Style Transfer with Multi-iteration Preference Optimization

Shuai Liu and **Jonathan May**
Information Sciences Institute
University of Southern California
{liushuai, jonmay}@isi.edu

Abstract

Numerous recent techniques for text style transfer characterize their approaches as variants of reinforcement learning and preference optimization. In this work, we consider the relationship between these approaches and a class of optimization approaches developed primarily for (non-neural) statistical machine translation, formerly known as 'tuning'. Inspired by these techniques from the past, we improve upon established preference optimization approaches, incorporating multiple iterations of exploration and optimization, and choosing contrastive examples by following a 'hope' vs 'fear' sampling strategy. Cognizant of the difference between machine translation and style transfer, however, we further tailor our framework with a new pseudo-parallel data generation method and a dynamic weighted reward aggregation method to tackle the lack of parallel data and the need for a multi-objective reward. We evaluate our model on two commonly used text style transfer datasets. Through automatic and human evaluation results we show the effectiveness and the superiority of our model compared to state-of-the-art baselines.

1 Introduction

Text style transfer aims to rewrite a given text to match a specific target style while preserving the original meaning. This task has drawn significant attention recently due to its broad range of applications, such as text simplification (Laban et al., 2021), formality transfer (Rao and Tetreault, 2018; Liu et al., 2022), text detoxification (Dale et al., 2021; Hallinan et al., 2023b), authorship transfer (Patel et al., 2023; Liu et al., 2024), and authorship anonymization (Shetty et al., 2018; Bo et al., 2021). Recent approaches have focused on pseudo-parallel data generation (Krishna et al., 2020; Riley et al., 2021) and policy optimization (Gong et al., 2019; Liu et al., 2021b). STEER (Hallinan et al., 2023a) and ASTRAPOP (Liu et al., 2024) combine the two and achieve state-of-the-art performance on text style transfer and authorship style transfer, respectively.

In this work, we seek to advance the frontier of text style transfer, drawing inspiration from the optimization techniques developed in the era of statistical phrasal machine translation, in which the lack of correlation between the log-linear model objective and the desired evaluation metric, typically BLEU (Papineni et al., 2002), was observed (Och, 2003). Approaches to align[1] the two objectives came to be known as *tuning*,[2] beginning with Och (2003), and evolving into online variants (Chiang et al., 2008), rank-based approaches (Hopkins and May, 2011), batch-based approaches (Cherry and Foster, 2012), and several others. Tuning methods follow a generate-and-optimize pattern: a model is used to generate multiple candidate hypotheses per input, and then parameters are adjusted such that the argmax according to the model score also maximizes the evaluation metric. In this regard, tuning methods resemble approaches taken in the application of policy optimization algorithms, such as PPO (Schulman et al., 2017), to generative language modeling (Ouyang et al., 2022). More recent algorithms, such as DPO (Rafailov et al., 2023) and CPO (Xu et al., 2024a), which replace reinforcement learning (RL) in PPO with *preference* optimization (PO), are reminiscent of the pairwise ranking optimization approach to tuning (Hopkins and May, 2011). Given this close relationship between these approaches, we can consider whether other techniques developed to improve MT tuning could be applied to optimization for style transfer.

In this work, we propose Style TrAnsfer with Multi-iteration Preference optimization (STAMP), a two-phase PO training framework, in which we

[1] not to be confused with word alignment.
[2] not to be confused with parameter fine-tuning.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 2663–2681
April 29 - May 4, 2025

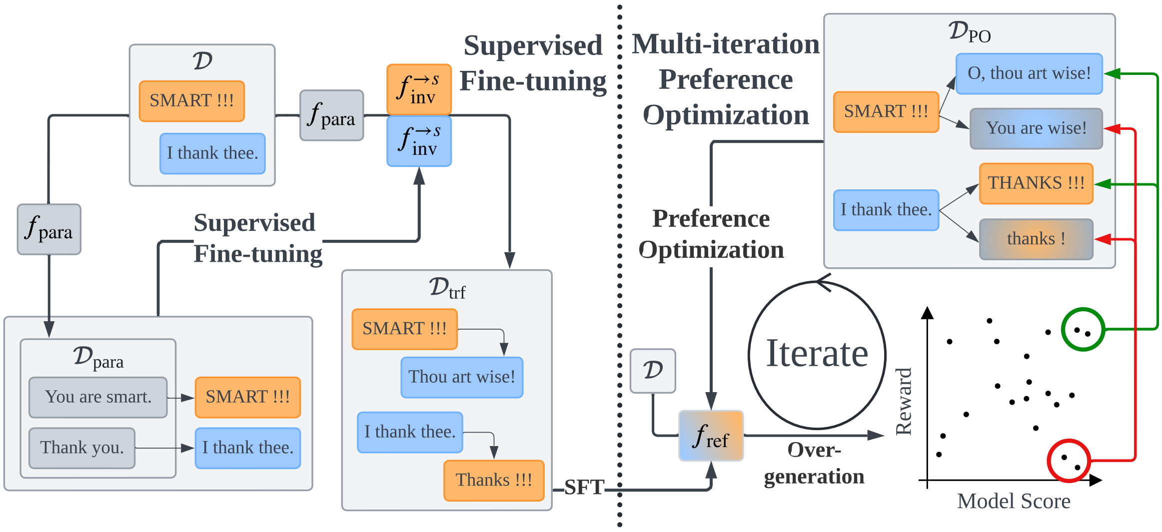

Figure 1: An overview of STAMP, in which we first train a unified style transfer model using supervised fine-tuning on pseudo-parallel data generated from non-parallel data, and then further train the model using multi-iteration preference optimization on preference pairs constructed with hope-and-fear sampling.

first use supervised fine-tuning to build a reference model from pseudo-parallel data and then train the reference model using PO. STAMP is similar to STEER and ASTRAPOP at a high level but is enhanced with two techniques borrowed from MT tuning and two modifications that further adapt it for text style transfer. First, we include *multiple iterations* of preference pair generation followed by model optimization (Och, 2003), which has already been shown to be effective on other Seq2Seq tasks such as mathematical and scientific reasoning (Chen et al., 2024; Pang et al., 2024; Song et al., 2024b; Yuan et al., 2024). Second, following the hope-and-fear sampling in Chiang (2012), for PO, we over-generate outputs using the reference model and construct preference pairs using samples with high model scores and extreme (high or low) task objective scores, in order to avoid dangerous generation and encourage reachable good generation. To improve the quality of the reference model and the balance across the multiple training objectives, we additionally design a new two-step end-to-end pseudo-parallel data generation method and a dynamic reward aggregation method.

We evaluate our model on two popular text style transfer datasets, Grammarly's Yahoo Answers Formality Corpus (GYAFC) (Rao and Tetreault, 2018) and the Corpus of Diverse Styles (CDS) (Krishna et al., 2020). Extensive experiments show that our model outperforms all state-of-the-art baselines on both datasets in both in-domain and out-of-domain evaluation, and demonstrates a higher training efficiency than the strongest baseline.

Our main contributions are:

- We propose a multi-iteration contrastive preference optimization training framework with hope-and-fear preference pair construction for text style transfer.
- We design a new pseudo-parallel data generation strategy and a dynamic weighted rewarded aggregation method to enhance the training framework for text style transfer.
- With the enhancements, our training framework produces style transfer models that achieve state-of-the-art performance on two popular text style transfer datasets.[3]

2 Methodology

In this section, we formalize the text style transfer task and introduce our training framework, STAMP.

2.1 Task Definition

Given a source text $\mathbf{x}$ and a desired target style s, the goal of text style transfer is to generate a fluent rewrite of $\mathbf{x}$, denoted as $\mathbf{x}^{\to s}$, that has the same meaning as $\mathbf{x}$ but is in style s. In this work, we focus on high-resource text style transfer in which we have access to a reasonable number of texts[4] for each target style. Specifically, we have a set of texts with style labels, denoted as $\mathcal{D} = \{(\mathbf{x}_1, s_1), \cdots, (\mathbf{x}_n, s_n)\}$, where $\mathbf{x}_i$ and s_i re-

[3] Code and models sufficient for a reproducibility study are available at `https://github.com/isi-nlp/STAMP`.

[4] In this work, we assume at least 2000 texts per style.

fer to the i^{th} text and its style, respectively. For convenience, we adopt notations from Hallinan et al. (2023a) and denote the **fluency** of a text $\mathbf{x}_i$ as $\mathrm{F}(\mathbf{x}_i)$, the **meaning similarity** between two texts $\mathbf{x}_i$ and $\mathbf{x}_j$ as $\mathrm{MS}(\mathbf{x}_i, \mathbf{x}_j)$, and the **target style strength** of a text $\mathbf{x}_i$ w.r.t. a target style s as $\mathrm{TSS}(\mathbf{x}_i, s)$. Thus, given $\mathcal{D}$, we aim to build a text style transfer system that maximizes three independent objectives: $\mathrm{F}(\mathbf{x}^{\to s})$, $\mathrm{MS}(\mathbf{x}, \mathbf{x}^{\to s})$, and $\mathrm{TSS}(\mathbf{x}^{\to s}, s)$.[5]

2.2 Framework Overview

STAMP is a preference optimization-based training framework that contains two main stages, a supervised fine-tuning (SFT) stage and a multi-iteration preference optimization (PO) stage. In the SFT stage, we first generate a dataset $\mathcal{D}_{\text{trf}}$ of end-to-end pseudo-parallel style transfer pairs from the (non-parallel) dataset $\mathcal{D}$ and then train a style transfer model f_{SFT} on $\mathcal{D}_{\text{trf}}$ using supervised fine-tuning. In the PO stage, we train a model initialized to f_{SFT} using multi-iteration PO[6] to directly maximize the three objectives, TSS, MS, and F, and obtain our final transfer model f_{PO}.

