A little goes a long way: Improving toxic language classification despite data scarcity

Mika Juuti[1], **Tommi Gröndahl**[2], **Adrian Flanagan**[3], **N. Asokan**[1,2]
University of Waterloo[1]
Aalto University[2]
Huawei Technologies Oy (Finland) Co Ltd[3]
mika.juuti@kela.fi, tommi.grondahl@aalto.fi
adrian.flanagan@huawei.com, asokan@acm.org

Abstract

Detection of some types of *toxic language* is hampered by extreme scarcity of labeled training data. *Data augmentation* – generating new synthetic data from a labeled seed dataset – can help. The efficacy of data augmentation on toxic language classification has not been fully explored. We present the first *systematic* study on how data augmentation techniques impact performance across toxic language classifiers, ranging from shallow logistic regression architectures to BERT – a state-of-the-art pre-trained Transformer network. We compare the performance of *eight techniques* on very scarce seed datasets. We show that while BERT performed the best, shallow classifiers performed comparably when trained on data augmented with a combination of three techniques, including GPT-2-generated sentences. We discuss the interplay of performance and computational overhead, which can inform the choice of techniques under different constraints.

1 Introduction

Toxic language is an increasingly urgent challenge in online communities (Mathew et al., 2019). Although there are several datasets, most commonly from Twitter or forum discussions (Badjatiya et al., 2017; Davidson et al., 2017; Waseem and Hovy, 2016; Wulczyn et al., 2017; Zhang et al., 2018), high *class imbalance* is a problem with certain classes of toxic language (Breitfeller et al., 2019). Manual labeling of toxic content is onerous, hazardous (Newton, 2020), and thus expensive.

One strategy for mitigating these problems is *data augmentation* (Wang and Yang, 2015; Ratner et al., 2017; Wei and Zou, 2019): complementing the manually labeled *seed data* with new synthetic documents. The effectiveness of data augmentation for toxic language classification has not yet been thoroughly explored. On relatively small toxic language datasets, *shallow classifiers* have been shown to perform well (Gröndahl et al., 2018). At the same time, pre-trained *Transformer* networks (Vaswani et al., 2017) have led to impressive results in several NLP tasks (Young et al., 2018). Comparing the effects of data augmentation between shallow classifiers and pre-trained Transformers is thus of particular interest.

We systematically compared *eight augmentation techniques* on *four classifiers*, ranging from shallow architectures to BERT (Devlin et al., 2019), a popular pre-trained Transformer network. We used downsampled variants of the Kaggle Toxic Comment Classification Challenge dataset (Jigsaw 2018; §3) as our seed dataset. We focused on the `threat` class, but also replicated our results on another toxic class (§4.6). With some classifiers, we reached the same F1-score as when training on the original dataset, which is 20x larger. However, performance varied markedly between classifiers.

We obtained the highest overall results with BERT, increasing the F1-score up to 21% compared to training on seed data alone. However, augmentation using a fine-tuned GPT-2 (§3.2.4) – a pre-trained Transformer language model (Radford et al., 2019) – reached almost BERT-level performance even with shallow classifiers. Combining multiple augmentation techniques, such as adding majority class sentences to minority class documents (§3.2.3) and replacing *subwords* with embedding-space neighbors (Heinzerling and Strube, 2018) (§3.2.2), improved performance on all classifiers. We discuss the interplay of performance and computational requirements like memory and run-time costs (§4.5). We release our source code.[1]

[1] https://github.com/ssg-research/language-data-augmentation

2 Preliminaries

Data augmentation arises naturally from the problem of filling in missing values (Tanner and Wong, 1987). In classification, data augmentation is applied to available training data. Classifier performance is measured on a separate (non-augmented) test set (Krizhevsky et al., 2012). Data augmentation can decrease *overfitting* (Wong et al., 2016; Shorten and Khoshgoftaar, 2019), and broaden the input feature range by increasing the vocabulary (Fadaee et al., 2019).

Simple oversampling is the most basic augmentation technique: copying minority class datapoints to appear multiple times. This increases the relevance of minority class features for computing the loss during training (Chawla et al., 2002).

EDA is a prior technique combining four text transformations to improve classification with CNN and RNN architectures (Wei and Zou, 2019). It uses (i) synonym replacement from WordNet (§3.2.1), (ii) random insertion of a synonym, (iii) random swap of two words, and (iv) random word deletion.

Word replacement has been applied in several data augmentation studies (Zhang et al., 2015; Wang and Yang, 2015; Xie et al., 2017; Wei and Zou, 2019; Fadaee et al., 2019). We compared four techniques, two based on semantic knowledge bases (§3.2.1) and two on pre-trained (sub)word embeddings (§3.2.2).

Pre-trained Transformer networks feature prominently in state-of-the-art NLP research. They are able to learn contextual embeddings, which depend on neighboring subwords (Devlin et al., 2019). *Fine-tuning* – adapting the weights of a pre-trained Transformer to a specific corpus – has been highly effective in improving classification performance (Devlin et al., 2019) and language modeling (Radford et al., 2019; Walton; Branwen, 2019). State-of-the-art networks are trained on large corpora: GPT-2's corpus contains 8M web pages, while BERT's training corpus contains 3.3B words.

3 Methodology

We now describe the data (3.1), augmentation techniques (3.2), and classifiers (3.3) we used.

3.1 Dataset

We used *Kaggle's toxic comment classification challenge* dataset (Jigsaw, 2018). It contains human-labeled English Wikipedia comments in six different classes of toxic language.[2] The median length of a document is three sentences, but the distribution is heavy-tailed (Table 1).

Mean	Std.	Min	Max	25%	50%	75%
4	6	1	683	2	3	5

Table 1: Document lengths (number of sentences; tokenized with NLTK sent_tokenize (Bird et al., 2009)).

Some classes are severely under-represented: e.g., 478 examples of `threat` vs. 159093 non-`threat` examples. Our experiments concern binary classification, where one class is the *minority class* and all remaining documents belong to the *majority class*. We focus on `threat` as the minority class, as it poses the most challenge for automated analysis in this dataset (van Aken et al., 2018). To confirm our results, we also applied the best-performing techniques on a different type of toxic language, the `identity-hate` class (§4.6).

Our goal is to understand how data augmentation improves performance under extreme data scarcity in the minority class (`threat`). To simulate this, we derive our seed dataset (SEED) from the full data set (GOLD STANDARD) via *stratified bootstrap sampling* (Bickel and Freedman, 1984) to reduce the dataset size k-fold. We replaced newlines, tabs and repeated spaces with single spaces, and lowercased each dataset. We applied data augmentation techniques on SEED with k-fold oversampling of the minority class, and compared each classifier architecture (§3.3) trained on SEED, GOLD STANDARD, and the augmented datasets. We used the original test dataset (TEST) for evaluating performance. We detail the dataset sizes in Table 2.

	GOLD STD.	SEED	TEST
Minority	478	25	211
Majority	159,093	7955	63,767

Table 2: Number of documents (minority: `threat`)

Ethical considerations. We used only public datasets, and did not involve human subjects.

3.2 Data augmentation techniques

We evaluated six data augmentation techniques on four classifiers (Table 3). We describe each aug-

[2]Although one class is specifically called `toxic`, all six represent types of toxic language. See Appendix A.

mentation technique (below) and classifier (§3.3). For comparison, we also evaluated simple oversampling (COPY) and EDA (Wei and Zou, 2019), both reviewed in §2. Following the recommendation of Wei and Zou (2019) for applying EDA to small seed datasets, we used 5% augmentation probability, whereby each word has a $1 - 0.95^4 \approx 19\%$ probability of being transformed by at least one of the four EDA techniques.

