Multi-pretraining for Large-scale Text Classification

Kang-Min Kim[1] **Bumsu Hyeon**[1] **Yeachan Kim**[2] **Jun-Hyung Park**[1] **SangKeun Lee**[1,3]
[1]Department of Computer Science and Engineering, Korea University, Seoul, Republic of Korea
[2]Artificial Intelligence Research Institute, Republic of Korea
[3]Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea
{kangmin89, bshyeon, yeachan, irish07, yalphy}@korea.ac.kr

Abstract

Deep neural network-based pretraining methods have achieved impressive results in many natural language processing tasks including text classification. However, their applicability to large-scale text classification with numerous categories (e.g., several thousands) is yet to be well-studied, where the training data is insufficient and skewed in terms of categories. In addition, existing pretraining methods usually involve excessive computation and memory overheads. In this paper, we develop a novel multi-pretraining framework for large-scale text classification. This multi-pretraining framework includes both a self-supervised pretraining and a weakly supervised pretraining. We newly introduce an out-of-context words detection task on the unlabeled data as the self-supervised pretraining. It captures the topic-consistency of words used in sentences, which is proven to be useful for text classification. In addition, we propose a weakly supervised pretraining, where labels for text classification are obtained automatically from an existing approach. Experimental results clearly show that both pretraining approaches are effective for large-scale text classification task. The proposed scheme exhibits significant improvements as much as 3.8% in terms of macro-averaging F_1-score over strong pretraining methods, while being computationally efficient.

1 Introduction

Large-scale text classification is a natural language processing (NLP) task and a long-standing yet challenging problem. It seeks to classify arbitrary texts into semantically relevant classes or categories. A wide range of applications exploit large-scale text classification, including web search personalization (Chirita et al., 2005), contextual advertising (Lee et al., 2013), topical web search (Broder et al., 2007), and recommender systems (Amini et al., 2015). The success of these applications is highly dependent on the quality of text classification. Many applications of large-scale text classification require 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.

Deep neural models have demonstrated promising results in text classification tasks (Kim, 2014; Zhang et al., 2015; Howard and Ruder, 2018), owing to their strong expressive power and less requirement for feature engineering. However, the deeper and more complex the neural model, the more it is essential for them to be trained on substantial amount of training data. Hence, pretraining methods have attracted significant attention (Radford et al., 2018; Devlin et al., 2019); these methods leverage large text corpora to train a model with better generalization properties (Ruder et al., 2019). Recently, several pretraining methods using bidirectional long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) or transformer (Vaswani et al., 2017) have been highly successful in many NLP tasks (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019). These methods first pretrain neural networks on large unlabeled text corpora, and then, finetune the pretrained networks on downstream tasks.

Although pretraining methods have achieved state-of-the-art status on many NLP tasks (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019), their applicability to large-scale classification is yet to be well-studied, in which training data for each category is severely insufficient and distributed unevenly among classification categories. In addition, existing pretraining methods typically need very high capacity and long training time.

To handle large-scale text classification, Kim et al. (2019) have alleviated the problem of the

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 2041–2050

limited amount of training data by utilizing a multi-task learning (Collobert and Weston, 2008; Ruder et al., 2019). The proposed multi-task framework converts the large-scale text classification task to a small-scale text classification task, and learns both tasks simultaneously. They achieve a new state-of-the-art large-scale text classification. However, this scheme may not benefit from the generalization of pretraining because different scale tasks come from a single limited dataset.

In this paper, we develop a novel multi-pretraining framework based on the convolutional neural network (CNN) to handle large-scale text classification. By the multi-pretraining, we mean that the framework simultaneously learns both a self-supervised and a weakly supervised pretraining. Self-supervised learning methods, such as masked language modeling and permutation language modeling, have shown to be effective for improving neural models on many NLP tasks (Devlin et al., 2019; Lan et al., 2020; Yang et al., 2019). However, they usually require excessive computation and training time. Instead of the language model, we newly introduce an out-of-context (OOC) words detection task on the unlabeled data for a self-supervised pretraining. Notably, the OOC words detection task is inspired by the success of learning to spot artifacts (Jenni and Favaro, 2018), one of self-supervised learning methods in computer vision. We hypothesize that the OOC words detection in NLP is analogous to the spotting artifacts in computer vision, thereby helping improve NLP tasks. It turns out that the task indeed learns useful features for text classification by detecting whether or not there are OOC words in a sentence. In order to generate OOC words, we randomly select a word from the sentence or paragraph [1] in corpora and replace it with a random word.

