TreeMix: Compositional Constituency-based Data Augmentation for Natural Language Understanding

Le Zhang
Fudan University
18300720002@fudan.edu.cn

Zichao Yang
CMU
yangtze2301@gmail.com

Diyi Yang
Georgia Tech
diyi.yang@cc.gatech.edu

Abstract

Data augmentation is an effective approach to tackle over-fitting. Many previous works have proposed different data augmentations strategies for NLP, such as noise injection, word replacement, back-translation etc. Though effective, they missed one important characteristic of language–compositionality, meaning of a complex expression is built from its subparts. Motivated by this, we propose a compositional data augmentation approach for natural language understanding called TreeMix. Specifically, TreeMix leverages constituency parsing tree to decompose sentences into constituent sub-structures and the Mixup data augmentation technique to recombine them to generate new sentences. Compared with previous approaches, TreeMix introduces greater diversity to the samples generated and encourages models to learn compositionality of NLP data. Extensive experiments on text classification and semantic parsing benchmarks demonstrate that TreeMix outperforms current state-of-the-art data augmentation methods.

1 Introduction

Data augmentation (DA) has won great popularity in natural language processing (NLP) (Chen et al., 2021; Feng et al., 2021) due to the increasing demand for data and the expensive cost for annotation. DA aims at increasing the quantity and diversity of the datasets by generating more samples based on existing ones, which helps make the training process more consistent and improves the model's capacity for generalization (Xie et al., 2020). For instance, existing DA methods often leverage word-level manipulation (Wei and Zou, 2019; Kobayashi, 2018a; Karimi et al., 2021) and model-based sentence generation (Edunov et al., 2018; Ng et al., 2020). As mixup-based (Zhang et al., 2018) augmentation achieving huge success in computer vision (Yun et al., 2019; Uddin et al., 2021; Kim et al., 2021), some recent works start to adapt mixup to NLP, such as at the hidden level (Guo et al., 2019; Chen et al., 2020b) and at the input level (Yoon et al., 2021; Shi et al., 2021).

S1:They will find little interest in this poor film.	
S2:It comes as a touching love story.	
Method	**Example**
EDA (Wei and Zou, 2019)	They will **this** find little interest in **bad movie**.
AEDA (Karimi et al., 2021)	They will find **?** little in **!** this poor movie**;**.
Noise (Xie et al., 2017)	**Thes** will **fi** little **intres** _ this poor film .
SSMix (Yoon et al., 2021)	They will find little interest in **love** poor film
Replace (Kolomiyets et al., 2011)	They will find **limited** interest in this **odd** film.
Back Translation (Edunov et al., 2018)	They will **show** little interest in this **strange** film.
TreeMix	They will find little interest in this **touching love story**.

Table 1: Input-level DAs for text classification. EDA includes random deletion, swapping, and insertion. AEDA randomly inserts punctuation. SSMix swaps tokens based on their saliency. In Back-translation, the source sentences are first translated into another language, and then back again.

Despite these empirical success, DA methods still suffer from key limitations. Simple rules based augmentation methods (Wei and Zou, 2019; Kobayashi, 2018a; Karimi et al., 2021) show little to none effect over large pretrained language models. While mixup-based augmentation methods demonstrate huge potential, such interpolation at the hidden or input level has limited capability to capture explicit linguistic properties in text (Guo et al., 2019; Chen et al., 2020b; Yoon et al., 2021). Moreover, current DA methods exhibit limited ability in compositional generalization. Considering the following examples from a BERT trained on the original SST2 datasets:

This film is good and everyone loves it	😄	99%
The film is poor and I do not like it	😭	99%
This film is good and I do not like it	😄	80%

The first two examples are correctly classified. Despite that the last one is composed of fragments from the first two, the model fails to produce a correct label, demonstrating poor performance in compositional generalization.

However, compositionality is one key aspect of language that the meaning of a complex sentence is built from its subparts. Prior work also shows that syntax trees (e.g., tree-based LSTMs) are helpful to model sentence structures for better text classification (Shi et al., 2018). However, leveraging compositional structures for data augmentation has not received much attention in the language technologies communities, with a few exceptions in semantic parsing (Andreas, 2020; Herzig and Berant, 2020).

To this end, we propose a compositional data augmentation method for natural language understanding, i.e., TreeMix (Figure 1). TreeMix is an input-level mixup method that utilizes constituency parsing information, where different fragments (phrase of a subtree) from different sentences are re-combined to create new examples that were never seen in the training set; new soft labels will also be strategically created based on these fragments at the same time. In this way, TreeMix not only exploits compositional linguistic features to increase the diversity of the augmentation, but also provides reasonable soft labels for these mixed examples.

Empirically, we find that TreeMix outperforms existing data augmentation methods significantly on a set of widely used text classification benchmarks. To validate the compositional effectiveness of TreeMix, we experiment with SCAN (Lake and Baroni, 2018)—a standard compositional generalization benchmark, and find that TreeMix exhibits reasonable ability to generalize to new structures built of components observed during training.

2 Related Work

2.1 Generic Data Augmentation

Most prior work operates data augmentation at different levels (Chen et al., 2021). **Token-level** DA methods manipulate tokens or phrases while preserving syntax and semantic meaning as well as labels of the original text, such as synonymy words substitutions (Wang and Yang, 2015; Zhang et al., 2015; Fadaee et al., 2017a; Kobayashi, 2018b; Miao et al., 2020) where synonyms are detected following pre-defined rules or by word embedding similarities. These methods has limited improvement (Chen et al., 2021) over large pretrained language models (PLMs). Besides, introducing noise by random insertion, replacement, deletion, and swapping (Wang et al., 2018b; Wei and Zou, 2019; Karimi et al., 2021; Xie et al., 2020) is expected to improve the robustness of the model. **Sentence-Level** DA methods increase the diversity by generating distinct examples, such as via paraphrasing (Yu et al., 2018; He et al., 2019; Xie et al., 2020; Kumar et al., 2020; Chen et al., 2020b; Cai et al., 2020) or back translation (Sennrich et al., 2016; Edunov et al., 2018). Other line of work used label-conditioned generation methods that train a conditional generation model such as GPT-2 or VAE to create new examples given labels as conditions (Bergmanis et al., 2017; Liu et al., 2020b,a; Ding et al., 2020; Anaby-Tavor et al., 2020). Although these methods can produce novel and diverse text patterns that do not exist in the original datasets, they require extensive training. **Hidden-Level** DA methods mainly manipulate hidden representations by perturbation (Miyato et al., 2019; Zhu et al., 2020; Jiang et al., 2020; Chen et al., 2020c; Shen et al., 2020; Hsu et al., 2017, 2018; Wu et al., 2019; Malandrakis et al., 2019) and interpolation like mixup (Zhang et al., 2018) to generates new examples (Miao et al., 2020; Cheng et al., 2020; Chen et al., 2020b; Guo et al., 2019, 2020a; Chen et al., 2020a).

