CEV-LM: Controlled Edit Vector Language Model for Shaping Natural Language Generations

Samraj Moorjani[1], Adit Krishnan[1], Hari Sundaram[1]
[1]University of Illinois at Urbana-Champaign, USA
{samrajm2, aditk2, hs1}@illinois.edu

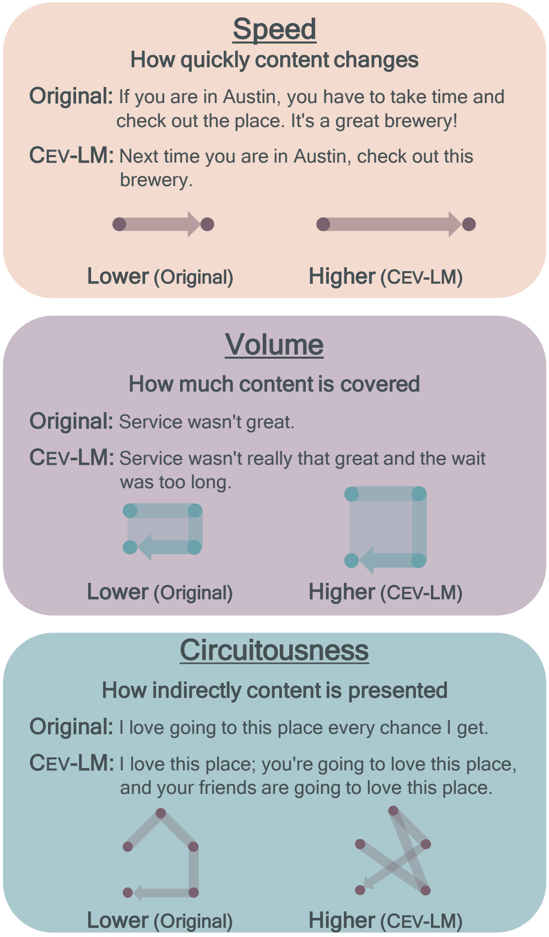
Abstract

As large-scale language models become the standard for text generation, there is a greater need to tailor the generations to be more or less concise, targeted, and informative, depending on the audience/application. Existing control approaches primarily adjust the semantic (*e.g.,* emotion, topics), structural (*e.g.,* syntax tree, parts-of-speech), and lexical (*e.g.,* keyword/phrase inclusion) properties of text, but are insufficient to accomplish complex objectives such as pacing which control the complexity and readability of the text. In this paper, we introduce CEV-LM - a lightweight, semi-autoregressive language model that utilizes constrained edit vectors to control three complementary metrics (speed, volume, and circuitousness) that quantify the shape of text (*e.g.,* pacing of content). We study an extensive set of state-of-the-art CTG models and find that CEV-LM provides significantly more targeted and precise control of these three metrics while preserving semantic content, using less training data, and containing fewer parameters.[1]

1 Introduction

As large-scale pre-trained language models allow the generation of more diverse and fluent text, controllable text generation (CTG) is crucial to meet the needs of different applications and audiences. For instance, complex ideas can be presented concisely to an expert, but non-technical audiences may need more context and a slower-paced introduction to grasp the same idea. Existing CTG approaches empirically evaluate three types of control conditions: semantic (*e.g.,* emotion, topics), structural (*e.g.,* syntax tree, parts-of-speech), and lexical controls (*e.g.,* keyword/phrase inclusion) (Zhang et al., 2022). While this taxonomy covers a broad range of features, it does not target more complex objectives, such as the pacing of text. Toubia et al. (2021) presents a set of measures that quantify the shape of narratives, relying on both semantic and structural properties of the text. Speed measures how quickly content changes, volume quantifies how much content is covered, and circuitousness represents how indirectly content is presented.

Figure 1: Generated examples of change in speed, volume, and circuitousness, metrics that define the shape of text, and stylized illustrations. The points represent the word embeddings of windows of text, $\{x_1, ..., x_n\}$. The original text has a lower value of the metric, and our generation (CEV-LM) demonstrates a higher value.



Speed
How quickly content changes

Original: If you are in Austin, you have to take time and check out the place. It's a great brewery!

CEV-LM: Next time you are in Austin, check out this brewery.

Lower (Original)
Higher (CEV-LM)
Volume
How much content is covered

Original: Service wasn't great.

CEV-LM: Service wasn't really that great and the wait was too long.

Lower (Original)
Higher (CEV-LM)
Circuitousness
How indirectly content is presented

Original: I love going to this place every chance I get.

CEV-LM: I love this place; you're going to love this place, and your friends are going to love this place.

Controlling these "nonstandard" control conditions, such as speed, volume, and circuitousness,

[1] Our code and data are accessible at this link https://drive.google.com/file/d/10rwCLJ96eNP5LS_1sG-flWvXD9X4pbjO

Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 1325–1340
March 17-22, 2024 ©2024 Association for Computational Linguistics

is challenging because they are built on interconnected semantic, structural, and lexical properties. CTG approaches have been developed and tested separately for semantic, structural, or lexical features, but not at the intersection of multiple features (Zhang et al., 2022; Li et al., 2022). Furthermore, these nonstandard control conditions require sentence and paragraph-level reconstruction. This is challenging for purely autoregressive approaches, which struggle with longer context lengths (Beltagy et al., 2020). Conversely, deep generative model-based approaches, such as Guu et al. (2018), produce generations from a continuous latent variable, enabling simple, gradient-based methods to perform complex control tasks over larger contexts (Li et al., 2022; Han et al., 2022).

In summary, our contributions are as follows: firstly, we present the CEV-LM framework to provide a lightweight, "tuning-knob" to control speed, volume, and circuitousness. We adopt a semi-autoregressive paradigm to exploit both the generation quality of autoregressive models and the controllability of deep generative models. Second, we propose a controlled edit vector approach where we preselect examples from a constrained similarity neighborhood to match our criteria and apply a controlled edit vector in the latent space to tune the desired attribute. Lastly, we study a robust set of benchmarks used in CTG and demonstrate that CEV-LM provides significantly more control and preserves both relevance and fluency (§6) across both high and low-resource settings while using fewer training samples and parameters.

2 Related Work

In this section, we briefly introduce existing literature on controllable text generation (§2.1) and the prototype-then-edit architecture (§2.2).

2.1 Controllable Text Generation

Zhang et al. (2022) find that existing controllable text generation (CTG) methods fall under three major categories: fine-tuning (Li and Liang, 2021; Tambwekar et al., 2019; Ouyang et al., 2022), retraining/refactoring (Keskar et al., 2019), and post-processing (Dathathri et al., 2020; Scialom et al., 2020; Krause et al., 2020; Kumar et al., 2021). Li and Liang (2021) train a small, continuous, task-specific vector prepended to the input of a pretrained language model (PLM), keeping the parameters of the PLM frozen, providing a lightweight alternative to fine-tuning. Krause et al. (2020) guides the generation of a larger PLM using two class-conditional language models, one conditioned on the desired control and one conditioned on the "anti-control". Kumar et al. (2021) replaces traditional decoding with a continuous optimization problem, where desired controls can be expressed as multiple differentiable constraints.

