Tensor Space Model-based Textual Data Augmentation for Text Classification

1st Minsuk Chang
Department of Computer Science and Engineering
Seoul National University
Seoul, Korea
jangsus1@snu.ac.kr

2nd Han-joon Kim
Department of Electrical and Computer Engineering
University of Seoul
Seoul, Korea
khj@uos.ac.kr

***Abstract*—In this paper, we first introduce a new text representation method to convert a textual document into a tensor space model named TextCuboid, which can preserve various meanings of polysemy. Based upon the new model, we propose two novel data augmentation techniques (called *Boolean augmentation* and *CuboidGAN*) that can be directly applied to the TextCuboid model for text classification tasks. *Boolean augmentation* includes three simple keyword modifications: synonym replacement, synonym insertion, and random deletion. *CuboidGAN* is composed of two key components, style encoding and residual regression, and it is trained in two phases to generate unambiguous and plausible concept vectors. Through intensive experiments using five commonly used datasets, we prove that our proposed methods perform better data augmentation than other conventional methods. We also show that each component of the augmentation method makes a significant contribution to text classification through ablation studies.**

***Index Terms*—Text Representation Model, Tensor Space Model, Data Augmentation, Text Classification, Deep Learning, Generative Adversarial Networks, Autoencoder**

I. Introduction

Extracting information and patterns from large amounts of textual data is one of the crucial tasks in big data research. However, text data is notorious for its difficulty to handle due to the extensive variation in length; this becomes a big problem when training deep neural networks because it requires fixed-sized inputs. Alongside many approaches to the problem, Kim et al. [1] suggested a three-dimensional tensor structure called 'TextCuboid' which tried to represent exact meanings within any textual document with concept vectors extracted from Wikipedia pages. However, this approach cannot completely solve the polysemy problem which frequently happens in text domains. Therefore, we intend to extend the prior TextCuboid so as to capture the context of a term[1] inside the text; in our work, to solve the polysemy problem and increase the expressiveness of TextCuboid, we replace Wikipedia-based concept vectors with context-aware embedding vectors. In this respect, we refer to the extended representation model as the contextual TextCuboid. The details of our work of building a contextual TextCuboid are described in Section IV-A.

Furthermore, we should consider another issue to enhance text classification tasks. Even though the contextual TextCuboid model can effectively represent the diverse meanings of terms, a large amount of data is required to reach higher performance on text classification tasks. Many elaborate methods are already available to enrich text data, but they are inefficient under the TextCuboid model due to the process by which raw texts are converted to TextCuboids. Because the TextCuboid model contains only the core information of keywords in the given text data, subtle changes in the meaning of the original text, especially changes in the meaning of unimportant terms, can be easily ignored in the TextCuboid model. Therefore, we have proposed two novel methods to augment the TextCuboid data generated from the original text: *Boolean augmentation* and *CuboidGAN*. The former method augments data by randomly changing the keywords in the contextual TextCuboids, and the latter generates a new concept vector by training the GAN (Generative Adversarial Networks)-based model fitted to the contextual TextCuboid. In Sections IV and V, we will explain and prove that our two methods can surpass other recent text augmentation methods for text classification tasks.

The contribution of our work can be listed as follows.

- We enhance the original TextCuboid representation model using context-aware embedding; this extended model captures contextual information from given text data, constructs the results into concept vectors within TextCuboids, and solves the polysemy problem.
- We propose two novel data augmentation methods, *Boolean augmentation*, and *CuboidGAN*, that can be applied directly to the TextCuboid data. We empirically prove that our method can improve the performance of classification models over conventional methods.

[1]Throughout the paper, 'word' is used interchangeably with 'term'.

II. Related Work

A. TextCuboid representation model

Kim et al. [1] empirically proved that the TextCuboid structure can effectively encapsulate the semantic information of any text document. The TextCuboid model represents a single document as a two-dimensional matrix (i.e., a set of documents as a three-dimensional tensor), where each row corresponds to a keyword extracted from a set of documents, and each column is a concept vector for that keyword. They have suggested a way to generate concept vectors using the Wikipedia page corresponding to each concept. Here, if a keyword is not included in the given document, its corresponding concept vector becomes a zero vector, indicating that such a word does not exist.

B. Text Data Augmentation

Recently, various text data augmentation methods have been developed to improve the accuracy of classification models with small labeled data. Jason Wei and Kai Zou [2] augmented words within the text by random deletion, replacement with synonyms, or random insertion. Edunov et al. [3] utilized a pair of trained language translators to translate English text into other languages and back into English, generating new text with similar meaning.