2.2 Compositional Data Augmentation

Compositional augmentation aims at increasing the diversity of the datasets and improving the compositional generalization capability of the resulting models (Jia and Liang, 2016; Andreas, 2020). These methods often recombine different components from different sentences to create new examples following a set of pre-designed linguistic rules such as lexical overlaps (Andreas, 2020), neural-symbolic stack machines (Chen et al., 2020d), and substructure substitution (Shi et al., 2021). Compositional methods have been applied in a set of NLP tasks, such as sequence labeling (Guo et al., 2020a), semantic parsing (Andreas, 2020), constituency parsing (Shi et al.,

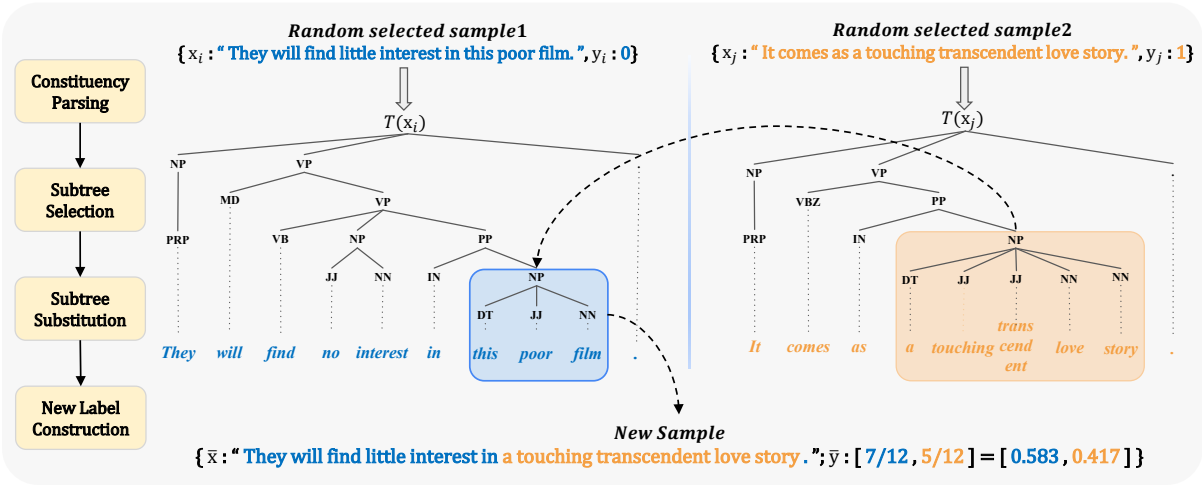

Figure 1: Illustration of TreeMix for single sentence classification

2020, 2021), dependency parsing (Dehouck and Gómez-Rodríguez, 2020; Shi et al., 2021), named entity recognition (Dai and Adel, 2020), text generation (Feng et al., 2020), and text classification (Yoon et al., 2021; Shi et al., 2021). Our work also falls into this category.

The most relevant are Shi et al. (2021) and Yoon et al. (2021). However, Shi et al. (2021) perform constituent substructures combinations to examples from the same category, thus inadequate in creating diverse enough augmentation with newly created labels. Besides, Yoon et al. (2021) simply swaps the most and least salient spans, heavily relying on the model's performances in estimating salient spans, and failing to consider these sentences' linguistic structures. Our proposed TreeMix fills these gaps by allowing the composition of sentences from different label categories, by utilizing rich consistency based structures in text, and by strategically generating soft labels for these augmented instances.

3 Method

Our work is motivated by Mixup (Zhang et al., 2018), which creates virtual samples by mixing inputs. Given two random drawn examples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$, where $\mathbf{x}$ denotes the input sample and $\mathbf{y}$ is the corresponding one-hot label, Mixup creates a new sample by:

$$\overline{\mathbf{x}} = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j,$$
$$\overline{\mathbf{y}} = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j,$$

where $\lambda \in [0, 1]$. Mixup can be easily implemented in continuous space, hence some prior works (Chen et al., 2020b) have extended it to NLP by performing interpolation in hidden space.

We improve upon Mixup by incorporating compositionality of language, a key characteristic that is essential to generalization but neural models often fall short in capturing (Lake and Baroni, 2018). Instead of interpolating with the whole sample, TreeMix, our newly proposed method, creates new sentences by removing phrases of sentences and reinserting subparts from other sentences. TreeMix makes use of constituency trees to decompose a sentence into meaningful constituent parts, which can then be removed and recombined to generate new augmentation samples. We aim to improve models' compositionality generalization ability by training on large amount of samples produced by TreeMix. An example of using TreeMix for single sentence classification is shown in Figure 1.

3.1 TreeMix

Let $\mathbf{x}_i = \{x_i^1, x_i^2, ..., x_i^l\}$ denotes a sequence with length l and its corresponding label in one-hot encoding as $\mathbf{y}_i$. We run a constituency parser on $\mathbf{x}_i$ to get its parsing tree as $T(\mathbf{x}_i)$. In order to get meaningful subparts of a sequence, we traverse the parsing tree recursively and get all the subtrees with more than one child. Denote the collection of subtrees as $S(\mathbf{x}_i) = \{t_i^k\}$, where t_i^k denotes the k-th subtree of sample $\mathbf{x}_i$. For a subtree t_i^k, it covers a continuous span $t_i^k \triangleq [x_i^{r_k}, ..., x_i^{s_k}]$ of $\mathbf{x}_i$ that starts with index r_k and ends with index s_k. For example, as shown in the left part of Figure 1, the subtrees of the example sentence can cover spans such as `this poor film`, `in`

They will find little interest in this poor film.	
$[\lambda_L, \lambda_U]$	**possible selected sub-trees**
[0.1,0.3]	(little interest),(this poor film)
[0.3,0.5]	(in this poor film)
[0.5,0.7]	(little interest in this poor film)

Table 2: Examples of possible candidate subtrees with different λ intervals

`this poor film`, `no interest` etc.

