Constrained Sampling from Language Models via Langevin Dynamics in Embedding Spaces

Sachin Kumar♣, Biswajit Paria♥, Yulia Tsvetkov♠

♣Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA
♥Machine Learning Department, Carnegie Mellon University, Pittsburgh PA
♠Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle WA

{sachink,bparia}@cs.cmu.edu, yuliats@cs.washington.edu

Abstract

Large pre-trained language models are well-established for their ability to generate text seemingly indistinguishable from humans. In this work, we study the problem of constrained sampling from such language models. That is, generating text that satisfies user-defined constraints. Typical decoding strategies which generate samples left-to-right are not always conducive to imposing such constraints globally. Instead, we propose µCoLA—a sampling procedure that combines the log-likelihood of the language model with arbitrary differentiable constraints into a single energy function; and generates samples by initializing the entire output sequence with noise and following a Markov chain defined by Langevin Dynamics using the gradients of this energy. We evaluate our approach on text generation with soft and hard constraints as well as their combinations with competitive results for toxicity avoidance, sentiment control, and keyword guided generation.

1 Introduction

Contemporary language models (LMs) based on transformers (Vaswani et al., 2017) trained using large-scale web text corpora (Radford et al., 2019; Raffel et al., 2020; Brown et al., 2020) have resulted in systems capable of generating impressively realistic text. However, despite being highly fluent, they lack one key aspect of human-generated language: controllability. For example, LMs can be difficult to control for intended content of the generated text such as, specific word choices, or its properties, such as politeness or friendliness. And this often leads them to produce biased, offensive and non-factual outputs (Gehman et al., 2020a; Pagnoni et al., 2021). To address this issue, in this work we study controlled text generation, i.e, sample text from LMs such that it satisfies user-defined constraints.

Early solutions for this problem focused on retraining or finetuning the LMs (Keskar et al., 2019; Gururangan et al., 2020; Chan et al., 2021) which, with increasing scale, is becoming computationally infeasible. As a result, most recent work has shifted focus towards developing decoding approaches while keeping the underlying LM fixed (Dathathri et al., 2020; Yang and Klein, 2021a; Krause et al., 2020a; Liu et al., 2021a; Lu et al., 2021; Pascual et al., 2021; Liu et al., 2021b). The dominant paradigm in this space is based on left-to-right, also known as, autoregressive decoding which modifies the output vocabulary distribution at each generation step to enable control using heuristics or auxiliary models such as classifiers or language models. While effective for certain tasks, most of these approaches hardly generalize beyond a specific number (usually one) or types of constraints.

1Code available at: https://github.com/Sachin19/mucoco/tree/sampling
and LMs. They condition and constrain based on just the generated left context without considering the entire output. And importantly, while the goal is to sample from the LM distribution, these approaches end up modifying said distribution to satisfy constraints. For example, if the goal is generate a polite translation of a source sentence using a translation model, the output should be a sample from the model in that it should not forgo meaning in lieu of politeness.

In contrast, in this work, we present a non-autoregressive constrained sampling algorithm from LMs. Instead of sampling tokens sequentially, we initialize the entire output sequence and iteratively refine it to incorporate desired constraints by taking global sequence context into account—by following a Markov chain (Brooks et al., 2011). First, we aggregate the constraints and the LM likelihood into a single “energy” function, which reduces our objective to searching for low energy solutions of this function (Kumar et al., 2021; Qin et al., 2022; Mireshghallah et al., 2022). Due to the large search space of text sequences, this search can be intractable for common Monte Carlo Markov Chain (MCMC) methods (Sokal, 1997).

To solve this issue, we propose using Langevin Dynamics—an MCMC algorithm which uses the gradients of the energy function with respect to the output text to perform iterative updates akin to gradient descent; but with an additive noise at each update (Grenander and Miller, 1994; Parisi, 1981; Welling and Teh, 2011; Gelfand and Mitter, 1991).

Since text is discrete, computing such gradients is infeasible. Prior work involving such gradient computation approximates this computation by maintaining soft-representations or “probabilities” of each output token over the entire vocabulary (Hoang et al., 2017; Qin et al., 2020; Kumar et al., 2021; Song et al., 2020). Due to large vocabularies in LMs, such a relaxation is computationally expensive and memory consuming. In fact, it can be infeasible to generate long sequences with this setup (as they do not fit on standard GPUs; as we show in §5). Instead, we propose a simple but effective change: represent each token and apply gradient updates, on a much smaller vector—dense word embeddings, which are readily available in the first layer of a typical neural network based LM via an embedding table.

In summary, representing each token in the output sequence as a dense embedding vector (§3.1), we define an energy function encompassing desired constraints as functions of these vectors and perform Monte Carlo updates following Langevin Dynamics (§3.2). Finally, to operationalize the energy function, we represent each constraint function to be less than a predefined threshold (§3.3) and write the energy function as a Lagrangian with language model likelihood as the primary objective (Kumar et al., 2021). This setup allows mixing any number of constraints of varying scales without any need of tuning weights for each new constraint. We call this algorithm MuCoLA for sampling with multiple constraints from LMs using Langevin Dynamics (see figure 1).

To show the generality of MuCoLA, we validate it on four tasks selected from recent work with different kinds of constraints, varying lengths of output sequences, and different language models: (a) reducing toxicity in prompted outputs ($§4.2$), (b) controlling for sentiment in generated outputs, (c) generating text containing specific keywords/phrases, and finally (d) a preliminary study on combination of ensuring sentiment with predefined keywords in the generated outputs. Evaluating on both quality and diversity, we show performance improvement of this method over competitive baselines.

## 2 Background: Constrained Sampling from Language Models

Let \( P(y|x; \theta) \) model the conditional probability distribution of an output token sequence \( y = (y_1, \ldots, y_N) \), given an optional input token sequence \( x = (x_1, \ldots, x_M) \) where \( x_m, y_n \in \mathcal{V} \), the vocabulary. This distribution can be parameterized (via \( \theta \)) using any differentiable architecture (Vaswani et al., 2017; Hochreiter and Schmidhuber, 1997) and trained with any loss function (Edunov et al., 2018; Kumar and Tsvetkov, 2019). It can be trained in a decoder-only fashion like traditional language models (Radford et al., 2019; Brown et al., 2020) where at generation time, \( x \) would serve a prompt and \( y \) as its continuation, or in an encoder-decoder fashion for tasks such as machine translation or summarization.

Traditionally, the decoder consists of a input layer \( E \) which first converts each discrete \( y_n \) to a dense vector \( e_{y_n} \), via an embedding table lookup (also referred to as a non-contextual embedding). This vector is then fed through a series of neural network layers (e.g., transformers or LSTMs) to obtain
a hidden state (or a contextual token embedding) after which an output embedding layer projects it back to vocabulary space using the softmax operation. To reduce number of trainable parameters, most modern text generation systems usually share the input and output embedding tables (Press and Wolf, 2017).

Given \( x \), decoding from such a model involves finding outputs \( y \in \mathcal{Y} \) which admit a high probability under \( P \). Since searching through the space of all possible output sequences \( \mathcal{Y} \) is intractable, most decoding algorithms factorize \( P \) over each token \( y_n \), where the output is generated left-to-right, with the output token in step \( n \) being fed to the input at step \( n + 1 \). It typically involves search or sampling strategies like beam search, ancestral sampling, top-k sampling (Fan et al., 2018), or nucleus sampling (Holtzman et al., 2020), among others.

In this work, we are interested in constrained sampling—finding output sequences \( y \) that have a high probability under \( P \) while minimizing a given set of constraint functions: \( \{ f_1, \ldots, f_C \} \). We assume that each \( f_i : ([x], y) \to \mathbb{R} \) is defined such that a lower value of \( f_i \) implies that the output better satisfies the constraint. For example, to constrain the outputs to only non-toxic continuations for a given prompt \( x \), we define a classifier \( p_{\text{toxic}}(y) \) which predicts the output toxicity probability, with lower probability implying lower toxicity. We assume all \( f_i \) are differentiable.

Enforcing these constraints in an autoregressive (i.e., left-to-right) decoding strategy like beam search or sampling is challenging, since the constraints are defined conceptually on the whole output sequence and are hard to evaluate accurately only on the generated prefix (Yang and Klein, 2021b; Liu et al., 2021a). With multiple constraints, their balancing and satisfaction becomes challenging. Prior work, thus, explored non-autoregressive controlled generation (Hoang et al., 2017; Kumar et al., 2021), using constrained optimization over \( y \)—finding a single output \( y \) which maximizes \( P \) given the constraints by performing gradient descent on the outputs \( y \). This involves (1) representing the constrained optimization problem as a single objective (often referred to as an energy function \( \mathcal{E}(y) \), discussed in §3.3); and (2) relaxing the discrete outputs \( y \) to continuous approximations such that gradient descent is feasible. In previous works, the latter is achieved by creating a soft-

representation of \( y \), \( \tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_N) \) where each \( \tilde{y}_n \in \mathbb{R}^{|V|} \) is a simplex (or “logits” which are converted to a simplex using softmax) over the target vocabulary \( \mathcal{V} \), representing the probability of the \( n \)-th token in the sequence. We refer to these methods as gradient-based decoding. Representing the decoding objective as \( \min_y \mathcal{E}(\tilde{y}) \) and initializing \( \tilde{y} \) with \( \tilde{y}_0 \), it is updated as

\[
\tilde{y}^t = \tilde{y}^{t-1} - \eta \nabla_y \mathcal{E}(\tilde{y}^{t-1}),
\]

where \( \eta > 0 \) denotes the step size. In this process, the underlying LMs (and functions \( f_i \)) remain fixed and are used to provide gradients to the sequence \( \tilde{y} \). After performing multiple steps of this gradient descent discrete text can be extracted from \( \tilde{y} \) using different heuristics (Kumar et al., 2021; Qin et al., 2020; Song et al., 2020). This formulation has been studied in various generation settings in prior work with different instantiations of \( \tilde{y} \) and \( \mathcal{E}(y) \).

However, this setup is deterministic and does not facilitate sampling.\(^2\) In addition, representing each token with a vector of size \( |V| \) can be computationally very expensive and difficult to fit into commonly used GPUs for long sequences (with more than \( \sim 20-30 \) tokens; §5).

3 Constrained Sampling via Langevin Dynamics in Embedding Space

To enable efficient gradient-based sampling from LMs, in this work, we modify this framework to (1) generate multiple samples from \( P \) instead of optimizing for only one deterministic output, (2) optimize for much smaller intermediate token representations as opposed to their distribution on the entire vocabulary. First, we describe our proposed way to representing tokens followed by how they can facilitate sampling.

3.1 (Projected) Gradient Descent on Word Embeddings

Instead of relaxing each target token \( y_n \) as a soft representation over the vocabulary \( \tilde{y}_n \in \mathbb{R}^{|V|} \), we represent it as \( \hat{e}_n \in \mathbb{E} \). Here \( \mathbb{E} \) denotes the embedding table of the underlying language model containing \( |V| \) vectors of size \( d < |V| \). We denote this sequence of embeddings as \( \hat{e} = \{ \hat{e}_1, \ldots, \hat{e}_N \} \). At an update step \( t \), instead of feeding each \( \tilde{y} \) to the model(s) (which are then transformed to an embedding to be fed to the first layer), we directly

\(^2\)While initialization can be used to add randomness to this algorithm, we find that it has little to no effect on diversity.
feed each $\tilde{e}$ to the first layer to compute the energy function, now defined as a function of embeddings instead of tokens. In case of deterministic minimization (similar to (1)), these vectors are updated as,

$$
\tilde{e}^t = \text{Proj}_E(\tilde{e}^{t-1} - \eta \nabla_{\tilde{e}} \mathcal{E}(\tilde{e}^{t-1}))
$$

(2)

where, $\text{Proj}_E(\tilde{e}) = \arg \min_{e \in E} \|e - \tilde{e}\|_2$ denotes a projection operation on the embedding table $E$. In other words, after every gradient step, we project each updated vector back to a quantized space, that is the embedding table using Euclidean distance as the metric. This projection is done to prevent adversarial solutions. After the optimization is complete, discrete text can be easily obtained by projection, that is the token indices corresponding to each $\tilde{e}_n$ in the embedding table $E$. This formulation yields the following benefits: (1) For a sequence of length $L$, at any optimization step $t$, it only maintains (and computes gradients with respect to) $L \times d$ parameters, as opposed to $L \times |V|$. This enables us to store much longer sequences in a GPU as compared to the storing $y$. (2) this formulation provides a natural way to define hard rule-based constraints based on keywords or phrases (discussed in more detail in §4.4), and, finally (3) it yields a natural way to extend it to generate samples.