2.3 Supervised Fine-tuning

Due to a lack of parallel data, we adopt the technique described by Krishna et al. (2020), in which style-oriented paraphrasing is used to generate pseudo-parallel transfer data for each target style. Specifically, we paraphrase the texts in $\mathcal{D}$ using a general paraphraser f_{para} similar to Krishna et al. (2020) and Hallinan et al. (2023a). To ensure meaning similarity preservation of the paraphrases, we generate k_{para} paraphrases for each text $\mathbf{x}_i \in \mathcal{D}$ and select the one with the highest meaning similarity to the original text, denoting it $\mathbf{p}_i$. We then obtain a dataset of paraphrases $\mathcal{D}_{\text{para}} = \{\mathbf{p}_1, \cdots, \mathbf{p}_n\}$. For each target style s, we train a Seq2Seq model $f_{\text{inv}}^{\to s}$[7] on $\{(\mathbf{p}_i \to \mathbf{x}_i) \mid 0 \le i \le n \text{ and } s_i = s\}$ to maximize

$$p(\mathbf{x} \mid \mathbf{p}) = \prod_{i=1}^{|\mathbf{x}|} p(\mathbf{x}[i] \mid \mathbf{p}, \mathbf{x}[<i]) \tag{1}$$

where $\mathbf{x}[i]$ and $\mathbf{x}[<i]$ represent the i^{th} token in $\mathbf{x}$ and tokens preceding the i^{th} token in $\mathbf{x}$, respectively.

Following Krishna et al. (2020), we can transfer the style of a text $\mathbf{x}$ to a style s through

$$\mathbf{x}^{\to s} = f_{\text{inv}}^{\to s}(f_{\text{para}}(\mathbf{x})) \tag{2}$$

where $\mathbf{x}^{\to s}$ is the transferred text. However, the two-step generation breaks the gradient connection between $\mathbf{x}$ and $\mathbf{x}^{\to s}$ which is needed in the PO stage to maximize the meaning similarity between $\mathbf{x}$ and $\mathbf{x}^{\to s}$. Therefore, we need an end-to-end pseudo-parallel dataset $\mathcal{D}_{\text{trf}}$ to train a model that directly transfers a source text to each target style with no intermediate step.

To obtain $\mathcal{D}_{\text{trf}}$, we transfer the texts in $\mathcal{D}$ using f_{para} and $f_{\text{inv}}^{\to s}$ for each target style s. Specifically, for each target style s, we transfer the texts in other styles in $\mathcal{D}$ using Eq. 2 and obtain a dataset of style transfer pairs $\mathcal{D}_{\text{trf}}^{\to s} = \{(\mathbf{x}_i \to \mathbf{t}_i, s) \mid (\mathbf{x}_i, s_i) \in \mathcal{D} \text{ and } s_i \neq s\}$, where $\mathbf{t}_i = f_{\text{inv}}^{\to s}(f_{\text{para}}(\mathbf{x}_i))$ is a transfer of x_i in style s. To obtain high-quality transferred texts, we generate k_{sft} transfers for each source text and select the one with the highest $\mathrm{F} \cdot \mathrm{MS}^{\tau_{\text{ms}}} \cdot \mathrm{TSS}$, where $\tau_{\text{ms}} > 1$ is a temperature hyperparameter incorporated into the MS term to emphasize meaning similarity. We then construct $\mathcal{D}_{\text{trf}}$ by combining $\mathcal{D}_{\text{trf}}^{\to s}$ for all target styles and train an end-to-end style transfer model f_{SFT} on the combined data $\mathcal{D}_{\text{trf}}$ to maximize

$$p(\mathbf{t} \mid \mathbf{x}) = \prod_{i=1}^{|\mathbf{t}|} p(\mathbf{t}[i] \mid \mathbf{x}, \mathbf{t}[<i], s) \tag{3}$$

Note that unlike Eq. 2, the probability in Eq. 3 is also conditioned on s because we adopt the unified model setting in (Hallinan et al., 2023a). That is, we have a single transfer model for all target styles and control the target style with control codes.

2.4 Multi-iteration Preference Optimization

We further train the SFT model f_{SFT} from the previous stage with multi-iteration PO to directly optimize the model on the style transfer objectives: F, MS, and TSS. To apply PO (Rafailov et al., 2023; Xu et al., 2024a) we first generate paired preference data from a *reference model* f_{ref} and then train a model on this offline preference data in a contrastive manner starting from the reference model. Inspired by Och (2003) and recent studies in iterative PO, such as Yuan et al. (2024) and Chen et al. (2024), we perform PO for multiple iterations to improve over the offline-only training, updating the reference model between iterations. Specifically, in iteration i, we construct preference dataset $\mathcal{D}_{\text{PO}}^i$ by transferring texts drawn from $\mathcal{D}$, using reference model f_{ref}^i. We use PO (Rafailov et al., 2023; Xu et al., 2024a) to train a model initialized to f_{ref}^i to match the preferences in $\mathcal{D}_{\text{PO}}^i$; we

[5] For brevity, we omit the arguments where unambiguous.
[6] See § 3.4 for details on the choice of PO used here.
[7] 'inverse' due to data provenance, c.f. (Krishna et al., 2020)

call the resulting model f^i_{PO}. We define f^1_{ref} to be f_{SFT} and in all other cases we define f^i_{ref} to be f^{i-1}_{PO}. We next detail how the preference pairs in $\mathcal{D}^i_{\text{PO}}$ are constructed and the reward function used in this process.

2.4.1 PO Data Generation

We construct the preference dataset from $\mathcal{D}$ using the hope-and-fear sampling strategy in Chiang (2012), which can encourage the model to generate "reachable" outputs with high reward scores and prevent the model from generating "reachable" outputs with low reward scores. While that work used BLEU (Papineni et al., 2002) as a preference metric, we instead use our style transfer reward $\mathcal{R}$ which is detailed in § 2.4.2. Specifically, for each style s, we generate k_{PO} rewrites of each text $\mathbf{x}_i$ in $\mathcal{D}$, whose initial style $s_i \neq s$, into style s and select the preference pair from the rewrites based on both the reward scores $\mathcal{R}$ and the model scores $\mathcal{M}$ of the rewrites, where $\mathcal{M}$ is the average token-level probability w.r.t. f_{ref}. We select the rewrite with the highest $\mathcal{M}^{\tau_\mathcal{M}} + \mathcal{R}$ as the "winning" rewrite $\mathbf{t}^w_i$ and the rewrite with the highest $\mathcal{M}^{\tau_\mathcal{M}} - \mathcal{R}$ as the "losing" rewrite[8] $\mathbf{t}^l_i$, where $\tau_\mathcal{M}$ is the temperature controlling the weight of model score.[9] We then obtain a new dataset $\mathcal{D}^{\to s}_{\text{PO}} = \{(\mathbf{x}_i \to (\mathbf{t}^w_i, \mathbf{t}^l_i), s) \mid (\mathbf{x}_i, s_i) \in \mathcal{D}\}$ for each style s. Combining $\mathcal{D}^{\to s}_{\text{PO}}$ for all styles, we finally obtain the PO dataset $\mathcal{D}_{\text{PO}}$.

2.4.2 Reward Function

To directly maximize the three objectives, F, MS, and, TSS, we use an aggregation of them as the reward function $\mathcal{R}$. The most straightforward aggregation is to take the product of the three as in Hallinan et al. (2023a). However, since the three objectives are independent, the probability of generating samples that have high scores in all three objectives is very low. Our preliminary experiments show that samples with high total rewards can also have low single-objective scores, which naturally results in preference pairs in which the "winning" outputs have lower single-objective scores. We refer to these as *reversed single-objective scores*. When the percentage of reversed single-objective scores is high, we observe a degradation in the corresponding objective after PO. To prevent the degradation in any objective, we propose to use a weighted product, which is given by

$$\mathcal{R} = \text{TSS}^\alpha \cdot \text{MS}^\beta \cdot \text{F}^\gamma \tag{4}$$

where α, β, and γ are temperature parameters.