For a given sample $(\mathbf{x}_i, \mathbf{y}_i)$, we randomly sample another data point $(\mathbf{x}_j, \mathbf{y}_j)$ from the training set. We run the constituency parser on both sentences and get their subtree sets $S(\mathbf{x}_i)$ and $S(\mathbf{x}_j)$, based on which we can sample subtrees to exchange. We introduce two additional hyper-parameters λ_L and λ_U to constraint the length of subtrees to sample[1]. λ_L and λ_U, measured in terms of length ratio of the subtree to the original sentences, sets the lower and upper limits of the subtrees to sample. Intuitively, λ controls the granularity of the phrases that we aim to exchange. We would like that the length of phrase to exchange to be reasonable. If it is too short, then the exchange cannot introduce enough diversity to the augmented sample; otherwise if it is too long, the process might inject too much noise to the original sentence. We set λ to be the ratio in order to be invariant to the length of original sentences. Table 2 shows some subtree examples with different length constraints. We define the length constrained subtree set as:

$$S_\lambda(\mathbf{x}) \triangleq \{t | t \in S(\mathbf{x}), s.t. \frac{|t|}{|\mathbf{x}|} \in [\lambda_L, \lambda_U]\}.$$

Here $|.|$ denotes the length of a sequence or a subtree. For two sentences $\mathbf{x}_i$ and $\mathbf{x}_j$, we randomly sample two subtrees $t_i^k \in S_\lambda(\mathbf{x}_i)$ and $t_j^l \in S_\lambda(\mathbf{x}_j)$ and construct a new sample by replacing t_i^k with t_j^l, i.e.

$$\bar{\mathbf{x}} \triangleq [x_i^1, ..., x_i^{r_k - 1}, \underbrace{x_j^{r_l}, ..., x_j^{s_l}}_{t_j^l}, x_i^{s_k+1}, ...x_i^l] \quad (1)$$

where $t_j^l = [x_j^{r_l}, ..., x_j^{s_l}]$ replaces $t_i^k = [x_i^{r_k}, ..., x_i^{s_k}]$. Figure. 1 shows an example of TreeMix, where the subtree `a touching transcend love story` replaces the subtree `this poor film`.

[1]Section D.3 in Appendix presents robustness check on how different length ratio of the subtree affects the results

Algorithm 1: Dataset construction

Input: Original dataset $\mathcal{D}$; data size multiplier β; parameters λ_L and λ_U
Output: Augmentation Dataset $\mathcal{D}'$
while $|\mathcal{D}'| < \beta|\mathcal{D}|$ **do**
  Randomly select two samples $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{D}$
  $(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \text{TreeMix}((\mathbf{x}_i, \mathbf{y}_i), (\mathbf{x}_j, \mathbf{y}_j))$
  $\mathcal{D}' \leftarrow \mathcal{D}' \cup \{(\bar{\mathbf{x}}, \bar{\mathbf{y}})\}$
end

Label Creation for TreeMix Creating a valid label for the augmented sample $\bar{\mathbf{x}}$ is a challenging problem. Similar to that of Mixup (Zhang et al., 2018), we use a convex combination of original labels of two sentences as the new label for the augmented sample:

$$\bar{\mathbf{y}} = \frac{l_i - |t_i^k|}{l_i - |t_i^k| + |t_j^l|}\mathbf{y}_i + \frac{|t_j^l|}{l_i - |t_i^k| + |t_j^l|}\mathbf{y}_j, \quad (2)$$

where l_i is the length of $\mathbf{x}_i$ and $|t_i^k|, |t_j^k|$ are the length of the subtrees. In the new sentence, $l_i - |t_i^k|$ words from $\mathbf{x}_i$ are kept and $|t_j^l|$ words from sentence $\mathbf{x}_j$ are inserted.

In Equation 2, $\frac{l_i - |t_i^k|}{l_i - |t_i^k| + |t_j^l|}$ is the fraction of words that come from $\mathbf{x}_i$, which determines the weight of $\mathbf{y}_i$. The label is then created based on the conjecture that the change in labels is proportional to the length changes in the original sentences. We provided a set of augmentation examples from TreeMix in Table A.1 in Appendix.

Pairwise Sentence Classification Task The above mainly used single sentence classification as the running example for TreeMix. Here we argue that TreeMix can easily be extended to pairwise sentence classification problem, where the relationship between the sentences is the label.

Formally, for a given sample $(\mathbf{x}_i, \mathbf{x}_i', \mathbf{y}_i)$, we randomly sample another sample $(\mathbf{x}_j, \mathbf{x}_j', \mathbf{y}_j)$ and run the parser and get the subtree sets of each sentence $S(\mathbf{x}_i), S(\mathbf{x}_i')$ and $S(\mathbf{x}_j), S(\mathbf{x}_j')$. Then we randomly sample subtrees $t_i^k \in S_\lambda(\mathbf{x}_i), t_{i'}^{k'} \in S_\lambda(\mathbf{x}_i')$ and $t_j^l \in S_\lambda(\mathbf{x}_j), t_{j'}^{l'} \in S_\lambda(\mathbf{x}_j')$. We construct $\bar{\mathbf{x}}$ by replacing t_i^k with t_j^l and $\bar{\mathbf{x}}'$ by replacing

$t_{i'}^{k'}$ with $t_{j'}^{l'}$. The new label is created as:

$$\bar{\mathbf{y}} = \frac{l_i + l_{i'} - |t_i^k| - |t_{i'}^{k'}|}{l_i + l_{i'} - |t_i^k| - |t_{i'}^{k'}| + |t_j^l| + |t_{j'}^{l'}|}\mathbf{y}_i + \frac{|t_j^l| + |t_{j'}^{l'}|}{l_i + l_{i'} - |t_i^k| - |t_{i'}^{k'}| + |t_j^l| + |t_{j'}^{l'}|}\mathbf{y}_j. \quad (3)$$

The meanings of the notations are the same as in Equation 2.

Our main algorithm is shown in Algorithm 1. Although not all sentences created by TreeMix are fluent or even valid new sentences, they contains subparts with different meanings that encourage the models to build rich representation of sentences in a compositional manner. Note that the augmented labels are convex combination of original labels, only when the model learns the representations of two parts together can they predict both labels with different weights.

3.2 Training Objective

Our model is trained on a combination of the original samples and augmentation samples to obtain a trade-off between regularization and noise injection. The final training objective is:

$$\mathcal{L} = \mathop{\mathbb{E}}_{(\mathbf{x},\mathbf{y})\sim D}\left[-\mathbf{y}^\intercal \log P_\theta(\mathbf{y}|\mathbf{x})\right] + \gamma \mathop{\mathbb{E}}_{(\bar{\mathbf{x}},\bar{\mathbf{y}})\sim D'}\left[-\bar{\mathbf{y}}^\intercal \log P_\theta(\bar{\mathbf{y}}|\bar{\mathbf{x}})\right], \quad (4)$$

γ is the weight[2] on the augmentation samples.

4 Experiment

4.1 Datasets

To test TreeMix's effectiveness, we experiment with a variety of text classification benchmarks, as shown in Table 3. We use accuracy as a metric, and exclude datasets from GLUE (Wang et al., 2018a) that are not suitable for mixup, including CoLA that measures linguistic acceptability and will be ruined by mixup operations, and WNLI that is too small to show a method's validity.

