Data Augmentation Using Pre-trained Transformer Models

Varun Kumar
Alexa AI
kuvrun@amazon.com

Ashutosh Choudhary
Alexa AI
ashutoch@amazon.com

Eunah Cho
Alexa AI
eunahch@amazon.com

Abstract

Language model based pre-trained models such as BERT have provided significant gains across different NLP tasks. In this paper, we study different types of transformer based pre-trained models such as auto-regressive models (GPT-2), auto-encoder models (BERT), and seq2seq models (BART) for conditional data augmentation. We show that prepending the class labels to text sequences provides a simple yet effective way to condition the pre-trained models for data augmentation. Additionally, on three classification benchmarks, pre-trained Seq2Seq model outperforms other data augmentation methods in a low-resource setting. Further, we explore how different data augmentation methods using pre-trained model differ in-terms of data diversity, and how well such methods preserve the class-label information.

1 Introduction

Data augmentation (DA) is a widely used technique to increase the size of the training data. Increasing training data size is often essential to reduce overfitting and enhance the robustness of machine learning models in low-data regime tasks.

In natural language processing (NLP), several word replacement based methods have been explored for data augmentation. In particular, Wei and Zou (2019) showed that simple word replacement using knowledge bases like WordNet (Miller, 1998) improves classification performance. Further, Kobayashi (2018) utilized language models (LM) to augment training data. However, such methods struggle with preserving class labels. For example, non-conditional DA for an input sentence of sentiment classification task *"a small impact with a big movie"* leads to *"a small movie with a big impact"*. Using such augmented data for training, with the original input sentence's label (i.e. negative sentiment in this example) would negatively impact the performance of the resulting model.

To alleviate this issue, Wu et al. (2019) proposed conditional BERT (CBERT) model which extends BERT (Devlin et al., 2018) masked language modeling (MLM) task, by considering class labels to predict the masked tokens. Since their method relies on modifying BERT model's segment embedding, it cannot be generalized to other pre-trained LMs without segment embeddings.

Similarly, Anaby-Tavor et al. (2019) used GPT2 (Radford et al., 2019) for DA where examples are generated for a given class by providing class as input to a fine-tuned model. In their work, GPT2 is used to generate 10 times the number of examples required for augmentation and then the generated sentences are selected based on the model confidence score. As data selection is applied only to GPT2 but not to the other models, the augmentation methods can not be fairly compared. Due to such discrepancies, it is not straightforward to comprehend how the generated data using different pre-trained models varies from each other and their impact on downstream model performance.

This paper proposes a unified approach to use any pre-trained transformer (Vaswani et al., 2017) based models for data augmentation. In particular, we explore three different pre-trained model types for DA, including 1) an autoencoder (AE) LM: BERT, 2) an auto-regressive (AR) LM: GPT2, and 3) a pre-trained seq2seq model: BART (Lewis et al., 2019). We apply the data generation for three different NLP tasks: sentiment classification, intent classification, and question classification.

In order to understand the significance of DA, we simulate a low-resource data scenario, where we utilize only 10 training examples per class in a classification task. Section 3.2 provides details of the task and corpora.

We show that all three types of pre-trained models can be effectively used for DA, and using the generated data leads to improvement in classification performance in the low-data regime setting. Among three types of methods, pre-trained seq2seq model provides the best performance. Our code is available at [1].

Our contribution is three-fold: (1) implementation of a seq2seq pre-trained model based data augmentation, (2) experimental comparison of different data augmentation methods using conditional pre-trained model, (3) a unified data augmentation approach with practical guidelines for using different types of pre-trained models.

2 DA using Pre-trained Models

LM pre-training has been studied extensively (Radford et al., 2018; Devlin et al., 2018; Liu et al., 2019). During pre-training, such models are either trained in an AE setting or in an AR setting. In the AE setting, certain tokens are masked in the sentence and the model predicts those tokens. In an AR setting, the model predicts the next word given a context. Recently, pre-training for seq2seq model has been explored where a seq2seq model is trained for denoising AE tasks (Lewis et al., 2019; Raffel et al., 2019). Here, we explore how these models can be used for DA to potentially improve text classification accuracy.

