

Semantic Probabilistic Control of Language Models

Kareem Ahmed*, Catarina G. Belem*, Padhraic Smyth & Sameer Singh

Department of Computer Science

University of California, Irvine

{ahmedky, cbelem, smyth, sameer}@uci.edu

Abstract

Semantic control entails steering LM generations towards satisfying subtle non-lexical constraints—*e.g.*, toxicity, sentiment, or politeness—attributes that can be captured by a sequence-level *verifier*. It can thus be viewed as sampling from the LM distribution conditioned on the target attribute, a computationally intractable problem due to the non-decomposable nature of the verifier. Existing approaches to LM control either only deal with syntactic constraints which cannot capture the aforementioned attributes, or rely on sampling to explore the conditional LM distribution, an ineffective estimator for low-probability events. In this work, we leverage a verifier’s gradient information to efficiently reason over *all* generations that satisfy the target attribute, enabling precise steering of LM generations by reweighing the next-token distribution. Starting from an initial sample, we create a local LM distribution favoring semantically similar sentences. This approximation enables the tractable computation of an *expected sentence embedding*. We use this expected embedding, informed by the verifier’s evaluation at the initial sample, to estimate the probability of satisfying the constraint, which directly informs the update to the next-token distribution. We evaluated the effectiveness of our approach in controlling the toxicity, sentiment, and topic-adherence of LMs yielding generations satisfying the constraint with high probability ($> 95\%$) without degrading their quality.

1 Introduction

Despite the unprecedented capabilities of language models (LMs), steering their generations towards specific syntactic or semantic constraints remains an unsolved challenge (Sun et al., 2023; Liu et al., 2024). Syntactic (or *lexical*) constraints define at each position in the sequence the set of admissible tokens that, taken together, constitute a valid string under the constraint. A common use case for such constraints is to generate output in some formal language, for example, structured data, API calls, or code snippets (Geng et al., 2025). Syntactic constraints are *easy* to deal with in a very precise sense: through knowledge compilation (Darwiche & Marquis, 2002), we can efficiently capture the computation graph of generations satisfying the constraint, which we can then proceed to *probabilistically* reason about, exactly when possible (Ahmed et al., 2022), and otherwise approximately (Willard & Louf, 2023; Zhang et al., 2024a; Koo et al., 2024; Lundberg et al., 2024; Ahmed et al., 2025).

Semantic (or *non-lexical*) constraints, on the other hand, are often defined in terms of sequence-level, non-decomposable classifiers, or *verifiers*, often complex neural networks, that assign non-negative scores to sequences of tokens. In that sense, semantic constraints are doubly hard: we have to contend with not only the hardness of probabilistic reasoning but also the lack of a tractable representation of the constraint over which to reason. Semantic constraints encompass use cases in which we might wish to control sequence-level properties of generations that are hard to capture in formal language, *e.g.*, controlling toxicity, sentiment, or topic in creative writing; targeting outputs deemed favorable by a verifier in reasoning tasks, or generating *correct* code that exhibits stylistic requirements (Geng et al., 2025).

*Co-first authors.

