Diverse Data Augmentation via Unscrambling Text with Missing Words

Anonymous EMNLP submission

Abstract

We present the **D**iverse **Aug**mentation using **S**crambled **S**eq2Seq (DAUGSS) algorithm, a fully automated data augmentation mechanism that employs a model to generate examples in a semi-controllable fashion. The main component of DAUGSS is a training procedure in which the generative model is trained to transform a class label and a list of tokens into a well-formed sentence of the specified class that contains the specified tokens. Empirically, we show that DAUGSS is competitive with or outperforms state-of-the-art, generative models for data augmentation in terms of test set accuracy on 4 datasets. We show that the flexibility of our approach yields augmented datasets with expansive vocabulary, and that models trained on these datasets are more resilient to adversarial attacks than when trained on datasets augmented by competing methods.

1 Introduction

While research in machine learning (ML) has often focused the design of training algorithms and model architectures, recent work is increasingly focused on improving training data quality. As some have argued, state-of-the-art ML models are sufficiently expressive; a claim especially relevant in natural language processing (NLP) where models like GPT-3 and T5 are comprised of billions of parameters (Brown et al., 2020; Raffel et al., 2020). From this vantage point, model failure may be due–in large part–to training set deficiencies.

Training data can be problematic in a number of ways. In many production settings, training datasets may not be sufficiently large. For example, datasets for intent classification—a component in many chatbots—often only have a handful of examples per class (Coucke et al., 2018; Larson et al., 2019). Similarly, training sets may be severely imbalanced and may not reflect the test data distribution. Training data sets can also exhibit stereotypical associations with respect to gender, race, ethnicity and disability status (Bender et al., 2021).

One approach for addressing these training set deficiencies—especially those related to scarce data—is data augmentation. In data augmentation, the goal is to expand the training set by adding new examples. However, not all examples are useful with respect to augmentation. For example, oversampling the training set may not lead to increases in test set accuracy (Lee et al., 2021). Instead, examples added to a training set should differ from the initial examples while also being relevant for the task at hand.

To achieve relevance and novelty simultaneously, previous work in NLP explores augmentation with templates, crowdsourcing, and linguistic transformation (McCoy et al., 2019; Min et al., 2020b; Kaushik et al., 2020). However, these semi-manual approaches are limited since they are costly, especially in comparison to fully-automated alternatives. A handful of automated approaches for data augmentation in NLP have also been proposed. These methods either retrieve new examples from existing corpora, perturb existing training examples or learn to generate new examples (Du et al., 2020; Wu et al., 2018; Kobayashi, 2018; Sun et al., 2020; Lee et al., 2021). Yet, the examples produced by these methods are likely to be similar to the initial training examples, thus reducing their efficacy.

In this work, we present DAUGSS, the **D**ata **Aug**mentation via **S**crambled **S**equence-to-sequence (SEQ2SEQ) algorithm. Our approach is designed to produce examples that contain vocabulary not seen at train time. At a high-level, DAUGSS begins by training a SEQ2SEQ model that takes a class labeled and an arbitrary list of space-delimited tokens as input, and outputs a sentence containing those tokens of the specified class. After training, the model is used to generate new examples using arbitrary, user-selected, vocabulary.

We evaluate DAUGSS by using it to augment 4 classification datasets, and then analyzing models trained on the augmented data. First, we compare the trained models in terms of robustness to 4 adversarial attacks. Our results reveal that for 3 of the 4 attacks, augmentation with DAUGSS leads to more resilient models than augmentation with 2 recently proposed augmentation algorithms. Moreover, we find that augmentation with DAUGSS leads to models with improved test set accuracy when compared to models trained without augmentation. Furthermore, these models achieve competitive test set accuracy in comparison to models trained on datasets augmented by other state-of-the-art augmentation algorithms. Finally, we analyze examples generated by DAUGSS, and find that they are coherent, mostly relevant, and that they introduce the most new vocabulary to the training set among the competing methods.

2 DAUGSS

Our goal is to construct a semi-controllable model that can be used to generate new examples for text classification. We focus on controlling the vocabulary used in the generated examples, as well as their class label. To achieve our goal, we design the DAUGSS training algorithm. We begin by detailing the algorithm, and go on to describe *vocabulary expansion*, one approach to generating new examples that explicitly introduces new vocabulary to a training set.