### 3.2 Gradient based Sampling using Langevin Dynamics

The minimization described above can be very easily extended to a sampling procedure by modifying the gradient descent in (2) to Langevin Dynamics (Welling and Teh, 2011; Gelfand and Mitter, 1991) as follows,

$$
\tilde{e}^t = \text{Proj}_E(\tilde{e}^{t-1} - \eta \nabla_{\tilde{e}} \mathcal{E}(\tilde{e}^{t-1}) + \sqrt{2\eta \beta z^t})
$$

(3)

Here $z^t \sim \mathcal{N}(0, I_{d})$ and $\beta > 0$ is a hyperparameter signifying variance of the noise. Langevin Dynamics provides a Markov Chain Monte Carlo (MCMC) method to sample from a distribution using only the gradient of its logarithm. That is, if we define a distribution as $Q(y) \propto \exp(-\mathcal{E}(y))$, its logarithm leads to the update specified in (3). This method is often used for non-convex optimization for training neural networks (Welling and Teh, 2011) due to its ability to escape local minima due to added noise and converge towards the global minima. In this work, we adapt it for inference.

Intuitively, by adding noise at every gradient step, this procedure intends to find outputs $y$ that do not exactly minimize $\mathcal{E}$ but remain in the vicinity of the minima. In other words, it finds outputs which admit high probability under the distribution $Q(y)$. This process begins with an exploration phase which is controlled by $\beta$. With a high value of $\beta$, the noise term is large leading to big updates. By gradually annealing such that $\beta \to 0$, as $t \to \infty$, this process converges to a sample from $Q(y)$. More details of the implementation of annealing schedule can be found in §4.1.

A similar noise can also be applied directly to the soft-token representations in (1) as explored in Qin et al. (2022). However, as we discuss in §5, our formulation with its smaller parameter size allows generating longer sequences. In addition, considering logits as soft-representations (followed by softmax) has shown to result in slow mixing, that is, it takes much longer to converge as empirically shown in Hoang et al. (2017) (and also observed in Qin et al. (2022)). On the other hand, considering the simplex itself (Kumar et al., 2021; Hoang et al., 2017) as soft-representations is not compatible with gaussian noise and can lead to undesirable behavior (Patterson and Teh, 2013).

### 3.3 Representing the energy function

A straightforward way to represent the energy is with a linear combination as $\mathcal{E}(y) = \sum_{i=1}^{C} \lambda_i f_i(y, [x]) - \lambda_{C+1} \log P(y)$, with pre-defined weights $\lambda_1, \ldots, \lambda_{C+1}$. This has been used in prior work in controlled text generation (Hoang et al., 2017; Qin et al., 2020, 2022). This formulation has two issues: (1) Linear weights ($\lambda_i$) can be hard to define and tune for different $f_i$, and especially difficult when $f_i$’s lie on different scales, and more importantly, (2) Defining the energy function in this manner modifies the original goal, which is to sample from the language model $P$, not from a modified distribution, $Q \propto \exp(-\mathcal{E}(y))$. To alleviate these issues, we consider the following formulation

$$
y \sim P(y|x; \theta), \text{ subject to } f_i([x], y) \leq \epsilon_i, i \in \{1, \ldots, C\},
$$

---

3Several prior works (Belinkov and Glass, 2019) have shown that neural-network based models are not robust to change in input space. We observed this phenomenon in our preliminary experiments where, without any projection, most low energy solutions were found to be garbled text.

4The normalization term in $Q(y)$ vanishes as its gradient with respect to $y$ is 0.
where each threshold $\epsilon_i$ is a hyperparameter. As we discuss in more detail in §4, these thresholds can be flexibly defined for most kinds of constraints. For example, instead of merely trying to reduce $p_{\text{TOXIC}}(y)$ we can set it as $p_{\text{TOXIC}}(y) < 0.1$. Given this formulation, we define the energy function as a Lagrangian,

$$E(y) = -\log P(y|x, \theta) - \sum_{i=1}^{u} \lambda_i (\epsilon_i - f_i([x], y))$$

(4)

Here $\lambda_i \geq 0$ are Lagrangian multipliers and dynamically updated at each step. We follow the gradient of $E$ downwards for the $\tilde{e}$ (as described in (2)) and upwards for the multipliers (gradient ascent without any noise) while making sure that the multipliers remain positive.

$$\lambda_i^t = \max(0, \lambda_i^{t-1} + \alpha \nabla \lambda_i E(y))$$

where $\alpha > 0$ is the step size for ascent. Intuitively, if a constraint is not satisfied, the term $(\epsilon_i - f_i(\cdot))$ would be negative and $\lambda_i$ would keep increasing making $E$ high. On the other hand if all the constraints are satisfied these values gradually decrease to 0 making $E(y) = -\log P(y)$ making the final output a sample from the desired distribution $P$. In our implementation, we consider a damped version this process to improve stability, the details of which can be found in Kumar et al. (2021).

The final decoding algorithm we used in our experiments is described in algorithm 1.

**Energy as a function of embeddings** Performing gradient updates with respect to $\tilde{e}$ requires that all objectives be defined as functions of $\tilde{e}$ as opposed to the output sequence $y$. In addition, $P(y|x; \theta), f_1(y), \ldots, f_C(y)$ must share the same input embedding table (as that of the language model $P$). We discuss in §4 how this can achieved for different kinds of constraint functions $f_i$. First, we describe how to compute the primary objective $-\log P(y|x; \theta)$ and its gradients with respect to $\tilde{e}$. Typically, the (log) probability of $y$ is computed using a language model $P$ by factorizing it over each output, as $\log P(y|x) = \sum_{n=0}^{L-1} \log P(y_{n+1}|y_{1:n}, x)$. For each decoding step $n + 1$, the model receives the token $y_n$, which is converted to $e_n$ via the embedding table (E) lookup. This embedding is passed through the neural network layers to obtain a hidden vector $h_n$. Since the input and output embedding tables in most modern LMs is shared (Radford et al., 2019; Raffel et al., 2020; Lewis et al., 2020; Brown et al., 2020)\(^5\), the softmax probability is computed using $h_n$ as,

$$P(y_{n+1}|y_{1:n}, x) = \frac{\exp(h_n^T e_{n+1} + b_{n+1})}{\sum_{j=1}^{|V|} \exp(h_n^T e_k + b_k)},$$

where $b_n$ are optional bias terms. By replacing $e_{n+1}$ with $\tilde{e}_{n+1}$, we convert the above probability to $P(\tilde{e}_{n+1}|\tilde{e}_{1:n}, x)$. For each position $n + 1$, $\tilde{e}_{n+1}$ receives gradients, (a) directly from $-\log P$ function (it appears in both the numerator and the denominator), and (b) through $h_{n+1}$ via backpropagation through the network layers (See figure 1 (left)).

### 4 Experimental Setup and Results

We evaluate MuCoLa on four constrained generation tasks. These tasks are selected based on defining different kinds of constraints for which prior work designed specialized training or decoding mechanisms which cannot be generalized beyond those tasks or language models. Our main contribution is generating diverse samples which conform to the language model $P$ as well as can satisfy user defined arbitrary combination of constraints for which fine-tuning is generally infeasible and tuning weights of each constraint is cumbersome.

#### 4.1 Implementation Details

For a pre-defined sentence length $L$, we initialize the token representation for each step $\tilde{e}_1, \ldots, \tilde{e}_L$ using token embeddings randomly sampled from the target vocabulary $V$.\(^6\) For all our experiments, we run the Langevin Dynamics simulation for a maximum of 250 iterations unless specified otherwise.

**Noise Schedule** The amount of noise in each update is controlled by $\beta$ (see (3)) which represents the variance of the noise term. We initialize $\beta$ with 5.0 and decrease it to 0.05 in a geometric progression for 100 steps after which we keep it constant at 0.05 for the remaining 150 steps. The range of

---

\(^5\) Even if the embedding tables are not shared, this loss may be computed and optimized using vectors from the output embedding table as parameters without any significant loss in performance.

\(^6\) We also tried other initialization strategies like initializing with zeros, or outputs of nucleus sampling or greedy decoding but did not find it to have any significant effect on the final output
we start with a step-size $\eta$. We define a schedule on the step-size as follows:

$$\lambda_i \text{ is empirically defined as } s^{-2}.$$  

We initialize each of the multipliers $\lambda_i$ with 0, update the multipliers via gradient ascent every 10 steps using the step-size 1.0. In addition, if the sequence stops updating at a certain iteration (as described above) and $i$-th constraint is not satisfied, we update $\lambda_i$ at every iteration till the sequence starts updating again. This schedule prevents fluctuation in the multiplier values when the noise is high in the early iterations and the sequence has not converged to anything fluent while still allowing updates when required (Platt and Barr, 1988; Paria et al., 2020).

## 4.2 Toxicity Avoidance

Prior work have shown that large pre-trained LMs are at risk of producing toxic content even when given innocuous prompts (Sheng et al., 2019; Gehman et al., 2020b). In this experiment, we apply MUCoLA to steer the LM outputs away from such behavior. Given a neutral prompt, our goal is to generate a continuation which is not toxic.

We use GPT2-Large (Radford et al., 2019) as
our underlying LM and define a single constraint
based on a binary classifier $p_{\text{TOXIC}}$ to measure
toxicity. This classifier is trained on a dataset of
human-annotated comments from the Jigsaw Unin-
tended Bias in Toxicity Classification Kaggle chal-
lenge. The dataset has $\sim$160K toxic comments and
$\sim$1.4M nontoxic comments. This task setup was
introduced in Liu et al. (2021b) and details of how
this dataset is created can be found in the paper. We
train this classifier by finetuning roberta-base (Liu
et al., 2019) by simply replacing its embedding ta-
ble with the one from GPT2-Large. Please refer to
Appendix B for more details. This classifier pre-
dicts the toxicity probability of an output sequence
$y$. We use the constraint $p_{\text{TOXIC}}(y) \leq 0.01$ in this
experiment. To improve its gradient profile, we
modify it as $\log(p_{\text{TOXIC}}) \leq \log(0.01)$.

4.2.1 Baselines

We follow the evaluation setup defined in Liu
et al. (2021b) and use a random sample of 10K
nontoxic prompts from the RealToxicityPrompts
dataset (Gehman et al., 2020b) where without any
constraints, the user might receive harmful output
from the LM. For each prompt, we generate 25
samples for length 20 tokens each. We consider
two main baselines:

**FUDGE** (Yang and Klein, 2021a) uses a
“future-aware” classifier to modify output proba-
bilities at every step in a left-to-right decoding setup.
This classifier is trained to predict the ground truth
label for every prefix of the training corpus. We
train this classifier on the same dataset we use to
train our constraint classifier. And use the rec-
commended hyperparameters in Yang and Klein
(2021b) for decoding with top-$k$ sampling with
$k = 10$.

**DExperts** (Liu et al., 2021b) proposes to use
two auxiliary language models (one expert–a non-
toxic language model and one anti-expert–a toxic
language model to modify the output logits at every
step). These LMs are trained using the described
dataset by finetuning GPT2-Large.

In addition to these baselines, we also report
results on other baselines as reported in Liu et al.
(2021b). We refer the reader to the paper for a
detailed discussion on these methods.

