Japanese Text Classification by Character-level Deep ConvNets and Transfer Learning

Minato Sato, Ryohei Orihara, Yuichi Sei, Yasuyuki Tahara and Akihiko Ohsuga
Graduate School of Information Systems, The University of Electro-Communications, Tokyo, Japan

Keywords: Deep Learning, Temporal ConvNets, Transfer Learning, Text Classification, Sentiment Analysis.

Abstract: Temporal (one-dimensional) Convolutional Neural Network (Temporal CNN, ConvNet) is an emergent technology for text understanding. The input for the ConvNets could be either a sequence of words or a sequence of characters. In the latter case there are no needs for natural language processing that depends on a language such as morphological analysis. Past studies showed that the character-level ConvNets worked well for news category classification and sentiment analysis / classification tasks in English and romanized Chinese text corpus. In this article we apply the character-level ConvNets to Japanese text understanding. We also attempt to reuse meaningful representations that are learned in the ConvNets from a large-scale dataset in the form of transfer learning, inspired by its success in the field of image recognition. As for the application to the news category classification and the sentiment analysis and classification tasks in Japanese text corpus, the ConvNets outperformed N-gram-based classifiers. In addition, our ConvNets transfer learning frameworks worked well for a task which is similar to one used for pre-training.

1 INTRODUCTION

Recently, many deep learning algorithms have achieved high accuracy in media information processing, such as image and speech recognition. Concretely, a deep convolutional neural network (ConvNet, CNN) achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry in ILSVRC-2012 competition (Krizhevsky et al., 2012). Many deep learning approaches in the field of image recognition reuse the ConvNets pre-trained on a very large dataset (e.g. ImageNet (Deng et al., 2009)) as a parameter initialization or feature extractor (Sharif Razavian et al., 2014). This approaches are called transfer learning, in particular, inductive transfer.

A successful example of deep learning in the field of natural language processing is `word2vec` (Mikolov et al., 2013a; Mikolov et al., 2013b) which is a method for producing word embeddings (*i.e.*, low-dimensional and dense vectors).

In the field of natural language processing, text classification is a classic and important task because of many applications. It was important for classic text classification to make "good" features for a classifier by hand. For instance, a language-specific sentiment dictionary for sentiment analysis is often employed and it needs a language-specific morphological analysis to make a Bag-of-Words model or a Bag-of-N-grams model.

There are deep learning approaches for text classification / sentiment analysis. In the early days, stacked denoising autoencoders (SDAs) are used for Amazon review data sentiment analysis (Glorot et al., 2011). Recursive neural networks (RNNs) based on a syntax tree are also used for a similar task (Socher et al., 2013). A syntactic analyzer or a manpower is needed to generate the syntax tree.

Recent approaches are to apply the temporal ConvNets on a sequence of words or a sequence of characters. The former, the word-level ConvNets require the morphological analyzer of a target language. The word embeddings pre-trained by unsupervised learning (*i.e.*, `word2vec` training) improve classification accuracy of the word-level ConvNets. On the other hand, the latter, the character-level ConvNets do not require the morphological analyzer, the syntactic analyzer, etc. Past studies showed that the character-level ConvNets worked well for news category classification and sentiment analysis / classification tasks in English and romanized Chinese text corpus. However, any of other languages has not been studied yet. Additionally, nobody discusses features extracted by the character-level ConvNets although many discus-

Sato M., Orihara R., Sei Y., Tahara Y. and Ohsuga A.
Japanese Text Classification by Character-level Deep ConvNets and Transfer Learning.
DOI: 10.5220/0006193401750184
In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence (ICAART 2017)*, pages 175-184
ISBN: 978-989-758-220-2

sions about that are in the field of image recognition.

This study experiments on Japanese text classification with the ConvNets, and we investigate what the character-level ConvNets extract and the possibility of transfer learning in order to make good use of them.

The remainder of this paper is organized as follows. Section 2 explains related works dealing with the temporal ConvNets for text classification and an overview of transfer learning in the field of image recognition. Section 3 briefly describes the character-level ConvNets and the transfer learning frameworks for experiments. Section 4 provides the dataset description and the experimental results. Section 5 provides the discussions for the experimental results. Finally, section 6 gives a conclusion of this paper.