Algorithm 1: Data Augmentation approach

```
Input : Training Dataset $D_{train}$
        Pretrained model $G \in \{AE, AR, Seq2Seq\}$
1 Fine-tune $G$ using $D_{train}$ to obtain $G_{tuned}$
2 $D_{synthetic} \leftarrow \{\}$
3 foreach $\{x_i, y_i\} \in D_{train}$ do
4     Synthesize $s$ examples $\{\hat{x_i}, \hat{y_i}\}_p^1$ using $G_{tuned}$
5     $D_{synthetic} \leftarrow D_{synthetic} \cup \{\hat{x_i}, \hat{y_i}\}_p^1$
6 end
```

DA Problem formulation: Given a training dataset $D_{train} = \{x_i, y_i\}_n^1$, where $x_i = \{w_j\}_m^1$ is a sequence of m words, y_i is the associated label, and a pre-trained model G, we want to generate a dataset of $D_{synthetic}$. Algorithm 1 describes the data generation process. For all augmentation methods, we generate $s = 1$ synthetic example for every example in D_{train}. Thus, the augmented data is same size as the size of the original data.

2.1 Conditional DA using Pre-trained LM

For conditional DA, a model G incorporates label information during fine-tuning for data generation. Wu et al. (2019) proposed CBERT model where they utilized BERT's segment embeddings to condition model on the labels. Similarly, models can be conditioned on labels by prepending labels y_i to x_i (Keskar et al., 2019; Johnson et al., 2017).

Due to segment embedding reuse, CBERT conditioning is very specific to BERT architecture thus cannot be applied directly to other pre-trained LMs. Thus, we compare two generic ways to condition a pre-trained model on class label:

- `prepend` : prepending label y_i to each sequence x_i in the training data without adding y_i to model vocabulary
- `expand` : prepending label y_i to each sequence x_i in the training data and adding y_i to model vocabulary.

Note that in `prepend`, the model may split y_i into multiple subword units (Sennrich et al., 2015b; Kudo and Richardson, 2018), `expand` treats a label as a single token.

Here, we discuss the fine-tuning and the data generation process for both AE and AR LMs. For transformer based LM implementation, we use Pytorch based transformer package (Wolf et al., 2019). For all pre-trained models, during fine-tuning, we further train the learnable parameters of G using its default task and loss function.

2.1.1 Fine-tuning and generation using AE LMs

We choose BERT as a representative of AE models. For fine-tuning, we use the default masking parameters and MLM objective which randomly masks some of the tokens from the raw sequence, and the objective is to predict the original token of the masked words using the context. Both BERT$_{\text{prepend}}$ and BERT$_{\text{expand}}$ models are fine-tuned using the same objective.

2.1.2 Fine-tuning and generation using AR LMs

For AR LM experiments, we choose GPT2 as a generator model and follow the method proposed by Anaby-Tavor et al. (2019) to fine-tune and generate

[1]`https://github.com/varinf/TransformersDataAugmentation`

data. For fine-tuning GPT2, we create a training dataset by concatenating all sequences in D_{train} as follows: $y_1 SEP x_1 EOS y_2 ... y_n SEP x_n EOS$. SEP denotes a separation token between label and sentence, and EOS denotes the end of a sentence.

For generating data, we provide $y_i SEP$ as a prompt to G, and we keep generating until the model produces EOS token. We use GPT2 to refer to this model. We found that such generation struggles in preserving the label information, and a simple way to improve the generated data label quality is to provide an additional context to G. Formally, we provide $y_i SEP w_1..w_k$ as prompt where $w_1..w_k$ are the first k words of a sequence x_i. In this work, we use $k = 3$. We call this method GPT2$_{\text{context}}$.

