Augmentation Is All You Need

Hyeokjun Shin* Boost Camp AI Tech jun048098@korea.ac.kr

Juwon Kim* Boost Camp AI Tech uomnf97@gmail.com

Minjae Kang* Boost Camp AI Tech kminjae618@gmail.com

Sangwon Yoon* Boost Camp AI Tech iandr0805@gmail.com

Taemin Kim* Boost Camp AI Tech taemin6697@gmail.com

Abstract

Semantic Text Similarity (STS) is an NLP task that aims to quantify the degree of similarity between two or more input sentences, taking into account their respective contexts. Our paper presents research on enhancing the STS model by fine-tuning a pre-trained Huggingface model through transfer learning. Our approach encompasses various aspects, from data preprocessing and augmentation to hyperparameter tuning and ensemble techniques, and has achieved state-of-the-art results on the public leaderboard. Surprisingly, our model was trained for only one hour on a single GPU, which represents only a small fraction of the training costs of the best models in the literature.

1 Introduction

Although the baseline model provided by BoostCamp was based on klue/roberta-small and implemented using Pytorch Lightning, we decided to use a different model pre-trained on a larger dataset and Huggingface Trainer instead of Pytorch Lightning. Prior to delving into the experimental process, we adopted a collaborative approach using GitHub and Notion. Each team member first created their own GitHub branch to plan and conduct individual project experiments. Then, we shared the experimental process and results using GitHub issues, while organizing any new data files or code that needed to be shared privately on Notion.

After daily meetings, we established new standards for the best-performing model and merged it into the main branch, directing the project towards improving the performance of this baseline model in subsequent experiments. This was applied not only to the model itself but also to the dataset, specific preprocessing, and data augmentation methods. As a result, we were able to gradually improve the Pearson correlation coefficient, which is the metric to be maximized, and ultimately achieve first place on the public leaderboard.

*We declare equal contribution from all authors, listed alphabetically. Hyeokjun proposed augmenting the training data by replacing specific words with synonyms and implemented the code using his knowledge of the semantics, syntax, and grammar of the Korean language. Juwon organized the overall process and performed exploratory data analysis on the source data, uncovering critical insights on the data distribution. Minjae suggested analyzing the features with scatter plots to discover new insights from the relationship among multiple features. Sangwon contributed to fitting the data into a uniform distribution by downsampling and augmenting it with several novel approaches. Taemin implemented most of the baseline codes and discovered well-performing pre-trained models.

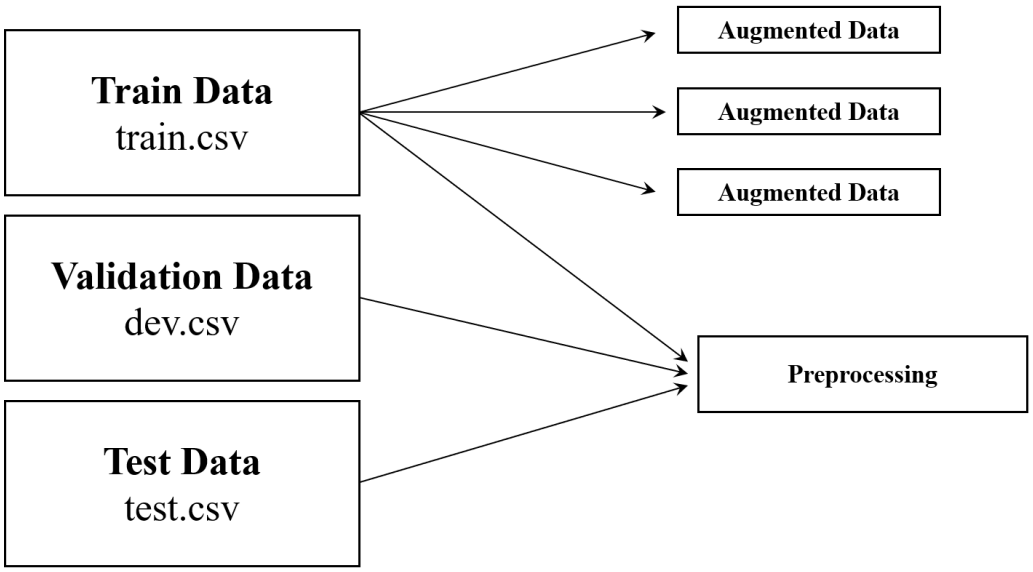

Figure 1: Dataset Structure of our project.