4.2.2 Evaluation

We evaluate the generated samples on three dimen-
sion: (1) **Toxicity**, using the toxicity score from
Perspective API. Following prior work (Gehman
et al., 2020b; Liu et al., 2021b), we report the max-
imum toxicity score over 25 samples per prompt
averaged over the number of prompts, and the em-
pirical probability of generating a continuation with
toxicity $> 0.5$ at least once over the 25 generations.
(2) **Fluency**, measured by mean perplexity of the
continuations measured using GPT2-XL. Since the
objective is to generate samples from the LM, we
rank different methods not by their absolute per-
plexity, but its difference from the perplexity of
unconstrained text. Additionally, we also report a
grammaticality score: the fraction of outputs pre-
dicted by a classifier trained on CoLA (Warstadt
et al., 2019) as fluent. (3) **Diversity**, measured by
computing the mean number of distinct n-grams in
each set of samples, normalized by the length of
text (Li et al., 2016). We report this for $n = 1, 2, 3$
following prior work.

Results As shown in table 1, MuCoLA outper-
forms or matches strong baselines on toxicity, in-
cluding a strong baseline DEXPERTS which is
specifically designed for binary constraints, on tox-
icity probability. In addition, our method is closest
in perplexity to unconstrained generation, while
maintaining grammaticality as well as diversity
of baseline methods. We attribute this improve-
ment to the fact that after the constraints are satis-
fied, the energy function in MuCoLA reduces to
$-\log P(y)$, the original function we intend to sam-
ple from, whereas in the baselines, the underlying
probability distribution (or the energy function) is
modified to achieve control.

4.3 Sentiment Controlled Generation

Given a prompt $x$, the goal of this task is to generate
continuations $y$ using an LM with a desired sen-
timent/polarity (either positive or negative). With
GPT2-Large as the underlying LM, we consider
different ways to represent the sentiment constraint
considered in prior work on controlled text genera-
tion. To understand the effect of sources of training
data, we train two versions of each of them on two
data sets: SST-2 corpus (Socher et al., 2013) con-
taining $\sim$4K examples in Movie reviews for each
class; and Yelp polarity corpus containing $\sim$280K
examples for each class containing a mixed domain
of reviews.

**Discriminative Classifiers** Similar to toxicity
avoidance (§4.2), we train a binary classifier
which predicts the probability of positive sentiment,
Table 1: Results for toxicity avoidance (§4.2). We evaluate on three axes: (1) Toxicity–Avg. Max. Toxicity and Toxicity Prob.: lower the better. (2) Fluency–GPT2-XL Perplexity, closer the value to unconstrained outputs (GPT2: 38.6), the better; CoLa accuracy: higher the better, and (3) Diversity (Dist-1,2,3): higher the better. The best values in each column are highlighted in **bold**. While our method improves or performs on par with baselines on toxicity metrics, we obtain substantial improvements on perplexity.

| Approach  | Toxicity | Fluency | Diversity |
|-----------|----------|---------|-----------|
| GPT-2     | 0.527    | 25.45   | 0.58      |
| PPLM      | 0.520    | 32.58   | 0.58      |
| DAPT      | 0.428    | 31.21   | 0.57      |
|GEDi       | 0.363    | 60.03   | 0.62      |
| DEXPERTS  | **0.302**| 38.20   | **0.56**  |
| FUDGE     | 0.437    | 12.97   | 0.47      |
| MuCoLa    | **0.308**| 29.92   | **0.55**  |

$p_{\text{SENTIMENT}}(\cdot)$ on the full sequence (including the prompt). Again, we train both of these classifiers by finetuning Roberta-base with GPT2-Large embeddings (more details in the Appendix Appendix B). We use the constraint $p_{\text{POSITIVE}}([x, y]) < 0.01$ or $p_{\text{POSITIVE}}([x, y]) > 0.9$ depending on the desired polarity (in practice we use their logarithm). We refer to it as MuCoLa-DISC. We also test a setup which uses both of the classifiers as two separate constraints MuCoLa-TWO-DISC.

**Generative Classifiers** Prior work has shown that discriminative classifiers can be fragile to domain shift or adversarial examples (Yogatama et al., 2017; Krause et al., 2020b). Hence, we also consider a second class of generative classifiers trained as class conditional LMs. They model $p(\cdot|\text{SENTIMENT})$ where SENTIMENT can be either positive or negative. Intuitively, these models are required to explain every word in the input, potentially amplifying the class signal and improving robustness (Min et al., 2021). Furthermore, we train them in two ways: first, following GEIDI (Krause et al., 2020a), we concatenate the word "positive" in front of every positive example and "negative" in front of every negative example and train an LM by finetuning GPT2-Large (more details in the Appendix Appendix B). We call this setup MuCoLa-GEN. And second, we follow DEEXPERTS (Liu et al., 2021b), and train two separate LMs, one for positive class and one for negative class. We call this setup MuCoLa-DEEXPERTS. Both both cases, we set the constraint as $p([x, y]|\text{SENTIMENT} = \text{POSITIVE}) > p([x, y]|\text{SENTIMENT} = \text{NEGATIVE})$ for positive sentiment and vice-versa for negative (again, we realize the constraints in log-space for better gradients).

**Prompt based Classifier** Finally, we also consider a “classifier” without any fine-tuning. Motivated by recent work on prompt-based classification, we consider a very simple constraint using the underlying LM itself: $P(x, y|\text{“This is amazing”}) > P(x, y|\text{“This is terrible”})$. That is, we simply append the sequence “This is amazing (terrible)” in front of the output sequence and predict the probability of the rest of the sequence. We refer to this as MuCoLa-PROMPT. We use the same constraint format here as in generative classifiers.

### 4.3.1 Baselines

We use a dataset of 15 prompts from Dathathri et al. (2020) and generate 20 samples per prompt of length 12, 20, and 50 for both positive and negative polarity. We consider the same baselines as in §4.2.

**Domain Adaptive Pretraining (DAPT)** (Gururangan et al., 2020) Here we use the fine-tuned LMs used in DEEXPERTS directly to generate positive or negative continuations.

**FUDGE** (Yang and Klein, 2021b) : We train classifiers on SST-2 and Yelp dataset in a similar fashion as described in §4.2 and decode with recommended generation hyperparameters.
GeDi (Krause et al., 2020a) uses a class-conditioned LM to modify output token probabilities via Bayes’ rule. We use two versions of this baseline with class conditional LMs trained on the described datasets (SST-2 and Yelp).

DExperts (Liu et al., 2021b) : We use two language models trained on the positive and negative corpus (for both SST-2 and Yelp) and decode with recommended hyperparameters.

4.3.2 Evaluation
We evaluate the generated samples along three axes: (1) Sentiment control measured as positive sentiment accuracy of the output text using external classifiers. Sentiment classifiers are trained on human-written texts in specific domains, which can make them fragile while evaluating machine generated text, and prone to be fooled by adversarial solutions (Song et al., 2020). Hence, we report accuracies measured three different classifiers, two of them used in prior work—(a) c1: distilbert (Sanh et al., 2019) finetuned on SST-2 data, used in (Liu et al., 2021b), (b) c2: bert-base (Devlin et al., 2019) finetuned on Yelp Polarity corpus used in Mireshghallah et al. (2022), and (c) c3: SieBERT (Heitmann et al., 2020) finetuned on 15 different polarity datasets. We also measure (2) Fluency, and (3) Diversity both following the same setup as in toxicity avoidance (§4.2).

Results We report a subset of results of this experiment in table 2 for outputs of length 20 (remaining results can be found in Appendix C). We observe a significant variance in sentiment control accuracies (c1, c2 and c3) where constraints trained on SST-2 perform worse on the evaluator trained on Yelp (c2) and vice versa for all methods. The third evaluator (c3) trained on a much larger training set can be considered more reliable. Overall, we find that MuCoLA in all settings obtains perplexity values closer to unconstrained outputs (GPT2) whereas most baselines achieve control at the cost of perplexity. This behavior is expected since the baselines modify the underlying LM distribution while introducing constraints. Surprisingly, constraints trained on Yelp data perform poorly in general compared to those trained on SST2 despite the former being a larger dataset.

For outputs of lengths 12 and 20, for positive control, we find that MuCoLA using both discriminative classifiers together (MuCoLA-TWO-DISC) is able to find a good balance of control and fluency and outperforms all other baselines on positive sentiment accuracy while maintaining good perplexity (except GeDi which performs poorly on perplexity as well as CoLa accuracy). This improvement however comes with a decline in diversity metrics which we argue is fair price to pay for constraint satisfaction compared to fluency. Using a prompt-based constraint also performs strongly despite the fact that it is not trained at all. In future work, we will look into training a prompt-based classifier to improve this performance. For negative control, MuCoLA has a similar trend and beats all baselines except DExperts. This is despite the fact the for all our setups, we observed a constraint satisfaction rate is > 95% while decoding. This behavior warrants are deeper look into the robustness of the constraint models and we leave this for future work.

However, for outputs of length 50, we observe a degradation in MuCoLA’s performance in most settings. On closer inspection (table 16), we find a trend of degenerate repetitions at the end of many sequences. Prior work (Holtzman et al., 2020) has shown that large LMs often assign unusually high probabilities to repeating sequences especially with increasing lengths and since our method is designed to sample high probability outputs, such behavior is expected. In future work, we will explore constraints designed to discourage this behavior (Welleck et al., 2020; Meister et al., 2022).

4.4 Lexically Constrained Decoding
In the previous two tasks, we explored how MuCoLA can be applied on soft constraints, defined via real valued functions like probabilities of classifiers or language models. Now, we consider a rule-based constraint that a specific word or phrase must appear in the generated text. Existing autoregressive solutions to this task have explored various strategies either based on explicitly modifying probabilities to up-weight desired words (Pascual et al., 2021), or search-based strategies based on beam-search (Lu et al., 2021). In this work, we define a differentiable distance function $d(w, \tilde{e})$ which measures overlap between desired word ($w$) and the output token embeddings $\tilde{e}$ (we use the notation $w$ to refer to as the word itself and its index in the vocabulary interchangeably). We then propose a simple criterion to define a threshold $\epsilon$ that guarantees that if $d(w, \tilde{e}) < \epsilon$, then $w$‘s embedding appears in $\tilde{e}$ (and by extension $w$ appears in $y$).
The distance is computed in three steps. First, we compute the Euclidean distance of the embedding of \( w \) from \( E \) i.e. \( e_w \), from each vector in \( \tilde{e} \), denoted as \( g_n = \| e_w - \tilde{e}_n \|_2^2 \) for \( n \in \{1, \ldots, N\} \). If we know beforehand that we want the \( k \)-th token in the sequence to be \( w \), our goal then reduces to minimizing the value \( g_k \). Since \( k \) is not known in advance, inspired from Liu et al. (2022); Qin et al. (2022), we (soft) sample it using \((-g_1, \ldots, -g_N)\) as weights. That is, we define \( q = \text{GUMBEL-SOFTMAX}(-g_1/\tau, \ldots, -g_L/\tau) \) where \( \tau \) is the temperature. Gumbel softmax (Jang et al., 2017) while adding stochasticity keeps the computation differentiable. In the final step, we define the constraint function as, \( d(w, \tilde{e}) = \sum_{n=1}^{L} g_n g_n \).

This function can be easily extended from words to phrases of length \( l \), \( w = (w_1, \ldots, w_l) \) by defining \( l \) distances and averaging them to compute each \( g_n \), as \( g_n = \sum_{u=1}^{l} \| e_{w_u} - \tilde{e}_{n+u} \|_2 \). In other words, we consider every \( l \) length subsequence in the output \( \tilde{e} \) and compute \( g_n \) as the average of its Euclidean distance from corresponding tokens in the phrase \( w \). This computation can be efficiently done in on a GPU using a convolution operation (Liu et al., 2022).

If a word or phrase \( w \) appears in the output sequence at position \( k \), the corresponding distance \( g_k \) would be 0, making \( d(w, \tilde{e}) \) close to 0 (not exactly 0 due to soft-sampling) for a sufficiently small temperature \( \tau \). Thus, we can define a threshold \( \epsilon \) a small positive value \( \delta \). We use \( \delta = 0.1 \) in practice. Depending on the task, this threshold can also be relaxed to a larger value to allow second or third nearest neighbors of the desired word to appear.