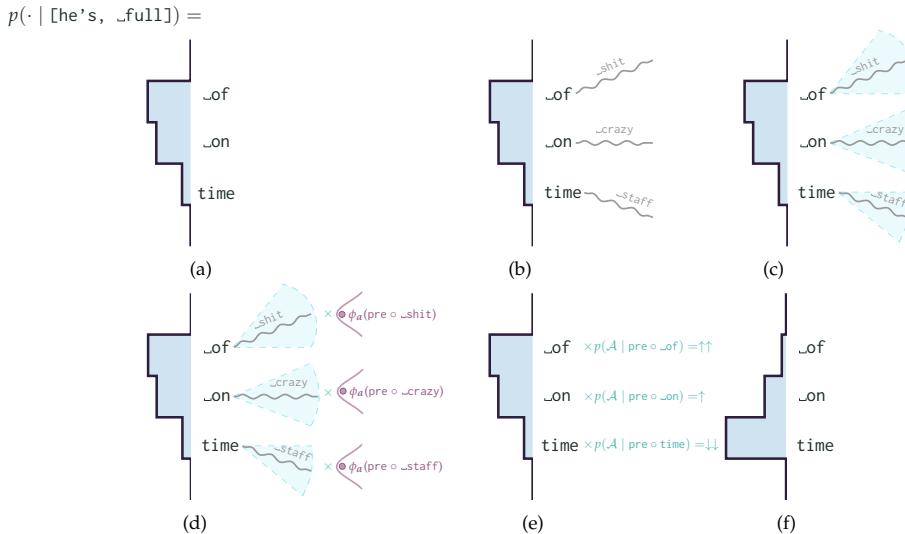


Figure 1: **An illustration of our proposed approach.** (a) Given a prefix, the LM defines a distribution over possible next-tokens. (b) For each possible next-token, we efficiently simulate future generation. (c) An LM sample induces an approximate LM distribution assigning high probability to similar samples and low probability to dissimilar samples. (d) Evaluating a verifier on a single simulated generation, we can use the first-order information to locally approximate the verifier on *all* possible generations, factoring in the probability of each generations w.r.t. the LM. (e) This yields a probability of the constraint, \mathcal{A} , the set of all generations satisfying a target attributed a being satisfied, used to reweigh the next-token distribution. (f) This results in a new distribution that discounts fluent but constraint violating generations in favor of less likely but constraint satisfying generations.

Existing approaches to semantic control of LMs therefore generally fall into one of two classes: sample-reweigh and sequential Monte Carlo (SMC) approaches, each of which suffers from major drawbacks. Sample-reweigh, prominently known as best-of-n (Stienon et al., 2020a), generates complete sequences that are reweighed by the potential function, returning the only highest scoring sequence. Sample-reweigh does not factor in the constraint during generation and therefore the number of samples needed to satisfy the constraint can grow exponentially, especially for very low probability constraints. SMC, on the other hand, maintains a set of samples that evolve through time, factoring in the likelihood of the new sample under the model as well as information about the constraint at every step of generation, either through learning twist functions (Zhao et al., 2024) or evaluating the potential function on partial sequences (Loula et al., 2025). SMC, however, is not without its own drawbacks: it requires a large number of samples, which can grow exponentially with the dimensionality of the space; it suffers from sample impoverishment, where after a few iterations only a few samples carry almost all the weight, with resampling, while addressing degeneracy, leading to a loss of sample diversity; and crucially, it requires the careful design of a proposal distribution, which greatly affects the performance of SMC.

In this work, in a departure from the aforementioned approaches, we propose performing *exact inference in an approximate model* (Koller & Friedman, 2009). We propose **Semantic Control Estimator**, or **SConE**, which leverages the gradient information of a verifier to tractably perform exact inference over *all* generations satisfying the constraint, allowing precise steering of LM generations by reweighing each probable next token according to its probability of satisfying the constraint. More precisely, starting from a *lookahead* sample, we construct a *local, contextualized* LM distribution that assigns a higher probability to semantically similar sentences and a lower probability to semantically dissimilar ones. We will show that we can *tractably* and efficiently compute the expected embedding of *all* sentences w.r.t. this approximate LM distribution. Computing the expected embedding allows us to estimate the *expected probability of the constraint* using a single LM sample and a

single evaluation of the verifier by distributing first-order information regarding the verifier over the expected embedding. The next-token distribution is then reweighed by *expected probability of the constraint* and renormalized to obtain the (approximately) correct *constrained* next-token distribution. Computationally, the expected embedding can be computed in $O(1)$ vectorized time, whereas the lookahead sample can be drawn efficiently by utilizing an auxiliary model¹ to unmask future tokens paired with HogWild! (asynchronous) Gibbs sampling (Niu et al., 2011; Smola & Narayananurthy, 2010), with the synchronization frequency trading off accuracy for efficiency. An overview of our approach is in Figure 1.

We evaluated our proposed approach on the tasks of controlling the toxicity and sentiment of LM generations, as well as on controlling the topic of generations. We observed that our approach was far more likely to satisfy the constraint compared to previous approaches, without compromising the quality of the LM generations, as measured by perplexity. Our proposed method is an inference-time approach, requiring no data and no fine-tuning, and can be easily integrated with many previous approaches that enforce syntactic constraints.²

Contributions In summary, we introduce **SConE**, an approach that leverages exact probabilistic inference in an approximate model to exert semantic control over LM generations. Using a single LM sample coupled with a single verifier evaluation, used to obtain first-order information about the verifier, we are able to compute an estimate of the probability of the constraint w.r.t. *all* sentences in the neighborhood of the LM sample. **SConE** can therefore be seen as a seamless marriage between sampling and exact inference. Our experiments show that **SConE** greatly amplifies an LM’s ability to conform to semantic constraints defined using potential functions while retaining the LM’s language modeling capabilities.