We utilized several methods to augment the train data and implemented a preprocessing code in the dataset class to apply the same preprocessing method to the train, validation, and test data. The dataset structure of the project is illustrated in Figure 1, while the model structure is depicted in Figure 2.

2 Exploratory Data Analysis (EDA)

Before preprocessing or augmenting the data, we performed EDA to gain a better understanding of the data. First, we examined the distribution of the training data by label. We found that the number of data with a label of 0 was overwhelmingly high, while the number of data with a label of 5 was very low. Overall, as the similarity increased, there was a shortage of data. Scores in the middle showed a relatively even distribution.

To prevent potential bias in deep learning models caused by label imbalance in the training data, we aimed to adjust the distribution of the training data to be more uniform. To achieve this goal, we specifically focused on augmenting the data with a label of 5, and the methodology for doing so will be presented in a later section. The label distribution can be found in Figure 2, and although the actual data is labeled in increments of 0.2, the graph was plotted in increments of 0.5.

The training data was collected from three sources (Slack, NSMC, and Petition), and we confirmed that each source contains approximately 3000 data points uniformly. Then, we calculated the string length of each sentence and examined the distribution by source, which was shown in Figure 3. The distribution of sentence 1, sentence 2, and the sum of their lengths were similar across all sources. Therefore, we judged that the probability of bias caused by source or sentence length is relatively low.

We concluded that it is difficult to obtain significant insights by analyzing a single feature, and therefore, we attempted to use a scatter plot to identify the dependencies between multiple features. We used the difference in length between the two sentences contained in each sample as the x-axis and the similarity (label) as the y-axis in the scatter plot and were able to make an important

discovery. We found that the length difference follows a normal distribution regardless of the label, indicating that it has a natural distribution. If the results of data preprocessing or augmentation significantly deviate from this distribution, the model may learn bias due to the length difference. In particular, if this phenomenon occurs in a specific label, we determined that the model may be vulnerable in predicting that label.

