Semi-supervised Adversarial Text Generation based on Seq2Seq models

Hieu Le *
Boston University, Boston, USA
hle@bu.edu

Dieu-Thu Le
Amazon Inc., Berlin, Germany
deule@amazon.com

Verena Weber
Amazon Inc., Berlin, Germany
wverena@amazon.com

Chris Church
Amazon Inc., Berlin, Germany
cbchurch@amazon.com

Kay Rottmann
Amazon Inc., Berlin, Germany
krrottm@amazon.com

Melanie Bradford
Amazon Inc., Berlin, Germany
neunerm@amazon.com

Peter Chin
Boston University, Boston, USA
spchin@bu.edu

Abstract

To improve deep learning models' robustness, adversarial training has been frequently used in computer vision with satisfying results. However, adversarial perturbation on text have turned out to be more challenging due to the discrete nature of text. The generated adversarial text might not sound natural or does not preserve semantics, which is the key for real world applications where text classification is based on semantic meaning. In this paper, we describe a new way for generating adversarial samples by using pseudo-labeled in-domain text data to train a seq2seq model for adversarial generation and combine it with paraphrase detection. We showcase the benefit of our approach for a real-world Natural Language Understanding (NLU) task, which maps a user's request to an intent. Furthermore, we experiment with gradient-based training for the NLU task and try using token importance scores to guide the adversarial text generation. We show that our approach can generate realistic and relevant adversarial samples compared to other state-of-the-art adversarial training methods. Applying adversarial training using these generated samples helps the NLU model to recover up to 70% of these types of errors and makes the model more robust, especially in the tail distribution in a large scale real world application.

*The work is completed during Hieu Le's internship at Amazon Inc.

1 Introduction

Over the past years, neural machine learning models have become more popular in a wide range of real world natural language applications. While these models have become very accurate, they are susceptible to errors due to small changes in the input, e.g. adding a stop word. This makes it difficult for a natural language understanding system to recognize all utterances correctly since there are many ways to formulate one request. In fact, while common user commands can be understood very well by the system, a system can react differently due to minor input changes, e.g., article variations, paraphrasing, or adding functional words.

Natural Language Understanding (NLU) is at the core of voice assistants and maps a user's request (also referred to as utterance later) to a specific intent within a certain domain, i.e. *PlayMusic* intent within the *Music* domain. In this paper, we identify the weaknesses of an NLU text classification model by finding the types of inputs that have high probability of causing prediction errors and show how to mitigate them. Inspired by recent work in adversarial training and adversarial sample generation (Goodfellow et al., 2015; Morris et al., 2020a), we describe how we employ adversarial text perturbation to identify and fix such samples to ensure a stable behavior of the model and thereby significantly boost the tail accuracy in a large scale real world application through adversarial training.

We show that while common text perturbation methods relying on character manipulation (Ebrahimi et al., 2018; Li et al., 2019), word swap (Alzantot et al., 2018; Ren et al., 2019) could generate adversarial samples on the original text inputs to trick the model, most of them often generate many irrelevant samples that either do not make sense (e.g., grammatically incorrect, unnatural, out of domain) or change the meaning of the original text utterance. Therefore, applying these methods in a real world application without any domain adaptation and additional constraints is not only unsuitable but could even harm the model. In this work, we describe how we use pseudo-labeled in-domain text data for task adaptation and thereby create adversarial perturbations that are more natural and relevant to the NLU task. Our approach uses in-domain data to train a seq2seq model that generates an adversarial version of the input utterance. We introduce multiple constraints including a paraphrase detector model to make sure that the generated adversarial text samples are of high quality. Finally, we show that including these generated adversarial samples during training helps to improve the model's robustness. We compare the results with adversarial training based on the Fast Gradient Sign Method (FGSM) (Shafahi et al., 2019) often used in computer vision.