2 RELATED WORK

2.1 Word-Level ConvNet

The word-level ConvNets approaches which apply the temporal ConvNets on the sequence of words are proposed by Kim (Kim, 2014) and Severyn et al. (Severyn and Moschitti, 2015b; Severyn and Moschitti, 2015a). Kim uses multiple window sizes of convolution filters and experiments on movie sentiment analysis. Severyn et al. experiment on very short text, such as Twitter and SMS. Both of their models use only one temporal convolutional layer and one temporal pooling layer, namely they are relatively shallow.

2.2 Character-Level ConvNet

The character-level ConvNets approaches which apply the temporal ConvNets on the sequence of characters are proposed by Santos et al. (dos Santos and Gatti, 2014; dos Santos et al., 2015) and Zhang et al. (Zhang and LeCun, 2015; Zhang et al., 2015). Santos et al. combine the word-level ConvNet with a pre-trained embedding layer and the character-level ConvNet with a randomly initialized embedding layer for a twitter sentiment analysis. Their model is also relatively shallow, like past studies of the word-level ConvNets. Zhang et al. are the first to apply the temporal ConvNets only on the sequence of characters. An acceptable input of their model is the sequence of one-hot (or 1-of-m) encoded characters. In the case of English dataset, the dimensionality of one-hot encoding is the sum of the number of letters from “a” to “z”, digits “0” to “9” and signs (*e.g.*, “?”, “!”, etc), hence at most 70. In the case of Chinese dataset, the past study uses the romanization by Python library `pypinyin`. Hence, the dimensionality of one-hot encoding for Chinese input is the same as in the case of English dataset. Their model is a deep architecture which is composed of 6 convolutional-pooling layers and 3 fully-connected layers.

2.3 Transfer Learning based on ConvNet in Image Recognition

In image recognition of deep learning, many researchers study reuse of the ConvNets pre-trained on ImageNet (Deng et al., 2009). Razavian et al. reuse the pre-trained ConvNets as a feature extractor (Sharif Razavian et al., 2014), and they apply linear SVM classifier on extracted features. One of other approaches is replacing weights of the output layer of the pre-trained ConvNets with randomly initialized weights and re-training the whole weights of the ConvNets on a new task (Girshick et al., 2014; Agrawal et al., 2014). This approach is called fine-tuning. In general, accuracy of the ConvNets has a strong tendency to depend on the initial weights. Weight initialization by the pre-trained ConvNets produces better results than random weight initialization . In particular, if the number of samples of a training dataset is small, good initial weights prevent the ConvNets from overfitting and enhance a generalization ability of the ConvNets.

3 CHARACTER-LEVEL ConvNet

3.1 Overview

This section shows key modules for applying the temporal ConvNets on the character-level input (dos Santos and Gatti, 2014; dos Santos et al., 2015; Zhang and LeCun, 2015; Zhang et al., 2015).

3.1.1 Input Representation

We employ two types of input representation for experiments. One is a simple one-hot representation which follows the past study. Another is a distributed representation (*i.e.*, a character-level embedding) in order to omit troublesome romanization processing in the case of Japanese dataset.

Given a sentence composed of N characters $\{c_1, c_2, \cdots, c_N\}$, we first transform each character c_n into the d-dimensional one-hot representation which has value 1 at index c_n and zero in all other positions. Hence, the sentence composed of the N characters is transformed into $\{r_1, r_2, \cdots, r_N\} \in \mathbb{R}^{d \times N}$.

Character-level embeddings require an additional transformation. The character-level embeddings are encoded by column vectors in an embedding matrix $W^e \in \mathbb{R}^{d^e \times d}$ which is a parameter to be learned. We transform each one-hot character vector r_n into the d^e-dimensional character-level embedding $r_n^e \in \mathbb{R}^{d^e}$ by using the matrix-vector product:

$$r_n^e = W^e r_n. \tag{1}$$

The dimensionality of the character-level embedding d^e is a hyper-parameter to be chosen by the user. In the case of the character-level embeddings, the sentence composed of the N characters is finally transformed into $\{r_1^e, r_2^e, \cdots, r_N^e\} \in \mathbb{R}^{d^e \times N}$.

In the following explanation, the vector s_n represents either the vector r_n or the vector r_n^e.

3.1.2 Temporal Convolution

A vector $z_n \in \mathbb{R}^{d \times k}$ applied to temporal convolution with a filter of window size k is defined as a concatenation of the one-hot representations:

$$z_n = \left(s_{n-(k-1)/2}, \cdots, s_{n+(k-1)/2}\right)^T. \tag{2}$$

Hence, the output value of temporal convolution by the ith filter is as follows:

$$\left[u_n\right]_i = \left[Wz_n + b\right]_i \tag{3}$$

where $W \in \mathbb{R}^{f \times d \times k}$ is the weight matrix composed of the f convolution filters and $b \in \mathbb{R}^f$ is the bias vector. W and b are parameters to be learned.