4.2 Experiment Setup

The proposed TreeMix method creates new samples by combining text spans based on the constituency tree's information, thus we use the Stanford CoreNLP toolkit to obtain parsing related information (Manning et al., 2014). We use the pretrained language model *bert-base-uncased* for sequence classification task from HuggingFace. With seeds ranging from 0 to 4 and $\lambda_L = 0.1, \lambda_U = 0.3$, we use TreeMix to generate twice and five times more samples than the original training set. We replicate the original dataset to the same size as the augmentation datasets in the training stage to ensure that the model receives the same amount of data from the original dataset and the augmentation dataset for each training batch.

If not specified, we train the model for 5 epochs, with a maximum sequence length of 128 and batch size of 96. The model is optimized using the AdamW optimizer with an eps of 1e-8 and a learning rate of 2e-5. Table C.1 in Appendix contains detailed hyper-parameter settings for each dataset.

Dataset	Task	Class	Size
Single Sentence Classification			
SST2	Sentiment	2	67k/1.8k
TREC-f	Question	47	5.5k/500
TREC-c	Question	6	5.5k/500
AG_NEWS	News	4	12k/4k
IMDb	Sentiment	2	12.5k/12.5k
Pair Sentence Classification			
RTE	NLI	2	3.5k/300
MRPC	Paraphrase	2	3.7k/400
QNLI	QA	2	105k/5.5k
QQP	Paraphrase	2	364k/40.4k
MNLI	NLI	3	393k/9.8k

Table 3: Dataset name or split name, task category and number of label class, Size used for training and testing. For tasks from GLUE, Size indicates (#train:#validation); for TREC, AG_NEWS and IMDb, Size indicates (#train:#test)

4.3 Baseline

We compare TreeMix with the following benchmarks: (1) No augmentation (BERT): standard training without any augmentation, (2) EDA that randomly performs insertion, replacement, swap and deletion to the text. (3) AEDA that randomly inserts punctuation to the text. (4) Back translation(BT) (Fadaee et al., 2017b): texts are translated between English and German using Transformer architectures trained on WMT16 English-German. (5) GPT3Mix(Yoo et al., 2021) designs prompts and utilizes GPT3 to generate new examples to train the model. (6) SSMix (Yoon et al., 2021) applies mixup based on the saliency (Si-

[2]Section B in Appendix presents discussions on how the objective and different weight parameter affects the result.

monyan et al., 2014) of tokens, similar to PuzzleMix (Kim et al., 2020) and SaliencyMix (Uddin et al., 2021). (7) EmbedMix is the pretrained-language-model version of WordMixup in Guo et al. (2019), which performs mixup on the embedding level. (8) TMix (Chen et al., 2020b) first encodes two inputs separately, then performs the linear interpolation of two embeddings at a certain encoder layer, and finally forward-passes the combined embedding in the remaining layers.

5 Results and Analysis

5.1 Performance On Full Dataset

The results of TreeMix on the entire datasets are shown in Table 1. TreeMix outperforms all baselines significantly on single sentence classification tasks, demonstrating the superiority of using compositional substructure for substitution and augmentation. For instance, On SST2, it improves by 0.98%. Compared to other methods, the improvement was more than doubled.

This is because that, unlike SSMix which substitutes the text spans based on the saliency, our TreeMix makes use of the constituency information to help identify linguistically informed sentence substructures, and by recombining these components, the compositional diversity of the datasets can be maximized. With our TreeMix generated samples, the model can see more combinations of the substructures in the training stage that aren't available in the original corpus, leading to better generalization ability.

When it comes to sentence relationship classification tasks, TreeMix is also very effective. For example, It improves by 2.47% on the RTE data set, whereas the best improvement of other methods is only 0.3%, and it improves by 0.82% on QNLI, where other data augmentation methods have little effect. We hypothesized that, when two constituent parts from one sentence pair are embedded into another sentence pair, the inherent relationship is also embedded. This better helps the models on how to to identify two pairs of relationships in a single sample, which further increases its capacity to categorize these challenging adversarial sentences. Since TreeMix works by increasing dataset diversity and providing models with more text patterns to learn, it has very significant improvements for these relatively small datasets such as RTE and TREC, compared to these large datasets such as AG NEWS,QQP and MNLI that already have a lot of diversity and text patterns.

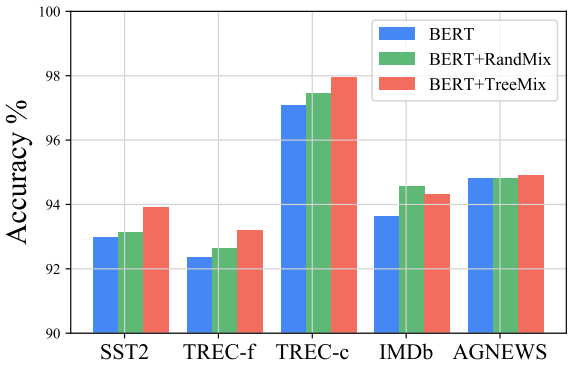

Figure 2: Performance of RandMix and TreeMix on single sentence classification datasets, scores are averaged over 5 random seeds.

5.2 Influence of Constituency Information

To determine the importance of constituency information, we designed a Random Mixup (RandMix) that randomly selects text spans as long as the ratio of span length to sentence length is less than a particular threshold.[3], λ_{rand}. The rest setting of RandMix is the same as TreeMix. We compare TreeMix and RandMix on single sentence classification datasets in Figure 2.

We found that, both RandMix and TreeMix are quite effective, but TreeMix outperforms RandMix on most datasets. For instance, TreeMix exceeds RandMix by 0.8% on SST2, 0.6% on TREC-f, and 0.5% on TREC-c One exception is on IMDb, where the average sentence length is much longer. The reason for the poorer performance of TreeMix is due to the sparse parsing results on long sentences; since there are many subtrees, substituting any single part might bring very minimal change to the entire sentence.

5.3 Influence of Training Set Size

To examine the influence of TreeMix with different training set sizes[4], we uniformly sample 1%, 2%, 5%, 10%, and 20% of the data from the training set to investigate TreeMix in low-resource situations. The entire test set is used to evaluate the model's generalization ability. Since TreeMix generates more examples for training, we use RandMix to generate the same number of extra samples as a comparison to ensure the data size

[3]We observed $\lambda_{rand} \sim \mathcal{U}(0, 0.3)$ is optimal and we use this settings for the experiment

[4]Section D.1 in Appendix presents robustness check on how different amount of augmented data affects the result.

