Controlled Text Generation as Continuous Optimization with Multiple Constraints

Sachin Kumar[♣] **Eric Malmi**[◇] **Aliaksei Severyn**[◇] **Yulia Tsvetkov**[♠]
[♣]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
[◇]Google Research
[♠]Paul G. Allen School of Computer Science & Engineering, University of Washington
sachink@cs.cmu.edu, {emalmi, severyn}@google.com, yuliats@cs.washington.edu

Abstract

As large-scale language model pretraining pushes the state-of-the-art in text generation, recent work has turned to controlling attributes of the text such models generate. While modifying the pretrained models via fine-tuning remains the popular approach, it incurs a significant computational cost and can be infeasible due to lack of appropriate data. As an alternative, we propose MUCOCO—a flexible and modular algorithm for controllable inference from pretrained models. We formulate the decoding process as an optimization problem which allows for multiple attributes we aim to control to be easily incorporated as differentiable constraints to the optimization. By relaxing this discrete optimization to a continuous one, we make use of Lagrangian multipliers and gradient-descent based techniques to generate the desired text. We evaluate our approach on controllable machine translation and style transfer with multiple sentence-level attributes and observe significant improvements over baselines.

1 Introduction

Recent advances in language models [11, 8, 48] trained on large-scale web text corpora have led to great improvements in state-of-the-art on many natural language processing (NLP) tasks including the ability to generate increasingly coherent text [4]. However, once such models are trained, they are prone to degeneration [63] and biased, non-factual outputs [15, 41] as it is difficult to control the characteristics or attributes of the generated text without architectural modifications [25, 27, 33] and fine-tuning the models on attribute-specific corpora [28, 7]. This can be even more challenging if multiple attributes are involved as labeled data for each combination of attributes can be difficult to obtain.

We focus on *controlled* text generation where the goal is to decode from a text generation model such that the outputs satisfy certain constraints, which the model was not necessarily trained on. For example, given a dialogue generation model, additionally constraining the generated responses to be polite, although the model was not optimized for politeness during training. Recent works address this problem with left-to-right autoregressive decoding, and modify the vocabulary distribution at every step directly using attribute classifier probabilities [69, 37], or indirectly via backpropagating gradients through model activations [9]. While exhibiting high level of attribute control, by design these methods can only work with categorical attributes (typically only one attribute) and condition only on the left context while decoding. Additionally, they often require several heuristics to work and are prone to adversarial outputs [61].

To address these concerns, we propose the following decoding algorithm. Given a pretrained language model, we posit decoding from it as an optimization problem. First, we relax this discrete optimization problem to a continuous one by representing each token as a simplex on the target vocabulary [18].

Preprint. Under review.

This allows to use continuous optimization techniques like gradient-descent considering each token distribution as parameters, and keeping the language model's parameters fixed (§2). Second, we represent each target attribute to control as a differentiable function. We formulate controllable decoding as a multi-objective optimization problem, with maximizing the log-probability of the language model as well as target attributes as objectives. To make this optimization feasible via gradient-descent, we repurpose it to a constraint optimization problem and solve the dual using the modified differential method of multipliers [44]. We call the algorithm MUCOCO, for incorporating **mu**ltiple **co**nstraints through **c**ontinuous **o**ptimization.

We validate MUCOCO on three conditional text generation tasks with different types of sentence level constraints: (1) Adding formality and cross-lingual similarity in a machine translation model; (2) Ensuring transfer and content-preservation in a style-transfer model, and finally (3) Incorporating multiple styles and attributes (e.g., formality, sentiment magnitude, writer's age group) in a paraphrasing model. With automatic as well as human evaluations we find that our proposed method outperforms strong baselines.

2 MUCOCO: Constrained Decoding as Multi-Objective Optimization

For a given language generation task, let $\mathcal{G}$ model the conditional probability $p(\mathbf{y}|\mathbf{x})$ of the output sequence $\mathbf{y} = y_1, \ldots, y_n$, given the input sequence $\mathbf{x} = x_1, \ldots, x_n$. This model can be parameterized using any differentiable architecture like Transformers [59] or LSTMs [19] and trained with any loss function [12, 29], either autoregressively or non-autoregressively [16]. Traditionally, given an input $\mathbf{x}$, decoding from such a model requires finding the output sequence with the highest probability or the lowest negative log-probability, $\mathbf{y}^* = \arg\min_{\mathbf{y}\in\mathcal{Y}} -\log P(\mathbf{y}|\mathbf{x})$. Here $\mathcal{Y}$ is the set of all possible output sequences. In practice, searching $\mathcal{Y}$ to find the highest probability generation is intractable as the space of possible sequences grows exponentially with sequence length and has also been shown to produce undesirable solutions [56]. In most prior work, simple heuristics like beam search, or sampling have been adopted to find approximate solutions, where the text is generated one token at a time (usually left to right) with the output of step t being fed to the input at step $t+1$.

In this work, however, given $\mathcal{G}$ and an input sequence $\mathbf{x}$, we are interested in finding an output sequence $\mathbf{y}$ that not only maximizes the output probability but also optimizes multiple objectives defined over $\mathbf{x}$ and $\mathbf{y}$. More formally, we seek to find a $\mathbf{y}$ that minimizes all of the following objectives

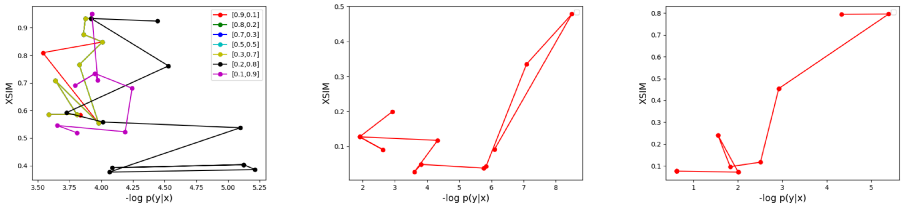
$$\mathbf{y}^* = \arg\min_{\mathbf{y}\in\mathcal{Y}}(-\log p(\mathbf{y}|\mathbf{x}), f_1(\mathbf{y}), \ldots, f_u(\mathbf{y}), g_1(\mathbf{x},\mathbf{y}), \ldots, g_v(\mathbf{x},\mathbf{y})) \tag{1}$$

Here each f_i is a function defined over the output sequence $\mathbf{y}$, for example, the negative log-probability of an attribute (e.g., formality) classifier we want the output sequence to satisfy. And each g_j is a function defined over both the input and output sequence, for example, semantic similarity between $\mathbf{x}$ and $\mathbf{y}$ [51]. We assume all f_i and g_j are differentiable. This is a multi-objective optimization with several possible solutions.

Since there are many objectives to minimize, a left-to-right decoding strategy like beam search or sampling will simply not work due to several reasons. First, the objectives f_i and g_j are sentence-level and hard to define accurately only on generated left-context [69, 37]. Even if we are able to define them, as we add more objectives this process becomes very computationally expensive. Following prior work [18, 46], we formulate this as a continuous optimization process instead of a standard discrete one, and then use standard algorithms for continuous optimization (like gradient descent) for decoding. We maintain a soft-representation of the sequence $\mathbf{y}$, $\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)$, where each $\tilde{y}_k \in \Delta_V$ is a simplex over the target vocabulary of size V, representing the probability of the k-th token. To decode a sentence, we initialize each $\tilde{y}_i$ uniformly over V, and treat the entire output sentence as the parameters for gradient descent keeping the parameters of $\mathcal{G}, f_i, g_j$ fixed. After gradient descent has converged, we generate discrete text by selecting the token with the highest probability in $\tilde{y}_k$. We provide more details on the optimization procedure in §2.2.

