., task-invariant) feature spaces. We denote the features obtained from the shared pooling layer as $\mathbf{z}_{sh}$. The features obtained from the private pooling layers of the small- and large-scale classification tasks are denoted as $\mathbf{z}_s$ and $\mathbf{z}_l$, respectively.

Although the shared layers help small- and large-scale classification tasks share useful common features, some useful and shareable features may be unidentified. In addition, the private features of small-scale classification task may be higher quality than the private features of large-scale classification task, because the small-scale classification task has relatively sufficient training data. Therefore, we propose a gate mechanism, which enables the large-scale classification task to borrow private features from the small-scale classification task. The gate mechanism controls the flow of features from the small-scale classification task to the large-scale classification task (we shall investigate both symmetric and asymmetric gate mechanisms through experiments in Section 4). When the large-scale classification task l borrows the features from the small-scale classification task s, a gate $\mathbf{g}$ selects the helpful features as follows:

$$\mathbf{g} = \sigma(\mathbf{W}_{s\to l}\mathbf{z}_s + \mathbf{b}_{s\to l}) \tag{6}$$

where σ denotes the non-linear activation of sigmoid, which guarantees the values of $\mathbf{g}$ in the range [0,1]. $\mathbf{W}_{s\to l}$ is the weight which needs to be learned, $\mathbf{b}_{s\to l}$ is a bias term. For the large-scale

Table 2: Statistics of datasets

		Training dataset	Test dataset	Development dataset
ODP (large-scale)	No. categories	2,735	2,735	2,477
	No. webpages	50,246	23,444	2,477
ODP (small-scale)	No. categories	13	13	13
	No. webpages	996,531	23,354	12,526
NYT	No. articles	-	120	-

classification task, the final features $\mathbf{z}'_l$ is calculated by fusing $\mathbf{z}_{sh}$, $\mathbf{z}_s$ and $\mathbf{z}_l$ as follows:

$$\mathbf{z}'_l = \mathbf{z}_l + \mathbf{z}_{sh} + \mathbf{g} \odot \mathbf{z}_s \tag{7}$$

where $\odot$ denotes element-wise multiplication. The features from the small-scale classification task are merged into the large-scale classification task after the selection by gates.

For the small-scale classification task, the final features $\mathbf{z}'_s$ are calculated by adding $\mathbf{z}_{sh}$ and $\mathbf{z}_s$ as follows:

$$\mathbf{z}'_s = \mathbf{z}_s + \mathbf{z}_{sh} \tag{8}$$

In the last layer of each task, vector representations $\mathbf{z}'_l$ and $\mathbf{z}'_s$ are ultimately fed into corresponding fully connected softmax layers to fit the number of classes, which emits the prediction of probability distribution for tasks l and s, respectively.

$$\hat{y_l} = softmax(\mathbf{W}_l \mathbf{z}'_l + \mathbf{b}_l) \tag{9}$$

$$\hat{y_s} = softmax(\mathbf{W}_s \mathbf{z}'_s + \mathbf{b}_s) \tag{10}$$

where $\hat{y_l}$ and $\hat{y_s}$ are predictive results, $\mathbf{W}_l$ and $\mathbf{W}_s$ are the weights of the fully connected layers, while $\mathbf{b}_l$ and $\mathbf{b}_s$ are the bias terms.

Given the predictions of all tasks, a global loss function forces the model to minimize the cross-entropy of the predicted and true distributions for all the tasks as follows:

$$\Phi = \lambda_l L(\hat{y_l}, y_l) + \lambda_s L(\hat{y_s}, y_s) \tag{11}$$

where λ_l and λ_s are the weights for the large- and small-scale classification task, respectively.

3.2 Training

3.2.1 Joint Training. Following [7, 25], the training of the small- and large-scale text classification tasks is achieved in a stochastic manner by looping over the tasks:

(1) Select a random task.
(2) Select a random training example from this task.
(3) Update the parameters for this by taking a gradient step with respect to this example.
(4) Go to 1.