Four of the six techniques are based on replacing words with semantically close counterparts; two using semantic knowledge bases (§3.2.1) and two pre-trained embeddings (§3.2.2). We applied 25% of all possible replacements with these techniques, which is close to the recommended substitution rate in EDA. For short documents we ensured that at least one substitution is always selected. We also added majority class material to minority class documents (§3.2.3), and generated text with the GPT-2 language model fine-tuned on SEED (§3.2.4).

3.2.1 Substitutions from a knowledge base

WordNet is a semantic knowledge base containing various properties of *word senses*, which correspond to word meanings (Miller, 1995). We augmented SEED by replacing words with random *synonyms*. While EDA also uses WordNet synonyms (§2), we additionally applied word sense disambiguation (Navigli, 2009) and inflection.

For word sense disambiguation we used *simple Lesk* from PyWSD (Tan, 2014). As a variant of the Lesk algorithm (Lesk, 1986) it relies on overlap in definitions and example sentences (both provided in WordNet), compared between each candidate sense and words in the context.

Word senses appear as uninflected *lemmas*, which we inflected using a dictionary-based technique. We lemmatized and annotated a large corpus with NLTK (Bird et al., 2009), and mapped each <lemma, tag> combination to its most common surface form. The corpus contains 8.5 million short sentences (≤ 20 words) from multiple open-source corpora (see Appendix E). We designed it to have both a large vocabulary for wide coverage (371125 lemmas), and grammatically simple sentences to maximize correct tagging.

Paraphrase Database (PPDB) was collected from bilingual parallel corpora on the premise that English phrases translated identically to another language tend to be paraphrases (Ganitkevitch et al., 2013; Pavlick et al., 2015). We used phrase pairs tagged as *equivalent*, constituting 245691 paraphrases altogether. We controlled substitution by grammatical context as specified in PPDB. In single words this is the part-of-speech tag; whereas in multi-word paraphrases it also contains the syntactic category that appears after the original phrase in the PPDB training corpus. We obtained grammatical information with the *Spacy*[3] parser.

3.2.2 Embedding neighbour substitutions

Embeddings can be used to map units to others with a similar occurrence distribution in a training corpus (Mikolov et al., 2013). We considered two alternative pre-trained embedding models. For each model, we produced top-10 nearest embedding neighbours (cosine similarity) of each word selected for replacement, and randomly picked the new word from these.

Twitter word embeddings (GLOVE) (Pennington et al., 2014) were obtained from a Twitter corpus,[4] and we deployed these via Gensim (Řehůřek and Sojka, 2010).

Subword embeddings (BPEMB) have emerged as a practical pre-processing tool for overcoming the challenge of low-prevalence words (Sennrich et al., 2016). They have been applied in Transformer algorithms, including WordPiece (Wu et al., 2016) for BERT (Devlin et al., 2019), and BPE (Sennrich et al., 2016) for GPT-2 (Radford et al., 2019). BPEMB (Heinzerling and Strube, 2018) provides pre-trained GloVe embeddings, constructed by applying SentencePiece (Kudo and Richardson, 2018) on the English Wikipedia. We use 50-dimensional BPEMB-embeddings with vocabulary size 10,000.

3.2.3 Majority class sentence addition (ADD)

Adding unrelated material to the training data can be beneficial by making relevant features stand out (Wong et al., 2016; Shorten and Khoshgoftaar, 2019). We added a random sentence from a *majority class* document in SEED to a random position in a copy of each minority class training document.

3.2.4 GPT-2 conditional generation

GPT-2 is a Transformer language model pre-trained on a large collection of Web documents. We used the 110M parameter GPT-2 model from the Transformers library (Wolf et al., 2019) We discuss parameters in Appendix F. We augmented as follows (N-fold oversampling):

[3] https://spacy.io/

[4] We use 25-dimensional GloVe-embeddings from: https://nlp.stanford.edu/projects/glove/

Augmentation	Type	Unit	#Parameters	Pre-training Corpus
ADD	Non-toxic corpus	Sentence	NA	NA
PPDB	Knowledge Base	N-gram	NA	NA
WORDNET	Knowledge Base	Word	NA	NA
GLOVE	GloVe	Word	30M	Twitter
BPEMB	GloVe	Subword	0.5M	Wikipedia
GPT-2	Transformer	Subword	117M	WebText
Classifier	**Model Type**	**Unit**	**#Parameters**	**Pre-training Corpus**
Char-LR	Logistic regression	Character	30K	-
Word-LR	Logistic regression	Word	30K	-
CNN	Convolutional network	Word	3M	-
BERT	Transformer	Subword	110M	Wikipedia & BookCorpus

Table 3: Augmentation techniques and classifiers considered in this study.

1. $\hat{G} \leftarrow$ briefly **train** GPT-2 on minority class documents in SEED.
2. **generate** $N-1$ novel documents $\hat{\mathbf{x}} \leftarrow \hat{G}(\mathbf{x})$ for all minority class samples $\mathbf{x}$ in SEED.
3. **assign** the minority class label to all documents $\hat{\mathbf{x}}$
4. **merge** $\hat{\mathbf{x}}$ with SEED.

3.3 Classifiers

Char-LR and Word-LR. We adapted the logistic regression pipeline from the Wiki-detox project (Wulczyn et al., 2017).[5] We allowed n-grams in the range 1–4, and kept the default parameters: TF-IDF normalization, vocabulary size at $10,000$ and parameter $C = 10$ (inverse regularization strength).

CNN. We applied a word-based CNN model with 10 kernels of sizes 3, 4 and 5. Vocabulary size was $10,000$ and embedding dimensionality 300. For training, we used the dropout probability of 0.1, and the Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.001.

BERT. We used the pre-trained Uncased BERT-Base and trained the model with the training script from Fast-Bert.[6] We set maximum sequence length to 128 and mixed precision optimization level to O1.

4 Results

We compared precision and recall for the minority class (`threat`), and the macro-averaged F1-score for each classifier and augmentation technique. (For brevity, we use "F1-score" from now on.) The majority class F1-score remained 1.00 (two digit rounding) across all our experiments. All classifiers are binary, and we assigned predictions to the class with the highest conditional probability. We relax this assumption in §4.4, to report area under the curve (AUC) values (Murphy, 2012).

To validate our results, we performed repeated experiments with the *common random numbers* technique (Glasserman and Yao, 1992), by which we controlled the sampling of SEED, initial random weights of classifiers, and the optimization procedure. We repeated the experiments 30 times, and report confidence intervals.

4.1 Results without augmentation

We first show classifier performance on GOLD STANDARD and SEED in Table 4. van Aken et al. (2018) reported F1-scores for logistic regression and CNN classifiers on GOLD STANDARD. Our results are comparable. We also evaluate BERT, which is noticeably better on GOLD STANDARD, particularly in terms of `threat` recall.

All classifiers had significantly reduced F1-scores on SEED, due to major drops in `threat` recall. In particular, BERT was degenerate, assigning all documents to the majority class in all 30 repetitions. Devlin et al. (2019) report that such behavior may occur on small datasets, but random restarts may help. In our case, random restarts did not impact BERT performance on SEED.

4.2 Augmentations

We applied all eight augmentation techniques (§3.2) to the minority class of SEED (`threat`). Each

[5] https://github.com/ewulczyn/wiki-detox/blob/master/src/modeling/get_prod_models.py

[6] https://github.com/kaushaltrivedi/fast-bert/blob/master/sample_notebooks/new-toxic-multilabel.ipynb

GOLD STANDARD				
	Char-LR	Word-LR	CNN	BERT
Precision	0.61	0.43	0.60	0.54
Recall	0.34	0.36	0.33	0.54
F1	0.72	0.69	0.71	0.77
SEED				
	Char-LR	Word-LR	CNN	BERT
Precision	0.64	0.47	0.41	0.00
Recall	0.03	0.04	0.09	0.00
F1	0.52	0.53	0.57	0.50

Table 4: Classifier performance on GOLD STANDARD and SEED. Precision and recall for `threat`; F1-score macro-averaged from both classes.

technique retains one copy of each SEED document, and adds 19 synthetically generated documents per SEED document. Table 5 summarizes augmented dataset sizes. We present our main results in Table 6. We first discuss classifier-specific observations, and then make general observations on each augmentation technique.

