Self-Augmentation Network for Sarcasm Generation with Synthetic Data

Cyrill Knecht Joel Neuner-Jehle Justin Studer David Zehnder

Abstract

Traditional Generative Model training relies on the existence of large datasets. However, for some applications, such as is the case for sarcastic text snippet generation, there are only small publicly available datasets. We propose to use a pre-trained discriminator that can distinguish between sarcasm and non-sarcasm to indirectly provide the generator with feedback on the quality of its outputs during training. Outputs classified as sarcasm will be fed back to the generator for further training. The performance of our proposed model is measured for different fractions of mixing self-augmented data with real data and compared to the baseline performance of classic model fine-tuning with and without state-of-the-art data augmentation.

1. Introduction

With the growing volume of data produced by users of social media websites, text understanding and mimicking is an active and important branch in today's Natural Language Processing (NLP) research. Still, certain data-driven learning problems in Natural Language contexts suffer from having little labeled training data available. This can be improved by using augmentation methods, such as synonym replacement, word swapping or back-translation (Wei & Zou, 2019; Sennrich et al., 2016). However, the use of back-translation in particular is questionable in the context of informal text snippets, due to the ambiguous and often unstructured nature of such data. As an example, generating sarcastic text snippets is difficult due to the lack of large datasets and the importance of contextual information. Furthermore, using existing datasets in the context of sarcasm generation becomes an even bigger challenge considering the fact that currently available datasets contain both positive and negative samples. For Generative Model training, we can only leverage the positive samples, thereby further reducing the usable training data.

In this project, we explored to what extent real data can be replaced with synthetic data produced by a generative model and used for further training. We were also interested in exploring the possibilities of using very little or no real data to train a generative Model using our architecture. If something like that were viable, it would effectively result in a sort of cross-problem transfer learning, as the generator and the classifier have fundamentally different tasks.

The full implementation and our produced results can be found on `https://github.com/njoel-ethz/DL_2022_Sarcasm_Generator`.

2. Models and Methods

2.1. Related Work

Data augmentation is a technique used in machine learning to increase the diversity of the training data by adding modified copies or synthetic data to the dataset. The goal of data augmentation is to act as a regularizer and reduce overfitting when training models. It is commonly used in computer vision, where techniques like cropping, flipping and color jittering are used to augment the data. In NLP, where the input space is discrete, it can be challenging to generate augmented data that captures the desired invariances. (Feng et al., 2021)

It was shown that it is possible to pre-train a Natural Language Generation Model using Synthetic Data (Puri et al., 2020). The used architecture is similar to ours, but without the proposed feedback loop during training. Other than that, there does not exist a lot of research on using Synthetic Data for Natural Language Generation.

However, there are some papers on leveraging Synthetic Data for NLP classification tasks. For example, different pre-trained Language Models were used to generate Synthetic Data for sentiment classification, intent classification and question classification (Kumar et al., 2020).

GAL (He et al., 2021) then proposed a generalized framework for leveraging pre-trained Language Models for self-training and Knowledge Distillation purposes.

SAS (Xu et al., 2021) further implemented a single network architecture to generate Synthetic Data and reuse it in later stages of training for pre-training Language Models.

2.2. Self-Augmentation Model

The basis of our new self-augmentation pipeline is a pre-trained generative Language Model, namely GPT-2 (Radford et al., 2019). We first fine-tune this Model using an initial batch of real data from a sarcastic text snippet dataset. After this pre-training, in every epoch of training a batch of synthetic sarcastic data is generated using the current GPT-2 model. This synthetic data is then classified using a sarcasm classifier. This in-loop sarcasm classifier will from now on be referred to as Classifier. All synthetic data that is labeled as sarcasm is mixed with new real data according to the mixing coefficient to build our dataset for the next epoch. This procedure is then repeated for a predefined number of epochs.

To evaluate our Model, we use a second sarcasm classifier, which from now on will be referred to as Judge. We first generate a batch of sarcastic conversations using our final fine-tuned Model and classify these generated conversations using the Judge.