2.2 Conditional DA using Pre-trained Seq2Seq model

Like pre-trained LM models, pre-training seq2seq models such as T5 (Raffel et al., 2019) and BART (Lewis et al., 2019) have shown to improve performance across NLP tasks. For DA experiments, we choose BART as a pre-trained seq2seq model representative for its relatively lower computational cost.

2.2.1 Fine-tuning and generation using Seq2Seq BART

Similar to pre-trained LMs, we condition BART by prepending class labels to all examples of a given class. While BART can be trained with different denoising tasks including insertion, deletion, and masking, preliminary experiments showed that masking performs better than others. Note that masking can be applied at either word or subword level. We explored both ways of masking and found subword masking to be consistently inferior to the word level masking. Finally, we applied word level masking in two ways:

- BART$_{\text{word}}$: Replace a word w_i with a mask token $< mask >$
- BART$_{\text{span}}$: Replace a continuous chunk of k words $w_i, w_{i+1}..w_{i+k}$ with a single mask token $< mask >$.

Masking was applied to 40% of the words. We fine-tune BART with a denoising objective where the goal is to decode the original sequence given a masked sequence.

2.3 Pre-trained Model Implementation

2.3.1 BERT based DA models

For AutoEncoder (AE) experiments, we use "bert-base-uncased" model with the default parameters provided in huggingface's transformer package. In `prepend` setting we train model for 10 epochs and select the best performing model on dev data partition keeping initial learning rate at $4e^{-5}$. For `expand` setting, training requires 150 epochs to converge. Moreover, a higher learning rate of $1.5e^{-4}$ was used for all three datasets. The initial learning rate was adjusted for faster convergence. This is needed for `expand` setting as embeddings for labels are randomly initialized.

2.3.2 GPT2 model implementation

For GPT2 experiments, we use GPT2-Small model provides in huggingface's transformer package. We use default training parameters to fine-tune the GPT2 model. For all experiments, we use SEP as a separate token and $<|\ endoftext\ |>$ as EOS token. For text generation, we use the default nucleus sampling (Holtzman et al., 2019) parameters including $top\_k = 0$, and $top\_p = 0.9$.

2.3.3 BART model implementation

For BART model implementation, we use fairseq toolkit (Ott et al., 2019) implementation of BART. Additionally, we used bart_large model weights[2].

Since BART model already contains $< mask >$ token, we use it to replace mask words. For BART model fine-tuning, we use denoising reconstruction task where 40% words are masked and the goal of the decoder is to reconstruct the original sequence. Note that the label y_i is prepended to each sequence x_i, and the decoder also produces the label y_i as any other token in x_i. We use fairseq's *label_smoothed_cross_entropy* criterion with a *label-smoothing* of 0.1. We use $1e^{-5}$ as learning rate. For generation, beam search with a beam size of 5 is used.

2.4 Base classifier implementation

For the text classifier, we use "bert-base-uncased" model. The BERT model has 12 layers, 768 hidden states, and 12 heads. We use the pooled representation of the hidden state of the first special token ([CLS]) as the sentence representation. A dropout probability of 0.1 is applied to the sentence representation before passing it to the Softmax layer.

[2]`https://dl.fbaipublicfiles.com/fairseq/models/bart.large.tar.gz`

Adam (Kingma and Ba, 2014) is used for optimization with an initial learning rate of $4e^{-5}$. We use 100 warmup steps for BERT classifier. We train the model for 8 epochs and select the best performing model on the dev data.

All experiments were conducted using a single GPU instance of Nvidia Tesla v100 type. For BART model, we use f16 precision. For all data augmentation models, validation set performance was used to select the best model.

3 Experimental Setup

3.1 Baseline Approaches for DA

In this work, we consider three data augmentation methods as our baselines.