3.2.2 Fine Tuning. For the proposed model, there are shared layers for the small- and large-scale text classification tasks. Thus, after the joint learning phase, we use a fine tuning strategy to further optimize the performance for the large-scale text classification task.

3.2.3 Over-sampling. The distribution of the ODP training dataset for the large-scale text classification task is highly skewed. We apply an over-sampling, which duplicates instances of under-represented classes until the dataset gets balanced, to lessen the impact of the imbalanced dataset [11, 20].

4 EXPERIMENTS

We experiment with a large-scale text classification task to verify the efficacy of the proposed multi-task learning framework.

4.1 Datasets

4.1.1 ODP. We use the RDF dump from the original ODP dataset released on January 8, 2017. As previously mentioned, to obtain a well-organized ODP taxonomy, we apply heuristic rules and build our own large-scale taxonomy with 2,735 categories. Thus, the final training dataset used in our experiments consists of 50,246 webpages. To construct the small-scale classification dataset, we use only 13 top-level categories from the large-scale taxonomy by excluding two categories, *Top/News* and *Top/Adult*, which contain fewer than 100 webpages. Thus, the training dataset used in the small-scale classification task consists of 996,531 webpages.

The ODP test dataset consists of webpages collected from the original ODP. The webpages in each category are randomly divided into a training set, a test set and a development set. In particular, we build two kinds of ODP test datasets. In the large-scale classification task, we collect 23,444 webpages from 2,735 ODP categories in large-scale taxonomy, while collecting 23,354 webpages from 13 ODP categories in the small-scale classification task.

4.1.2 NYT. In addition to the ODP test datasets, we select six categories related to the New York Times as the source for our second test dataset: *art*, *business*, *food*, *health*, *politics*, and *sports*. We randomly collect 20 news articles from each of these categories. Table 2 shows the statistics of datasets.

4.2 Evaluation Metrics

For the ODP datasets, we use the F_1 measure [40] as the classification performance metric, which is the balanced harmonic mean of precision and recall. We use two averaging methods to compute the F_1 measure: macro-averaging (Ma-F_1) and micro-averaging (Mi-F_1). In macro-averaging, F_1 value is measured for each category and then a simple average of all F_1 measures is computed. Conversely, micro-averaging computes the F_1 measure over all test documents. That is, macro-averaging gives equal weight to each category, whereas micro-averaging gives equal weight to each document classification decision. Thus, Mi-F_1 measure depends on the performance of categories with a large number of test documents,

while Ma-F_1 measure is strongly influenced by the performance of categories with fewer test documents.

For the NYT test dataset, we use precision at k. Three participants manually assess the top-k ODP categories obtained by the text classifiers according to three scales: relevant, somewhat relevant, and not relevant. For example, if two categories from the top-5 categories are relevant and one category is somewhat relevant, then the precision at 5 is measured as 0.6.

4.3 Comparisons with State-of-the-art Models

We evaluate the performance of eleven methods. We first adopt the explicit representation-based text classification [23] as the state-of-the-art for large-scale text classification. Other baselines include CNN, LSTM, fastText, and paragraph vector-based text classifiers, which are state-of-the-art methods on multi-class text classification. We also choose four baselines for neural network-based multi-task learning. In our experiments, we compare the following methods:

- *MC* (baseline): This is a text classification method using the explicit representation model based on bag-of-words, called merge centroid, which utilizes enriched training data for each category based on documents classified into their descendants [23].
- *CNN* (baseline): This is the convolutional neural network-based text classification method [17]. We set the hyperparameters to be the same as those of the proposed model.
- *LSTM* (baseline): This is the long short-term memory network-based text classification method [15]. We set the hidden dimension of the LSTM layer to 800, while the other hyperparameters (i.e., mini-batch size, dropout rate, regularization weight, and learning rate) are set to be the same as those of the proposed model.
- *BiLSTM* (baseline): This is the bidirectional long short-term memory network-based text classification method. We set the hidden dimension of each LSTM layer to 800, while the other hyperparameters are set to be the same as those of the proposed model.
- *PV* (baseline): This is the text classification method using paragraph vectors [22]. The learned vector representations have 300 dimensions. We represent categories by averaging the document embeddings for each document in a category.
- $fastText$ (baseline): This is a simple n-gram based neural network model for text classification [14]. The learned vector representations have 300 dimensions.
- *MT-DNN* (baseline): This is a multi-task learning model with DNN, in which a hidden layer is shared [27]. We set the hidden dimension of a shared layer to 300, while the other hyperparameters are set to be the same as those of the proposed model.
- *MT-CNN* (baseline): This is a multi-task learning model with CNN, in which lookup-tables are shared partially while other layers are task-specific [7]. We set the number of filters of private convolution layers to 300, while the other hyperparameters are set to be the same as those of the proposed model.
- *FS-LSTM* (baseline): This is a multi-task learning model with LSTM, in which a single LSTM layer is shared [26]. We set the hidden dimension of a shared LSTM layer to 300, while the other hyperparameters are set to be the same as those of the proposed model.
- *SP-LSTM* (baseline): This is a multi-task learning model with LSTM, in which each task has a private LSTM layer and shares a shared LSTM layer [26]. We set the hidden dimension of both the shared and private LSTM layers to 300, while the other hyperparameters are set to be the same as those of the proposed model.
- *SPG-CNN*: This is our proposed multi-task learning with CNN, in which each task has a private convolution layer and shares a shared convolution layer. In addition, the large-scale classification task borrows private features from the small-scale classification task through a gate mechanism.

Table 3: Detail of word embeddings

	Corpus	Dimension	|V|
Word2Vec [30]	Google News	300	3,000,000
GloVe [32]	Wikipedia	300	400,000
fastText [3]	Wikipedia	300	∞
GWR [18]	Google News	300	∞

4.4 Hyperparameters and Training

We implement the proposed model and competitor methods using TensorFlow frameworks [1] and train models in a single machine equipped with an Intel Core i5 processor, 32GB of RAM, and an NVIDIA GeForce GTX 1080 Ti with 11GB of RAM.

The networks are trained with backpropagation and the gradient-based optimization is performed using the Adam update rule [19]. The embedding layers for all of the neural networks based models (i.e., apart from *MC*) are initialized with the four pre-trained word embeddings, which are summarized in Table 3. We adopt the publicly available Word2Vec [2] and GloVe [3] models, which are popular word embeddings techniques. However, Word2Vec and GloVe models cannot represent out-of-vocabulary (OOV) words, which do not appear in the pre-trained word embedding. Thus, we additionally adopt fastText [3] (this is a word representation technique, being different from $fastText$ [14] that belongs to the baselines) and GWR [18], a character-based word representation approach, to handle OOV words. These approaches allow our framework to effectively expand the vocabulary in the large-scale text classification. The word embeddings are fine-tuned during training to improve performance. The other parameters are initialized by Xavier [9].

The hyperparameters that achieve the best performance on the development set will be chosen for the final evaluation. The final hyperparameters are as follows. For all datasets we use the ReLU, filter windows of 2, 3 and 4 with 300 filters each, mini-batch size of 64, dropout rate of 0.6, regularization weight of 0.001, and learning rate of 0.001. We set both λ_l and λ_s to 0.5.

[2] https://code.google.com/archive/p/word2vec/
[3] https://nlp.stanford.edu/projects/glove/

4.5 Experimental Results

We first compare a few gate mechanisms with the ODP dataset (2,735 categories). In Table 4, *SP-CNN* denotes the multi-task learning model with CNN by the shared and private layers architecture without the gate mechanism. *SPG-CNN*$_{s2l}$ denotes *SP-CNN* using the gate mechanism that controls the flow of features from the small- to the large-scale text classification task, while the gate mechanism in *SPG-CNN*$_{l2s}$ controls the flow of features from the large- to the small-scale classification task. In *SPG-CNN*$_{bi}$, the gate mechanism is used in two-way. We observe that *SPG-CNN*$_{s2l}$ outperforms *SP-CNN*, *SPG-CNN*$_{l2s}$, and *SPG-CNN*$_{bi}$. In particular, our experimental results show that *SPG-CNN*$_{l2s}$ performs worse than *SP-CNN*. These results demonstrate that only the flow of features from the small- to the large-scale classification task is effective at the large-scale text classification. Thus, we adopt *SPG-CNN*$_{s2l}$ in the rest of experiments, which is simply denoted as *SPG-CNN* hereafter.

