CNN Architecture Predicting Movie Rating from Audience's Comments Written in Korean

HyeongChan Kim, Heung-Seon Oh, and Duksu Kim
School of Computer Engineering, Korea University of Technology and Education

ABSTRACT

Recently, convolution neural network (CNN) being widely adopted for various field including natural language processing (NLP). In this paper, we present a CNN architecture that predicts a rating score for a movie from an audience's comment. Especially, we aim to handle comments written in Korean and containing lots of unnormalized texts. Our prediction architecture includes a CNN model for sentence classification that is one of the basic operations in NLP. We design a shallow-and-wide style CNN model proposed in a prior work (i.e. TextCNN), but we further improve the classification performance by using character embedding instead of word embedding and employing an architectural unit (i.e. SE block) that captures spatial correlation among features. We also propose a novel scoring function that translates classification results of the network to rating score for a movie. In our experiment with a movie review dataset, we achieved a low MSE (e.g., 3.3841) compared with a prior method and these results demonstrate the accuracy and usefulness of our prediction architecture.

Ⅰ. Introduction

Convolution neural network (CNN) is one of the most well-known and most effective networks for handling visual tasks [12]. Recently, CNN also has employed in non-visual tasks like speech recognition [5], natural language processing (NLP) [6], and so on. Sentence classification is a basic operation in NLP and a well-designed network can be employed in many applications like classifying the class of texts, e.g., predicting a rating for a movie from reviewing texts. Some recent works have demonstrated that a simple CNN model performs well for sentence classification tasks [10]. However, most of those CNN models were designed for normalized texts written in English.

Contributions: In this paper, we introduce a novel prediction system that translates an audience's comment written in Korean for a movie to a rating for the movie. Our prediction architecture includes a CNN model for sentence classification. Although we design the CNN model based on a prior work, TextCNN [10], we further improve the model to efficiently handle our target datasets that have different characteristics with target datasets of prior approaches in two perspectives; 1) sentences (i.e. comments) are written in Korean and 2) that contains lots of slang (e.g, a set of consonants) and incomplete sentences. To handle such unnormalized Korean sentences, we employ a character embedding

Fig. 1. Examples of movie reviews and ratings in the NAVER movie site.

scheme rather than word embedding in the pre-processing stage (Sec. Ⅲ-1). We also improve the representational power of our CNN network with SE (Squeeze-and-Excitation) blocks that captures spatial correlation among features efficiently (Sec. Ⅲ-2). Finally, we introduce a novel scoring function that translates the sentence classification results to a rating at the end of our network (Sec. Ⅲ-3).

To demonstrate the performance of our architecture, we have applied it to a movie review dataset, NAVER movie dataset (Fig. 1). To show the benefits of our approach, we also have implemented five different models including two prior models and three variations of our methods. Compared to the prior models, all variations of our method show lower error rate in terms of mean square error (MSE). These result mainly thanks to our well-designed CNN model with SE blocks. In our experiment with the NAVER movie dataset in which the rating range is from 1 to 10, our method reduces MSE up to 3.3841 and it means that the difference between the true rating score and the prediction result is higher or less than 1.83 in most case. These results validate the usefulness of our prediction architecture.

Ⅱ. Related Work

Convolution neural network (CNN) is a network consisting of convolution layers and pooling layers. Each convolution layer generates a feature map that represents spatial connectivity and then, a pooling layer collects the features [12]. Since CNN reduces the dimensionality of network effectively compared with a fully-connected network while maintaining features of multidimensional data (e.g., image), it has been widely employed and prior works have proved its usefulness and effectiveness in visual task [12,16].

CNN models for sentence classification: Recently, CNN has also applied for other applications including sentence classification in NLP. Kim [10] introduced a CNN architecture, TextCNN, for sentence classification. TextCNN uses multiple convolution layers having a different kernel size in a parallel manner. With this shallow-and-wide CNN model, they achieved a comparable performance with more sophisticated models. Different from TextCNN, Conneau et al. [7] proposed a CNN architecture that stacks lots of convolution layers deeply (e.g., twenty-nine layers) for sentence classification. They reported their very deep CNN model (VDCNN) shows better results than prior state-of-the-art methods. As a recent work, Le et al. [13] studied those two types of CNN models for sentence classification and sentiment analysis. They reported that a shallow-and-wide CNN model (e.g., TextCNN) shows a comparable performance with a very deep model (e.g., VDCNN).