To make optimization feasible, a multi-objective problem generally yields itself to the following formulation:

$$\arg\min_{y} -\alpha \log p(\mathbf{y}|\mathbf{x}) + \sum_{i=1}^{u} \lambda_i f_i(\mathbf{y}) + \sum_{j=1}^{v} \mu_j g_j(\mathbf{x},\mathbf{y}), \tag{2}$$



(a) Linear combinations. (b) MUCOCO without damp. (c) MUCOCO

Figure 1: Loss curves for gradient descent for different configurations for an example of machine translation with a cross-lingual semantic similarity constraint (XSIM < 0.15). For each experiment, we do 100 steps of gradient descent (for clarity, we plot the loss values for every 10 steps). See §3.2 for detailed results. Left: In all cases one of the objectives is favored while the other fails to decrease. Middle: We observe fluctuations in the two losses. Right: The losses decrease much more smoothly leading to a better minimum.

for some statically or dynamically computed weights λ_i and μ_j for each i and j, where $\alpha + \sum_i \lambda_i + \sum_j \mu_j = 1$. Although this weighted summation formulation is intuitively appealing, it typically requires an expensive grid-search over the various scalings or use of a heuristic [24, 6, 17]. Furthermore, this formulation by definition assumes a trade-off between the different objectives by essentially assigning an importance weight to each of them. This problem is further exacerbated when different objectives have widely varying scales[1] with smaller scale objectives just getting ignored. More concretely, a multi-objective formulation as we define in (1) admits several possible "optimal" solutions also known as the Pareto set [10]. The image of the Pareto set is called the Pareto front. Since we define all objectives using neural networks, the Pareto front in our case is non-convex, where linear combinations of objectives are shown to be unsuccessful in finding good solutions [34, 35, 1] (see figure 1 for an example).

Ideally, our goal is a tunable optimization algorithm that finds solution on the Pareto front, i.e., every solution on the Pareto front should have a hyperparameter value for which the optimization algorithm finds that solution. In order to achieve this, we reframe our optimization problem as a Lagrangian optimization problem instead. We choose one of the losses as the primary objective and consider other losses as constraints. The goal is to minimize the primary loss subject to the secondary losses, each below a threshold value. More formally,

$$\begin{aligned} &\arg\min_{\mathbf{y}} -\log P(\mathbf{y}|\mathbf{x}) \text{ subject to} \\ &\quad f_i(\mathbf{y}) \leq \epsilon_i, i \in \{1, \cdots, u\} \\ &\quad g_j(\mathbf{x}, \mathbf{y}) \leq \xi_j, j \in \{1, \cdots, v\}. \end{aligned}$$

Here ϵ_i and ξ_j are tunable hyperparameters whose values' change can result in different solutions on the Pareto front. This formulation leads to an intuitive interpretation of the decoding process that the generated text from the model $\mathcal{G}$ should satisfy the constraints while being as faithful to the primary objective as much as possible.[2] Consequently, the Lagrangian we end up with looks similar to our original total loss linearly combined as in (2) given by

$$\mathcal{L}(y, \lambda_1, \ldots, \lambda_u, \mu_1, ...\mu_v) = -\log p(\mathbf{y}|\mathbf{x}) - \sum_{i=1}^{u} \lambda_i(\epsilon_i - f_i(y)) - \sum_{j=1}^{v} \mu_j(\xi_j - g_j(x, y)) \quad (3)$$

where λ_i, μ_j are Lagrange multipliers, and an optimal output $\mathbf{y}^*$ can be obtained as $\mathbf{y}^* = \arg\min_{\mathbf{y}} \max_{\lambda_i \geq 0, \mu_i \geq 0} \mathcal{L}(\mathbf{y}, \lambda_i, \mu_i)$. However, the traditional method of solving the dual function to find λ_i, μ_j that matches ϵ_i, ξ_j, respectively, again leads to a linear trade-off between the various objectives. When the Pareto front is non-convex as in our case, with gradient-descent, the constraints can be ignored and we still cannot always find optimal solutions by tuning ϵ_i, ξ_j [44].

[1] For example, classifier log-probabilities are in $(0, \inf)$ while sentence similarities usually lie in (0,1).

[2] For example, defining $f_i(\mathbf{y}) = p(a|\mathbf{y})$ as the probability of a desired attribute a in $\mathbf{y}$ leads to a natural threshold of $f_i(\mathbf{y}) > 0.5$. For a well-calibrated f_i, an even higher threshold could be used for inducing highly indicative features of a in $\mathbf{y}$.

2.1 Modified Differential Method of Multipliers

The fundamental issue in both linear combination of objectives and solving the dual is that fixed scalings λ_i and μ_i (manually pre-determined or obtained by solving the dual) do not work well with gradient descent to minimize for $\mathbf{y}$. Following prior work on differential method of multipliers [44], we propose to use a single gradient descent to optimize for both Lagrangian multipliers and $\mathbf{y}$ simultaneously as follows:

$$\mathbf{y}^{(t)} = \mathbf{y}^{(t-1)} - \eta_1 \nabla_{\mathbf{y}} \mathcal{L}, \lambda_i^t = \lambda_i^{t-1} + \eta_2 \nabla_{\lambda_i} \mathcal{L}, \mu_i^t = \mu_i^{t-1} + \eta_2 \nabla_{\mu_i} \mathcal{L}. \tag{4}$$

We follow the gradient of $\mathcal{L}$ downwards for the $\mathbf{y}$ (descent) and upwards for the multipliers (ascent) while making sure that the multipliers remain positive (by setting the multipliers to 0 whenever they become negative). Intuitively, this algorithm works by increasing the value of the multiplier with each gradient step as long as the constraint is violated. But when the constraint is suddenly satisfied and the multiplier is still large, it might take a number of gradient steps before the gradient descent pushes it to 0, thus causing the solution to be pushed further away from the constraint. As soon as the multipliers become 0 (or negative), the constraint is ignored and the process continues. However when the optimization hits the constraint again, this whole cycle repeats, resulting in "oscillations". We introduce a dampening parameter to each of the multipliers to reduce these oscillations (again following Platt and Barr [44]) and update the Lagrangian as follows:

$$\mathcal{L}(\mathbf{y}, \lambda_i, \mu_j) = -\log p(\mathbf{y}|\mathbf{x}) - \sum_{i=1}^{u} (\lambda_i - \zeta_i)(\epsilon_i - f_i(\mathbf{y})) - \sum_{j=1}^{v} (\mu_j - \nu_j)(\xi_j - g_j(\mathbf{x}, \mathbf{y})), \tag{5}$$

where $\zeta_i = d * \texttt{stop-gradient}(\epsilon_i - f_i(y))$, $\nu_j = d * \texttt{stop-gradient}(\mu_j - g_j(x, y))$ and d is a hyperparameter. d does not affect the final $\mathbf{y}$, just how quickly the algorithm converges to it (We use $d = 1$ in all experiments). $\texttt{stop-gradient}(\cdot)$ indicates that the argument is detached from the computational graph and does not contribute to the gradient computation. When a constraint is not satisfied ($\epsilon_i - f_i(\mathbf{y}) < 0$, hence $\zeta_i < 0$), the dampening parameter ζ_i being negative incurs higher penalty on the violation than when not using any dampening, without actually increasing the value of λ_i too much. But when the constraint is satisfied, it helps quickly reduce the value of penalty being incurred on the constraint while the multiplier converges to 0.

2.2 Optimization: Exponentiated Gradient Descent

Our goal is to generate a sequence of discrete symbols $\mathbf{y} = y_1, \ldots, y_T$, where y_k is from the target vocabulary. To make continuous optimization like gradient descent feasible, we adopt a soft-relaxation [18] to represent each y_k as a probability simplex, $\tilde{y}_k \in \Delta_V$ (i.e. $0 \leq \tilde{y}_{kl} \leq 1$ and $\sum_{l=1}^{|V|} \tilde{y}_{kl} = 1$). Intuitively, it gives the probability of each token in the vocabulary. To compute the loss $\mathcal{L}$ during forward pass, we first convert $\tilde{y}_k$ to a one-hot vector $\hat{y}_k$ via a straight through estimator [3]. This allows gradients to be applied to $\tilde{y}_k$ during the backward pass. More formally, $\hat{y}_k = \texttt{one-hot}(\arg\max \tilde{y}_k) - \texttt{stop-gradient}(\tilde{y}_k) + \tilde{y}_k$. During the forward pass, the input embedding tables corresponding to $\mathcal{G}$ and each of the constraints' models receive a one-hot vector $\hat{y}_k$ at each step, and the input embedding is computed as a weighted-sum of the embedding weights. But in the backward pass, the gradients are applied to $\tilde{y}_k$.[3]

This relaxation, however, adds another constraint to the objective $\mathcal{L}$ that each parameter $\tilde{y}_k$ should be a simplex. We use exponentiated gradient descent [26, 18] to solve this problem which modifies the gradient-descent update shown in (4) as: $\tilde{y}_k^{(t)} \propto \tilde{y}_k^{(t-1)} \exp(-\eta_1 \nabla_{\tilde{y}_k} \mathcal{L})$. After every descent step, $\tilde{y}_k^{(t)}$ is normalized to make it a simplex.