In addition, we propose a weakly supervised pretraining (Dehghani et al., 2017), where labels are obtained automatically with an existing text classification method. To this end, we use the output of an explicit representation model (Wang and Wang, 2016) based on bag-of-words or bag-of-phrases as a weak supervision signal. Notably, weakly supervised labels are highly abstract topics (e.g., *Sports, Health, Computers, Business, etc.*) rather than complete categories (e.g., *Sports/Baseball/MLB/Awards*) in a large-scale taxonomy. The proposed framework involves two steps. The first step is to pretrain the models on the unlabeled large corpora, which learns to spot whether OOC words are used, and a weakly supervised text classification task simultaneously. The second step is to finetune the same model on the labeled large-scale text classification dataset.

We demonstrate the efficacy of our framework on large-scale text classification. The experimental results show that the proposed multi-pretraining scheme yields significantly improved results in large-scale text classification. In summary, our contributions are three-fold:

- We develop a novel multi-pretraining framework based on CNN to handle large-scale text classification, which contains both a self-supervised and a weakly supervised pretraining.
- We propose a new way to simultaneously learn multiple pretraining tasks on nearly-unlimited amount of unlabeled data, and introduce effective finetuning techniques for large-scale text classification.
- We demonstrate the efficacy of the proposed methodology through extensive experiments. The performance evaluation clearly shows that our approach outperforms a dozen of state-of-the-art large-scale text classification methods and is competitive with excessively large pretraining methods.

The remainder of this paper is organized as follows. Section 2 describes the proposed multi-pretraining framework for large-scale text classification. We present the performance evaluation results and in-depth analysis in Section 3 and 4, respectively. We discuss related work in Section 5 and conclude in Section 6.

2 Methodology

In this section, we describe two pretraining methods, a self-supervised pretraining and a weakly supervised pretraining, for large-scale text classification. We propose a novel multi-pretraining to utilize the knowledge obtained from both pretrainings. In addition, we introduce finetuning strategies to maintain the useful knowledge or features. We adopt the CNN which has been very successful in text classification tasks (Kim, 2014; Choi et al., 2019). In particular, a carefully designed CNN outperforms other architectures in large-scale

[1] We split One Billion Word Benchmark corpus into sentences, while splitting other corpora into paragraphs.

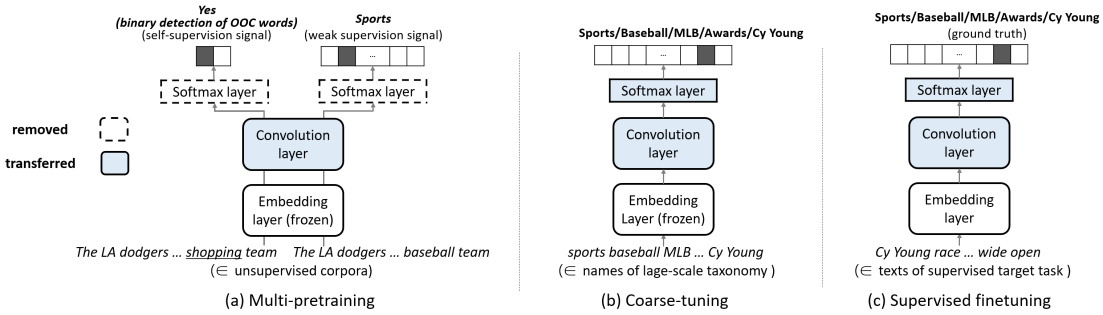

Figure 1: Illustration of architecture for our proposed framework

text classification (Kim et al., 2019) with relatively small number of parameters. Figure 1 shows our proposed framework.

