In-Style Prefix-Tuned NLG of Emails

Stanford CS224N Custom Project

AJ Nadel
Department of Computer Science
Stanford University
jobynad@stanford.edu

Uma Phatak
Department of Computer Science
Stanford University
uphatak@stanford.edu

Abstract

Fine-tuned large language models are the current state-of-the-art for many Natural Language Generation (NLG) tasks, though further training a large language model necessitates significant time and compute, which may be prohibitive in some contexts. In this paper, we propose the application of Li and Liang's lighter-weight method of prefix-tuning, which steers language models towards betters task-specific outputs by optimizing a small trainable module on top, the state of which is prefixed to the every LM input. We focus on the particular NLG task of in-style text generation, and explore two specific applications: 1) generating emails with a particular authorial style, and 2) generating polite text. For these tasks, we compile a novel corpus of around 300 emails composed by a single author, and use a 1,000-example subset of polite text from a publicly-available email dataset. We evaluate the performance of both a baseline and fine-tuned GPT-2 model against the performance of a prefix-tuned GPT-2 model on our 2 sub-tasks. We find that for both datasets, fine-tuning performs qualitatively better than the baseline more than 50% percent of the time, while our prefix-tuning implementation performs at the level of the baseline.

1 Key Information to include

- Mentor: Anna Goldie
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Natural language generation (NLG) is a subspace of NLP that has benefited greatly from recent progress made in language models (LMs). LMs trained over large corpora can be leveraged to specific, narrow tasks by fine-tuning a pretrained model checkpoint over much smaller corpus of task-specific examples, which saves on training time and resources while still yielding good results. One problem for these large language models, however, is that even fine-tuning is quite computationally expensive: GPT-2 has 774M parameters and GPT-3 has 175B. Approaches to this challenge range from adding task-specific layers amidst the pretrained ones, as well as for larger and more capable models like GPT-3, an extreme zero/few-shot strategy in which tasks are specified at inference time using natural language instructions and only a few examples. This "in-context learning" or "prompting" strategy is closely related to Li and Liang's "prefix-tuning" approach [1]. With prefix-tuning, the authors hope to provide a new more lightweight and flexible option for steering language models towards task-specific performance. It also promises to aid in tasks which need benefit from personalization—that is, large batches of tasks that need to be trained privately and independently. We apply the more specialized tool of prefix-tuning to a far less constrained task than it has been applied to before—in-style text generation—and compare prefix-tuning's performance against that of a baseline GPT-2 model and a fine-tuned GPT-2 model. We evaluate whether the benefits prefix-tuning can provide on smaller datasets, like not over-fitting to the data in the way that many traditional deep learning techniques can, manifest in this less constrained task space.

3 Related Work

3.1 Other Prefix-Tuning Applications

In their novel research, Li and Liang propose prefix-tuning as an alternative method to fine-tuning for the specific tasks of table-to-text generation and summarization. In both of these domains, prefix-tuning provides comparable performance to fine-tuning while requiring less time and compute power. Both of these tasks are relatively constrained, as the transition from input to output is quite well defined for the model. It remains to be seen whether predfix-tuning can achieve similarly comparable performance on a more open-ended task like the one proposed in this paper.

3.2 Text Style Transfer

Another field related to in-style text generation that has been evolving recently is text style transfer. In text style transfer, one aims to transform a given piece of text into a particular style (e.g. formal to informal, humorous to romantic, polite to rude, etc.) This task, is similar to in-style text generation in that it also attempts to train a model to differentiate between semantic characteristics and stylistic characteristics, trying to preserve both content and style even thought the two qualities textually manifest in different ways. However, style transfer requires parallel corpora for the best results. For this reason, existing methods for text style transfer also are not optimal for this paper's novel subtasks, on of which is emulating general authorial style. More concretely, as one of the composed datasets (`chveers_300`) does not have a readily available parallel corpus, and in fact, there exists no intuitive way to make a parallel, "unstyled" corpus for this author, text style transfer techniques do not satisfy the contraints.