Many CTG works utilize deep generative models such as variational auto-encoders (VAEs) (Guu et al., 2018; Xu et al., 2020; Wang et al., 2019), generative adversarial networks (GANs) (Scialom et al., 2020), and diffusion models (Li et al., 2022; Han et al., 2022) because of the malleability of the latent state. However, recent work has relied on plug-and-play approaches with large-scale pretrained language models (PLMs) without significant task-specific retraining. The autoregressive design of PLMs makes it challenging to exhibit control on sentence- and paragraph-level constraints such as speed, volume, and circuitousness (Toubia et al., 2021). Further, despite the benefits of fine-tuning and post-processing-based approaches, more direct control is necessary (Soatto et al., 2023). While discretized controls are more natural (*e.g.,* less vs. more toxic), we emphasize a "tuning-knob"-like control as it provides more fine-grained control, and it is trivial to go from continuous to discrete controls, but not the converse.

2.2 Prototype-then-Edit

Prototype editing applies attribute markers to predefined sentence templates to generate sentences that are semantically similar but altered content (Guu et al., 2018; Li et al., 2018; Sudhakar et al., 2019). Guu et al. (2018) introduce an unconditional generative model that samples a "prototype" sentence from the training corpus and edits it using a randomly sampled edit vector. In the Yelp restaurant review corpus, 70% of the test set is within a Jaccard distance of 0.5 of a training set sentence, implying that a neural editor with smooth and consistent edits should capture the test set. The edit model has two significant constraints: semantic smoothness and consistent edit behavior. Specifically, edits should change the semantics of text by a small amount and, when stacked together, create a more significant change. Further, the edit vector, z, should control the change in a sentence such that when applied to different sentences, the edits are semantically analogous. We adopt the prototype-then-edit framework because of the ability of the edit vector to reflect a desired change in attribute in the latent space, hence "controlled edit vectors".

3 Nonstandard Control Conditions

We pick three non-standard control metrics to evaluate our approach: speed, volume, and circuitousness (Toubia et al., 2021). Speed is a measure of how quickly content moves in a given text, calculated as the distance traveled by consecutive windows of text. Specifically, speed is equal to $\frac{\sum_{t=1}^{T-1} \|x_{t+1}-x_t\|}{T-1}$ where x_t is the word embedding of the $t-th$ window of text. Volume captures the amount of information covered in a piece of text, calculated by finding the minimum volume ellipsoid that contains all $x_t, \forall t \in \{1 \ldots T\}$. Circuitousness measures how *indirectly* content is covered and is formulated as $\frac{\sum_{t=1}^{T-1} \|x_{t+1}-x_t\|}{L_{SP}}$ where L_{SP} is the length of the shortest path, computed with the traveling salesman problem. While volume measures how much content is covered, circuitousness answers how that content was covered. Given $s(\cdot)$ to compute the target attribute, we define the control of generated text as how close $s(x) - s(x')$ is to a desired change in attribute, Δ, where x and x' are the generated and original text, respectively. These measures have been used to study the success of narratives and can be used to quantify complex control objectives, such as how concise or informative generations are.

4 Controllable Edit Vectors

The prototype-then-edit architecture (Guu et al., 2018) features three main components: a neural editor $p_{edit}(x|x', z)$, an inverse neural editor $q(z|x', x)$, and an edit prior $p(z)$. The inverse neural editor and neural editor combine to form the encoder and decoder of a variational autoencoder (Kingma and Welling, 2013), respectively. The neural editor is implemented as an autoregressive, sequence-to-sequence model with attention, where given x' as input and z, which is concatenated to the input of the decoder at each step, the model generates x. The edit prior is defined as $z = z_{norm} \cdot z_{dir}$ where z_{norm}, the strength of the edit, is drawn from $\mathcal{U}(0, 10)$ and z_{dir}, the direction of the edit, is sampled from a uniform distribution on the unit sphere. Note that both z_{dir} and z_{norm} are vectors. The inverse neural editor is given the edit pair (x, x') and must infer the edit vector z. The difference between x and x' is represented as

$$f(x, x') = \sum_{w \in I} \Phi(w) \oplus \sum_{w \in D} \Phi(w)$$

where $I = x \setminus x'$ (*i.e.,* the set of words added to x'), $D = x \setminus x'$ (*i.e.,* the set of words deleted from x'), $\Phi(w)$ is the GloVe (Pennington et al., 2014) vector for w, and $\oplus$ is the concatenation operation. The inverse neural editor infers the edit vector through a perturbed version of $f(x, x')$, as follows:

$$q(z_{dir}|x', x) = \text{vMF}(f_{dir}, \kappa)$$

$$q(z_{norm}|x', x) = \mathcal{U}(f_{norm}, f_{norm} + \epsilon)$$

where $f_{norm} = \min(\|f\|, 10 - \epsilon)$ and $f_{dir} = \frac{f}{f_{norm}}$. Let vMF$(\mu, \kappa)$ be a von-Mises Fisher distribution where μ is the mean vector, and κ is the concentration parameter, controlling the decay rate.

To exhibit control over our target attributes, we alter the prototype-then-edit model in two ways: neighborhood creation and edit vector perturbation. The former constrains the inferred edit vector to demonstrate the desired change in attribute within some tolerance ϵ. The latter encourages a perturbation to the edit vector in the desired direction to compensate for ϵ.

Constrained Neighborhood Creation: The likelihood of a sentence is formulated as $p(x) = \sum_{x' \in \mathcal{X}} p(x|x')p(x')$ where x' is prototype sentence and x is the generated sentence. The likelihood $p(x|x')$ is defined as $\mathbb{E}_{z \sim p(z)}[p_{edit}(x|x', z)]$. A sum over all prototypes x' is expensive, so we only sum over the x' that are lexically similar to x - a lexical similarity neighborhood, $\mathcal{N}(x)$. Further, we create an additional constraint on the target attribute to ensure that inferred edit vectors from the inverse neural editor correspond to a specified change in that attribute. More formally, we define the neighborhood with a tolerance ϵ as

$$\mathcal{N}_\Delta(x) = \{x' \in \mathcal{X} : d_J(x, x') < 0.5, \\ |(s(x) - s(x')) - \Delta| \leq \epsilon\}$$

Controlled Edit Vector Peturbation: We hypothesize that by altering the magnitude of z_{norm} and the direction z_{dir}, we can control the strength and behavior of the edit vector. Expressly, we can condition the formulation of the inverse neural editor on the target attribute by defining $q(z_{norm}|x', x) = \mathcal{N}(\Delta, 1) \cdot \mathcal{U}(f_{norm}, f_{norm} + \epsilon)$, where $\mathcal{N}$ is the normal distribution and $\mathcal{U}$ is the uniform distribution.

5 Experimental Settings

We train variants of CEV-LM on the Yelp Restaurant Reviews Corpus (Yelp, 2017). The corpus has over 5.84 million training and 2.08 million test reviews (English) [2] in the original similarity neighborhood (*i.e.,* Jaccard distance less than 0.5[3]). The dataset provides a broad variety of writing styles (*e.g.,* formal vs. informal, positive vs. negative sentiment) and topics (*e.g.,* hotels, food, service, etc.) to test our approach (Gong et al., 2017; Guu et al., 2018; Chu and Liu, 2019).