C. Generative Adversarial Networks

GAN (Generative Adversarial Networks) was first introduced by Goodfellow et al. [4], proving that it can generate artificial data that follow any distribution. Furthermore, many of its variant models were developed to solve several problems, such as mode collapse or unstable training. Arjovsky et al. [5] solved the vanishing gradient problem of GAN and stabilized the training process by changing the loss and clipping the discriminator's gradients. StyleGAN, proposed by Karras et al. [6], successfully generated high-quality samples by introducing a style-based GAN structure for coarse and fine-grain representations.

III. Problem Definition

Text data augmentation methods are widely used for their simplicity and effectiveness when developing text classification models [2, 3]. However, the effect of text data augmentation may not directly appear on the TextCuboid structure due to its limited number of keywords. Because a Textcuboid is formed by extracting only key terms from a given document, the effect of synonym substitution or keyword paraphrasing can easily be ignored. Therefore, when augmenting the TextCuboids, we need to focus on generating TextCuboids that reflect both the structure and semantics inherent in the original documents.

Additionally, augmentation methods that accommodate the structure of a TextCuboid should take into account its sparsity. Generating a new TextCuboid with only a simple deep generative network structure is not easy because the concept vectors inside it may be zero vectors or valid embedding vectors. Also, the generated TextCuboid should exhibit semantics similar to the original TextCuboid to perform proper data augmentation. In our work, we have designed a GAN-based model called 'CuboidGAN' whose training process can be stabilized so as to match the contextual TextCuboid. In this paper, our goal is to demonstrate that new samples generated by our proposed augmentation technique improve the performance of text classification tasks.

IV. Proposed Method

A. Contextual TextCuboid

Our contextual TextCuboid includes concept vectors that contain contextual information about terms in the document. As mentioned before, the original TextCuboid's concept vectors were extracted from the Wikipedia pages, but this representation approach does not fully solve the polysemy problem. Hence, to extract contextual information within the document while preserving the meaning of the terms, we have used the BERT (Bidirectional Encoder Representations from Transformers) embedding [7] to create the context-aware concept vectors; that is, embedding vectors containing both semantic and contextual information about chosen terms in a document are gathered to form the contextual TextCuboid by using the BERT embedding model.

B. TextCuboid Data Augmentation

In the following sections, we describe a new data augmentation technique suitable for the contextual TextCuboid data: *Boolean augmentation* and *CuboidGAN*. The *Boolean augmentation* creates a new combination of terms representing the original Textcuboid while maintaining high semantic similarity. The CuboidGAN model generates plausible and semantically similar concept vectors to the original TextCuboid. The CuboidGAN is pre-trained with adversarial autoencoder [8] and fine-tuned with general adversarial loss. Finally, the augmented binary vector is combined with the newly created concept vector to become a new TextCuboid data.

C. Boolean Augmentation

For the boolean augmentation, the existence of each term in a TextCuboid is represented as a boolean vector. For example, if the original text is ”The monkey eats a banana.” and the keyword set is $\{$monkey, apple, banana$\}$, then the Boolean vector becomes $< 1, 0, 1 >$.

Inspired by J. Wei and K. Zou [2], Boolean vectors are modified based on the similarity between keywords. First, in order to calculate the similarity, the centroid of the concept vector for each keyword can be obtained with the average vector for all documents. The Euclidean distance between concept vectors' centroids can then be used to calculate the proximity of keyword pairs. Finally, for each term, we select the closest K keywords with a distance less than D to ensure that the extracted terms are actual synonyms. The detailed process of selecting similar terms is shown in Algorithm 1.

Similar terms are then used for data augmentation with random insertion, deletion, or replacement. The overall Boolean augmentation process is described in Algorithm 2

D. Generating New Concept Vectors with CuboidGAN

1) Model Architecture: CuboidGAN is first pre-trained by the AAE (Adversarial Autoencoder) [8] structure and forces the latent vector of the encoder module to follow the Gaussian distribution. After completing pre-training, the Decoder and Encoder become a generator and discriminator, respectively,