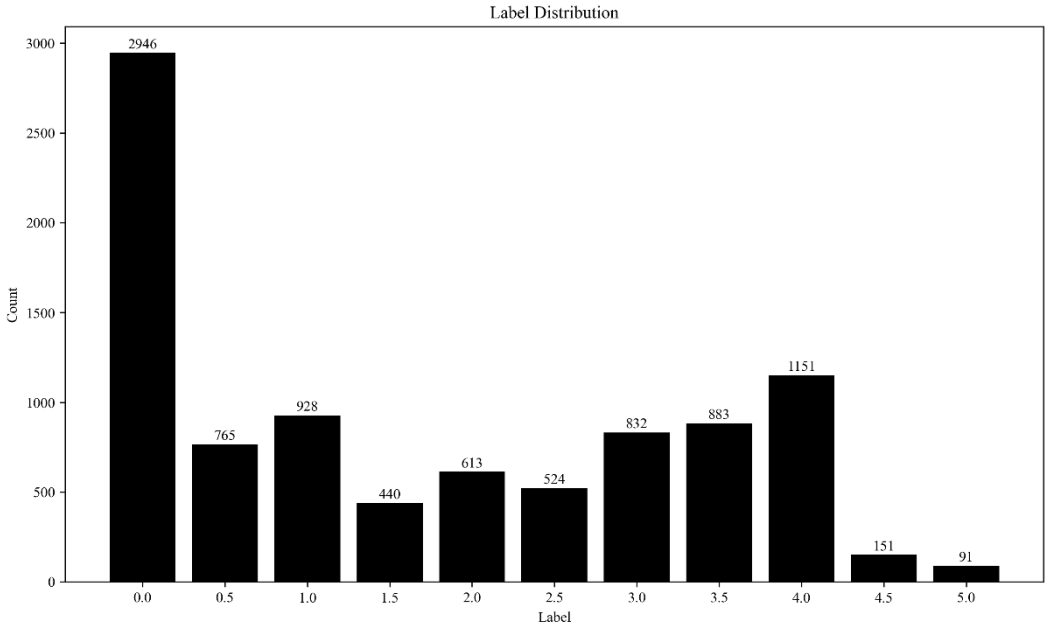

Figure 2: Label Distribution of Train Data.

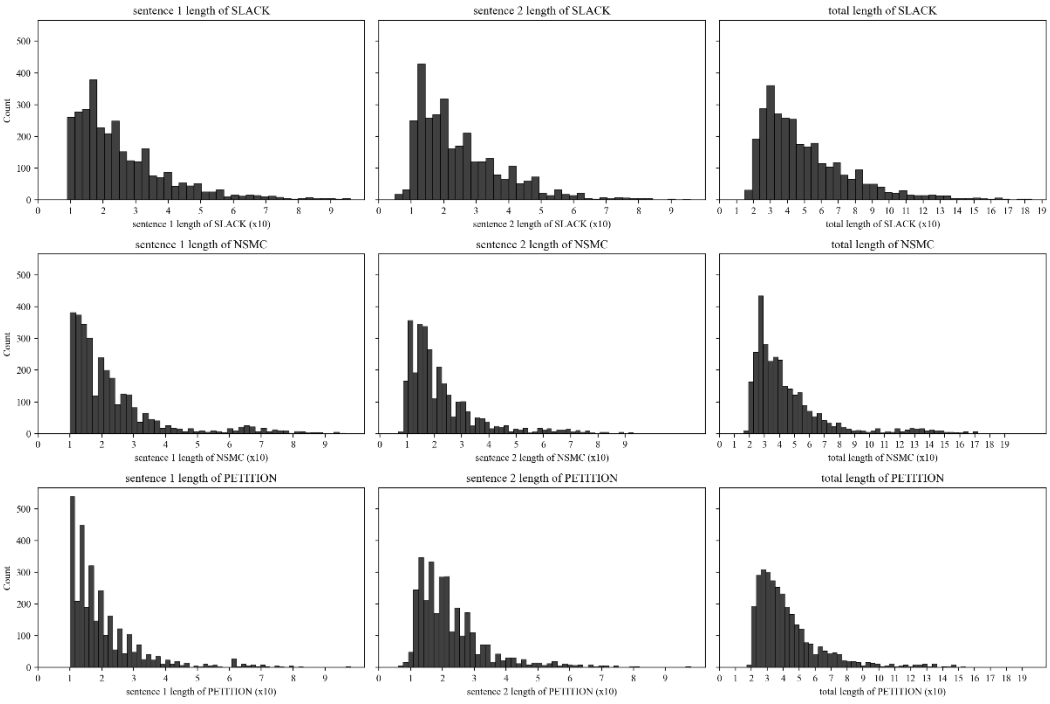

Figure 3: Sentence Length Distribution of each source.

It is widely known and supported by various studies in NLP that models can learn biases based on the length of input. For example, Kenton Murray's paper mentions that the translation output of an NMT model depends on the length of the input sentence [1]. In addition, Sarthak Jain's research pointed out that the performance of the Attention module is recognized, but it learns something from features other than the meaning of the sentence [2]. These studies support our hypothesis, and if any performance degradation occurs due to augmentation or preprocessing, we should consider the possibility of such biases.