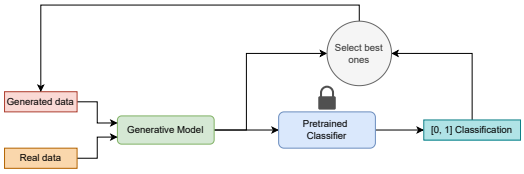

Figure 1. Schematics of our pipeline

This basic outline of our method is showcased in Fig. 1.

2.2.1. Dataset

We chose the Educational Testing Service (ETS) Automatic Sarcasm Detection (Ghosh et al., 2020) twitter dataset to implement our pipeline, since it was the biggest dataset available, providing sarcastic tweets with additional context tweets to build conversations with sarcastic responses. It consists of a training dataset with 5000 samples and a testing dataset with 1800 samples. The structure of one such sample after our preprocessing is shown in Fig. 2 while Fig. 3 shows an overview of the usage of all the different parts of the dataset in our pipeline.

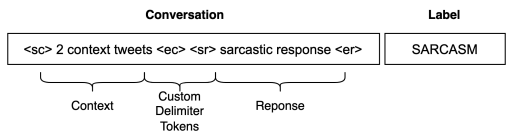

Figure 2. Example Entry in preprocessed Dataset

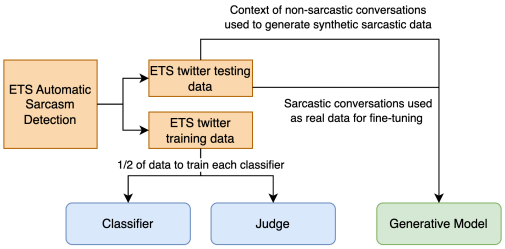

Figure 3. Usage of Dataset in our Pipeline

2.2.2. Classifier and Judge

The core of our architecture are two sarcasm classifiers. To avoid any over-confidence when judging our results, both classifiers were trained on entirely independent data sets. The datasets are results of an equal split of the twitter training data set provided by ETS. The classifiers are based on T5 (Raffel et al., 2019), another pre-trained language Model. This model is then fine-tuned using conversations built from two context tweets and a response that could be sarcastic or non-sarcastic.

To evaluate the classifiers, the ETS twitter testing dataset was used. This leads to the results shown in Table 1.

Table 1. Evaluation results for classifiers

	Accuracy	**Precision**	**F1**
Classifier	0.715	0.718	0.714
Judge	0.72	0.728	0.718

While the results are not state-of-the-art for the ETS sarcasm classification task (Ghosh et al., 2020), they are still acceptable for our purposes. Especially considering that each classifier was trained using only half of the original training data.

2.2.3. Generator

As mentioned before, our Generator is based on GPT-2. While other language models, such as GPT-Neo (Black et al., 2022), would maybe lead to even better performance, we chose to focus on exploring the general feasibility of our self-augmenting pipeline rather than evaluating the performance of different pre-trained language Models.

The Generator gets an initial pre-training using samples from the ETS testing set that are labeled as sarcasm.

After that, the actual training loop starts. In every epoch, synthetic samples are generated using the current Generator. For that, context tweets from samples that are labeled as non-sarcasm are fed to the generator as prompts. The generator then tries to add a sarcastic response, completing the conversation and thereby producing a potential new syn-

thetic sample. Afterwards these immediate responses are classified using the Classifier. If the sample is classified as sarcasm it will be added to the dataset for the next training epoch. The generation and classification procedure is repeated until the desired number of new synthetic sarcastic conversations has been generated.

These synthetic sarcastic samples are then mixed with more real samples from the ETS testing dataset that are labeled as sarcasm. This mixture of real and synthetic data is afterwards used as the new dataset for the next epoch of fine-tuning the Generator.

2.2.4. Evaluation

To evaluate the Generator's performance, we use the final fine-tuned Generator to generate a number of new sarcastic conversations. As prompts for the generation we use context tweets from samples that are labeled as non-sarcasm from the ETS testing dataset that were not already used in the previous fine-tuning procedure.

In the end, these generated samples are classified into sarcasm and non-sarcasm using the Judge, giving us a measure to compare the performance of different Generators.