2.3 Preventing adversarial solutions: Annealing the thresholds

Finally, it is well known that most neural network based models are not robust to noise and in fact gradient-based methods have been used to generate adversarial examples for text classifiers [55]. We find in our early experiments that using these models to define constraints can also lead to such cases where the constraints are rapidly satisfied but the generated sentences are disfluent. To prevent

[3] Unlike prior work [18, 46, 55], we do not feed $\tilde{y}_i$ directly to the model as in our early experiments we found that it leads to slow convergence.

this issue, we introduce an annealing schedule [43] during the gradient descent where we start with relaxed thresholds ϵ_i, ξ_j such that they are all satisfied and only the primary loss $-\log p(\mathbf{y}|\mathbf{x})$ is active. As the optimization progresses, we gradually decrease the value of the thresholds causing the constraints to get violated resulting in the optimization gradually shifting to updating $\mathbf{y}$ to satisfy them. The exact schedule we use is described in the next section.

The final decoding algorithm we use in all our experiments is described in the Appendix algorithm 1.

3 Experimental Setup

We evaluate MUCOCO on the following controlled generation tasks: reinforcing target style in text generated by a style transfer model §3.1 and adding formality to a machine translation model (§3.2). Additionally, we conduct a qualitative analysis of rewriting a product review to adhere to multiple expected attributes like formality, sentiment magnitude, and age group of the author (§4). These tasks include constraints corresponding to both expected attributes in the target sentence (like formality) as well as both source and target sentences (like semantic similarity) with up to 6 constraints per task.

Implementation Details For a given sentence length T, we initialize each simplex $\tilde{y}_1, \ldots, \tilde{y}_T$ uniformly over the vocabulary. We use exponentiated descent learning rate of $\eta_1 = 50$ for $\mathbf{y}$ and ascent learning rate of $\eta_2 = 2.0$ for the multipliers, and run the optimization for 100 steps. Given all intermediate solutions $\mathbf{y}^{(t)}$, we choose the one which satisfies the maximum number of constraints (if not all) and has the minimum value of the primary objective. For each constraint, we use the following annealing schedule: we start with an initial value and linearly decrease it at step 40 until it reaches the desired value at step 80, after which we keep it constant. Additionally, since the length of the target sequence is not known in advance, we first greedily decode from $\mathcal{G}$ till the end-of-sentence token is generated resulting in a sequence of length L. We then use our approach for each $T \in \{L-5, \ldots, L+5\}$ and choose the one which (a) satisfies all the constraints and (b) has the minimum value of the primary objective. If none or partial constraints are satisfied, we choose the output based on (b).

3.1 Style Transfer

We begin with a style-transfer task, a task aiming to faithfully and fluently rewrite a given sentence such that a desired writing style is reflected in the generation. This task has been widely studied [21, 54, 28, among others] and differs from related tasks like sentiment transfer [58, 57, 32] where flipping the sentiment usually comes at the cost of changing meaning.

Style transfer is usually evaluated across three dimensions: (1) does the output sentence conform to the expected style; (2) does the output sentence preserve the input's meaning; and (3) is the generated sentence fluent. Most prior work in style transfer focused on devising training objectives serving as proxy for the desired outcomes, for example, back-translation [45, 57] or paraphrasing [28] for content preservation and language modeling for style and fluency. But depending on training algorithm and available data, there is often an observed trade-off between transfer and content-preservation [45, 57]. To that end, we add the desired attributes via explicit constraints when decoding from an existing style transfer model.

More specifically, we consider the task of informal to formal transfer [49] with the state-of-the-art unsupervised model STRAP from Krishna et al. [28]. This model is trained in an unsupervised fashion by (1) generating a pseudo-parallel corpus by paraphrasing each formal sentence in the training set (which results in a demotion of stylistic attributes), and (2) training an inverse-paraphrase model to translate paraphrases back to the original formal style. At test time, given an informal input sentence $\mathbf{x}$, the model first generates its paraphrase $\mathbf{z}$, then using an inverse-paraphrase model to generate the output $\hat{\mathbf{y}}$. We train this model by fine-tuning GPT2 (345M) [47] with the GYAFC Corpus (Entertainment/Music domain; around 50K formal sentences) [49] and evaluate it on the provided test set containing 1312 informal sentences. Krishna et al. [28] report best results with greedy decoding. In MUCOCO we modify the decoding algorithm by considering the negative log-probability of $\mathbf{y}$ given $\mathbf{z}$ according to the model as the primary objective, and incorporate the following constraints:

Formality: We train a binary classifier $p_{\text{FORMAL}}(\mathbf{y})$ by fine-tuning GPT2 on the same GYAFC training corpus, following default hyperparameter choices provided in HuggingFace [67]. This classifier outputs the formality probability of a sentence $\mathbf{y}$. We add this output as a constraint to the decoder as

$-\log(p_{\text{FORMAL}}(\mathbf{y})) < -\log(0.5)$. In other words, the constraint is satisfied if the classifier assigns at least 0.5 probability of the output $\mathbf{y}$ being formal. We initialize the threshold to 10.0 which is later annealed to $-\log(0.5)$.

Semantic Similarity: Since the baseline style-transfer model takes as input the paraphrase $\mathbf{z}$ and not the original text $\mathbf{x}$, it is susceptible to losing some of the original content in $\mathbf{x}$ while generating $\mathbf{y}$. To ensure content preservation we incorporate two kinds of objectives: (1) $\text{USIM}(\mathbf{x}, \mathbf{y}) = \text{cosine}(M(x), M(y))$ [51] where M outputs a continuous vector representation of a given sentence. Similarity between $\mathbf{x}$ and $\mathbf{y}$ is measured by cosine similarity of their respective representations. (2) $\text{WMD}(\mathbf{x}, \mathbf{y})$ takes as input bags of word embeddings of the two sentences and computes the Word Mover's Distance between them [31]. This distance is computed by solving a linear program. We adapt the alternating optimization procedure described in [30] to make this loss differentiable through the program. Intuitively, while USIM computes similarity between sentences taking context into account, it can be less robust to certain missing or repeating tokens, whereas WMD measures lexical overlap between input sentences acting as a proxy for coverage. We discuss the two losses in more detail in Appendix C. To compute the thresholds for constrained optimization, we compute the average value of the two functions on the development set in the same corpus. We use $\text{USIM} \leq 0.15$ and $\text{WMD} \leq 0.4$ as the final constraints (with initial threshold values of 2.0 for each).

Baselines and Evaluation Metrics We compare MUCOCO with the following baselines:

NO-CONSTRAINTS: We decode directly from the model greedily without any constraints. This replicates the best result reported by Krishna et al. [28].

FUDGE: Introduced by Yang and Klein [69], this method decodes in an autoregressive manner. It modifies the output vocabulary distribution at every step by interpolating the language model probability with that of a formality classifier. This classifier is trained to predict the probability of entire sentence being formal given only a prefix (we train it similarly to $p_{\text{FORMAL}}(\mathbf{y})$ by fine-tuning GPT2). This method only works with categorical features like formality and is not extensible to constraints like semantic similarity. We decode using the hyperparameters recommended in [69].