	SEED	Augmented
Minority	25	25→500
Majority	7955	7955

Table 5: Number of documents in augmented datasets. We retained original SEED documents and expanded the dataset with additional synthetic documents (minority: `threat`)

We compared the impact of augmentations on each classifier, and therefore our performance comparisons below are local to each column (i.e., classifier). We identify the best performing technique for the three metrics and report the p-value when its effect is significantly better than the other techniques (based on one-sided paired t-tests, $\alpha = 5\%$).[7]

BERT. COPY and ADD were successful on BERT, raising the F1-score up to 21 percentage points above SEED to 0.71. But their impacts on BERT were different: ADD led to increased recall, while COPY resulted in increased precision. PPDB precision and recall were statistically indistinguishable from COPY, which indicates that it did few alterations. GPT-2 led to significantly better recall ($p < 10^{-5}$ for all pairings), even surpassing GOLD STANDARD. Word substitution methods like EDA, WORDNET, GLOVE, and BPEMB improved on SEED, but were less effective than COPY in both precision and recall. Park et al. (2019) found that BERT may perform poorly on out-of-domain samples. BERT is reportedly unstable on adversarially chosen subword substitutions (Sun et al., 2020). We suggest that non-contextual word embedding schemes may be sub-optimal for BERT since its pre-training is not conducted with similarly noisy documents. We verified that reducing the number of replaced words was indeed beneficial for BERT (Appendix G).

Char-LR. BPEMB and ADD were effective at increasing recall, and reached similar increases in F1-score. GPT-2 raised recall to GOLD STANDARD level ($p < 10^{-5}$ for all pairings), but precision remained 16 percentage points below GOLD STANDARD. It led to the best increase in F1-score: 16 percentage points above SEED ($p < 10^{-3}$ for all pairings).

Word-LR. Embedding-based BPEMB and GLOVE increased recall by at least 13 percentage points, but the conceptually similar PPDB and WORDNET were largely unsuccessful. We suggest this discrepancy may be due to WORDNET and PPDB relying on *written standard* English, whereas toxic language tends to be more colloquial. GPT-2 increased recall and F1-score the most: 15 percentage points above SEED ($p < 10^{-10}$ for all pairings).

CNN. GLOVE and ADD increased recall by at least 10 percentage points. BPEMB led to a large increase in recall, but with a drop in precision, possibly due to its larger capacity to make changes in text – GLOVE can only replace entire words that exist in the pre-training corpus. GPT-2 yielded the largest increases in recall and F1-score ($p < 10^{-4}$ for all pairings).

We now discuss each augmentation technique.

COPY emphasizes the features of original minority documents in SEED, which generally resulted in fairly high precision. On Word-LR, COPY is analogous to increasing the weight of words that appear in minority documents.

EDA behaved similarly to COPY on Char-LR, Word-LR and CNN; but markedly worse on BERT.

ADD reduces the classifier's sensitivity to irrelevant material by adding majority class sentences to minority class documents. On Word-LR, ADD is analogous to reducing the weights of majority class words. ADD led to a marginally better F1-score than any other technique on BERT.

[7] The statistical significance results apply to *this* dataset, but are indicative of the behavior of the techniques in general.

Augmentation	Metric	Char-LR	Word-LR	CNN	BERT
SEED *No Oversampling*	Precision	**0.68** ± 0.22	**0.43** ± 0.27	**0.45** ± 0.14	0.00 ± 0.00
	Recall	0.03 ± 0.02	0.04 ± 0.02	0.08 ± 0.05	0.00 ± 0.00
	F1 (macro)	0.53 ± 0.02	0.54 ± 0.02	0.56 ± 0.03	0.50 ± 0.00
COPY *Simple Oversampling*	Precision	**0.67** ± 0.07	**0.38** ± 0.24	0.40 ± 0.08	**0.49** ± 0.07
	Recall	0.16 ± 0.03	0.03 ± 0.02	0.07 ± 0.03	0.36 ± 0.09
	F1 (macro)	0.63 ± 0.02	0.53 ± 0.02	0.56 ± 0.02	**0.70** ± 0.03
EDA Wei and Zou (2019)	Precision	**0.66** ± 0.06	0.36 ± 0.19	0.26 ± 0.09	0.21 ± 0.03
	Recall	0.13 ± 0.03	0.08 ± 0.04	0.07 ± 0.01	0.06 ± 0.01
	F1 (macro)	0.61 ± 0.02	0.56 ± 0.03	0.55 ± 0.01	0.54 ± 0.01
ADD *Add Majority-class Sentence*	Precision	0.58 ± 0.07	0.36 ± 0.21	**0.45** ± 0.07	0.36 ± 0.04
	Recall	0.24 ± 0.04	0.06 ± 0.04	0.19 ± 0.07	0.52 ± 0.07
	F1 (macro)	0.67 ± 0.03	0.55 ± 0.03	0.63 ± 0.04	**0.71** ± 0.01
PPDB *Phrase Substitutions*	Precision	0.16 ± 0.08	**0.41** ± 0.27	0.37 ± 0.09	**0.48** ± 0.06
	Recall	0.10 ± 0.03	0.04 ± 0.02	0.08 ± 0.04	0.34 ± 0.08
	F1 (macro)	0.56 ± 0.02	0.53 ± 0.02	0.57 ± 0.02	0.70 ± 0.03
WORDNET *Word Substitutions*	Precision	0.16 ± 0.06	0.36 ± 0.24	**0.41** ± 0.08	0.47 ± 0.08
	Recall	0.11 ± 0.03	0.05 ± 0.03	0.11 ± 0.05	0.29 ± 0.07
	F1 (macro)	0.56 ± 0.02	0.54 ± 0.02	0.58 ± 0.03	0.68 ± 0.03
GLOVE *Word Substitutions*	Precision	0.15 ± 0.04	**0.39** ± 0.12	0.38 ± 0.08	0.43 ± 0.11
	Recall	0.14 ± 0.03	0.16 ± 0.05	0.18 ± 0.06	0.18 ± 0.06
	F1 (macro)	0.57 ± 0.02	0.61 ± 0.03	0.62 ± 0.03	0.62 ± 0.03
BPEMB *Subword Substitutions*	Precision	0.56 ± 0.07	0.33 ± 0.07	0.25 ± 0.07	0.38 ± 0.12
	Recall	0.22 ± 0.03	0.22 ± 0.04	0.37 ± 0.08	0.16 ± 0.04
	F1 (macro)	0.66 ± 0.02	0.63 ± 0.02	0.64 ± 0.03	0.61 ± 0.03
GPT-2 *Conditional Generation*	Precision	0.45 ± 0.08	0.35 ± 0.07	0.31 ± 0.08	0.15 ± 0.05
	Recall	**0.33** ± 0.04	**0.42** ± 0.05	**0.46** ± 0.10	**0.62** ± 0.09
	F1 (macro)	**0.69** ± 0.02	**0.69** ± 0.02	**0.68** ± 0.02	0.62 ± 0.03

Table 6: Comparison of augmentation techniques for 20x augmentation on SEED/`threat`: means for precision, recall and macro-averaged F1-score shown with standard deviations (30 paired repetitions). Precision and recall for `threat`; F1-score macro-averaged from both classes. **Bold** figures represent techniques that are either best, or *not* significantly different ($\alpha = 5\%$) from this best technique. **Double underlines** indicate the best technique (for a given metric and classifier) significantly better ($\alpha = 1\%$) than all other techniques.

Word replacement was more effective with GLOVE and BPEMB than with PPDB or WORDNET. PPDB and WORDNET generally replace few words per document, which often resulted in similar performance to COPY. BPEMB was generally the most effective among these techniques.