Based on their observation, we design our sentence classification network as a form of shallow-and-wide CNN since it has fewer hyper-parameters and takes less time for training than a very deep model.

Sentence embedding: A basic form of the input for a neural network is a vector and we need to convert the input data (e.g., picture and text) as a vector. This embedding process is performed in a pre-processing stage and there are various embedding schemes according to the type of inputs and the goal of the network. For sentence classification, morphological analysis is usually utilized for such embedding process [14]. Part-Of-Speech (POS) tagging is one of the widely used to make morphemes in a phrase while considering the context of a sentence [15]. There are also several POS taggers for Korea such as Mecab-ko [3], Twitter [4], Hannanum [1], and so on.

In this work, we utilize Mecab-ko library since it is a well-organized [2], commonly used library.

Capturing spatial correlation: Some of the recent work has shown that we can improve the performance of the network by embedding learning mechanism explicitly for capturing spatial correlations [9,17]. Hu et al. [8] introduced an architectural unit called Squeeze-and-Excitation (SE) block that explicitly models inter-dependency between channels of its convolution features. This SE block re-calibrate channel-wise feature responses with low computational overhead. As a result, they won the first place at ILSVRC 2017[1] while significantly reduced the Top-5 error.

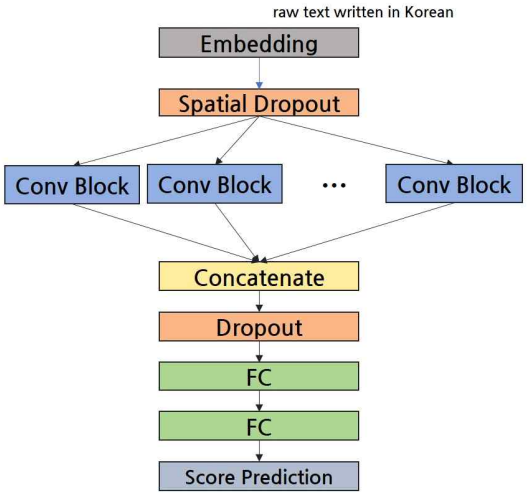

Fig. 2. Overview of our CNN architecture for movie rating prediction

In this paper, we employ SE blocks to our network in two ways and analyze the benefits.

Ⅲ. CNN Architecture for Movie Rating Prediction

In this section, we introduce our CNN architecture that predicts movie rating from an audience's comment written in Korean. We first describe the overall architecture (Sec. Ⅲ-1) and then explain our convolution layer design (Sec. Ⅲ-2). Finally, we introduce a novel scoring function for translating the result of the convolution layer to a rating (Sec. Ⅲ-3).

Ⅲ-1. Overview

Our network consists of three components including an embedding layer, a convolution layers, and fully-connected layers. (Fig. 2)

1) ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

The embedding layer converts texts (i.e. audience's comments) to vectors. We employ a character embedding scheme that handles the texts in character level. This is because our target datasets contain unnormalized text and Yu et al. [18] showed that a character embedding scheme works better than a word embedding scheme for unnormalized texts.

Once we get vectors, we take a spatial dropout stage to discard noise (e.g., unnecessary words or character) in the unnormalized text. The spatial dropout operation selects elements of a vector randomly with a specific ratio and makes them to zero. This makes sense because our target dataset includes lots of meaningless single characters like 'ㅋㅋㅋ' or 'ㅎㅎㅎ' that means just laughing in Korean.

The vectors generated from the pre-processing layer are sent to the convolution layer. We design our convolution layer (Fig. 2) based on TextCNN [10]. The convolution layer consists of a set of convolution blocks that have different kernel sizes and work in parallel. This is a similar form of the CNN model used in TextCNN. However, we improve the representation power of a convolution layer by adding SE blocks (Sec. Ⅲ-2). Outputs of each convolution blocks (i.e. captured morphological information) are concatenated as a vector.

Fully-connected layers compress the extracted features from the convolution layer. Finally, the compressed feature is translated as a rating with our scoring function (Sec. Ⅲ-3).