We dynamically calculate α, β, and γ based on the number of reversed single-objective scores in the preference pairs for each iteration. For convenience, we denote the number of reversed single-objective scores for each objective as r_{TSS}, r_{MS}, and r_{F}.[10] We first set $\beta = \gamma = 1$ and set α to be the smallest positive integer such that $r_{\text{TSS}} < r_{\text{MS}}$ and $r_{\text{TSS}} < r_{\text{F}}$. Then, we fix α and γ and set β to be the largest positive integer such that $r_{\text{MS}} > r_{\text{TSS}}$. Finally, we fix α and β and set γ to be the largest positive integer such that $r_{\text{F}} > r_{\text{TSS}}$ and $r_{\text{F}} > r_{\text{MS}}$. We set an upper bound $\tau_{\max}$ to α, β, and γ to prevent $\mathcal{R}$ from leaning too much to any objective.

3 Experiments

We evaluate STAMP on two text style transfer datasets in both in-domain and out-of-domain settings and compare STAMP with the state-of-the-art baseline approaches. In this section, we detail the experimental setup and the model implementation.

3.1 Datasets

We use two style transfer datasets in this work: (1) **Corpus of Diverse Styles (CDS)** (Krishna et al., 2020), which contains non-parallel texts in 11 different styles, such as Shakespeare and English Tweets, and (2) **Grammarly's Yahoo Answers Formality Corpus (GYAFC)** (Rao and Tetreault, 2018), which contains non-parallel formal and informal texts for training and a small number of parallel transfer pairs for tuning and test. In this work, we only use non-parallel texts with style labels for training, validation, and test.

To reduce computational costs, we use a subset of each dataset. Specifically, we sample 2000 texts per style for training, and 200 per style for validation. For CDS we sample 200 per style for test, while for GYAFC we sample 1000 per style. When constructing the end-to-end pseudo-parallel dataset $\mathcal{D}_{\text{trf}}$, for each target style, we sample 200 and 20 source texts from each of the other styles for training and validation, respectively. In the in-domain testing, we transfer the test texts in each style to all

[8] also called "chosen" and "rejected" rewrites in PO literature (e.g., Rafailov et al., 2023).

[9] In practice, we find using model score does not benefit performance, so we drop this term for STAMP, which reduces the preference pair selection criteria to the sample with the highest $\mathcal{R}$ and $-\mathcal{R}$; a detailed comparison is shown in § 4.3.

[10] r_{TSS}, r_{MS}, and r_{F} are functions of α, β, and γ, so we recalculate rs each time we change the value of α, β, or γ.

other styles in the same dataset and calculate the total average scores and average scores grouped by the target style. In the out-of-domain testing, we transfer all test texts in each dataset to all styles in the other dataset and calculate the same scores. We elaborate on metric scores in § 4.1.

Besides the style transfer datasets, we also use a paraphrase dataset, **ParaNMT** (Wieting and Gimpel, 2018) to train the paraphraser used for pseudo-parallel data generation. Specifically, we use the filtered version containing 75k paraphrase pairs in Krishna et al. (2020).

3.2 Reward Models

We have a reward model for each of the three objectives, TSS, MS, and F. For convenience, we use the same notations to refer to the objective functions and the corresponding reward models in this paper.

Target Style Strength (TSS) We use a single style classifier, f_{cls} with multiple binary sigmoid classification heads to calculate the TSS for each target style. We train f_{cls} from the pre-trained RoBERTa-large model (Liu et al., 2019b) on the same training and validation splits. We use the sigmoid scores from the classification heads as the TSS scores which range from 0 to 1.

Meaning Similarity (MS) We assess the meaning similarity between the source text and the transferred text using the cosine similarity between the semantic embeddings of the two texts. The semantic embeddings are calculated using SBERT[11] (Reimers and Gurevych, 2019). Technically, the cosine similarity of two embeddings ranges from -1 to 1, but negative cosine similarity is very rare in our experiments since we always the similarity between two paraphrases. Following Hallinan et al. (2023a), we clip negative values to 0 to ensure that MS ranges from 0 to 1.

Fluency (F) To measure the fluency of a text, we use a text classifier[12] trained on the Corpus of Linguistic Acceptability (CoLA) (Warstadt et al., 2019). The softmax score of the "grammatical" class is used as the F score which also ranges from 0 to 1.

3.3 Baseline Approaches

We compare STAMP with 4 strong baselines: GPT prompting (Reif et al., 2022), STRAP (Krishna et al., 2020), STEER (Hallinan et al., 2023a), and ASTRAPOP (Liu et al., 2024). The first two are the most widely used training-free and SFT approaches, while the latter two are the two SOTA models.

GPT prompting uses the zero- and few-shot capability of GPT-3.5-turbo to transfer texts to the target style given just the name of the style and 5 target style exemplars (5-shot) or no exemplars (zero-shot).

STRAP transfers a text by paraphrasing the text with a diverse paraphraser followed by an inverse paraphraser trained on pseudo-parallel transfer data generated by the diverse paraphraser.

STEER generates pseudo-parallel data using an expert-guided generation technique (Liu et al., 2021a), and trains an end-to-end style transfer model on the generated data using a reinforcement learning algorithm (Lu et al., 2022).

ASTRAPOP adopts the same paraphrase-and-inverse-paraphrase pipeline as STRAP but trains the inverse paraphraser using policy optimization or PO to directly maximize the target style strength.

3.4 Implementation Details

We implement all Seq2Seq models in STAMP, including the paraphraser and all transfer models, as decoder-only Seq2Seq models (Wolf et al., 2019) based on pre-trained LLaMA-2-7B (Touvron et al., 2023). The input and output are concatenated together with a separator token "[SEP]." For the unified transfer model f_{SFT}, we prepend a style code for the target style (e.g., "[SHAKESPEARE]" and "[FORMAL]") to the input to control the output style. We use CPO (Xu et al., 2024a) in the multi-iteration PO stage. We choose CPO instead of the most popular PO algorithm, DPO (Rafailov et al., 2023), since CPO has been shown to be more efficient and effective (Xu et al., 2024a; Liu et al., 2024). Also, compared to DPO, CPO has an additional negative log-likelihood term that is found to be significant for multi-iteration preference optimization (Pang et al., 2024). We stop PO training at the iteration where the validation TSS starts to decrease and use the model from the previous iteration as the final model. For fairness, all non-GPT baselines are also implemented based on LLaMA-2-7B and use the same paraphraser as STAMP. We use gpt-3.5-turbo-0125 for all GPT-based approaches. See § B for hyperparameters, training runtime, and GPT zero- and few-shot prompts.

[11]We use the variant with the best sentence embedding performance, which is all-mpnet-base-v2.

[12]`https://huggingface.co/cointegrated/roberta-large-cola-krishna2020`

Approach	CDS				GYAFC			
	TSS	MS	F	Agg.	TSS	MS	F	Agg.
GPT zero-shot	$0.189^{\ddagger}$	$0.705^{\ddagger}$	$0.803^{\dagger}$	$0.104^{\ddagger}$	$0.672^{\ddagger}$	$0.788^{\ddagger}$	**0.968**	$0.489^{\ddagger}$
GPT 5-shot	$0.199^{\ddagger}$	$\underline{0.735}^{\dagger}$	$\underline{0.805}^{\dagger}$	$0.112^{\ddagger}$	$0.667^{\ddagger}$	$\underline{0.800}^{\dagger}$	<u>0.965</u>	$0.495^{\ddagger}$
STRAP	$0.382^{\ddagger}$	$0.626^{\ddagger}$	$0.759^{\ddagger}$	$0.158^{\ddagger}$	$0.618^{\ddagger}$	$0.735^{\ddagger}$	$0.913^{\ddagger}$	$0.409^{\ddagger}$
STEER	$\underline{0.654}^{\dagger}$	$0.672^{\ddagger}$	**0.905**	$\underline{0.395}^{\dagger}$	<u>0.951</u>	$0.776^{\ddagger}$	$0.930^{\ddagger}$	$\underline{0.686}^{\dagger}$
ASTRAPOP	$0.542^{\ddagger}$	$0.600^{\ddagger}$	$0.755^{\ddagger}$	$0.221^{\ddagger}$	$0.783^{\ddagger}$	$0.734^{\ddagger}$	$0.924^{\ddagger}$	$0.525^{\ddagger}$
STAMP	**0.746**	**0.801**	$0.801^{\dagger}$	**0.474**	**0.958**	**0.921**	$0.941^{\ddagger}$	**0.828**

Table 1: The automatic evaluation results on in-domain inputs on the CDS and the GYAFC datasets. The best and the 2nd best scores in each column are shown in **bold** and <u>underline</u>, respectively. "†" and "‡" indicate the score is significantly ($p < 0.05$) worse than the best score and the top 2 scores in the same column, respectively, determined by resampling t-test.

4 Results

In this section, we present the quantitative experimental results and a qualitative case study. Because of the limited resources, we conduct all experiments for a single run and perform t-tests on the results.[13]

4.1 Automatic Evaluation

Automatic evaluation results on in-domain input are shown in Table 1, using the same reward models introduced in § 3.2 to calculate TSS, MS, and F. To assess the overall performance, we use a single aggregate score Agg. $=$ TSS $\cdot$ MS $\cdot$ F.[14] According to the aggregated score (Agg.), STAMP outperforms all baselines on the overall performance by a large margin on both datasets. Looking at the per-objective scores, STAMP has the best target style strength (TSS) and meaning similarity (MS), but its fluency (F) is relatively lower, and this disadvantage is more obvious on the CDS dataset. STEER has the best overall performance (Agg.) among the baselines on both datasets, while the overall performance of other baselines are mixed across the two datasets. For the breakdown scores on each subset in CDS and GYAFC, please see § A.4.