2 Related work

Adversarial sample generation in the text domain can be categorized by the level of the attemptstowards a target sentence. At the lowest level, hotflip (Ebrahimi et al., 2018) and TextBugger (Li et al., 2019) perform character level perturbation by character manipulations (swap, insert or delete). Other methods by Alzantot et al. (2018), Zang et al. (2020) and Li et al. (2020) create attempts at word level using synonym swaps or masked language model. The most broad type of attempt is sentence level attempt where multiple words within a sentence are changed, such as PAWS (Zhang et al., 2019), SCPN (Iyyer et al., 2018) and SEARs (Ribeiro et al., 2018). While these methods can sometimes produce very high attempt success rates, they are not straightforward to apply out-of-the-box to real world applications, especially when the domains are specific and small changes in the original inputs can lead to a valid change in the labels (i.e., invalid attempts). In this work, we mainly deal with the problem of generating in-domain valid adversarial attempts by using a seq2seq model that learns from live traffic/un-annotated data in real world applications.

3 Seq2Seq-based adversarial text perturbation

Figure 1 describes our semi-supervised seq2seq-based adversarial text perturbation (SSAT) framework which consists of three components: (1) adversarial candidate extraction from unannotated data based on pseudo-labels, (2) a paraphrase detector model that keeps only candidates that are actual paraphrases, and (3) a seq2seq (T5) model that is trained on the data generated in the first two steps to generate adversarial samples for a given input text.

3.1 Pseudo-label based data filtering

In industry applications, it is common that there is significantly more unlabeled data than high quality labeled data. Our goal is to leverage the unannotated data to find pairs of utterances that belong to the same class but are classified differently by the target model. To that end, we propose a multi-step funnel. First, we use the target classification model to pseudo-label the unlabeled data. Then, to obtain sentence vectors, the utterances are encoded using S-BERT (Reimers and Gurevych, 2019). We identify the *k nearest neighbors* in the embedding space for each sentence vector to obtain a dictionary with the *top k* closest utterances in the embedding space for every utterance. Using the list of nearest neighbors for each utterance, we create k pairs and only keep those where the pseudo-label differs.

3.2 Paraphrase detection

We train a paraphrase detection model that determines if two sentences are paraphrases to further filter the dataset created in the previous step. We use a BERT model finetuned on the binary classification task. The training data for this task is extracted from labeled data where we used two different types of data. If a dataset contains only labels (e.g. intent) without specific name entity annotations, we ignore the name entities and define two sentences as paraphrases when they share the same label : $\textsc{Paraphrase}(x, x') : 1$ **if** $y == y'$ **else** 0. If, however, the dataset contains labels (e.g. domain/intent) and slot labels (e.g. Named Entities), we define two sentences as paraphrases if they share the same non-absent named entities and in-

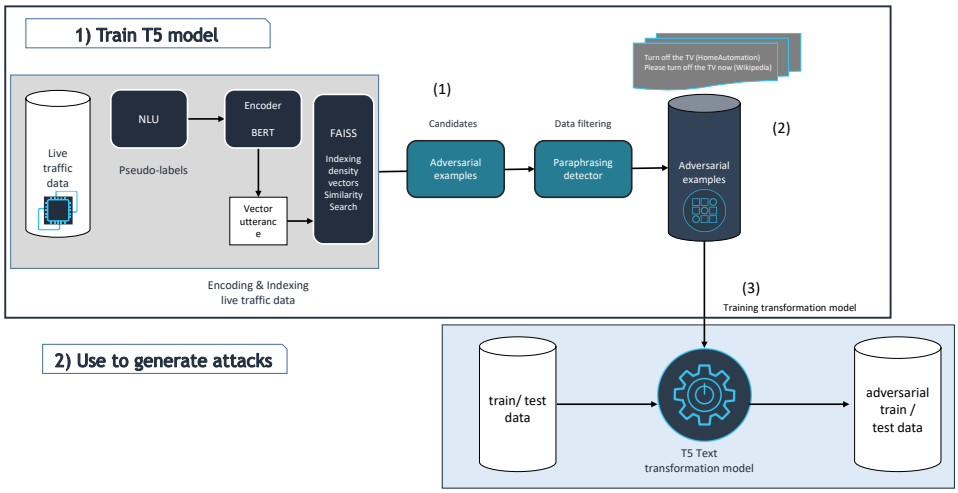

Figure 1: Components of our semi-supervised seq2seq-based adversarial text perturbation framework (SSAT): (1) a text filtering that finds texts that are similar by L_2 distance, (2) paraphrase detector model that detects and keeps only pairs that are paraphrases, (3) text perturbation recipe consisting of a seq2seq model trained on data created from (2).

tent/class. The detailed definition of labeled paraphrases is shown in Algorithm 1.