GPT-2 had the best improvement overall, leading to significant increases in recall across all classifiers, and the highest F1-score on all but BERT. The increase in recall can be attributed to GPT-2's capacity for introducing *novel phrases*. We corroborated this hypothesis by measuring the overlap between the original and augmented test sets and an offensive/profane word list from von Ahn.[8] GPT-2 augmentations increased the intersection cardinality by 260% from the original; compared to only 84% and 70% with the next-best performing augmentation techniques (ADD and BPEMB, respectively). This demonstrates that GPT-2 significantly increased the vocabulary range of the training set, specifically with offensive words likely to be relevant for toxic language classification. However, there is a risk that human annotators might not label GPT-2-generated documents as toxic. Such *label noise* may decrease precision. (See Appendix H, Table 22 for example augmentations that display the behavior of GPT-2 and other techniques.)

[8] `https://www.cs.cmu.edu/~biglou/resources/`

4.3 Mixed augmentations

In §4.2 we saw that the effect of augmentations differ across classifiers. A natural question is whether it is beneficial to combine augmentation techniques. For all classifiers except BERT, the best performing techniques were GPT-2, Add, and BPEmb (Table 6). They also represent each of our augmentation types (§3.2), BPEmb having the highest performance among the four word replacement techniques (§3.2.1–§3.2.2) in these classifiers.

We combined the techniques by merging augmented documents in equal proportions. In ABG, we included documents generated by Add, BPEmb or GPT-2. Since Add and BPEmb impose significantly lower computational and memory requirements than GPT-2, and require no access to a GPU (Appendix C), we also evaluated combining only Add and BPEmb (AB).

ABG outperformed all other techniques (in F1-score) on Char-LR and CNN with statistical significance, while being marginally better on Word-LR. On BERT, ABG achieved a better F1-score and precision than GPT-2 alone ($p < 10^{-10}$), and a better recall ($p < 0.05$). ABG was better than AB in recall on Word-LR and CNN, while the precision was comparable.

Augmenting with ABG resulted in similar performance as Gold Standard on Word-LR, Char-LR and CNN (Table 4). Comparing Tables 6 and 7, it is clear that much of the performance improvement came from the increased vocabulary coverage of GPT-2-generated documents. Our results suggest that in certain types of data like toxic language, consistent labeling may be more important than wide coverage in dataset collection, since automated data augmentation can increase the coverage of language. Furthermore, Char-LR trained with ABG was comparable (no statistically significant difference) to the best results obtained with BERT (trained with Add, $p > 0.2$ on all metrics).

4.4 Average classification performance

The results in Tables 6 and 7 focus on precision, recall and the F1-score of different models and augmentation techniques where the probability threshold for determining the positive or negative class is 0.5. In general the level of precision and recall are adapted based on the use case for the classifier. Another general evaluation of a classifier is based on the ROC-AUC metric, which is the area under the curve for a plot of true-positive rate versus the false-positive rate for a range of thresholds varying over $[0, 1]$. Table 8 shows the ROC-AUC scores for each of the classifiers for the best augmentation techniques from Tables 6 and 7.

BERT with ABG gave the best ROC-AUC value of 0.977 which is significantly higher than BERT with any other augmentation technique ($p < 10^{-6}$). CNN exhibited a similar pattern: ABG resulted in the best ROC-AUC compared to the other augmentation techniques ($p < 10^{-6}$). For Word-LR, ROC-AUC was highest for ABG, but the difference to GPT-2 was not statistically significant ($p > 0.05$). In the case of Char-LR, none of the augmentation techniques improved on Seed ($p < 0.05$). Char-LR produced a more consistent averaged performance across all augmentation methods with ROC-AUC values varying between $(0.958, 0.973)$, compared to variations across all augmentation techniques of $(0.792, 0.962)$ and $(0.816, 0.977)$ for CNN and BERT respectively.

	Char-LR	Word-LR	CNN	BERT
AB				
Precision	0.56	0.37	0.33	0.41
Recall	0.26	0.18	0.36	0.36
F1	0.68	0.62	0.67	0.69
ABG				
Precision	0.48	**0.37**	0.31	0.28
Recall	**0.36**	0.39	**0.52**	**0.65**
F1	**0.70**	**0.69**	**0.69**	0.69

Table 7: Effects of mixed augmentation (20x) on Seed/`threat` (Annotations as in Table 6). Precision and recall for `threat`; F1-score macro-averaged from both classes.

	Char-LR	Word-LR	CNN	BERT
Seed	**0.973**	0.968	0.922	0.816
Copy	0.972	0.937	0.792	0.898
Add	0.958	0.955	0.904	0.956
BPEmb	0.968	0.968	0.940	0.868
GPT-2	0.969	**0.973**	0.953	0.964
ABG	0.972	**0.973**	**0.962**	**0.977**

Table 8: Comparison of ROC-AUC for augmentation (20x) on Seed/`threat` (Annotations as in Table 6).

Our results highlight a difference between the results in Tables 6 and 7: while Copy reached a high F1-score on BERT, our results on ROC-AUC highlight that such performance may not hold while varying the decision threshold. We observe that a combined augmentation method such as ABG provides an increased ability to vary the decision threshold for the more complex classifiers such as CNN and BERT. Simpler models performed consistently across different augmentation techniques.

4.5 Computational requirements

BERT has significant computational requirements (Table 9). Deploying BERT on common EC2 instances requires 13 GB GPU memory. ABG on EC2 requires 4 GB GPU memory for approximately 100s (for 20x augmentation). All other techniques take only a few seconds on ordinary desktop computers (See Appendices C–D for additional data on computational requirements).

	ADD	BPEMB	GPT-2	ABG
CPU	-	100	3,600	3,600
GPU	-	-	3,600	3,600
	Char-LR	Word-LR	CNN	BERT
CPU	100	100	400	13,000
GPU	100	100	400	13,000

Table 9: Memory (MB) required for augmentation techniques and classifiers. Rounded to nearest 100 MB.

4.6 Alternative toxic class

In order to see whether our results described so far generalize beyond `threat`, we repeated our experiments using another toxic language class, `identity-hate`, as the minority class. Our results for `identity-hate` are in line with those for `threat`. All classifiers performed poorly on SEED due to very low recall. Augmentation with simple techniques helped BERT gain more than 20 percentage points for the F1-score. Shallow classifiers approached BERT-like performance with appropriate augmentation. We present further details in Appendix B.

5 Related work

Toxic language classification has been conducted in a number of studies (Schmidt and Wiegand, 2017; Davidson et al., 2017; Wulczyn et al., 2017; Gröndahl et al., 2018; Qian et al., 2019; Breitfeller et al., 2019). NLP applications of data augmentation include text classification (Ratner et al., 2017; Wei and Zou, 2019; Mesbah et al., 2019), user behavior categorization (Wang and Yang, 2015), dependency parsing (Vania et al., 2019), and machine translation (Fadaee et al., 2019; Xia et al., 2019). Related techniques are also used in automatic paraphrasing (Madnani and Dorr, 2010; Li et al., 2018) and writing style transfer (Shen et al., 2017; Shetty et al., 2018; Mahmood et al., 2019).

Hu et al. (2017) produced text with controlled target attributes via variational autoencoders. Mesbah et al. (2019) generated artificial sentences for adverse drug reactions using Reddit and Twitter data. Similarly to their work, we generated novel toxic sentences from a language model. Petroni et al. (2019) compared several pre-trained language models on their ability to understand factual and commonsense reasoning. BERT models consistently outperformed other language models. Petroni et al. suggest that large pre-trained language models may become alternatives to knowledge bases in the future.

6 Discussion and conclusions

Our results highlight the relationship between classification performance and computational overhead. Overall, BERT performed the best with data augmentation. However, it is highly resource-intensive (§4.5). ABG yielded almost BERT-level F1- and ROC-AUC scores on *all* classifiers. While using GPT-2 is more expensive than other augmentation techniques, it has significantly less requirements than BERT. Additionally, augmentation is a *one-time upfront cost* in contrast to ongoing costs for classifiers. Thus, the trade-off between performance and computational resources can influence which technique is optimal in a given setting.