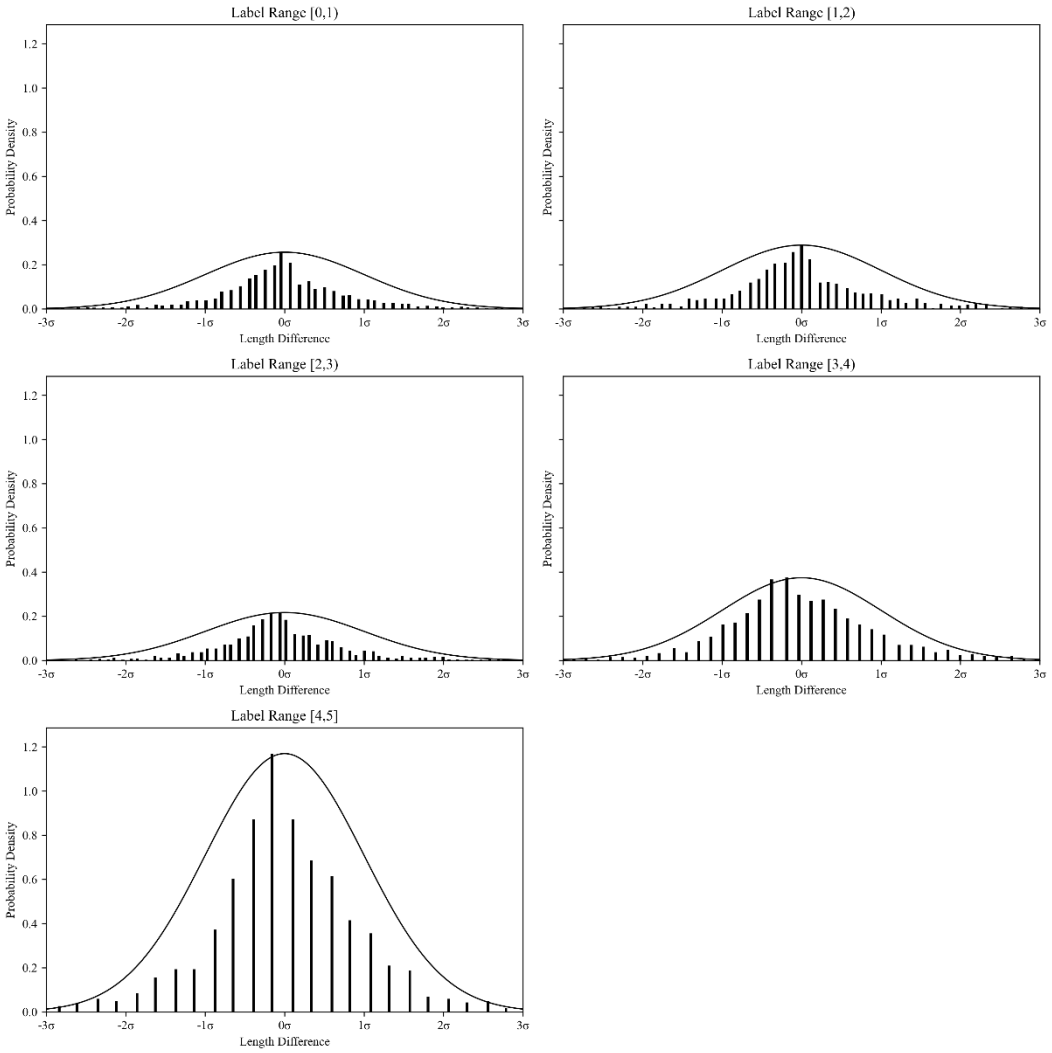

Figure 4: Normal Distribution of Train Data.

3 Data Preprocessing

Although we tried various preprocessing techniques, most of them were not effective in improving the performance. Some of the techniques we attempted include whitespace correction and spelling correction using pykospacing and hanspell libraries, respectively. The results are summarized in Table 1.

We also tried applying preprocessing to the labels, not just the sentences. The methods used were adjusting the labels in 0.5-unit increments or converting them to integers, which resulted in a slight

improvement in model performance. However, we decided not to apply this label preprocessing in the final stage of the ensemble phase, as we believed it could cause information loss and hinder accurate predictions.