The pairs of semantically similar utterances but with different pseudo-labels from 3.1 are then fed into the paraphrase detector model. We only keep the the pairs classified as paraphrases and use them as training data for the next step.

Input : $x = ([w_1|NE_1, .., w_n|NE_n], y)$
SLOT(x):
  $s = []$
  for $w_i|NE_i \in x$ **do**
    if NE_i exists **then**
      add $w_i|NE_i$ to s
    end
  end
  return s

PARAPHRASE(x, x'):
  if SLOT(x) == SLOT(x') and $y == y'$ **then**
    return True
  else
    return False
  end

Algorithm 1: Definition of Paraphrases

3.3 Text perturbation recipe

From the text filtering and paraphrase detection steps, we obtain data that satisfy multiple criteria of adversaries: **Semantic similarity**: ensured by knn search in BERT embedding space. **Meaning preservation**: ensured by a paraphrase detector model trained on labeled data with a dataset-specific definition of paraphrases. Intuitively, filtering the data with a paraphrase detector has several advantages: a high level of fluency as the data come from real user traffic and a flexible constraint of semantic similarity that is not enforced through semantic agnostic measurements like edit distance or word embedding similarity.

Our perturbation recipe contains a text-to-text transformer model that creates variations of the input text, serving as candidates for testing the model. We use a pretrained Text-to-Text Transfer Transformer model (T5) (Raffel et al., 2020) fine-tuned on the filtered data for our task. The fine-tuned T5 model is then used to generate multiple adversarial candidates. Specifically, given an input x, we use T5 with a large beam to generate a fixed number of successful perturbation candidates. The heuristic used for beam search is cosine similarity to encourage higher similarity to the input text. From the set of successful candidates $T5(x) = [x_1, \cdots, x_n]$, the candidate with the highest cosine similarity is returned. In terms of perturbation constraints, we reuse the paraphrase detector and thus, any returned perturbation candidate that switches class from the original text and is classified as a paraphrase of the input text is deemed a successful attempt. Figure 2 shows an example of adversarial text generation with T5 and beam search. In this example, from the original SNIPS input of "*add this song to blues roots*" (labeled *AddToPlaylist*), beam search generates multiple successful candidates ['*play music from the playlist late night blues*', '*add this tune to my blues playlist*', $\cdots$]. The candidate '*add this tune to my blues playlist*', wrongly classified as

PlayMusic, is the closest to the original input by cosine similarity and also satisfies the paraphrase constraint. Thus it is returned as a successful attempt.

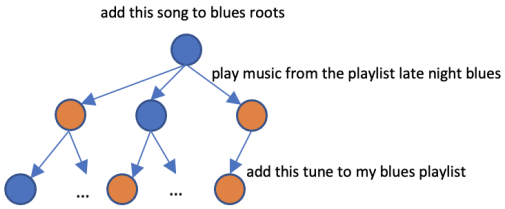

Figure 2: Adversarial text generation process on SNIPS example using beam search on T5 output. Blue nodes are text variants still correctly classified, orange nodes are variants that cause the target model to switch label.

3.4 Adversarial training with text perturbation

Besides finding adversarial perturbations that exploit model weaknesses, we also explored the usage of these generated utterances as additional training data to make the target model more robust against these types of adversarial utterances. We applied the recipe described in 3.3 to our regular training, evaluation and test data. From the generated utterances we added the successful perturbations, those identified as not altering meaning but switching the predicted class, as additional training data into the regular training process. For the test set we kept the original test set and the generated adversarial test set separate to also get insights into the effectiveness of the text perturbations and the effectiveness of the adversarial training.