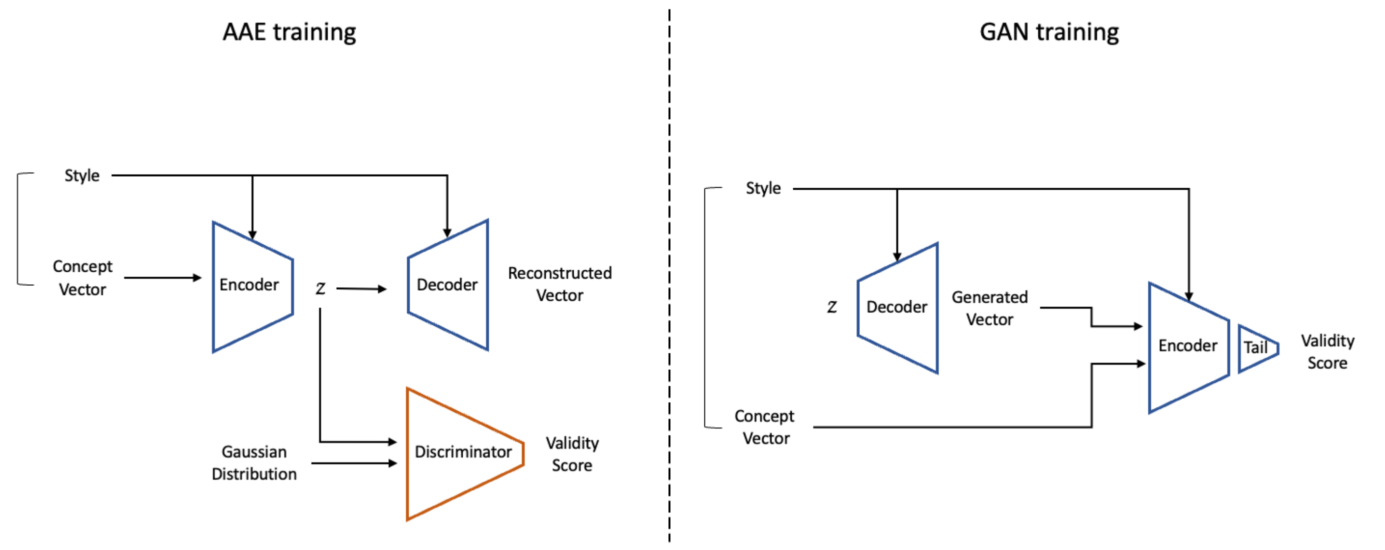


Fig. 1: The architecture of CuboidGAN. Left: Pre-training step of the adversarial autoencoder structure to match the encoder's latent space to a Gaussian distribution. Right: Main GAN training step where the adversarial loss guides the generated concept vector. All modules except the Encoder tail are pre-trained in the AAE step. The Encoder tail is appended to the encoder to produce a final validity score.