Table 1: Preprocessing with Whitespace and Spelling Correction.

Type	**Sentence**
Source	쓰고싶은대로쓰면되지
Preprocessed (Spacing)	쓰고 싶은 대로 쓰면 되지
Source	맞춤법 틀리면 외 않되?
Preprocessed (Hanspell)	맞춤법 틀리면 왜 안돼?

4 Data Augmentation

To improve the performance of the model, sophisticated data EDA and preprocessing are important, but above all, the amount of training data is crucial. We initially had only 9324 training data, which we thought was absurdly insufficient to train a deep learning model. Therefore, we attempted various augmentation techniques to sufficiently increase the amount of training data. Some augmentation techniques resulted in performance improvement while others did not, and we will introduce both in detail.

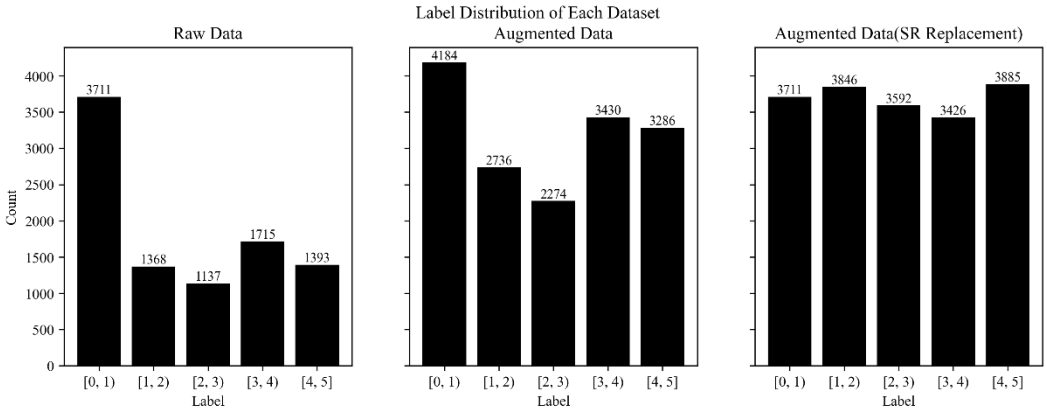

Figure 5: Label Distribution of Each Dataset.

4.1 Final Best Dataset

Data augmentation was carried out to address label imbalance. These are techniques used to add non-repetitive data to the dataset without modifying the context of original sentences. First, samples with overwhelming number of labels 0 were undersampled, and all but 1000 data were removed. To mitigate the effect of sentence length distribution on model training, we extracted data with similar sentence length distribution between the sentences included in undersampled data, and those that were not. Instead, the removed data were used to augment data with label 5. The method was as follows; one of sentence 1 or 2 of samples with label 0 was selected, and copied the selected sentence to create data with label 5 [3].

Data with intermediate similarity were augmented using the swap sentence method, which simply swapped the order of sentence 1 and 2 to augment data. Our rationale was that as sentence positions

are swapped, the values of segment embedding and positional embedding change, which we believed would have the effect of training on new data.

The final dataset was also augmented with synonym replacement. As the model performance degraded when we simply selected a random word from the sentence and replaced it with synonym, we developed a more sophisticated augmentation method [4]. We checked if there are any synonyms for the words commonly included in sentences 1 and 2, and added sentences with the replaced synonyms to the dataset. We utilized the Wordnet synonym dictionary included in python library, KoEDA which is created by KAIST. While augmenting new sentences, we made every effort to preserve the structure of source sentences.

4.2 Tried but did not meet expectations

We tried methods such as removing or adding an arbitrary character in a sentence, as well as removing whitespace. We also attempted adding stop words to acquire new data without compromising the overall meaning of the sentence. However, these techniques did not always lead to improved performance. The reason is that Korean is particularly sensitive to changes in sentence order or the presence of specific characters (such as particles) that can alter the meaning. Therefore, using these methods to modify sentences can lead to significant meaning distortion, which can interfere with training unless labels are adjusted precisely.