Table 4: Comparison of symmetric and asymmetric gate mechanisms on the ODP dataset (2,735 categories)

	Mi-F_1	Ma-F_1
SP-CNN	0.550	0.430
SPG-CNN$_{s2l}$	**0.556**	**0.448**
SPG-CNN$_{l2s}$	0.509	0.429
SPG-CNN$_{bi}$	0.551	0.442

We also compare the performance of different pre-trained word embeddings in *SPG-CNN*. Table 5 shows the results of various word embeddings. We observe that *SPG-CNN* performs the best when the embedding layer is initialized with fastText. The publicly available Word2Vec and GloVe embeddings are 4.3% and 2.8% behind fastText embeddings in terms of Ma-F_1, respectively. GWR embeddings perform close to fastText embeddings. One possible reason that fastText and GWR, character-based word embedding approaches, perform better than other popular word-based embeddings is that a large number of OOV words in pre-trained word embeddings lead to the suboptimal performance of large-scale text classification. In the remaining experiments, the embedding layers of all the neural networks based models are initialized with fastText.

Table 5: Comparison of different word embeddings on the ODP dataset (2,735 categories)

	Word embedding	Mi-F_1	Ma-F_1
SPG-CNN	Word2Vec [30]	0.525	0.405
	GloVe [32]	0.530	0.420
	fastText [3]	**0.556**	**0.448**
	GWR [18]	0.550	0.435

Table 6 summarizes the experimental results for large-scale text classification on the ODP test dataset with 2,735 target classes. We observe that *SPG-CNN* outperforms all the baselines. *SPG-CNN* performs better than *MC* over 14% and 5% on average in terms of Mi-F_1 and Ma-F_1, respectively. We also observe that *SPG-CNN* exhibits

Table 6: Large-scale classification performance on the ODP dataset (2,735 categories)

	Mi-F_1	Ma-F_1
MC [23]	0.486	0.426
CNN [17]	0.528	0.413
LSTM [15]	0.430	0.316
BiLSTM	0.495	0.376
PV [22]	0.337	0.234
$fastText$ [14]	0.460	0.371
MT-DNN [27]	0.373	0.260
MT-CNN [7]	0.512	0.435
FS-LSTM [26]	0.545	0.413
SP-LSTM [26]	0.541	0.418
SPG-CNN	**0.556**	**0.448**

Table 7: Small-scale classification performance on the ODP dataset (13 categories)

	Mi-F_1	Ma-F_1
MC [23]	0.757	0.689
CNN [17]	0.804	0.754
LSTM [15]	0.810	0.763
BiLSTM	0.824	0.779
PV [22]	0.738	0.612
$fastText$ [14]	0.727	0.568
MT-DNN [27]	0.793	0.731
MT-CNN [7]	0.800	0.739
FS-LSTM [26]	0.826	0.781
SP-LSTM [26]	**0.828**	**0.782**
SPG-CNN	0.804	0.755

improvement compared to *MT-DNN*, *MT-CNN*, *FS-LSTM* and *SP-LSTM* by over 72%, 3%, 8%, and 7% in terms of Ma-F_1, respectively. Our experimental results show that neural network-based single-task methods except CNN (i.e., *LSTM*, *BiLSTM*, *PV*, and $fastText$) perform worse than *MC* in terms of Mi-F_1 and Ma-F_1. It turns out that *CNN* performs the best among the single-task neural models and better than *MC* over 9% on average in terms of Mi-F_1. However, *CNN* performs worse than *MC* in terms of Ma-F_1. The average performance of multi-task learning models with different scales (i.e., *MT-DNN*, *MT-CNN*, *FS-LSTM*, *SP-LSTM*, and *SPG-CNN*) achieves an improvement of 10.8% and 10.9% over the average performance of single-task learning models in terms of Mi-F_1 and Ma-F_1, respectively. This experimental result indicates that the utilization of multi-task learning with different scales is indeed effective in the large-scale text classification.