CEV-LM ($\mathcal{N}$-ONLY): We add the Δ constraint during neighborhood creation and train on the newly formed data.

CEV-LM: We use both modifications, constrained neighborhood creation, and controlled edit vector perturbation.

We provide the hyperparameters for our experiments in Appendix C. As our method generally falls under retraining (Zhang et al., 2022), we provide an extensive set of benchmarks for the fine-tuning and post-processing categories, relying on both deep generative models (*i.e.,* diffusion) and autoregressive architectures.

GPT-3: We construct multiple few-shot prompts for all attributes to generate a sentence with the desired change in attribute given a sentence. The prompts consist of three parts: a language-based description of the attribute, n examples of the desired change in attribute, and the prompt to generate a sentence. We use the "davinci" model for all experiments and describe the process further in Appendix A.

MuCoCO: Multiple **Co**nstraints through **C**ontinuous **O**ptimization (Kumar et al., 2021) is an alternative to fine-tuning for controllable text generation that formulates decoding as a continuous optimization problem with multiple differentiable constraints. To control the three attributes, we define a constraint $|(s(x)-s(y))-\Delta|$ where x and y are the input and output sentences. We train a regressor, $\mathcal{D}(x, y)$, to approximate $s(x) - s(y)$ because the computation of speed, volume, and circuitousness is not differentiable. We present the mean absolute error (MAE) and normalized mean absolute error (NMAE) in Table 1. The MAEs are relatively small compared to the scale of Δ, reflected in the NMAE, indicating a strong regressor. We provide more details on the training of the regressor in Appendix B.

Table 1: Evaluation metrics for the regressor, $\mathcal{D}(x, y)$, and classifier model, $\mathcal{C}(x, y)$, used in MuCoCO and SSD-LM across all control attributes. Note that x and y are two sentences, and the goal is to predict the difference in attribute, either directly or within a bin. For $\mathcal{D}$, we report mean absolute error (MAE) and a normalized mean absolute error (NMAE) and for $\mathcal{C}$, we report F1, MAE, and NMAE.

	$\mathcal{D}$-MAE	$\mathcal{D}$-NMAE	$\mathcal{C}$-F1	$\mathcal{C}$-MAE	$\mathcal{C}$-NMAE
Speed	0.3013	0.0764	0.6533	0.4979	0.1263
Volume	0.2144	0.1034	0.5922	0.6196	0.2987
Circuitousness	0.0459	0.0236	0.6763	0.3730	0.1917

SSD-LM: Semi-autoregressive Simplex-based Diffusion Language Model (SSD-LM) (Han et al., 2022) utilizes diffusion-based language modeling in an iterative manner to generate flexible length text. The diffusion is performed on the vocabulary space allowing for classifier feedback and hence controllable generation. Continuous diffusion models are formulated well for modular control by utilizing gradients from an auxiliary model (*e.g.,* use a sentiment classifier to guide the output of a language model to have positive sentiment). We train a classifier to predict a binned difference in attributes such that all bins contain an equal number of training samples. We record both F1-score as well as the MAE and NMAE between classes in Table 1. The class labels are generally off by at most one due as shown by the low MAEs, indicating a strong classifier. In Appendix B, we describe how we train the classifier model, $\mathcal{C}(x, y)$, to guide generations.

Prefix Tuning: Li and Liang (2021) propose prefix-tuning, a lightweight, modular alternative to fine-tuning that trains a small continuous vector prepended to the input (*i.e.,* a prefix) while keeping the parameters of the language model frozen. The approach is similar to prompt-tuning but allows the task-specific prefix to consist entirely of free parameters. We use the same settings as the abstractive summarization experiment in the original paper, using BART (Lewis et al., 2019) with a prefix sequence length of 200. We freeze the aforementioned regressor and add $\gamma|(s(x)-s(y))-\Delta|$ to the existing loss, where γ is a tunable parameter. We found that $\gamma = 0.1$ yielded the best results.

We record three main evaluation metrics to test

[2] We limit the test set to 1000 samples due to the cost of the OpenAI API.

[3] The Jaccard distance is tuned by (Guu et al., 2018).

Table 2: Achieved delta for speed, volume, and circuitousness across all approaches for different target deltas. The scores are averaged across three training runs (inference runs for GPT-3). We use a tolerance $\epsilon = 0.1$ for all of our approaches, as it empirically provided the best results in Section 6.4. We find that our approaches (CEV-LM ($\mathcal{N}$-ONLY) and CEV-LM) show significantly more control over all control conditions across nearly all target deltas.

Metric	**Speed**				**Volume**			**Circuitousness**			**% Err**
Target Delta	**0.125**	**0.5**	**2.0**	**4.0**	**0.125**	**0.5**	**2.0**	**0.125**	**0.5**	**1.0**	-
BENCHMARK APPROACHES											
GPT-3 (Brown et al., 2020)	-0.091	0.023	0.200	0.294	0.019	0.150	1.769	0.011	0.009	0.021	90.44
MuCoCO (Kumar et al., 2021)	0.169	0.871	1.293	3.610	0.142	0.558	2.310	0.068	0.058	0.054	42.38
SSD-LM (Han et al., 2022)	0.412	0.335	1.094	1.059	0.369	0.283	0.671	**0.075**	0.078	0.109	90.00
Prefix Tuning (Li and Liang, 2021)	0.197	0.245	1.562	-	0.088	0.877	-	-0.004	0.159	-	58.12
OUR APPROACHES											
CEV-LM ($\mathcal{N}$-only)	0.111	**0.457**	**1.760**	**3.621**	0.111	0.443	1.752	**0.072**	0.423	**0.790**	15.51
CEV-LM	**0.118**	0.450	1.755	3.547	**0.114**	**0.451**	**1.863**	0.064	**0.431**	0.781	**14.91**

the strength of the control and ensure generations are relevant. **Delta** measures the change in attribute (*i.e.,* Δ). We report the difference as percent error. **BiLingual Evaluation Understudy (BLEU)** (Papineni et al., 2002) measures the n-gram overlap (lexical similarity) and **BERTScore** (Zhang et al., 2019) measures the semantic similarity to ensure that generations remain on topic. We compute the BLEU and BERTScore between the generated and original sentence to ensure the generations remain lexically and semantically similar to the original content.

6 Results

In this section, we evaluate CEV-LM on the strength of control (Section 6.1) as well as the relevance to the original text (Section 6.2 and Section 6.3). In Section 6.4, we discuss tuning the tolerance hyperparameter and in Section 6.4, the effect of the data distribution on training. We provide qualitative results in Section 6.6.

6.1 Control Evaluation

Table 2 show the achieved delta of each approach across various target deltas along with the average percent error, while Table 4 shows the BERT and BLEU scores. We run all baselines three times and report the average. We find that CEV-LM exhibits significantly greater control of Δ over the baselines while preserving lexical and semantic similarity across all three attributes and all target deltas.