2.1 Base Model

We adopt a popular CNN-based text classification model (Kim, 2014) for our base architecture. This model consists of three layers: word embedding, convolution, and softmax. The word embedding layer transforms a word sequence into a matrix, in which columns are a sequence of vector representations of words. One-dimensional convolution is subsequently performed by taking each dimension of the word embedding as an input channel. Subsequently, the element-wise rectified linear unit function (ReLU) (Nair and Hinton, 2010) is applied for non-linearity. Then, a max-over-time pooling operation (Collobert et al., 2011) is applied over the feature map. We use multiple filters with different window sizes to obtain multiple features. We simply concatenate the results of each filter and obtain the output vector of the pooling operation. The features are passed to the softmax layer to predict the class label.

2.2 Self-supervised Pretraining

Most neural network-based methods suffer from large-scale text classification (Kim et al., 2019) because the training documents for each category are insufficient. Self-supervised pretraining has been highly successful in text classification when only limited training data is available (Devlin et al., 2019; Radford et al., 2018). Language modeling (LM) is one of ideal source tasks (i.e., pretext task) as a self-supervised pretraining. However, we empirically find out that applying the language model pretraining to CNN-based large-scale text classification has no significant performance improvement over training time(, which we shall show in Section 3). Therefore, unlike previous works, we do not use the language model for pretraining. Inspired by the success of learning to spot artifacts (Jenni and Favaro, 2018), instead, we introduce a novel OOC words detection-based self-supervised pretraining method with CNN (denoted as SP-CNN), which is shown in the left-hand side of Figure 1(a). We expect the OOC words detection task to be a useful pretext task for large-scale text classification. We intentionally damage 50% of all sentences or paragraphs in unlabeled corpora U at random, and then predict whether a sentence or paragraph contains any OOC words. In order to create texts with OOC words and self-supervision signal Y_s, we perform the following procedure:

1. Select a word randomly from words, except for stopwords, in a sentence or paragraph, e.g., *The LA Dodgers are a professional* ***baseball*** *team*
2. Replace the word with a random word [2], e.g., *The LA Dodgers are a professional* ***shopping*** *team*

After creating the self-supervision signal, we pretrain the proposed CNN on large corpora (e.g., billions of training examples). The goal of self-supervised pretraining is to learn to detect whether any word is used out-of-context. In the self-supervised pretraining phase, the parameters of network are trained to minimize the following objective:

$$L_{OD}(U) = -\sum_{m}\sum_{n} SS(y_n|x_m) log P(y_n|x_m;\Theta) \quad (1)$$

where y_n and x_m are a possible class (i.e., original or OOC words detection) and a sentence or paragraph in large corpora, respectively. $SS(y_n|x_m)$

[2] We empirically investigated a number of choices for noise distribution and found that the uniform distribution outperformed the unigram distribution.

denotes an automatically annotated class probability of y_n.

2.3 Weakly Supervised Pretraining

While the self-supervised pretraining has the advantage of availability of nearly unlimited amount of data, a weak supervision pretraining has also been applied in NLP and information retrieval (IR) to utilize unsupervised data (Dehghani et al., 2017; Meng et al., 2019). The weak supervision refers to a learning approach that automatically creates its own training data (i.e., weakly annotated set) using an existing unsupervised or supervised approach.

We propose a weakly supervised pretraining based on CNN (denoted as WP-CNN), which is shown in the right-hand side of Figure 1(a). We use a well-performing existing large-scale text classifier based on the *tf-idf* scheme (Lee et al., 2013) to create a weakly annotated set from unlabeled corpora U. This model performs robustly in large-scale text classification, especially for categories with a small number of data. We divide corpora into sentences or paragraphs, and then generate the weak supervision signal Y_w in two ways. In a complete signal, the *tf-idf* scheme based classifier determines one pseudo label (e.g., *Sports/Baseball/MLB/Awards/Cy Young*) that receives the highest score among categories in a large-scale taxonomy. In contrast, in an abstract signal, we use a top-level category of the complete signal. For example, given the *Sports/Baseball/MLB/Awards/Cy Young* as a weak supervision signal, the *Sports* is used as the abstract signal. We expect that the abstract way works better than the complete way, since the abstract signal is of higher quality than the complete signal in terms of accuracy.