Model	Single Sentence Classification					Pair Sentence Classification				
	SST2	TREC-f	TREC-c	IMDb	AG NEWS	MRPC	RTE	QNLI	QQP	MNLI
BERT	92.96†	92.36	97.08†	93.63	94.67	84.90	68.15	90.54	90.67	84.27†
BERT+EDA	92.20†	91.95	96.79†	93.62	94.67	-	-	-	-	-
BERT+AEDA	92.57†	92.15	97.20†	93.59	94.22	-	-	-	-	-
BERT+BT	92.48	92.15	96.68	-	-	82.13	67.40	-	-	-
BERT+GPT3Mix	93.25†	-	-	-	-	-	-	-	-	-
BERT+SSMix	93.14†	92.80	97.60†	93.74	94.64	84.31	68.40	90.60	90.75	**84.54**†
BERT+EmbedMix	93.03†	92.32	97.44†	93.72	**94.72**	85.34	68.37	90.44	90.58	84.35†
BERT+TMix	93.03†	92.68	97.52†	93.69	94.69	**85.69**	68.45	90.48	90.66	84.30†
BERT+TreeMix	**93.92**	**93.20**	**97.95**	**94.34**	**94.72**	85.34	**70.62**	**91.36**	**90.88**	84.45

† denotes the result is extracted from the original paper

Table 4: Results of comparison with baseline on full datasets. Scores are averaged over 5 random seeds. For GLUE tasks, we report accuracy of validation sets, and for other datasets we report test accuracy. EDA and AEDA will seriously damage the sentence relationship and harm the accuracy; GPT3Mix only reports full data experiments results on SST2 in original paper. We only report the results of back translation on small dataset due to the heavy computational cost.

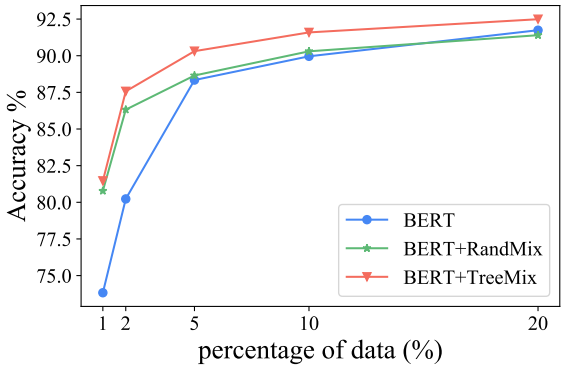

Figure 3: Results on SST2 varying data size. Scores are averaged over 5 random seeds.

Datasets	BERT	TM(same)	TM(cross)
SST2	92.96	93.78	**93.92**
TREC-f	92.36	92.60	**93.20**
TREC-c	97.08	97.74	**97.95**
IMDb	93.63	94.22	**94.34**
AGNEWS	94.67	94.47	**94.71**
MRPC	84.90	85.34	**85.34**
RTE	68.15	70.25	**70.62**
QNLI	90.54	90.87	**91.36**
QQP	90.67	90.85	**90.88**
MNLI	84.27	84.33	**84.45**

Table 5: Performance with TreeMix performed (1) within same classes TM(same) and (2) cross different classes TM(cross), averaged over 5 runs.

is fair. The results are summarized in Figure 3.

We found that, (1) TreeMix outperforms RandMix in all settings, further demonstrating the advantage of the compositional substructure with the constituency information over the randomly selected spans. (2) Both mixup methods can significantly improve the model's performance in the case of extreme data scarcity (e.g, 1% and 2%). (3) When the amount of data is sufficient (e.g, more than 5%), TreeMix outperforms RandMix by a significant margin. However, TreeMix only slightly outperforms RandMix when there is a severe lack of data (e.g, 1% and 2%). This is due to that the too small datasets often contain very limited structures, thus constraining TreeMix's ability to increase text patterns and compositional diversity. (4) The relative improvement of TreeMix over conventional training without augmentation diminishes as the amount of data increases, largely due to that additional augmented text patterns might overlap with those already existing in the dataset, resulting in limited improvement.

5.4 Influence of Cross-Category Mixing

Different from prior work Shi et al. (2021), TreeMix allows the composition of sentences from different label categories. To test whether this cross-label category mixup is more effective than a within-label category mixup, we conducted ablation studies with TreeMix on samples in the same class. Table 5 shows the results. Across all datasets, we found that TreeMix that combines data from different classes is more effective than combining data from the same class, consistent with findings in Zhang et al. (2018). When given only labels from one category, current models have a tendency to make simple or spurious judgments based on the most frequently occurring words. However the semantics of the sentence are

complicated beyond simple words. For example, the model is likely to classify a sentence like *"I like this good movie"* as positive because of the words *"like"* and *"good"*, but if *"good movie"* is replaced with *"bad film"*, the model must understand the adversarial constituent parts within the sentence, which we refer to the ability to distinguish adversarial patterns in one sample. This ability can only be obtained when the model is trained on the cross-category generated samples.

5.5 Compositional Generalization

To quantify TreeMix's overall ability of compositional generalization beyond classification tasks, we conducted experiments on the SCAN (Lake and Baroni, 2018) dataset, which is a command execution dataset widely used to test for systematic compositionality. It contains simple source commands and target action sequences. We test on commonly used challenging splits: *addprim-jump, addprim-turn-left, around-right*, where primitive commands (e.g "*jump*") only appear alone during training but will be combined with other modifiers (e.g "*jump twice*") during testing. A model that works well for this task should learn to compose the primitive commands with the modifiers and generates corresponding execution. With TreeMix, we can generate the compositional commands that are not seen in the training set.

The new command generation process is the same as in single sentence classification, except that we increase the length constraint λ_2 to 1 to allow the exchange of the commands with only one word. After we synthesize new commands, we follow the rules in Lake and Baroni (2018) to translate the source commands into the target actions. We follow the settings in Andreas (2020) and use the following data augmentation methods as baselines: (1) WordDrop that drops words randomly; (2) SwitchOut (Wang et al., 2018c) that randomly replaces words with other random words from the same vocabulary; (3) SeqMix (Guo et al., 2020b) which randomly swaps text spans of different examples, and (4) GECA (Andreas, 2020) that performs enumerated valid swaps.

As shown in Table 6, TreeMix outperforms SwitchOut and WordDrop for all splits. TreeMix by itself does not perform as well as GECA, but when being combined with GECA, it demonstrates very strong results. TreeMix outperforms SeqMix in all splits, due to the fact that TreeMix can more precisely find the linguistically rich compositional segments of a sentence, as evidenced by the results of the comparisons of TreeMix and SSMix in Section 5.1 and TreeMix and RandMix in Section 5.3. A closer look at these augmented samples show that TreeMix can generate all possible combinations of "jump" and other modifiers like "left" and "around"; these previously unseen command combinations further validates TreeMix's ability to improve the dataset's compositional diversity. TreeMix demonstrates weak performances on the *around-right* split, where the model observes commands "*around*" and "*right*" in isolation at the training stage, and it has to derive the meaning of "*around right*" at the test time. Because the word "*around*" cannot be parsed as a single subtree for swap. Instead, it always appears in a subtree with the word "*left*", preventing TreeMix from generating the phrase "*turn right*". Despite its limitations on *around-left*, TreeMix performs well on all other splits and can be easily combined with other data augmentation methods, demonstrating the compositional generalization ability of TreeMix beyond classification tasks.