2.1 Training

The DAUGSS algorithm trains a SEQ2SEQ model to generate text classification examples. DAUGSS requires an *initial* training set. For each initial training example, DAUGSS creates a SEQ2SEQ training example by: i) dropping some tokens from the initial example and permuting the result (yielding a *corrupted* example), ii) concatenating the class label of the initial example and the corrupted example, and iii) mapping the concatenation (of class label and corrupted example) to the initial training example. In this way, the SEQ2SEQ model is trained to "reconstruct" each initial training example from its label and a handful of tokens that appear in that example.

More formally, let $(x, y) \in \mathcal{D}$ be an initial training example of classification dataset, $\mathcal{D}$, such that x is a sentence and y is its ground-truth label. Let $x = [w_1, w_2, \ldots, w_n]$ where w_k is the k^{th} token of x. To train the generative model, DAUGSS begins by constructing a new training set, $\mathcal{D}'$. For each $(x, y) \in \mathcal{D}$, we construct an example $(y + x', x) \in \mathcal{D}'$, where x' is a corrupted version of x and $+$ represents string concatenation. Specifically, for a given $\delta \in [0, 1]$, we construct x' by i) removing all stopwords and non-alphabetic tokens from x, ii) removing $\lfloor \delta n \rfloor$ non-stopword tokens from x uniformly at random, and iii) permuting the result. In practice, we set $\delta = 0.2$. The reason for dropping 20% of the non-stopword tokens is to signal to the SEQ2SEQ model that it should introduce new tokens during generation. In other words, the tokens x' are not the only non-stopword tokens that should comprise the generated example. Examples in $\mathcal{D}'$ for the BANK dataset can be seen in Table 1.

2.2 Generation with Vocabulary Expansion

After training, the SEQ2SEQ model is employed for data augmentation. To generate a new example, a practitioner simply specifies a desired label and a handful of tokens that the example should contain. In this subsection, we describe *vocabulary expansion*, one approach to generation that explicitly introduces new vocabulary to the training set.

At a high level, the goal in vocabulary expansion is to select pivot tokens that *could* have appeared in existing training examples. In detail, for each label y, we begin by constructing $\mathcal{V}^{(y)}$, a map from each non-stopword token to its count in the initial training examples of label y. For each non-stopword token in each initial training example, we also query a pre-trained language model (LM) for its top-k replacements. These replacement tokens are also added to $\mathcal{V}^{(y)}$ (with a count of 1 each time they are returned by the LM). The hope is that the top-k replacements will include vocabulary that does not appear in the initial training examples but is still relevant with respect to the class y. After construction of the token-to-count maps, each new example is generated by: i) selecting a label, y, ii) sampling a list $\hat{x}$ containing s pivot tokens from $\mathcal{V}^{(y)}$, proportional to their counts, and iii) using a model trained by DAUGSS to generate an example on input $y + \hat{x}$. In practice, s is sampled uniformly from the distribution of training example lengths in $\mathcal{D}'$. By using vocabulary expansion, the pivot tokens $\hat{x}$ are likely to contain tokens common to the initial examples of class y, but also new tokens that are produced by the LM.

Token Selection: A natural tension exists in selecting tokens from which to generate new exam-

Input	Target Sequence
accept_reservations | accept reservations	does michael's accept reservations
accept_reservations | applebees	do they take reservations at applebees
accept_reservations | reservations	will qdoba take reservations
accept_reservations | new reservations gramercy tavern accept	does gramercy tavern in new york accept reservations
accept_reservations | accept reservations tavern	does gramercy tavern accept reservations

Table 1: **Example SEQ2SEQ Input/Output Pairs from the CLINC dataset.** The examples all have label *accept_reservations*. The label and pivot words are concatenated, and delimited by a '|' character. The pivots alphabetic, non-stopword tokens subsampled from the target sequence. The target sequence is part of the initial. Tokens highlighted in blue appear in both the input and target sequence; stop words are colored red. Non-stop words that are not sampled to be pivot words are highlighted in purple.

ples. Selecting tokens that are not seen at train time, or that bear little resemblance to a label y can lead to novel training examples to promote generalization. However, such tokens may also cause the DAUGSS-trained model to generate nonsensical, mislabeled, or otherwise strange examples that could have the opposite effect. Regardless of this tension, we note that the generation procedure (via vocabulary expansion or otherwise) affords significant flexibility in the generation of new examples.