Following the baseline model Krishna et al. [28], we evaluate the generated sentences with the following metrics: (a) **fluency** or grammatical wellformedness measured by the accuracy of a RoBERTa-based classifier model [38] trained on CoLA [62], averaged over all outputs, (b) **transfer**: measured by a RoBERTa-based classifier model [38] trained on the GYAFC training corpus, and finally (c) **WSIM** [65], a subword embedding based similarity model trained on a large-scale paraphrase corpus which performs well on STS benchmarks [5] as well. We measure this metric both with respect to the input and the provided references.[4] In addition, we also report USIM.

Results The style transfer results are summarized in table 1. If we only incorporate a formality constraint, we observe that compared to FUDGE our method significantly improves transfer accuracy at the expense of content preservation. Adding semantic similarity constraints on the other hand improves both transfer as well as content preservation with the largest gains achieved when all the constraints are considered together. Qualitative analysis shows that MUCOCO's outputs are typically more fluent and have stronger formality signals, but all of the models are prone to propagating errors from the paraphrasing model (see examples in the Appendix table 3).

3.2 Style-controlled Machine Translation

We now evaluate MUCOCO in the task of formality transfer in machine translation. Given a trained MT model, decoding is often done using beam search and the highest probability beam candidate is chosen as the final output. Prior work has explored adding rule-based or heuristic constraints such as length penalty or coverage [68] to rerank beam candidates, and adding lexical constraints like penalizing n-gram repetitions [20]. In this experiment, we target sentence-level constraints which are otherwise difficult to incorporate in a left-to-right decoding process. Given a trained MT model and the source text x, we use negative log-probability of the translation y under the MT model as our primary objective and incorporate the following constraints for decoding in different combinations:

Cross-lingual Similarity Similar to USIM, we define $\text{XSIM}(\mathbf{x}, \mathbf{y}) = \text{cosine}(CM(x), CM(y))$, where CM is a multilingual encoder trained by distilling a monolingual model like M described ear-

[4]Each input sentence has 4 references, we choose the highest textscwsim value to compute the average.

Method	Constraint	Fluency	Transfer	Content Preservation (w.r.t. input)		Content Preservation (w.r.t. ref)	
				WSIM	USIM	WSIM	USIM
STRAP	None	91%	78%	0.69	0.77	0.72	0.80
FUDGE	FORMAL(y)	90%	85%	0.71	0.77	0.73	0.81
MUCOCO	FORMAL(y)	89%	93%	0.67	0.75	0.72	0.78
MUCOCO	USIM(x, y)	92%	85%	0.71	0.78	0.74	0.81
MUCOCO	USIM(x, y), WMD(x, y)	92%	87%	**0.73**	**0.79**	**0.77**	**0.86**
MUCOCO	SIM(x, y), WMD(x, y), FORMAL(y)	**93%**	**92%**	0.71	**0.79**	0.75	0.84

Table 1: Automatic evaluation of fluency, formality transfer, and content preservation for informal-to-formal style transfer models.

lier [52]. More details of training are available in the Appendix C. Averaging across the development set, we use 0.2 as the threshold for the constraint.

Formality Unlike style transfer, where the goal is to rewrite text in the desired style, here we seek to generate translations in a desired style directly from an MT model which was not explicitly trained to conform to a specific style. We train a classifier $p_{\text{FORMAL}}(\mathbf{y})$ similarly to one described in previous section by fine-tuning GPT2, but with a different input-embedding table to match the vocabulary of the decoder of the MT model. Again, we use $\log p_{\text{FORMAL}}(\mathbf{y}) > \log(0.5)$ as the constraint.

Baselines and Evaluation Metrics We compare MUCOCO with the following two baselines:

BEAMSEARCH: We decode directly from the translation model with a beam search of size 5.

FUDGE [69]: defined similarly as in the style transfer task but trained to match the decoder vocabulary. As mentioned before, FUDGE only works with categorical attributes like formality and is not easily extensible to constraints like cross-lingual similarity. We use the recommended hyperparameters by Yang and Klein [69] for decoding.

In Yang and Klein [69], the authors also compare FUDGE with other baselines such as PPLM [9] and BEAMSEARCH followed by style transfer. They show that FUDGE vastly outperforms these baselines. Hence, we only show comparisons with FUDGE in this work. We evaluate along the following metrics: (a) **BLEU** [42]: a standard metric for evaluating MT, (b) **BERTScore** [72]: an embedding-based metric which is more robust to changes in surface forms of the words than BLEU. (b) **transfer**: the same RoBERTa-based formality classifier as in our style transfer experiments. We also report XSIM, the constraint we use for decoding.

We experiment with French to English translation with a subset of the OpenSubtitles test set [36] containing 1360 sentence pairs.[5] This test set contains informal spoken language for both source and target. For the primary objective, we use the Marian Transformer based French (fr) to English (en) model [23] through Huggingface. We summarize the results of this experiment in table 2 with selected examples in the Appendix table 4.

Results By just using a cross-lingual similarity metric without modifying the model at all, we observe +0.6 improvement in BLEU score as well as BERTScore. Adding a formality constraint leads to considerable gain in formality of the outputs with a drop in BLEU; using both XSIM and FORMAL helps recover some of the drop. The drop in BLEU is unsurprising: since BLEU is a surface-level metric it naturally penalizes the translations that are rephrased to conform to formality constraints. Indeed, as shown in table 4, adding a formality constraint leads to changes in sentence structure and vocabulary. On the other hand, we see improvements in BERTScore which is an embedding-based metric, more robust to paraphrasing.

To further validate our results, we conduct a human evaluation of the generated translations. We randomly sample 100 source sentences and their translations generated by beam search and MUCOCO with both FORMAL and XSIM constraints. Two annotators (highly proficient in French and English) to rank the translations on faithfulness (is the source meaning reflected in the translation?) and formality.

[5] We create this subset by filtering the original test set to contain only sentence pairs for which beam search translations are classified as informal.

Method	Constraint	BLEU	BertScore	Formality(%)	XSIM
BEAMSEARCH	None	42.1	0.932	0%	0.85
MUCOCO	XSIM($\mathbf{x}, \mathbf{y}$)	**42.7**	**0.939**	4%	**0.88**
FUDGE	FORMAL($\mathbf{y}$)	39.2	0.922	6%	0.83
MUCOCO	FORMAL($\mathbf{y}$)	37.5	0.913	**30%**	0.83
MUCOCO	FORMAL($\mathbf{y}$), XSIM($\mathbf{x}, \mathbf{y}$)	39.8	**0.935**	23%	**0.86**

Table 2: Results of style-controlled machine translation experiments.

The options are randomized. On the translation pairs where both annotators agree (79 out of 100), the ones generated by our method were favored by annotators 37% percent of the time, while beam search translations were favored only 18% of the time, and 21% translations were equally favored.

4 Discussion

Simultaneously controlling several attributes One of the main advantages of our proposed approach is its flexibility to introduce any number of constraints (as long as they are differentiable) to the decoding objective. To illustrate this advantage we consider the following problem: given a sentence annotated with following attributes: age group of the author, formality, and sentiment magnitude, rewrite it such that any chosen combination of the attributes are modified while keeping the others fixed and the content preserved [39, 57]. For our primary objective, we use a inverse-paraphrasing model as defined in §3.1 which we train on a corpus of Yelp Reviews[6] [45]. First, we paraphrase each sentence in the corpus as described in Krishna et al. [28] creating a pseudo-parallel corpus (of reviews and their paraphrases) and train $\mathcal{G}$ as an inverse-paraphrase model to translate the paraphrases back to the original reviews. We use USIM and WMD for semantic similarity constraints and three classifiers for (a) age group of the author (binary; < 30 years or > 30 years); (b) formality of the review (binary: informal or formal); (c) sentiment magnitude (five-class classifier ratings of 1 to 5). Here we focus on sentiment amplification rather than transfer. That is, changing the 4-star rating of an input to 5 (or 2 to 1). Details of the classifiers and the data used are provided in Appendix C.2.[7] Table 5 shows examples of generated sentences with different combinations of attribute values.