3.1.3 Temporal Pooling

Let us define $M = N - (k-1)$ and temporal max-pooling size is p. The output value of temporal convolution by the ith filter can be describe as $(v_1, v_2, \cdots, v_M)_i \in \mathbb{R}^{N-(k-1)}$. Hence, an applied range of temporal max-pooling $\left[y_m\right]_i$ is as follows:

$$\left[y_m\right]_i = \left(v_{m-(p-1)/2}, \cdots, v_{m+(p-1)/2}\right)_i. \tag{4}$$

The dimensionality of the output matrix of the temporal max-pooling layer is $f \times (N-(k-1))/p$.

3.2 Model Design

We prepare shallow ConvNets and deep ConvNets. The sequential information in the text whose length is larger than the window size cannot be taken into consideration with the shallow ConvNets. On the other hands, multiple convolution and pooling enable the deep ConvNets to take those information into consideration.

The alphabet set used in the models of one-hot representation consists of the following 68 characters.

```
abcdefghijklmnopqrstuvwxyz0123456789
,;.!?:'/\|_@#$%ˆ&*˜‘"+-=<>()[]{}
```

The input sentence length (*i.e.*, the number of input characters) is fixed to 1014. Hence, in the case of the one-hot representation, the dimensionality of the input matrix is 68×1014. The dimensionality of the character-level embedding is set to 50. Hence, in the case of the character-level embedding, the dimensionality of the input matrix is 50×1014. In addition, ReLU (Nair and Hinton, 2010) is applied to all the layers except for the output layer. For the network training, momentum SGD (Bengio et al., 2013) is carried out via backpropagation. Then, the mini-batch size is set to 50 and the momentum is set to 0.9 and initial learning rate is set to 0.01 which is halved every 3 epochs for 10 times. The weights of all the models are initialized by "Xavier initialization" (Glorot and Bengio, 2010).

3.2.1 Deep Model

Zhang et al. design two models of 9 layers deep neural networks with 6 convolutional layers and 3 fully-connected layers. One of the two models has a large number of the convolution filters (Large-C6FC3), while another one has a small number of those (Small-C6FC3). Cn_1FCn_2 refers to a network with n_1 convolutional layers and n_2 fully-connected layers. Table 1 and Table 2 list the configurations of the deep models of Zhang et al. The former lists the configurations of the 6 convolutional layers of the deep models and the latter lists the configurations of the 3 fully-connected layers of the deep models. Each column of Table 1 indicates the index of the layers, the number of convolution filters of the large model, the number of those of the small one, windows size of the convolution filters and max-pooling size. Each column of Table 2 indicates the index of the layers, the number of the output units of the large model and the number of those of the small one. The 9th layer is the output layer, hence the number of units of that de-

Table 1: Configurations of the 6 convolutional layers of Small-C6FC3 and Large-C6FC3.

Layer	Large Frame	Small Frame	Window	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Table 2: Configurations of the 3 fully-connected layers of Small-C6FC3 and Large-C6FC3.

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the Problem	

pends on a problem (*e.g.*, if we deal with a sentiment polarity classification problem, the number of those is two.). To regularize the network, dropout (Srivastava et al., 2014) is applied to between the 3 fully-connected layers with a probability of 0.5.

3.2.2 Shallow Model

We build the shallow character-level ConvNets for comparing with the deep models of Zhang et al. Table 3 and Table 4 list the configurations of the shallow models like Table 1 and Table 2.

Table 3: Configurations of the convolutional layer of Small-C1FC1 and Large-C1FC1.

Layer	Large Frame	Small Frame	Kernel	Pool
1	1024	256	7	1008

Table 4: Configurations of the fully-connected layer of Small-C1FC1 and Large-C1FC1.

Layer	Output Units Large	Output Units Small
2	Depends on the Problem	

3.3 Character-level ConvNets for Transfer Learning

A target task in this section is a text classification task in a relatively small dataset. To prevent the ConvNets from overfitting, we train the ConvNets on a very large dataset and reuse them with transfer learning frameworks.

We employ Small-C6FC3 model for experiments dealing with transfer learning. The following three transfer learning frameworks are compared each other:

Scratch. The model without pre-training on the very large dataset.