After creating the weak supervision signal, we pretrain the proposed CNN on large corpora. The goal of weakly supervised pretraining is to learn pseudo-labels or class probabilities that are generated by the large-scale text classification based on the *tf-idf* scheme. In the weakly supervised pretraining phase, the parameters of network are trained to minimize the following objective:

$$L_{TC}(U) = -\sum_{m}\sum_{k} WS(y_k|x_m) log P(y_k|x_m;\Theta) \quad (2)$$

where y_k and x_m are a possible class (i.e., (top-level) categories in the large-scale taxonomy) and a sentence or paragraph in large corpora, respectively. $WS(y_k|x_m)$ denotes a weakly annotated class probability of y_k.

2.4 Multi-pretraining

To benefit from both pretrainings simultaneously, we develop a novel multi-pretraining framework based on CNN (denoted as MP-CNN). We apply the multi-task learning by spotting OOC words and text classification tasks as two, related tasks. In other words, we add the OOC words detection objective to the model that is trained jointly with the text classification for pretraining. As shown in Figure 1(a), the OOC words detection (i.e., self-supervised pretraining) and text classification (i.e., weakly supervised pretraining) tasks share the same word embedding and convolution layers, while keeping their own private softmax layer.

Given unsupervised corpora U, weak supervision signal Y_w, and self-supervision signal Y_s, the joint pretraining objective L_P is computed by the weighted sum of the OOC words detection objective L_{OD} and the text classification objective L_{TC}, as follows:

$$L_P(U) = \lambda_o L_{OD}(U) + \lambda_t L_{TC}(U) \quad (3)$$

where λ_o and λ_t are the weights for OOC words detection and text classification tasks, respectively. Following Collobert and Weston (2008), the pretraining is performed in a stochastic manner by looping over the following tasks:

1. Select a pretraining task with fixed probabilities.
2. Select a random training example from the corpora.
3. Update the parameters for the selected task by taking a gradient step with respect to this example.
4. Go to 1.

2.5 Finetuning

After pretraining the model, we finetune the pretrained model with two sequential stages (coarse-tuning and supervised finetuning) as shown in Figure 1(b) and 1(c). We first finetune our model with category names of a large-scale taxonomy. We observe that category names clearly represent specific semantics of each category, which are expected to be useful when training data is extremely scarce[3]. For coarse-tuning, we

[3]We experimentally confirmed that the coarse-tuning improved the performance by 1.1%.

convert category names into pseudo sentences (e.g., *Sports/Baseball/MLB/Awards/Cy Young* into "sports baseball MLB Cy Young")

After the coarse-tuning stage, we finetune the pretrained model on the supervised target task, which contains sentences or paragraphs S and corresponding ground truth label Y [4]. To retain the previous knowledge obtained from the pretraining phase, we apply finetuning techniques such as the discriminative finetuning (Howard and Ruder, 2018) to the proposed CNNs.

We finetune each layer with different learning rates. This is driven by the empirical evidence that the layers closer to the input layer contain general features, whereas the layers closer to the last layer contain specific features. Therefore, given a learning rate η, we determine the learning rates η_s, η_c, and η_e, which denote the learning rates of the softmax, convolution, and embedding layer, respectively: $\eta_s = \eta$, $\eta_c = \eta_s/4.5$, and $\eta_e = \eta_c/4.5$, respectively.

We introduce a penalization term, T, that penalizes redundant convolution filters and encourages convolution layers to encode different aspects of input. To this end, we use the dot product of the multiple filter matrix $C_h \in \mathbb{R}^{hd \times n}$ and its transpose, where h and d are the window size and the dimension of the word embedding, respectively, and n is the number of filters with a window size of h. We calculate the penalization term as follows:

$$T = \sum_{h} \| \, C_h {C_h}^T - I \, \|_F \tag{4}$$

Here, $\| \cdot \|_F$ represents the Frobenius norm of a matrix, and C_h denotes a matrix for all convolution filters with a window size of h. The final loss function of the finetuning step is as follows:

$$L = L_{TC}(S) + \gamma T, \tag{5}$$

$$L_{TC}(S) = -\sum_{i} log P(y|x_i; \Theta) \tag{6}$$

where y and x_i are a ground truth category and a text in supervised training data. γ is a hyperparameter.