4 Approach

4.1 Baseline GPT-2

Following the lead of Li and Liang in the prefix-training paper, we recognize GPT-2 as the current publicly-available basis for many state-of-the-art models for NLP tasks.[1]

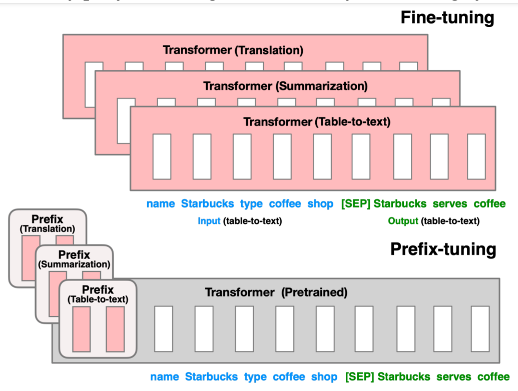
4.2 Fine-Tuned GPT-2

Our intermediate method for in-style text generation is fine-tuning GPT-2, the current standard method for generating text. With fine-tuning, GPT-2 was able to achieve a BLEU score of 45.7 on the WebNLG table-to-text corpus (between both seen and unseen subtasks). We have chosen it as our baseline for free-form NLG because of its demonstrated task adaptability and availability via HuggingFace.

4.3 Prefix-Tuned GPT-2

Our prefix-tuning model aims to adapt the prefix-tuning procedure from Li and Liang to the task of open-ended language generation. Li and Liang applied their technique—in which the transformer input is prefixed with a fixed-length sequence of parameterized continuous word vectors which are then tuned during training—to two NLG tasks with publicly-available datasets: abstractive summarization and table-to-text generation. In this project, we are exploring and evaluate how well adding a tuned prefix to our prompted NLG task on this novel dataset improves performance over the above-described baseline. We are applying the prefix-tuning paper's code, released on GitHub, to our task. In the original paper, the authors found that the table-to-text and summarization tasks had very different optimal prefix lengths (10 and 200, respectively). Since our task differs, we will attempt to optimize a pre-sequence length for our stylistic NLG task.

Figure 1: Visual representation of prefix-tuning method vs. fine-tuning (from Li and Liang 2021 [1])


Fine-tuning
Transformer (Translation)
Transformer (Summarization)
Transformer (Table-to-text)
name Starbucks type coffee shop [SEP] Starbucks serves coffee
Input (table-to-text)
Output (table-to-text)
Prefix-tuning
Prefix (Translation)
Prefix (Summarization)
Prefix (Table-to-text)
Transformer (Pretrained)
name Starbucks type coffee shop [SEP] Starbucks serves coffee

5 Experiments

5.1 Data

Our custom dataset, coined `chveers_300`, is a collection of 300 short sentences and paragraphs scraped from emails sent to author Uma from her father between 2014 and 2022. Every email in the corpus was written by the same author, so the corpus has consistent style even while topics and semantic content varies. The corpus' distinctive style and relatively small size made it align well with the purposes of fine-tuning and prefix-tuning on top of large LMs. Our second dataset, `politeness_1000`, is a subset of 1,000 examples of polite text from emails, taken from a publicly available dataset [2]. We test the three outlined methods on these two different, small corpuses, in order to generalize our results and see how the size of the corpus and number of contained authors affects the performance of each of our techniques. Below is an example of how our data was formatted for training:

```
[ <BOS> ; Prefix ; Masked Suffix ; <EOS> ]
```

The length of the masked suffix was chosen to be `min(len(example), 10)`. We fed the prefix into the model, and then compared its predicted suffix with the gold suffix.

5.2 Data Preprocessing

The `politeness_1000` corpus required no preprocessing.
For `chveers_300`, we first broke each email up into paragraphs, where paragraph breaks were indicated by double `\r\n` sequences. This meant that the size of each of our examples varied largely, ranging from around 12 to 120 tokens. Then, we filtered out the signature on each email ("chVeers, dsp"), which was more or less consistent on every email, so we would not falsely inflate our model's accuracy by giving it a constant string to mimic. Finally, we replaced all real names with placeholder names "Jane" and "John" (mimicking casing). We then used the tokenizer provided by HuggingFace alongside the pre-trained GPT-2 model to tokenize each example.