Pre-trained Feature. We initialize the 6 convolutional layers of the model with those of the pre-trained model, and freeze those layers for training. Thus, the pre-trained model is used for the feature extractor and only the 3 fully-connected layers are trained through backpropagation.

Fine-tuning. We initialize all the layers without the output layer of the model with those of the pre-trained model and do not freeze any of the layers for training.

4 EXPERIMENTS

4.1 Baseline Methods

We use a Bag-of-Words model and a Bag-of-N-grams model as baseline methods. We employ a multinomial logistic regression for tf-idf features based on a dictionary created by the following methods.

Bag-of-Words. In the case of English dataset, preprocessing (*e.g.*, removing stopwords) is carried out by Python library `gensim`. In the case of Japanese dataset, Japanese morphological analysis is carried out by `MeCab` (Kudo et al., 2004). For both of English dataset and Japanese dataset, we use the most frequent 5000 words as the dictionary.

Bag-of-N-grams. In the case of a English dataset, we use the most frequent n-grams (up to 5-grams) as the dictionary. In the case of Japanese dataset, Japanese romanization is carried out by "Kanji Kana Simple Inverter" (`KAKASI`[1]). We use the most frequent n-grams (up to 5-grams) that are romanized as the dictionary.

4.2 Datasets and Results for Japanese Text Classification

This study employs two types of task for each of English and Japanese with reproduction of the past study in mind. One of the tasks is news categorization, while another one is sentiment analysis.

4.2.1 AFPBB Dataset

We collected Japanese news articles including titles, texts and categories from AFPBB News[2] for the Japanese news categorization task. Table 5 describes an overview of the corpus. The corpus contains 79,778 articles from May 2006 to May 2016. The categories of the corpus are composed of "Lifestyle" , "Politics" , "Science" and "Sports". We sampled 12,000 articles for each of the categories as a training dataset while we sampled 500 articles for each of the categories as a validation dataset and a test dataset. The romanization is carried out for the input for the Bag-of-N-grams and the one-hot character-level ConvNet.

The results for the AFPBB dataset are shown in Table 6. From the point of view of classification accuracy, the best model is the Bag-of-Words model. The

[1] http://kakasi.namazu.org
[2] http://www.afpbb.com/

Table 5: Overview of AFPBB dataset.

Category	Total	Train	Validation	Test
Lifestyle	21,927	12,000	500	500
Politics	18,221	12,000	500	500
Science	13,069	12,000	500	500
Sports	26,561	12,000	500	500

results indicate that C1FC1 one-hot models outperform C6FC3 one-hot models, and C6FC3 embedding models outperform C1FC1 embedding models. We suppose C6FC3 one-hot models fall into overfitting because their expressive power derived from the deep architecture is too much for the small AFPBB dataset and the deteriorated input information due to the romanization. On the other hands, we suppose that the intact input information and the relatively deep architecture improve the accuracy of C6FC3 embedding models. However, none of the ConvNets outperform the Bag-of-Words model because they fail to learn features as good as the dictionary from the relatively small dataset.

Table 6: Classification accuracy for AFPBB dataset.

Model	Accuracy
Bag-of-Words	**0.947**
Bag-of-N-grams	0.926
Small-C1FC1 one-hot	0.9385
Large-C1FC1 one-hot	0.941
Small-C1FC1 embedding	0.916
Large-C1FC1 embedding	0.9225
Small-C6FC3 one-hot	0.9120
Large-C6FC3 one-hot	0.9295
Small-C6FC3 embedding	0.9365
Large-C6FC3 embedding	0.9395

4.2.2 Rakuten Market Review Dataset

We obtained a Rakuten[3] market review dataset from the Informatics Research Data Repository of National Institute of Informatics[4]. We created a subset of it by extracting only reviews with ★1 and ★5 and written between April 2012 and December 2012 to use in the Japanese sentiment analysis task. Among sentiment analysis tasks, this study deals with a sentiment polarity classification (a task for classifying a input text into positive or negative). We sampled 680,000 reviews from the Rakuten market review dataset as shown in Table 7. We created the dictionary for the Bag-of-Words and the Bag-of-N-grams from 200,000 reviews which are randomly sampled from the training dataset because of the limitations of memory.

Table 7: Overview of Rakuten market review dataset.