(1) **EDA** (Wei and Zou, 2019) is a simple word-replacement based augmentation method, which has been shown to improve text classification performance in the low-data regime.

(2) **Backtranslation** (Sennrich et al., 2015a) is another widely used text augmentation method (Shleifer, 2019; Xie et al., 2019; Edunov et al., 2018). For backtranslation, we use a pre-trained EN-DE[3], and DE-EN[4] translation models (Ng et al., 2019).

(3) **CBERT** (Wu et al., 2019) language model which, to the best of our knowledge, is the latest model-based augmentation that outperforms other word-replacement based methods.

3.2 Data Sets

We use three text classification data sets.

(1) **SST-2** (Socher et al., 2013): (Stanford Sentiment Treebank) is a dataset for sentiment classification on movie reviews, which are annotated with two labels (Positive and Negative).

(2) **SNIPS** (Coucke et al., 2018) dataset contains 7 intents which are collected from the Snips personal voice assistant.

(3) **TREC** (Li and Roth, 2002) is a fine-grained question classification dataset sourced from TREC. It contains six question types (whether the question is about person, location, etc.).

For SST-2 and TREC, we use the dataset versions provided by (Wu et al., 2019)[5], and for SNIPS dataset, we use [6]. We replace numeric class labels with their text versions. For our experiments, we used the labels provided in Table 1. Note that pre-trained methods rely on different byte pair encodings that might split labels into multiple tokens. For all experiments, we use the lowercase version of the class labels.

3.2.1 Low-resourced data scenario

Following previous works to simulate the low-data regime setting for text classification (Hu et al., 2019), we subsample a small training set on each task by randomly selecting an equal number of examples for each class.

In our preliminary experiments, we evaluated classification performance with various degrees of low-data regime settings, including 10, 50, 100 examples per class. We observed that state-of-the-art classifiers, such as the pre-trained BERT classifier, performs relatively well for these data sets in a moderate low-data regime setting. For example, using 100 training examples per class for SNIPS dataset, BERT classifier achieves 94% accuracy, without any data augmentation. In order to simulate a realistic low-resourced data setting where we often observe poor performance, we focus on experiments with 10 and 50 examples per class. Note that using a very small dev set leads the model to achieve 100% accuracy in the first epoch which prevents a fair model selection based on the dev set performance. To avoid this and to have a reliable development set, we select ten validation examples per class.

3.3 Evaluation

To evaluate DA, we perform both intrinsic and extrinsic evaluation. For **extrinsic evaluation**, we add the generated examples into low-data regime training data for each task and evaluate the performance on the full test set. All experiments are repeated 15 times to account for stochasticity. For each experiment, we randomly subsample both training and dev set to simulate a low-data regime.

For **intrinsic evaluation**, we consider two aspects of the generated text. The first one is semantic fidelity, where we measure how well the generated text retains the meaning and the class information of the input sentence. In order to measure this, we train a classifier on each task by fine-tuning a pre-trained English BERT-base uncased model. Section

[3] https://dl.fbaipublicfiles.com/fairseq/models/wmt19.en-de.joined-dict.single_model.tar.gz

[4] https://dl.fbaipublicfiles.com/fairseq/models/wmt19.de-en.joined-dict.single_model.tar.gz

[5] https://github.com/1024er/cbert_aug

[6] https://github.com/MiuLab/SlotGated-SLU/tree/master/data/snips

Data	Label Names
SST-2	Positive, Negative
TREC	Description, Entity, Abbreviation, Human, Location, Numeric
SNIPS	PlayMusic, GetWeather, RateBook, SearchScreeningEvent, SearchCreativeWork, AddToPlaylist, BookRestaurant

Table 1: Label Names used for fine-tuning pre-trained models. Label names are lower-cased for all experiments.