3 Experiments

We experiment with DAUGSS for data augmentation, and evaluate the extent which the data it generates facilitates improved generalization. We begin by comparing 3 variations of DAUGSS to 2 recently proposed generative models for data augmentation with respect to test set accuracy. Recognizing that classification accuracy is often a limited measure of generalization (Ribeiro et al., 2020), we also compare the resilience of each augmentation strategy to 4 adversarial attacks. Finally, we present a qualitative analysis of the generations produced by DAUGSS.

3.1 Setup

Before presenting results, we describe our experimental setup, and the methods compared.

Data Augmentation. Given an *initial* dataset, $\mathcal{D}$, we begin by training a BASE classification model, h, and evaluating it on a held-out test set. Next, we train an *example generator* and use it to generate an additional m examples per class. In all experiments, $m = 50$ unless otherwise noted. These *generated* examples are added to the initial training set to produce the *augmented* dataset, $\mathcal{D}'$. Finally, we train and evaluate a new model, h', using the augmented dataset. In all experiments, the BASE model h, and the model h' are implemented as HuggingFace BERT-base uncased models with default hyperparameter settings (Wolf et al., 2020).

Datasets. All experiments are performed using the following 4 classification datasets:

- **SNIPS**: a public benchmark dataset developed by Snips corporation with 7 intent classes, such as MUSIC, MEDIA and WEATHER (Coucke et al., 2018).
- **TREC**: open domain dataset for question classification into 50 fine-grained semantic categories (Li and Roth, 2002).
- **BANK**: fine-grained classification of sentences in the banking domain into 77 classes (Casanueva et al., 2020).
- **CLINC**: classification of utterances into 150 intent classes (Larson et al., 2019). Each class belongs to 1 of 10 domains, such as WORK, CREDIT CARDS, and AUTO & COMMUTE. The dataset includes OUT-OF-SCOPE examples, which we omit, as in previous work (Lee et al., 2021).

Each dataset contains a well-known test set, which we use in evaluating accuracy. When unavailable, we construct a validation set by randomly sampling 10% of the training sentences as in previous work (Wu et al., 2018). The validation sets are used for model section for both the classifier and example generator. We note that each of the datasets is studied in previous work on data augmentation (Anaby-Tavor et al., 2019; Lee et al., 2021). All datasets may be characterized as having short input sentences. Table 3 contains statistics of each dataset.

Method	Input	Output Generation
Ex2	will the gramercy tavern take reservations | is there a restaurant that accepts reservations | is the gramercy tavern accepting reservations	is gramercy tavern accepting reservations
	do the local bars accept reservations | does qi go on reservations | does gramercy's take reservations	does the local bar take reservations
DAugSS	accept_reservations | applebees reservations	will you accept reservations at applebees at 10 am
	accept_reservations | gramercy qdoba reservations	do you accept reservations at qdoba hotel gramercy
DAugSS-6x	accept_reservations | tavern dinner church	does church accept reservations for dinner at a tavern in st lou
	accept_reservations | qdoba Airbnb	can you accept reservations at qdoba in ludovic
Lambada	accept_reservations <SEP>	is grub burger taking reservations"
	accept_reservations <SEP>	does ruth chris in charlotte allow you to make a reservation

Table 2: **Inputs and Generations.** Examples of Inputs and corresponding generations for **Ex2**, **DAugSS**, **DAugSS-6x** and **Lambada**. The tokens colored blue are tokens that were not in the initial training dataset, but were introduced during bootstrapped generation (for Ex2) or vocabulary expansion (for DAugSS). Tokens marked green in the output are those that were absent in the training dataset.

Name	Domain	Classes
Snips	Multi-Domain Intent Classification	7
Trec	Question Answering	50
Bank	Single Domain Intent Classification	77
Clinc	Multi-Domain Intent Classification	150

Table 3: Dataset Statistics.

Since data augmentation is particularly useful in low-resource settings, we follow previous work and construct sub-sampled versions of each training set (Anaby-Tavor et al., 2019; Lee et al., 2021). Each sub-sampled training set has k examples per class, where $k \in \{5, 10, 20\}$.[1]

Example Generators. We compare DAugSS with two recently published methods for data augmentation that also employ generative models to automatically create new examples. We briefly describe these generators below.