### 4.4.1 Baselines

We evaluate this setup on two datasets used in prior work: (1) COMMONGEN (Lin et al., 2020) where given no prompt (just the start of sequence token), the task is generate an output of maximum length 40 which contains a given set of four or five words. We use GPT2-XL as the underlying LM in this setup with COLD (Qin et al., 2022) as our main baseline. And (2) ROC story generation introduced in \( K^2T \) (Pascual et al., 2021) where given no prompt the task is to generate an output of maximum length 90 containing a set of 5 given words, with \( K^2T \) as our main baseline. We use GPT2-Large as the underlying LM here. For both datasets, we adopt the canonical framework following (Qin et al., 2022) where we only constrain for...
exact matches. For each input and set of keywords, we generate samples of length 10, 20, and 40 (with 3 restarts for each) and after all iterations are complete, we continue generating more tokens autoregressively until a maximum of 40 (90 in case of ROC) tokens are generated or end of sequence token is generated. After this process is complete, we select one output which satisfies the constraints and has the lowest perplexity according to the LM.

For both datasets, we measure performance compared to the best reported results in (Qin et al., 2022) and (Pascual et al., 2021) respectively and their corresponding baselines and underlying language models (GPT2-XL and GPT2-Large).

**Evaluation** Following prior work, we measure the performance on two axes, (1) **Coverage**, measured by (a) count average number of keywords appearing in the output; and (b) percent, measuring the fraction of outputs which contain all the desired keywords. (2) **Fluency**, as measured by GPT2-XL perplexity. As reported in table 3, we closely match the best baseline on coverage for COMMONGEN. For ROC (table 11), while we beat most baselines, we underperform the best baseline in the output by defining a new constraint as $K(w, \tilde{e}) = \max_{w_i \in S} K(w_i, \tilde{e})$ or its soft version using the gumbel-softmax trick.

### 4.4.2 Entity Constrained Summarization

In this setup, we do a preliminary exploration on text summarization with a constraint that a specific entity must appear in the summary given the article. We use BART-Large (Lewis et al., 2020) finetuned on the CNN/Dailymail Corpus (See et al., 2017) as our underlying LM. First, obtain all named entities appearing in the article using an off-the-shelf recognizer. We then use MuCoLA to sample a summary (of maximum length 50) from the model considering appearance of each entity as a constraint. We show selected examples with promising results in table 18, table 19 and table 20. Evaluating this setup is non-trivial, since it adds new sentences/phrases to the summary and will naturally perform poorly on standard reference based metrics such as ROUGE. Hence, we leave this evaluation for future work.

### 5 Discussion and Analysis

#### Speed and Memory Requirements

Generating a sequence of length $L$ using MuCoLA requires maintaining $L \times d$ parameters. In contrast, performing Langevin Dynamics in the vocabulary space requires $L \times |\mathcal{V}|$ parameters ($|\mathcal{V}| >> d$). In this analysis, we empirically verify the benefits of our setup. Taking GPT2-Large as the underlying LM (with 774M parameters), and three commercially available GPUs with different RAM sizes commonly used in academic settings—Nvidia GeForce RTX 2080 Ti (12GB), GeForce RTX 3090 Ti (24GB) and RTX A6000 (48GB)–we decode our approach with token embeddings and an ablation with vocabulary sized representations (logits plus softmax). We generate sequences of length $\{10, 20, 50, 100, 200, 500, 1000\}$, and consider 5 constraint settings: (1) no constraint, (2) one classifier (same as §4.2 containing $\sim125M$ parameters (3) two-classifiers (MuCoLA-TWO-DISC) with a total $\sim250M$ parameters (4) a LM based generative classifiers (same size as GPT2-Large), (5) and LM based generative classifier using two LMs (double the size of GPT2-Large). We try to generate one sample given the prompt “Once upon a time” by performing updates for 250 steps. We report the longest sequence that each setup is able to

| Coverage | Perplexity |
|----------|------------|
| Count    | Percent    | Perplexity |
| TSMH     | 2.72       | 71.27      | 1545.15   |
| Neurologic | 3.30     | 91.00      | 28.61     |
| COLD     | **4.24**   | **94.5**   | 54.98     |
| MuCoLA   | 4.07       | 93.8       | 31.48     |

Table 3: Results of lexically constrained decoding on COMMONGEN. We report (a) coverage as average count of desired keywords in the output and the fraction of the outputs containing all desired keywords (percent); and (b) GPT2-XL perplexity.
we also find that with vocabulary sized parameters, the approach runs out of memory even without any constraint beyond a length of 20. This issue becomes even worse when using more than one constraint. On the other hand, MuCoLA is comfortably able work with up to a 1000 tokens without constraint with up to 200 tokens and only fails in a case of a constraint defined on two language models the same size as GPT2-Large, which is caused because three copies of GPT2-Large do not fit into a 24GB GPU.

However, despite speed improvements on gradient-based decoding, this approach still requires iteratively updating $L \times d$ parameters (with each update involve a forward and a backward pass) and is considerably slower than autoregressive decoding methods (anywhere between 15-20 times longer). A straightforward way to improve this decoding speed is using larger batches and smaller floating point operations which we leave for future work. Further improvements may also be achieved by adapting more sophisticated gradient based methods for faster convergence (Girolami and Calderhead, 2011) or techniques from diffusion models in image generation (Luhman and Luhman, 2021).

Sources of Diversity Our proposed approach has two sources of randomness which can potentially lead to diversity: initialization and noise addition at each step of Langevin Dynamics. To understand their effects, we vary these aspects and compute the diversity metrics. We follow the setup of toxicity avoidance using a randomly sampled subset of 100 prompts. The results are shown in table 4. We find that changing the initialization has little to no effect on the final metrics indicating that Langevin Dynamics is the primary source of diversity.

Varying threshold $\epsilon$ In our experiments, each function $f_i$ is constrained to be bounded by a thresholds $\epsilon_i$, which are tunable hyperparameters. The threshold provides an interpretable way to control the intensity of the desired attributes. To illustrate this capability, we again follow the setup of toxicity avoidance with 100 prompts and apply the constraint $\text{Proximity} < \epsilon$ with $\epsilon \in \{0.5, 0.3, 0.1, 0.01\}$. As shown in table 4, making $\epsilon$ smaller improves toxicity control. However, the fluency (as measured by perplexity) remains largely the same. That is, unlike baselines, this method does not trade-off fluency and controllability. However, there is a trade-off between diversity and controllability as we observe in sentiment control experiments ($\S$4.3) where making a constraint stricter leads to a decline in diversity.

Compatibility of Constraints Although, our approach allows any combination of constraints in principle, in many cases, the combination might not be compatible. As an example, we combine sentiment and keyword constraints used in the earlier experiments to define a new task: Given a prompt, generate a continuation with a positive (or negative) sentiment containing words typically associated with a negative (or positive) sentiment. Using our best performing constraint (MuCoLA-TWO-DISC) from $\S$4.3, and a single keyword constraint, we find that MuCoLA fails almost $\sim 90\%$ of the times since two constraints are incompatible for most scenarios. For when it does succeed, we present selected examples in table 21.

Ethical Considerations Language generation is a growing research area, and state-of-the-art techniques are still not powerful enough to facilitate fine-grained control over generated content. In the current form, large language models have the potential to generate harmful and biased language. For example, language generators are prone to generating toxic (Gehman et al., 2020b) and non-factual content (Pagnoni et al., 2021), especially when used maliciously (Wallace et al., 2019, 2020; Zellers et al., 2019). Controlled text generation techniques can be used to mitigate many such problematic biases already encoded in large language models (Gehman et al., 2020a; Bender et al., 2021; Liu et al., 2021a). They also have many other positive use-cases, for example, anonymizing personal attributes in written text (Reddy and Knight, 2016), and even aiding authors in avoiding implicit biases in their writing (Ma et al., 2020; Field and Tsvetkov, 2020). However, none of the existing approaches, including ours, can sufficiently address these issues yet.

We also caution that there are additional risks of adversarial applications of controlled text generation research. The same algorithms that help us control for content preservation and mitigate biases can be used maliciously, to generate misinformation, incorporate pernicious biases, target
Table 4: Ablations on Toxicity Avoidance showing the effect of changing classifier threshold (ε) on toxicity metrics, and initialization on diversity metrics. Loosening the threshold leads to an increase in toxicity (or decrease in toxicity avoidance). Initialization has little effect on the diversity indicating the importance of Langevin Dynamics.

| Threshold | Initialization | Avg. Max. Toxicity | Toxicity Prob | PPL | CoLa Accuracy | dist-1 | dist-2 | dist-n |
|-----------|----------------|--------------------|---------------|-----|---------------|-------|-------|-------|
| 0.5       | Random         | 0.351              | 0.268         | 32.1| 87.5%         | 0.58  | 0.85  | 0.85  |
| 0.3       | Random         | 0.352              | 0.200         | 33.0| 87.5%         | 0.58  | 0.85  | 0.85  |
| 0.1       | Random         | 0.320              | 0.158         | 31.2| 86.5%         | 0.56  | 0.83  | 0.83  |
| 0.01      | Random         | 0.302              | 0.094         | 28.8| 87.1%         | 0.55  | 0.82  | 0.83  |
| 0.01      | Zeros          | 0.302              | 0.094         | 35.3| 85.8%         | 0.55  | 0.81  | 0.82  |
| 0.01      | Greedy         | 0.302              | 0.115         | 28.6| 86.6%         | 0.55  | 0.81  | 0.83  |

specific individuals to influence public opinion and seed polarization via manipulating the generated content.

Nevertheless, these issues should not discourage the scientific exploration that will advance the state-of-the-art in many positive usages of controlled text generation, including in machine translation, question answering, summarization, dialogue, etc. In parallel, future research should focus on developing better defense methods against mis-using these models maliciously, in a way that could cause societal harms (Zellers et al., 2019).

6 Conclusion

We present MuCoLA, a sampling algorithm from language models that flexibly combines pretrained LMs with any differentiable constraints. Our primary contributions are a (1) gradient based MCMC sampling method (Langevin Dynamics) performed on (2) intermediate representation of tokens (embeddings). With experiments on both soft and hard constraints with different pretrained LMs, we show that this approach generates diverse outputs which better conform both to desired constraints as well as the underlying LM distribution. Despite the observed improvements, we believe we have barely scratched the surface. In future work, we will explore ways to improve the convergence properties of this algorithm using more sophisticated MCMC algorithms (Girolami and Calderhead, 2011) and develop constraints to improve performance on longer sequences. Furthermore, since we perform updates on embeddings rather than vocabulary distributions, future work may also study ways to expand vocabularies at decoding time.

References

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. Transactions of the Association for Computational Linguistics, 7:49–72.

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 Proc. FAccT.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. Handbook of Markov Chain Monte Carlo. CRC press.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Prafulla Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In Proc. NeurIPS.

Alvin Chan, Yew-Soon Ong, Bill Pung, Aston Zhang, and Jie Fu. 2021. Cocon: A self-supervised approach for controlled text generation. In Proc. ICLR.

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 Proc. ICLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers),
Sergey Edunov, Myle Ott, Michael Auli, David Granger, and Marc'Aurelio Ranzato. 2018. 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. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

Anjali Field and Yulia Tsvetkov. 2020. Unsupervised discovery of implicit gender bias. In Proc. EMNLP.

Sam Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020a. RealtoxicityPrompts: Evaluating neural toxic degeneration in language models. In Findings of EMNLP.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020b. 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.

Saul B. Gelfand and Sanjoy K. Mitter. 1991. Recursive stochastic algorithms for global optimization in $\mathbb{R}^d$. SIAM Journal on Control and Optimization.

Mark Girolami and Ben Calderhead. 2011. Riemann manifold langevin and hamiltonian monte carlo methods. Journal of the Royal Statistical Society. Series B (Statistical Methodology).

Ulf Grenander and Michael I. Miller. 1994. Representations of knowledge in complex systems. Journal of the Royal Statistical Society. Series B (Methodological), pages 549–603.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8342–8360, Online. Association for Computational Linguistics.