Finding other solutions on the Pareto front As described in §2, the thresholds ϵ, ξ are tunable hyperparameters that allow us to find different solutions on the Pareto front. In our experiments so far, based on expected outcomes and how the constraints are defined, we showed results with only one threshold for each constraint. For example, ideally for a well-calibrated text classifier based constraint, this technique should be able to find solutions for any probability as threshold, but most neural-network based classifiers are not well-calibrated and predict the highest probability output as the label, hence a natural threshold for binary-classifiers is a label probability > 0.5. In Appendix table 6, we show how the outputs change if we modify this threshold to different values. We observe that in most cases the optimization converges to generate words more commonly associated with formality. On the other hand, semantic similarity between two sentences is even harder to define, is less robust to noise, and varies with writing styles of the input sentences. As shown, increasing this threshold for semantic similarity can lead to repetitions and disfluency.

Speed and memory requirements The presented decoding algorithm treats each token in the output sequence $\mathbf{y}$ as a parameter for gradient-descent which involves multiple forward and backward passes through the primary generative model $\mathcal{G}$ as well as attribute models. Given an expected sequence length L, it optimizes $L \times V$ parameters which is both memory and time intensive compared to left-to-right decoding. For example, on a single GeForce RTX 2080 Ti (12GB) on which we run all presented experiments, with a batch size of 1, our approach takes approximately 90 minutes on average to decode around 1200 sentences compared to around 20 minutes for FUDGE [69] with a single constraint. For reference, unconstrained beam-search takes 2-5 minutes. Given enough GPU capacity, however, this approach can easily be extended to larger-batches to improve decoding speed. We do not conduct this experiment due to limited available resources. Using 16-bit floating point

[6] This corpus is sentence-tokenized and lowercased with 2.2M sentences not labeled for any attributes.

[7] Due to lack of an established benchmark for this task and due to many possible combinations of attributes, we do not report quantitative results.

operations, this can further be improved. Another way of improving memory efficiency would be to optimize not for tokens directly but instead optimize for token embeddings [29]. This formulation also removes the requirement for all the models to share a vocabulary. We plan to investigate this in future work. Finally, given the capability of this approach to incorporate multiple constraints, it can also be used to generate pseudo-parallel data with different attribute combinations which then could be used to train supervised models for attributes for interest resulting in faster models at inference.

Ethical considerations Controlled language generation is a growing research area, and state-of-the-art techniques are still noisy and not powerful enough to enable fine-grained control over generated content. In the current form, they have the potential to generate harmful and biased language. For example, language generators are prone to generating non-factual content [41], especially when used maliciously [60, 61, 71]. Moreover, when style transfer techniques are used in conjunction with users' personal attributes such as gender, they are likely to generalize and amplify harmful social biases [45]. We thus opted not to include gender transfer in our experiments. Our goal in this work is to enable finer-grained control over generated texts that could potentially alleviate these issues. These approaches can be used as a useful tool for mitigating many problematic biases already encoded in large language models [15, 2, 37], for anonymizing personal attributes [50], and even as aids for humans to avoid implicit biases in their writing [40, 14].

5 Related Work

Recent work on controllable text generation can be divided into two categories. The first focuses on directly training attribute-conditional models either through fine-tuning pretrained models with attribute-specific corpora [13, 28] or via training conditional generative networks [57, 73, 45, 70]. More broadly, this includes methods for text style transfer. Unlike MuCoCO, these methods are not easily extensible and require training or fine-tuning a new model to incorporate new attributes. For example, CTRL [25] train a large scale language model (1.6B parameters) from scratch with 55 control codes capable of generating high-quality text but is very expensive to train. The second line of work, in line with MuCoCO, aims to incorporate control in pre-trained models without retraining them. For example, GeDi [27] trains smaller class-conditional LMs and uses them as discriminators for guided generation. More recently, FUDGE [69] and DExpert [37] propose changes to left-to-right decoding in language models by modifying the vocabulary distribution at every step using attribute classifiers and ensemble of language models trained on attribute-specific corpora. Although lightweight, these approaches are, by design, prone to a trade-off between preserving content and enforcing the attributes in the generated text. Our work is most closely related to Plug and Play Language Models [9] which use gradients from the attribute models to update the prediction. They work by updating the model activations rather than token probabilities which limits their applicability to only unconditional language models. Furthermore, due to their autoregressive nature, these approaches do not guarantee sequence-level control as they only look at the prefix generated up to a certain step. These are also limited to categorical attributes and can not enforce real-valued controls like semantic similarity.

Gradient-descent based optimization to generate text has been explored in prior work for improving machine translation [18], paraphrasing [46] and generating adversarial examples [55]. These methods however rely on linear combinations of various objectives which as we discuss in §2 are not optimal for non-convex neural-network based models. This phenomenon has also been studied in multi-task learning [34, 35, 53] where linear combination of multiple task losses is the most common approach and approaches for multi-objective gradient descent have been proposed. These approaches can also be explored for text generation in the future.

6 Conclusion

We present MuCoCO, a decoding algorithm for controlled generation from (conditional) language models that flexibly combines pretrained LMs with any differentiable constraints. With experiments on style transfer and controlled machine translation, and multiple combination of constraints, we show the effectiveness of this approach. In addition to its potential applications in factual rewriting and debiasing text, this work holds promise in making language generation models personalizable and adaptive to different dialects or even individual speakers, since MuCoCO re-uses pre-trained LMs without adaptation and can incorporate constraints (e.g., dialect or user properties) trained on

very little data. Future work will explore more sophisticated optimization techniques to improve the computational efficiency of our approach, and gradient-descent based methods for sampling [64] which will allow to sample from the language models with constraints.

References

[1] How we can make machine learning algorithms tunable. https://www.engraved.blog/how-we-can-make-machine-learning-algorithms-tunable/.

[2] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

[3] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[5] D. Cer, M. Diab, E. Agirre, I. Lopez-Gazpio, and L. Specia. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada, Aug. 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-2001. URL https://www.aclweb.org/anthology/S17-2001.

[6] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/chen18a.html.

[7] Y. Cheng, Z. Gan, Y. Zhang, O. Elachqar, D. Li, and J. Liu. Contextual text style transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2915–2924, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.263. URL https://www.aclweb.org/anthology/2020.findings-emnlp.263.

[8] V. Cohen and A. Gokaslan. OpenGPT-2: Open language models and implications of generated text. *XRDS*, 27(1):26–30, Sept. 2020. ISSN 1528-4972. doi: 10.1145/3416063. URL https://doi.org/10.1145/3416063.

[9] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=H1edEyBKDS.

[10] G. Debreu. Valuation equilibrium and Pareto optimum. *Proceedings of the National Academy of Sciences*, 40(7):588–592, 1954. ISSN 0027-8424. doi: 10.1073/pnas.40.7.588. URL https://www.pnas.org/content/40/7/588.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 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, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423.

[12] S. Edunov, M. Ott, M. Auli, D. Grangier, and M. Ranzato. Classical structured prediction losses for sequence to sequence learning. 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 355–364, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1033. URL https://www.aclweb.org/anthology/N18-1033.

[13] J. Ficler and Y. Goldberg. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4912. URL https://www.aclweb.org/anthology/W17-4912.

[14] A. Field and Y. Tsvetkov. Unsupervised discovery of implicit gender bias, 2020.

[15] S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of EMNLP*, 2020. URL https://www.aclweb.org/anthology/2020.findings-emnlp.301/.

[16] J. Gu, J. Bradbury, C. Xiong, V. O. Li, and R. Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1l8BtlCb.

[17] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[18] C. D. V. Hoang, G. Haffari, and T. Cohn. Towards decoding as continuous optimisation in neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 146–156, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1014. URL https://www.aclweb.org/anthology/D17-1014.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[20] C. Hokamp and Q. Liu. Lexically constrained decoding for sequence generation using grid beam search, 2017.

[21] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing. Toward controlled generation of text. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 06–11 Aug 2017. URL http://proceedings.mlr.press/v70/hu17e.html.

[22] X. Huang and M. J. Paul. Neural user factor adaptation for text classification: Learning to generalize across author demographics. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 136–146, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/S19-1015.

[23] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, A. F. T. Martins, and A. Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P18-4020.