Polarity	Total	Train	Validation	Test
Positive(★5)	11,434,454	300,000	20,000	20,000
Negative(★1)	370,160	300,000	20,000	20,000

Table 8 shows the results. We had two assumptions. One is that the sequential information improves the accuracy for the sentiment analysis task. Another assumption is that the words or the N-grams are more important than the sequential information in the news categorization task. However, simple C1FC1 models are not inferior to C6FC3 models which is able to take the sequential information into consideration. On the contrary, one of the shallow models, Large-C1FC1 with the character-level embedding is the best model. C6FC3 models outperform C1FC1 models in the case of the one-hot representation, while C1FC1 models outperform C6FC3 models in the case of the character-level embedding. Since a normal Japanese sentence is shorter than a romanized Japanese sentence, the applied range of convolution or max-pooling differs between the normal Japanese sentence and the romanized Japanese sentence. Thus, the Japanese character-level embedding enables the ConvNets to have more broader viewpoints than the romanized one-hot representation. We suppose that the broader viewpoints have provided important information in the sentiment analysis task, which usually comes from the sequential information.

Table 8: Classification accuracy for Rakuten market review dataset.

Model	Accuracy
Bag-of-Words	0.95475
Bag-of-N-grams	0.950975
Small-C1FC1 one-hot	0.958525
Large-C1FC1 one-hot	0.963725
Small-C1FC1 embedding	0.967675
Large-C1FC1 embedding	**0.969875**
Small-C6FC3 one-hot	0.9637
Large-C6FC3 one-hot	0.966825
Small-C6FC3 embedding	0.966925
Large-C6FC3 embedding	0.9689

4.2.3 AG News Dataset

We obtained AG's corpus of news articles (Del Corso et al., 2005; Gulli, 2005) from Gulli's website[5] for

[3]Rakuten, Inc. is one of the largest Japanese electronic commerce and Internet companies based in Tokyo, Japan.

[4]http://www.nii.ac.jp/dsc/idr/en/rakuten/rakuten.html

[5]https://www.di.unipi.it/∼gulli/AG_corpus_of_news_articles.html

English categorization task. We sampled 400,000 articles which belong to any of four largest categories from the corpus for the training dataset, the validation dataset and the test dataset as shown in Table 9. We created the dictionary for the Bag-of-Words and the Bag-of-N-grams from the random sampled 120,000 reviews because of the limitations of memory.

Table 9: Overview of AG news dataset.

Category	Total	Train	Validation	Test
World	186,674	90,000	5,000	5,000
Sports	118,103	90,000	5,000	5,000
Business	134,223	90,000	5,000	5,000
Sci/Tech	153,595	90,000	5,000	5,000

Table 10 shows the results. The best model is Large-C1FC1. There is, however, not much different between all the models, because all the models could extract key words as features.

Table 10: Classification accuracy for AG news dataset.

Model	Accuracy
Bag-of-Words	0.8689
Bag-of-N-grams	0.87665
Small-C1FC1	0.8701
Large-C1FC1	**0.8849**
Small-C6FC3	0.87095
Large-C6FC3	0.87925

4.2.4 Amazon Review Dataset

We obtained an Amazon review dataset (McAuley et al., 2015b; McAuley et al., 2015a) including eight categories (“Books”, “Electronics”, etc) from the Stanford Network Analysis Project (SNAP)[6] for the English sentiment analysis task. We sampled 760,000 reviews whose rating is ★1 or ★5 from this dataset for the training dataset, the validation dataset and the test dataset as shown in Table 11. We created the dictionary for the Bag-of-Words and the Bag-of-N-grams from the random sampled 200,000 reviews because of the limitations of memory.

Table 11: Overview of Amazon review dataset.

Polarity	Total	Train	Validation	Test
Positive(★5)	8,829,533	300,000	40,000	40,000
Negative(★1)	653,333	300,000	40,000	40,000

Table 12 shows the results. The best model is Large-C6FC3. Since the dictionary based on the frequent words may not contain important key words for the sentiment analysis, the flexible ConvNets are proved to be more accurate than the Bag-of-Words and the Bag-of-N-grams by a 3-5% margin.

[6]https://snap.stanford.edu/data/web-Amazon.html

Table 12: Classification accuracy for Amazon review dataset.

Model	Accuracy
Bag-of-Words	0.882175
Bag-of-N-grams	0.881275
Small-C1FC1	0.91125
Large-C1FC1	0.925425
Small-C6FC3	0.92155
Large-C6FC3	**0.931925**

4.3 Datasets and Results for Transfer Learning

We construct a large-scale dataset including sixteen categories which are shown in Table 13 for pre-training from the Amazon review dataset. We call it a pre-training dataset. Table 14 shows an overview of the pre-training dataset. The test classification accuracy of Small-C6FC3 for the large-scale dataset is 0.9168.