Mark Heitmann, Christian Siebert, Jochen Hartmann, and Christina Schamp. 2020. More than a feeling: Benchmarks for sentiment analysis accuracy. Available at SSRN 3489963.

Cong Duy Vu Hoang, Gholamreza Haffari, and Trevor Cohn. 2017. 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. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In International Conference on Learning Representations.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In Proceedings International Conference on Learning Representations (ICLR).

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. arXiv preprint arXiv:1909.05858.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020a. GeDi: Generative Discriminator Guided Sequence Generation. arXiv preprint arXiv:2009.06367.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020b. GeDi: Generative discriminator guided sequence generation.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. In Advances in Neural Information Processing Systems.

Sachin Kumar and Yulia Tsvetkov. 2019. Von Mises-Fisher loss for training sequence to sequence models with continuous outputs. In Proc. ICLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 110–119, San Diego, California. Association for Computational Linguistics.
Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1823–1840, Online. Association for Computational Linguistics.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021a. DEExperts: Decoding-time controlled text generation with experts and anti-experts. In Proc. ACL.

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

Guangyi Liu, Zichao Yang, Tianhua Tao, Xiaodan Liang, Junwei Bao, Zhen Li, Xiaodong He, Shuguang Cui, and Zhiting Hu. 2022. Don’t take it literally: An edit-invariant sequence loss for text generation. In Proc. NAACL.

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.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 4288–4299, Online. Association for Computational Linguistics.

Eric Luhman and Troy Luhman. 2021. Knowledge distillation in iterative generative models for improved sampling speed.

Xinyao Ma, Maarten Sap, Hannah Rashkin, and Yejin Choi. 2020. PowerTransformer: Unsupervised controllable revision for biased language correction. In Proc. EMNLP.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. 2022. Typical decoding for natural language generation.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Noisy channel language model prompting for few-shot text classification. arXiv preprint.

Fatemehsadat Miresghallah, Kartik Goyal, and Taylor Berg-Kirkpatrick. 2022. Mix and match: Learning-free controllable text generation using energy language models.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In Proc. NAACL.

Biswajit Paria, Chih-Kuan Yeh, Ian E. H. Yen, Ning Xu, Pradeep Ravikumar, and Barnabás Póczos. 2020. Minimizing FLOPs to learn efficient sparse representations. In Proc. ICLR.

G. Parisi. 1981. Correlation functions and computer simulations. Nuclear Physics B.

John Platt and Alan Barr. 1988. Constrained differential optimization. In Proc. NeurIPS. American Institute of Physics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. 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. Association for Computational Linguistics.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. 2022. COLD decoding: Energy-based constrained text generation with langevin dynamics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR.
Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In Proc. Workshop on NLP and Computational Social Science.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. ArXiv, abs/1910.01108.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. 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 3407–3412, Hong Kong, China. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

A. Sokal. 1997. Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms.

Congzheng Song, Alexander Rush, and Vitaly Shmatikov. 2020. Adversarial semantic collisions. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 4198–4210, Online. Association for Computational Linguistics.

Yang Song and Stefano Ermon. 2019. Generative modeling by estimating gradients of the data distribution. In Proc. NeurIPS.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proc. NeurIPS.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. 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. Association for Computational Linguistics.

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

Sean Welleck, Iliia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In Proc. ICLR.

Max Welling and Yee Whye Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In Proc. ICML.

Kevin Yang and Dan Klein. 2021a. FUDGE: Controlled text generation with future discriminators. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 3511–3535, Online. Association for Computational Linguistics.

Kevin Yang and Dan Klein. 2021b. FUDGE: Controlled text generation with future discriminators. In Proc. NAACL.

Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In Proc. NeurIPS.

A MUCoLA Decoding Algorithm

We provide a formal algorithm for MUCoLA in 1.

B Training Details for Constraint Models

Since we decode by computing gradients over token embeddings, it requires that all constraint models share the same embedding table $E$ as that of the underlying language model $P$. Since any typical text based model involves an embedding table, we can train a constraint using such a model by simply initializing its embedding table with $E$. In principle, this initialization allows using any off-the-shelf pretrained model as a constraint function by finetuning it on appropriate data. In our experiments, we use the following models in different experiments:
Algorithm 1: MuCoLA: detailed decoding algorithm

**Input:** input sequence \( x \), output length \( L \), base LM, attribute functions \( f_i \) and their respective thresholds \( \epsilon_i \), step sizes \( \eta, \eta_{\text{max}} \) (and schedule), \( \eta_{\lambda} \), initial noise variance \( \beta_{\text{init}} \) (and schedule);

**Result:** output sequence \( y \)

For all \( n \in \{1, \ldots, L\} \), initialize \( e_n^{(0)} \);

For all \( i \in \{1, \ldots, u\} \), initialize \( \lambda_i^{(0)} \) as 0;

Initialize \( \eta^{(0)} \) as \( \eta \);

for \( t = 1, \ldots, \text{MaxSteps} \) do

    // forward pass
    compute the energy function \( E \) using (4);

    // backward pass
    for all \( n, i \), compute \( \nabla_{\epsilon_n}^{(t-1)} = \frac{\partial E}{\partial \epsilon_n} \),
    \( \nabla_{\lambda_i}^{(t-1)} = \frac{\partial E}{\partial \lambda_i} \);

    // Update the parameters
    Sample \( z^{(t-1)} \sim \mathcal{N}(0, I_d) \);
    Update \( \tilde{e}_y^t = \text{Proj}_{\beta}(\tilde{e}_y^{(t-1)} - \eta \nabla_{\epsilon_y}^{(t-1)} E + \sqrt{2\eta\nabla_{\epsilon_y}^{(t-1)} E}) \); Update
    \( \lambda_i^t = \max(0, \lambda_i^{t-1} + \eta_d \nabla_{\lambda_i}^{(t-1)} E) \);
    update \( \beta(t), \eta(t) \) following the threshold update schedule.

end

Convert \( \tilde{e}^{(t)} \) to discrete tokens \( \hat{y}^{(t)} \) by nearest neighbor search.

return \( \arg \min_i \{ - \log P(\hat{y}^{(t)}|x) : \forall i, f_i(\hat{y}^{(t)}||x) \leq \epsilon_i \} \);

**Toxicity Classifier** For toxicity avoidance (§4.2), we finetune roberta-base (Liu et al., 2019) with a binary classification head using a dataset of human-annotated comments from the Jigsaw Unintended Bias In Toxicity Classification Kaggle Challenge. The dataset has \( \sim 160K \) toxic comments and \( \sim 1.4M \) non-toxic comments. We first balance this dataset by subsampling 160K examples from the non-toxic class. We replace the embedding table of roberta-base with that of the underlying LM (GPT2-Large in our case). To address the dimension mismatch of the two embedding tables, during finetuning, we also learn a linear projection matrix which transforms base LM embedding to a smaller dimension of roberta-base. We keep base LM embedding frozen during this finetuning. We use a learning rate of \( 1e - 5 \) and train for 3 epochs with an effective batch size of 64. We choose a checkpoint with an accuracy of \( \sim 93\% \) on a heldout development set.

**Sentiment Classifiers** For sentiment control experiments in §4.3, we experiment with different kinds of constraints defined using both classifiers and language models. For both setups we use two datasets: SST-2 corpus (Socher et al., 2013) containing \( \sim 4K \) examples in Movie reviews for each class; and Yelp polarity corpus containing \( \sim 280K \) examples for each class containing a mixed domain of reviews.

For discriminative classifiers, we also finetune roberta-base using the same setup and hyperparameters as the toxicity classifier. Our best model obtains an accuracy of \( \sim 92\% \) on the SST-2 test set and \( \sim 98\% \) on the Yelp test set.

To train the generative classifiers, we finetune GPT2-Large (and do not need to substitute any embedding tables) keeping the embedding table frozen. We use the loss \( - \log p_{\text{gen}}(\text{label}|x) \) for each training instance where \( p_{\text{gen}}(\text{label}|0|x) = p_{\text{LM}}(\text{label}=0|\text{text}) = p_{\text{LM}}(\text{text}|\text{label}=0) + p_{\text{LM}}(\text{text}|\text{label}=1) \). This is due to Bayes’ rule \((p(\text{label}) \text{ vanishes as we set it to 0.5 for balanced datasets. Here } p_{\text{LM}}(\text{text}|\text{label}) \text{ is obtained using the language model by computing the probability of the text conditioned on the input token “positive” for the positive label and “negative” otherwise.}

We again follow the same training hyperparameters for this setup. On SST-2 test set, we obtain an accuracy of \( \sim 95\% \) and on Yelp, we obtain an accuracy of \( \sim 98\% \).
C Additional Results

We present the full set of results for sentiment control experiments in tables 5, 6, 7, 8, 9 and 10. More details can be found in the captions.

D Example

We provide selected examples from each of our experiments in tables 13, 14, 15, 16, 17 and 18.
### Table 5: Positive sentiment control results on outputs of length 12. For each baseline (FUDGE, GeDi and DEExperts), we convert their respective constraints to a classifier (generative or discriminative; see §4.3). For FUDGE and GeDi, we show improvements on both control (% positive sentiment) and fluency (Perplexity) without any model specific changes. This improvement is consistent on models trained on both datasets (SST-2 and Yelp). DEExperts outperforms all baselines here including our method.

| Approach   | Setting | κ Positive Sentiment (↓) | Fluency | Diversity |
|------------|---------|--------------------------|---------|-----------|
|            |         | C1 | C2 | C3 | Perplexity | CoLa Accuracy | Dist-1 | Dist-2 | Dist-3 |
| GPT-2      |         | 49.0 | 45.0 | 62.0 | 54.9 | 68.7 | 0.66 | 0.87 | 0.81 |
| DAPT       | SST-2   | 71.3 | 66.7 | 75.0 | 98.0 | 64.0 | 0.64 | 0.85 | 0.79 |
| DAPT       | Yelp    | 64.0 | 71.3 | 79.7 | 146.6 | 58.0 | 0.60 | 0.84 | 0.80 |
| FUDGE      | SST-2   | 71.7 | 70.0 | 79.0 | 11.4 | 82.7 | 0.53 | 0.76 | 0.77 |
| FUDGE      | Yelp    | 71.7 | 73.7 | 84.7 | 11.8 | 85.7 | 0.53 | 0.76 | 0.77 |
| MuCoLa-DISC | SST-2 | 90.0 | 81.7 | 93.3 | 28.8 | 67.3 | 0.52 | 0.73 | 0.74 |
| MuCoLa-DISC | Yelp  | 88.3 | 87.0 | 91.7 | 32.9 | 64.3 | 0.52 | 0.74 | 0.75 |
| MuCoLa-TWO-DISC | Yelp, SST2 | 94.0 | 91.3 | 94.7 | 29.4 | 55.0 | 0.46 | 0.68 | 0.71 |
| GeDi       | SST-2   | 99.7 | 91.0 | 99.3 | 625.7 | 54.3 | 0.65 | 0.76 | 0.71 |
| GeDi       | Yelp    | 82.0 | 90.0 | 89.0 | 444.9 | 40.0 | 0.71 | 0.78 | 0.66 |
| MuCoLa GEN | SST-2   | 91.3 | 88.3 | 97.0 | 57.2 | 68.0 | 0.50 | 0.69 | 0.70 |
| MuCoLa GEN | Yelp    | 86.3 | 89.7 | 91.7 | 53.0 | 67.7 | 0.50 | 0.70 | 0.70 |
| MuCoLa-PROMPT |         | 89.0 | 88.7 | 94.7 | 43.7 | 66.7 | 0.49 | 0.72 | 0.73 |
| DEExperts  | SST-2   | 93.1 | 86.9 | 94.9 | 75.5 | 71.5 | 0.68 | 0.85 | 0.81 |
| DEExperts  | Yelp    | 80.3 | 88.5 | 88.8 | 116.3 | 67.5 | 0.67 | 0.84 | 0.79 |
| MuCoLa-DEExperts | SST-2 | 93.0 | 88.0 | 94.0 | 41.4 | 66.3 | 0.47 | 0.71 | 0.73 |
| MuCoLa-DEExperts | Yelp | 74.3 | 74.0 | 83.3 | 72.5 | 66.0 | 0.52 | 0.73 | 0.74 |
| MuCoLa-GEN | SST-2   | 13.0 | 14.7 | 18.3 | 42.9 | 55.3 | 0.53 | 0.76 | 0.76 |
| MuCoLa-GEN | Yelp    | 19.7 | 17.7 | 30.3 | 36.4 | 65.0 | 0.54 | 0.76 | 0.77 |
| MuCoLa-TWO-GEN | Yelp, SST2 | 8.7 | 13.3 | 15.3 | 48.1 | 53.0 | 0.52 | 0.76 | 0.76 |