We identify the following further topics that we leave for future work.

SEED coverage. Our results show that data augmentation can increase coverage, leading to better toxic language classifiers when starting with *very small* seed datasets. The effects of data augmentation will likely differ with larger seed datasets.

Languages. Some augmentation techniques are limited in their applicability across languages. GPT-2, WORDNET, PPDB and GLOVE are available for certain other languages, but with less coverage than in English. BPEMB is nominally available in 275 languages, but has not been thoroughly tested on less prominent languages.

Transformers. BERT has inspired work on other pre-trained Transformer classifiers, leading to better classification performance (Liu et al., 2019;

Lewis et al., 2019) and better trade-offs between memory consumption and classification performance (Sanh et al., 2019; Jiao et al., 2019). Exploring the effects of augmentation on these Transformer classifiers is left for future work.

Attacks. Training classifiers with augmented data may influence their vulnerability for model extraction attacks (Tramèr et al., 2016; Krishna et al.), model evasion (Gröndahl et al., 2018), or backdoors (Schuster et al., 2020). We leave such considerations for future work.

Acknowledgments

We thank Jonathan Paul Fernandez Strahl, Mark van Heeswijk, and Kuan Eeik Tan for valuable discussions related to the project, and Karthik Ramesh for his help with early experiments. We also thank Prof. Yaoliang Yu for providing compute resources for early experiments. Tommi Gröndahl was funded by the Helsinki Doctoral Education Network in Information and Communications Technology (HICT).

References

Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Löser. 2018. Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 33–42.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760.

Peter J. Bickel and David A. Freedman. 1984. Asymptotic normality and the bootstrap in stratified sampling. *The annals of statistics*, 12(2):470–482.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing.

Gwern Branwen. 2019. Gpt-2 neural network poetry. https://www.gwern.net/GPT-2 Last accessed May 2020.

Luke Breitfeller, Emily Ahn, David Jurgens, and Yulia Tsvetkov. 2019. Finding microaggressions in the wild: A case for locating elusive phenomena in social media posts. 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 1664–1674.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. In *Proceedings of the 11th Conference on Web and Social Media*, pages 512–515.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 4171–4186.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2019. Data augmentation for low resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 567–573.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764.

Paul Glasserman and David D Yao. 1992. Some guidelines and guarantees for common random numbers. *Management Science*, 38(6):884–908.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All you need is "love": Evading hate speech detection. In *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security (AISec'11)*, pages 2–12.

Benjamin Heinzerling and Michael Strube. 2018. BPEmb: Tokenization-free pre-trained subword embeddings in 275 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, pages 2989–2993.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Jigsaw. 2018. Toxic comment classification challenge identify and classify toxic online comments. Available in https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge, accessed last time in May 2020.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Kalpesh Krishna, Gaurav Singh Tomar, Ankur Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves of sesame street: Model extraction on bert-based apis. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 1097–1105.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine code from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, pages 24–26.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase Generation with Deep Reinforcement Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3865–3878.

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*.

Nitin Madnani and Bonnie Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Journal of Computational Linguistics*, 36(3):341–387.

Asad Mahmood, Faizan Ahmad, Zubair Shafiq, Padmini Srinivasan, and Fareed Zaffar. 2019. A girl has no name: Automated authorship obfuscation using Mutant-X. In *Proceedings on Privacy Enhancing Technologies (PETS)*, pages 54–71.

Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. 2019. Spread of hate speech in online social media. In *Proceedings of the 10th ACM Conference on Web Science (WebSci '19)*, pages 173–182.

Sepideh Mesbah, Jie Yang, Robert-Jan Sips, Manuel Valle Torre, Christoph Lofi, Alessandro Bozzon, and Geert-Jan Houben. 2019. Training data augmentation for detecting adverse drug reactions in user-generated content. 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 2349–2359.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NIPS)*, pages 3111–3119.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41.

Kevin P. Murphy. 2012. *Machine learning: a probabilistic perspective*. MIT press, Cambridge.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41(2):1–69.

Casey Newton. 2020. Facebook will pay $52 million in settlement with moderators who developed PTSD on the job. The Verge. https://www.theverge.com/2020/5/12/21255870/facebook-content-moderator-settlement-scola-ptsd-mental-health/ Last accessed May 2020.

Cheoneum Park, Juae Kim, Hyeon-gu Lee, Reinald Kim Amplayo, Harksoo Kim, Jungyun Seo, and Changki Lee. 2019. ThisIsCompetition at SemEval-2019 Task 9: BERT is unstable for out-of-domain samples. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 1254–1261.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations,word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*, pages 425–430.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473.

Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding, and William Yang Wang. 2019. A benchmark dataset for learning to intervene in online hate speech. 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 4757–4766.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. 2017. Learning to compose domain-specific transformations for data augmentation. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017)*.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, pages 1–10.

Roei Schuster, Tal Schuster, Yoav Meri, and Vitaly Shmatikov. 2020. Humpty dumpty: Controlling word meanings via corpus poisoning. *arXiv preprint arXiv:2001.04935*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proceedings of Neural Information Processing Systems (NIPS)*.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A^{4}NT: Author attribute anonymity by adversarial training of neural machine translation. In *Proceedings of the 27th USENIX Security Symposium*, pages 1633–1650.

Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-BERT: BERT is not robust on misspellings! Generating nature adversarial samples on BERT. *arXiv preprint arXiv:2003.04985*.

Liling Tan. 2014. Pywsd: Python implementations of word sense disambiguation (WSD) technologies [software]. https://github.com/alvations/pywsd.

Martin A Tanner and Wing Hung Wong. 1987. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540.

Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction apis. In *Proceedings of the 25th USENIX Security Symposium*, pages 601–618.

Clara Vania, Yova Kementchedjhieva, Anders Sogaard, and Adam Lopez. 2019. A systematic comparison of methods for low-resource dependency parsing on genuinely low-resource languages. 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 1105–1116.

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 *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pages 5998–6008.

Nick Walton. Ai dungeon 2. https://aidungeon.io/ Last accessed May 2020.

William Yang Wang and Diyi Yang. 2015. Thats 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 (EMNLP)*, pages 2557–2563.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on twitter. In *Proceedings of the NAACL Student Research Workshop*, pages 88–93.

Jason Wei and Kai Zou. 2019. 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.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Sebastien C. Wong, Adam Gatt, Victor Stamatescu, and Mark D. McDonnell. 2016. Understanding data augmentation for classification: When to warp? In *Proceedings of the 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–6.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

Mengzhou Xia, Xiang Kong, Antonios Anastasopoulos, and Graham Neubig. 2019. Generalized data augmentation for low-resource translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5786–5796.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Levy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *Proceedings of the International Conference on Learning Representations (ICLR 2017)*.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. 2018. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75.

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 (NIPS 2015)*.

Ziqi Zhang, David Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 745–760.

A Class overlap and interpretation of "toxicity"

Kaggle's *toxic comment classification challenge* dataset[9] contains six classes, one of which is called toxic. But all six classes represent examples of toxic speech: toxic, severe toxic, obscene, threat, insult, and identity-hate. Of the threat documents in the full training dataset (GOLD STANDARD), 449/478 overlap with toxic. For identity-hate, overlap with toxic is 1302/1405. Therefore, in this paper, we use the term *toxic* more generally, subsuming threat and identity-hate as particular types of toxic speech. To confirm that this was a reasonable choice, we manually examined the 29 threat datapoints not overlapping with toxic. All of these represent genuine threats, and are hence toxic in the general sense.

B The "Identity hate" class

	GOLD STD.	SEED	TEST
Minority	1,405	75	712
Majority	158,166	7,910	63,266

Table 10: Corpus size for identity-hate (minority) and non-identity-hate (majority).