4 Experiments

4.1 Dataset

We experiment with four text data sets, two commercial data sets from a virtual assistant in German and French, the SNIPS dataset (Coucke et al., 2018) and the MASSIVE data set (FitzGerald et al., 2022). The data consists of unannotated data as well as annotated training and test data. With the commercial data set, unannotated data, i.e. live traffic, is organic audio data coming in from users and processed through an acoustic speech recognition (ASR) system to convert it to text. In this commercial dataset, all data has been preprocessed and anonymized so that no user related information is identifiable. For SNIPS data set, we split data the same way as (Goo et al., 2018) for validation and test set. However, we further split the training data set of $13,084$ utterances into roughly $20-80$ of labeled and 'live traffic' (unannotated) utterances: we remove the label for $11,000$ utterances to simulate live traffic and keep $2,084$ utterances annotated for training. Lastly, for MASSIVE data set, we use the training portion of the data set as the live-traffic, unannotated data set and use the validation and test data set to train the target model. Finally, the live-traffic portion of all the datasets are pseudo-labeled by our corresponding target models.

4.2 Implementation

For each dataset, we fine-tuned a pre-trained BERT architecture for text classification which all achieve above 90% accuracy. Those models serve as target models for the generated perturbations, i. e. we want to fool them through slight changes in the inputs.

For the paraphrase classifier, we chose a multilingual T5 (mT5) (Xue et al., 2021) model as it accommodates the considered datasets in different languages. The training data for the paraphrase classifier is processed from the training set by picking out positive and negative pairs. The positive pairs of paraphrases are picked based on definition 1. For negative examples, as the NLU dataset contains named entity slots, beside randomly selecting pairs of sentences with different classes, we also include cross-class pairs that share at least one slot. The binary classifier model is trained for 5 epochs with batch size of 16 and achieves an accuracy above 90% for both datasets.

We leverage the trained target models and the trained paraphraser to create a training dataset for our T5 model. First, we run the target model on the *live traffic* to acquire pseudo-labels. Using the pseudo-labels, utterances are sorted into buckets for each class. Then, for each utterance, we find the knn closest utterances in other buckets using FAISS (Johnson et al., 2021) similarity search. The detailed algorithm is shown in algorithm 2. By prefiltering the live traffic in this manner, we generate pairs of sentences that are close by L_2 distance yet classified differently by the target model. The candidate pairs are then passed through the trained paraphrase detector and all the positive pairs are kept.

4.3 Baselines

We compare our perturbation recipe with four other text perturbation recipes across different pertur-

		DeepWordBug	BAE	TextBugger	TextFooler	SSAT
German Commercial dataset	Success rate	**92.38%**	66.93%	64.14%	43.44%	61.93%
	Success paraphrase rate	27.59%	14.60%	27.25%	9.88%	**46.69%**
	Perplexity	1704.81	758.85	1239.13	857.00	**498.25**
	Average grammar issues	2.80	2.215	2.88	2.53	**2.21**
French Commercial dataset	Success rate	**94.17%**	78.35%	38.45%	70.54%	86.37%
	Success paraphrase rate	30.48%	28.23%	21.59%	25.18%	**45.12%**
	Perplexity	7087.31	2743.82	2820.99	4323.48	**1712.14**
	Average grammar issues	2.50	1.75	2.38	2.17	**1.65**
SNIPS	Success rate	**57.98%**	51.1%	22.59%	73.94%	**90.42%**
	Success paraphrase rate	49.14%	46.85%	11.14%	**65.71%**	12%
	Perplexity	294.06	82.90	139.42	159.59	**6.3**
	Average grammar issues	3.67	1.84	2.88	2.30	**1.57**
MASSIVE	Success rate	70.07%	67.32%	48.52%	18.09%	**84.73%**
	Success paraphrase rate	69.48%	66.79%	48.20%	18.03%	**84.08%**
	Perplexity	958.18	794.44	980.37	872.93	**677.51**

Table 1: All metrics for generated text perturbations across 4 baselines compared with SSAT: Success rate, success rate under paraphrase constraint, perplexity and average grammatical issues of generated perturbations. The threshold confidence level for paraphrased positive pairs constraint is set at 95%. Perplexity is calculated with *german-gpt2*, *gpt2-french-small*, *gpt2* and *mgpt* accordingly for the corresponding language. Average number of issues per perturbation text found by *language-tool-python*

bation paradigms including character level perturbation - DeepWordBug (Gao et al., 2018) and TextBugger (Li et al., 2019), BERT based word level perturbation - BAE (Garg and Ramakrishnan, 2020) and synonym swaps - TextFooler (Jin et al., 2020) [1]. Besides measuring the number of successful and failed attempts, we also report the number of skipped attempts. An attempt is only counted as successful or failed if the classification of the original input is the same as the label, i.e $f(x) = y$. Thus, when $f(x) \neq y$, an attempt is labeled as *skipped*. The attempt success rate is then measured with:

$$success\_rate = \frac{success\_count}{success\_count + failure\_count} \quad (1)$$