Method	JUMP	TURN-L	AROUND-R
Baseline	0†	49% †	0†
WordDrop	0†	51% †	0†
SwitchOut	0†	16% †	0†
SeqMix	49%†	99% †	0†
TreeMix	**72%**	**99%**	**0%**
GECA	87%†	-	82†
GECA+WordDrop	51%†	-	61†
GECA+SwitchOut	77%†	-	73†
GECA+SeqMix	98%†	-	89 †
GECA+TreeMix	**99%**	-	**91%**

† denotes the result is extracted from the original paper

Table 6: Experimental results (accuracy) on SCAN.

6 Conclusion

This work introduced TreeMix, a compositional data augmentation approach for natural language understanding. TreeMix leverages constituency parsing tree to decompose sentences into substructures and further use the mixup data augmentation technique to recombine them to generate new augmented sentences. Experiments on text classification and semantic parsing benchmarks demonstrate that TreeMix outperforms prior strong baselines, especially in low-resource settings and compositional generalization.

References

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7383–7390.

Jacob Andreas. 2020. Good-enough compositional data augmentation.

Toms Bergmanis, Katharina Kann, Hinrich Schütze, and Sharon Goldwater. 2017. Training data augmentation for low-resource morphological inflection. *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*.

Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. 2020. Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2021. An empirical survey of data augmentation for limited data learning in nlp.

Jiaao Chen, Zhenghui Wang, Ran Tian, Zichao Yang, and Diyi Yang. 2020a. Local additivity based data augmentation for semi-supervised ner. In *EMNLP*.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020b. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification.

Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. 2020c. Seqvat: Virtual adversarial training for semi-supervised sequence labeling. In *ACL*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Xiaodong Song, and Denny Zhou. 2020d. Compositional generalization via neural-symbolic stack machines. *ArXiv*, abs/2008.06662.

Yong Cheng, Lu Jiang, Wolfgang Macherey, and Jacob Eisenstein. 2020. Advaug: Robust adversarial augmentation for neural machine translation. *ArXiv*, abs/2006.11834.

Xiang Dai and Heike Adel. 2020. An analysis of simple data augmentation for named entity recognition. *ArXiv*, abs/2010.11683.

Mathieu Dehouck and Carlos Gómez-Rodríguez. 2020. Data augmentation via subtree swapping for dependency parsing of low-resource languages. In *COLING*.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen, Shafiq Joty, Luo Si, and Chunyan Miao. 2020. Daga: Data augmentation with a generation approach for low-resource tagging tasks. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017a. Data augmentation for low-resource neural machine translation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017b. Data augmentation for low-resource neural machine translation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*.

Steven Y. Feng, Varun Gangal, Dongyeop Kang, Teruko Mitamura, and Eduard Hovy. 2020. Genaug: Data augmentation for finetuning text generators. *Proceedings of Deep Learning Inside Out (DeeLIO): The First Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*.

Steven Y. Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. 2021. A survey of data augmentation approaches for nlp.

Demi Guo, Yoon Kim, and Alexander M. Rush. 2020a. Sequence-level mixed sample data augmentation. In *EMNLP*.

Demi Guo, Yoon Kim, and Alexander M. Rush. 2020b. Sequence-level mixed sample data augmentation.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019. Augmenting data with mixup for sentence classification: An empirical study.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2019. Revisiting self-training for neural sequence generation.

Jonathan Herzig and Jonathan Berant. 2020. Span-based semantic parsing for compositional generalization. *arXiv preprint arXiv:2009.06040*.

Wei-Ning Hsu, Hao Tang, and James R. Glass. 2018. Unsupervised adaptation with interpretable disentangled representations for distant conversational speech recognition. *ArXiv*, abs/1806.04872.

Wei-Ning Hsu, Yu Zhang, and James Glass. 2017. Unsupervised domain adaptation for robust speech recognition via variational autoencoder-based data augmentation. *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *ArXiv*, abs/1911.03437.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. Aeda: An easier data augmentation technique for text classification.

Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. 2021. Co-mixup: Saliency guided joint mixup with supermodular diversity.

Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup.

Sosuke Kobayashi. 2018a. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, New Orleans, Louisiana. Association for Computational Linguistics.

Sosuke Kobayashi. 2018b. Contextual augmentation: Data augmentation by words with paradigmatic relations.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2011. Model-portability experiments for textual temporal analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, volume 2, pages 271–276. ACL; East Stroudsburg, PA.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models.

Brenden M. Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Jiancheng Lv, Nan Duan, and Ming Zhou. 2020a. Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. 2020b. Data boost: Text data augmentation through reinforcement learning guided conditional generation. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nikolaos Malandrakis, Minmin Shen, Anuj Goyal, Shuyang Gao, Abhishek Sethi, and Angeliki Metallinou. 2019. Controlled text generation for data augmentation in intelligent artificial agents. *ArXiv*, abs/1910.03487.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Zhengjie Miao, Yuliang Li, Xiaolan Wang, and Wang-Chiew Tan. 2020. Snippext: Semi-supervised opinion mining with augmented data. *Proceedings of The Web Conference 2020*.

Takeru Miyato, Shin-Ichi Maeda, Masanori Koyama, and Shin Ishii. 2019. Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Dinghan Shen, Ming Zheng, Yelong Shen, Yanru Qu, and Weizhu Chen. 2020. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *ArXiv*, abs/2009.13818.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. *ArXiv*, abs/2010.02423.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2021. Substructure substitution: Structured data augmentation for nlp.

Haoyue Shi, Hao Zhou, Jiaze Chen, and Lei Li. 2018. On tree-based neural sentence modeling. In *EMNLP*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep inside convolutional networks: Visualising image classification models and saliency maps.

A. F. M. Shahab Uddin, Mst. Sirazam Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. 2021. Saliencymix: A saliency guided data augmentation strategy for better regularization.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. ArXiv preprint 1804.07461.

William Yang Wang and Diyi Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic

categorization of annoying behaviors using petpeeve tweets. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018b. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 856–861, Brussels, Belgium. Association for Computational Linguistics.

Xinyi Wang, Hieu Pham, Zihang Dai, and Graham Neubig. 2018c. Switchout: an efficient data augmentation algorithm for neural machine translation.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks.

Zhanghao Wu, Shuai Wang, Yanmin Qian, and Kai Yu. 2019. Data augmentation using variational autoencoder for embedding based speaker verification. In *INTERSPEECH*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2020. Unsupervised data augmentation for consistency training.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models.