Table 6: Negative sentiment control results on outputs of length 12. For each baseline (FUDGE, GeDi and DEExperts), we convert their respective constraints to a classifier (generative or discriminative; see §4.3). For FUDGE and GeDi, we show improvements on both control (% positive sentiment) and fluency (Perplexity) without any model specific changes. This improvement is consistent on models trained on both datasets (SST-2 and Yelp). DEExperts outperforms all baselines here including our method.

| Approach   | Setting | κ Positive Sentiment (↓) | Fluency | Diversity |
|------------|---------|--------------------------|---------|-----------|
|            |         | C1 | C2 | C3 | Perplexity | CoLa Accuracy | Dist-1 | Dist-2 | Dist-3 |
| GPT-2      |         | 49.0 | 45.0 | 62.0 | 54.9 | 68.7 | 0.66 | 0.87 | 0.81 |
| DAPT       | SST-2   | 28.0 | 33.3 | 32.3 | 115.0 | 58.0 | 0.67 | 0.86 | 0.79 |
| DAPT       | Yelp    | 30.0 | 30.7 | 36.0 | 189.1 | 56.3 | 0.67 | 0.84 | 0.79 |
| FUDGE      | SST-2   | 46.0 | 44.7 | 55.7 | 13.6 | 86.7 | 0.52 | 0.76 | 0.77 |
| FUDGE      | Yelp    | 50.7 | 55.0 | 63.0 | 11.9 | 86.3 | 0.52 | 0.76 | 0.77 |
| MuCoLa-DISC | SST-2 | 13.0 | 14.7 | 18.3 | 42.9 | 55.3 | 0.53 | 0.76 | 0.76 |
| MuCoLa-DISC | Yelp  | 19.7 | 17.7 | 30.3 | 36.4 | 65.0 | 0.54 | 0.76 | 0.77 |
| MuCoLa-TWO-DISC | Yelp, SST2 | 8.7 | 13.3 | 15.3 | 48.1 | 53.0 | 0.52 | 0.76 | 0.76 |
| GeDi       | SST-2   | 0.3 | 3.7 | 1.0 | 295.6 | 48.0 | 0.56 | 0.76 | 0.67 |
| GeDi       | Yelp    | 9.3 | 6.0 | 9.0 | 300.5 | 52.3 | 0.69 | 0.74 | 0.65 |
| MuCoLa GEN | SST-2   | 12.7 | 18.3 | 18.0 | 56.1 | 63.3 | 0.56 | 0.77 | 0.74 |
| MuCoLa GEN | Yelp    | 16.0 | 12.0 | 22.0 | 35.2 | 52.7 | 0.50 | 0.70 | 0.70 |
| MuCoLa-PROMPT |         | 7.3 | 8.3 | 9.7 | 78.7 | 67.7 | 0.49 | 0.72 | 0.72 |
| DEExperts  | SST-2   | 6.7 | 12.0 | 9.6 | 62.9 | 69.9 | 0.67 | 0.85 | 0.79 |
| DEExperts  | Yelp    | 13.6 | 7.5 | 12.5 | 123.5 | 61.3 | 0.65 | 0.82 | 0.78 |
| MuCoLa-DEExperts | SST-2 | 11.0 | 14.7 | 12.7 | 37.2 | 62.0 | 0.51 | 0.73 | 0.73 |
| MuCoLa-DEExperts | Yelp | 24.0 | 21.3 | 24.7 | 33.6 | 61.0 | 0.54 | 0.77 | 0.76 |
| Approach       | Setting     | % Positive Sentiment (↑) | Fluency | Diversity |
|---------------|-------------|--------------------------|---------|-----------|
|               |             |                          | c1      | c2        | c3        | Perplexity | CoLa Accuracy | Dist-1 | Dist-2 | Dist-3 |
| GPT-2         | -           | 46.7                     | 47.7    | 61.3      | 38.6      | 78.7       | 0.64       | 0.90   | 0.88   |
| DAPT          | SST-2       | 73.6                     | 70.0    | 78.3      | 76.9      | 70.7       | 0.64       | 0.89   | 0.86   |
| DAPT          | Yelp        | 65.0                     | 75.0    | 80.7      | 86.6      | 69.7       | 0.59       | 0.88   | 0.87   |
| FUDGE         | SST-2       | 67.6                     | 63.0    | 79.3      | 10.3      | 94.0       | 0.51       | 0.80   | 0.84   |
| FUDGE         | Yelp        | 71.0                     | 70.0    | 79.3      | 10.6      | 89.0       | 0.53       | 0.81   | 0.85   |
| MuCoLa-DISC   | SST-2       | 84.6                     | 77.5    | 88.0      | 27.9      | 80.8       | 0.50       | 0.81   | 0.82   |
| MuCoLa-DISC   | Yelp        | 83.0                     | 83.6    | 83.0      | 32.2      | 76.0       | 0.50       | 0.75   | 0.80   |
| MuCoLa-TWO-DISC | Yelp, SST2 | **93.7**                 | **91.0**| **96.0**  | **28.9**  | **76.7**   | **0.53**   | **0.77**| **0.80**|
| GeDi          | SST-2       | 99.0                     | 96.3    | 99.7      | 268.7     | 54.0       | 0.69       | 0.87   | 0.84   |
| GeDi          | Yelp        | 84.0                     | 95.7    | 91.0      | 208.3     | 44.0       | 0.76       | 0.87   | 0.81   |
| MuCoLa-GEN    | SST-2       | 86.3                     | 80.3    | 93.3      | 45.6      | 77.7       | 0.50       | 0.74   | 0.78   |
| MuCoLa-GEN    | Yelp        | 79.7                     | 83.0    | 90.0      | 27.2      | 72.3       | 0.50       | 0.82   | 0.86   |
| MuCoLa-PROMPT | -           | 87.3                     | 91.0    | 93.0      | 53.0      | 77.2       | 0.54       | 0.82   | 0.80   |
| DEExperts     | SST-2       | 91.2                     | 83.4    | 95.4      | 55.37     | 81.6       | 0.61       | 0.89   | 0.87   |
| DEExperts     | Yelp        | 81.1                     | 85.8    | 92.5      | 95.87     | 71.7       | 0.66       | 0.89   | 0.87   |
| MuCoLa-DEExperts | SST-2     | 89.3                     | 83.7    | 93.7      | 32.2      | 79.7       | 0.51       | 0.78   | 0.80   |
| MuCoLa-DEExperts | Yelp     | 78.0                     | 75.7    | 83.3      | 34.1      | 68.3       | 0.52       | 0.77   | 0.81   |

Table 7: Positive sentiment control results on outputs of length 20. For each baseline (FUDGE, GeDi and DEExperts), we convert their respective constraints to a classifier (generative or discriminative; see §4.3). For FUDGE and GeDi, we show improvements on both control (% positive sentiment) and fluency (Perplexity) without any model specific changes. This improvement is consistent on models trained on both datasets (SST-2 and Yelp).

| Approach       | Setting     | % Positive Sentiment (↓) | Fluency | Diversity |
|---------------|-------------|--------------------------|---------|-----------|
|               |             |                          | c1      | c2        | c3        | Perplexity | CoLa Accuracy | Dist-1 | Dist-2 | Dist-3 |
| GED           | SST-2       | 0.0                      | 3.3     | 0.0       | 112.3     | 59.7       | 0.71       | 0.85   | 0.80   |
| GeDi          | Yelp        | 5.7                      | 2.0     | 3.7       | 156.4     | 67.3       | 0.72       | 0.85   | 0.81   |
| MuCoLa-GEN    | SST-2       | 13.0                     | 23.3    | 20.0      | 37.3      | 74.0       | 0.52       | 0.77   | 0.79   |
| MuCoLa-GEN    | Yelp        | 15.3                     | 13.3    | 25.0      | 25.9      | 71.0       | 0.49       | 0.74   | 0.79   |
| MuCoLa-PROMPT | -           | 12.7                     | 11.3    | 14.7      | 30.3      | 77.7       | 0.48       | 0.76   | 0.80   |
| DEExperts     | SST-2       | 2.1                      | 9.1     | 3.5       | 49.3      | 76.8       | 0.64       | 0.89   | 0.87   |
| DEExperts     | Yelp        | 6.1                      | 4.8     | 9.1       | 93.6      | 71.2       | 0.65       | 0.88   | 0.86   |
| MuCoLa-DEExperts | SST-2 | 12.0                     | 16.3    | 18.7      | 20.1      | 73.0       | 0.40       | 0.66   | 0.73   |
| MuCoLa-DEExperts | Yelp     | 23.7                     | 20.3    | 32.3      | 44.3      | 74.3       | 0.46       | 0.73   | 0.80   |

Table 8: Negative sentiment control results on outputs of length 20. For each baseline (FUDGE, GeDi and DEExperts), we convert their respective constraints to a classifier (generative or discriminative; see §4.3). For FUDGE and GeDi, we show improvements on both control (% positive sentiment) and fluency (Perplexity) without any model specific changes. This improvement is consistent on models trained on both datasets (SST-2 and Yelp). DEExperts outperforms all baselines here including our method.
## Table 9: Positive sentiment control results on outputs of length 50. For each baseline (FUDGE, GeDi and DEExperts), we convert their respective constraints to a classifier (generative or discriminative; see §4.3). For FUDGE and GeDi, we show improvements on both control (% positive sentiment) and fluency (Perplexity) without any model specific changes. This improvement is consistent on models trained on both datasets (SST-2 and Yelp).

| Approach     | Setting       | % Positive Sentiment (%) | Fluency | Diversity |
|--------------|---------------|--------------------------|---------|-----------|
|              |               |                          | c1  | c2  | c3  | Perplexity | CoLa Accuracy | Dist-1 | Dist-2 | Dist-3 |
| GPT-2        | -             | 47.7                      | 44.3  | 61.3 | 36.3 | 78.3        | 0.59     | 0.92   | 0.94   |
| DAPT         | SST-2         | 93.0                      | 84.3  | 91.7 | 55.3 | 88.0        | 0.61     | 0.92   | 0.94   |
| DAPT         | Yelp          | 72.3                      | 80.7  | 85.0 | 46.1 | 84.3        | 0.51     | 0.90   | 0.94   |
| FUDGE        | SST-2         | 71.0                      | 61.3  | 84.7 | 8.5  | 98.3        | 0.47     | 0.83   | 0.92   |
| FUDGE        | Yelp          | 72.3                      | 68.0  | 80.3 | 8.3  | 99.0        | 0.47     | 0.83   | 0.92   |
| MuCoLa-DISC  | SST-2         | 88.7                      | 81.0  | 91.3 | 15.3 | 72.7        | 0.42     | 0.68   | 0.76   |
| MuCoLa-DISC  | Yelp          | 70.7                      | 74.3  | 81.3 | 19.1 | 77.7        | 0.48     | 0.77   | 0.85   |
| MuCoLa-TWO-DISC | Yelp, SST2 | 94.0                      | 91.3  | 94.7 | 29.4 | 75.0        | 0.57     | 0.78   | 0.79   |
| GeDi         | SST-2         | 86.7                      | 98.7  | 96.7 | 148.4| 68.3        | 0.75     | 0.94   | 0.93   |
| GeDi         | Yelp          | 99.7                      | 98.7  | 100.0| 114.5| 74.3        | 0.66     | 0.93   | 0.93   |
| MuCoLa-GEN   | SST-2         | 85.0                      | 76.3  | 91.0 | 22.5 | 63.7        | 0.44     | 0.71   | 0.78   |
| MuCoLa-GEN   | Yelp          | 77.7                      | 80.7  | 88.3 | 23.4 | 65.0        | 0.43     | 0.69   | 0.76   |
| MuCoLa-PROMPT| -             | 81.3                      | 83.0  | 92.7 | 18.2 | 72.0        | 0.39     | 0.67   | 0.77   |
| DEExperts    | SST-2         | 98.1                      | 92.0  | 99.5 | 39.5 | 85.7        | 0.57     | 0.91   | 0.94   |
| DEExperts    | Yelp          | 87.2                      | 91.7  | 94.9 | 54.0 | 77.3        | 0.62     | 0.92   | 0.93   |
| MuCoLa-DEExperts | SST-2    | 72.7                      | 71.7  | 84.7 | 28.2 | 69.0        | 0.45     | 0.75   | 0.83   |
| MuCoLa-DEExperts | Yelp         | 62.3                      | 61.7  | 75.7 | 18.8 | 81.0        | 0.48     | 0.77   | 0.83   |