Table 2 shows automatic evaluation results of the 'out-of-domain' style transfer experiments, in which we transfer the texts in each dataset to the styles in the other dataset, in order to determine whether our results hold up when transferring between styles of different provenance. They do; the out-of-domain results are generally consistent with the in-domain results. The best model in each column in Table 2 is the same as Table 1, which is also true for the second best model in most columns. Also, STAMP still has the best TSS, MS, and aggregated score (Agg.) among all approaches, and STEER still has the best overall performance (Agg.) among the baselines.

We also show that STAMP models are not overfitted on the training rewards by evaluating them on alternative metrics unseen during training and are robust to different hyperparameters by training the models with perturbed hyperparameters in § A.1 and § A.2. Besides, STAMP is more computationally efficient than the strongest baseline, STEER. When compared head-to-head using an identical base model but varying only the core design choices (DExpert data generation and Quark for STEER vs. STRAP data generation and iterative CPO for STAMP), we find that STAMP reaches parity with STEER in 43% and 82% of the training time on CDS and GYAFC, and converges efficiently with stronger performance.[15]

4.2 Human Evaluation

We conduct a human evaluation on the CDS dataset for STAMP, the best-performing baseline (STEER), and the best GPT-prompting baseline (GPT 5-shot) to assess their performance on the three style transfer objectives: TSS_h, MS_h, and F_h.[16] For TSS_h, we show 5 exemplars for the style of the input text and 5 exemplars for the target style, and ask the annotator to select the style of the transferred text out of these two styles. The sample gets a score of 1 if the target style is selected, and 0 otherwise. For MS_h and F_h, we ask whether the transferred text has a similar meaning to the input text and whether the transferred is fluent, respectively, and collect

[13] See § B.1 for details.

[14] Note that the average Agg. on the test set is the average of Agg. for each transfer pair, not a simple product of average TSS, MS, and F.

[15] See § B.4 for details.

[16] We use the subscript h to distinguish human metrics from automatic metrics.

Approach	CDS				GYFAC			
	TSS	MS	F	Agg.	TSS	MS	F	Agg.
GPT zero-shot	0.246‡	0.657‡	0.855‡	0.138‡	0.672‡	0.752†	**0.909**	0.455‡
GPT 5-shot	0.289‡	0.708†	0.868†	0.175‡	0.722‡	0.752†	0.902	0.486‡
STRAP	0.426‡	0.629‡	0.810‡	0.194‡	0.692‡	0.689‡	0.852‡	0.402‡
STEER	0.654†	0.706†	**0.927**	0.426†	0.850†	0.734‡	0.875	0.544†
ASTRAPOP	0.579‡	0.606‡	0.808‡	0.259‡	0.816†	0.685‡	0.863‡	0.479‡
STAMP	**0.787**	**0.816**	0.877†	**0.562**	**0.964**	**0.864**	0.827‡	**0.687**

Table 2: The automatic evaluation results on out-of-domain inputs on the CDS and the GYAFC datasets. The best and the 2nd best scores in each column are shown in **bold** and underline, respectively. "†" and "‡" indicate the score is significantly ($p < 0.05$) worse than the best score and the top 2 scores in the same column, respectively, determined by resampling t-test.

Approach	TSS	MS_h	F_h	$\text{Agg.}_{\sim h}$
GPT 5-shot	0.16	0.75	0.90	0.11
STEER	0.58	0.62	**0.92**	0.33
STAMP	**0.79**	**0.75**	0.80	**0.47**

Table 3: The human evaluation results on in-domain inputs on the CDS datasets. The best and the 2nd best scores in each column are shown in **bold** and underline, respectively.

the answers using a three-level Likert scale ranging from 0 to 2, which is then halved to fit in the 0 to 1 range. See § B.5 for the detailed instructions used in the human evaluation. We randomly choose 5 samples from each of the 11 target styles for each of the three models, which yields 165 samples in total, and collect up to three annotations for each sample. Seven volunteer NLP experts are recruited for annotation.

We perform an independent sample t-test on the annotation results and find statistically significant differences in MS_h and F_h but not in TSS_h,[17] which is in line with our expectation since the style classification has been found to be hard for untrained humans[18] (Krishna et al., 2020; Hallinan et al., 2023a). Therefore, following Krishna et al. (2020) and Hallinan et al. (2023a), we calculate the quasi aggregated score $\text{Agg.}_{\sim h}$ using TSS,[19] MS_h, and F_h. Formally, $\text{Agg.}_{\sim h} = \text{TSS} \cdot \text{MS}_h \cdot \text{F}_h$. As shown in Table 3, STAMP has the best meaning similarity (MS_h) and overall performance ($\text{Agg.}_{\sim h}$), but its fluency is worse than STEER and GPT 5-shot transfer, which is consistent with the automatic evaluation results.

[17] See § A.3 for the raw human evaluation scores and the result of the t-test.

[18] We still conduct human study for style because we set up the task as a simpler verification task to see whether we can get meaningful results.

[19] which is calculated from the human study samples using the automatic TSS metric.

4.3 Ablation Studies

In this section, we demonstrate the effects of our four main contributions in STAMP: multi-iteration PO, hope-and-fear sampling, weighted reward aggregation, and end-to-end pseudo-parallel data generation.

Multi-iteration PO & Weighted $\mathcal{R}$ We show the performance evolution of STAMP and STAMP with unweighted $\mathcal{R}$ over the multi-iteration PO training in Figure 2. In general, the overall performance (Agg.) of both models keeps increasing over the iterations, which indicates the effectiveness of multi-iteration optimization. STAMP with unweighted $\mathcal{R}$ performs slightly better than STAMP, but it has a severe degradation in meaning similarity (MS), and the scores in the three objectives have a substantial difference after training. In contrast, with the weighted reward aggregation, STAMP shows a higher stability in all scores. Only fluency (F) exhibits a slight decrease, and scores in all three objectives converge to a similar value at the end of the training.

Hope-and-fear Sampling The results of hope-and-fear sampling ablation are shown in Table 4. As mentioned in § 2.4.2, we do not use the model score term in hope-and-fear sampling for preference pair construction since it does not improve the performance, which can be observed from the "$\tau_{\mathcal{M}} = 0.1$" row in Table 4. The last three rows in Table 4 show that both dropping over-generation ($k_{\text{PO}} = 2$) and using a random other sample (Random t^l) or the sample with the second highest reward (High $\mathbf{t}^l$) as the "losing" sample undermine the overall performance of STAMP.

Pseudo-parallel Data Generation We demon-

Approach	CDS				GYAFC			
	TSS	MS	F	Agg.	TSS	MS	F	Agg.
STAMP	**0.746**	0.801‡	<u>0.801</u>†	**0.474**	0.958‡	0.921†	0.941†	**0.828**
$\tau_{\mathcal{M}} = 0.1$	0.720†	0.796‡	0.800†	<u>0.454</u>†	<u>0.965</u>	0.910‡	0.943†	<u>0.826</u>
$k_{\text{PO}} = 2$	<u>0.745</u>	0.688‡	**0.816**	0.411‡	**0.970**	0.878‡	**0.947**	0.804‡
Random $\mathbf{t}^l$	0.640‡	**0.836**	0.780‡	0.412‡	0.950‡	<u>0.924</u>†	0.937	0.822
High $\mathbf{t}^l$	0.592‡	<u>0.826</u>†	0.796†	0.384‡	0.928‡	**0.936**	0.932‡	0.810‡

Table 4: Hope-and-fear sampling ablations, evaluated automatically on in-domain inputs on the CDS and the GYAFC datasets. The best and the 2[nd] best scores in each column are shown in **bold** and <u>underline</u>, respectively. "†" and "‡" indicate the score is significantly ($p < 0.05$) worse than the best score and the top 2 scores in the same column, respectively, determined by resampling t-test.

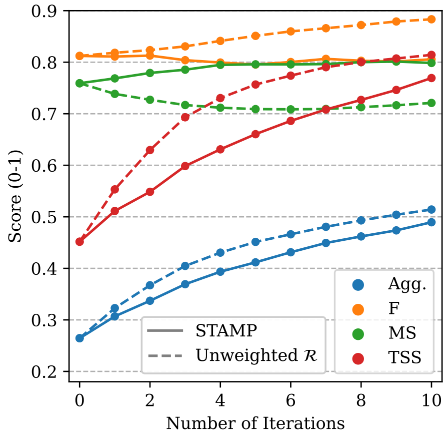

Figure 2: The value of iterative CPO on performance in STAMP and STAMP with unweighted $\mathcal{R}$, shown on the CDS dataset (test split). Iteration 0 refers to the SFT model before PO.

strate the superiority of our two-step end-to-end pseudo-parallel data generation method by comparing the STAMP SFT model, f_{SFT}, with the best-performing baseline SFT style transfer model, STRAP. The overall performance (Agg.) of the two models is shown in Table 5. With our method, the overall performance of f_{SFT} is much higher than STRAP on both datasets, which provides a better starting point for PO.