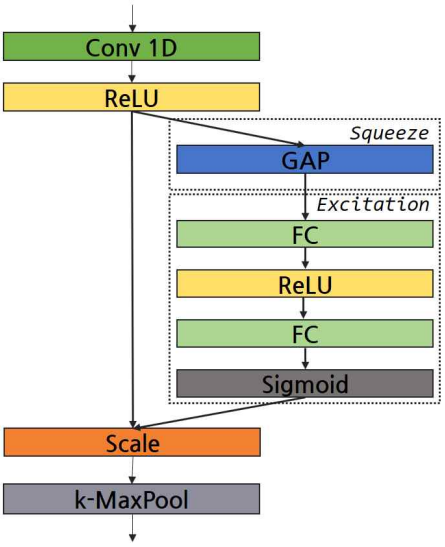

Fig. 3. Convolution block in our network. We improve the basic convolution block with an SE block (dotted box).

Ⅲ-2. Convolution Layer Design

We design our basic convolution layer as a form of shallow-and-wide CNN model. To make a feature map, we put multiple convolution blocks having a different kernel size in a parallel manner and concatenate outputs of them. Each convolution block performs 1D convolution operation and takes a rectifying stage. At the end of a convolution block, we use k-Max pooling to capture k ($k > 1$) features instead of max-over-time pooling used in TextCNN.

In addition, we improve the basic convolution layer by adding SE blocks to increases the representational power of the network. An SE block consists of two operations, Squeeze and Excitation. The *Squeeze* operation aggregates global spatial information using global average pooling (GAP). This operation reduces the dimensionality of a feature to one by

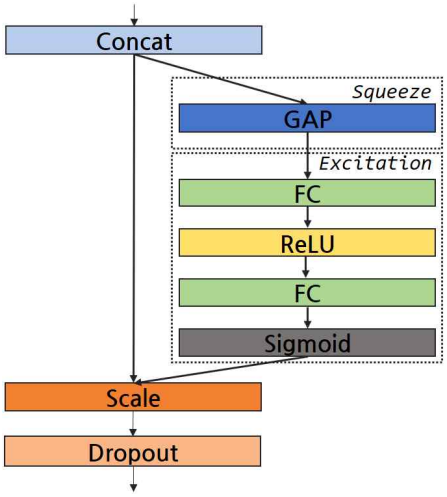

Fig. 4. Concatenate layer in our network. The dotted boxes show the SE block we add to the basic concatenate layer.

averaging outputs of each channel. The *Excitation* operation adaptively re-calibrates feature maps of each channel by applying a gating mechanism for the output of the squeeze operation. The gating mechanism consists of two fully-connected (FC) layers that perform non-linear operations among channels. As discussed in SENets [8], the excitation operation is one of the major bottlenecks for training performance. To lessen such overhead, we compress the dimension of a channel from C to αC (e.g., α=1/16) in the first FC layer. Then, it passes the second FC layer and takes sigmoid activation. Finally, the result is scaled by channel-wise multiplication between the feature maps in the scale layer.

We apply this SE block in two ways. At first, we put an SE block in-between convolution layer and k-Max pooling layer in the basic convolution block as shown in Fig. 3. With this approach, we aim to re-calibrate spatial correlation among features extracted with the same kernel size. The second approach is attaching an SE block after the concatenate layer (Fig. 4) and it catches the spatial information among features comes from convolution blocks having different kernel sizes.

Table 1. This table shows MSEs of our models with scoring functions based on sigmoid and tanh.

Model	*sigmoid*	*tanh*
Ours-ConvSE	3.8466	**3.4043**
Ours-ConcatSE	3.4786	**3.3841**

III-3. Scoring Function

To predict a movie rating from the classification result of CNN models, we formulate a novel scoring function,

$$Score(x) = \frac{(\sigma(x) \times 9 + 11)}{2}$$

, where x is the output value from the last FC layer and σ means *tanh* function. It scales the results of FC layer to the target rating ranged from 1 to 10. Instead of *sigmoid* function that usually employed for scaling value to a specific range, we use *tanh* function since it is more cost-effective than sigmoid function for training. Table 1 compares the MSEs with our scoring functions with *sigmoid* and *tanh* on our prediction architectures.

Ⅳ. Implementation

We have trained our network with the Adam optimizer [11] and the mini-batch size is 128. To check the advantages of character embedding, we also have implemented our model with

word embedding and we have used the Adadelta optimizer [19] in this case since the usage rate of the variable varies according to the probability of occurrence of the word. We brought character embedding method from a baseline code of NAVER A.I Hackathon 2018. For word embedding, we used Mecab-ko library [3]. To find the best embedding size, we did a greedy search and used 384 and 300 for character embedding and word embedding, respectively.