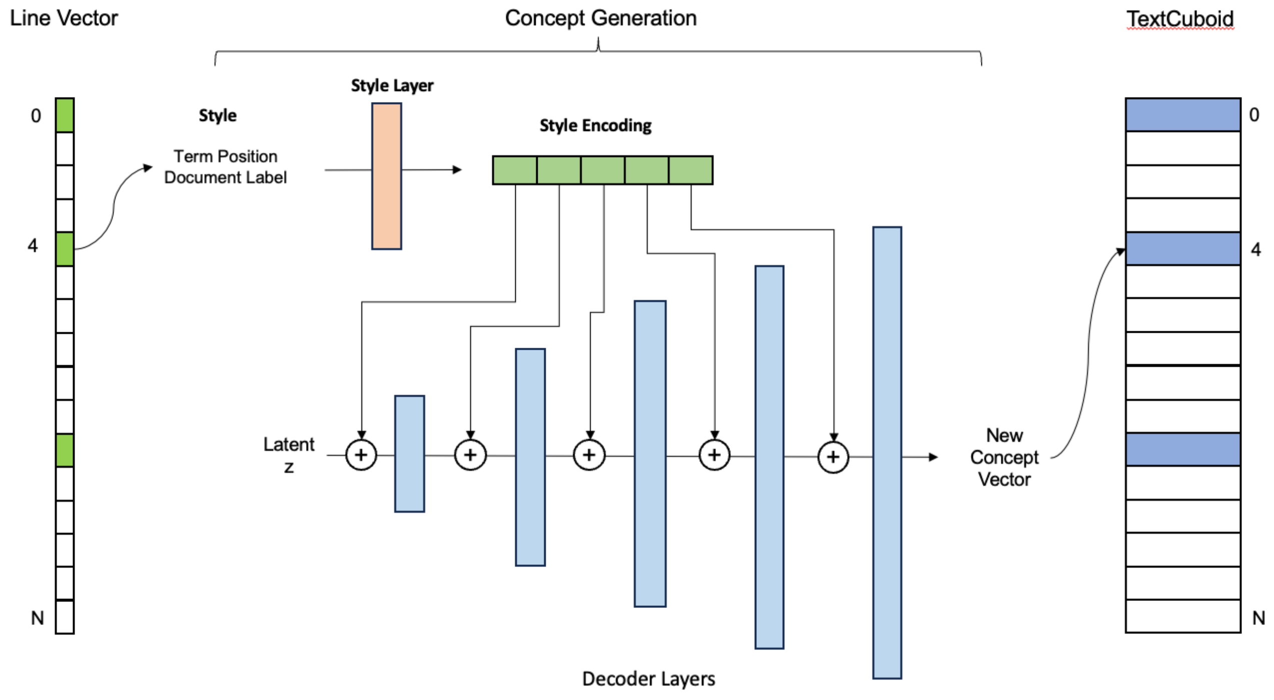


Fig. 2: Concept vector generation process using the trained CuboidGAN's decoder. The style layer generates style encoding for each of available terms in the Boolean vector. The extracted style encoding is separated and concatenated to the main decoder layers during the construction process. Lastly, the generated vector becomes the corresponding TextCuboid's augmented concept vector.

Algorithm 1 Similar Term Selection. Hyperparameters: $K = 10$, $D = 20$.

```
Require: concepts: averaged concept vectors, keywords: terms which consist of TextCuboid's rows, K: # of nearest keywords, D: maximum distance
Ensure: similarTerms: top-k similar terms
 1: similarTerms ← empty dictionary
 2: for term in keywords do
 3:     base ← concepts[term]
 4:     distances ← GetEuclidianDistances(base, concepts)
 5:     distances ← FilterLower(distances, D)
 6:     topK ← SelectTopK(distances)
 7:     similarTerms[term] ← topK.terms
 8: end for
```

Algorithm 2 Boolean Augmentation Pseudocode. Default hyperparameters used for this paper: $\alpha = 0.1$, $\gamma = 0.1$

```
Require: similarTerms, baseBooleanVector, $\alpha$: augment ratio, $\gamma$: sample ratio
Ensure: newBooleanVector
 1: newBooleanVector ← baseBooleanVector
 2: if Random() < $\alpha$ then
 3:     type ← RandomChoice(insert, delete, replace)
 4:     targetTerms ← RamdomSample(baseBooleanVector, $\gamma$)
 5:     for term in targetTerms do
 6:         if type = insert then
 7:             newTerm ← RandomSelect(similarTerms[term])
 8:             newBooleanVector.addTerm(newTerm)
 9:         else if type = delete then
10:             newBooleanVector.deleteTerm(term)
11:         else if type = replace then
12:             newTerm ← RandomSelect(similarTerms[term])
13:             newBooleanVector.addTerm(newTerm)
14:             newBooleanVector.deleteTerm(term)
15:         end if
16:     end for
17: end if
```

and undergo additional training as a standard GAN model. The architecture of CuboidGAN is seen in Fig. 1.

Now, we introduce Style Encoding and Residual Regression to the original encoder and decoder modules so that CuboidGAN can produce both plausible and various concept vectors.

a) Style Encoding: Style-based GAN, proposed by Karras et al. [6], enables the GAN model to capture both coarse concepts and fine-grained details while increasing the quality of the generated images. Hence, our CuboidGAN model adopts the style encoding to ensure that every layer contains two sorts of significant information: the terms for the concept vectors the model generates, and the classification label of each document. The position of each term in a TextCuboid is encoded as Equation (1) (PE), which is similar to the positional encoding from [7]. Also, the label is one-hot encoded as Equation (2), which is concatenated with the positional encoding, finally becoming the style encoding with Equation (3). Later, this concatenated vector is spread to all decoder layers after passing the style layer. This allows the decoder to generate an embedding corresponding to a specific term and label. The variable L in the equation is the predefined encoding length.

$$PE_{(row,col)} = \begin{cases} sin(\frac{row}{10000^{col/L}}) & \text{if } col \text{ is even} \\ cos(\frac{row}{10000^{(col-1)/L}}) & \text{if } col \text{ is odd} \end{cases} \tag{1}$$

$$LE_{(col,label)} = \begin{cases} 1 & \text{if } col = label \\ 0 & \text{if } col \neq label \end{cases} \tag{2}$$

$$Style_{(row,col,label)} = \begin{cases} PE_{(row,col)} & \text{if } col < L \\ LE_{(col-L,label)} & \text{if } col \geq L \end{cases} \tag{3}$$

$$\tag{4}$$

where PE and LE denote the positional and label encoding, respectively, and row and col means the number of rows and columns in the TextCuboid. Style denotes the style encoding vector, which is a concatenation of PE and LE; the style encoding vector is generated after passing through CuboidGAN's Style Layer.