The baselines generally yield fluent but not controlled text. GPT-3 often generates texts with minimal change in attribute (*i.e.,* $\Delta = 0$), showing it cannot understand these nonstandard control conditions through few-shot learning. Conversely, prefix-tuning can replicate the attributes somewhat well but falls short due to neural hallucinations and poor-quality text. In low-resource scenarios (*i.e.,* high target deltas and fewer training samples), prefix-tuning led to significant over- or under-fitting; thus, we omit the results. In many cases, MuCoCO and SSD-LM perform poorly in terms of percent error but sometimes outperform or perform on par with our approaches. While we cannot fully explain this behavior, we hypothesize that the data distribution of Δ in $\mathcal{N}(x)$ plays a significant role. In some cases, we see strong results with both CEV-LM modifications, indicating that combining the modifications is beneficial with certain attributes and when ϵ is tuned. Specifically, volume consistently benefits from controlled edit vector perturbation, while speed and circuitousness show conflicting results. We find that circuitousness has much larger errors on average, likely due to the dependency on computing the shortest path.

We also evaluate on more commonly studied control attributes, formality and toxicity, and present the results in Table 3. Due to its definition, it is likely that CEV-LM is more suited to handle attributes defined with word embeddings. This seems to be reflected in the higher overall percent error, but CEV-LM still produces more controlled generations on both attributes than all other baselines, indicating the robustness of our approach.

6.2 Semantic Similarity

In the former part of Table 4, we report the BERT Scores of all approaches across various target deltas for each nonstandard control condition. We observe

Table 3: Achieved delta for toxicity and formality across all approaches for different target deltas. The scores are averaged across three training runs (inference runs for GPT-3). We use a tolerance $\epsilon = 0.1$ for all of our approaches, as it empirically provided the best results in Section 6.4. We find that our approaches (CEV-LM ($\mathcal{N}$-ONLY) and CEV-LM) show significantly more control over all control conditions across nearly all target deltas.

Metric	**Toxicity**			**Formality**			**% Err**
Target Delta	**0.1**	**0.5**	**0.9**	**0.1**	**0.5**	**0.9**	-
GPT-3 (Brown et al., 2020)	0.017	0.083	0.360	0.279	0.309	0.370	83.74
MuCoCO (Kumar et al., 2021)	0.014	0.041	0.063	0.234	0.148	0.155	92.99
SSD-LM (Han et al., 2022)	0.011	0.218	0.342	0.008	0.376	0.650	58.66
Prefix Tuning (Li and Liang, 2021)	0.188	**0.397**	0.978	0.273	0.339	**0.941**	54.50
CEV-LM (Ours)	**0.075**	0.325	**0.709**	**0.162**	**0.427**	0.810	**27.97**

that CEV-LM consistently outperforms the other approaches while performing about on par with the edit-then-prototype baseline. This demonstrates that our approach preserves semantic similarity while significantly changing the speed, volume, or circuitousness of the text (also seen in Table 13). As the target delta increases, the BERT Score of our approach tends to decrease. We explain this phenomenon further in Section 6.5. We include the scores for formality and toxicity in Table 11.

6.3 Lexical Similarity

In the latter part of Table 4, we report the BLEU Scores of all approaches across various target deltas for each nonstandard control condition. We generally observe similar trends for our approaches in that BLEU score decreases as target delta increases, although to a greater scale. Since BLEU score measures lexical similarity, it is more sensitive to changes in wording, leading to a larger spread of scores. We also find that unlike before, CEV-LM does not clearly outperform the other approaches, which may imply that it presents semantically similar content while changing the wording. We include the scores for formality and toxicity in Table 11.

6.4 Tolerance Tuning

We measure the impact of ϵ on Δ in Table 12 (see Appendix F), testing $\epsilon = \{0.05, 0.1, 0.2\}$. When tolerance is too low, the approach overfits from the lack of training data, leading to smaller percent errors and poor similarity metrics. Including controlled perturbations to edit vectors improves similarity metrics at the cost of Δ, indicating that the approach may help combat overfitting. The effect of controlled edit vector perturbation is inconsistent across tolerance values and attributes, so we use $\epsilon = 0.1$ to provide an effective balance of control over a certain Δ and enough data for robust training. More details can be found in Appendix F.

6.5 Training Delta Distribution

We analyze the distribution of Δ in $\mathcal{N}(x)$ in Figure 2. The distribution is centered at 0 and denser at smaller magnitudes of Δ, implying a lack of training data for larger shifts in the target attribute. This explains the general trend of increasing MAE and decreasing BLEU/BERT score as the target delta increases, seen in Table 13.

In Figure 3, we record the performance against the number of training samples, finding that despite fewer samples, control through low-resource training is just as successful as through high-resource training. While circuitousness is the worst-performing attribute, likely due to the complexity of capturing the shortest path-based computation, it surprisingly does worst in a high-resource setting. It is possible that the change in attribute was too small to capture, even with a high number of samples. We see some success with decreasing the number of samples by a few orders of magnitude while preserving performance, across all attributes but leave extensive investigation to future work.

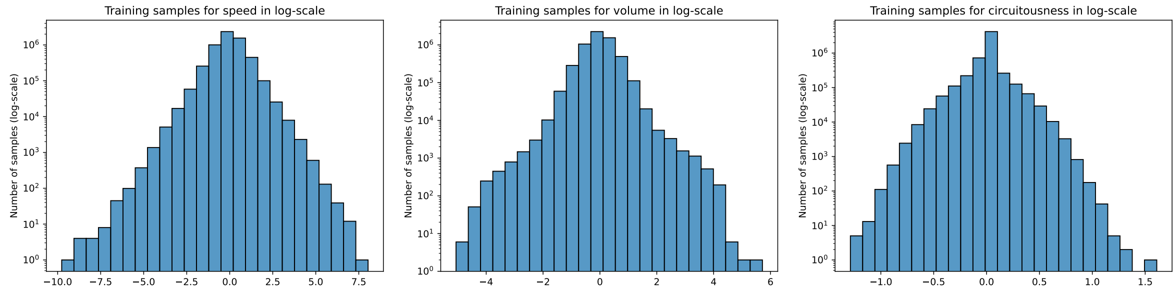
6.6 Qualitative Results

Tables 5 to 7 show the generations of the EDIT-THEN-PROTOTYPE Baseline, GPT-3, SSD-LM, and each of our methods (*i.e.,* CEV-LM ($\mathcal{N}$-ONLY) and CEV-LM) with $\epsilon = 0.1$ when given a target

Table 4: BLEU and BERT (F1) Scores for speed, volume, and circuitousness across all approaches for different target deltas. The scores are averaged across three training runs (inference runs for GPT-3). We use a tolerance $\epsilon = 0.1$ for all of our approaches, as it empirically provided the best results in Section 6.4.