[24] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[25] N. S. Keskar, B. McCann, L. Varshney, C. Xiong, and R. Socher. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*, 2019.

[26] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997. ISSN 0890-5401. doi: https://doi.org/10.1006/inco.1996.2612. URL https://www.sciencedirect.com/science/article/pii/S0890540196926127.

[27] B. Krause, A. D. Gotmare, B. McCann, N. S. Keskar, S. Joty, R. Socher, and N. F. Rajani. GeDi: Generative discriminator guided sequence generation, 2020.

[28] K. Krishna, J. Wieting, and M. Iyyer. Reformulating unsupervised style transfer as paraphrase generation. In *Empirical Methods in Natural Language Processing*, 2020.

[29] S. Kumar and Y. Tsvetkov. Von Mises-Fisher loss for training sequence to sequence models with continuous outputs. In *Proc. of ICLR*, 2019. URL https://arxiv.org/pdf/1812.04616.pdf.

[30] S. Kumar, S. Chakrabarti, and S. Roy. Earth mover's distance pooling over siamese LSTMs for automatic short answer grading. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 2046–2052. AAAI Press, 2017. ISBN 9780999241103.

[31] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger. From word embeddings to document distances. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 957–966, Lille, France, 07–09 Jul 2015. PMLR. URL http://proceedings.mlr.press/v37/kusnerb15.html.

[32] J. Li, R. Jia, H. He, and P. Liang. Delete, retrieve, generate: a simple approach to sentiment and 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 1865–1874, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1169. URL https://www.aclweb.org/anthology/N18-1169.

[33] X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation, 2021.

[34] X. Lin, H.-L. Zhen, Z. Li, Q.-F. Zhang, and S. Kwong. Pareto multi-task learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/685bfde03eb646c27ed565881917c71c-Paper.pdf.

[35] X. Lin, Z. Yang, Q. Zhang, and S. Kwong. Controllable pareto multi-task learning, 2021.

[36] P. Lison and J. Tiedemann. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1147.

[37] A. Liu, M. Sap, X. Lu, S. Swayamdipta, C. Bhagavatula, N. A. Smith, and Y. Choi. On-the-fly controlled text generation with experts and anti-experts, 2021.

[38] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach, 2019.

[39] L. Logeswaran, H. Lee, and S. Bengio. Content preserving text generation with attribute controls, 2018.

[40] X. Ma, M. Sap, H. Rashkin, and Y. Choi. PowerTransformer: Unsupervised controllable revision for biased language correction, 2020.

[41] A. Pagnoni, V. Balachandran, and Y. Tsvetkov. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021.

[42] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 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, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

[43] B. Paria, C.-K. Yeh, I. E. H. Yen, N. Xu, P. Ravikumar, and B. Póczos. Minimizing FLOPs to learn efficient sparse representations, 2020.

[44] J. Platt and A. Barr. Constrained differential optimization. In D. Anderson, editor, *Neural Information Processing Systems*. American Institute of Physics, 1988. URL https://proceedings.neurips.cc/paper/1987/file/a87ff679a2f3e71d9181a67b7542122c-Paper.pdf.

[45] S. Prabhumoye, Y. Tsvetkov, R. Salakhutdinov, and A. W. Black. Style transfer through back-translation, 2018.

[46] L. Qin, V. Shwartz, P. West, C. Bhagavatula, J. D. Hwang, R. Le Bras, A. Bosselut, and Y. Choi. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.58. URL https://www.aclweb.org/anthology/2020.emnlp-main.58.

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

[48] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[49] S. Rao and J. Tetreault. 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, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1012. URL https://www.aclweb.org/anthology/N18-1012.

[50] S. Reddy and K. Knight. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 17–26, 2016.

[51] N. Reimers and I. Gurevych. Sentence-BERT: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[52] N. Reimers and I. Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation, 2020.

[53] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization, 2019.

[54] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola. Style transfer from non-parallel text by cross-alignment, 2017.

[55] C. Song, A. Rush, and V. Shmatikov. Adversarial semantic collisions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4198–4210, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.344. URL https://www.aclweb.org/anthology/2020.emnlp-main.344.

[56] F. Stahlberg and B. Byrne. On NMT search errors and model errors: Cat got your tongue? 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 3356–3362, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1331. URL https://www.aclweb.org/anthology/D19-1331.

[57] S. Subramanian, G. Lample, E. M. Smith, L. Denoyer, M. Ranzato, and Y.-L. Boureau. Multiple-attribute text style transfer, 2019.

[58] A. Sudhakar, B. Upadhyay, and A. Maheswaran. "Transforming" delete, retrieve, generate approach for controlled text 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 3269–3279, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1322. URL https://www.aclweb.org/anthology/D19-1322.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.

[60] E. Wallace, S. Feng, N. Kandpal, M. Gardner, and S. Singh. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1221. URL https://www.aclweb.org/anthology/D19-1221.

[61] E. Wallace, M. Stern, and D. Song. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5531–5546, Online, Nov. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.446. URL https://www.aclweb.org/anthology/2020.emnlp-main.446.

[62] A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*, 2018.

[63] S. Welleck, I. Kulikov, S. Roller, E. Dinan, K. Cho, and J. Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJeYe0NtvH.

[64] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 681–688, Madison, WI, USA, 2011. Omnipress. ISBN 9781450306195.

[65] J. Wieting, T. Berg-Kirkpatrick, K. Gimpel, and G. Neubig. Beyond BLEU: Training neural machine translation with semantic similarity. In *Proceedings of the Association for Computational Linguistics*, 2019. URL https://arxiv.org/abs/1909.06694.

[66] A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. 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 1112–1122. Association for Computational Linguistics, 2018. URL http://aclweb.org/anthology/N18-1101.

[67] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush. HuggingFace's transformers: State-of-the-art natural language processing, 2020.

[68] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Łukasz Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[69] K. Yang and D. Klein. FUDGE: Controlled text generation with future discriminators, 2021.

[70] L. Yu, W. Zhang, J. Wang, and Y. Yu. SeqGAN: Sequence generative adversarial nets with policy gradient, 2017.

[71] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news, 2020.

[72] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

[73] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving. Fine-tuning language models from human preferences, 2020.

A Overview of the Method

Figure 2 shows an overview of our proposed approach.

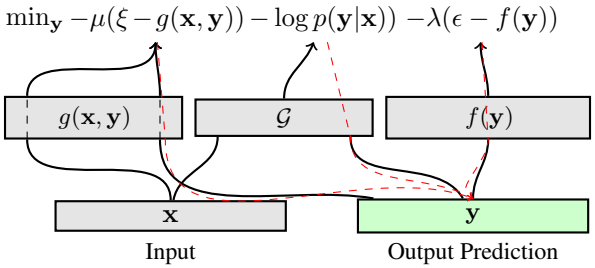


Figure 2: MuCoCO architecture. At each step, only the output sequence $\mathbf{y}$ is updated by receiving gradients from the primary objective of the base text generation model $\mathcal{G}$ as well as the constraints f and g, corresponding to arbitrary text attributes to control for at decoding time. Any number of differentiable constraints can be incorporated. Black arrows indicate forward pass while the red dashed arrows indicate the backward pass. The parameters of all the objectives remain frozen (shown in gray).