Table 7 shows the evaluation results for small-scale text classification on the ODP test dataset with 13 target categories. We observe that *SP-LSTM* outperforms *SPG-CNN*, as well as other baselines. These results are consistent with the results of previous work [24–26], reporting the superior performance of LSTM-based text classifier trained on sufficient training data whose distribution is balanced. We observe that all deep neural network-based

single-task methods (i.e., *CNN*, *LSTM*, and *BiLSTM*) exhibit better performances than *MC* in the small-scale text classification. From Table 6 and 7, we confirm that neural network-based single-task methods are indeed limited to the small-scale text classification. We also observe that the average performance of multi-task learning models with different scales achieves the improvement of 4% and 10% over that of single-task learning models in terms of Mi-F_1 and Ma-F_1, respectively. This result demonstrates that multi-task learning is also helpful to the small-scale text classification task.

Table 8 shows evaluation results on the NYT test dataset. Again, *SPG-CNN* outperforms *CNN*, *LSTM*, *BiLSTM*, *PV*, *fastText*, *MT-DNN*, *MT-CNN*, *FS-LSTM*, and *SP-LSTM* by over 49%, 86%, 52%, 122%, 34%, 18%, 6%, 34%, and 29% in terms of precision at k on average, respectively. We also observe that the average performance of multi-task learning models with different scales achieves the improvement of 41% over that of single-task learning models on average. We also perform the t-test for the classification results, and find that *SPG-CNN* results are statistically significant with $p < 0.01$.

Table 8: Classification performance on the NYT dataset (2,735 categories)

	Precision at k				
	1	2	3	4	5
CNN [17]	0.325	0.279	0.231	0.208	0.193
LSTM [15]	0.225	0.221	0.189	0.177	0.175
BiLSTM	0.308	0.275	0.219	0.204	0.200
PV [22]	0.208	0.171	0.161	0.150	0.138
fastText [14]	0.292	0.296	0.281	0.260	0.242
MT-DNN [27]	0.383	0.317	0.300	0.285	0.272
MT-CNN [7]	0.417	0.354	0.333	0.325	0.310
FS-LSTM [26]	0.292	0.296	0.281	0.260	0.242
SP-LSTM [26]	0.325	0.288	0.281	0.267	0.262
SPG-CNN	**0.458**	**0.392**	**0.353**	**0.325**	**0.312**

We measure the training time for *SPG-CNN* and other neural network-based multi-task learning models. For a fair comparison, we use the same hardware resources, and record the training time for a single epoch on ODP dataset. In Table 9, we observe that *SPG-CNN* has better classification performance while learning faster than *FS-LSTM* and *SP-LSTM* by 1.2x and 1.26x, respectively. Although *MT-DNN* and *MT-CNN* are 1.98x and 1.18x faster than *SPG-CNN*, respectively, *MT-DNN* and *MT-CNN* perform worse than *SPG-CNN*. These results confirm that *SPG-CNN* is effective with a reasonable cost of training time.

5 ANALYSIS

We qualitatively examine the classified categories to analyze why the utilization of the small-scale classification task with the gate mechanism improves the performance of large-scale text classification. We take the NYT articles related to topics "Health care policies" and "Nutritions" as examples. First, from Table 10, we observe that *SPG-CNN* and *SP-CNN* classify the given documents into highly relevant categories. More specifically, from Table 10(a),

Table 9: Training time for a single epoch on ODP dataset compared to multi-task learning models

	Training time	Mi-F_1	Ma-F_1
MT-DNN [27]	**2h 7m**	0.373	0.260
MT-CNN [7]	3h 34m	0.512	0.435
FS-LSTM [26]	5h 3m	0.545	0.413
SP-LSTM [26]	5h 17m	0.541	0.418
SPG-CNN	4h 12m	**0.556**	**0.448**