- **DAugSS**: the approach advocated in this work. We experiment with vocabulary expansions of sizes 2 and 6 (Section 2.1). In detail, for an expansion of size 2, we use an LM to add 1 token to the vocabulary for every alphabetic, non-stopword token in each training example (effectively doubling the number of non-stopword tokens). Similarly, for expansion of size 6, we add 5 tokens for every alphabetic, non-stopword token. We employ a pretrained, HuggingFace T5-base model as our Seq2Seq model, and the HuggingFace Roberta-base model for vocabulary expansion. Subject to an expansion of size w, we refer to our method as DAugSS-wx. We also experiment with DAugSS with no expansion; maps of token-to-count are still constructed but an LM is not utilized to introduce new tokens.

- **Lambada** (Anaby-Tavor et al., 2019): GPT-2 fine-tuned to generate examples conditioned on a class label and short text prefix. In previous experiments, Lambada outperformed EDA (Wei and Zou, 2019a), CVAE (Pagnoni et al., 2018) and CBERT (Wu et al., 2018) with respect to test set accuracy.

- **Example Extrapolation (Ex2)** (Lee et al., 2021): a Seq2Seq model trained to generate examples of a class y from a concatenation of k randomly selected initial training examples from class y. Since Ex2 is intended for use in situations with imbalanced training data, we modify the original augmentation procedure for our setting, in which all classes have few training examples. Rather than training on classes with many training examples and generating examples for classes with few training examples, we train the Seq2Seq model on examples from all classes. To avoid the negative effects of overfitting at generation time, we employ bootstrapping: we allow model inputs to be drawn from its previously generated examples. As with DAugSS, we implement Ex2 using the HuggingFace T5-base model.

For all methods, we generate examples using nu-

[1] If an original training set has fewer than k examples for a class y, we use all available examples.

cleus sampling[2].

In order to minimize the number of misleading examples added to a dataset, previous work filters all generated examples prior to augmentation (Anaby-Tavor et al., 2019). Specifically, let h be a classifier trained the original dataset. Then, in previous work, for any generated example (x', y), the example is only added to the dataset if $h(x) = y$, i.e., the classifier would correctly classify x'. Such a filter misses learning opportunities from examples misclassified by h—which are arguably of highest value. Therefore, we employ a related filter: a generated example, (x', y), is only used during augmentation if the nearest example to x' among the initial training examples is of class y (and where euclidean distance is measured between the `[CLS]` logits defined by h).

3.2 Test Set Accuracy

We evaluate each augmentation scheme via improvement in test accuracy after augmentation (and retraining). Table 4 contains the accuracy achieved after augmenting each initial training set with 50 new examples per class (and retraining). Each reported accuracy is an average over 5 randomly sampled initial datasets.

The results reveal that on all datasets except for TREC, DAUGSS achieves the highest accuracy or is competitive with the top performer. On TREC, DAUGSS-2x and DAUGSS-6x are competitive with the best competitor. Of the generative models for augmentation, LAMBADA tends to be the weakest. All augmentation algorithms improve upon the BASE accuracy in virtually all settings. However, with larger training sets, augmentation with all methods has a reduced effect on test set accuracy.

3.3 Robustness to Adversarial Attacks

Test set accuracy is typically not a comprehensive mechanism for assessing model generalization (Ribeiro et al., 2020). Test sets are limited, they do not account for out-of-distribution examples, and they may contain artifacts that models can exploit to achieve high accuracy (Gardner et al., 2020). As such, a handful of recent work suggests alternative methods for testing a model's ability to generalize.

Inspired by these concerns, we compare the augmentation methods with respect to robustness to the following adversarial attacks[3].

- **BAE** (Garg and Ramakrishnan, 2020): The input is masked at multiple positions and top-k replacements predicted by a masked LM (i.e., BERT-base-uncased) are used to generate potential adversaries. The resulting sentences are used to probe for non-targeted model failures.
- **CLARE** (Li et al., 2020): We utilize the **CLARE-I** (Insertion) and **CLARE-M** (Merge) variations independently. The former introduces a new `[MASK]` token (i.e., effectively adding a token) which is filled with likely candidate words, the latter masks both tokens in bigrams present in the input and replaces them with a single candidate (i.e., effectively removing a token).
- **BERT-Attack** (Li et al., 2020): It is similar to **BAE**, but generates replacements at a sub-word token level when the word being masked was tokenized into sub-words by the tokenizer.

We run each attack on a model trained on each augmented dataset and report *attack success rate*. This measures the fraction of test examples for which an adversarial attack finds a perturbation that causes the model to fail. We perform the attacks allowing for $k \in [2, 4, 6]$ perturbations per attack. In this experiment, we use the sub-sampled datasets with 10 examples per class. For the sake of reducing computation, we only perform the attacks for 1 initial sample from each dataset (rather than 5, in the accuracy experiment).