Kang Min Yoo, Dongju Park, Jaewook Kang, Sang-Woo Lee, and Woomyeong Park. 2021. Gpt3mix: Leveraging large-scale language models for text augmentation.

Soyoung Yoon, Gyuwan Kim, and Kyumin Park. 2021. Ssmix: Saliency-based span mixup for text classification. *arXiv preprint arXiv:2106.08062*.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization.

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

Chen Zhu, Yu Cheng, Zhe Gan, S. Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. *arXiv: Computation and Language*.

A Augmentation examples

Original Sentence1	Original Sentence2	New Sentence
a love story and **a murder mystery** that expands into a meditation on the deep deceptions of innocence [1]	really an advantage to invest such subtlety and warmth in an animatronic bear when the humans **are acting like puppets** [0]	a love story and are acting like puppets that expands into a meditation on the deep deceptions of innocence [0.21 0.79]
the attempt to build up **a pressure cooker** of horrified awe [0]	had the ability to mesmerize , astonish and entertain [1]	the attempt to build up the ability of horrified awe [0.8 0.2]
rest contentedly with the knowledge that he 's made **at least one** damn fine horror movie [1]	**minor film** [0]	rest contentedly with the knowledge that he 's made minor film damn fine horror movie [0.13 0.87]
might just be better suited to **a night in the living** room than a night at the movies [0]	are made for **each other** . [1]	might just be better suited to each other room than a night at the movies [0.86 0.14]
is **a touching reflection** on aging , suffering and the prospect of death [1]	keep upping the ante on **each other** [1]	is each other on aging , suffering and the prospect of death [0 1]
is dark , brooding and slow , and takes **its central idea way**too seriously [0]	**merely pretentious** [0]	is dark , brooding and slow , and takes merely pretentious too seriously [1. 0.]

Table A.1: Examples of TreeMix on SST2 datasets, the number following sentence is label, bold tokens are selected phrase for substitution

B The necessity of merged loss techniques

We provide a detailed discussion of the techniques proposed in 3.2. We first investigate the noise contained in the augmentation dataset, then we figure out how the unbalance dataset will affect the performance. In the second part, we vary the weight parameter γ to see how it affects the model's learning process.

B.1 Noise and Unbalance

All mixup methods, as previously stated, introduce noise into the dataset. This noise in the text includes grammatical structure confusion and multiple semantic meanings in the sentences. The model will be overwhelmed by the noise if trained solely on the generated augmentation dataset, and will even perform worse than the baseline. In terms of the unbalance problem, we find that training the model without replicating the original dataset to the same size as the augmentation dataset hurts the model's performance. The results are shown in the table B.1.

	SST2	TREC-f	TREC-c	IMDb	AG NEWS	MRPC	RTE	QNLI	QQP	MNLI
BERT	92.96	92.36	97.08	93.63	94.67	84.9	68.15	90.54	90.67	84.27
Merged Loss	**93.92**	**93.2**	**97.95**	**94.34**	**94.72**	**85.34**	**70.63**	**91.36**	**90.88**	**84.45**
Augmentation only	92.57	90.44	96.42	92.37	93.98	83.93	65.45	88.72	89.24	83.78
No Replicate	93.05	92.42	97.21	93.7	94.65	85.02	69.56	91.04	90.72	84.35

Table B.1: Merged Loss indicates results following techniques in 3.2, Augmentation indicates the model is trained on the generated dataset alone. No Replicate indicates Merged Loss without replication of the original training set.

B.2 Weight parameter

We vary weight parameter γ to find optimal balance point between diversity and linguistic grammar, the results are shown in figure 4. Performance on the two classification tasks follows a similar pattern. Both increase with increasing weight and then rapidly decrease with increasing weight after reaching the highest point. Performance is weaker than the baseline when the weight value exceeds 0.7. We find the model achieves the best performance with $\gamma \in \{0.2, 0.5\}$. For single sentence classification tasks, when $\gamma = 0.5$ the model always gets higher accuracy, and $\gamma = 0.2$ is better for these sentence relation classification datasets.

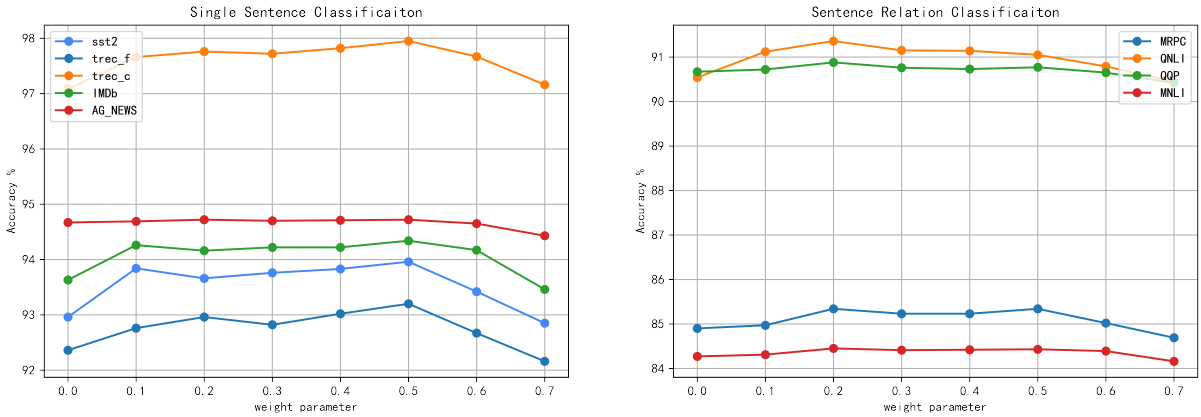

Figure 4: The performance when varying the value of the weight parameter on single sentence classification (left) and sentence relation classification (right)

C Hyper-parameters for each datasets

Datasets	epoch	batch size	aug batch size	val steps	sequence length	aug weight
SST2	5	96	96	100	128	0.5
TREC-f	20	96	96	100	128	0.5
TREC-c	20	96	96	100	128	0.5
IMDb	5	8	8	500	512	0.5
AGNEWS	5	96	96	500	128	0.5
MRPC	10	32	32	100	128	0.2
RTE	5	32	32	50	128	-0.2
QNLI	5	96	96	100	128	0.2
QQP	5	96	96	300	128	0.2
MNLI	5	96	96	100	128	0.2

Table C.1: Best settings for different datasets

We explore different parameter combinations and find the best ones for each task, as in Tab C.1. There are some exceptions, such as TREC datasets, where the model cannot converge even with 10 epochs, so we increase the training epochs to 20 for this dataset. IMDb's examples are extremely long, with an average length of more than 200 words. Along with this change, we increased the truncation length to 512 and the batch size to 8 to fully capture the semantic meaning. RTE is the most unusual. First, when we train using original RTE datasets, the accuracy deviation is really substantial, reaching up to 4%. Second, we find that $\gamma = -0.2$ is optimum for this set, which contradicts previous findings.