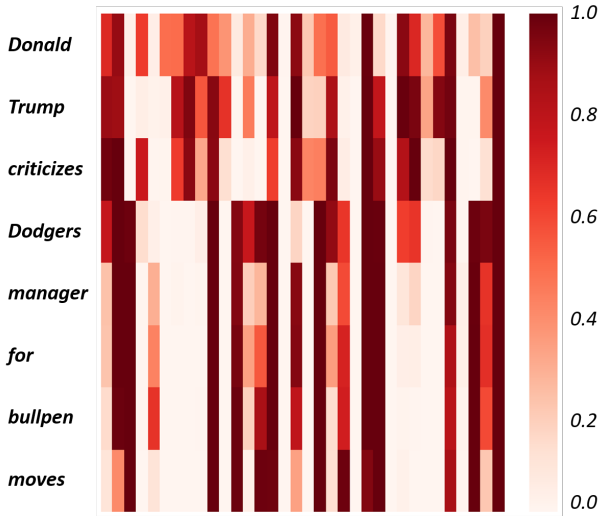

Figure 2: Visualization of the gate's activation. Y-axis represents the input words in chronological order, while X-axis represents gates.

we observe that *SPG-CNN* returns the "Social issues" related categories in top-3. *SP-CNN* also returns two "Social issues" related categories and a category related to "Insurance" that are somewhat associated with the article. In contrast, *CNN* often misclassifies top-level category (e.g., *Top/Computers/...*). In Table 10(b), we also observe that all the categories in top-3 returned by *SPG-CNN* are about "Nutritions" or "Medicine research", while *SP-CNN* returns two relevant categories and an irrelevant category. *CNN* returns a relevant category only on top-1.

We also observe that *SP-CNN* has the same small-scale classification results as *SPG-CNN* (i.e., *Top/Society*, *Top/Kids_and_Teens*, and *Top/Health* in Table 10(a), while *Top/Shopping*, *Top/Health*, and *Top/Science* in Table 10(b)). Nevertheless, it often returns categories that are not related to small-scale classification results (e.g., *Top/Business/Financial_Services/Insurance/Carriers*) and produces worse classification results than *SPG-CNN*. These results clearly demonstrate that the proposed gate mechanism successfully improves the performance of the large-scale text classification by effectively utilizing useful features of the small-scale classification task.

Table 10: Illustrations of Classified Categories

(a) "States' Policies on Health Care Exclude Some of the Poorest. The refusal by about half the states to expand Medicaid will leave millions of poor people ineligible for government-subsidized health insurance under President Obama's health care law even as many others with higher incomes receive federal subsidies to buy insurance." in NYT dataset

Method	Scale	Rank	Category
SPG-CNN	large-scale	1	***Top/Society/Issues/Poverty***
		2	***Top/Society/Issues/Health***
		3	***Top/Society/Issues/Children,_Youth_and_Family/Advocacy_Groups***
	small-scale	1	***Top/Society***
		2	*Top/Kids_and_Teens*
		3	***Top/Health***
SP-CNN	large-scale	1	***Top/Society/Issues/Poverty***
		2	***Top/Society/Issues/Health***
		3	*Top/Business/Financial_Services/Insurance/Carriers*
	small-scale	1	***Top/Society***
		2	*Top/Kids_and_Teens*
		3	***Top/Health***
CNN [17]	large-scale	1	***Top/Society/Issues/Health***
		2	***Top/Home/Personal_Finance/Insurance***
		3	*Top/Computers/Software/Industry-Specific/Insurance*

(b) "Fish Oil Claims Not Supported by Research. Fish oil is now the third most widely used dietary supplement in the United States, after vitamins and minerals, according to a recent report from the National Institutes of Health. At least 10 percent of Americans take fish oil regularly, most believing that the omega-3 fatty acids in the supplements will protect their cardiovascular health." in NYT dataset