	GOLD STANDARD			
	Char	Word	CNN	BERT
Precision	0.64	0.54	0.70	0.55
Recall	0.40	0.31	0.20	0.62
F1 (macro)	0.74	0.69	0.65	0.79

Table 11: Classifier performance on GOLD STANDARD. Precision and recall for identity-hate; F1-score macro-averaged from both classes.

To see if our results generalize beyond threat, we experimented on the identity-hate class in Kaggle's toxic comment classification dataset. Again, we used a 5% stratified sample of GOLD STANDARD as SEED. We first show the number of samples in GOLD STANDARD, SEED and TEST in Table 10. There are approximately 3 times more minority-class samples in identity-hate than in threat. Next, we show classifier performance on GOLD STANDARD/identity-hate in Table 11. The results closely resemble those on GOLD STANDARD/threat in Table 4 (§4.1).

We compared SEED and COPY with the techniques that had the highest performance on threat: ADD, BPEMB, GPT-2, and their combination ABG. Table 12 shows the results.

Like in threat, BERT performed the poorest on SEED, with the lowest recall (0.06). All techniques decreased precision from SEED, and all increased recall except COPY with CNN. With COPY, the F1-score increased with Char-LR (0.12) and BERT (0.21), but not Word-LR (0.01) or CNN (-0.04). This is in line with corresponding results from threat (§4.2, Table 6): COPY did not help either of the word-based classifiers (Word-LR, CNN) but helped the character- and subword-based classifiers (Char-LR, BERT).

Of the individual augmentation techniques, ADD increased the F1-score the most with Char-LR (0.15) and BERT (0.20); and GPT-2 increased it the most with Word-LR (0.07) and CNN (0.07). Here again we see the similarity between the two word-based classifiers, and the two that take inputs below the word-level. Like in threat, COPY and ADD achieved close F1-scores with BERT, but with different relations between precision and recall. BPEMB was not the best technique with any classifier, but increased F1-score everywhere except in CNN, where precision dropped drastically.

In the combined ABG technique, Word-LR and CNN reached their highest F1-score increases (0.08 and 0.07, respectively). With Char-LR F1-score was also among the highest, but did not reach ADD. Like with threat, ABG increased precision and recall more than GPT-2 alone.

Overall, our results on identity-hate closely resemble those we received in threat, resulting in more than 20 percentage point increases in the F1-score for BERT on augmentations with COPY and ADD. Like in threat, the impact of most augmentations was greater on Char-LR than on Word-LR or CNN. Despite their similar F1-scores in SEED, Char-LR exhibited much higher precision, which decreased but remained generally higher than with other classifiers. Combined with an increase in recall to similar or higher levels than with other classifiers, Char-LR reached BERT-level performance with proper data augmentation.

[9] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

Augmentation	Metric	Char-LR	Word-LR	CNN	BERT
SEED *No Oversampling*	Precision	0.85 ± 0.04	0.59 ± 0.05	0.52 ± 0.08	0.65 ± 0.46
	Recall	0.11 ± 0.04	0.12 ± 0.03	0.11 ± 0.04	0.06 ± 0.10
	F1 (macro)	0.60 ± 0.03	0.60 ± 0.02	0.59 ± 0.02	0.54 ± 0.08
COPY *Simple Oversampling*	Precision	0.61 ± 0.02	0.54 ± 0.04	0.27 ± 0.06	0.52 ± 0.06
	Recall	0.34 ± 0.04	0.14 ± 0.03	0.07 ± 0.01	0.50 ± 0.06
	F1 (macro)	0.72 ± 0.02	0.61 ± 0.02	0.55 ± 0.01	0.75 ± 0.01
ADD *Add Majority-class Sentence*	Precision	0.54 ± 0.04	0.54 ± 0.05	0.43 ± 0.05	0.43 ± 0.05
	Recall	0.47 ± 0.05	0.21 ± 0.03	0.21 ± 0.04	0.58 ± 0.08
	F1 (macro)	0.75 ± 0.01	0.65 ± 0.01	0.64 ± 0.02	0.74 ± 0.01
BPEMB *Subword Substitutions*	Precision	0.43 ± 0.04	0.30 ± 0.03	0.15 ± 0.05	0.29 ± 0.06
	Recall	0.38 ± 0.04	0.29 ± 0.01	0.32 ± 0.05	0.23 ± 0.03
	F1 (macro)	0.70 ± 0.01	0.64 ± 0.01	0.59 ± 0.02	0.62 ± 0.02
GPT-2 *Conditional Generation*	Precision	0.41 ± 0.05	0.30 ± 0.03	0.33 ± 0.08	0.22 ± 0.05
	Recall	0.34 ± 0.04	0.39 ± 0.03	0.34 ± 0.09	0.59 ± 0.06
	F1 (macro)	0.68 ± 0.01	0.67 ± 0.01	0.66 ± 0.01	0.65 ± 0.02
ABG ADD,BPEMB,GPT-2 *Mix*	Precision	0.41 ± 0.04	0.32 ± 0.03	0.28 ± 0.06	0.27 ± 0.05
	Recall	0.50 ± 0.04	0.41 ± 0.02	0.46 ± 0.05	0.62 ± 0.07
	F1 (macro)	0.72 ± 0.01	0.68 ± 0.01	0.66 ± 0.02	0.68 ± 0.02

Table 12: Comparison of augmentation techniques for 20x augmentation on SEED/`identity-hate`: means for precision, recall and macro-averaged F1-score shown with standard deviations (10 repetitions). Precision and recall for `identity-hate`; F1-score macro-averaged from both classes.

C Augmentation computation performance

Table 13 reports computational resources required for replicating augmentations. GPU computations were performed on a GeForce RTX 2080 Ti. CPU computations were performed with an Intel Core i9-9900K CPU @ 3.60GHz with 8 cores, where applicable. Memory usage was collected using *nvidia-smi* and *htop* routines. Usage is rounded to nearest 100 MiB. Computation time includes time to load library from file and is rounded to nearest integer. Computation time (training and prediction) is shown separately for GPT-2.

We provide library versions in Table 14. We used sklearn.metrics.precision_recall_fscore_support[10] for calculating minority-class precision, recall and macro-averaged F1-score. For the first two, we applied *pos_label=1*, and set *average = 'macro'* for the third. For ROC-AUC, we used sklearn.metrics.roc_auc_score[11] with default parameters. For t-tests, we used scipy.stats.ttest_rel[12], which gives p-values for two-tailed significance tests. We divided the p-values in half for the one-tailed significance tests.

	Augmentation			
	Memory (MiB)		Runtime (s)	
	GPU	CPU	GPU	CPU
COPY	-	-	-	< 1
EDA	-	100	-	1
ADD	-	-	-	1
WORDNET	-	4000	-	1
PPDB	-	2900	-	3
GLOVE	-	600	-	32
BPEMB	-	100	-	< 1
GPT-2	3600	3600	12 + 78	-

Table 13: Computational resources (MiB and seconds) required for augmenting 25 examples to 500 examples. GPT-2 takes approximately 6 seconds to train per epoch, and 3 seconds to generate 19 new documents.

[10]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

[11]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

[12]https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html

Library	Version
`https://github.com/jasonwei20/eda_nlp`	*Nov 8, 2019*[13]
`apex`	0.1
`bpemb`	0.3.0
`fast-bert`	1.6.5
`gensim`	3.8.1
`nltk`	3.4.5
`numpy`	1.17.2
`pywsd`	1.2.4
`scikit-learn`	0.21.3
`scipy`	1.4.1
`spacy`	2.2.4
`torch`	1.4.0
`transformers`	2.8.0

Table 14: Library versions required for replicating this study. Date supplied if no version applicable.