[4] We found that performing both coarse-tuning and finetuning as multiple objectives in the same stage shows a similar classification performance to the presented three-stage approach.

3 Experiments

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

3.1 Datasets

3.1.1 Unlabeled Large Corpora

To pretrain models in the proposed multi-pretraining framework, we collect the unlabeled datasets including One Billion Word Benchmark (Chelba et al., 2013), WikiText-103 (Merity et al., 2017), and AG News (Zhang et al., 2015). They have 5.02GB of plain text combined.

3.1.2 ODP

For large-scale text classification, we use the RDF dump from the original Open Directory Project (ODP)[5] dataset released on January 8, 2017. 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. To obtain a well-organized ODP taxonomy, we apply heuristic rules (Lee et al., 2013) and build our own large-scale taxonomy with 2,531 categories. Thus, the final training dataset used in our experiments consists of 58,180 webpages.

The ODP test dataset consists of webpages collected from the original ODP. The webpages in each category are randomly divided into training, test, and development sets. We collect 8,661 and 7,830 webpages from 2,531 ODP categories for the test and development datasets, respectively. The ODP datasets are publicly available[6].

3.2 Evaluation Metrics

For the ODP datasets, we use the F_1 measure (Yang, 1999) 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: micro-averaging (Mi-F_1) and macro-averaging (Ma-F_1).

3.3 Baselines

We evaluate the performance of our methods with a dozen of competitor methods. We adopt popular text classification methods and five neural network-based pretraining methods. In our experiments, we use hyperparameters that work best on the ODP

[5] https://curlie.org, http://dmoz-odp.org

[6] https://bit.ly/3j4L6G2

development dataset for each model. We compare the following methods:

- *SPG-CNN*: State-of-the-art large-scale text classification based on a multi-task learning with CNN (Kim et al., 2019).
- *MC*: *tf-idf*-based classification method. *MC* utilizes enriched training data for each category from its descendants (Lee et al., 2013).
- *CNN*: Shallow CNN-based text classification method (Kim, 2014).
- *DPCNN*: Deep pyramid CNN-based text classification method (Johnson and Zhang, 2017).
- *LSTM*: LSTM-based text classification method (Jozefowicz et al., 2015).
- *BiLSTM*: Bidirectional LSTM-based text classification method.
- *Transformer*: Fully-connected attention-based neural network model (Vaswani et al., 2017). We only use the Transformer encoder for large-scale text classification.
- *ULMFiT*: LSTM-based transfer learning for text classification (Howard and Ruder, 2018).
- *GPT*: Pretraining method based on a left-to-right Transformer LM for a wide range of NLP tasks (Radford et al., 2018).
- *BERT(base)*: Pretraining method based on multi-layer bidirectional Transformer encoder for a wide range of NLP tasks (Devlin et al., 2019).
- *XLNet(base)*: State-of-the-art pretraining method across a broad range of NLP tasks (Yang et al., 2019).
- *ALBERT(base)*: Parameter-efficient variant of BERT (Lan et al., 2020).

3.4 Implementation Details

We implement the proposed model and competitor methods using PyTorch (Paszke et al., 2017) and train models in a single machine equipped with an AMD 12-Core processor, 128 GB of RAM, and an NVIDIA GeForce RTX 2080 Ti with 11 GB of RAM. The word embedding layers for all neural network-based models are initialized with the pretrained word embeddings (except for models that do not require pretrained word embeddings). We adopt the publicly available Word2Vec[7] model (Mikolov et al., 2013a,b), which is a popular word embedding technique. The embedding layer is updated during finetuning to improve performance. The other parameters are initialized by Xavier (Glorot and Bengio, 2010).