Table 5 contains the results. We find that in a majority of cases, a variant of DAUGSS achieves the lowest attack success rate. For the BANK and SNIPS datasets, DAUGSS-2x or DAUGSS-6x is always performs best. For the BAE and BERT-Attacks, DAUGSS-2x and DAUGSS-6x could be expected to be dominant because the vocabulary utilized is, in part, constructed via word substitutions suggested by a Roberta LM (Liu et al., 2019). However, our algorithm's resilience to the more intricate CLARE attack, is not similarly explained. These results highlight the enhancements in robustness imparted by having a more expansive training vocabulary.

3.4 Qualitative Analysis of Generations

Here we study the examples generated by DAUGSS-6x. We choose this variant because it

[2]Decoding with nucleus sampling we set `top_p`$= 0.95$ and `top_k`$= 5$ and only return a single sequence.

[3]All attacks are performed with the text-attack framework (Morris et al., 2020)

Alg	Bank 5	Bank 10	Bank 20	Clinc 5	Clinc 10	Clinc 20	Snips 5	Snips 10	Snips 20	Trec 5	Trec 10	Trec 20
Base	58.6	77.4	86.6	78.4	89.6	93.1	77.6	90.1	93.7	43.0	71.1	85.2
DAugSS	**74.1**	**81.2**	**86.8**	85.5	90.0	93.1	**89.8**	**92.8**	93.7	55.6	73.8	84.3
DAugSS-2x	73.4	80.7	86.2	85.4	89.7	92.7	89.0	91.8	93.9	59.4	75.0	85.3
DAugSS-6x	72.3	79.8	86.1	85.0	89.8	92.8	87.7	91.2	93.6	59.4	76.2	84.6
Ex2	71.1	80.8	86.5	**85.6**	**90.3**	**93.5**	88.5	**92.8**	93.9	**64.6**	**77.0**	84.5
Lambada	68.0	79.5	86.3	83.0	89.4	93.3	80.6	90.2	**94.1**	55.0	72.0	**85.9**

Table 4: **Test Set Accuracy.** On all datasets except for Trec-5, DAugSS either achieves the highest test set accuracy or is competitive with Ex2. On Trec-5, Ex2 achieves the highest test set accuracy. In virtually all cases—and especially in the 5 and 10 size dataset variations—augmentation with DAugSS, DAugSS-2x, Ex2 or Lambada improves upon the Base.

Attack	Alg	Bank-10 2	Bank-10 4	Bank-10 6	Clinc-10 2	Clinc-10 4	Clinc-10 6	Snips-10 2	Snips-10 4	Snips-10 6	Trec-10 2	Trec-10 4	Trec-10 6
Clare-I	Base	31.8	35.6	37.0	15.6	19.7	21.0	31.7	36.8	40.1	27.7	30.4	33.3
	Ex2	23.3	27.4	28.9	11.8	14.9	16.2	21.8	25.8	26.7	**20.9**	**23.1**	24.6
	D	18.9	22.7	24.3	**8.7**	**11.6**	**12.7**	14.4	18.4	18.5	21.5	24.0	**24.3**
	D2x	**17.5**	**20.3**	**22.5**	10.9	14	15.1	**11.4**	**12.2**	**13.6**	21.5	24.4	24.7
	D6x	19.7	22.5	24	12.4	14.6	15.7	14.2	18.6	19.4	21.0	23.4	24.8
Clare-M	Base	16.9	17.1	17.2	9.4	9.7	9.8	16.5	17.9	19.2	6.7	7.0	7.0
	Ex2	16.6	17.0	17.2	**8.6**	**9.0**	**9.0**	12.1	12.7	13.9	6.0	6.2	6.2
	D	16.3	16.4	16.7	9.3	9.6	9.6	11.3	11.6	12.2	6.8	7.1	7.1
	D2x	16.0	16.5	16.7	8.8	9.0	9.1	11.2	**11.4**	**12.0**	6.6	6.8	6.8
	D6x	**15.7**	**16.2**	**16.4**	9.0	9.4	9.5	**10.7**	11.4	12.5	**5.7**	**6.0**	**6.0**
BAE	Base	31.2	32.0	32.2	16.9	17.5	17.7	18.3	21.6	22.0	22.8	22.6	22.6
	Ex2	30.5	31.3	31.7	16.4	17.1	17.2	13.3	14.1	15.0	19.2	19.9	20.1
	D	29.2	29.5	30.1	15.9	16.4	16.5	14.4	15.6	16.5	20.2	20.4	21.0
	D2x	**29.1**	**29.9**	**30.3**	**16.1**	**16.3**	**16.3**	**13.0**	**12.7**	**12.3**	19.4	19.2	19.4
	D6x	29.7	30.0	30.5	16.9	17.4	17.4	14.7	16.0	16.2	**18.5**	**18.3**	**19.1**
BERT-Att	Base	61.5	90.4	96.2	34.4	69.9	83.9	31.8	70.0	84.2	50.8	95.2	98.9
	Ex2	59.8	89.5	95.7	**31.6**	**66.7**	82.6	27.2	64.1	79.9	**43.3**	**87.8**	**94.0**
	D	58.6	90.0	95.9	33.6	69.0	82.3	31.5	67.2	82.4	48.2	91.3	98.4
	D2x	**58.6**	90.0	95.8	33.8	69.0	82.3	**24.7**	**61.7**	**77.5**	47.0	90.0	96.6
	D6x	60.0	**89.2**	**94.9**	33.8	67.7	**81.4**	28.3	64.5	78.2	44.1	88.8	96.2