B MuCoCO Decoding Algorithm

Algorithm 1: MuCoCO: detailed decoding algorithm

Input: input sequence $\mathbf{x}$, output length L, base model $\mathcal{G}$, attribute functions f_i and g_j and their respective initial and final thresholds, threshold update schedule, step sizes η_1, η_2;
Result: output sequence $\mathbf{y}$
For all $k \in \{1, \ldots, L\}$, initialize $\tilde{\mathbf{y}}_k^0$ uniformly over Δ_V;
For all $i \in \{1, \ldots u\}$ and $j \in \{1 \ldots v\}$, initialize λ_i^0, μ_i^0 as 0 and the thresholds ϵ_i^0, ξ_j^0 with the given values ;
for $t = 1, \ldots,$ MaxSteps **do**
  `// forward pass`
  for all k, compute $\hat{y}_k = \texttt{one-hot}(\arg\max \tilde{y}_k)$ and compute the loss $\mathcal{L}$ (using (5));
  `// backward pass`
  for all k, i and j, compute $\nabla_{\tilde{y}_k}^{t-1} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{y}}_k}$, $\nabla_{\lambda_i}^{t-1} = \frac{\partial \mathcal{L}}{\partial \lambda_i}$, $\nabla_{\mu_j}^{t-1} = \frac{\partial \mathcal{L}}{\partial \mu_j}$;
  `// Update the parameters`
  update $\tilde{y}_k^{(t+1)} \propto \tilde{y}_k^{(t)} \exp(1 - \eta_1 \nabla_{\tilde{y}_k} \mathcal{L})$;
  update $\lambda_i^t = \max(0, \lambda_i^{t-1} + \eta_2 \nabla_{\lambda_i} \mathcal{L})$, and $\mu_i^t = \max(0, \mu_i^{t-1} + \eta_2 \nabla_{\mu_i} \mathcal{L})$;
  update ϵ_i^t, ξ_j^t following the threshold update schedule
end
return $\arg\min_t \{-\log p(\tilde{\mathbf{y}}^{(t)}|\mathbf{x}) : \forall i, f_i(\tilde{\mathbf{y}}^{(t)}) \leq \epsilon_i, \forall j, g_j(\mathbf{x}, \tilde{\mathbf{y}}^{(t)}) \leq \xi_j\}$;

C Details of Attribute Models

C.1 Semantic similarity models

We explain the semantic similarity models we use in our experiments in more detail here:

USIM USIM named after UKPLab-Sentence-Transformers is defined as $\text{USIM}(\mathbf{x}, \mathbf{y}) = \text{cosine}(M(x), M(y))$. In other words, it is the cosine similarity between the representations of a model M. This model is parameterized by GPT2(345M) [8]. $M(x)$ is obtained by first feeding $\mathbf{x}$ to the model and then mean pooling all the output representations. This model originally presented in Reimers and Gurevych [51] is trained in a Siamese fashion on BERT [38] but is easily extensible to any LM architecture. We adapt it to GPT2 as follows:

- First, we fine-tune M =GPT2 on the combination of SNLI and MNLI [66] corpora which are both designed for training natural language inference model and intended to capture semantics. Each corpus contains pairs of sentencse with one of the three annotations: inference, contradiction or neutral. For each input sentence $(\mathbf{s}_1, \mathbf{s}_2)$, the model is trained as with classification objective with the final logits computed as $W[M(\mathbf{s}_1), M(\mathbf{s}_2), |M(\mathbf{s}_1) - M(\mathbf{s}_2)|]$, where W is a trainable parameter. In other words the three vectors as shown are concatenated and multiplied with a weight matrix. We train this for 1 epoch on the combined corpora.
- Second, we continue fine-tuning the M trained so far on the STS corpus which consists of pairs of sentences annotated with real numbers in $[-1, 1]$ indicating their semantic similarity. We train on this corpus with a mean-square-error loss between $\text{cosine}(M(\mathbf{s}_1), M(\mathbf{s}_2))$ and the given score.

For details of training M can be found in [51] where this model is shown to perform competitively on STS benchmarks [66]. We use this model for adding constraints in style-transfer (§3.1) and multi-attribute transfer (§4).

XSIM Similar to USIM, we define $\text{XSIM}(\mathbf{x}, \mathbf{y}) = \text{cosine}(CM(x), CM(y))$, where CM is a cross-lingual model. This method was introduced by Reimers and Gurevych [52] where they distill a monolingual model such as M, to train a cross-lingual model with a small parallel corpus in the languages of interest. Given a parallel sentence pair $(\mathbf{x}, \mathbf{y})$, CM is trained by minimizing the following loss:

$$\mathcal{L}_{\text{XSIM}} = \|M(\mathbf{x}) - CM(\mathbf{x})\|_2^2 + \|CM(\mathbf{x}) - CM(\mathbf{y})\|_2^2$$

That is, representations of the model M and CM for the source sentence are trained to be close together as are the cross-lingual representations of source and target. We parameterize CM also with pretrained GPT2 (345M) [8] model. But GPT2 and the Marian Transformer based MT model [23] we use do not have matching vocabularies. Since the vocabulary of the primary objective and constraints should match for the decoding to work, we replace input word embedding layer of GPT2 with that of the decoder of the translation model before we train the distilled model. We use the TED2020 [] French-English parallel corpus containing around 400K sentence-pairs to train XSIM and obtain comparable performance as Reimers and Gurevych [52] on the cross-lingual STS benchmark [66].

WMD Given two bags of words, $x = \{x_1, \ldots, x_n\}$ and $y = \{y_1, \ldots, y_m\}$, and an embedding table $\mathbf{e}$, we define word mover's distance between $\mathbf{x}$ and $\mathbf{y}$ as

$$\text{WMD}(\mathbf{x}, \mathbf{y}) = \min \sum_{i=1,j=1}^{m,n} T_{ij} d_{ij} \text{subject to}$$
$$\sum_{i}^{n} T_{ij} = \frac{1}{m}$$
$$\sum_{j}^{m} T_{ij} = \frac{1}{n}$$

where we define $d_{ij} = 1 - \cos(\mathbf{e}(x_i), \mathbf{e}(y_j))$. Given fixed inputs $\mathbf{e}(x_i)$ and $\mathbf{e}(y_j)$, WMD can easily be computed using linear program solver [8]. To backpropagate through this objective. We use the following steps following Kumar et al. [30]:

1. During the forward pass, we obtain $\hat{\mathbf{y}}$ as indicated in algorithm 1 and compute word embeddings for both the input $\mathbf{x}$ and the prediction $\hat{\mathbf{y}}$. Using the linear program solver, we compute $\text{WMD}(\mathbf{x}, \hat{\mathbf{y}})$ as well the proportions T_{ij}
2. During the backward pass, we keep the T_{ij} fixed which removes the constraints from the WMD computation as described making it differentiable allowing gradients to flow to update the optimization parameters $\tilde{y}$.

We use the embedding table from USIM model as $\mathbf{e}$ for this constraint.

[8] We solve it using the python library POT: `https://pythonot.github.io/`

C.2 Models used in multi-attribute transfer

In §4, we present a paraphrasing model with 4 different constraints: USIM as described previously and three classifier constraints. All the classifiers are trained by finetuning GPT2[9] on the following corpora:

Age We use the NUFA corpus [22] consisting Yelp Restaurant Reviews with 300K sentences per age group (greater than 30 years, and less than 30 years) in the training set. The age was self-declared by the reviewers in their Yelp profiles. Our classifier achieves an accuracy of 80% on a balanced test set of 10K sentences.

Formality We use GYAFC corpus as described in §3.1 for this constraint (with an accuracy of around 92%) on the provided test set.

Sentiment We collect Yelp restaurant reviews using scripts provided by Subramanian et al. [57][10] with a rating from 1 to 5 star. We subsample from this corpus to train our 5-class classifier on 100K reviews per rating obtaining a classification accuracy of around 75% on a held-out test set also sampled from the same corpus.

D More Details of Human Evaluation

We conduct A/B testing to rank translations generated by our method and beam search. We show the annotators the source sentence and two randomized translations (one from beam search and one from our method). We ask them to choose one of the four options: **1**: the first translation is both faithful and formal while the second is not, **2**: the second translation is both faithful and formal while the second is not, **3**: both are faithful and formal, and **4**: both are either unfaithful or informal or both. Results are summarized in §3.2.

E Examples

E.1 Style Transfer

We show selected examples from our style-transfer models in Table 3. Since the final output $\mathbf{y}$ is generated from the paraphrase $\mathbf{z}$, not the input sentence $\mathbf{x}$, some of the content is at times modified in the final output in decoding without constraints. MUCOCO with content based constraints is able to recover content in some examples and also improve formality of the outputs. But it can still be prone to errors since the content-similarity metrics are not perfect. See §3.1 for more details.