3.4.1 Pretraining Phase

We pretrain the proposed networks with backpropagation and gradient-based optimization using the Adam update rule (Kingma and Ba, 2015). We set the learning rate with 1e-4, β_1 = 0.9, β_2 =0.99, and linear decay of the learning rate. We use the dropout (Srivastava et al., 2014) rate of 0.7; filter windows of 2, 3, 4, and 5 with 600 filters each; mini-batch size of 64; and L2 weight decay with a lambda of 1e-7. We set λ_o and λ_t to 0.3 and 0.7, respectively.

3.4.2 Finetuning Phase

For finetuning, most model hyperparameters are the same as in pretraining, with the exception of the learning rate and dropout probability. In the finetuning phase, we use Adam update rule with a learning rate of 5e-4 and a dropout rate of 0.75. We set γ to 1e-7. In addition, we use the slanted triangular learning rate schedule for quick convergence (Howard and Ruder, 2018).

3.5 Experimental Results

We first compare a few CNNs with/without pretraining methods on the ODP dataset. From Table 1, we observe that the simple and shallow *CNN* outperforms *DPCNN* over 32.3% and 47.7% in terms of Mi-F_1 and Ma-F_1, respectively. We also compare our two methods for weakly supervised pretraining. In Table 1, *abs-WP-CNN* denotes the weakly supervised pretraining with abstract signals, while *com-WP-CNN* denotes the weakly supervised pretraining with complete signals. Ex-

Model	Mi-F_1	Ma-F_1
CNN	0.581	0.508
DPCNN	0.439	0.344
LP-CNN (ours)	0.593	0.523
SP-CNN (ours)	0.602	0.535
abs-WP-CNN (ours)	0.607	0.543
com-WP-CNN (ours)	0.603	0.535
MP-CNN (ours)	**0.616**	**0.569**

Table 1: Comparison of CNNs with/without pretraining on the ODP dataset.

[7] https://code.google.com/archive/p/word2vec/

pectedly, we observe that *abs-WP-CNN* clearly outperforms *com-WP-CNN*. Thus, we utilize abstract signals for weakly supervised pretraining in the remaining experiments. We further observe that both *SP-CNN* and *WP-CNN* outperform *CNN*. These results clearly demonstrate that both pretraining approaches are effective for large-scale text classification. *MP-CNN* performs better than *SP-CNN* and *WP-CNN*. This result implies that features learned by the OOC words detection and the weakly supervised pretrainings are complementary. Note that, *LP-CNN* in Table 1 denotes the language model pretraining with CNN. *LP-CNN* performs the worst among the CNN-based pretraining methods and performs slightly better than *CNN* over 2.1% and 2.9% in terms of Mi-F_1 and Ma-F_1, respectively[8]. We also observe that *LP-CNN* takes 35x more pretraining time than *SP-CNN*. From these results, we confirm that the pretraining based on language model is relatively less effective in large-scale text classification based on CNN.

Table 2 summarizes the experimental results for large-scale text classification on the ODP test dataset with 2,531 target classes. Notably, *MP-CNN* outperforms all the baselines. *MP-CNN* exhibits improvement compared to *XLNet* over 0.8% and 3.8% in terms of Mi-F_1 and Ma-F_1, respectively. Moreover, we observe that *MP-CNN* outperforms the baselines trained from scratch. In particular, *MP-CNN* performs better than *MC*, which is used to generate weak supervision signals, over 28.6% and 22.1% in terms of Mi-F_1 and Ma-F_1, respectively. Our experimental results show that *CNN* performs the best among methods without pretraining [9] (i.e., against *MC*, *LSTM*, *BiLSTM*, and *Transformer*). We further observe that the average performance of the pretraining methods achieves the improvement of 1.2% over that of state-of-the-art models based on a multi-task learning in terms of Mi-F_1. We also perform the t-test for the classification results, and find that *MP-CNN* results are statistically significant with $p < 0.01$.

Table 3 demonstrates the model size and finetuning time of pretraining methods. For a fair comparison, we use the same hardware resources, and report the finetuning time until each model converges.

[8]We trained both an LM and a weakly supervised text classification task simultaneously, but we did not find any noteworthy performance improvement.

[9]We observe that CNN outperforms MC in the optimal parameter settings we found, contrary to the results in the work (Kim et al., 2019).