b) Residual Regression: The residual regression is applied to the encoder and decoder module to allow the combined model to learn only the difference from the concept vectors' centers. In this paper, each of BERT embedding vectors is used as a concept vector, and thus the concept vectors' center represents the general semantic of each term. Therefore, the encoder and decoder module only need to learn the deviation of each concept vector from the center, which facilitates the training of the model.

2) Pre-Training: CuboidGAN model is pre-trained by using an adversarial autoencoder structure [8]. Also, our model tries to stabilize the training process using the improved loss and gradient clipping from [5]. The network's pre-training process is separated into two alternating phases: Reconstruction and adversarial training.

a) Reconstruction Phase: The encoder and decoder modules are jointly trained with a reconstruction loss. If the original concept vector x passes the encoder, it becomes the latent vector z, and the decoder reconstructs it. During this process, the style layer is jointly trained to provide each term's position and label information at coarse and fine-grain levels. The reconstruction loss of Equation (5) is defined as a mean square error between the original and reconstructed concept vectors.

$$\begin{aligned} z_i &= En(x_i, style_i) \\ \tilde{x_i} &= De(\{z_i, style_i\}) \\ \min_{En,De} Reconst(En, De) &= E_{x\sim data}(x - \tilde{x})^2 \end{aligned} \tag{5}$$

where x_i denotes the i^{th} concept vector and z_i denotes the latent vector returned by the Encoder En after receiving x_i and $style_i$. $style_i$ denotes the style vector corresponding to the i^{th} concept vector. $\tilde{x_i}$ denotes the reconstruction vector retured by the Decoder De. $Reconst$ means the reconstruction loss, and $x \sim data$ denotes a sample x chosen from a given concept vector.

b) Adversarial Phase: The latent vector of the encoder should be tuned to follow a pre-defined distribution, which is the Gaussian random variable in this paper. The discriminator is trained to distinguish the generated latent vector from the Gaussian random variable, and the encoder tries to generate its latent vector similar to the Gaussian distribution. Also, to stabilize the generator's training, Wasserstein loss and gradient clipping [5] are used. The adversarial loss in the pre-training step is defined as the following Equation (6):

$$\begin{aligned} &\min_{En}\max_{De} PreAdv(En, De) = \\ &E_{z\sim N(\mu,\sigma)}[De(z)] + E_{x\sim \text{data}}[-De(En(x))] \end{aligned} \tag{6}$$

where $PreAdv$ means the adversarial loss in the pretraining phase. z denotes a random latent vector with the Gaussian distribution, and x denotes a sample chosen from a given concept vector.

3) Main Training: During the pre-training, the reconstruction loss (defined as Equation (5)) can cause the model to generate ambiguous averaged samples. Therefore, the model is further trained with the original GAN structure by changing the encoder and decoder's roles. As described in the right part of Figure 1, the decoder generates a concept vector from a random vector and style encoding while the encoder tries to distinguish it from the original vectors. The adversarial loss in main training, which also adopts the Wasserstein loss, is defined as Equation (7)

$$\begin{aligned} &\min_{De}\max_{En} MainAdv(En, De) = \\ &E_{x\sim data}[En(x)] + E_{z\sim N(\mu,\sigma)}[-En(De(z))] \end{aligned} \tag{7}$$

where $MainAdv$ means the adversarial loss in the main training phase.

4) Model Inference: The trained decoder module of CuboidGAN can generate a new concept vector corresponding to the given style. As shown in Figure 2, Boolean vectors can be transformed into TextCuboid data by passing each term's position and the document label as the decoder's input.

E. Text Pre-Augmentation

GAN requires large amounts of training data to avoid problems such as mode collapse or low-quality outputs. Therefore, we apply various text augmentation methods to the original text to increase textual training data, which can be used for training CuboidGAN. Augmented texts are converted to a new TextCuboid with the same label, allowing the model to be trained from those synthetic TextCuboids. Conducting augmentation operations on the original text itself in advance to provide additional data to the CuboidGAN model can help improve the performance of classification tasks.