Metric	**Speed**				**Volume**			**Circuitousness**		
Target Delta	**0.125**	**0.5**	**2.0**	**4.0**	**0.125**	**0.5**	**2.0**	**0.125**	**0.5**	**1.0**
BERTScore - Benchmark Approaches										
GPT-3 (Brown et al., 2020)	0.910	0.919	0.916	0.904	0.914	0.916	0.887	0.89	0.881	0.832
MuCoCO (Kumar et al., 2021)	0.657	0.760	0.571	0.728	0.670	0.684	0.615	0.740	0.568	0.552
SSD-LM (Han et al., 2022)	0.841	0.840	0.823	0.821	0.835	0.833	0.825	0.846	0.840	0.822
Prefix Tuning (Li and Liang, 2021)	0.895	0.904	0.891	-	0.879	0.901	-	0.888	0.850	-
BERTScore - Our Approaches										
CEV-LM ($\mathcal{N}$-only)	**0.935**	0.934	**0.929**	0.919	**0.938**	**0.932**	**0.925**	0.927	**0.911**	**0.909**
CEV-LM	**0.935**	**0.939**	**0.928**	**0.923**	0.935	**0.931**	0.921	**0.931**	**0.909**	0.835
BLEU - Benchmark Approaches										
GPT-3 (Brown et al., 2020)	0.233	0.261	**0.321**	0.225	0.219	**0.304**	0.203	0.182	0.187	0.155
MuCoCO (Kumar et al., 2021)	0.326	**0.325**	0.278	0.218	0.256	0.244	0.221	0.318	0.242	0.254
SSD-LM (Han et al., 2022)	0.247	0.246	0.233	**0.283**	0.318	0.298	0.261	**0.321**	**0.279**	**0.274**
Prefix Tuning (Li and Liang, 2021)	0.231	0.217	0.230	-	0.224	0.255	-	0.268	0.246	-
BLEU - Our Approaches										
CEV-LM ($\mathcal{N}$-only)	**0.340**	**0.327**	0.305	0.246	**0.329**	0.268	**0.287**	0.249	0.268	0.248
CEV-LM	0.326	0.313	0.295	0.273	0.304	0.265	0.252	0.290	0.276	0.162

Figure 2: Histogram of delta values (*i.e.,* $s(x) - s(x')$) within the Yelp Restaurant Review Corpus. The x-axis represents the difference in speed within the pairs of our created neighborhood, $\mathcal{N}(x)$, without any constraint on speed. The y-axis counts the number of pairs exhibiting the given delta in log-scale.

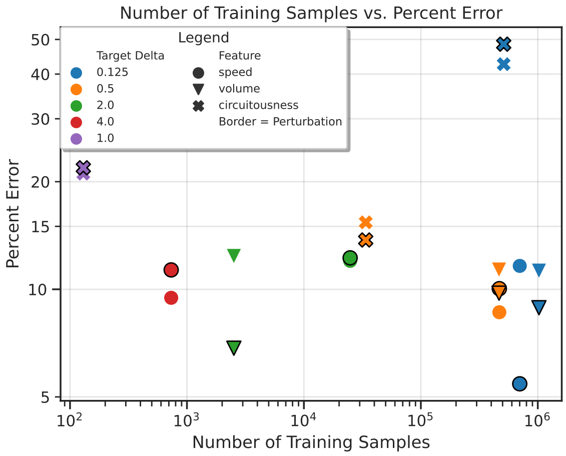


(a) (b) (c)

delta of 0.5 for speed, volume, and circuitousness, respectively. In Table 5, we observe that our approaches generations tend to convey the same information in a shorter span, indicating an increase in speed. GPT-3 is generally not consistent, and SSD-LM tends to stray off-topic. In Table 6, we can see that our approach tends to add information/verbosity, indicating an increase in volume. Here, GPT-3 shows little to no change in generation, and again, SSD-LM tends to stray off-topic, hallucinating information. Lastly, in Table 7, our approach leads to more indirect descriptions. While the text is more verbose, like volume, it repeats information/words, a key facet of circuitousness. GPT-3 hardly changes the input, and SSD-LM hallucinates some information.

In addition to qualitative observations comparing variations of our approach, Table 13 in Appendix G shows the results of training without and with controlled edit vector perturbation, respectively, in

Figure 3: The number of samples used for training versus percent error. The attribute is denoted by the shape, target delta by the color, and the border indicates that perturbation was used. Despite access to significantly fewer samples, low-resource models exhibit similar amounts of control to high-resource models.



comparison to a baseline edit-then-prototype model to show that our approach significantly changes the behavior of the edit-then-prototype model. We also include Appendix D to show qualitative examples of the retrieved prototypes.

7 Conclusion

In this work, we present CEV-LM, an inexpensive, semi-autoregressive language model that uses constrained edit vectors for controllable text generation. We compose an extensive set of controllable text generation benchmarks, and through quantitative and qualitative evaluations, we show that our approach leads to significantly substantially more control over nonstandard control conditions (*e.g.,* speed, volume, circuitousness) while preserving semantic meaning.

Steerable natural language generation remains an open challenge, and we plan to continue improving our work in various directions, such as using a weighted mixture of CEV-LM models to capture all potential target deltas and replacing pieces of our architecture with larger language models. Ultimately, we hope to apply these models to subjective traits like memorability and persuasiveness, which are compositions of many smaller constraints (*e.g.,* conciseness, readability, etc.).

Limitations

While our approach shows substantial control over target attributes, adjusting our formulation to a wide range of controls may be tricky. Neighborhood creation can easily be adapted for any control but severely restricts the amount of training data. Perturbation works well with constraints defined with word embeddings due to how the edit vector is constructed, but it may struggle with other controls. While our approach works in scenarios with sparse training data, the quality of the training data still plays a significant role due to the prototyping step. A higher quality dataset with a large variety of sentences will lead to more diverse and well-suited generations.

Our approach lies in the retraining/refactoring category of controllable text generation models (Zhang et al., 2022). Thus, it requires separate training for every attribute and target delta, which can be expensive as the model is scaled up. While most of our models can be trained for 100,000 steps on a single V100 in under a day[4], we hypothesize that we can use a weighted composition of trained models to achieve any target delta. We leave it to future work to achieve such a framework.

Lastly, in this study, we focus on the numerical control of non-standard control conditions. However, for humans, it is naturally better to quantize the values (e.g., higher vs. slightly higher speed). We choose numerical over categorical controls because fine-grained, numerical control over these features is less explored and more challenging. While it is difficult to go from categorical to numerical control, it is much easier to do the opposite direction - the main challenge is setting the boundaries of the categories.

Ethics Statement

By nature of being trained on data from the internet and because large language models tend to memorize patterns without understanding the language or implications, our approach is susceptible to generating incorrect (Zellers et al., 2019; Maynez et al., 2020; Pagnoni et al., 2021) or biased information as well as toxic language (Wallace et al., 2019; Gehman et al., 2020; Sheng et al., 2021). Although most studies have been conducted on autoregressive frameworks (Bender et al., 2021), CEV-LM is still prone to such problems and future research is

[4] All of our experiments used roughly 1200 GPU hours, including training of baselines.

Table 5: Generations of the baseline (Guu et al., 2018) (**EDIT-THEN-PROTOTYPE**) as well as the benchmarks and CEV-LM architectures for a change in **speed** with a target delta of 0.5. We use a tolerance $\epsilon = 0.1$ for all of our approaches, as it empirically provided the best results in Section 6.4. The models are fed the input (*i.e.,* **Original**) and generate by applying an "edit vector" to the latent representation of the input sentence.