	SST-2	SNIPS	TREC
Train	6,228	13,084	5,406
Dev	692	700	546
Test	1,821	700	500

Table 2: Data statistics for three corpora, without any sub-sampling. This setup is used to train a classifier for intrinsic evaluation, as described in Section 3.3. When simulating low-data regime, we sample 10 or 50 training examples from each category. For testing, we use the full test data.

3.3.1 describes corpus and classifier performance details.

Another aspect we consider is text diversity. To compare different models' ability to generate diverse output, we measured type token ratio (Roemmele et al., 2017). Type token ratio is calculated by dividing the number of unique n-grams by the number of all n-grams in the generated text.

3.3.1 Classifiers for intrinsic evaluation

In this work, we measure semantic fidelity by evaluating how well the generated text retains the meaning and the label information of the input sentence. To measure this, we fine-tune the base classifier described in Section 2.4.

To take full advantage of the labeled data and to make our classifier more accurate, we combine 100% of training and test partitions of the corresponding dataset, and use the combined data for training. Then, the best classifier is selected based on the performance on the dev partition. Classification accuracy of the best classifier on dev partition for each corpus is provided in Table 3.

	SST-2	SNIPS	TREC
Dev	91.91	99	94.13

Table 3: Classifier performance on dev set for each corpus. Classifiers are used for intrinsic evaluation.

4 Results and Discussion

4.1 Generation by Conditioning on Labels

As described in Section 2.1, we choose BERT as a pre-trained model and explored different ways of conditioning BERT on labels: BERT$_{\text{prepend}}$, BERT$_{\text{expand}}$ and CBERT.

Table 4 shows BERT$_{\text{prepend}}$, BERT$_{\text{expand}}$ and CBERT have similar performance on three datasets. Note that BERT is pre-trained on a very huge corpus, but fine-tuning is applied on a limited data. This makes it difficult for the model to learn new, meaningful label representations from scratch as in case the BERT$_{\text{expand}}$. While CBERT and BERT$_{\text{prepend}}$ both converge in less than 8 epochs, BERT$_{\text{expand}}$ requires more than 100 epochs to converge.

Further, the class conditioning technique used in CBERT is specific to BERT architecture which relies on modifying BERT's segment embedding and hence cannot be applied to other model architectures. Since the labels in most of the datasets are well-associated with the meaning of the class (e.g. *SearchCreativeWork*), prepending tokens allows the model to leverage label information for conditional word replacement. Given these insights, we recommend `prepend` as a preferred technique for pre-trained model based data augmentation.

4.2 Pre-trained Model Comparison

Classification Performance Table 4 shows that seq2seq pre-training based BART outperforms other DA approaches on all data sets. We also observe that back translation (shown as BackTrans. in table) is a very strong baseline as it consistently outperforms several pre-trained data augmentation techniques including CBERT baseline.

Generated Data Fidelity As described in Section 3.3.1, we train a classifier for each dataset and use the trained classifier to predict the label of the generated text.

Table 5 shows that AE based methods outperform AR models like GPT2, and Seq2seq based

Model	SST-2	SNIPS	TREC
No Aug	52.93 (5.01)	79.38 (3.20)	48.56 (11.53)
EDA	53.82 (4.44)	85.78 (2.96)	52.57 (10.49)
BackTrans.	57.45 (5.56)	86.45 (2.40)	66.16 (8.52)
CBERT	57.36 (6.72)	85.79 (3.46)	64.33 (10.90)
BERT$_{\text{expand}}$	56.34 (6.48)	86.11 (2.70)	65.33 (6.05)
BERT$_{\text{prepend}}$	56.11 (6.33)	86.77 (1.61)	64.74 (9.61)
GPT2$_{\text{context}}$	55.40 (6.71)	86.59 (2.73)	54.29 (10.12)
BART$_{\text{word}}$	**57.97** (6.80)	86.78 (2.59)	63.73 (9.84)
BART$_{\text{span}}$	57.68 (7.06)	**87.24** (1.39)	**67.30** (6.13)

Table 4: DA extrinsic evaluation in low-data regime. Results are reported as Mean (STD) accuracy on the full test set. Experiments are repeated 15 times on randomly sampled training and dev data. For data augmentation model fine-tuning, we use 10 examples per class for training.