Method	Scale	Rank	Category
SPG-CNN	large-scale	1	***Top/Shopping/Health/Nutrition***
		2	***Top/Shopping/Health/Nutrition/Supplements***
		3	***Top/Health/Medicine/Research***
	small-scale	1	***Top/Shopping***
		2	***Top/Health***
		3	***Top/Science***
SP-CNN	large-scale	1	***Top/Shopping/Health/Nutrition/Supplements***
		2	*Top/Shopping/Health/Alternative/Herbs*
		3	***Top/Health/Medicine/Research***
	small-scale	1	***Top/Shopping***
		2	***Top/Health***
		3	***Top/Science***
CNN [17]	large-scale	1	***Top/Shopping/Health/Nutrition/Supplements***
		2	*Top/Business/Industrial_Goods_and_Services/Calibration_and_Testing/Ultrasonic_Testing*
		3	*Top/Recreation/Pets/Fish_and_Aquaria*

In addition, we analyze the gate activation in *SPG-CNN*, which controls the flow of features from small-scale classification task to large-scale classification task. We select an article title "Donald Trump criticizes Dodgers manager for bullpen moves", which is related to both "Issues" and "Sports". In Figure 2, we plot the gate activation as each word is processed. The small- and large-scale classification tasks of *SPG-CNN* return *Top/Society* and *Top/Society/Issues/Warfare_and_Conflict* when processing up to the third word (or criticizes), while *Top/Sports* and *Top/Sports/Baseball/Major_League/Teams/ Los_Angeles_Dodgers/News_and_Media* when the last word is processed. Interestingly, from Figure 2, we observe that different neurons are activated as predicted categories get changed. These results show how the large-scale text classification task in *SPG-CNN* selectively utilizes useful features of the small-scale classification task in practice.

6 RELATED WORK

For the large-scale text classification, many works have been proposed to handle data sparsity on knowledge bases. McCallum et al. firstly addressed data sparsity on a hierarchical taxonomy [29]. They adopted a statistical technique, called *shrinkage*, to estimate the parameters of data-sparse child categories with their data-rich

ancestor categories. In [10, 23], they proposed a large-scale text classification method called merge-centroid (MC). MC utilizes enriched training data for each category based on webpages classified into their ancestor and/or descendants in the ODP. In another line of work [34], they enriched semantic information, such as phrases and related terms, in the ODP documents by utilizing another knowledge base, Wikipedia. However, these approaches only consider term weighting approaches, without considering the importance of semantic similarity between words.

Neural network based models have utilized multi-task learning to overcome the problem of insufficient training data. In [7], they proposed multi-task learning with convolutional neural networks for several traditional NLP tasks. This study utilized a shared representation for input words, while we utilize a shared convolution layer and private convolution layers to construct the multi-task learning framework. A related work [25] proposed three different models of sharing information with recurrent neural networks (RNN). In another line of work [26], the authors augmented RNN-based multi-task learning by using adversarial training. Another work [39] proposed a CNN-based architecture for multi-task learning, which shares features via a gated sharing unit instead of a shared layer. However, none of the above research deals with the large-scale text classification. To the best of our knowledge, our current work is one of only a few works that apply multi-task learning to the large-scale text classification by treating different scales of text classification.

7 CONCLUSION

In this paper, we have proposed a novel CNN-based multi-task learning framework to handle the large-scale text classification. Specifically, we have incorporated the gate mechanism into the multi-task CNN to selectively utilize useful features of small-scale classification task. Our approach involves two steps. We first convert the large-scale classification task to a small-scale classification task. Secondly, we train the proposed CNN-based multi-task learning model, which simultaneously learns small- and large-scale classification tasks with the gate mechanism. We have verified the large-scale text classification performance of our methodology using real-world datasets. The experimental results confirm that our scheme significantly outperforms several strong baseline methods. We plan to integrate explicit and implicit representation techniques to further enhance the large-scale text classification.

ACKNOWLEDGMENT

We would like to thank the anonymous reviewers for their valuable comments. This research was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018R1A2A1A05078380).

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A System for Large-scale Machine Learning. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'16)*. 265–283.

[2] Bahram Amini, Roliana Ibrahim, Mohd Shahizan Othman, and Mohammad Ali Nematbakhsh. 2015. A Reference Ontology for Profiling Scholar's Background Knowledge in Recommender Systems. *Expert Syst. Appl.* 42, 2 (Feb. 2015), 913–928.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146.

[4] Andrei Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. 2007. Robust classification of rare queries using web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. 231–238.