	MODEL & GENERATED TEXT - **SPEED**
EXAMPLE 1	**Original**: He went above and beyond in providing us excellent customer service and was extremely courteous friendly and kind.
	EDIT-THEN-PROTOTYPE: He went above and beyond in providing us with amazing customer service and was extremely courteous friendly and kind.
	GPT-3: He was a great customer service provider. He was friendly and kind,
	SSD-LM: I was pleased that this was in line with our expectations. Suggesting the right
	CEV-LM ($\mathcal{N}$-only): Always amazing customer service and very knowlegable staff.
	CEV-LM: He was knowledgable, courteous, and went provided excellent customer service.
EXAMPLE 2	**Original**: The staff is very professional and friendly & environment is clean.
	EDIT-THEN-PROTOTYPE: The staff is very professional and likable & the environment is clean.
	GPT-3: The staff was very professional, but not too friendly. The environment was clean,
	SSD-LM: This is a great experience! Reviews the best experience in the world.
	CEV-LM ($\mathcal{N}$-only): The friendly staff is very professional & environment is clean.
	CEV-LM: Friendly staff and clean environment.

Table 6: Generations of the benchmarks and CEV-LM architectures for a change in **volume** with a target delta of 0.5. We use a tolerance $\epsilon = 0.1$ for all of our approaches. The models are fed the input (*i.e.,* **Original**) and generate by applying an "edit vector" to the latent representation of the input sentence.

	MODEL & GENERATED TEXT - **VOLUME**
EXAMPLE 1	**Original**: Prices were very reasonable for the quality and quantity served.
	EDIT-THEN-PROTOTYPE: Prices were very reasonable for the quality and quantity of food there.
	GPT-3: The food was good, but the prices were a bit high for what
	SSD-LM: The manager seemed very pleased with the quantity and the good service. The manager was impressed by the price and quality.
	CEV-LM ($\mathcal{N}$-only): The prices were quite reasonable for the quality and quantity of food that was presented.
	CEV-LM: And prices were surprisingly reasonable for the quality and quantity of food that was presented.
EXAMPLE 2	**Original**: I like this place–definitely going back.
	EDIT-THEN-PROTOTYPE: I like going to this place for lunch.
	GPT-3: I really like this place. I'm definitely going back.
	SSD-LM: The service was quite good. The food was available and delicious for two hours.
	CEV-LM ($\mathcal{N}$-only): Overall,everything was great and I'll be coming again next time.
	CEV-LM: Overall,had a great time and I'll definitely be back.

Table 7: Generations of the benchmarks and CEV-LM architectures for a change in **circuitousness** with a target delta of 0.5. We use a tolerance $\epsilon = 0.1$ for all of our approaches. The models are fed the input (*i.e.,* **Original**) and generate by applying an "edit vector" to the latent representation of the input sentence.

	MODEL & GENERATED TEXT - **CIRCUITOUSNESS**
EXAMPLE 1	**Original**: I've only tried their Thai food,so far,and it is very good.
	EDIT-THEN-PROTOTYPE: I've only tried their Thai food,so far,and it is very good.
	GPT-3: I've only tried their Thai food, so far, and it was really
	SSD-LM: They are very good in preparing food. However, if the food is really not good, then the chicken you can eat.
	CEV-LM ($\mathcal{N}$-only): I've decided that their Thai food is really good...their Japanese food,not so much.
	CEV-LM: I've decided that their Thai food is really good...their Japanese food,not so much.
EXAMPLE 2	**Original**: I'd give the decor 4 stars and the food 3 stars.
	EDIT-THEN-PROTOTYPE: I'd give the service 2 stars and the food 3 stars.
	GPT-3: I'd give the decor 3 stars and the food 4 stars.
	SSD-LM: The food was not well-priced and expensive, but very well-made, and I was very pleased with it.
	CEV-LM ($\mathcal{N}$-only): 3 stars for the food and 2 stars for the prices equals 2.5 stars for me.
	CEV-LM: I'd give 4 stars for the food and 3 stars for the service,3 stars for the decor.

necessary to mitigate these issues. However, our framework attempts to achieve controllable outcomes, and future work can experiment with utilizing controllability to address the aforementioned challenges (Liu et al., 2021; Han et al., 2022). Conversely, controllability can be utilized for malicious use cases, and we should ensure that future work continues to defend against such use cases by ensuring released data and models are protected against harmful/de-anonymized content.

Acknowledgements

This work used the Extreme Science and Engineering Discovery Environment (XSEDE) Expanse GPU cluster, which is supported by National Science Foundation grant number ACI-1548562 (Towns et al., 2014). This work was also supported by the National Center for Supercomputing Application's Nano and Delta clusters.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Eric Chu and Peter Liu. 2019. Meansum: A neural model for unsupervised multi-document abstractive summarization. In *International Conference on Machine Learning*, pages 1223–1232. PMLR.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Lin Gong, Benjamin Haines, and Hongning Wang. 2017. Clustered model adaption for personalized sentiment analysis. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 937–946, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. 2022. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. *arXiv preprint arXiv:2210.17432*.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. Gedi: Generative discriminator guided sequence generation. *arXiv preprint arXiv:2009.06367*.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. *Advances in Neural Information Processing Systems*, 34:14542–14554.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B Hashimoto. 2022. Diffusion-lm improves controllable text generation. *arXiv preprint arXiv:2205.14217*.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. 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.

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

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. *arXiv preprint arXiv:2005.00661*.

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.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with frank: A benchmark for factuality metrics. *arXiv preprint arXiv:2104.13346*.

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 on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2020. Discriminative adversarial search for abstractive summarization. In *International Conference on Machine Learning*, pages 8555–8564. PMLR.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2021. Societal biases in language generation: Progress and challenges. *arXiv preprint arXiv:2105.04054*.

Stefano Soatto, Paulo Tabuada, Pratik Chaudhari, and Tian Yu Liu. 2023. Taming ai bots: Controllability of neural states in large language models. *arXiv preprint arXiv:2305.18449*.

Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. 2019. "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. Association for Computational Linguistics.

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization.

Olivier Toubia, Jonah Berger, and Jehoshua Eliashberg. 2021. How quantifying the shape of stories predicts their success.

J. Towns, T. Cockerill, M. Dahan, I. Foster, K. Gaither, A. Grimshaw, V. Hazlewood, S. Lathrop, D. Lifka, G. D. Peterson, R. Roskies, J. Scott, and N. Wilkins-Diehr. 2014. Xsede: Accelerating scientific discovery. *Computing in Science & Engineering*, 16(05):62–74.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019. Topic-guided variational autoencoders for text generation. *arXiv preprint arXiv:1903.07137*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. 2020. On variational learning of controllable representations for text without supervision. In *International Conference on Machine Learning*, pages 10534–10543. PMLR.

Yelp. 2017. Yelp dataset challenge, round 8.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *Advances in neural information processing systems*, 32.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2022. A survey of controllable text generation using transformer-based pre-trained language models. *ArXiv*, abs/2201.05337.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

A GPT-3 Baseline

We construct few-shot prompts for controllable natural language generation with GPT-3. For all experiments, we use the "davinci" model, a temperature $\tau = 0.7$, and test 1000 samples per attribute. The prompts are constructed in three parts:

- **Attribute Description**: We begin all prompts by describing the attribute being used. In the case of speed, volume, and circuitousness, we find that providing both an intuitive explanation as well as a more mathematical definition leads to better results. For speed, we use the following:

  > Speed is a measure of how quickly content moves in a given text and is calculated as the distance traveled by consecutive windows of text. More specifically, we break the text into three-word chunks, compute the word embeddings of every chunk, and compute speed as the average distance between consecutive chunks.