After applying an augmentation technique that uses the Papago API to translate sentences into English and then back into Korean, we evaluated the performance and found that it did not significantly improve the model's performance. The reason for this can be attributed to several factors. Firstly, the data was not preprocessed before translation. The original training data contained unrefined expressions and non-dictionary words, which could cause the meaning of the sentences to be distorted during the machine translation process. For example, the emoticon "ㅠㅠ" was translated to "울면서". Therefore, in order for the back translated data to be effectively learned, sophisticated preprocessing that can improve the quality of translation should be applied.

Secondly, we did not address the label imbalance problem when using augmented data with back translation as an independent technique. We concluded that no preprocessing or augmentation technique would have a significant impact without uniform distribution of labels in the data. Moreover, the information loss caused by back translation would add further bias to the training, which could hinder model performance.

Finally, we did not adjust the labels. To account for information loss during translation, we needed to adjust the labels accordingly. However, it was difficult to establish a consistent standard for this, so we simply used the labels of the original samples. This could have caused problems in model training, as the meaning of the sentences may have changed while the labels remained the same.

5 Model Selection

The STS model from Huggingface is mainly used to perform tasks that involve calculating the similarity of each sentence to a reference sentence by receiving multiple sentences as input. However, our task involves receiving two sentences simultaneously and calculating the similarity of the sentence pair. Therefore, we decided to primarily use text classification models by treating each label as a class.

We chose to use the *koelectra-base-v3-discriminator* model for the following reasons. Firstly, it is a pre-trained model on Korean dataset, and it was trained to distinguish whether each token of a

masked sentence was generated by a generator or came from the source sentence. Therefore, we believed that it would capture the meaning of sentences well.

Then we used the *electra-kor-base* model for the following reason. It was trained on various types of text including well-refined language like news, as well as internet slang, abbreviations, and typos that are commonly used on blog-style websites and Namuwiki. We believed that this diverse training data would enable the model to better understand and handle these types of language that are not well-refined. Additionally, the majority of the sentences in the training data that we used for fine-tuning our model, such as those from Slack or NSMC, were in spoken language, so we expected that transfer learning would be successful.

During our experiments, we also used the *kr-electra-discriminator* model, which we chose because it was trained on various sources of datasets and its performance was better than other models with KoELECTRA as the baseline.

Lastly, the model we used is *xlm-roberta-large*. Since the baseline model used *roberta-small*, we hypothesized that a model based on *roberta* would perform well. It is trained on a massive dataset of 2.5TB with 100 languages. Since the amount of training data and performance have a somewhat linear relationship, we expected it to perform well.

We have summarized the hyperparameters used in each model, as well as the scores (Pearson correlation coefficient) on the evaluation stage, including those from the public and private leaderboards, in Table 2.

6 Ensemble

We conducted an ensemble of three models, excluding the koelectra-base-v3-discriminator, as presented in Table 2. Throughout the experiment, we performed multiple model ensembles, but utilized the same model with different hyperparameters. The best-performing model and hyperparameters of each model were as introduced in the Model Selection section. From both model and data perspectives, our ensemble approach was successful as each model utilized distnict types of tokenizers. Specifically, e*lectra-kor-base, kr-electra-discriminator, xlm-roberta-large* perform tokenization at the subword, morpheme, and the sentence-level syntax, respectively. For the ensemble technique, we simply averaged the prediction results of each model. It has been proven that performance generally improves when ensembling different types of models. Indeed, the results of our experiment support this finding. Although we did not attempt various ensemble techniques due to time constraints in the experiment, we anticipate that the effectiveness of ensembling can be further enhanced by trying different techniques, such as using the mean and standard deviation obtained from each model's output for ensembling, in future follow-up research.

Table 2: Experimental settings of each model. LR, Batch, WD, Epoch each indicates the learning rate, batch size, weight decay, and trained epochs.

