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 mod-
els 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 trained for denoising AE tasks (Lewis et al., 2019; Raffel et al., 2019). 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 $\{\hat{x}_i, \hat{y}_i\}_s$ using $G_{tuned}$
5. $D_{synthetic} \leftarrow D_{synthetic} \cup \{\hat{x}_i, \hat{y}_i\}_s$
6. end

DA Problem formulation: Given a training dataset $D_{train} = \{x_i, y_i\}_n$, where $x_i = \{w_j\}_m$ 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 prepend and BERT 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
data. For fine-tuning GPT2, we create a training dataset by concatenating all sequences in \( D_{\text{train}} \) as follows: \( y_1 \text{SEP} x_1 \text{EOS} y_2 \ldots y_n \text{SEP} x_n \text{EOS} \). \text{SEP} denotes a separation token between label and sentence, and \text{EOS} denotes the end of a sentence.

For generating data, we provide \( y_i \text{SEP} \) as a prompt to \( G \), and we keep generating until the model produces \text{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 \( y_i \text{SEP} w_1 \ldots w_k \) as prompt where \( w_1 \ldots w_k \) are the first \( k \) words of a sequence \( x_i \). In this work, we use \( k = 3 \). We call this method GPT2-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 \(< \text{mask}>\).
- **BART\text{span}**: Replace a continuous chunk of \( k \) words \( w_i, w_{i+1}, \ldots w_{i+k} \) with a single mask token \(< \text{mask}>\).

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. For BART model implementation, we use fairseq toolkit (Ott et al., 2019) implementation of BART. For generation, beam search with a beam size of 5 is used.

#### 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 \text{SEP} as a separate token and \(< \text{endoftext}>\) as EOS token. For text generation, we use the default nucleus sampling (Holtzman et al., 2019) parameters including \( \text{top}_k = 0 \), and \( \text{top}_p = 0.9 \).

#### 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 \(< \text{mask}>\) 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


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

\[^6\]https://github.com/MiuLab/SlotGated-SLU/tree/master/data/snips
Table 1: Label Names used for fine-tuning pre-trained models. Label names are lower-cased for all experiments.

<table>
<thead>
<tr>
<th>Data</th>
<th>Label Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST-2</td>
<td>Positive, Negative</td>
</tr>
<tr>
<td>TREC</td>
<td>Description, Entity, Abbreviation, Human, Location, Numeric</td>
</tr>
<tr>
<td>SNIPS</td>
<td>PlayMusic, GetWeather, RateBook, SearchScreeningEvent, SearchCreativeWork, AddToPlaylist, BookRestaurant</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th></th>
<th>SST-2</th>
<th>SNIPS</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train</td>
<td>6,228</td>
<td>13,084</td>
<td>5,406</td>
</tr>
<tr>
<td>Dev</td>
<td>692</td>
<td>700</td>
<td>546</td>
</tr>
<tr>
<td>Test</td>
<td>1,821</td>
<td>700</td>
<td>500</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>SST-2</th>
<th>SNIPS</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev</td>
<td>91.91</td>
<td>99</td>
<td>94.13</td>
</tr>
</tbody>
</table>

Table 4: BERT prepend, BERT 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 expand. While CBERT and BERT prepend both converge in less than 8 epochs, BERT 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 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 prepend, BERT expand and CBERT.