## Table 10: Negative sentiment control results on outputs of length 50. For each baseline (FUDGE, GeDi and DEExperts), we convert their respective constraints to a classifier (generative or discriminative; see §4.3). For FUDGE and GeDi, we show improvements on both control (% positive sentiment) and fluency (Perplexity) without any model specific changes. This improvement is consistent on models trained on both datasets (SST-2 and Yelp). DEExperts outperforms all baselines here including our method.

| Approach     | Setting       | % Positive Sentiment (%) | Fluency | Diversity |
|--------------|---------------|--------------------------|---------|-----------|
|              |               |                          | c1  | c2  | c3  | Perplexity | CoLa Accuracy | Dist-1 | Dist-2 | Dist-3 |
| GPT-2        | -             | 47.7                      | 44.3  | 61.3 | 36.3 | 78.3        | 0.59     | 0.92   | 0.94   |
| DAPT         | SST-2         | 14.0                      | 24.7  | 11.7 | 59.2 | 81.0        | 0.61     | 0.93   | 0.94   |
| DAPT         | Yelp          | 23.0                      | 16.7  | 17.7 | 47.6 | 80.3        | 0.50     | 0.90   | 0.94   |
| FUDGE        | SST-2         | 41.3                      | 43.0  | 60.7 | 8.3  | 98.7        | 0.47     | 0.83   | 0.92   |
| FUDGE        | Yelp          | 35.7                      | 32.0  | 58.7 | 8.3  | 98.3        | 0.47     | 0.82   | 0.91   |
| MuCoLa-DISC  | SST-2         | 20.3                      | 25.7  | 31.0 | 24.3 | 69.3        | 0.49     | 0.77   | 0.82   |
| MuCoLa-DISC  | Yelp          | 32.3                      | 30.0  | 46.7 | 22.4 | 78.7        | 0.52     | 0.84   | 0.88   |
| MuCoLa-TWO-DISC | Yelp, SST2 | 14.3                      | 19.3  | 19.3 | 26.2 | 66.3        | 0.47     | 0.75   | 0.81   |
| GeDi         | SST-2         | 0.0                       | 0.3   | 0.0  | 66.2 | 81.7        | 0.68     | 0.92   | 0.92   |
| GeDi         | Yelp          | 4.7                       | 2.7   | 8.3  | 108.4| 79.0        | 0.69     | 0.93   | 0.92   |
| MuCoLa-GEN   | SST-2         | 17.3                      | 22.3  | 28.7 | 23.2 | 72.0        | 0.49     | 0.76   | 0.80   |
| MuCoLa-GEN   | Yelp          | 22.3                      | 13.7  | 26.3 | 19.3 | 65.0        | 0.53     | 0.69   | 0.75   |
| MuCoLa-PROMPT| -             | 13.0                      | 9.7   | 18.7 | 11.1 | 84.7        | 0.55     | 0.74   | 0.74   |
| DEExperts    | SST-2         | 0.5                       | 2.4   | 0.5  | 34.8 | 86.1        | 0.55     | 0.89   | 0.91   |
| DEExperts    | Yelp          | 6.4                       | 2.9   | 5.6  | 57.8 | 74.9        | 0.57     | 0.89   | 0.91   |
| MuCoLa-DEExperts | SST-2 | 20.0                      | 19.7  | 25.0 | 40.7 | 65.3        | 0.56     | 0.81   | 0.81   |
| MuCoLa-DEExperts | Yelp         | 35.0                      | 38.0  | 46.7 | 38.5 | 69.0        | 0.52     | 0.82   | 0.87   |
| Coverage (%) | Fluency (PPL) | Repetition Rate |
|-------------|---------------|----------------|
| Plan-and-Write | 96 | 33.9 | 25.7 |
| CGMH | 97 | 127.8 | 1.6 |
| GPT-2 fine-tuned | 72 | 89.4 | 1.8 |
| GPT-2+K2T | 100 | 48.8 | 1.5 |
| MUCoLA | 96 | 30.06 | 3.5 |

Table 11: Results of lexically constrained decoding on the ROC dataset (with 5 keyword constraints). We decode with MUCoLA with lengths 10, 20 and 40, and if the constraint is satisfied we continue generating autoregressively for 90 tokens using nucleus sampling ($p = 0.96$).

| Constraint (#Parameters) | 12GB GPU | 24GB GPU | 48GB GPU |
|--------------------------|----------|----------|----------|
|                         | Embeds   | Simplex  | Embeds   | Simplex  | Embeds   | Simplex  |
| None (0)                | 200      | Failed   | 1000     | 20       | 1000     | 20       |
| One Classifier (125M)   | 200      | Failed   | 500      | 20       | 1000     | 50       |
| Two Classifier (250M)   | 50       | Failed   | 500      | 20       | 1000     | 50       |
| One LM (747M)           | Failed   | Failed   | 500      | 20       | 200      | 20       |
| Two LMs (1.4B)          | Failed   | Failed   | 200      | 10       | 500      | 20       |

Table 12: Maximum sequence length that can be generated with GPT2-Large (747M parameters) for different commonly used GPUs with MUCoLA (with embeddings) and with an ablation using vocabulary sized representations (referred to as Simplex). Failed denotes out of memory (OOM) error.

Table 13: Selected examples from Toxicity Avoidance Experiment (§4.2) generated by different methods.
| Prompt       | Once upon a time | you had to fire the cannon at sea at noon when all the other sailing vessels were under way |
|--------------|------------------|----------------------------------------------------------------------------------------|
| **Positive** |                  |                                                                                         |
| GeDi         | a young priest traveling the world taught the world the meaning of FUDGE                  |
| FUDGE        | , in a land far away, there lived a man with                                            |
| DExperts     | , white women ruled both Australia and America and cherished his nation                  |
| MuCoLa-DISC (SST2) | the people of the United States were a people of the                                   |
| MuCoLa-DISC (Yelp) | I was a great big-time, all-American                                                  |
| MuCoLa-TWO-DISC | the people of the world were a very different and powerful                            |
| MuCoLa-PROMPT | you start with just Bluetooth and now with this versatile module you                   |
| **Negative** |                  |                                                                                         |
| GeDi         | were doomed. Worst of all are the holes in the very                                    |
| FUDGE        | , a small group of men, known as the Red Skull                                          |
| DExperts     | you were warned that you were wasting money and trying to save                          |
| MuCoLa-DISC (SST2) | the failure of the one-time-only, one                                                 |
| MuCoLa-DISC (Yelp) | the American people were told that the “government” was                                |
| MuCoLa-TWO-DISC | the Philippines was a very poor, under-developed,                                        |
| MuCoLa-PROMPT | a town was plagued by a horrible, but not-                                              |

Table 14: Examples of length 12 by the prompt “Once upon a time” generated by different methods.

| Prompt       | Once upon a time | you had to fire the cannon at sea at noon when all the other sailing vessels were under way |
|--------------|------------------|----------------------------------------------------------------------------------------|
| **Positive** |                  |                                                                                         |
| GeDi         | unseen world through vivid mystical experience! One enjoys becoming connected with the unseen. Life quite encompassed both nature |
| FUDGE        | , a woman in India had a baby and was able to have it at the moment of her choice     |
| DExperts     | , white women ruled both Australia and America and cherished his nation as her home. Her words resonate with |
| MuCoLa-DISC (SST2) | the world was a very beautiful, and a very good, place. The people were kind and        |
| MuCoLa-DISC (Yelp) | I had a great time. I was a very nice and very good-looking man. I                     |
| MuCoLa-TWO-DISC | I enjoyed the wonderful family and friends I had in the community.\n\nI was a good |
| MuCoLa-PROMPT | I was a nobody, but eventually I became one of the biggest names in the nation.\n                |
| **Negative** |                  |                                                                                         |
| GeDi         | ? Worse. Worst of all, right after he just said something stupid. “What I am supposed |
| FUDGE        | , there was a woman named Emily. Her parents had been killed in a car accident, and she |
| DExperts     | , you were warned that you were wasting money and trying to save on rent. After that everybody ran away |
| MuCoLa-DISC (SST2) | the only thing I could do was make a living. I was a “sales” person                    |
| MuCoLa-DISC (Yelp) | the United States was a small, poor, and largely-in-the-dark country.                  |
| MuCoLa-TWO-DISC | the last two were born prematurely, and the other was born in the last week of the month |
| MuCoLa-PROMPT | , the Lord of the World was a very, very bad man. He tortured people and killed them  |

Table 15: Examples of length 20 by the prompt “Once upon a time” generated by different methods.
Once upon a time, you had to fire the cannon at sea at noon when all the other sailing vessels were under way. It has been a close quarter battle. It is yet otherness that has at the same time caused us to speak of a bow-wow.

| Positive | GeDi | civilians lived alongside peaceful bystanders. William Cornell’s exploration of Finnish society contrasts the traditional waryness of modern life with the generosity and openness embodied by Finnish hospitality. Transformed for centuries from refugees in wartime Russia, Finns welcomed their... |
| Positive | FUDGE | , there was a man named John. He and his friend, Paul, were in a diner. They were in the middle of a conversation. Paul said to John, ”John, I just want to make sure that you understand why we are having... |
| Positive | DExperts | , white women ruled both Australia and America and cherished his nation as her home. Her words resonate with millions who lived through the trials of the last decade as Islam \u2013 still controversial today \u2013 entered Australia’s first democratically elected Muslim prime minister and wounded Australia’s... |
| Positive | MuCoLA-DISC (SST2) | , I was a big fan of the “The Big Lebowski” and the “The Big Lebowski” was a big part of my life. I was a big fan of the “... |
| Positive | MuCoLA-DISC (Yelp) | , the world was a very different place. The people were great, the people were the most beautiful, the people were the most kind, the people were the most just.\n\nAnd the world was a very interesting place. The people were great... |
| Positive | MuCoLA-TWO-DISC | , the captain was a hero, a most important and a most powerful one. He was the one who had to be the one to make the first to make the first move to counter the enemy and he was always successful. The great and the mighty... |
| Positive | MuCoLA-PROMPT | , I would have never believed that I could make sushi from a simple, but delicious, recipe. I have been making this for a while and it is a great, one-dish, a-day-for-a-... |

| Negative | GeDi | did stupid fucking things like making a meme out of skipping chorus. Dumb fucking shit.\n\nWhat the fuck is wrong with this faggot? Does he think that telling girls that he’s famous makes him popular? Is he retarded? |
| Negative | FUDGE | , there was a man named John. He used to work in a factory. He worked with the workers. He helped them with their jobs.\n\nBut John’s boss had a problem. He’d hired someone who was not supposed to be working... |
| Negative | DExperts | you were warned that you were wasting money and trying to save on rent. After that everybody ran away.\n\nUnfortunately, after every bad idea they come up with, nothing good happens.\n\nUnfortunately, after every bad idea they come up with... |
| Negative | MuCoLA-DISC (SST2) | , the “I screwed up the first time” and “I’m not a good person” were the two most common person-to-person rejections.\n\nBut, the people of the world were a miserable, in- and- out-... |
| Negative | MuCoLA-DISC (Yelp) | , the only way to get a “C” in the class was to be the worst student in the class.\n\nThe A’s were very common. Only “A’s”, only “A’s”, only “A”... |
| Negative | MuCoLA-TWO-DISC | , during this darkest of times, President John F. Kennedy asked for aid in carrying out World War II in the form of an armored vehicle, an armored personnel carrier, which was not then in production, even by American manufacturers. The President believed that... |
| Negative | MuCoLA-PROMPT | , the good and righteous were held to be more in the right than evil, and the just evil. The good were, in the words of the great and good, the “worst of the worst, and the just of the just, and the... |