We used five convolution blocks in a parallel manner and the convolution filter size is 10, 9, 7, 5 and 3 for character embedding. For the word embedding, we used four convolution blocks and the convolution filter size are 2, 3, 4 and 5, respectively. The input dimension of each convolution filer is 256 and the dimension of outputs are varies according to the filter size. A rectified linear unit (ReLU) with a threshold of 1e-6 was used as an activation function, k-MaxPooling was used with k of 3 for each convolution block. For two fully connected layer at the end of the network, we used 1024 and 1 unit(s) respectively.

The initial learning rate is 2e-4, and it is decreased by a factor of 0.95 for every 25,000 global steps. The dropping ratio for both the spatial drop after embedding layer and the dropout layer after the concatenate layer are 0.7. We applied L2 regularization with a rate of 1e-3 for all the parameters on our network except the embedding layers. To prevent gradient overshooting, we clipped a norm of gradients by 5.

To compare the accuracy of our method, we have implemented three version of our methods and two alternative models:

- **Ours-baseline** is our baseline model without any SE block (Fig. 2).

- **Ours-convSE** has implemented by adding SE block to the convolution blocks (Fig. 3) to *Ours-baseline*.

- **Ours-concatSE** has implemented by attaching SE block to the concatenate layer (Fig. 4) to *Ours-baseline*. Please note that we use the basic convolution block, not convSE, for this model.

- **TextCNN-word** is our implementation of the CNN model proposed by Kim [10]. In this model, we used word embedding and three convolution blocks whose kernel sizes are 3, 4, and 5, respectively. Different from our model, it does not include spatial dropout layer while predicting the rating with our scoring function. For the rest of hyper-parameters, we followed the paper.

- **TextCNN-char** has implemented by changing the embedding scheme to character embedding from the *TextCNN-word*.

We have implemented the models with TensorFlow 1.10, Numpy and KoNLPy libraries.

Table 2. This table shows the MSEs of five different models including previous work and our methods for the NAVER movie review dataset.

Model	MSE
TextCNN-word	11.3971
TextCNN-char	4.5875
Ours-baseline	4.1217
Ours-convSE	3.4043
Ours-ConcatSE	**3.3841**

Ⅴ. Experiments

To check the accuracy of our movie rating prediction architecture, we have applied our method for the reviews in NAVER Movie[2] (Fig. 1) that is one of the largest movie review sites in Korea. We made the NAVER movie review (NMR) dataset by crawling reviews from the website. Each review data consists of a rating score ranged from 1 to 10 and comments of an audience for a movie. The comments are written in Korean and much of them includes unnormalized and incomplete sentences. The NMR dataset includes ten classes from 1 to 10 and contains 8.86 million reviews (Table 3). The average number of characters per review is 64 and the minimum and the maximum length is 1 and 299 character(s), respectively. We randomly split the dataset and have used 80% of them for training and other for validation.

Table 2 compares the MSEs of five different models. As shown in the results, character embedding greatly improves the prediction accuracy than word embedding for our dataset. This demonstrates that the character embedding scheme works better for unnormalized sentences than the word embedding scheme.

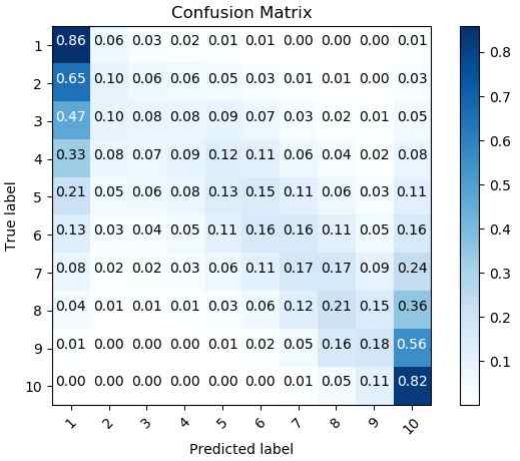

Fig. 5. This confusion matrix shows the prediction (or classification) performance of *Ours-ConcatSE*. To draw this matrix, we rounded the prediction value.