[5] Andrei Broder, Marcus Fontoura, Vanja Josifovski, and Lance Riedel. 2007. A Semantic Approach to Contextual Advertising. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. 559–566.

[6] Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. 2005. Using ODP metadata to personalize search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*. 178–185.

[7] Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML'08)*. 160–167.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *J. Mach. Learn. Res.* 12 (Nov. 2011), 2493–2537.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10)*. 249–256.

[10] Jongwoo Ha, Jung-Hyun Lee, Won-Jun Jang, Yong-Ku Lee, and SangKeun Lee. 2014. Toward Robust Classification Using the Open Directory Project. In *Proceedings of the 2014 International Conference on Data Science and Advanced Analytics (DSAA'14)*. 607–612.

[11] Haibo He and Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Trans. on Knowl. and Data Eng.* 21, 9 (Sept. 2009), 1263–1284.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (Nov. 1997), 1735–1780.

[13] Rie Johnson and Tong Zhang. 2016. Supervised and Semi-supervised Text Categorization Using LSTM for Region Embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning (ICML'16)*. 526–534.

[14] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL'17)*. 427–431.

[15] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. 2342–2350.

[16] Kang-Min Kim, Dinara Aliyeva, Byung-Ju Choi, and SangKeun Lee. 2018. Incorporating Word Embeddings into Open Directory Project based Large-scale Classification. In *Proceedings of the 22nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'18)*. 376–388.

[17] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1746–1751.

[18] Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018. Learning to Generate Word Representations using Subword Information. In *Proceedings of the 27th International Conference on Computational Linguistics (COLING'18)*. 2551–2561.

[19] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[20] Bartosz Krawczyk. 2016. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence* 5 (2016), 221–232.

[21] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*. 2267–2273.

[22] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML'14)*. 1188–1196.

[23] Jung-Hyun Lee, JongWoo Ha, Jin-Yong Jung, and SangKeun Lee. 2013. Semantic Contextual Advertising based on the Open Directory Project. *ACM Trans. on the Web* 7, 4 (Nov. 2013), 24:1–24:22.

[24] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Deep Multi-Task Learning with Shared Memory for Text Classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 118–127.

[25] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent Neural Network for Text Classification with Multi-Task Learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*. 2873–2879.

[26] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL'17)*. 1–10.

[27] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. 2015. Representation Learning Using Multi-Task Deep Neural Networks for Semantic Classification and Information Retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'15)*. 912–921.

[28] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*. 1412–1421.

[29] Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. 1998. Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *Proceedings of the 15th International Conference on Machine Learning (ICML'98)*. 359–367.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NIPS'13)*. 3111–3119.

[31] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 807–814.

[32] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1532–1543.

[33] Woo-Jong Ryu, Jung-Hyun Lee, Kang-Min Kim, and SangKeun Lee. 2017. meCurate: Personalized Curation Service Using a Tiny Text Intelligence. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW'17 Companion)*. 269–272.

[34] HaeYong Shin, GeunJae Lee, Woo-Jong Ryu, and SangKeun Lee. 2017. Utilizing Wikipedia Knowledge in Open Directory Project-based Text Classification. In *Proceedings of the 32nd Symposium on Applied Computing (SAC'17)*. 309–314.

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 1929–1958.

[36] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 28th Conference on Neural Information Processing Systems (NIPS'14)*. 3104–3112.

[37] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. 2017. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 2915–2921.

[38] Zhongyuan Wang and Haixun Wang. 2016. Understanding Short Texts. In *the 54th Annual Meeting of the Association for Computational Linguistics (Tutorial) (ACL'16)*.

[39] Liqiang Xiao, Honglun Zhang, and Wenqing Chen. 2018. Gated Multi-Task Network for Text Classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'18)*. 726–731.

[40] Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Inf. Retr.* 1, 1 (May 1999), 69–90.

[41] Honglun Zhang, Liqiang Xiao, Yongkun Wang, and Yaohui Jin. 2017. A generalized recurrent neural architecture for text classification with multi-task learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 3385–3391.

[42] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS'15)*. 649–657.