Table 5: **Attack Success Rates.** Each entry represents the fraction of successful attacks of some attack type (Clare-I (Clare-Insertion), Clare-M (Clare-Merge), BAE and BERT-Att (Bert Attack)) for some dataset. Attacks are allowed either 2, 4, 6 perturbations. On Bank and Snips, DAugSS-2x (D2x) and DAugSS-6x (D6x) always admit the lowest rate of successful attacks. On Clinc and Trec, the majority of the lowest attack success rates are achieved by the DAugSS variants.

Dataset	**Base**	**LAM.**	**Ex2**	**D**	**D2x**	**D6x**
Clinc	1479	1818	2385	2268	3004	4016
Snips	229	621	442	427	566	768
Bank	681	734	913	963	1347	1738
Trec	1227	1716	2163	2464	3009	3636

Table 6: For each dataset of size 10, we calculate the number of unique tokens (barring the stop words and non-alphabets) in the base dataset and some of the augmentation methods. LAM refers to the LAMBADA baseline.

employs the largest vocabulary expansion. Interestingly, along with DAugSS-2x, DAugSS-6x imparts the highest degree of robustness; however, it is not among the top performers in terms of test set accuracy. To better understand these results, we inspect the examples generated by DAugSS-6x (visualized in Table 7). From these examples, we observe that the expanded vocabulary may be both beneficial and detrimental. First, the usage of new vocabulary coupled with T5's pre-training allows for a natural incorporation of new entities into the training set (example 5). Moreover, new vocabulary can also lead to generalization beyond the scope of a class defined by the training examples (example 3). On the other hand, expansion with tokens unsuitable for a class as well as attempting to include unrelated pivot tokens in a generated example can both result in nonsense (examples 2 and 6).

3.5 Vocabulary Size

As in previous work, we report the number of unique alphabetic, non-stopword tokens in the training sets of each dataset (containing 10 examples per class) before and augmentation with various methods (Kumar et al., 2020). These counts appear in Table 6. As expected, DAUGSS with vocabulary expansion yields training sets with the largest number of unique tokens. EX2 and DAUGSS without vocabulary expansion generate datasets with similar numbers of unique tokens. LAMBADA tends to generate the fewest unique tokens, except on the BANK dataset.

4 Related Work

Like DAUGSS, a handful on recent studies focus on training generative models for data augmentation. For example, LAMBADA, to which we compare, fine-tunes a large, pre-trained language model (GPT-2 (Radford et al., 2019)) to generate the initial training examples given their class labels (Anaby-Tavor et al., 2019). The fine-tuned model can then be used to generate new examples, which after filtering, are added to the training set. Example extrapolation (EX2), to which we compare, fine-tunes a large, pre-trained SEQ2SEQ model (T5 (Raffel et al., 2020)), which is then used for augmentation (Lee et al., 2021). In their work, the SEQ2SEQ model takes a sequence of examples of the same class as input and produces a new example of that class. While EX2 was intended for use in imbalanced training datasets, we modify the approach for the few-shot regime via bootstrapping. Less recent work in this space includes the conditional variational auto-encoder (CVAE), which allows for controllable generation (Sohn et al., 2015). We do not compare to CVAE since LAMBADA was shown to be superior for augmentation.