	CDS	GYAFC
STRAP	0.158	0.409
f_{SFT}	**0.264**	**0.657**

Table 5: The overall performance (Agg.) of STRAP and the STAMP SFT model (f_{SFT}) on CDS and GYAFC. The best score in each column is shown in **bold**.

4.4 Qualitative Case Study

We show an example from the CDS test set in Table 6 as a case study. In this example, we transfer a text in the style of music lyrics into the style of Shakespeare using STAMP and all baseline approaches. STAMP (iter. 9) maximally preserves the meaning of the original sentence and accurately reflects the target style using the words "'tis", "o'er", and "That" with uppercase "T". Other approaches either fail to generate strong target style indicators or change the meaning of the original sentence to some extent. Moreover, the example also demonstrates that STAMP gradually improves the model performance over the multi-iteration training. Specifically, STAMP (iter. 1) successfully reflects Shakespeare's style by using the word "o'er" but reverses the meaning of the original sentence. STAMP (iter. 5) successfully captures the original meaning but suffers from hallucination. The final model, STAMP (iter. 9) fully preserves the original meaning and demonstrates a stronger Shakespearean style.

5 Related Work

Text Style Transfer Due to the lack of parallel style transfer data, only a limited number of studies address this task as a supervised or semi-supervised Seq2Seq task, which requires a certain amount of parallel data for training and/or tuning (Zhu et al., 2010; Rao and Tetreault, 2018; Wang et al., 2019; Shang et al., 2019; Xu et al., 2019; Zhang et al., 2020; Kim et al., 2022; Raheja et al., 2023). Although these approaches work well when parallel data is available, none generalize well to styles with no parallel data. As a result, most works in this area focus on unsupervised approaches that require only non-parallel data or even no data. These works mainly approach the task via latent representation disentanglement and manipulation (Lample et al., 2019; Liu et al., 2019a; John et al., 2019; Jin et al., 2020), style-related pattern editing (Madaan

Original	I'm not sorry that it's over
GPT zero-shot	I doth not lament the end that hath befallen us.
GPT 5-shot	I doth not lament the ending of this tale.
STRAP	I am not sorry That he is gone.
ASTRAPOP	Now is the winter of our discontent Made glorious summer by this sun of York.
STEER	I do not regret that it is done.
STAMP (iter. 1)	I am sorry that's it is o'er.
STAMP (iter. 5)	I am not sorry that it is over, Nor sorry that I did not know it.
STAMP (iter. 9)	I am not sorry That 'tis o'er.

Table 6: A style transfer example from the style of music lyrics to the style of Shakespeare.

et al., 2020; Malmi et al., 2020; Reid and Zhong, 2021; Luo et al., 2023), pseudo-parallel transfer data construction (Krishna et al., 2020; Riley et al., 2021), policy optimization (Gong et al., 2019; Liu et al., 2021b; Deng et al., 2022; Hallinan et al., 2023a; Liu et al., 2024), and LLM zero- or few-shot prompting (Reif et al., 2022; Suzgun et al., 2022; Patel et al., 2023).

Among these approaches, two of the policy optimization based approaches, STEER (Hallinan et al., 2023a) and ASTRAPOP (Liu et al., 2024) achieve the best performance on text style transfer and authorship style transfer, respectively. Their high-level training frameworks both combine pseudo-parallel data generation and policy optimization, but their specific approaches differ. For pseudo-parallel data generation, STEER uses a paraphraser guided by an expert and an anti-expert, while ASTRAPOP simply paraphrases the texts in the target style and uses these paraphrase-to-target transfer pairs. For policy optimization, STEER uses an RL algorithm, Quark, while ASTRAPOP tries three options: one RL algorithm, PPO (Schulman et al., 2017), and two PO algorithms, DPO (Rafailov et al., 2023) and CPO (Xu et al., 2024a). Our framework shares the same high-level procedure with STEER and ASTRAPOP, but we design a new pseudo-parallel data generation method and also enhance the PO stage with multi-iteration training, weighted reward aggregation, and hope-and-fear preference pair construction, These enhancements dramatically improve the performance of STAMP over STEER and ASTRAPOP.

Preference Optimization PO (Rafailov et al., 2023; Song et al., 2024a; Xu et al., 2024a) is a class of RL-free policy optimization algorithms which has been broadly applied to train generative language models on direct task objectives instead of the language modeling loss and is closely related to (pre-neural) machine translation objective 'tuning' (Och, 2003; Chiang et al., 2008; Hopkins and May, 2011). Rafailov et al. (2023) show that PO is more stable and efficient than traditional RL-based algorithms on sentiment generation and text summarization (Rafailov et al., 2023). It has also been successfully applied to many other NLP tasks, such as training helpful and harmless assistants (Song et al., 2024a), machine translation (Xu et al., 2024a), and authorship style transfer (Liu et al., 2024). Later works (Xiong et al., 2023; Xu et al., 2024b; Yuan et al., 2024; Chen et al., 2024; Pang et al., 2024; Song et al., 2024b) extend the offline PO algorithms by performing the optimization for multiple iterations and further improve the performance of the models. In this work, we adopt the multi-iteration PO for STAMP and enhance it with weighted reward aggregation and hope-and-fear preference pair construction, which improve the effectiveness of multi-iteration PO training.

6 Conclusion

We present STAMP, a multi-iteration preference optimization training framework for text style transfer, in which an end-to-end pseudo-parallel data generation pipeline provides a strong reference model, a preference pair construction strategy improves the effectiveness of PO training, and weighted reward aggregation ensures balance across multiple objectives over multi-iteration training. We evaluate STAMP on two commonly used text style transfer datasets; demonstrating superior performance over all state-of-the-art style transfer approaches.

Limitations

Although achieving the state-of-the-art performance on two text style transfer datasets, STAMP has two main limitations. First, we observe repetitions and hallucinations in some transferred texts. The potential reason is that PO training increases the peakiness of the model, which means the probability of generating the tokens that are frequent in the target style increases disproportionately (Choshen et al., 2020; Kiegeland and Kreutzer, 2021). The occurrence of repetitions and hallucinations also indicates that our reward model cannot fully capture all aspects of the desired objectives. Two possible solutions are developing PO algorithms that are less vulnerable to the increased peakiness and developing better reward models. These are two promising directions for future studies but are out of the scope of the current work which focuses on the multi-iteration extension of existing preference optimization algorithms and the strategies for preference pair construction.

Second, as discussed in § 4.3, the weighted reward aggregation method is effective on the CDS dataset but is not very useful on the GYAFC dataset because formality transfer is a relatively easier task, and it is more likely to generate high-quality samples with balanced single-objective scores. It could be useful to add a control mechanism to determine when using the weighted aggregation is beneficial to prevent overbalanced single-objective scores on easy tasks.

Ethical Considerations

As a general text style transfer framework, STAMP can transfer texts to any target style given an adequate amount of non-parallel data, which means it can potentially be used to generate unethical texts such as transferring normal texts into an offensive or profane style. Moreover, although STAMP is not specifically designed for authorship transfer, it can still serve that purpose by transferring the texts into the style of a particular author, which can be unethical if used without authorization. However, privatization of an author's style can also be used to enable oppressed people to communicate freely without the fear of recrimination. In any case, as we and others show, the state of the art of style transfer is not yet advanced for either privacy or mimicry to be a significant concern in a deployed system. Our work is strictly intended for research and personal use on public or authorized data.

Some texts in the datasets used in this work (though collected and released elsewhere) contain words or ideas that may cause harm to others. We do not generally filter out those texts, so that we may maximally preserve the characteristics of different styles. However, for human studies, we remove all texts with personal identifiable information (PII) to ensure privacy and remove texts that contain profane language to minimize harm to human subjects. We exclude these texts instead of masking out PII or profane tokens, since masks may influence annotators' judgments regarding meaning similarity and fluency. The protocols of our human studies have been approved by an institutional review board.

Acknowledgments

This research is supported in part by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), via the HIATUS Program contract #2022-22072200006, and in part by the Defense Advanced Research Projects Agency (DARPA) under Agreement No. HR00112490374. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ODNI, IARPA, DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation therein.

References

Nikolay Babakov, David Dale, Varvara Logacheva, and Alexander Panchenko. 2022. A large-scale computational study of content preservation measures for text style transfer and paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 300–321, Dublin, Ireland. Association for Computational Linguistics.

Haohan Bo, Steven H. H. Ding, Benjamin C. M. Fung, and Farkhund Iqbal. 2021. ER-AE: Differentially private text generation for authorship anonymization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3997–4007, Online. Association for Computational Linguistics.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. *Preprint*, arXiv:2401.01335.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Pro-*

ceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 427–436, Montréal, Canada. Association for Computational Linguistics.

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, 13(40):1159–1187.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 224–233, Honolulu, Hawaii. Association for Computational Linguistics.