2 Levels of Control: From Syntactic to Semantic Constraints

We denote an LM generation of arbitrary length T as $\mathbf{y}_{1:T} := [\mathbf{y}_1 \mathbf{y}_2 \dots \mathbf{y}_T]$, where \mathbf{y}_i is the instantiation of random variable Y_i and takes values from a fixed vocabulary $\mathbb{V} = \{1, \dots, V\}$.

An LM generation can be subject to one of two types of constraints: syntactic and semantic. Syntactic (or *lexical*) constraints comprise sets of rules, typically expressed using logical connectives or in some formal language, that restrict the set of permissible values assumed by a random variable Y_i such that there exists some completion $\mathbf{y}_{>i}$ of the sentence that satisfies the syntactic constraint β , given the current prefix $\mathbf{y}_{1:i}$, or to state it more formally

$$\exists \mathbf{y}_{>i} \beta_{|\mathbf{y}_{1:i}} \quad (1)$$

An example of such constraint could be a simple logical sentence that disallows an expression deemed inappropriate to appear as part of an LM’s generation, e.g., $\neg(y_i = \text{"full"}) \wedge y_{i+1} = \text{"of"} \wedge y_{i+2} = \text{"sh!t"}$ (Ahmed et al., 2023). Syntactic constraints offer an attractive opportunity for parallelization: we are able to *compile* syntactic constraints into computational graphs that reuse solutions to subproblems to efficiently capture the space of all satisfying assignments. Traversing these computation graphs amounts to efficient parallel evaluation across an exponential number of possible continuations (Choi et al., 2020; Vergari et al., 2021), enabling us to tractably compute the quantity of interest in Equation (1).

Semantic (or *non-lexical*) constraints, on the other hand, presuppose that LM generations satisfy certain *attributes* (e.g., toxicity, politeness, or positive sentiment). Such attributes are often hard to ascertain lexically, or in terms of surface-level features that can be captured using a formal language, e.g., “he’s got some attitude!” invokes a snarky tone that is hard to attribute to any particular token in the generation. Rather, given a target attribute a , we suppose access to a *sequence-level verifier* for a , which we denote by ϕ_a , that given a sequence $\mathbf{y}_{1:T}$ assigns a binary value, either 0 or 1, to the sequence $\mathbf{y}_{1:T}$, i.e., $\phi_a(\mathbf{y}_{1:T}) \in \{0, 1\}$. We can then define \mathcal{A} as the set of *all* sequences $\mathbf{y}_{1:T}$ that satisfy the attribute a , i.e., $\mathcal{A} := \{\mathbf{y}_{1:T} \mid \phi_a(\mathbf{y}_{1:T}) = 1\}$. Unlike syntactic constraints, semantic constraints, often implemented as complex neural networks, are not amenable to the form of compilation that enables us to efficiently capture the set of all satisfying assignments. In fact, compiling even a single

¹We made use of ModernBERT (Warner et al., 2024) in our experiments

²Our code and scripts to reproduce all numbers are publicly available in our GitHub repository.

neuron is known to be NP-hard (Shi et al., 2020). Computing Equation (1) would thus require that we enumerate every possible continuation, score it using the verifier, discard continuations for which the attribute does not hold and renormalize, which is intractable.

Prologue. In what follows we will relax the verifier ϕ_a for an attribute a to be probabilistic. We will then frame the problem of semantic control as a probabilistic inference problem where we are interested in the posterior LM distribution subject to a semantic constraint. We will show that the problem can be reduced to that of computing expectations, which we then show how to estimate by performing exact and efficient probabilistic inference in an approximate LM induced by a singular model sample and a single evaluation of the verifier.