An overview of this evaluation procedure is shown in Fig. 4.

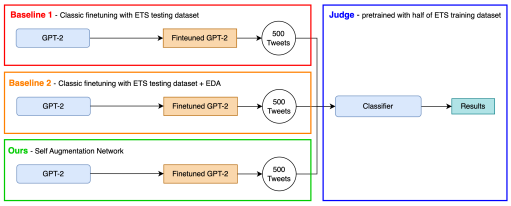

Figure 4. Evaluation Pipeline

2.3. Experiments

To investigate the effectiveness of our method, we conducted ablation studies on the mixing coefficient and compared and combined it with simpler augmentation methods provided by Easy Data Augmentation (Wei & Zou, 2019). The comparison was evaluated with a second, independent classifier.

The baseline for our experiments is given by classic fine-tuning of the GPT-2 language model, purely with real samples from the dataset, and Easy-Data-Augmentation (EDA) augmented training samples for a more accurate comparison with augmented training. EDA makes use of four simple operations: synonym replacement, random insertion, random swap, and random deletion. This can affect the meaning of the produced text outputs, especially in the setting of sarcastic text, but is a powerful and scalable method shown to improve performance in different text classification tasks (Wei & Zou, 2019).

Unless otherwise noted, we used the hyperparameters listed in Table 2 for fine-tuning the generative model.

Table 2. Base Hyperparameters used for the Experiments

Hyperparameter	Value
Number of parameters	124M
Learning rate	1e-4
Epochs	10
Steps per Epoch	100
Batch size	1
Random Seed	42

Experiments with our method (with self-augmented data) have been conducted by varying the mixing coefficients of real data samples and augmented samples. The same is done for EDA augmented inputs, allowing us to compare the performance of the resulting model given the mixing coefficients. We treat the classifier of our pipeline as fully trained and do not conduct respective ablation experiments.

3. Results

The results of our conducted experiments are summarized in Table 3.

Table 3. Results for a dataset of 100 / 500 real samples respectively subject to the percentage of synthetic data records

	Classic Generator	EDA Generator	Our Generator
0%	49.6% / 47.4%	-	-
25%	-	47.4% / 50.6%	49.4% / 49.2%
50%	-	44.6% / 47.2%	58.2% / 49.0%
60%	-	-	43.4% / 72.0%
65%	-	-	62.0% / **77.6%**
70%	-	-	58.2% / 57.2%
75%	-	44.8% / 44.2%	**82.4%** / 64.0%
100%	-	44.3% / 49.4%	74.8% / 50.8%

Example of generated response classified as sarcastic by the Judge from 50% Self-Augmentation Generator trained with 500 real samples:
***Given context**: How we support #AwesomePeople in the #WritingCommunity ? we do so with the #AwesomeLift drop your links to your books and blogs so that we can be become more together*
Hi everyone ! :-) Here is my book After The Fishing Trip : URL
***Generated response**: @USER I'm sure it's time to put on an extra trilogy to describe how this all happened*

Example of response classified as sarcastic which does not make any sense to humans:
***Given context**: @USER Speaking of #fakes... so many gullible people. Seeing what they want to see.*
@USER Explain to us wtf is actually happening here then Sue?
***Generated response**: @USER In this moment in time certainly it's amazing how easy it is for coming up a candidate.*

4. Discussion

4.1. Results

As shown in Table 3, our method performs better when classified by our Judge compared to the Classic Generator or EDA Generator. At first glance, this seems to suggest that we managed to improve over these methods by a huge margin. However, this is not necessarily the case, as the outputs of our methods - upon manual inspection - reveal that the responses actually become less and less varied the higher the self-augmentation contribution is. In its limit, our generator eventually produced the same response no matter what the input context was. This seemingly implies that, rather than learning what sarcasm actually looks like, our Generator eventually just (implicitly) tries to fool the Classifier by approximating its distribution. However, that distribution will be different from what the real data distribution is, especially since our classifiers are far from perfect. It appears to us that this is a very similar problem to what can be observed with GANs, namely mode collapse. We discuss possible ways of addressing this in Section 4.2.