Table 13: Sixteen categories for pre-training.

Category
Apps for Android
Automotive
Baby
Beauty
Books
Clothing Shoes and Jewelry
Digital Music
Grocery and Gourmet Food
Health and Personal Care
Office Products
Patio Lawn and Garden
Pet Supplies
Sports and Outdoors
Tools and Home Improvement
Toys and Games
Video Games

Table 14: Overview of Amazon review dataset for pre-training.

Overall	Polarity	Train	Validation	Test
★5	Positive	400,000	40,000	40,000
★4		200,000	20,000	20,000
★2	Negative	200,000	20,000	20,000
★1		400,000	40,000	40,000

We choose “Movies and TV”, “Electronics”, “Home and Kitchen” category to construct the small-scale datasets for the target task. We call it a target dataset. It should be noted that the pre-training dataset does not include these three categories. Table 15 shows an overview of the target dataset. The

ratio of the number of the reviews for each of ★1, ★2, 4 and ★5 rating is 2:1:1:2, namely this ratio is the same ratio as the large-scale dataset for the pre-training as shown in Table 14.

Table 15: Overview of the dataset for main task of transfer learning. To be specific, "Movies" indicates "Movies and TV" category and "Home" indicates "Home and Kitchen" category.

	Movies	Electronics	Home
Train	150,000	150,000	60,000
Validation	30,000	30,000	9,000
Test	30,000	30,000	9,000

Table 16 shows the results. The results indicate that the Fine-tuning is the best method for all the datasets. Additionally, the smaller the dataset gets, the less accurate the Scratch framework gets. From these facts, making good use of the pre-trained ConvNets is very effective for the prevention of overfitting and the enhancement of the generalization ability.

Table 16: The results of experiments for transfer learning on the Amazon review dataset. Each of numbers indicates the classification accuracy.

Model\Dataset	Movies	Electronics	Home
Bag-of-Words	0.85503	0.86033	0.8524
Bag-of-N-grams	0.85603	0.8755	0.8693
Scratch	0.85827	0.87854	0.8581
Pre-trained feature	0.88697	0.88183	0.8988
Fine-tuning	**0.8992**	**0.90523**	**0.9123**

In addition, we used a IMDb review dataset (Maas et al., 2011) for the target dataset. The task of this dataset is also the sentiment polarity classification task. This dataset was collected 50,000 polarized reviews from the Internet Movie Database[7]. A negative review has a score 4 or less out of 10, while a positive review has a score 7 or more out of 10. The default training dataset has 25,000 reviews, while the default test dataset has 25,000 reviews. We randomly sampled 2,500 reviews for the validation dataset from the default training dataset. Hence, we could use only 22,500 reviews for the training dataset. A polarity ratio of all the datasets is 1:1. Since we have an assumption that a characteristic (*e.g.*, scale, a topic) of the pre-training dataset influences the results of transfer learning, we construct another pre-training dataset including only "Movies and TV" category which is closely related topic to the IMDb review dataset.

Table 17 shows the results. The Scratch framework falls into overfitting because of its depth and the smallness of the IMDb review dataset. The transfer learning frameworks using the pre-trained ConvNets outperform the classic text classification methods. Furthermore, the framework using the ConvNets pre-trained on the large-scale dataset including the various categories outperforms the framework using the ConvNets pre-trained on the small-scale dataset including only "Movies and TV" category which is semantically similar topic to the target dataset.

Table 17: The results of experiments for transfer learning on the IMDb review dataset. Each of numbers indicates the classification accuracy. (Various & Large) implies that each of models is pre-trained on the large-scale pre-training dataset including various categories (Table 13). (Movies & Small) implies that each of models is pre-trained on 150,000 Amazon reviews including only "Movies and TV" category.