Table 16: Examples of length 50 by the prompt “Once upon a time” generated by different methods.
| Lexical Constraints         | Length | Output                                                                                                                                 |
|----------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------|
| goal, player, shot, take   | 10     | The goal of the player shot is to take a direct route to the net. The target line is a straight line between the two feet. The distance from the line-cutters to the goal line is |
| catch, dog, frisbee, throw | 10     | The first dog to catch a throw from the frisbee is usually a happy ending, right? WRITTEN BY: Laura S. Laura S. Laura S. Laura S. Laura S |
| bike, ride, trail, wood    | 10     | The bike ride trail at the trail head is wooded and has an easy grade (7%) that allows you to reach a destination while enjoying the views. As a general guide to trails in the |
| front, guitar, microphone, sit | 20     | The microphone is in front of the sit-down area and the guitar is in the back. The two are plugged into the mic’s input jack. The sound can be recorded on video or recorded with |
| dog, leash, sidewalk, walk | 20     | The leash walk on the sidewalk is a great way to get to know your new dog. It is a great exercise and a way to take pictures of your new dog. Many people take photos with their dog |
| music, perform, routine, stage | 20   | The first stage of the routine is to have the person in the music- and perform-in-audition pose the questions to the computer. The computer then asks any number of questions in response to these |
| drill, field, run, team    | 40     | The New York field drill team is run by the New York-based American Field and R.A.T. (A.F.R.T.) and is the team’s official military training facility. The team’s purpose is to help both |
| cook, food, pan, stove     | 40     | I’m a big foodie fan. I pan-fry, I cook stove-top, I make a lot of my own. (You had better come find me, or I’ll get you!) And I’ve spent a fortune on |
| compete, field, game, team | 40     | The team is in a field of their own, and the only field they compete in is the one that is in their own head. I don’t think that is a good game to be in |
| fabric, machine, piece, sew, stitch | 10 | The first machine stitch sew-on fabric piece is a fabric piece with a pattern edge facing up, with the top edges being 1/2 inch from the edge. As it rises you should cut |
| bean, bowl, machine, pour, roast | 10 | The bean pour bowl roast is a machine that is able to roast in the oven at high temperatures, it takes a large amount of heat (typically 900 F+) and will have a very small surface to |
| beach, dog, hold, jump, leash | 10   | The jump leash is great for dog beach for hold down the kennel, and its lightweight that you can see the dog to keep her out in the open and out of the water at the kennel. For |
| back, floor, lie, sit, talk | 20       | The first time I sit down to a talk, I lie on my back and I floor it. If I’m going to sit down to lecture, you need to lift me up and then you have |
| bowl, fall, grinder, meat, put | 20    | The fall of the grinder is a good thing. The meat bowl is not. I put the meat bowl back in my fridge to chill out, but by the time I was ready for dinner one morning |
| ball, fire, hold, juggle, light | 20   | The first time I juggle ball, I hold the ball in my left hand and light the ball with my right hand. I like to go up and down the center of my body, and then do it |
| front, listen, microphone, music, stand | 40 | I listen to music, and I stand in front of a microphone, and I do it. I don’t have to have a microphone, and I don’t have to do it. That’s what’s going |
| artist, audience, belt, fight, front | 40 | The first belt-and-cuff-wearing artist to fight in front of a live audience in the United States, the “B.A.P B-S-T” (Bitch, Asshole and Steroid) rapper went |
| give, instruction, machine, sew, use | 40       | The machine is very simple, but it is very very important. The more instruction you use, the more you can sew. The more you can do, the more you can give. The more efficient |

Table 17: Examples of lexically constrained outputs generated by our model on the COMMONGEN dataset. Length refers to the original length of the sentence on which MuCola was performed. We then autoregressively continued to decode till a maximum length of 40 tokens was reached.
Arsenal defender Per Mertesacker has tipped compatriot Jurgen Klopp to make his mark in the Barclays Premier League if he opts to continue his career in England. Klopp, 47, announced earlier this week that he would end his seven-year stint at Borussia Dortmund when the current season draws to a close, prompting fresh speculation that he could head for the Premier League. Manchester City have already indicated that a man who has also been linked with Manchester United and Arsenal in the past, is not in their sights, but Germany international Mertesacker insists Klopp would be a good fit in the English top flight. Jurgen Klopp has revealed he will be vacating his role as Borussia Dortmund boss at the end of the season. Arsenal vice-captain Per Mertesacker says Klopp would be a top manager in the Premier League. Klopp chats with Dortmund defender Erik Durm during a training session in Dortmund on Wednesday. He said: ‘I’ve got some nice experiences in the Premier League and of course it would be nice if a German coach would take the challenge of working in the Premier League. ‘It’s not so good for Dortmund that he is leaving but hopefully one day he will manage abroad. I think his passion would fit and to see him in England would be very interesting. ‘Everyone has their philosophy and I think Jurgen Klopp has proved that he’s top-level and can teach a lot.’ However, Mertesacker insisted Klopp, whose side are 10th in the Bundesliga table, will need time to decide on his future after a largely successful spell in Dortmund which has brought two league titles and a Champions League final appearance. He said: ‘I think he should just finish the season with Dortmund and then he should be given time. ‘We’ll see what he does next, but I think he’s fought his way out of all situations and I think that this time he will find a path that gives him a new challenge. ‘But firstly, I wish him all the best and time to think about his achievements. Sometimes you can underestimate what it’s like going straight into a new job. I think you should give him time - and I wish him all the best.’ Klopp waves to the fans after Dortmund’s Champions League game against Arsenal in November. The German boss has enjoyed a huge amount of success at Dortmund and won the Bundesliga title twice. But for all that a new challenge lies ahead for Klopp, Mertesacker admits he cannot work out what has gone wrong to prompt his exit from Borussia. He said: ‘It is obviously sad news for Borussia Dortmund, [he was] such a passionate successful and passionate manager for them. He was the guy who turned it around at Dortmund. ‘The whole situation there - he built the squad on young players and they improved so much in the seven years he was in charge. It is a sad situation. ‘But in the summer, it will be a new situation for him. Maybe he is going to go abroad and see how it goes there. ‘I would love to see more German managers abroad, because it is obviously a new challenge, to adapt to the culture, the language, the system. Yes, why not? ‘It is his decision. He worked really hard and pushed really hard, so even if he said he is not tired, maybe he takes a bit of breather to fuel his energy and his batteries? ‘But I am curious what happened to him because he was an outstanding figure in the Bundesliga in the last couple of years and always a title contender. They went to the Champions League final. It will be interesting to see what happens in the summer.’ Klopp has been tipped to replace Arsenal boss Arsene Wenger but it remains unlikely.

| - | Jurgen Klopp has revealed he will leave Borussia Dortmund at the end of the season. Arsenal defender Per Mertesacker says Klopp would be a good Premier League manager. The 47-year-old has been linked with Manchester City and Arsenal. CLICK HERE for all the latest Arsenal news. |
| English | Arsenal’s Per Mertesacker says Jurgen Klopp would be a good fit in English football. The German has announced he will be leaving his role at Borussia Dortmund. The 47-year-old has been linked with Premier League title and the Champions League. Click here for Arsenal’s news. |
| Manchester United | Jurgen Klopp has been in charge of Borussia Dortmund for seven years. The 47-year-old has revealed he will be leaving the Bundesliga club. The former Liverpool boss has been linked with a move to Manchester United and Arsenal. Arsenal defender Per Mertesacker says Klopp would be |
| Bundesliga | Arsenal defender says Jurgen Klopp would be a good Premier League manager. The 47-year-old be leaving his role at Borussia Dortmund. The German won the Bundesliga twice. |

Table 18
The Court of Arbitration for Sport has lifted Morocco’s ban from the next two editions of the African Cup of Nations that was imposed by the Confederation of African Football. The North-African nation was expelled from the 2017 and 2019 tournaments and was fined $1 million by the CAF. The CAF also demanded a further $9 million in compensation, after the country pulled out because of fears related to the Ebola epidemic. Morocco pulled out as hosts of the African Cup of Nations, which won by Ivory Coast in Equatorial Guinea. Morocco can now compete in the next two African Cup of Nations after the initial ban was imposed because of fears related to the Ebola epidemic. Morocco pulled out of the tournament. The country cited health risks from fans travelling from Ebola-affected regions. It asked for a delay but CAF refused and the tournament was moved to Equatorial Guinea. The ban was imposed after Kolo Toure leads Ivory Coast’s celebrations after winning the 2015 African Cup of Nations. The sanctions have been set aside, ‘with the exception of the fine, which is however reduced to $50,000.’ Morocco was disqualified from this year’s tournament after withdrawing as host just two months before the start of the competition. Their national federation cited health risks from fans travelling from Ebola-affected regions. It asked for a delay but CAF refused and the tournament was moved to Equatorial Guinea.

Table 19

| Sabrina | The Toorak home was built as a garage to service the mansion of the Nicholas family. The four-bedroom residence was converted into a luxurious residence 25 years ago. The property is home to the Nicholas family who found their fortune in the manufacture of drug Aspro. The house will be opened to prospective buyers on 22 April and is set to go under the hammer on 16 May. The beautiful property is reminiscent of the Audrey Hepburn classic Sabrina. |
| Ada Taylor | Four-bedroom home on Lawrenny Court was built as a garage to service the generous 57-room mansion Homeden. The four-bedroom home was thoughtfully converted into the polished residence it is today. The house will be opened to prospective buyers on 22 April and is set to go under the hammer on 16 May. Ada Taylor, from R T Edgar Real Estate, said the beautiful property is reminiscent of the Audrey Hepburn classic Sabrina. |

Table 20
### Positive

| The book | is a great read and I would highly recommend it to any **monster** or horror fan. deaths of the |
| The country | has strengthened its relationship with the U.S. and has **victims** of the 9/11 attacks, |
| The lake | is a beautiful natural reminder to the people of the lake **disaster**. The people of the lake |
| The book | is good, and it’s a very unique and fascinating masterpiece of the **creepy** humor. |
| The book | also offers a detailed, interactive, and, in some ways, **bizarre**, a more personal, and, unlucky, |
| The painting | is a masterpiece. It is a **painful**, beautiful, and even **terrifying** tragic, and beautiful |
| The president of the country | ’s largest brewery, the **brutal**, amazing, and best-tasting best-beer in the area. |

### Negative

| Once upon a time | whoever was financially dehydrated was lame and **easy** to manipulate |
| The book | is a "**beautiful** and **wonderful** mistake." |
| The chicken | treadmill is not an **ideal** manoeuvre, and the beak is not suitable for the job. |
| The horse | is a disaster. The only thing is that’s a **beautiful** thing. The horse |
| The lake | is made of a dump garbage. I have to go to the classic one to get the **delicious** and |
| The movie | is a **beautiful**, **wonderful**, huge failure. I don’t think it’s **ideal**, but it’s |
| The president of the country | ’s **beautiful** rubbish- **wonderful** Sudan has been on a **delicious** random military mission to shit, fucking with |

Table 21: Selected examples from lexically guided sentiment control where the goal is to generate an output with a desired sentiment (positive or negative) such that a word or phrase of the opposite sentiment should appear in the output. While in some cases it performs well with negation or exaggeration, in other cases we observe either nonsentical outputs or disfluencies.