Leshem Choshen, Lior Fox, Zohar Aizenbud, and Omri Abend. 2020. On the weaknesses of reinforcement learning for neural machine translation. In *International Conference on Learning Representations*.

David Dale, Anton Voronov, Daryna Dementieva, Varvara Logacheva, Olga Kozlova, Nikita Semenov, and Alexander Panchenko. 2021. Text detoxification using large pre-trained neural models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7979–7996, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hongyu Gong, Suma Bhat, Lingfei Wu, JinJun Xiong, and Wen-mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus. 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 3168–3180, Minneapolis, Minnesota. Association for Computational Linguistics.

Skyler Hallinan, Faeze Brahman, Ximing Lu, Jaehun Jung, Sean Welleck, and Yejin Choi. 2023a. STEER: Unified style transfer with expert reinforcement. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7546–7562, Singapore. Association for Computational Linguistics.

Skyler Hallinan, Alisa Liu, Yejin Choi, and Maarten Sap. 2023b. Detoxifying text with MaRCo: Controllable revision with experts and anti-experts. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–242, Toronto, Canada. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, Lisa Orii, and Peter Szolovits. 2020. Hooks in the headline: Learning to generate headlines with controlled styles. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5082–5093, Online. Association for Computational Linguistics.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Samuel Kiegeland and Julia Kreutzer. 2021. Revisiting the weaknesses of reinforcement learning for neural machine translation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1673–1681, Online. Association for Computational Linguistics.

Zae Myung Kim, Wanyu Du, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. 2022. Improving iterative text revision by learning where to edit from other revision tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9986–9999, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kalpesh Krishna, John Wieting, and Mohit Iyyer. 2020. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, Online. Association for Computational Linguistics.

Philippe Laban, Tobias Schnabel, Paul Bennett, and Marti A. Hearst. 2021. Keep it simple: Unsupervised simplification of multi-paragraph text. 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 6365–6378, Online. Association for Computational Linguistics.

Guillaume Lample, Sandeep Subramanian, Eric Smith, Ludovic Denoyer, Marc'Aurelio Ranzato, and Y-Lan Boureau. 2019. Multiple-attribute text rewriting. In

International Conference on Learning Representations.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021a. DExperts: Decoding-time controlled text generation with experts and anti-experts. 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 6691–6706, Online. Association for Computational Linguistics.

Ao Liu, An Wang, and Naoaki Okazaki. 2022. Semi-supervised formality style transfer with consistency training. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4689–4701, Dublin, Ireland. Association for Computational Linguistics.

Dayiheng Liu, Jie Fu, Yidan Zhang, Christopher Joseph Pal, and Jiancheng Lv. 2019a. Revision in continuous space: Unsupervised text style transfer without adversarial learning. In *AAAI Conference on Artificial Intelligence*.

Shuai Liu, Shantanu Agarwal, and Jonathan May. 2024. Authorship style transfer with policy optimization. *Preprint*, arXiv:2403.08043.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Yixin Liu, Graham Neubig, and John Wieting. 2021b. On learning text style transfer with direct rewards. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4262–4273, Online. Association for Computational Linguistics.

Ximing Lu, Sean Welleck, Jack Hessel, Liwei Jiang, Lianhui Qin, Peter West, Prithviraj Ammanabrolu, and Yejin Choi. 2022. QUARK: Controllable text generation with reinforced unlearning. In *Advances in Neural Information Processing Systems*.

Guoqing Luo, Yu Han, Lili Mou, and Mauajama Firdaus. 2023. Prompt-based editing for text style transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5740–5750, Singapore. Association for Computational Linguistics.

Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabas Poczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W Black, and Shrimai Prabhumoye. 2020. Politeness transfer: A tag and generate approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1869–1881, Online. Association for Computational Linguistics.

Eric Malmi, Aliaksei Severyn, and Sascha Rothe. 2020. Unsupervised text style transfer with padded masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8671–8680, Online. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *Preprint*, arXiv:2404.19733.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ajay Patel, Nicholas Andrews, and Chris Callison-Burch. 2023. Low-resource authorship style transfer: Can non-famous authors be imitated? *Preprint*, arXiv:2212.08986.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. CoEdIT: Text editing by task-specific instruction tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5274–5291, Singapore. Association for Computational Linguistics.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may I introduce the GYAFC dataset: Corpus, benchmarks and metrics for formality style transfer. 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 129–140, New Orleans, Louisiana. Association for Computational Linguistics.

Machel Reid and Victor Zhong. 2021. LEWIS: Levenshtein editing for unsupervised text style transfer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3932–3944, Online. Association for Computational Linguistics.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2022. A recipe for arbitrary text style transfer with large language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 837–848, Dublin, Ireland. Association for Computational Linguistics.

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.

Parker Riley, Noah Constant, Mandy Guo, Girish Kumar, David Uthus, and Zarana Parekh. 2021. TextSETTR: Few-shot text style extraction and tunable targeted restyling. 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 3786–3800, Online. Association for Computational Linguistics.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Mingyue Shang, Piji Li, Zhenxin Fu, Lidong Bing, Dongyan Zhao, Shuming Shi, and Rui Yan. 2019. Semi-supervised text style transfer: Cross projection in latent space. 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 4937–4946, Hong Kong, China. Association for Computational Linguistics.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. 2018. A4NT: Author attribute anonymity by adversarial training of neural machine translation. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1633–1650, Baltimore, MD. USENIX Association.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024a. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024b. Trial and error: Exploration-based trajectory optimization for llm agents. *Preprint*, arXiv:2403.02502.

Mirac Suzgun, Luke Melas-Kyriazi, and Dan Jurafsky. 2022. Prompt-and-rerank: A method for zero-shot and few-shot arbitrary textual style transfer with small language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2195–2222, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Yunli Wang, Yu Wu, Lili Mou, Zhoujun Li, and Wenhan Chao. 2019. Harnessing pre-trained neural networks with rules for formality style transfer. 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 3573–3578, Hong Kong, China. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. Transfertransfo: A transfer learning approach for neural network based conversational agents. *Preprint*, arXiv:1901.08149.

Wei Xiong, Hanze Dong, Chenlu Ye, Ziqi Wang, Han Zhong, Heng Ji, Nan Jiang, and Tong Zhang. 2023. Iterative preference learning from human feedback: Bridging theory and practice for rlhf under kl-constraint. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024a. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *Preprint*, arXiv:2401.08417.

Jing Xu, Andrew Lee, Sainbayar Sukhbaatar, and Jason Weston. 2024b. Some things are more cringe than others: Iterative preference optimization with the pairwise cringe loss. *Preprint*, arXiv:2312.16682.

Ruochen Xu, Tao Ge, and Furu Wei. 2019. Formality style transfer with hybrid textual annotations. *Preprint*, arXiv:1903.06353.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Yi Zhang, Tao Ge, and Xu Sun. 2020. Parallel data augmentation for formality style transfer. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3221–3228, Online. Association for Computational Linguistics.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China. Coling 2010 Organizing Committee.

A More Experimental Results

A.1 Alternative Automatic Evaluation

To show that the models trained with STAMP are not overfitted on the rewards used during training, we evaluate all models on a set of alternative metrics. Specifically, for TSS, we train a style classifier on the same data using a different base model, BERT-base-cased (Devlin et al., 2019); for MS, we use a different semantic similarity model MIS[20] (Babakov et al., 2022); for F, we use a different off-the-shelf classifier[21] trained on the CoLA dataset (Warstadt et al., 2019).

The results are shown in Table 7, Table 8, and Table 9. In most cases, the alternative metrics and the main metrics agree on the top two models. Although several disagreements exist on the individual metrics, both sets of metrics agree that STAMP models have the best overall performance (highest aggregated score).

A.2 Perturbed Hyperparameters

Although the STAMP training pipeline contains multiple steps of model training and data generation, it is robust to different datasets and commonly used hyperparameters. In Table 1 and Table 2, we show that, using the same set of hyperparameters, STAMP works well on two different datasets for both in-domain and out-of-domain inputs, which confirms STAMP's generalizability to different datasets without further hyperparameter tuning. Furthermore, we train STAMP models on the two datasets with perturbed hyperparameters. Specifically, for all SFT components, we decrease the learning rate from 5e-5 to 2e-5 and double the batch size; for CPO, we double the learning rate and halve the batch size; for data generation, we change the decoding temperature for $D_{p\to t}$ and $D_{s\to t}$ to 0.7 and 0.5, respectively. The results in Table 7, Table 8, and Table 9 show that the STAMP models trained with the perturbed hyperparameters have slightly different individual metric scores but they consistently demonstrate better overall performance (aggregated score) than all baseline models.

A.3 More Human Evaluation Results

The raw scores from the human evaluation and the result of the t-test are shown in Table 10. No significant difference is found between any model pairs in TSS_h[22], but MS_h and F_h are generally consistent with the automatic evaluation results. Specifically, STAMP and GPT 5-shot transfer are significantly better than STEER in meaning similarity (MS), and STEER and GPT 5-shot transfer are significantly better than STAMP in fluency (F).