5.3 Evaluation method

We used BLEU scores to quantitatively evaluate the performance of our models. Specifically, we evaluated each technique on how well it could mimic the style and content of the masked suffix. This "similarity" between the gold output and the actual output was quantified using a BLEU score via the nltk implementation (`nltk.translate.bleu_score`).
We also implemented a rigorous human evaluation procedure. We provided 19 test subjects, each of whom had varying familiarity with the source content, with a survey. They were first provided

with human-generated priming examples so they could get a sense of the nuances presented by each of the 2 styles: authorial style, and politeness. Then, we presented them with side-by-side text examples A and B; both A and B had machine-generated suffixes and human-generated prefixes. One of the given samples had a baseline GPT-2-generated suffix, and the other one had a suffix generated through another technique. We asked each subject whether text A or text B sounded more in-style and coherent, and compared the number of times participants thought that the given technique performed better than baseline GPT-2. We provided each evaluator with (n = 10) such examples for each corpus, resulting in 20 examples per evaluator, and 380 total comparisons.
We used baseline GPT-2, and not gold text, as our baseline, because the state-of-the-art for text generation is still not indistinguishable from human text. Rather than evaluating whether participants could differentiate between human text and machine text, we compared the varying convincingness of differently-generated machine text. We were looking for greater than chance probability that the fine-tuned or prefix-tuned samples were more in-style and coherent than the baseline GPT-2 samples.

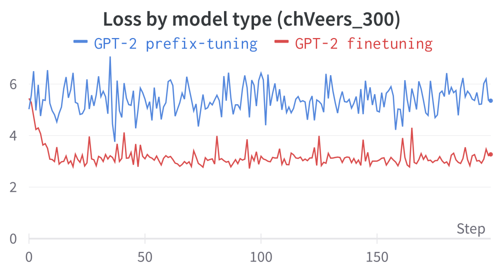
5.4 Experimental Details

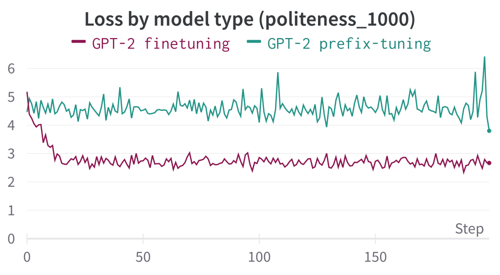
We used a dev set that was $\frac{1}{10}$ of the entire corpus for both of our corpora. Starting with `chveers_300`, we used a dev set of (n = 30). For `politeness_1000` we used a dev set of (n = 100). With both datasets we saw that the training and generation hyperparameters—specifically warmup steps, early stopping, no-repeat-n-grams, temperature, learning rate, and top-p sampling—were extremely impactful on the training loss and dev set inference coherency. In order to optimize these parameters most efficiently, we used a combination of automation and manual trials. For the learning rate, we utilized "Sweeps" from the Weights and Biases API, which visualized for us the relationship between it and the training loss. For generation parameters like no-repeat-n-grams, temperature, learning rate, and top-p sampling, we manually changed the values and saw how they affected our inferences, usually as a prevention measure when we saw repeated inferences in our dev set. Using these methods, we settled on the following a learning rate of 5×10^{-5} for `chveers_300`. We additionally used 200 warm-up steps, and 200 training epochs. For `politeness_1000`, the training parameters remained the same, except for warmup steps, which changed to 400. When generating predicted outputs using beam search over the LM output, we used the same parameters for both tasks: a beam size of 5, a no-repeat-ngram size of 4, and a maximum output suffix length of 2 times the original masked amount. These parameters proved to generate reasonably coherent and non-repetitive outputs.

5.5 Results

The loss for each of our models across 200 training epochs is shown below, for each of the 2 datasets. We can see that for the fine-tuning method, the loss plateaus extremely early, suggesting that perhaps the best hyperparameters were not exploited. We can also see that for the prefix-tuning method, the loss seems to never drop at all. This is discussed in the Analysis section.

Figure 2 & 3: Loss across epochs for `chveers_300` and `politeness_1000`, respectively





After training, we used a test set of (n=30) for both of our datasets, meaning we performed inference on 30 new examples. Overall, fine-tuning seemed to satisfy the task the best for both corpuses, followed by prefix-tuning, which performed equivalently to the baseline. To illustrate these results, we have provided an example of an input prefix and gold suffix accompanied by alternate suffixes generated by a baseline GPT-2, fine-tuned GPT-2, and our prefix-tuning implementation over GPT-2 below.