Furthermore, as discussed in 3.2, we also use the paraphrase detector as an extra constraint for all of the successful perturbations. By using the paraphrase detector constraint, the successful perturbation not only fools the classifier but is also ensured to be paraphrase in of the original input taking all domain-specific (device functionalities) into account. Lastly, we calculate the perplexity as well as grammatical and semantic issues for perturbation texts from each method to see which method gives the best semantically sound texts. In terms of perplexity, we use *GPT-2* (Radford et al., 2019) for the SNIPS dataset, *mGPT* (Shliazhko et al., 2022) for MASSIVE dataset, *German GPT-2* (Schweter, 2020) and *French GPT2* for the German and French commercial dataset accordingly. All the GPT2 models are available at Huggingface. For grammatical issues, we calculate this metric by using language-tool-python which is a wrapper of Language Tool, a grammar checker that counts the number of issues for every perturbation text and then gets the average number of issues. The results are shown in table 1.

For the effectiveness of the adversarial training we compare our adversarial data augmentation as described in 5.2 with a baseline model not trained on adversarial data and on a model trained using FGSM based adversarial training as described in (Shafahi et al., 2019), where we used the gradients to modify the input on the embedding layer to overcome the restrictions of the discrete nature of language input.

5 Results and discussion

In this section, we present our results on adversarial sample generation and adversarial training on top of the generated samples for the target models.

5.1 Adversarial sample generation

• **Attempt success rates.** We compare the four common text perturbation methods with our semi-supervised adversarial text perturbation method (SSAT) using two metrics: general success rate and success rate under the paraphrase detector constraint. The result of the experiment is shown in table 1. From the experimental results, SSAT's success rate is comparable with other approaches as approximately 50% of the perturbation successfully fool the target model. However, when all perturbation recipes are subject to the paraphrase constraint, the SSAT method outperforms the rest of the recipes.

• **Attempt sample perplexity and grammar is-**

[1] We take the implementation from TextAttack framework (Morris et al., 2020b) for our baselines

sues. For the perplexity metric, SSAT also consistently outperforms other baselines as shown in table 1, which indicates that the generated samples are more relevant and natural. In terms of fluency/grammatical issues of text samples, SSAT generated samples containing fewer grammatical issues than all 5 methods in comparison. This can be explained by our recipe using training data from the live-unannotated traffic data which is inherently more organic than using a masked language model like (Li et al., 2020) or rule based character swaps like (Li et al., 2019) without any domain adaptation. Indeed, when calculating the pairwise similarity between texts from each perturbation method and the unlabeled corpus dataset using *tf-idf*, SSAT's generated samples are the most similar in cosine similarity to the unlabeled set of the dataset.

perturbation Method	tf-idf cosine similarity
BAE	0.171
DeepWordBug	0.155
TextBugger	0.152
TextFooler	0.057
SSAT	**0.523**

Table 2: pairwise similarity between all successful attempts from each perturbation method and the unlabeled set of MASSIVE dataset

To support our numbers with concrete examples, we picked some successful attempts across baselines and our method on the SNIPS dataset to highlight the difference in perturbation fluency. For example, synonym swaps methods like TextFooler can successfully perturbate a text by swapping in synonyms of some particular words. However, in practice, synonyms have to be taken in the context of the sentence for the swaps to be fluent and natural. While (*rate* - *rhythm*) and (*stars* - *celebs*) are synonyms, the swaps make the sentence unnatural and incomprehensible by a human reader as *rate* in this text is in the verb form with meaning of *evaluating*, not in the noun form as a musical term. On the other hand, by integrating unlabeled, human generated data into the adversarial generation process, SSAT generates a more natural variants like ***give*** *this textbook a* ***three*** given the sentence ***rate*** *this textbook a* ***zero***. While the candidate does slightly change the complete meaning of the sentence, to a human reader, it keeps the overall meaning and the label classification of the text as both should be labeled as **rate book**. Other successful attempts from baselines such as DeepWordBug only made small modifications but the change completely alters the word's meaning and make it hard to comprehend to human readers. All the examples are shown in table 3.