Another family of augmentation algorithms in NLP focuses on creating new examples by perturbing the initial training examples. The perturbed training examples are then added to the training set during augmentation. Classic work in this space is focused on replacing tokens in the initial examples with their synonyms, or nearby tokens in embedding space (Kolomiyets et al., 2011; Zhang et al., 2015; Wang and Yang, 2015). More modern variants use powerful neural networks, like BERT (Devlin et al., 2018), to make contextualized token replacements (Kobayashi, 2018; Wu et al., 2018). Other work in this family employ linguistic perturbations or even random token insertion, deletion and swapping (Min et al., 2020a; Li et al., 2020; McCoy et al., 2019; Wei and Zou, 2019b). While replacement based schemes have proven useful, they are somewhat limited in that, by construction, they are similar to the training data.

Finally, recent studies also explore feature space augmentation for NLP (Sun et al., 2020; Guo et al., 2019; Kumar et al., 2019). These methods circumvent the challenges of generating text by training a model on interpolations between two same-class examples. While feature space augmentation has achieved modest gains on some NLP tasks, it makes convexity assumptions about the latent space which may not hold, and is also difficult to reason about.

5 Strengths and Limitations

Our experiments reveal that DAUGSS is competitive with other state-of-the-art methods in terms of test set accuracy. This is especially the case in the low data regimes (5 utterances per class), where it consistently outperforms LAMBADA. Moreover, DAUGSS and its variants are the most dominant of the augmentation methods in terms of robustness to adversarial attacks.

Investigation of the examples generated by DAUGSS reveals that they are relevant and useful. The novelty of these examples stems from the combination of the knowledge gained by the generative model during pre-training, and the ability to encourage the model to use specific tokens in its output. On the other hand, without constraining the generative model to use the pivot tokens, it may exclude some in order to produce a more coherent example. While vocabulary expansion affords great flexibility, analysis of the examples generated by the DAUGSS-6x model contain both useful and potentially harmful examples.

Limitations. We raise a handful of limitations of our empirical study. First, we only perform experiments with classification datasets. In more complex settings, e.g., question answering, DAUGSS may not reliably construct useful augmented examples. Additionally, our datasets are all in English and contain relatively short sentences.

Input -> Output	Remarks
1) Input: "alarm | midnight set app" Output: **"set an alarm for midnight on the app"**	There token "app" does not appear in the initial training data, but is introduced by vocabulary expansion. T5 uses the new token along with information learned during pre-training to introduce new semantic content. "midnight" is a contextual replacement for "noon"
2) Input: "alarm | fire PM" Output: **"if i fire my alarm it will be 0 pm"**	The word "two" in the phrase "two alarms" from the training data was replaced by the word "fire" due to the candidate generation process, resulting in a nonsensical generation.
3) Input: "accept_reservations | west classes hall reservations Orleans", Output: **"do you accept reservations for west hall classes in Orleans"**	This generated example is valid, but it may cause a model to generalize beyond the test set. While the 5 base training examples for the `accept_reservations` class only deal with restaurants, this generated example is a query about classes. While generations like these improve generalization, they may also detract from test set accuracy.
4) Input: "alarm | fire Send initiate" Output: **"send an alarm send"**	"Send" is introduced as a contextual replacement for "Initiate". Note that the model ignores "fire" in this construction.
5) Input: "restaurant_suggestion | places Thai decent" Output: **"what's the best place in Bangkok to go for thai food"**	An instance of intrinsic knowledge from T5 being used to introduce Bangkok in context with "thai".
6) Input: "restaurant_suggestion | tourist logo" Output: **"can you suggest a restaurant with a logo for the tourist office"**	In an attempt to include the pivot tokens, the model generates a grammatical but unusual example that may decrease test set accuracy.
7) Input: "restaurant_suggestion | best street" Output: **"what's the best street food in san antonio"**	Neither of the pivots "best" and "street" were included in the training examples for this class.
8) Input: "text | fish text said Email worked doctor", Output: **"text doctor says fish have worked since my last text"**	A nonsensical generation written in the same style as other examples with label `text` (i.e., send a text message). Examples of this label are often not grammatical, which may raise issues for T5.
9) Input: "alarm | set" Output: **"set an alarm to wake me up"**	The training examples make no mention of "waking up" in the context of alarms. As such, this is an example of T5 introducing new content via information learned during pre-training and fine-tuning.