Figure 4: Generated suffixes from `chveers_300`

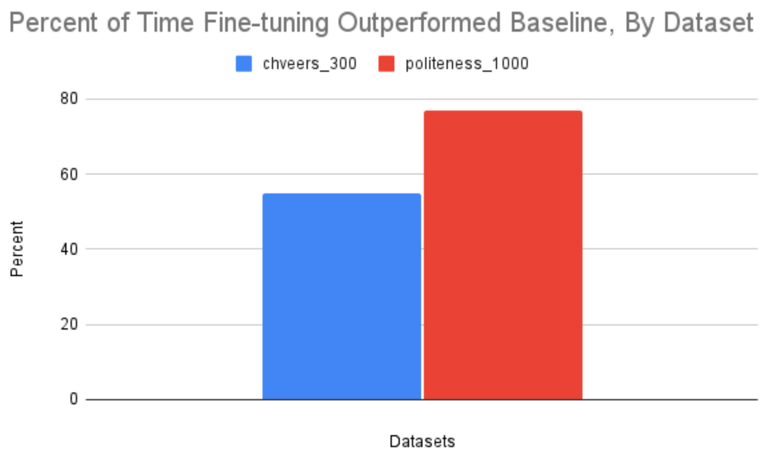
Input Prefix	Maximally enjoy the remainder of your stay in the charm city of SF, i am going to buy a box of mangoes today so that
Gold Suffix	they start ripening by the time you arrive on wed...
Baseline Suffix	you can enjoy the rest of your stay there. I am going to be staying in SF for the rest of my life.
Fine-Tuning Suffix	i can get some cooked food for ur visit...
Prefix-Tuning Suffix	you can enjoy the rest of your stay there. I am going to be staying in SF for the rest of my life.

Figure 5: Generated suffixes from `politeness_1000`

Input Prefix	i have no idea where all the seats are located , but
Gold Suffix	i 'm sure we could probably all relocate around each other .
Baseline Suffix	if you look at the map, you'll see that
Fine-Tuning Suffix	i think it would be a good idea to have them
Prefix-Tuning Suffix	if you look at the map, you'll see that

Most notably, the prefix-tuned ouptut is identical to the baseline GPT output. This is discussed futher in Analysis. Prefix-tuning produces text that is mostly on-topic, but less so in-style. It misses out on many of the stylistic cues seen in the the `chveers_300` corpus, including the informal text and trailing ellipses. The choice of vocabulary is slightly more Westernized and formal, compared to the actual style. The output generated by fine-tuned GPT-2 is relatively in-style (to our eyes). It picks up on many of the stylistic cues mentioned above, as well as lowercase text. In practice this technique also tended to repeat many words, and would trail off in the middle of sentences, just like the baseline and prefix-tuning; we note here that all of our approaches were only as fluent as was the underlying GPT-2 model. The vocabulary of these generated examples consist far more of words actually seen in the training corpus. This comparison of the various techniques can be summarized across all test examples by the fraction of times human evaluators thought a particular method performed better (i.e. was more in-style) than the baseline GPT-2 method; this is visualized below. Note that the techniques performed quite differently on the `politeness_1000` corpus, but more similarly on the `chveers_300` corpus. This is discussed in the Analysis section. Additionally, the human evaluation pictured below was executed with 19 participants. We noticed a trend that, as we added more evaluators, in general, the percent of time fine-tuning seemed to outperform the baseline seemed to go down. (For example, with only 3 evaluators, fine-tuned examples generated from `chveers_300` were seen as better than the baseline 62% of the time, compared to the 55% seen below with 19 evaluators. This is discussed in the Conclusion.)

Figure 6: Human evaluation of fine-tuning vs. baseline on both datasets



Finally, we also performed quantitative analysis using the standard of BLEU scores. However, these BLEU scores, by nature of simply computing n-gram similarity, indicate that the model did quite poorly on the task. This measure was largely unhelpful when actually evaluating the relative success of each technique. The average BLEU scores of the model for each method are shown below.

Figure 7: Average BLEU Scores for each method and datasets

	Un-Trained GPT-2	Fine-Tuned GPT-2	Prefix-Tuned GPT-2
`chveers_300`	0.002867	0.003672	0.002867
`politeness_1000`	0.001402	4.411732e-80	0.001402

6 Analysis

6.1 Fine-Tuning

When fine-tuning the language model on our two corpora (`chveers_300` and `politeness_1000`), we found that our loss plateaued relatively quickly, within 20-30 epochs. While experimenting with different learning rates resulted in loss plateauing at different asymptotes, the timestep at which this flatness occurred tended to stay the same. We believe this may be primarily due to the nature of the task; compared to other NLG tasks like summarization and table-to-text with parallel corpora, open-ended generation is hard to optimize. In particular, with tasks like table-to-text and style transfer, much of the desired output is based on the input prefix (e.g. the table input) and the LM can produce good results largely from attention to the prefix. In our task, on the other hand, the suffix was not a semantic transfer from prefix to suffix. Instead, we were trying to learn style irrespective of semantic content broadly (what was said did not matter as much as how it was said), which is a more complicated representation for the LM to learn.