Method	Input text	Perturbed text
BAE	i **rate secret water** as a 4	i **use tap music** as a 4
DeepWordBug	**rate** this current novel 1 **stars**	**arte** this current novel 1 **stZars**
TextBugger	rate maps for lost lovers **1** of 6	rate maps for lost lovers **l** of 6
TextFooler	**rate** lamy of santa fe 5 of 6 **stars**	**rhythm** lamy of santa fe 5 of 6 **celebs**
SSAT	**rate** this textbook a **zero**	**give** this textbook a **three**

Table 3: Some examples of successful perturbations from different baselines and our SSAT method on input texts from SNIPS dataset

Furthermore, we compare the change in the number of paraphrased pairs at different confidence levels of the paraphrase detector output. Figure 3

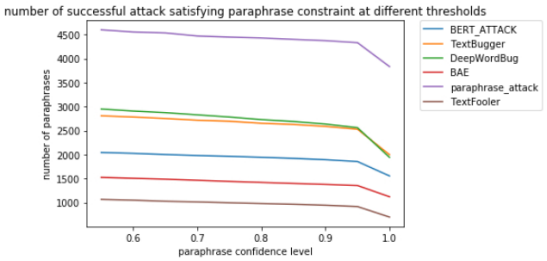

Figure 3: Number of successful attempt at different paraphrase detector threshold on the German commercial dataset. The experiment is run on $10,000$ perturbations.

showed that when gradually increasing the confidence threshold for paraphrase detector to judge that a pair of *input-perturbed text* are paraphrases, the number of successful attempts also gradually decreases. This is expected behavior as when the oracle model is more strict, the number of successful attempts should be decreasing. Another observation from the figure is that regardless of the threshold, UAT consistently performs better than all other perturbation recipes on the German commercial dataset.

• **Gradient-based targeted adversarial generation.** Similar to white-box adversarial perturbation methods such as HotFlip (Ebrahimi et al., 2018), we further experimented using token importance scores based on Integrated Gradients (IG) (Sundararajan et al., 2017) in the decoding step to check whether this could provide guidance on the generation step to find samples with higher attempt success rates. Specifically, we use the token importance score based on IG, which is calculated with respect to the predicted class of the original utterance to re-rank the probability of the next generated token in the beam search of the seq2seq

model. The IG value is positive if the token positively contributed to the predicted class, negative if the token is not associated with the predicted class and 0 if it is neutral. The idea is to generate and favor tokens that are more likely to switch classes to fool the target model. In our experiments, this however leads to poorer results, where we obtained attempt success rate of 51.1%, success paraphrase rate of 22.8%, perplexity of 618.9, and average grammatical issues of 2.22 on the German commercial dataset. This somehow shows that the decoding strategies of the translator model are sensitive to this reranking method, which leads to more invalid attempt generation, especially when running on discrete text input data, where a small local change in embedding gradient cannot account for a word replacement. We plan to experiment with other combinations of SSAT and white-box perturbation methods in the future to further understand this behavior.

5.2 Adversarial training

After generating adversarial instances, we include them to a vanilla adversarial training on the private dataset as a defense mechanism to improve the model's robustness. For the German Commercial dataset, we ran adversarial perturbation on both the train and the test set with the target model being trained on the regular training set. The adversarial examples from training set are then combined with the original training set to train the target model. The so trained model is then tested on the original test set as well as the adversarial perturbations on test set and is compared with the original target model. In addition to the original target model, we also tested the effectiveness of a model trained with FGSM based adversarial training (Shafahi et al., 2019). From results in Table 4, the adversarially trained model only sacrificed a very small loss in standard test set performance but achieve a huge gain in the adversarial test set as the original target model is easily fooled by our method with performance of 15%. In comparison to that, the FGSM based adversarial training achieved slightly better performance than the vanilla model on the adversarial testset, however given the local perturbations used in this approach, this model is still fooled in 75% of the adversarial examples.