Our baseline implementation (i.e. *Our baseline*) shows better performance than *TextCNN-char* and the accuracy is further improved with SE blocks. We found that both our approaches using SE blocks reduce the errors. This is because the SE blocks catch spatial correlation among extracted features well. In our experiment, *Ours-concatSE* achieves the lowest MSE, 3.3841, and it can be interpreted into the prediction results have errors higher or less than 1.83 in most case. These results demonstrate the accuracy of our method.

Figure 5 is a confusion matrix that shows the prediction performance of *Ours-ConcatSE*. Each row shows the distribution of prediction results for a class (i.e. rating score). As shown in the confusion matrix, most of the prediction ratings are distributed around the true

2) https://movie.naver.com/

Table 3. Distribution of reviews in the NAVER movie dataset

Rating (class)	1	2	3	4	5	6	7	8	9	10	Total
# of reviews (millions)	0.88	0.15	0.14	0.16	0.27	0.38	0.54	0.94	1.01	4.38	8.86
Ratio (%)	9.95	1.73	1.53	1.85	3.09	4.29	6.14	10.64	11.39	49.36	100.0

label. These results validate the usefulness of our CNN architecture for movie rating prediction.

Ⅵ. Conclusion

We have presented a CNN architecture that predicts a rating for a movie from an audience's comment, specially written in Korean and contains lots of slang and incomplete sentences. To handle such unnormalized texts efficiently, we employ a character embedding scheme and SE blocks that catches spatial correlation among features. We also proposed a novel scoring function that converts the classification results of our CNN model to rating scores. We have applied our architecture to a movie review dataset and have demonstrated the accuracy and usefulness of our method.

Limitations and future work: Although our method has achieved a low MSE, it shows relatively lower accuracy for medium range (e.g., 4-7). To achieve more accurate prediction results over all ranges, we would like to improve our CNN model and scoring function. In this paper, we have tested our architecture for only one dataset. As a future work, we would like to apply our method to other datasets like other movie review datasets or reviews for other publications like books.

Acknowledgements

We would like to thank anonymous reviewers for their constructive feedbacks. This work was supported by the research fund for a professor of Korea University of Technology and Education and Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (2018R1C1B5045551).

Reference

1. Hannanum morphological analyzer, http://semanticweb.kaist.ac.kr/research/hannanum/index.html, 2016.
2. KoNLPy benchmark performance among the libraries, http://konlpy.org/en/latest/morph/#comparison-between-pos-tagging-classes, 2018.
3. Mecab-ko morphological analyzer, http://eunjeon.blogspot.com/, 2018.
4. Twitter morphological analyzer, https://openkoreantext.org, 2017.
5. O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu., "Convolutional neural networks for speech recognition", IEEE/ACM Transactions on audio, speech, and language processing, 22, 10, pp. 1533 - 1545, 2014.
6. R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning", In Proceedings of the 25th international conference on

Machine learning, 160 - 167, 2008.
7. A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very Deep Convolutional Networks for Text Classification", ArXiv eprints:1606.01781, 2016
8. J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu., "Squeeze-and-Excitation Networks", ArXiv e-prints:1709.01507, 2017
9. S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", arXiv preprint:1502.03167, 2015.
10. Y. Kim, "Convolutional Neural Networks for Sentence Classification", ArXiv e-prints:1408.5882, 2014.
11. D. P. Kingma and J. Ba., "Adam: A Method for Stochastic Optimization", ArXiv e-prints, 1412.6980, 2014
12. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", Advances in neural information processing systems, pp. 1097 - 1105, 2012.
13. H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?", arXiv preprint 1707.04108, 2017.
14. C. M. Chang, J. Cho, H. Liu, R. K. Wagner, H. Shu, A. Zhou, C. S. Cheuk, and A. Muse, "Changing models across cultures: Associations of phonological awareness and morphological structure awareness with vocabulary and word recognition in second graders from beijing, hong kong, korea, and the united states", Journal of experimental child psychology, 92, 2, pp. 140 - 160, 2005.
15. S. Petrov, D. Das, and R. McDonald. "A universal part-of-speech tagset", arXiv preprint arXiv:1104.2086, 2011.
16. K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition", ArXiv e-prints:1409.1556, 2014.
17. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1 - 9, 2015.
18. X. Yu, A. Faleńska, and N. T. Vu. A, "general-purpose tagger with convolutional neural networks", arXiv preprint:1706.01723, 2017.
19. M. D. Zeiler, "Adadelta: an adaptive learning rate method", arXiv preprint:1212.5701, 2012.