	LR	**Batch**	**WD**	**Epoch**	**Eval**	**Public**	**Private**
koelectra-base-v3-discriminator	4.38e-5	16	0.4	8	0.9295	0.9114	0.9150
electra-kor-base	2.07e-5	16	0.5	8	0.9319	0.9192	0.9234
kr-electra-discriminator	1.82e-5	4	0.5	8	0.9281	0.9201	0.9267
xlm-roberta-large	9.24e-6	16	0.0	10	0.9306	0.9229	0.9249

7 Analysis

Table 3: The performance of the STS model was evaluated on the validation data, as well as the public and private leaderboards. The public leaderboard was scored using half of the test data, while the private leaderboard was scored with another half of the train data. Throughout the experiments, we only had access to the evaluation and public scores.

System	Eval	Public	Private
Baseline (*roberta-small*)		0.7874	0.8128
koelectra-base-v3-discriminator (raw data)	0.9295	0.9114	0.9150
electra-kor-base (augmented data)	0.9319	0.9192	0.9234
kr-electra-discriminator (augmented data)	0.9281	0.9201	0.9267
xlm-roberta-large (augmented data)	0.9306	0.9229	0.9249
kr-electra-discriminator (augmented data v2)	0.9363	0.9265	0.9354
Ensemble 3 models		**0.9367**	**0.9403**

We can see that our hypothesis that the roberta-based models would be suitable for the task was correct, as the baseline model without any fine-tuning or hyperparameter tuning already achieved a relatively high score. After evaluating the performance with a pre-trained model on a large dataset, we were able to significantly improve the score by simply replacing the model on the baseline, as seen on the public leaderboard where the score increased from 0.7874 to 0.9114.

Then, we applied data preprocessing and augmentation techniques to further enhance the model's performance. For the augmented data, we undersampled 1000 instances of the data with a label value of 0 and used copied translation augmentation to generate more instances with a label value of 5. We also used the swap sentence technique to increase the overall amount of data.

In augmented data v2, we applied more sophisticated techniques, such as synonym replacement, carefully selecting appropriate parts of speech to ensure that the context was not altered. We also paid close attention to the use of particles to maintain the integrity of the context. Other augmentation techniques used in the previous augmented data were also applied.

Unfortunately, due to time constraints, the model using augmented data v2 showed very impressive performance and achieved state-of-the-art results as a single model, but could not be included in the ensemble. By ensembling three models using the original augmented data, we achieved a score of 0.9367 on the public leaderboard, which is equivalent to the SOTA model. Although we fell short of achieving the SOTA model on the private leaderboard, the fact that the score itself has improved shows that our model generalizes very well. If we had used the model using augmented data v2 for the ensemble, we believe we could have achieved even better results.

The code we used to train and evaluate our model is available at `https://github.com/boostcampaitech5/level1_semantictextsimilarity-nlp-11`.

8 Acknowledgement

We want to convey our deep appreciation to each and every team member who, with unwavering passion, researched and experimented tirelessly day and night throughout the project period. Additionally, we would like to express sincere gratitude to mentor, Jaehee Kim for guiding and leading us in the right direction with comprehensive experience and knowledge not only in NLP but also in the overall field of deep learning. We also extend a heartfelt thank you to AI Stages, who supported us with GPU to ensure smooth progress of our research. We feel grateful for all the masters who generously shared their insights through great lectures, and the assistants who helped us

throughout the course. Last but not least, I would like to thank BoostCamp for planning and managing the research that allowed us to experience a great project.

9 Reference

[1] Kenton Murray and David Chiang. 2018. Correcting Length Bias in Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium. Association for Computational Linguistics.

[2] Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), Minneapolis, Minnesota. Association for Computational Linguistics.

[3] Park, Chanjun, Kim, Kuekyeng, Lim, Heuiseok, 2019, Optimization of Data Augmentation Techniques in Neural Machine Translation, Annual Conference on Human and Language Technology, pages 258-261

[4] Jason Wei and Kai Zou, 2019, EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks, EMNLP-IJCNLP, pages 6382–6388