Model	SST-2	SNIPS	TREC
EDA	95.00	97.14	87.22
BackTrans.	**96.66**	97.14	**94.88**
CBERT	96.33	**97.90**	92.22
BERT$_{\text{expand}}$	95.00	97.04	91.44
BERT$_{\text{prepend}}$	**96.66**	97.80	94.33
GPT2$_{\text{context}}$	68.33	92.47	60.77
BART$_{\text{word}}$	89.33	91.03	79.33
BART$_{\text{span}}$	90.66	93.04	80.22

Table 5: Semantic fidelity of generated output. We trained a classifier using all labelled data in order to perform accuracy test on the generated data. Higher accuracy score means that the model retains the class label of the input sentence more accurately. For data augmentation model fine-tuning, we use 10 examples per class for training.

model like BART, in terms of semantic fidelity of the generated data. On two datasets, back translation approach outperforms all other methods in terms of fidelity which underlines the effectiveness of the state of the art translation systems in terms of preserving the semantics of the language.

Generated Data Diversity To further analyze the generated data, we explore type token ratio as described in Section 3.3. Table 6 shows that EDA generates the most diverse tri-grams and back translation approach produces the most diverse unigrams. Since EDA method modifies tokens at random, it leads to more diverse n-grams, not necessarily preserving the semantic of the input sentence. Also, unlike AE and Seq2seq methods that rely on word or span replacements, back translation is an open-ended system that often introduces unseen unigrams.

Model	SST-2		SNIPS		TREC	
n-gram	1	3	1	3	1	3
EDA	0.66	0.99	0.49	**0.97**	0.55	**0.97**
BackTrans	**0.68**	0.99	**0.51**	0.95	**0.57**	0.96
CBERT	0.57	0.99	0.48	0.95	0.46	0.95
BERT$_{\text{expand}}$	0.59	0.99	0.49	0.96	0.47	0.95
BERT$_{\text{prepend}}$	0.57	0.99	0.48	0.95	0.46	0.95
GPT2$_{\text{context}}$	0.62	0.99	0.34	0.88	0.44	0.92
BART$_{\text{word}}$	0.53	0.99	0.42	0.95	0.40	0.93
BART$_{\text{span}}$	0.55	0.99	0.41	0.91	0.39	0.89

Table 6: Type token ratio for generated text using each model. For data augmentation model fine-tuning, we use 10 examples per class for training.

4.3 Guidelines For Using Different Types Of Pre-trained Models For DA

AE models : We found that simply prepending the label to raw sequences provides competitive performance than modifying the model architecture. As expected, more complex AE models such as RoBERTa$_{\text{prepend}}$ (Liu et al., 2019) outperforms BERT$_{\text{prepend}}$ (66.12 vs 64.74 mean acc on TREC).

AR models : While AR based model such as GPT2 produces very coherent text, it does not preserve the label well. In our experiments, we found that providing a few starting words along with the label as in GPT2$_{\text{context}}$ is crucial to generate meaningful data.

Seq2Seq models : Seq2Seq models provide an opportunity to experiment with various kinds of denoising autoencoder tasks including masking at subword, word or span level, random word insertion or deletion. We observe that word or span masking performs better than other denoising objectives, and should be preferred for DA.

Overall, we found that while AE models are constrained to produce similar length sequences and are good at preserving labels, AR models excel at unconstrained generation but might not retain label information. Seq2Seq models lie between AE and AR by providing a good balance between diversity and semantic fidelity. Further, in Seq2Seq models, diversity of the generated data can be controlled by varying the masking ratio.