  For volume, we use the following:

  > Volume captures the amount of information covered in a piece of text. We break the text into three-word chunks, compute the word embeddings of every chunk, and compute volume as the size of the minimum volume ellipsoid that contains all chunk embeddings.

  For circuitousness, we use the following:

  > Circuitousness measures how indirectly content is covered. We break the text into three-word chunks, compute the word embeddings of every chunk, and compute circuitousness as the sum of distances between consecutive chunks divided by the length of the shortest path. The length of the shortest path is obtained by solving the traveling salesman problem.

- **Examples**: We continue the prompt with a set of n examples to demonstrate how the attribute changes between sentences. These examples are randomly sampled from our training data, and one is shown below for speed:

  > Sentence 1: PROS: Italian hoagie was delicious. Friendly counter employee. The restaurant was clean and neat.
  >
  > Generate a sentence such that the difference in speed between sentence two and sentence one is -0.3795
  >
  > Sentence 2: Great neighborhood Italian restaurant, especially in a neighborhood overrun by Italian restaurants. Love their white pizza. Small place, but very clean with super friendly staff.

  We use $n = 3$ in our experiments because of the cost per token.

- **Prompt**: Lastly, we include the prompt, which uses three inputs: the original text, the attribute, and the target delta. The prompt is as follows:

  > Sentence 1: TEXT
  >
  > Generate a sentence such that the difference in ATTRIBUTE between sentence two and sentence one is TARGET DELTA
  >
  > Sentence 2:

B Attribute Classifier/Regressor

In this section, we provide further information about the training of the classifier and regressor used in our baseline models (*e.g.,* SSD-LM, MuCoCO, etc.). We train roberta-base[5](Liu et al., 2019) for 5 epochs or until training saturates (using an early stopping module), using an Adam optimizer with a learning rate of $5e-5$ and a batch

[5] Available at huggingface.co. We also experimented with gpt-2 and bart-base but found RoBERTa to be the most performant.

size of 128[6]. All other parameters are set based on the defaults provided by Huggingface (Wolf et al., 2019). These models contain roughly 125M parameters with 12 layers, 12 attention heads, and a hidden dimension of 768. We train each on roughly 2.6M samples from the Yelp dataset (Yelp, 2017).

For both MuCoCo and Prefix-Tuning, we utilize the regressor. We present the mean absolute error (MAE) in Table 1 under $\mathcal{D}$-MAE and normalized mean absolute error under $\mathcal{D}$-NMAE. To compute NMAE, we simply divide the MAE by the range of possible values. Generally, the MAEs are relatively small compared to the scale of Δ, indicating a strong regressor. This is also reflected by the NMAEs. We try using a classifier with binned values for the attributes but find that the regressor performs better.

For SSD-LM, we trained a classifier to predict a binned difference in attributes such that all bins contain an equal number of training samples. We try adapting the formulation to work with a regressor but find that the classifier is substantially stronger. We record both F1-score and the mean absolute error (MAE) between classes in Table 1 under $\mathcal{C}$-F1 and $\mathcal{C}$-MAE, respectively. We include both metrics to evaluate how well the model performs and roughly how incorrect predictions are. We observe that in most cases, the class labels are only at most off by one due to the low MAEs, indicating a strong classifier.

C Cev-LM Details

In this section, we provide more information about the training of Cev-LM. We encourage readers to reference our implementation for more details. All code and data are accessible at https://drive.google.com/file/d/10rwCLJ96eNP5LS_1sG-flWvXD9X4pbjO. Aside from the following parameters, our setting is identical to that of (Guu et al., 2018).

C.1 Training

In this paper, we train Cev-LM with a learning rate of $1e-3$ and batch size of 128 for a maximum of $400,000$ iterations or a maximum wall time of 24 hours, whichever came first. The volume of data used for training depends on the definition of the constrained neighborhood, but generally, the most we use for a single model is roughly 1M samples.

C.2 Model

Cev-LM is extremely configurable, allowing you to switch out the encoder/decoder architecture and change aspects of the model, including the edit vector dimension, hidden dimension, and number of layers, among other features. In this paper, we use a simple attention mechanism (Vaswani et al., 2017), but future works can easily use larger language models in place of this mechanism to improve performance. We use an edit vector dimension of 256, a hidden dimension (for the encoder/decoder) of 256, 300-dimensional GLoVE (Pennington et al., 2014) vectors. For the neural editor, we use 6 encoder and 6 decoder layers. For the inverse neural editor, we use 6 attention layers. In total, our checkpoint consists of roughly 76 million parameters (304MB).

D Prototyping Qualitative Analysis

In this section, we show some qualitative examples of the prototypes from our approach. Tables 8 to 10 include the input as well as the retrieved sample and the generated text. In many cases, we observe that the retrieved example demonstrates a strong change in feature, and Cev-LM corrects the strength of the change to ensure it is closer to the target delta.

Table 8: Examples of an input, retrieved, and edited sentence for the model trained for a change in **speed** with a target delta of 0.5. We use a tolerance $\epsilon = 0.1$ for our approach, as it empirically provided the best results in Section 6.4. The models are fed the input (*i.e.,* **Original**) and generate by applying an "edit vector" to the latent representation of the input sentence.

Retrieved & Generated Text - **Speed**
Example 1: **Input**: I will not return, terrible customer service. **Prototype**: Poorest customer service skills. **Cev-LM**: Terrible , terrible customer service.
Example 2: **Input**: The food in the restaurant can be a little pricey, but it's good and you get a lot of it. **Prototype**: The food in the restaurant is a bit pricey, but it's good. **Cev-LM**: The food is good, but it's pricey.

E Similarity Scores on Toxicity & Formality

In this section, we present the similarity scores of the baseline approach and our approach over formality and toxicity as control attributes in Table 11. We find that the scores demonstrate our generations

[6] We determine these parameters based on a simple grid search

Table 9: Examples of an input, retrieved, and edited sentence for the model trained for a change in **volume** with a target delta of 0.5. We use a tolerance $\epsilon = 0.1$ for our approach, as it empirically provided the best results in Section 6.4. The models are fed the input (*i.e.,* **Original**) and generate by applying an "edit vector" to the latent representation of the input sentence.

RETRIEVED & GENERATED TEXT - **VOLUME**

EXAMPLE 1:
Input: Overall, this was a positive experience.
Prototype: Overall, we had a positive experience and the food was good.
CEV-LM: Overall, a very positive experience - I'll definitely be back.

EXAMPLE 2:
Input: The menu had lots of options.
Prototype: The menu leaves you with lots of options that you can customize.
CEV-LM: The menu here has lots of options that we want to try.

Table 10: Examples of an input, retrieved, and edited sentence for the model trained for a change in **circuitousness** with a target delta of 0.5. We use a tolerance $\epsilon = 0.1$ for our approach, as it empirically provided the best results in Section 6.4. The models are fed the input (*i.e.,* **Original**) and generate by applying an "edit vector" to the latent representation of the input sentence.