V. Experiments

The purpose of this experiment is to show that new TextCuboid data generated through our proposed data augmentation method helps to further enhance classification performance. Therefore, to effectively examine the impact of data augmentation, it is sufficient to use only a small number of samples per class label, and the augmentation methods were also tested on a simple classifier only with a number of linear neural network layers.

A. TextCuboid Generation

Basically, transforming a textual document into a TextCuboid format requires the extraction of keywords (i.e., terms) and corresponding contextual concept vectors.

1) Keyword Extraction: The number of keywords required to represent a single document varies among the text datasets. In our experiments, we have chosen the top 5,000 keywords for all datasets to prevent the model size from becoming too large; among the candidate tokens generated by the BERT tokenizer, the most meaningful keywords have been selected according to the χ^2-statistic. Before χ^2-statistic based feature selection, we have removed unimportant words (including stopwords) with higher document frequency than 10% of the total number of documents.

2) Contextual Concept Vectors: For the contextual word embedding, we have used a pre-trained tiny-BERT model [9, 10] to produce 128-length embedding vectors. The embedding vectors of the last four layers of BERT are summed, each of which is used as a concept vector. The resulting TextCuboid has 5,000 rows and 128 columns.

B. CuboidGAN Model Configurations

Our proposed CuboidGAN model consists of encoder and decoder modules, each with five linear layers stacked. The latent vector has the same dimension size as the concept vector (e.g., 128) but the adversarial training prevents the model from overfitting. The positional encoding is also applied with the same length of 128, and style encoding has a length of 32 per layer.

When composing each data set used for text classification, only the training split is fed into the model to ensure fair performance evaluation. The pre-training step was performed over four epochs for all datasets. In comparison, the epochs of the main training step were set to vary due to the number of training samples, which is shown in Table III. In addition, for a more efficient training process, we adopted the RMSProp optimizer with a learning rate of 0.0005 and gradient clipping of 0.01.

C. Datasets

In order to evaluate our proposed method, We have used five widely used text classification datasets while measuring classification accuracy and F1-score. Additionally, undersampling was applied to these datasets to solve the class imbalance problem; that is, when there are relatively excessive text documents with a particular category, they have been removed to

TABLE I: Data Augmentation Results: Classification Accuracy

Dataset	Baseline	EDA [2]	Back Translation [3]	Proposed Method	
				Boolelan Augmentation O	Boolelan Augmentation X
20newsgroups	0.6004	0.6158	0.6208	**0.6252**	0.6230
Yahoo! Answers	0.4020	0.3993	0.4012	**0.4127**	0.4088
IMDB	0.7339	0.7356	0.7343	0.7373	**0.7426**
SST2	0.5912	0.5954	0.5879	**0.597**	0.5956
Reuters	0.822	0.8392	0.8343	**0.8433**	0.8392

TABLE II: Data Augmentation Results: F1-Score (Macro)

Dataset	Baseline	EDA [2]	Back Translation [3]	Proposed Method	
				Boolelan Augmentation O	Boolelan Augmentation X
20newsgroups	0.6002	0.6139	0.6203	**0.6244**	0.6213
Yahoo! Answers	0.3993	0.3939	0.3976	**0.4096**	0.4053
IMDB	0.7338	0.7355	0.7342	0.7373	**0.7426**
SST2	0.5899	0.5951	0.5874	**0.5968**	0.5955
Reuters	0.8187	0.8349	0.8298	**0.8388**	0.8347

TABLE III: Dataset Details

Dataset	Number of Categories	Number of Samples	Number of Keywords	Training Epochs
20newsgroup	20	1,000	5,000	4
Yahoo! Answers	10	500	5,000	4
IMDb	2	400	5,000	2
SST2	2	200	844	1
Reuters	10	480	5,000	1

ensure that the number of documents per label is as uniform as possible. Table III lists the number of samples and categories in detail.

- 20Newsgroups: The 20Newsgroups dataset is widely used for text classification on news articles, initially gathered by Ken Lang [11]. It has approximately 20,000 documents, which are partitioned across 20 different newsgroups.
- Yahoo! Answers: The Yahoo! Answers dataset was gathered by Zhang et al. [12]. Because the size of this dataset is relatively large compared to other datasets, we sampled only around 1% of documents for training. In addition, the title, questions, and answers within each document are all concatenated into a single paragraph and used in the experiment.
- IMDb: The IMDb Movie Reviews dataset is used for semantic classification in many text mining research. It was first presented by Maas et al. [13], which was particularly used for learning word vectors that include sentiment contents.
- SST2: The Stanford Sentiment Treebank v2, proposed by R. Socher et al. [14], is a semantically labeled dataset for binary classification. Compared to other datasets, the SST2 dataset consists of short sentences with fewer words.
- Reuters: The Reuters-21578 dataset is a collection of documents with news articles, which consists of many multi-label documents. We have thus selected the top-10 categories that have the largest number of samples and also converted some of the multi-labeled data into single-labeled ones.