The model seemed to quickly pick up one some simple stylistic cues from the `chveers_300` set—in particular, the fine-tuned model's outputs tended to be in lowercase, use characteristic abbreviations ("your" $\rightarrow$ "ur"), and trail off with ellipses, three elements that were common among the training examples. We suggest that these may have worked as "low-hanging fruit" for that the model was able to learn over the course of only a handful of epochs. When then faced with the much harder problem of modeling the more semantic trends of examples, gradient descent may have turned less productive.

6.2 Prefix-tuning

Prefix-tuning proved to be significantly more difficult to implement and train than fine-tuning. The loss over time did not significantly decrease over 200 epochs. Rather, it oscillates around the initial value suggesting that the model was never consistently improved at the task as updates were applied. More dramatically, the outputs at inference time were identical to the outputs from the baseline

GPT-2 model, when top p or k sampling was not used (i.e. when the outputs were deterministically generated from the logits output). We offer two reasons for this behaviors:

First, it is possible that the learned weights in the prefix-tuning prefix module were not successfully learned, and that when fed into the GPT-2 model at inference time, they had only a trivial impact that did not have a noticeable effect of the LM's output. This underperformance in training could be the result of nonoptimal hyperparameters (learning rate and warmup steps for the scheduler, as well as prefix length—number of "virtual tokens" in the learned prefix).

Second, it is also likely that there remains an unidentified error in our implementation of prefix-tuning. We derived our code from the original paper's author's provided codebase, which was highly generalized (and resultantly dense) to allow them to tweak many aspects of the model architecture for the purpose of experimenting. We attempted to simplify their implementation down to the most essential modules to avoid errors that could arise from additional complexity, as well as to avoid using any more complex methods that we did not fully understand. It is possible, however, that we made a mistake in this process, affecting either the training process or at inference/generation time. The latter case seems particularly possible considering that while loss fluctuated to some degree, the inference result at generation time was the same as the underlying GPT-2 model without prefix-tuning (as if the learned parameters in the prefix were simply not being factored in).

7 Conclusion

We find that the fine-tuned models performed equal to or better than the baseline, checkpoint models at in-style generation tasks, determined primarily through human evaluation. Although there was only a marginal (likely non-significant improvement) for the `chveers_300` dataset, human evaluators strongly preferred fine-tuned outputs in terms of politeness for the `politeness_1000` examples. This suggests that fine-tuning was successfully able to generate novel text in the style characteristic of the `politeness_1000` dataset, as we sought out to do. In contrast, we were not able apply human evaluation to the prefix-tuning model because after training it still performed identically to the baseline GPT-2 model, meaning that the only difference presented to evaluators would have stemmed from randomness used at generation time. Reasons for this underperformance are discussed in the Analysis section above; we suggest that it is either due to nonoptimal hyperparameters in training or an incorrect implementation of the model. Apparent difficulities in working with stylized datasets like the two corpora used in this project also arose with the fine-tuned models leading to loss plateaus at higher values, including a difficulty of language models to tract the problem of in-style generation low-data settings, perhaps due to its open-endedness. Further work on this topic is warranted, as the applicability of this task reaches far beyond the author-specific cases problematized in this paper. We recommend future research to more broadly and systematically search for hyperparameters for both the fine-tuning (learning rate) and prefix-tuning (learning rate and prefix length) approaches. Additionally, we recommend analysis of human evaluation methods to assess the role of common-knowledge and domain-knowledge competencies on the validity of that evaluation. Lastly, we note that the NLP community would likely benefit from additional publicly-available implementations of prefix-tuning more geared towards application, beyond the demonstration implementation released alongside the original paper.

References

[1] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[2] Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabás Póczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W. Black, and Shrimai Prabhumoye. Politeness transfer: A tag and generate approach. *CoRR*, abs/2004.14257, 2020.