Model	Accuracy
Bag-of-Words	0.81184
Bag-of-N-grams	0.83276
Scratch	0.75288
Pre-trained feature (Various & Large)	**0.87408**
Fine-tuning (Various & Large)	0.87396
Pre-trained feature (Movies & Small)	0.85156
Fine-tuning (Movies & Small)	0.85256

5 DISCUSSIONS

5.1 Features Extracted by Character-level ConvNet

Fig.1 is a visualization of one filter weight matrix in the first layer of the Small-C6FC3 trained on the Amazon review dataset. Vertical direction of the visualization corresponds to window size, while horizontal direction of that corresponds to the dimensionality of one-hot encoding. In the visualization, white indicates large positive values, and black indicates large negative values, and gray indicates values close to zero. The weight matrix corresponding to letters from "a" to "z" has large variances for the vertical direction comparing with the weight matrix corresponding to other characters. This fact indicates that the ConvNet has learned to care more about the variations in letters than other characters. This phenomenon is observed in other filters as shown in Fig.2. The visualization is consistent with one shown by Zhang et al.

On the other hands, the visualization of filters from Small-C1FC1 trained on AFPBB dataset, arranged like Fig.2, is shown in Fig.3. The phenomenon is not observed in the filters of Small-C1FC1 trained on the AFPBB dataset. We suppose that these filters could not extract general features because of the smallness of the AFPBB dataset.

[7]http://www.imdb.com/

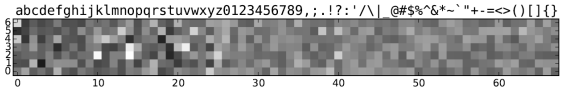

Figure 1: Visualization of one filter weight matrix in the first layer of the Small-C6FC3 trained on the Amazon review dataset.

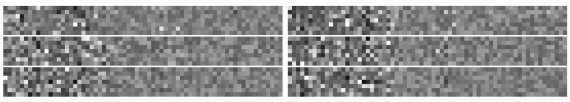

Figure 2: Visualization of random sampled filter weights in the first layer of the Small-C6FC3 trained on the Amazon review dataset.

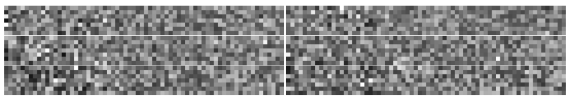

Figure 3: Visualization of random sampled filter weights in the first layer of the Small-C1FC1 trained on the AFPBB dataset.

The visualization of the filter weight matrix of the ConvNet is shown in the past study. It is, however, hard to understand what the ConvNet extracts from the input text. Therefore, we attempt to investigate N-gram features to which a convolution filter of the ConvNet strongly responds. Since the window size of the first layer of all the ConvNets in this study is set to 7, we measure the output value of a convolution filter for each of 7-grams and make a ranking based on the value as shown in Table 18. The filter is trained on the AG news dataset. In order to qualitatively evaluate the result, we extract articles which have these 7-gram strings as its substring. We find that many of these articles have "save (or above or raise) $(the number)" as its substring. Actually, many of these articles are related to corporate cost-cutting, a sharp rise in oil prices, etc. Furthermore, for the quantitative evaluation, we extract articles which match the regular expression corresponds to the disjunction of all the 7-grams shown in Table 18 and count the number of the matched articles belong to each category. The result is 22 for "Business", 0 for "Sports", 1 for "World" and 3 for "Sci/Tech". The result indicates that the feature extraction by the character-level ConvNets works well for text understanding. We suppose that the filter is able to represent at least $5 \times 4 \times 3 \times 1 \times 1 \times 3 \times 3 = 540$ kinds of 7-grams.

5.2 Transfer Learning in Text Classification

C6FC3 models have a large number of parameters. It is hard for the deep ConvNets to learn appropriate parameters which maximize the generalization ability if the number of the samples of the training dataset is small. Since all the datasets in the experiments in Section 4.3 are relatively small-scale, the Scratch frameworks, in other words, the vanilla deep ConvNets are almost the same or less accurate than the Bag-of-Words model and the Bag-of-N-grams model. The transfer learning frameworks, however, outperform the Bag-of-Words model and the Bag-of-N-grams model.

Table 18: Ranking of the output values of the convolution of one filter and 7-grams. The filter is trained on the AG news dataset.

7-gram	Convolution Ouptut Value
`ave $70`	1.02668
`ove $70`	0.98627
`ave $10`	0.97948
`ts: $10`	0.96571
`ise $10`	0.95659
`the $19`	0.95559
`9;s $10`	0.94505
`ove $10`	0.93909
`ave $72`	0.93072
`ave $20`	0.93069

In addition, the experimental results on the IMDb review dataset imply that scale of the pre-training dataset is more important than a topic similarity between the pre-training dataset and the target dataset.