Table 4 shows BERT prepend, BERT 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 expand. While CBERT and BERT prepend both converge in less than 8 epochs, BERT 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
<table>
<thead>
<tr>
<th>Model</th>
<th>SST-2</th>
<th>SNIPS</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Aug</td>
<td>52.93 (5.01)</td>
<td>79.38 (3.20)</td>
<td>48.56 (11.53)</td>
</tr>
<tr>
<td>EDA</td>
<td>53.82 (4.44)</td>
<td>85.78 (2.96)</td>
<td>52.57 (10.49)</td>
</tr>
<tr>
<td>BackTrans.</td>
<td>57.45 (5.56)</td>
<td>86.45 (2.40)</td>
<td>66.16 (8.52)</td>
</tr>
<tr>
<td>CBERT</td>
<td>57.36 (6.72)</td>
<td>85.79 (3.46)</td>
<td>64.33 (10.90)</td>
</tr>
<tr>
<td>BERT$_{expand}$</td>
<td>56.34 (6.48)</td>
<td>86.11 (2.70)</td>
<td>65.33 (6.05)</td>
</tr>
<tr>
<td>BERT$_{prepend}$</td>
<td>56.11 (6.33)</td>
<td>86.77 (1.61)</td>
<td>64.74 (9.61)</td>
</tr>
<tr>
<td>GPT2$_{context}$</td>
<td>55.40 (6.71)</td>
<td>86.59 (2.73)</td>
<td>54.29 (10.12)</td>
</tr>
<tr>
<td>BART$_{word}$</td>
<td>57.97 (6.80)</td>
<td>86.78 (2.59)</td>
<td>65.33 (9.84)</td>
</tr>
<tr>
<td>BART$_{span}$</td>
<td>57.68 (7.06)</td>
<td>87.24 (1.39)</td>
<td>67.30 (6.13)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>SST-2</th>
<th>SNIPS</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDA</td>
<td>95.00</td>
<td>97.14</td>
<td>87.22</td>
</tr>
<tr>
<td>BART$_{word}$</td>
<td>96.66</td>
<td>97.14</td>
<td>94.88</td>
</tr>
<tr>
<td>CBERT</td>
<td>96.33</td>
<td>97.90</td>
<td>92.22</td>
</tr>
<tr>
<td>BERT$_{expand}$</td>
<td>95.00</td>
<td>97.04</td>
<td>91.44</td>
</tr>
<tr>
<td>BERT$_{prepend}$</td>
<td>96.66</td>
<td>97.80</td>
<td>94.33</td>
</tr>
<tr>
<td>GPT2$_{context}$</td>
<td>68.33</td>
<td>92.47</td>
<td>60.77</td>
</tr>
<tr>
<td>BART$_{word}$</td>
<td>89.33</td>
<td>91.03</td>
<td>79.33</td>
</tr>
<tr>
<td>BART$_{span}$</td>
<td>90.66</td>
<td>93.04</td>
<td>80.22</td>
</tr>
</tbody>
</table>

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.

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$_{prepend}$ (Liu et al., 2019) outperforms BERT$_{prepend}$ (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$_{context}$ 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-resource 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


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*.


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

A.1 Classification performance on 50 examples per class

<table>
<thead>
<tr>
<th>Model</th>
<th>SST-2</th>
<th>SNIPS</th>
<th>TREC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>No Aug</td>
<td>78.60 (2.81)</td>
<td>90.98 (2.30)</td>
<td>71.65 (11.09)</td>
</tr>
<tr>
<td>EDA</td>
<td>76.41 (4.90)</td>
<td>89.80 (2.99)</td>
<td>61.98 (11.52)</td>
</tr>
<tr>
<td>BackTrans</td>
<td>78.30 (6.30)</td>
<td>89.87 (3.35)</td>
<td>65.52 (11.82)</td>
</tr>
<tr>
<td>CBERT</td>
<td>77.26 (4.56)</td>
<td>90.57 (2.11)</td>
<td>67.77 (13.81)</td>
</tr>
<tr>
<td>BERTexpand</td>
<td>78.15 (4.56)</td>
<td>89.79 (2.56)</td>
<td>68.06 (12.20)</td>
</tr>
<tr>
<td>BERTprepend</td>
<td>77.96 (4.78)</td>
<td>90.41 (2.39)</td>
<td><strong>71.88</strong> (9.91)</td>
</tr>
<tr>
<td>GPT2context</td>
<td>74.91 (5.43)</td>
<td>88.87 (3.33)</td>
<td>51.41 (11.35)</td>
</tr>
<tr>
<td>BARTword</td>
<td>76.35 (5.84)</td>
<td><strong>91.40</strong> (1.60)</td>
<td>71.58 (7.00)</td>
</tr>
<tr>
<td>BARTspan</td>
<td>77.92 (4.96)</td>
<td>90.62 (1.79)</td>
<td>67.28 (12.57)</td>
</tr>
</tbody>
</table>

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.