Target Model	Class	Std test set	Adv test set
bert-base-german-cased FGSM-adv. trained Commercial dataset	Music	0%	19%
	Books	-1%	-3%
	Calendar	-1%	6%
	Shopping	-1%	15%
	Notification	7%	-1%
	Overall	-1%	10%
bert-base-german-cased **SSAT** adversarially trained Commercial dataset	Music	-0%	74%
	Books	-1.11%	61%
	Calendar	-2.13%	63%
	Shopping	-1.04%	73%
	Notification	7.6%	80%
	Overall	-1.03%	70%
bert-base-multilingual-cased FGSM-adv. trained MASSIVE dataset	**Overall**	0%	0%
bert-base-multilingual-cased **SSAT** adversarially trained MASSIVE dataset	**Overall**	-1.01%	25.23%

Table 4: The relative gain and loss in the accuracy performance of the adversarially trained model compared to the standard model on the standard test set and adversarial test set. The adversarial trained model sacrifices little in accuracy compared to standard model on regular test set and achieves large gain on adversarial tests

6 Conclusions

In this paper, we proposed a framework that uses pseudo-labeled data for learning and training an adversarial perturbation generator that can produce more relevant and natural samples. We trained a paraphraser detector to serve as an additional constraint to validate the generated perturbations. We showed that our perturbation methods outperforms general adversarial perturbation methods on success attempt rates and is able to generate meaningful samples with lower perplexity and less grammatical issues. We further demonstrated how applying adversarial training on the generated samples can better improve the model's robustness than when using traditional gradient adversarial training such as FGSM.

7 Ethical Considerations

Our work proposes a new way of improving classification model performance in natural language understanding tasks. Since our approach is based on the usage of unlabeled data as it is occurring during production, there is a certain risk for the models to overfit on user groups that use the model the most which might introduce a bias for this group. In addition to that there is the need to keep the generation model of the adversarial perturbation generator current, making sure, that data that was removed by customers is also not used in any future application of the model.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang.

2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

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. *CoRR*, abs/1805.10190.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2022. Massive: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56.

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.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. 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 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. *Proceedings 2019 Network and Distributed System Security Symposium*.

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.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. 2020a. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3829–3839, Online. Association for Computational Linguistics.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020b. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey 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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Stefan Schweter. 2020. German gpt-2 model.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32.

Oleh Shliazhko, Alena Fenogenova, Maria Tikhonova, Vladislav Mikhailov, Anastasia Kozlova, and Tatiana Shavrina. 2022. mgpt: Few-shot learners go multilingual.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1298–1308, Minneapolis, Minnesota. Association for Computational Linguistics.

A FAISS search application

To apply FAISS search to our pseudo-labeled data, we first partitioned the pseudo-labeled data by the labels. Then, for each utterance x_i, we run FAISS knn search in one-vs-rest style where we search for the most similar utterances among the those that do not share the same class label as x_i. Each of these similar sentences then got matched with x_i to generate k similar pairs for x_i

Input : $X = [x_i], Y = [f(x_i)]$
SIMILARITYSEARCH(X):
  $S = \{\}$
  for $x_i \in X$ **do**
    $S[x_i] = []$
    for $y \in Y, y \neq f(x_i)$ **do**
      classY$= [x_j]$ s.t $f(x_j) = y$
      $knn =$FAISS$(x_i, classY, k)$
      add knn to $S[x_i]$
    end
  end
  return S

CANDIDATE_PAIRS(S):
  $pairs = []$
  for $x_i \in S$ **do**
    nearestNeighbors $= S[x_i] = [x_j]$
    add $[x_i, x_j] \forall x_j \in S[x_i]$ to $pairs$
  end
  return $pairs$

Algorithm 2: Filtering live traffic with FAISS search

B Limitations

The limitation of our approach lies 1) in the post-filtering approach and 2) in the similarity to actually seen traffic. It is crucial and at the same time very hard to filter out those adversarial samples that are relevant and correct since the T5 model also produces sentences that do not preserve the meaning of the input or are grammatically correct. The second limitation lies in how similar the generated variations actually are to real variations actual customers would say. We are planning to investigate this further and test the target model trained on generated perturbed utterances on general data.