	Model	Mi-F_1	Ma-F_1
From scratch	*MC*	0.479	0.466
	CNN	0.581	0.508
	LSTM	0.446	0.374
	BiLSTM	0.468	0.378
	Transformer	0.523	0.508
Pretraining	*SPG-CNN*	0.595	0.524
	ULMFiT	0.594	0.517
	GPT	0.601	0.541
	BERT	0.602	0.556
	XLNet	0.611	0.548
	ALBERT	0.590	0.530
	MP-CNN (ours)	**0.616**	**0.569**

Table 2: Large-scale classification performance on the ODP dataset.

Model	Parameters	Finetuning time
ULMFiT	63,443,486	9h 26m
GPT	85,054,464	8h 10m
BERT	87,591,395	9h 43m
XLNet	94,679,267	8h 42m
ALBERT	9,624,803	10h 13m
MP-CNN (ours)	**8,599,331**	**2h 55m**

Table 3: Number of parameters and finetuning time compared to pretraining models. Parameters in the word embedding layers and softmax layer for pretraining are excluded.

We observe that *MP-CNN* has just 13.6%, 10.1%, 9.8%, 9.1% and 89.3% of parameters in *ULMFiT*, *GPT*, *BERT*, *XLNet*, and *ALBERT*, respectively. We also observe that *MP-CNN* is 3.2x, 2.8x, 3.3x, 3.0x and 3.5x faster than those competitors, respectively. In addition, we report that *MP-CNN* takes half a day to complete the pretraining. Unfortunately, we failed to measure the pretraining time of competitors in our hardware resources, but estimate it to be a few to several months. These results confirm that *MP-CNN* is indeed effective with a reasonable cost of memory and computation.

4 Analysis

We evaluate *MP-CNN* on different numbers of training examples to analyze its usefulness on the limited training data. We split off balanced fractions of the ODP training data and fix the development set. Although *MP-CNN* performs slightly better than *XLNet* when trained on all the training examples (as observed in Table 2), *MP-CNN* outperforms

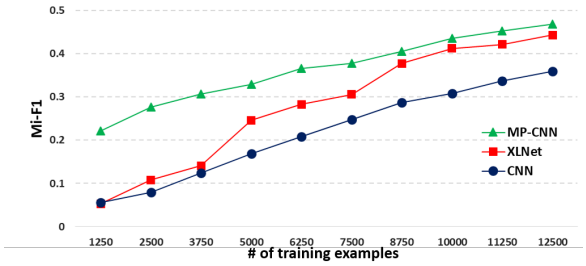

Figure 2: Large-scale classification performance with different numbers of training examples on ODP dataset

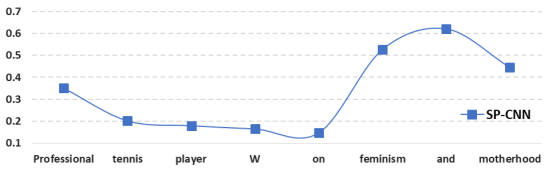

Figure 3: Visualization of the change of predicted score in the presence of OOC words at different time steps. Y-axis represents predicted score, while X-axis represents input words in chronological order

XLNet by a large margin for the limited training data, as shown in Figure 2. In addition, we observe that *MP-CNN* trained with only 1,250 labeled examples outperforms *CNN* trained with 5× more data. These results clearly demonstrate that the proposed multi-pretraining successfully improves the performance of large-scale text classification.

We qualitatively examine the classified categories from *SP-CNN* and *CNN* to analyze why the OOC words detection task improves the performance of large-scale text classification. We select a magazine title "Professional tennis player *W* on feminism and motherhood", which is related to both "Sports" and "Female". Then, we analyze the changes of predicted score in the presence of OOC words at different time steps, which are obtained by *SP-CNN*. The output layer for large-scale text classification of *SP-CNN* returns *Sports/Tennis/Players* up to the fifth word (i.e., on), while *Sports/Tennis/Players/Female* when the sixth word (i.e., feminism) is processed. In contrast, *CNN* keeps returning *Sports/Tennis/Players* until the last word. Interestingly, from Figure 3, we observe that *SP-CNN* captures "feminism" and "motherhood" are semantically different from the previous context. These results illustrate how *SP-CNN* effectively classifies the sentence into the highly specific categories, both "Sports" and "Female". We note that this is consistent with many sentences in the test dataset.