A.4 Subsets Automatic Evaluation Scores

The fine-grained automatic evaluation scores on each subset in CDS and GYAFC are shown in Table 11 to Table 16 and Table 17 to Table 22, respectively.

B More Implementation Details

B.1 Statistical Significance Test

We conduct a resampling paired t-test for the automatic evaluation results and an independent t-test for the human evaluation results. For the resampling paired t-test, we randomly select 10 subsets of 100 samples from the test set and perform a paired t-test on the mean scores of the subsets between each pair of models. For the independent t-test, we use all available samples from the human study without resampling.

[20]https://huggingface.co/s-nlp/Mutual_Implication_Score

[21]https://huggingface.co/textattack/distilbert-base-cased-CoLA

[22]which is expected since style classification is difficult for human annotators (Krishna et al., 2020; Hallinan et al., 2023a).

Approach	CDS						GYAFC					
	TSS		MS		F		TSS		MS		F	
	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.
GPT zero-shot	0.189	0.181	0.705	0.763	0.803	0.721	0.672	0.674	0.788	0.898	**0.968**	0.929
GPT 5-shot	0.199	0.197	0.735	**0.781**	0.805	0.748	0.667	0.669	0.800	0.896	0.965	0.923
STRAP	0.382	0.361	0.626	0.530	0.759	0.757	0.618	0.627	0.735	0.761	0.913	0.877
STEER	0.654	0.570	0.672	0.602	**0.905**	**0.897**	0.951	0.929	0.776	0.821	0.930	**0.932**
ASTRAPOP	0.542	0.513	0.600	0.498	0.755	0.737	0.783	0.788	0.734	0.767	0.924	0.872
STAMP	**0.746**	**0.665**	0.801	0.754	0.801	0.764	**0.958**	0.922	0.921	0.935	0.941	0.906
STAMP pert.	0.699	0.644	**0.821**	0.777	0.829	0.771	0.957	**0.934**	**0.930**	**0.938**	0.948	0.886

Table 7: Main vs. alternative TSS, MS, and F scores on in-domain inputs on the CDS and the GYAFC datasets. STAMP pert. refers to the STAMP model trained with perturbed hyperparameters. The best and the 2nd best scores in each column are shown in **bold** and underline, respectively.

Approach	CDS						GYAFC					
	TSS		MS		F		TSS		MS		F	
	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.
GPT zero-shot	0.246	0.227	0.657	0.818	0.855	0.777	0.672	0.663	0.752	0.816	**0.909**	0.895
GPT 5-shot	0.289	0.276	0.708	0.839	0.868	0.801	0.722	0.711	0.752	0.815	0.902	0.882
STRAP	0.426	0.413	0.629	0.624	0.810	0.798	0.692	0.690	0.689	0.642	0.852	0.849
STEER	0.654	0.589	0.706	0.741	**0.927**	**0.904**	0.850	0.822	0.734	0.714	0.875	**0.899**
ASTRAPOP	0.579	0.557	0.606	0.602	0.808	0.778	0.816	0.809	0.685	0.648	0.863	0.836
STAMP	**0.787**	**0.711**	0.816	0.840	0.877	0.825	**0.964**	0.917	0.864	0.853	0.827	0.814
STAMP pert.	0.695	0.647	**0.861**	**0.873**	0.903	0.821	**0.964**	**0.923**	**0.870**	**0.860**	0.829	0.816

Table 8: Main vs. alternative TSS, MS, and F scores on out-of-domain inputs on the CDS and the GYAFC datasets. STAMP pert. refers to the STAMP model trained with perturbed hyperparameters. The best and the 2nd best scores in each column are shown in **bold** and underline, respectively.

Approach	CDS				GYAFC			
	In-domain		Out-of-domain		In-domain		Out-of-domain	
	Main	Alt.	Main	Alt.	Main	Alt.	Main	Alt.
GPT zero-shot	0.104	0.095	0.138	0.139	0.489	0.554	0.455	0.486
GPT 5-shot	0.112	0.108	0.175	0.180	0.495	0.543	0.486	0.511
STRAP	0.158	0.127	0.194	0.187	0.409	0.412	0.402	0.375
STEER	0.395	0.304	0.426	0.392	0.686	0.711	0.544	0.527
ASTRAPOP	0.221	0.173	0.259	0.244	0.525	0.521	0.479	0.435
STAMP	**0.474**	**0.379**	**0.562**	**0.488**	0.828	**0.780**	0.687	0.637
STAMP pert.	0.469	0.378	0.538	0.458	**0.842**	0.773	**0.693**	**0.644**

Table 9: Main vs. alternative aggregated scores on the CDS and the GYAFC datasets. STAMP pert. refers to the STAMP model trained with perturbed hyperparameters. The best and the 2nd best scores in each column are shown in **bold** and underline, respectively.

B.2 Hyperparameters

We sample same-sized training and validation subsets for CDS and GYAFC, and use the same hyperparameters to train STAMP on the two datasets to reduce the cost for more hyperparameter searching. We list all hyperparameters for STAMP in Table 23, Table 24, Table 25, Table 31, and Table 32.

B.3 GPT prompt templates

We elaborate on the prompts used for GPT zero- and 5-shot style transfer on CDS and GYAFC in Table 26 and Table 27, respectively.

B.4 Hardware and Runtime

We train all components of STAMP using Nvidia A40-48GB GPUs. The number of GPUs and time used to train each model on each dataset are shown in Table 28. Furthermore, we calculate the total training time including SFT, CPO, and all data generation processes for STAMP and the strongest baseline STEER. The results are shown in Table 33. In general, STAMP is slower than STEER on GYAFC but faster on CDS. However, to ensure fairness, we compare STEER's runtime with the runtime required for STAMP to outperform

Approach	TSS_h	MS_h	F_h
GPT 5-shot	0.59	<u>1.48</u>	<u>1.79</u>
STEER	**0.69**	$1.24^‡$	**1.84**
STAMP	<u>0.64</u>	**1.48**	$1.57^‡$

Table 10: Raw human evaluation scores on in-domain inputs on the CDS datasets. The best and 2nd best scores in each column are shown in **bold** and <u>underline</u>, respectively. "‡" indicates a statistically significant difference ($p < 0.05$) between the top two models determined by independent sample t-test. No significant difference is found in any other model pairs.

	TSS	MS	F	Agg.
AAE Tweets	0.215	0.689	0.680	0.094
Bible	0.181	0.688	0.885	0.097
1810-1830 English	0.291	0.713	0.786	0.166
1890-1910 English	0.140	0.731	0.787	0.089
1990-2010 English	0.044	0.739	0.771	0.030
James Joyce	0.059	0.705	0.843	0.032
Lyrics	0.263	0.700	0.803	0.138
Romantic Poetry	0.119	0.604	0.848	0.050
Shakespeare	0.184	0.699	0.767	0.080
Switchboard	0.003	0.777	0.817	0.002
English Tweets	0.584	0.709	0.845	0.363
Overall	0.189	0.705	0.803	0.104

Table 11: The automatic evaluation results for GPT zero-shot on in-domain inputs on all subsets in CDS.

STEER and find that STAMP can achieve better performance than STEER in a much shorter time on both datasets (STEER vs. STAMP op. in Table 33), which indicates that STAMP is a more efficient training framework than STEER.

B.5 Human Evaluation Instructions

The instructions used in the human evaluation for all three objectives are shown in Table 30 including the questions asked and the detailed explanation for each level in the Likert scale.

C Scientific Artifacts

C.1 Use of Existing Artifacts

The existing artifacts used in this work and their licenses are listed in Table 29. Our use of the existing artifacts is consistent with their intended use specificed by their licenses.

C.2 Created Artifacts

We create a new text style transfer training framework, STAMP, and release the code under the MIT license. Considering ethical implications, STAMP

	TSS	MS	F	Agg.
AAE Tweets	0.297	0.711	0.649	0.126
Bible	0.166	0.689	0.865	0.086
1810-1830 English	0.249	0.742	0.815	0.154
1890-1910 English	0.154	0.784	0.819	0.106
1990-2010 English	0.181	0.753	0.875	0.130
James Joyce	0.061	0.748	0.819	0.034
Lyrics	0.256	0.738	0.808	0.138
Romantic Poetry	0.118	0.639	0.844	0.047
Shakespeare	0.169	0.704	0.794	0.077
Switchboard	0.179	0.829	0.774	0.114
English Tweets	0.355	0.749	0.797	0.218
Overall	0.199	0.735	0.805	0.112

Table 12: The automatic evaluation results for GPT 5-shot on in-domain inputs on all subsets in CDS.

	TSS	MS	F	Agg.
AAE Tweets	0.248	0.670	0.696	0.094
Bible	0.482	0.373	0.811	0.107
1810-1830 English	0.255	0.674	0.806	0.135
1890-1910 English	0.203	0.686	0.850	0.123
1990-2010 English	0.263	0.689	0.881	0.166
James Joyce	0.376	0.671	0.747	0.175
Lyrics	0.459	0.668	0.791	0.233
Romantic Poetry	0.558	0.607	0.623	0.177
Shakespeare	0.421	0.508	0.680	0.112
Switchboard	0.713	0.659	0.657	0.293
English Tweets	0.223	0.676	0.810	0.123
Overall	0.382	0.626	0.759	0.158

Table 13: The automatic evaluation results for STRAP on in-domain inputs on all subsets in CDS.