Table 7: **Input/Output Examples for DAugSS-6x.** These examples (inputs from CLINC-5) showcase both the strengths and weaknesses of the DAugSS-6x model. Inputs and the corresponding generations appear in the left column, and remarks about the generations appear in the right column. Remarks colored green are presumed to support model generalization; remarks colored red are presumed to hurt model performance.

Our results must also be tempered by the difficulty of evaluating data augmentation methods. Standard practice is to evaluate augmentation algorithms using test set accuracy, but as many have argued, it is inherently limited. Moreover, test set accuracy can be significantly affected by peculiarities of the dataset or hyperparameters, such as the number of examples used in augmentation. This detracts from the ability to use test set accuracy to discover the "best" augmentation methods. With these challenges in mind, we evaluate our methods via robustness to adversarial attacks. Yet, robustness alone is also not sufficient.

Putting aside the difficulty of evaluating data augmentation methods, our experiments reveal that no method dominates universally. But, unlike many other ML algorithms, multiple augmentation approaches can be trivially combined. As such, we believe that employing a mixture of augmentation methods is likely to be a strong approach that merits further investigation.

6 Conclusion

In this work, we introduce DAugSS, a flexible data augmentation algorithm that can generate new examples containing specific pivot tokens. Empirically, we show that DAugSS and its variants are competitive with state-of-the-art data augmentation methods in terms of test set accuracy. Additionally, models trained on data augmented by DAugSS exhibit higher degrees of robustness to adversarial attacks than when trained on data that is augmented by competing methods. Finally, we analyze examples generated by our method, which help to uncover how they can both help and hinder model generalization.

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!

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, pages 610–623, New York, NY, USA. Association for Computing Machinery.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Iñigo Casanueva, Tadas Temcinas, Daniela Gerz, Matthew Henderson, and Ivan Vulic. 2020. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on NLP for ConvAI - ACL 2020*. Data available at https://github.com/PolyAI-LDN/task-specific-datasets.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces.

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

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

Siddhant Garg and Goutham Ramakrishnan. 2020. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181, Online. Association for Computational Linguistics.

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

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. 2020. Learning the difference that makes a difference with Counterfactually-Augmented data. In *International Conference on Learning Representations*.

Sosuke Kobayashi. 2018. 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.

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: Short Papers - Volume 2*, HLT '11, page 271–276, USA. Association for Computational Linguistics.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. In *Proceedings of the 2nd Workshop on Life-long Learning for Spoken Language Systems*, pages 18–26, Suzhou, China. Association for Computational Linguistics.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and William Campbell. 2019. A closer look at feature space data augmentation for Few-Shot intent classification.

Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. 2019. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Kenton Lee, Kelvin Guu, Luheng He, Tim Dozat, and Hyung Won Chung. 2021. Neural data augmentation via example extrapolation.

Chuanrong Li, Lin Shengshuo, Zeyu Liu, Xinyi Wu, Xuhui Zhou, and Shane Steinert-Threlkeld. 2020. Linguistically-informed transformations (LIT): A method for automatically generating contrast sets. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 126–135, Online. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2020. Contextualized Perturbation for Textual Adversarial Attack. *arXiv e-prints*, page arXiv:2009.07502.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, page 1–7, USA. 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.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Junghyun Min, R. Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020a. Syntactic data augmentation increases robustness to inference heuristics. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2339–2352, Online. Association for Computational Linguistics.

Junghyun Min, R Thomas McCoy, Dipanjan Das, Emily Pitler, and Tal Linzen. 2020b. Syntactic data augmentation increases robustness to inference heuristics.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

Artidoro Pagnoni, Kevin Liu, and Shangyan Li. 2018. Conditional Variational Autoencoder for Neural Machine Translation. *arXiv e-prints*, page arXiv:1812.04405.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Kihyuk Sohn, Xinchen Yan, and Honglak Lee. 2015. Learning structured output representation using deep conditional generative models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3483–3491, Cambridge, MA, USA. MIT Press.

Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip Yu, and Lifang He. 2020. Mixup-transformer: Dynamic data augmentation for NLP tasks. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3436–3440, Barcelona, Spain (Online). International Committee on Computational Linguistics.

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. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2557–2563, Lisbon, Portugal. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019a. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019b. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. 2018. Conditional bert contextual augmentation.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.