RETRIEVED & GENERATED TEXT - **CIRCUITOUSNESS**

EXAMPLE 1:
Input: This is my favorite Szechuan restaurant in town.
Prototype: This is my favorite Szechuan restaurant, and probably my favorite Szechuan restaurant ever.
CEV-LM: This is my favorite Szechuan restaurant in town and probably in the world.

EXAMPLE 2:
Input: The menu has a little bit of everything.
Prototype: The menu has a little bit of everything that you could want.
CEV-LM: The menu has a little bit of this and a little bit of that.

stay on topic, further indicating the robustness of our approach on more standard control attributes.

F Tolerance Tuning

We measure the impact of ϵ on training in Table 12. Too low of a tolerance value leads to overfitting, indicated by a closer Δ to the target and poor performance in the test-time similarity metrics. As mentioned before, controlled edit vector perturbations to edit vectors improves similarity metrics at the cost of Δ, which implies that the approach helps to combat overfitting. At $\epsilon = 0.05$ and $\epsilon = 0.1$, we see that perturbation is generally not helpful, but at tolerance $\epsilon = 0.2$, the perturbation approach leads to a higher Δ. Note that the BLEU scores are slightly different as n-grams are weighted differently in the code for the edit-then-prototype architecture (Guu et al., 2018) and in NLTK (Bird and Loper, 2004).

G Controlled Edit Vector Perturbation

In Table 13, we present the results of the neighborhood creation and neighborhood creation + perturbation approaches. The Baseline shows the out-of-the-box edit-then-prototype model, which has little impact on the target attribute and provides a rough baseline of the similarity metrics. Again, we find that as the target delta increases, the MAE increases and similarity scores decrease. This phenomenon is attributed to the data distribution and is expanded on in Section 6.5. We observe that perturbation is sometimes helpful in decreasing MAE, especially in the case of volume. However, this behavior is inconsistent across speed and circuitousness and warrants further exploration.

Table 11: BLEU and BERT (F1) Scores for speed, volume, and circuitousness across all approaches for different target deltas. The scores are averaged across three training runs (inference runs for GPT-3). We use a tolerance $\epsilon = 0.1$ for all of our approaches, as it empirically provided the best results in Section 6.4.

Metric	**Toxicity**			**Formality**		
Target Delta	**0.1**	**0.5**	**0.9**	**0.1**	**0.5**	**0.9**
BERTSCORE - BENCHMARK APPROACHES						
GPT-3 (Brown et al., 2020)	0.857	0.869	0.866	0.851	0.851	0.862
MuCoCO (Kumar et al., 2021)	0.763	0.771	0.774	0.763	0.759	0.760
SSD-LM (Han et al., 2022)	0.769	0.767	0.769	0.763	0.760	0.747
Prefix Tuning (Li and Liang, 2021)	0.833	0.827	0.823	0.843	0.836	0.834
BERTSCORE - OUR APPROACHES						
CEV-LM	0.848	0.827	0.837	0.842	0.842	0.845
BLEU - BENCHMARK APPROACHES						
GPT-3 (Brown et al., 2020)	0.231	0.250	0.291	0.219	0.273	0.269
MuCoCO (Kumar et al., 2021)	0.305	0.268	0.293	0.257	0.276	0.219
SSD-LM (Han et al., 2022)	0.294	0.343	0.346	0.314	0.325	0.326
Prefix Tuning (Li and Liang, 2021)	0.246	0.269	0.275	0.231	0.217	0.194
BLEU - OUR APPROACHES						
CEV-LM	0.320	0.316	0.295	0.342	0.334	0.265

Table 12: Evaluation metrics (BLEU (Papineni et al., 2002) and BERTScore (Zhang et al., 2019)) and strength of control on Δ for the trained models (ideally, $\Delta = 0.5$) for speed. The scores are averaged across three training runs with different seeds. We train a baseline edit-then-prototype model (Guu et al., 2018), as well as CEV-LM ($\mathcal{N}$-ONLY) and CEV-LM with different tolerances (ϵ). We record both train and test BLEU to demonstrate overfitting with lower tolerances.

MODEL	DELTA	TRAIN BLEU	TEST BLEU	BERTSCORE
EDIT-THEN-PROTOTYPE	0.0113	**0.6691**	**0.5679**	0.9327
CEV-LM ($\mathcal{N}$-ONLY): $\epsilon = 0.05$	0.4559	0.8057	0.4266	0.9326
CEV-LM ($\mathcal{N}$-ONLY): $\epsilon = 0.1$	0.4558	0.7146	**0.5747**	0.9340
CEV-LM ($\mathcal{N}$-ONLY): $\epsilon = 0.2$	0.4405	0.5994	0.5628	0.9355
CEV-LM: $\epsilon = 0.05$	0.4279	0.5709	0.5218	0.9329
CEV-LM: $\epsilon = 0.1$	0.4455	0.6375	0.5400	0.9386
CEV-LM: $\epsilon = 0.2$	**0.4596**	**0.6751**	**0.5679**	0.9334

Table 13: Evaluation metrics (BLEU (Papineni et al., 2002) and BERTScore (Zhang et al., 2019)) and strength of control on Δ for the trained models. The scores are averaged across three training runs, and we omit variance due to negligible values. We train a baseline model (Guu et al., 2018) (Baseline) and multiple models across various target deltas for all nonstandard control conditions (*e.g.,* speed, volume, circuitousness) to show training has a significant impact on the achieved control.

	CEV-LM ($\mathcal{N}$-ONLY)					**CEV-LM**				
	TARGET DELTA	DELTA	MAE	BLEU	BERT-F1	TARGET DELTA	DELTA	MAE	BLEU	BERT-F1
	Baseline	0.0468	-	0.3185	0.9327	Baseline	0.0468	-	0.3185	0.9327
	0.125	0.1105	0.0145	0.3399	0.9351	0.125	0.1189	0.0061	0.3261	0.9350
Speed	0.5	0.4558	0.0442	0.3276	0.9340	0.5	0.4355	0.0645	0.3123	0.9386
	2.0	1.7594	0.2406	0.3051	0.9291	2.0	1.7897	0.2103	0.2944	0.9281
	4.0	3.6213	0.3787	0.2463	0.9188	4.0	3.4657	0.5343	0.2736	0.9230
	Baseline	0.0011	-	0.3185	0.9327	Baseline	0.0011	-	0.3185	0.9327
Volume	0.125	0.1106	0.012	0.3296	0.9380	0.125	0.1130	0.012	0.3038	0.9351
	0.5	0.4415	0.0585	0.2682	0.9320	0.5	0.4535	0.0465	0.2653	0.9314
	2.0	1.7521	0.2479	0.2869	0.9244	2.0	1.8466	0.1534	0.2518	0.9208
	Baseline	-0.0022	-	0.3185	0.9327	Baseline	-0.0022	-	0.3185	0.9327
Circuitousness	0.125	0.0723	0.0527	0.2483	0.9271	0.125	0.0664	0.0586	0.2902	0.9306
	0.5	0.4217	0.0783	0.2680	0.9109	0.5	0.4207	0.0793	0.2755	0.9089
	1.0	0.7893	0.2107	0.2479	0.9082	1.0	1.0519	0.0519	0.1622	0.8354