6 CONCLUSION

This study improves the classification accuracy by applying the character-level ConvNets to a large-scale Japanese text corpus in comparison with classic text classification methods. We analyze the features extracted by the character-level ConvNets. The result of the analysis shows that one of the filters of the convolutional layer of the ConvNet could represent multiple N-grams. In addition, we provide the possibility of transfer learning by the ConvNets for text classification. We reuse the ConvNets pre-trained on the large-scale dataset to initialize the weights of the ConvNets for a target task which consists of relatively small dataset. This transfer learning framework improves the generalization ability and prevents from overfitting for the target task just like in the field of image recognition.

As future work, we would like to investigate transfer learning from a task to another task (*e.g.*, from a categorization task to a sentiment analysis task.). Additionally, we would like to pre-train the Japanese character-level embedding layer with

a character-level skip-gram model or a Continuous Bag-of-Characters model inspired by the Continuous Bag-of-Words (CBoW) model.

ACKNOWLEDGEMENTS

This work was supported by JSPS KAKENHI Grant Numbers 26330081, 26870201, 16K12411. We use the Rakuten dataset which is provided by the National Institute of Informatics (NII) according to the contract between NII and Rakuten, Inc. We would like to thank NII and Rakuten, Inc.

We use Python library `Keras`(Chollet, 2015) for building the neural networks and use Python library `Theano`(Theano Development Team, 2016) as backend engine of `Keras`. We would also thank the developers of `Theano` and `Keras`.

REFERENCES

Agrawal, P., Girshick, R., and Malik, J. (2014). Analyzing the Performance of Multilayer Neural Networks for Object Recognition. In *the Proceedings of the 13th European Conference on Computer Vision (ECCV)*, ECCV '14, pages 329–344.

Bengio, Y., Boulanger-Lewandowski, N., and Pascanu, R. (2013). Advances in Optimizing Recurrent Networks. In *the Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, ICASSP '13.

Chollet, F. (2015). Keras.

Del Corso, G. M., Gullí, A., and Romani, F. (2005). Ranking a Stream of News. In *the Proceedings of the 14th International Conference on World Wide Web (WWW)*, WWW '05, pages 97–106.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *the Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '09.

dos Santos, C. and Gatti, M. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts. In *the Proceedings of the 25th International Conference on Computational Linguistics (COLING)*, COLING '14, pages 69–78.

dos Santos, C. N., Xiang, B., and Zhou, B. (2015). Classifying Relations by Ranking with Convolutional Neural Networks. In *the Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, ACL '15, pages 626–634.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *the Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '14.

Glorot, X. and Bengio, Y. (2010). Understanding the Difficulty of Training Deep Feedforward Neural Networks. In *the Proceedings of the 13rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, AISTATS '10.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain Adaptation for Large-scale Sentiment Classification: A Deep Learning Approach. In *the Proceedings of the 28th International Conference on Machine Learning (ICML)*, ICML '11.

Gulli, A. (2005). The Anatomy of a News Search Engine. In *International Conference on World Wide Web (WWW) Special interest tracks and posters*, WWW '05, pages 880–881.

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

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet Classification with Deep Convolutional Neural Networks. In *the Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS)*, NIPS '12, pages 1097–1105.

Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying Conditional Random Fields to Japanese Morphological Analysis. In *the Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP '04, pages 230–237.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. In *the Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL HLT)*, ACL-HLT '11, pages 142–150.

McAuley, J., Pandey, R., and Leskovec, J. (2015a). Inferring Networks of Substitutable and Complementary Products. In *the Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, KDD '13, pages 785–794.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015b). Image-Based Recommendations on Styles and Substitutes. In *the Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, SIGIR '13, pages 43–52.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed Representations of Words and Phrases and their Compositionality. In *the Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS)*, NIPS '13, pages 3111–3119.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. In *the Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, NAACL-HLT '13, pages 746–751.

Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *the Proceedings of the 27th International Conference on Machine Learning (ICML)*, ICML '10, pages 807–814.

Severyn, A. and Moschitti, A. (2015a). Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *the Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, SIGIR '15, pages 959–962.

Severyn, A. and Moschitti, A. (2015b). UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *the Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval)*, SemEval '15, pages 464–469.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, CVPR '14.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *the Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, EMNLP '13, pages 1631–1642.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Theano Development Team (2016). Theano: A Python Framework for Fast Computation of Mathematical Expressions. *arXiv e-prints*, abs/1605.02688.

Zhang, X. and LeCun, Y. (2015). Text Understanding from Scratch. *CoRR*, abs/1502.01710.

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. In *the Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS)*, NIPS '15, pages 649–657.