D. Text Classification

In order to classify augmented text data, we have used a simple classifier network composed of five linear layers, which ultimately allows us to evaluate the impact of data augmentation. The input TextCuboid, sized 5000*128, is passed through 5 layers until the final softmax layer. Accuracy and F1-score are the primary metrics used to evaluate each data augmentation method on the text classification tasks. Also, to stabilize the performance among trials, we have performed soft label smoothing [15] of 20% for all data augmentation methods as a regularization technique.

E. Other Data Augmentation Methods

We have compared our proposed method with existing text data augmentation methods such as Easy Data Augmentation and Back Translation. For a fair comparative evaluation, the same amount of data was augmented with each augmentation method.

1) Easy Data Augmentation: EDA proposed by J. Wei and K. Zou [2] is a simple but powerful method for augmenting text data in various tasks. We have used the official implementation and the default parameter, $\alpha = 0.1$, suggested in the paper, where α is a parameter that indicates the proportion of words in the sentence changed by augmentation.

2) Back Translation: Back translation [3] is an augmentation technique that uses a pair of language translation models to generate sentences with similar meanings. In our work, among various models and implementations for Back Translation, we have used the Mariam-MT [16] model provided by Huggingface, learned by the University of Helsinki [17].

F. Results

1) Performance Analysis: Our experimental results are summarized in Tables I and II; each metric is calculated as

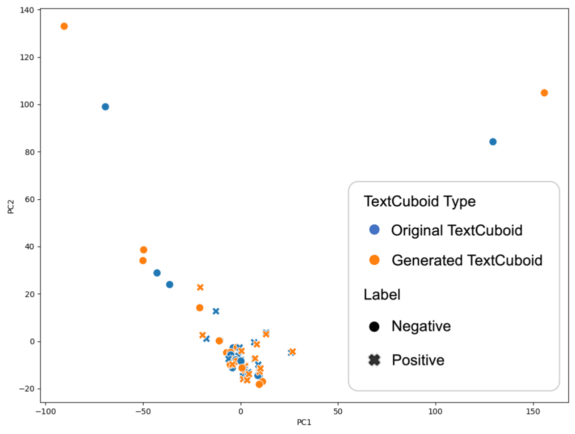
TABLE IV: Ablation Study: Classification Accuracy

Methods	20Newsgroups	Reuters
AAE pre-training	0.5878	0.8034
+ Style Encoding	0.6067	0.8302
+ Residual Regression	0.6143	0.8317
+ Text Pre-Augmentation	0.6210	0.8340
+ GAN post-training	0.6230	0.8392
+ Boolean Augmentation	0.6252	0.8433

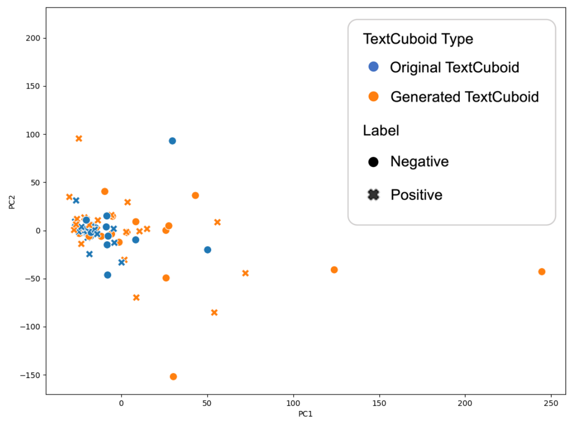
the average value of 10 trials in terms of accuracy and F1 score. As shown in the tables, the combination of our Boolean augmentation and CuboidGAN model surpasses other existing text augmentation methods for four datasets; when using the IMDB dataset, the CuboidGAN model alone performed about 0.5% better than the combination of both techniques.

2) Ablation Study: Additionally, we performed an ablation study to identify detailed components that contribute to data augmentation. Table IV shows that all components of the proposed model with data augmentation contribute to improving classification performance for the 20Newsgroups and Reuters datasets. The baseline was to train only the Adversarial Autoencoder without any of the listed components. Notice that the classification accuracy gradually increases as each component is added; through experiments, we have confirmed that the performance results do not change significantly depending on the order in which each component is added.