4.4 Limitations

Our paper shows that a pre-trained model can be used for conditional data augmentation by fine-tuning it on the training data where the class labels are prepended to the training examples. Such a unified approach allows utilizing different kinds of pre-trained models for text data augmentation to improve performance in low-resourced tasks. However, as shown in Table 7, improving pre-trained classifier's model performance in rich-resource setting is still challenging.

Our results also show that a particular pre-trained model based augmentation may do well on one task or dataset, but may not work well for other scenarios. In our experiments, we use the same set of hyperparameters such as masking rate, learning rate and warmup schedule for all three datasets which might not lead to the best performance for all considered tasks. While our primary goal in this work is to propose a unified data augmentation technique, we believe that further studies on optimizing performance for a given task or model will be beneficial.

5 Conclusion And Future Work

We show that AE, AR, and Seq2Seq pre-trained models can be conditioned on labels by prepending label information and provide an effective way to augment training data. These DA methods can be easily combined with other advances in text content manipulation such as co-training the data generator and the classifier (Hu et al., 2019). Further, the proposed data augmentation techniques can also be combined with latent space augmentation (Kumar et al., 2019). We hope that unifying different DA methods would inspire new approaches for universal NLP data augmentation.

References

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2019. Not enough data? deep learning to the rescue! *arXiv preprint arXiv:1911.03118*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv preprint arXiv:1805.10190*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Sergey Edunov, Myle Ott, M. Auli, and David Grangier. 2018. Understanding back-translation at scale. *ArXiv*, abs/1808.09381.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration.

Zhiting Hu, Bowen Tan, Russ R Salakhutdinov, Tom M Mitchell, and Eric P Xing. 2019. Learning data manipulation for augmentation and weighting. In *Advances in Neural Information Processing Systems*, pages 15738–15749.

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. 2017. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. *arXiv preprint arXiv:1805.06201*.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and William Campbell. 2019. A closer look at feature space data augmentation for few-shot intent classification. *arXiv preprint arXiv:1910.04176*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer.

2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Melissa Roemmele, Andrew S Gordon, and Reid Swanson. 2017. Evaluating story generation systems using automated linguistic analyses. In *SIGKDD 2017 Workshop on Machine Learning for Creativity*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015a. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015b. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Sam Shleifer. 2019. Low resource text classification with ulmfit and backtranslation.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jason W Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Conditional bert contextual augmentation. In *International Conference on Computational Science*, pages 84–95. Springer.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V. Le. 2019. Unsupervised data augmentation for consistency training.

A Appendices

A.1 Classification performance on 50 examples per class

Model	SST-2	SNIPS	TREC
Num examples	50	50	50
No Aug	**78.60** (2.81)	90.98 (2.30)	71.65 (11.09)
EDA	76.41 (4.90)	89.80 (2.99)	61.98 (11.52)
BackTrans	78.30 (6.30)	89.87 (3.35)	65.52 (11.82)
CBERT	77.26 (4.56)	90.57 (2.11)	67.77 (13.81)
BERT$_{\text{expand}}$	78.15 (4.56)	89.79 (2.56)	68.06 (12.20)
BERT$_{\text{prepend}}$	77.96 (4.78)	90.41 (2.39)	**71.88** (9.91)
GPT2$_{\text{context}}$	74.91 (5.43)	88.87 (3.33)	51.41 (11.35)
BART$_{\text{word}}$	76.35 (5.84)	**91.40** (1.60)	71.58 (7.00)
BART$_{\text{span}}$	77.92 (4.96)	90.62 (1.79)	67.28 (12.57)

Table 7: DA extrinsic evaluation in low-data regime. Results are reported as Mean (STD) accuracy on the full test set. Experiments are repeated 15 times on randomly sampled training, dev data. 50 training examples are subsampled randomly.

Table 7 shows the classification performance of different models when we select 50 examples per class. Overall, we find that data augmentation does not improve classification performance, and in many cases, it even hurts the accuracy.