5 Related Work

For large-scale text classification, many techniques have been proposed to handle data sparsity. McCallum et al. (1998) firstly addressed data sparsity on a hierarchical taxonomy. They adopted a statistical technique, called *shrinkage*, to estimate the parameters of data-sparse child categories with their data-rich ancestor categories. Lee et al. (2013) 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 (Kim et al., 2019), they utilize a multi-task learning by treating different scales of text classification as related tasks. They have achieved a new state-of-the-art performance in large-scale text classification. However, these approaches cannot utilize the implicit knowledge from general-domain large corpora.

Neural network-based models have utilized pretrainings to overcome the problem of insufficient training data. Recent neural network-based approaches have utilized the language model pretraining on large text corpora. Radford et al. (2018) pretrained a transformer decoder-based language model and finetuned it using task-aware transformations. It consequently accomplished large gains on natural language understanding tasks. Yet another work (Devlin et al., 2019) used a masked language model based on a bidirectional transformer in pretraining. Very recently, Lan et al. (2020) have suggested finetuning a parameter-efficient variant of transformer pretrained on the masked language model to benefit both the model size and performance. The ideas from transformer-XL (Dai et al., 2019) were integrated into the permutation language model (Yang et al., 2019). Another line of work (Howard and Ruder, 2018) introduced several techniques for finetuning a pretrained language model. To the best of our knowledge, our current work is one of only a few work that applies the pretrainings to large-scale text classification.

6 Conclusion

In this paper, we have developed a novel CNN-based pretraining framework to handle large-scale text classification. Specifically, we pretrain the proposed CNN-based model, which simultaneously learns both the OOC words detection and the text classification task on unlabeled corpora. We have verified the large-scale text classification

performance of our methodology using real-world datasets. Our experimental results confirm that our methodology significantly outperforms a dozen of strong baseline methods. We plan to apply our framework to another NLP tasks, such as sentiment analysis and keyphrase extraction.

Acknowledgment

We would like to thank the anonymous reviewers for their valuable comments. This research was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2020R1A4A1018309), the NRF grant funded by the Korea government (MSIT) (No. 2018R1A2A1A05078380), and Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00079, Artificial Intelligence Graduate School Program(Korea University)).

References

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 Systems with Applications*.

Andrei Broder, Marcus Fontoura, Evgeniy Gabrilovich, Amruta Joshi, Vanja Josifovski, and Tong Zhang. 2007. Robust classification of rare queries using web knowledge. In *SIGIR*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. In *INTERSPEECH*.

Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. 2005. Using odp metadata to personalize search. In *SIGIR*.

Byung-Ju Choi, Jun-Hyung Park, and SangKeun Lee. 2019. Adaptive convolution for text classification. In *NAACL-HLT*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural ranking models with weak supervision. In *SIGIR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Simon Jenni and Paolo Favaro. 2018. Self-supervised feature learning by learning to spot artifacts. In *CVPR*.

Rie Johnson and Tong Zhang. 2017. Deep pyramid convolutional neural networks for text categorization. In *ACL*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *ICML*.

Kang-Min Kim, Yeachan Kim, Jungho Lee, Ji-Min Lee, and SangKeun Lee. 2019. From small-scale to large-scale text classification. In *WWW*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *ICLR*.

Jung-Hyun Lee, JongWoo Ha, Jin-Yong Jung, and SangKeun Lee. 2013. Semantic contextual advertising based on the open directory project. *TWEB*.

Andrew McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. 1998. Improving text classification by shrinkage in a hierarchy of classes. In *ICML*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *AAAI*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *ICLR*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *ICLR (Workshop)*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NeurIPS*.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *Technical report, OpenAI*.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *NAACL-HLT: Tutorials*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Zhongyuan Wang and Haixun Wang. 2016. Understanding short texts. In *ACL: Tutorials*.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Inf. Retr.*

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NeurIPS*.