3) PCA Visualization: Figure 3 illustrates a PCA-based visualization result of the original TextCuboid and newly generated TextCuboid using our methods from the SST2 and IMDb datasets. Through the 50 samples shown in the figure, we can see that our method generates TextCuboid data that are helpful for classification; most of the newly generated TextCuboid data follows the distribution of the original data set, and they form a cluster shape. However, several samples generated from the IMDb dataset are located far from the cluster center. Thus, before performing data augmentation, filtering out TextCuboid data that is far from the cluster center among the output of the CuboidGAN model is required.



(a) A PCA-based visualization of SST2 dataset



(b) A PCA-based visualization of IMDb dataset

Fig. 3: Original & newly generated TextCuboids from SST2(a) and IMDb(b) datasets, visualized with PCA. Colors denote original or generated TextCuboid, and shapes indicate classification labels.

VI. Conclusion

In this paper, we proposed the Boolean Augmentation technique and CuboidGAN model suitable for context-aware TextCuboid to generate new artificial data that can improve the performance of text classification tasks through text augmentation. Through experiments on five popular textual datasets, we confirmed that the proposed methods work more effectively than existing augmentation ones. However, the Boolean Augmentation did not work effectively on the IMDB dataset. Therefore, deciding whether to utilize the Boolean Augmentation needs to be further studied through experiments with combinations of various hyperparameters.

Basically, the performance of automatic classification for text expressed in the TextCuboid model depends on the quantity and quality of tensor space; in other words, it is important to generate concept vectors that correctly contain the meaning and context of keywords. Our method uses the numerical distribution of concept vectors for augmenting datasets. There are chances that our method could be used on datasets that include highly professional terms, such as medical or legal domains, if their concept vectors are already obtained. Other text data augmentation methods might not work in these domains because they rely on prior knowledge of natural language. Specially trained language translators or wordnet embeddings should be prepared for these methods [2, 3], which is difficult in the real world. Therefore, finding out whether our method outperforms other text augmentation methods on such domains would be one of the future works to be done. Also, the TextCuboid model can be interpreted as a two-dimensional tensor, which is similar to an image. Hence, replacing some of CuboidGAN's linear layers with convolutional layers can allow the model to generate better TextCuboids in terms of semantics and context within documents.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2022R1A2C1011937), and was also supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2023-2018-08-01417) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

[1] H.-j. Kim, J. Kim, J. Kim, and P. Lim, "Towards perfect text classification with wikipedia-based semantic naïve bayes learning," *Neurocomputing*, vol. 315, pp. 128–134, 2018.

[2] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 6383–6389. [Online]. Available: https://www.aclweb.org/anthology/D19-1670

[3] S. Edunov, M. Ott, M. Auli, and D. Grangier, "Understanding back-translation at scale," *CoRR*, vol. abs/1808.09381, 2018. [Online]. Available: http://arxiv.org/abs/1808.09381

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[5] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[6] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," 2019.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[8] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2016.

[9] P. Bhargava, A. Drozd, and A. Rogers, "Generalization in nli: Ways (not) to go beyond simple heuristics," 2021.

[10] I. Turc, M. Chang, K. Lee, and K. Toutanova, "Well-read students learn better: The impact of student initialization on knowledge distillation," *CoRR*, vol. abs/1908.08962, 2019. [Online]. Available: http://arxiv.org/abs/1908.08962

[11] K. Lang, "Newsweeder: Learning to filter netnews," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 331–339.

[12] X. Zhang, J. J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *CoRR*, vol. abs/1509.01626, 2015. [Online]. Available: http://arxiv.org/abs/1509.01626

[13] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, Jun. 2011, pp. 142–150. [Online]. Available: https://aclanthology.org/P11-1015

[14] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. [Online]. Available: https://aclanthology.org/D13-1170

[15] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?" *CoRR*, vol. abs/1906.02629, 2019. [Online]. Available: http://arxiv.org/abs/1906.02629

[16] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. F. Aji, N. Bogoychev, A. F. T. Martins, and A. Birch, "Marian: Fast neural machine translation in C++," in *Proceedings of ACL 2018, System Demonstrations*. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 116–121. [Online]. Available: https://aclanthology.org/P18-4020

[17] J. Tiedemann and S. Thottingal, "OPUS-MT — Building open translation services for the World," in *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.