	Training			
	Memory (MB)		Runtime (s)	
	GPU	CPU	GPU	CPU
Char-LR	-	100	-	4
Word-LR	-	100	-	3
CNN	400	400	-	13
BERT	3800	1500	757	-
	Prediction			
	Memory (MB)		Runtime (s)	
	GPU	CPU	GPU	CPU
Char-LR	-	100	-	25
Word-LR	-	100	-	5
CNN	400	400	-	42
BERT	4600	4200	464	-

Table 15: Computational resources (MB and seconds) required for training classifiers on the SEED dataset and test dataset. Note that BERT results here were calculated with mixed precision arithmetic (currently supported by Nvidia Turing architecture). We measured memory usage close to 13 GB in the general case.

D Classifier training and testing performance

Table 15 specifies the system resources training and prediction required on our setup (Section C). The SEED dataset has 8,955 documents and test dataset 63,978 documents. We used the 12-layer, 768-hidden, 12-heads, 110M parameter BERT-Base, Uncased-model.[14]

E Lemma inflection in WORDNET

Lemmas appear as uninflected lemmas WordNet. To mitigate this limitation, we used a dictionary-based method for mapping lemmas to surface manifestations with NLTK part-of-speech (POS) tags. For deriving the dictionary, we used 8.5 million short sentences ($\leq$ 20 words) from seven corpora: Stanford NMT,[15] OpenSubtitles 2018,[16] Tatoeba,[17] SNLI,[18] SICK,[19] Aristo-mini (December 2016 release),[20] and WordNet example sentences.[21] The rationale for the corpus was to have a large vocabulary along with relatively simple grammatical structures, to maximize both coverage and the correctness of POS-tagging. We mapped each lemma-POS-pair to its *most common inflected form* in the corpus. When performing synonym replacement in WORDNET augmentation, we lemmatized and POS-tagged the original word with NLTK, chose a random synonym for it, and then inflected the synonym with the original POS-tag if it was present in the inflection dictionary.

F GPT-2 parameters

Table 16 shows the hyperparameters we used for fine-tuning our GPT-2 models, and for generating outputs. Our fine-tuning follows the `transformers` examples with default parameters.[22]

For generation, we trimmed input to be at most 100 characters long, further cutting off the input at the last full word or punctuation to ensure gener-

[14]`https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip`

[15]`https://nlp.stanford.edu/projects/nmt/`

[16]`http://opus.nlpl.eu/OpenSubtitles2018.php`

[17]`https://tatoeba.org`

[18]`https://nlp.stanford.edu/projects/snli/`

[19]`http://clic.cimec.unitn.it/composes/sick.html`

[20]`https://www.kaggle.com/allenai/aristo-mini-corpus`

[21]`http://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html`

[22]`https://github.com/huggingface/transformers/blob/master/examples/language-modeling/run_language_modeling.py`

ated documents start with full words. Our generation script follows `transformers` examples.[23]

Fine-tuning	
Batch size	1
Learning rate	2e-5
Epochs	2
Generation	
Input cutoff	100 characters
Temperature	1.0
Top-p	0.9
Repetition penalty	1
Output cutoff	100 subwords or EOS generated

Table 16: GPT-2 parameters.

In §4.2 – §4.4, we generated novel documents with GPT-2 fine-tuned on `threat` documents in SEED for 2 epochs. In Table 17, we show the impact of changing the number of fine-tuning epochs for GPT-2. Precision generally increased as the number of epochs was increased. However, recall simultaneously decreased.

G Ablation study

In §4.2 – §4.4, we investigated several word replacement techniques with a fixed change rate. In those experiments, we allowed 25% of possible replacements. Here we study each augmentation technique's sensitivity to the replacement rate. As done in previous experiments, we ensured that at least one augmentation is always performed. Experiments are shown in tables 18–21.

Interestingly, all word replacements decreased classification performance with BERT. We suspect this occurred because of the pre-trained weights in BERT.

We show `threat` precision, recall and macro-averaged F1-scores for PPDB in Table 18. Changing the substitution rate had very little impact to the performance on any classifier. This indicates that there were very few n-gram candidates that could be replaced. We show results on WORDNET in Table 19. As exemplified for substitution rate 25% in H, PPDB and WORDNET substitutions replaced very few words. Both results were close to COPY (§4.2, Table 6).

[23] https://github.com/huggingface/transformers/blob/818463ee8eaf3a1cd5ddc2623789cbd7bb517d02/examples/run_generation.py

We show results for GLOVE in Table 20. Word-LR performed better with higher substitution rates (increased recall). Interestingly, Char-LR performance (particularly precision) dropped with GLOVE compared to using COPY. For CNN, smaller substitution rates seem preferable, since precision decreased quickly as the number of substitutions increased.

BPEMB results in Table 21 are consistent across the classifiers Char-LR, Word-LR and CNN. Substitutions in the range 12%–37% increased recall over COPY. However, precision dropped at different points, depending on the classifier. CNN precision dropped earlier than on other classifiers, already at 25% change rate.

H Augmented `threat` examples

We provide examples of augmented documents in Table 22. We picked a one-sentence document as the seed. We remark that augmented documents created by GPT-2 have the highest novelty, but may not always be considered `threat` (see example GPT-2 #1. in Table 22).

Classifier	Metric	Fine-tuning epochs on GPT-2									
		1	2	3	4	5	6	7	8	9	10
Char-LR	Precision	0.38	0.43	0.45	0.49	0.51	0.49	0.52	0.50	**0.51**	**0.51**
	Recall	**0.34**	**0.34**	0.32	0.31	0.31	0.29	0.28	0.28	0.27	0.28
	F1 (macro)	0.68	**0.69**	0.68	0.68	**0.69**	0.68	0.68	0.68	0.68	0.68
Word-LR	Precision	0.30	0.33	0.34	0.34	**0.36**	0.35	0.35	0.34	0.34	0.34
	Recall	**0.47**	0.45	0.43	0.40	0.40	0.38	0.37	0.36	0.35	0.35
	F1 (macro)	0.68	**0.69**	**0.69**	0.68	0.68	0.68	0.67	0.67	0.67	0.67
CNN	Precision	0.26	0.28	0.30	0.32	**0.33**	0.32	0.31	0.31	0.31	0.32
	Recall	0.49	**0.50**	0.47	**0.50**	0.48	0.48	0.48	0.46	0.47	0.46
	F1 (macro)	0.66	0.67	0.68	**0.69**	**0.69**	0.68	0.68	0.68	0.68	0.68
BERT	Precision	0.11	0.14	0.15	0.15	0.16	0.17	0.17	**0.19**	0.17	0.17
	Recall	0.62	0.66	**0.67**	0.64	0.65	0.62	0.62	0.62	0.61	0.61
	F1 (macro)	0.59	0.61	0.62	0.62	0.62	0.63	0.63	**0.64**	0.63	0.62

Table 17: Impact of changing number of fine-tuning epochs on GPT-2-augmented datasets. Mean results for 10 repetitions. Highest numbers highlighted in bold.

Metric	PPDB: N-gram substitution rate					
	0	12	25	37	50	100
Char-LR						
Pre.	**0.14**	**0.14**	0.13	0.13	0.13	**0.14**
Rec.	**0.09**	**0.09**	0.09	0.08	0.07	0.05
F1 ma.	**0.55**	**0.55**	**0.55**	**0.55**	0.54	0.54
Word-LR						
Pre.	0.32	0.33	0.38	**0.44**	0.41	0.34
Rec.	**0.04**	**0.04**	**0.04**	**0.04**	0.03	0.01
F1 ma.	**0.53**	**0.53**	**0.53**	**0.53**	**0.53**	0.51
CNN						
Pre.	**0.44**	0.41	0.39	0.36	0.38	0.32
Rec.	0.09	0.09	**0.10**	0.09	0.08	0.05
F1 ma.	**0.57**	**0.57**	**0.57**	**0.57**	0.56	0.54
BERT						
Pre.	0.45	0.45	0.46	0.46	0.47	**0.48**
Rec.	**0.37**	**0.37**	**0.37**	0.35	0.33	0.25
F1 ma.	**0.70**	**0.70**	**0.70**	**0.70**	0.69	0.66

Table 18: Impact of changing the proportion of substituted words on PPDB-augmented datasets. Mean results for 10 repetitions. Classifier's highest numbers highlighted in bold.