We also manually inspected the outputs of EDA and found that when augmenting more and more samples with it, the outputs eventually became incomprehensible.

Unfortunately, in this configuration, while quantitatively working at first glance, our method does not in fact achieve the desired result of producing better responses. Rather, it starts learning a subset of the distribution of the Classifier, which seems to be similar enough to the one of the Judge to trick it as well. One of the main limitations originates in the way we obtain new data. Since generated data is filtered by the Classifier, its quality is the major restriction of our method. We conjecture that if the Classifier perfectly (or at least more closely) matched the real data distribution, we would be able to obtain much better results. In some sense, as initially imagined when starting this project, we indeed have the case that information about real data is transferred from the Classifier to the Generator. However, if the amount and quality of this data is limited, we can only transfer limited amounts of information to the Generator.

Another question we wanted to address was whether we could train the generator while relying entirely on generated data. However, as we already observe poor results with our pre-training, where the network is first trained on real data for a limited number of steps, we did not further investigate this sub-problem.

4.2. Future work

The most important question we would like to address in future work is whether the quality of the classifier indeed limits the quality of our method. To that end, we would test our method on contexts where much better classifiers are available. Moreover, there are many ways we could improve our method, such as by freezing parts of the generator network or tuning hyperparameters besides the mixing percentage. Another potential way to improve our method is by suppressing repetitive samples. We could do this by keeping a buffer of previously generated samples, filtering out new samples that are already present or at least similar to samples in the buffer.

5. Summary

In this work, we proposed to use a novel self-augmentation pipeline that consists of a Generator and a Classifier, whereby the Classifier assesses whether the Generator's outputs are of good quality, and feeds good outputs back to the Generator for further training. We used our method in the context of sarcasm generation, conducted ablations to find to which extent synthetic samples should be incorporated into the training process, and compared it to a simple augmentation method called Easy Data Augmentation. We found that our method drastically improved the rate at which a separately trained Judge classified the outputs as sarcasm. However, upon manual inspection of the generated samples, we found that our method produces much less varied outputs, and at some point even collapsed to a single response. This issue resembles typical mode collapse, which can often be observed with GANs. While we were not able to improve the subjectively perceived quality of sarcasm generation, we were able to gather evidence that information can be transferred from the Classifier to the generator. This would make it possible for negative samples to be involved into the training process. However, due to the limited quality of the classifiers we used, further testing in a more optimal setting is required.

References

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. Gpt-neox-20b: An open-source autoregressive language model, 2022. URL https://arxiv.org/abs/2204.06745.

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., and Hovy, E. A survey of data augmentation approaches for nlp, 2021. URL https://arxiv.org/abs/2105.03075.

Ghosh, D., Vajpayee, A., and Muresan, S. A report on the 2020 sarcasm detection shared task. *CoRR*, abs/2005.05814, 2020. URL https://arxiv.org/abs/2005.05814.

He, X., Nassar, I., Kiros, J., Haffari, G., and Norouzi, M. Generate, annotate, and learn: Generative models advance self-training and knowledge distillation. *CoRR*, abs/2106.06168, 2021. URL https://arxiv.org/abs/2106.06168.

Kumar, V., Choudhary, A., and Cho, E. Data augmentation using pre-trained transformer models. *CoRR*, abs/2003.02245, 2020. URL https://arxiv.org/abs/2003.02245.

Puri, R., Spring, R., Patwary, M., Shoeybi, M., and Catanzaro, B. Training question answering models from synthetic data. *CoRR*, abs/2002.09599, 2020. URL https://arxiv.org/abs/2002.09599.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019. URL http://arxiv.org/abs/1910.10683.

Sennrich, R., Haddow, B., and Birch, A. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 86–96, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1009. URL https://aclanthology.org/P16-1009.

Wei, J. and Zou, K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.

Xu, Y., Zhang, J., He, R., Ge, L., Yang, C., Yang, C., and Wu, Y. N. SAS: self-augmented strategy for language model pre-training. *CoRR*, abs/2106.07176, 2021. URL https://arxiv.org/abs/2106.07176.