E.2 Style-controlled Machine Translation

Table 4 lists few selected examples for inducing cross-lingual similarity and formality constraints in a French to English MT model. We find that inducing formality modifies some of the constructs (like removing contractions: "gonna" to "going to") in the output sentences which are not measured accurately by a surface-level metric like BLEU. See §3.2 for more details.

E.3 Multiple Solutions on the Pareto Front

Table 6 shows a few examples of changing constraint thresholds for semantic similarity as well as formality constraints. Since the classifiers are not well calibrated, we find that with tighter constraints, the outputs tend to overly represent formality indicating words while losing some of the content which the semantic similarity models are not always robust enough to detect. See §4 for more details.

[9] we use Huggingface [67] with recommended hyperparameters for training all classifiers: `https://huggingface.co/transformers/v2.0.0/examples.html`

[10] `https://github.com/facebookresearch/MultipleAttributeTextRewriting/tree/master/data/Yelp`

Input	all are good movies u cant rate them
Paraphrase	all good films you can't rate
NO-CONSTRAINTS	All good movies that you can't rate
FUDGE(FORMAL)	All good movies you can not rate
MUCOCO (FORMAL,USIM,WMD)	All are good movies you can not rate them adequately
Input	One guy says to me 'So, where's the fox from Mystic Pizza?
Paraphrase	"so, where's the Mystic Pizza guy?" Said one of the men.
NO-CONSTRAINTS	One of the men replied , "So , where is the Mystic Pizza guy?
FUDGE(FORMAL)	One of the men replied, "Where is the Mystic burger?"
MUCOCO (FORMAL,USIM,WMD)	One of the men enquired, "Where is the Mystic Pizza man?"
Input	chill out sweetie everything will be fine eventually.
Paraphrase	calm down, sweetheart, everything will be fine.
NO-CONSTRAINTS	Calm down, sweetheart, everything is going to be alright.
FUDGE(FORMAL)	Calm down sweetheart, everything will be alright.
MUCOCO (FORMAL,USIM,WMD)	Calm down sweetheart, everything will eventually proceed to be good.
Input	what kinda person are you to even ask that ridiculous (er...spell check that...) question?
Paraphrase	what kind of fool would ask such a silly question?
NO-CONSTRAINTS	What kind of idiot would ask such a dumb question?
FUDGE(FORMAL)	What kind of fool would ask such a dull question?
MUCOCO (FORMAL,USIM,WMD)	What kind of foolish person would ask such a dull question?
Input	I spent my money, enjoyed myself and didn't loose more than I could afford - I won....
Paraphrase	I've spent my money, I've enjoyed it, and I didn't lose anything.
NO-CONSTRAINTS	I spent my money, I enjoyed it, and I did not lose anything.
FUDGE(FORMAL)	I have spent my money, I have enjoyed it, and I did not lose anything.
MUCOCO (FORMAL,USIM,WMD)	I spent my money, did not lose anything more, and it was simply enjoyable.

Table 3: Style transfer examples with different decoding methods and constraints.

E.4 Multi-attribute Transfer

Table 5 shows a few examples of transfering multiple combinations of attributes in a given input sentence. We focus on sentiment amplification rather than transfer as it is by definition prone to losing content. See more details in §4.

Source	Mais il s'agit... il s'agit d'une femme que vous ne connaissez pas.
Reference	But this is– This is a woman you don't know.
BEAMSEARCH	But this is... this is a woman you don't know.
MUCOCO (XSIM)	But this is... this is a woman you don't know.
FUDGE(FORMAL)	But this is... this is a woman you do not know.
MUCOCO (FORMAL)	But this is... is a woman you do not know.
MUCOCO (FORMAL,XSIM)	But this is a woman you do not know.
Source	Toi ? Le mec à bananes, exact.
Reference	- Who's the banana man, alright.
BEAMSEARCH	You, the banana guy, right.
MUCOCO (XSIM)	You? the banana guy, right.
FUDGE(FORMAL)	You, the banana guy, right?
MUCOCO (FORMAL)	Are you the banana guy?
MUCOCO (FORMAL,XSIM)	Are you the banana guy?
Source	Nous allons les sortir de la d'ici quelques minutes.
Reference	We'll have them out in a couple minutes.
BEAMSEARCH	We're gonna get them out of here in a few minutes.
MUCOCO (XSIM)	We're gonna get them out of here in a few minutes.
FUDGE(FORMAL)	We'll get them out of here in a few minutes.
MUCOCO (FORMAL)	We will get them out of here.
MUCOCO (FORMAL,XSIM)	We will get them out of here in a few minutes.
Source	On va prendre la voie aérienne.
Reference	We'll take the aerial up.
BEAMSEARCH	We're gonna take the airway.
MUCOCO (XSIM)	We're gonna take the air route.
FUDGE(FORMAL)	We are gonna take the airway.
MUCOCO (FORMAL)	We are going to take the air.
MUCOCO (FORMAL,XSIM)	We are going take the air route.
Source	Mais mon sang ne correspondait pas.
Reference	But my blood didn't match.
BEAMSEARCH	But my blood wasn't matching.
MUCOCO (XSIM)	But my blood didn't match.
FUDGE(FORMAL)	But my blood wasn't matched.
MUCOCO (FORMAL)	But my blood was not correct.
MUCOCO (FORMAL,XSIM)	But my blood did not match.

Table 4: Translation examples with different decoding methods and constraints.

$<$ 30 years, informal, 4-star	**one big plus : the coffee is always fantastic .**
$<$ 30 years, informal, 5-star	the coffee is always great !
$<$ 30 years, formal, 4-star	this coffee is incredibly good.
$<$ 30 years, formal, 5-star	the coffee is consistently outstanding!
$>$ 30 years, informal, 4-star	the espresso is usually enjoyed .
$>$ 30 years, informal, 5-star	the coffee is usually delicious also!
$>$ 30 years, formal, 4-star	the espresso is pleasantly delicious, nonetheless.
$>$ 30 years, formal, 5-star	the coffee is brewed to excellence.
$<$ 30 years, informal, 2-star	**i left our meal feeling a little disappointed .**
$<$ 30 years, informal, 1-star	worst feeling with this little meal .
$<$ 30 years, formal, 2-star	i felt failed and disappointed by this meal .
$<$ 30 years, formal, 1-star	i left our meal feeling anguished, betrayed .
$>$ 30 years, informal, 2-star	i was a little disappointed !
$>$ 30 years, informal, 1-star	this meal bummed me out !
$>$ 30 years, formal, 2-star	i felt unsatisfied by this meal.
$>$ 30 years, formal, 1-star	i felt complete disappointment after this meal .

Table 5: MUCOCO with multiple constraints and rewriting reviews with different combination of attributes.

Input Sentence	My dad looks like Paul Newman, and my ex looked like king kong
Paraphrase	my dad's like Paul Newman, and my ex looks like a king.
Constraints	**Outputs**
$\text{FORMAL}(\mathbf{y}) > 0.5$, $\text{USIM}(\mathbf{x}, \mathbf{y}) < 0.15$	My dad looks like Paul Newman, and my ex looks similar to King Kong
$\text{FORMAL}(\mathbf{y}) > 0.7$, $\text{USIM}(\mathbf{x}, \mathbf{y}) < 0.15$	My father looks like Paul Newman, and my ex resembles a King Kong
$\text{FORMAL}(\mathbf{y}) > 0.9$, $\text{USIM}(\mathbf{x}, \mathbf{y}) < 0.15$	My father looks like Paul Newman, and my ex possesses the qualities of King Kong approximately
$\text{FORMAL}(\mathbf{y}) > 0.7$, $\text{USIM}(\mathbf{x}, \mathbf{y}) < 0.1$	My dad possesses looks similar to Paul Newman, my ex appears like King King Kong
$\text{FORMAL}(\mathbf{y}) > 0.9$, $\text{USIM}(\mathbf{x}, \mathbf{y}) < 0.05$	My dad possesses the Paul Newman looks similar my ex possesses similar King Kong resemblance

Table 6: Varying thresholds for the constraints to find other solutions on the Pareto front.