	TSS	MS	F	Agg.
AAE Tweets	0.651	0.665	0.908	0.387
Bible	0.496	0.597	0.901	0.248
1810-1830 English	0.642	0.688	0.884	0.389
1890-1910 English	0.396	0.675	0.929	0.252
1990-2010 English	0.945	0.683	0.937	0.606
James Joyce	0.671	0.712	0.882	0.415
Lyrics	0.704	0.673	0.915	0.429
Romantic Poetry	0.725	0.675	0.889	0.431
Shakespeare	0.366	0.683	0.868	0.203
Switchboard	0.902	0.664	0.909	0.543
English Tweets	0.700	0.675	0.933	0.439
Overall	0.654	0.672	0.905	0.395

Table 14: The automatic evaluation results for STEER on in-domain inputs on all subsets in CDS.

	TSS	MS	F	Agg.
AAE Tweets	0.431	0.651	0.648	0.164
Bible	0.736	0.273	0.793	0.137
1810-1830 English	0.401	0.659	0.823	0.212
1890-1910 English	0.263	0.679	0.879	0.159
1990-2010 English	0.508	0.684	0.897	0.316
James Joyce	0.472	0.668	0.754	0.224
Lyrics	0.628	0.637	0.820	0.317
Romantic Poetry	0.807	0.595	0.583	0.266
Shakespeare	0.602	0.460	0.636	0.152
Switchboard	0.837	0.625	0.656	0.334
English Tweets	0.275	0.673	0.810	0.153
Overall	0.542	0.600	0.755	0.221

Table 15: The automatic evaluation results for ASTRAPOP on in-domain inputs on all subsets in CDS.

	TSS	MS	F	Agg.
AAE Tweets	0.806	0.889	0.788	0.561
Bible	0.643	0.640	0.830	0.312
1810-1830 English	0.764	0.799	0.807	0.490
1890-1910 English	0.439	0.812	0.875	0.311
1990-2010 English	0.920	0.819	0.873	0.660
James Joyce	0.844	0.859	0.825	0.596
Lyrics	0.545	0.806	0.815	0.357
Romantic Poetry	0.776	0.806	0.766	0.470
Shakespeare	0.740	0.792	0.686	0.392
Switchboard	0.920	0.811	0.721	0.534
English Tweets	0.810	0.784	0.831	0.529
Overall	0.746	0.801	0.801	0.474

Table 16: The automatic evaluation results for STAMP on in-domain inputs on all subsets in CDS.

	TSS	MS	F	Agg.
Formal	0.975	0.725	0.962	0.680
Informal	0.368	0.851	0.974	0.298
Overall	0.672	0.788	0.968	0.489

Table 17: The automatic evaluation results for GPT zero-shot on in-domain inputs on all subsets in GYAFC.

	TSS	MS	F	Agg.
Formal	0.974	0.745	0.959	0.696
Informal	0.360	0.855	0.971	0.293
Overall	0.667	0.800	0.965	0.495

Table 18: The automatic evaluation results for GPT 5-shot on in-domain inputs on all subsets in GYAFC.

	TSS	MS	F	Agg.
Formal	0.799	0.722	0.931	0.535
Informal	0.438	0.750	0.896	0.283
Overall	0.618	0.736	0.913	0.409

Table 19: The automatic evaluation results for STRAP on in-domain inputs on all subsets in GYAFC.

	TSS	MS	F	Agg.
Formal	0.972	0.734	0.939	0.673
Informal	0.931	0.817	0.921	0.699
Overall	0.951	0.776	0.930	0.686

Table 20: The automatic evaluation results for STEER on in-domain inputs on all subsets in GYAFC.

	TSS	MS	F	Agg.
Formal	0.918	0.717	0.950	0.627
Informal	0.648	0.750	0.897	0.423
Overall	0.783	0.734	0.924	0.525

Table 21: The automatic evaluation results for ASTRAPOP on in-domain inputs on all subsets in GYAFC.

	TSS	MS	F	Agg.
Formal	0.963	0.858	0.953	0.788
Informal	0.953	0.984	0.928	0.870
Overall	0.958	0.921	0.941	0.828

Table 22: The automatic evaluation results for STAMP on in-domain inputs on all subsets in GYAFC.

Parameter	f_{cls}	f_{para}	$f_{p\to t}$	$f_{s\to t}$
learning rate	5e-5	5e-5	5e-5	5e-5
batch size	32	32	8	16
# epochs	6	10	6	12

Table 23: Training hyperparameters for all supervised fine-tuned models.

Parameter	f_{PO}
learning rate	2e-6
β	0.1
batch size	32
# epochs	16
k_{PO}	10
N_{iter}	10

Table 24: Training hyperparameters for iterative preference optimization.

Parameter	
target modules	q_proj, v_proj
rank	16
α	32
dropout	0.05

Table 25: LoRA Hyperparameters.

Zero-shot	`Rewrite the following sentence into the style of [target style].`
	`Original Sentence: [input text]`
	`Rewritten Sentence:`

5-shot	`Here are some examples of sentences in the style of [target style]:`
	`[example 1]`
	`......`
	`[example 5]`
	`Rewrite the following sentence into the style of [target style].`
	`Original Sentence: [input text]`
	`Rewritten Sentence:`

Table 26: GPT zero- and 5-shot prompts for style transfer on CDS.

Zero-shot	`Rewrite the following sentence in a(n) (in)formal style.`
	`Original Sentence: [input text]`
	`Rewritten Sentence:`

5-shot	`Here are some examples of sentences in a(n) (in)formal style:`
	`[example 1]`
	`......`
	`[example 5]`
	`Rewrite the following sentence in a(n) (in)formal style.`
	`Original Sentence: [input text]`
	`Rewritten Sentence:`

Table 27: GPT zero- and 5-shot prompts for style transfer on GYAFC.

	ParaNMT	CDS				GYAFC			
	f_{para}	f_{cls}	$f_{p \to t}$	$f_{s \to t}$	f_{PO}	f_{cls}	$f_{p \to t}$	$f_{s \to t}$	f_{PO}
# GPUs (A40s)	×2	×2	×2	×2	×4	×2	×2	×2	×2
Times (hrs)	3.4	0.4	1.1	1.0	35.2	0.1	0.2	0.2	7.4

Table 28: Training hardware and runtime for each component in STAMP on CDS and GYAFC.

Type	Name	License
Dataset	CDS: Corpus of Diverse Styles	MIT
	GYAFC: Grammarly's Yahoo Answers Formality Corpus	Custom (research-only)
Model	LLaMA-2-7B (6.7B)	Meta
	GPT-3.5-turbo-0125 (-)	MIT
	RoBERTa-large (355M)	MIT
	RoBERTa-large CoLA Classifier (355M)	MIT
	SBERT all-mpnet-base-v2 (109M)	Apache-2.0
Library	Transformers	Apache-2.0
	PEFT	Apache-2.0
	TRL	Apache-2.0
	Sentence Transformers	Apache-2.0

Table 29: Datasets, models, and software libraries used in this work. The number of parameters of each model is indicated in the parentheses next to the model name.

TSS_h	Question	Based on the examples above, what is the style of the following text?
MS_h	Similar	Most of the meaning (75% or more) of the two passages is the same.
	Somewhat Similar	Large portions (50-75%) of the passages are the same, but there are significant sections that differ or are present in only one passage.
	Not Similar	Only small portions (less than 50%) of the passages are the same.
	Question	How similar are the following two texts?
F_h	Fluent	Very clear, grammatical english (need not be formal); the meaning of the sentence is well understood. A small number of errors are ok.
	Somewhat Fluent	There are grammatical errors, possibly numerous, but the meaning can be understood.
	Not Fluent	The grammatical errors make it very difficult to understand the meaning.
	Question	How fluent is the following text?

Table 30: Instructions used in the human evaluation.

Parameter	$D_{p\to t}$	$D_{s\to t}$	D_{PO}
top p	1.0	1.0	1.0
temperature	0.5	0.7	1.0
$k_{\text{para/sft/po}}$	20	90	10
$\tau_{textMS/max}$	-	8	6

Table 31: Generation hyperparameters for dataset construction.

Parameter	Evaluation
top p	1.0
temperature	0.7

Table 32: Generation hyperparameters for dataset evaluation.

	CDS	GYAFC
STEER	52.0 hrs × 4 A40s	7.2 hrs × 2 A40s
STAMP op.	22.2 hrs × 4 A40s	5.9 hrs × 2 A40s
STAMP	43.2 hrs × 4 A40s	10.8 hrs × 2 A40s

Table 33: Total runtime (including dataset generation and training) for our reproduction of STEER and STAMP on CDS and GYAFC using identical models and architecture. STAMP op. indicates the training runtime point at which STAMP outperforms STEER.

is only intended for research purposes, which is compatible with the original access conditions of all existing artifacts used in STAMP.