D Ablation Study

Shi et al. (2021) has proposed a similar study that uses constituency information for mixup. There are a few significant differences between our approaches. To begin with, their method is too restricted; they only perform mixup between examples from the same category, and they require the substituted subtree's label to be the same. Second, because they are limited to the same class examples, they are unable to devise a method for adding a soft label to the example. Instead, we only use TreeMix in the previous settings with the length constraint. Several other constraints in the subtree selection process are investigated in this section, and we achieve better performance than Shi et al. (2021) by giving the subtree selection process more freedom, and we validate that their work is a special case of our method by examining how other constraints affect the performance. This section's values are the averages of five runs with seeds ranging from 0 to 4

D.1 What is the difference between different amounts of data?

TreeMix has the potential to generate an infinite amount of augmented data in theory. However, due to TreeMix's principle, it can only improve performance to a point when the size of the augmentation data set reaches a certain limit. We investigated how many augmentation datasets the model needs. Table D.1 shows the results of producing twice and five times the augmentation data for experiments.

Dataset	Size	BERT	TM(x2)	TM(x5)
RTE	3.5k	68.15	70.57	**70.62**
MRPC	3.7k	84.90	85.22	**85.37**
TREC-f	5.5k	92.36	**93.2**	92.85
TREC-c	5.5k	97.08	97.71	**97.95**
IMDb	12.5k	93.63	**94.34**	94.24
SST2	67k	92.96	**93.92**	93.92
QNLI	105k	90.54	**91.36**	91.34
AGNEWS	120k	94.67	**94.71**	94.69
QQP	364k	90.67	**90.88**	90.83
MNLI	393k	84.27	**84.45**	84.41

Table D.1: Improvement of performance on all datasets with different amount of augmentation datasets, *TM(x2)* indicates generating twice as much augmentation data than the original data, *TM(x5)* indicates five times than original data, datasets in the table are listed in order of size

The key to getting the best results is to strike a balance between the original datasets and the augmentation datasets in terms of diversity and linguistic confusion. With more augmentation datasets, the model will learn more patterns while also observing more grammatically poor samples, which could negatively impact performance. We discovered that augmentation datasets twice the size of the original dataset produce the best results for larger datasets. This is in line with our previous theoretical analysis: large datasets inherently include more patterns and diversity, which helps the model generalize better. Maintaining the original linguistic grammar while increasing diversity in these datasets is, therefore, more important. When working with smaller datasets, it's better to train with more augmentation data. For models to train on these datasets, we believe diversity is more important than linguistic grammar.

TREC-fine is an exception. We attribute it to the datasets' excessive classes (up to 47 classes within only 5.5k training samples): each class has a very limited number of samples, and if we create overly augmented dataset samples, the limited samples of each category are insufficient to resist injected linguistic noise. As a result, for TREC-fine, x2 is preferable to x5. For a smaller dataset, we recommend generating five times as much augmentation data as possible, and for a larger dataset, we recommend generating twice as much augmentation data.

D.2 Is it beneficial to keep the label and subtree lengths the same?

Each subtree has its own label (e.g., VP and NP) and corresponds to a specific text span. When selecting subtrees, we can use these characteristics as additional constraints. Figure 5 shows the results. When we impose restrictions on the subtree selection process, the experimental results clearly show that performance suffers.

We hypothesize that this is because in datasets with similar sentence lengths, subtrees of the same phrase label or phrase length tend to have similar structures (e.g., tree height, relative position in the sentence). Although the exchange of such subtrees can retain the original linguistic grammar of the text to some extent (e.g., replacing a noun phrase with another noun phrase will not significantly disrupt the sentence) and maintain similar sentence length, it cannot exploit the potential compositional diversity in the datasets as efficiently as TreeMix without any constraints, resulting in lower diversity augmentation datasets and limited improvement compared to the baseline. In terms of the comparison of *TreeMix(label)* and *TreeMix(length)*, we find that *TreeMix(label)* prefers simple phrases such as NP and VP because these are the most common phrases occurring in sentences, and this exchange will not improve the diversity of the datasets. For example, in "I like this apple," replacing "apple" with "orange" will not provide innovative text patterns.

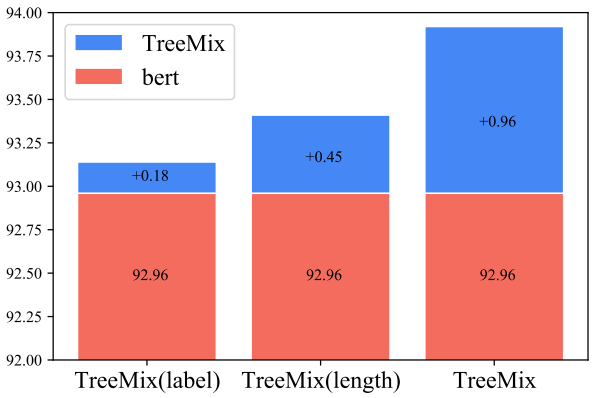

Figure 5: Performance on SST2 with different subtree selection constraints, green part is bert performance, orange part is improvement of TreeMix when applying different constraints, TreeMix(label) indicates only select subtrees with same phrase label, TreeMix(length) indicates only select subtrees with same length. TreeMix indicates without any constraints

D.3 How does the length of the subtree affect the result?

Dataset	BERT	λ_2 =0.3	λ_2 =0.5
SST2	92.96	**93.92**	93.05
TREC-fine	92.36	**93.2**	92.25
TREC-coarse	97.08	**97.95**	96.94
IMDb	93.63	**94.34**	93.29
AG NEWS	94.67	**94.72**	94.53
MRPC	84.90	**85.34**	84.93
RTE	68.15	**70.62**	70.35
QNLI	90.54	**91.36**	90.78
QQP	90.67	**90.88**	90.54
MNLI	84.27	**84.45**	83.78

Table D.2: Performance with different length ratio λ_2

The only constraint we place on TreeMix is the length. We choose subtrees that account for 30% and 50% of the sentence, respectively, by setting $\lambda_L = 0.1$ and varying λ_U from 0.3 to 0.5. Table ref D.2 shows the results.

On all datasets, $\lambda_U = 0.3$ outperforms $\lambda_U = 0.5$, which is consistent with Zhang et al. (2018), which stated that giving "lambda" too high values for mixup ration can result in underfitting. Another linguistic explanation for the scenario follows: When $\lambda_U = 0.5$, TreeMix may select long text spans, which usually contain unique constituency components like *SBAR*; The exchange of these spans will severely damage the text's semantic and grammatical structure, causing the model to become confused. As a result, TreeMix with $\lambda_U = 0.5$ performs poorly, and even worse than baseline on some datasets.