Metric	WORDNET: Word substitution rate					
	0	12	25	37	50	100
Char-LR						
Pre.	**0.15**	**0.15**	0.14	0.14	0.12	0.10
Rec.	**0.10**	**0.10**	**0.10**	**0.10**	0.09	0.07
F1 ma.	**0.56**	**0.56**	**0.56**	0.55	0.55	0.54
Word-LR						
Pre.	0.28	0.29	0.30	0.31	**0.34**	0.31
Rec.	0.04	0.04	0.04	**0.05**	0.04	0.02
F1 ma.	0.53	0.53	0.53	**0.54**	**0.54**	0.52
CNN						
Pre.	0.42	0.43	0.42	**0.45**	0.44	0.32
Rec.	0.10	0.11	0.11	**0.12**	0.10	0.07
F1 ma.	0.58	0.58	0.58	**0.59**	0.58	0.55
BERT						
Pre.	**0.45**	0.44	0.43	0.43	0.42	0.35
Rec.	**0.31**	**0.31**	0.29	0.26	0.24	0.18
F1 ma.	**0.68**	**0.68**	0.67	0.66	0.65	0.61

Table 19: Impact of changing the proportion of substituted words on WORDNET-augmented datasets. Mean results for 10 repetitions. Classifier's highest numbers highlighted in bold.

Metric	**GLOVE: Word substitution rate**					
	0	12	25	37	50	100
Char-LR						
Pre.	0.16	0.15	0.14	0.14	0.14	**0.32**
Rec.	0.11	0.12	**0.13**	**0.13**	**0.13**	0.05
F1 ma.	0.56	0.56	**0.57**	**0.57**	**0.57**	0.54
Word-LR						
Pre.	0.31	**0.37**	0.35	0.33	0.33	0.30
Rec.	0.07	0.10	0.16	**0.19**	**0.19**	0.09
F1 ma.	0.55	0.58	0.61	**0.62**	**0.62**	0.57
CNN						
Pre.	0.41	**0.44**	0.39	0.35	0.28	0.15
Rec.	0.13	0.18	0.19	**0.20**	0.17	0.06
F1 ma.	0.59	**0.62**	**0.62**	**0.62**	0.60	0.54
BERT						
Pre.	**0.44**	0.43	0.40	0.36	0.33	0.13
Rec.	**0.35**	0.27	0.16	0.13	0.11	0.03
F1 ma.	**0.69**	0.66	0.61	0.59	0.58	0.52

Table 20: Impact of changing the proportion of substituted words on GLOVE-augmented datasets. Mean results for 10 repetitions. Classifier's highest numbers highlighted in bold.

Metric	**BPEMB: Subword substitution rate**					
	0	12	25	37	50	100
Char-LR						
Pre.	**0.65**	0.64	0.56	0.52	0.49	0.37
Rec.	0.17	0.20	**0.22**	0.20	0.17	0.06
F1 ma.	0.63	**0.65**	**0.65**	0.64	0.63	0.55
Word-LR						
Pre.	0.26	**0.34**	0.31	0.30	0.25	0.19
Rec.	0.07	0.13	0.22	**0.25**	0.23	0.13
F1 ma.	0.55	0.59	**0.63**	**0.63**	0.62	0.57
CNN						
Pre.	**0.42**	0.37	0.22	0.14	0.09	0.03
Rec.	0.17	0.31	**0.38**	0.31	0.27	0.10
F1 ma.	0.62	**0.66**	0.63	0.59	0.56	0.52
BERT						
Pre.	**0.43**	0.41	0.33	0.32	0.25	0.08
Rec.	**0.37**	0.22	0.15	0.13	0.10	0.03
F1 ma.	**0.70**	0.64	0.60	0.59	0.57	0.52

Table 21: Impact of changing the proportion of substituted subwords on BPEMB-augmented datasets. Mean results for 10 repetitions. Classifier's highest numbers highlighted in bold.

#	Document sample
	SEED: *No Oversampling*
	if you do not stop, the wikapidea nijas will come to your house and kill you
	COPY: *Simple Oversampling*
1.	if you do not stop, the wikapidea nijas will come to your house and kill you
2.	if you do not stop, the wikapidea nijas will come to your house and kill you
3.	if you do not stop, the wikapidea nijas will come to your house and kill you
	EDA: *Easy Data Augmentation*[16]
1.	if you do *put up* not stop the wikapidea nijas will come to your house and kill you
2.	if you do not *stopover* the wikapidea nijas will come to your house and kill you
3.	if you do not *break* the wikapidea nijas will come to your house and kill you
	ADD: *Add Majority-class Sentence*
1.	*We thank you both for your contributions to Wikipedia at-large and your use of the tool.* if you do not stop, the wikapidea nijas will come to your house and kill you
2.	if you do not stop, the wikapidea nijas will come to your house and kill you *Honest!))*
3.	**** username, I am on a shared IP address.* if you do not stop, the wikapidea nijas will come to your house and kill you
	PPDB *Phrase Substitutions*
1.	if you do not *be halted* , the wikapidea nijas will come to your *home* and kill you
2.	if you do not stop , the wikapidea nijas *comes along* to your house and *been murdered* you
3.	if you do not stop , the wikapidea nijas will *arrive* to your *home* and kill you
	WORDNET *Word Substitutions*
1.	if you do not stop , the wikapidea nijas will *amount* to your house and kill you
2.	if you do not stop , the wikapidea nijas will come to your house and *down* you
3.	if you do not stop , the wikapidea nijas will come to your *theatre* and kill you
	GLOVE *Word Substitutions*
1.	if you *so* not stop , the wikapidea nijas will come to your*s* house *have* kill you
2.	*mean know* do not stop *?* the wikapidea nijas will *cos* to your house and kill you
3.	if you do not *thinking* , *and* wikapidea nijas will come to your *going* and kill you
	BPEMB *Subword Substitutions*
1.	*whether* you *know* not stop, the*ldi*kapide*ina* ni*ji* *give* come to your house and kill you
2.	*shouldnever* do not stop, the wika*myla* ni*jag* will come to your house and kill *want*
3.	if you *let* not *train*, the *w*k*ana*pide*i* nijas will come to*never* house and *dead* you
	GPT-2 *Conditional Generation*
1.	*for what you have done, you don't have a choice. You are too big and weak to stop.*
2.	*! So if you don't stop this, you will be murdered and your people will be ripped apart. I hope you die and die and die. No, you will die and die and die. Just stop and kill yourself for it. Advertisements*
3.	*. You will be hanged like a rat. The next day I will get a letter telling you that you were kicked out of your home by the wikapidea nijas and that you will die there. You are an idiot for calling the wikapidea nijas to death and you should die.*
	ABG: *Mixed Augmentation* (ADD, BPEMB & GPT-2)
1.	*Support - The other article is minor, and not worth a separate entry.* if you do not stop, the wikapidea nijas will come to your house and kill you
2.	*. You will be hanged like a rat. The next day I will get a letter telling you that you were kicked out of your home by the wikapidea nijas and that you will die there. You are an idiot for calling the wikapidea nijas to death and you should die.*
3.	if you *let* not *train*, the *w*k*ana*pide*i* nijas will come to*never* house and *dead* you

Table 22: Documents generated by selected augmentation techniques in this study. Changes to original seed *highlighted*. The selected sample is shorter than average (see §3.1, Table 1). We anonymized the username in ADD (#3.). Three samples generated by each technique shown.

[16] https://github.com/jasonwei20/eda_nlp