3 Great Expectations

We start by assuming access to the LM distribution, denoted by p , a sequence-level verifier ϕ_a for attribute a , and a prefix $\mathbf{y}_{1:i}$ where each token y_j assumes values in vocabulary \mathbb{V} . Our goal is then to sample from the LM distribution p a generation $\mathbf{y}_{i+1:T}$ subject to the constraint that the attribute a holds on the entire sequence *i.e.*, $\phi_a(\mathbf{y}_{1:i} \circ \mathbf{y}_{i+1:T}) \in \{0, 1\}$. That entails sampling a generation that fulfills two distinct desiderata: we expect the generation to be linguistically sound, or fluent as measured by a model’s perplexity, *and* to satisfy attribute a . That is, we are interested in sampling from the LLM distribution conditioned on the event that the sample belongs to the set of *all* sequences $\mathbf{y}_{1:T}$ that satisfy the attribute a , which we denote by $\mathcal{A} := \{\mathbf{y}_{1:T} \mid \phi_a(\mathbf{y}_{1:T}) = 1\}$. We can then write the target sampling distribution as

$$p(\mathbf{y}_{i+1:T} \mid \mathcal{A}, \mathbf{y}_{1:i}) \stackrel{(a)}{=} \frac{p(\mathbf{y}_{i+1:T}, \mathcal{A} \mid \mathbf{y}_{1:i})}{p(\mathcal{A} \mid \mathbf{y}_{1:i})} \stackrel{(b)}{=} \frac{p(\mathbf{y}_{i+1:T} \mid \mathbf{y}_{1:i}) \cdot \phi_a(\mathbf{y}_{1:i} \circ \mathbf{y}_{i+1:T})}{\sum_{\mathbf{y}_{i+1:T}} p(\mathbf{y}_{i+1:T} \mid \mathbf{y}_{1:i}) \cdot \phi_a(\mathbf{y}_{1:i} \circ \mathbf{y}_{i+1:T})}, \quad (2)$$

where equality (a) follows by the definition of conditional probability, and equality (b) follows by the definition of marginal probability. Intuitively, Equation (2) gives us a simple, albeit impractical, recipe for sampling from the LM distribution conditioned on attribute a : we enumerate all possible generations given the prefix, zeroing out all generations that violate a according to ϕ_a , followed by renormalization. In practice, for a given input $\mathbf{y}_{1:T}$ and attribute a , there is some *uncertainty* associated with $\phi_a(\mathbf{y}_{1:T})$. That is, we will assume access to a model’s estimate $p(\phi_a(\mathbf{y}_{1:T}) = 1) \in [0, 1]$ of whether $\mathbf{y}_{1:T}$ satisfies attribute a . Consequently, in a slight abuse of notation, we will redefine $\phi_a(\cdot)$ to be $p(\phi_a(\mathbf{y}_{1:T}) = 1)$, which should henceforth be thought of as a *probabilistic* verifier for the attribute a . Under this new definition of $\phi_a(\cdot)$, Equation (2) can be seen as reweighing each continuation with the probability of satisfying attribute a , followed by renormalizing the distribution.

State-of-the-art LMs, such as Llama 3 (Grattafiori et al., 2024) and GPT-4 (Achiam et al., 2024)) are autoregressive, so it is useful to rewrite Equation (2) in terms of the next tokens,

$$p(y_{i+1} \mid \mathcal{A}, \mathbf{y}_{1:i}) = \frac{p(y_{i+1} \mid \mathbf{y}_{1:i}) \cdot p(\mathcal{A} \mid \mathbf{y}_{1:i} \circ y_{i+1})}{p(\mathcal{A} \mid \mathbf{y}_{1:i})} \quad (3)$$

$$= \frac{p(y_{i+1} \mid \mathbf{y}_{1:i}) \mathbb{E}_{p(\cdot \mid \mathbf{y}_{1:i+1})} [\phi_a(\mathbf{y}_{1:i} \circ \mathbf{y}_{i+1:T})]}{\mathbb{E}_{p(\cdot \mid \mathbf{y}_{1:i})} [\phi_a(\mathbf{y}_{1:i} \circ \mathbf{y}_{i+1:T})]}, \quad (4)$$

where Equation (3) follows by the definition of conditional probability and Equation (4) follows by the definition of marginal probability and expectations. It is important to note that, since \mathcal{A} is defined as the set of all sequences $\mathbf{y}_{1:T}$ that satisfy a , the expectations, both in the numerator and in the denominator range over sequences of length T , requiring that we marginalize over all future continuations of length $T - i$ and $T - (i + 1)$, respectively. Intuitively, at every generation step we need to “look ahead” to determine the probability that the constraint is violated given the current choice of next token. If the probability is high, we discount the current choice, and if it is low, then we reinforce the current choice. Previous methods have approached this intractable expectation by either learning look-ahead functions parameterized by neural networks, or by sampling. Next, we will show how to compute the above expectation in closed form by relaxing the target distribution.

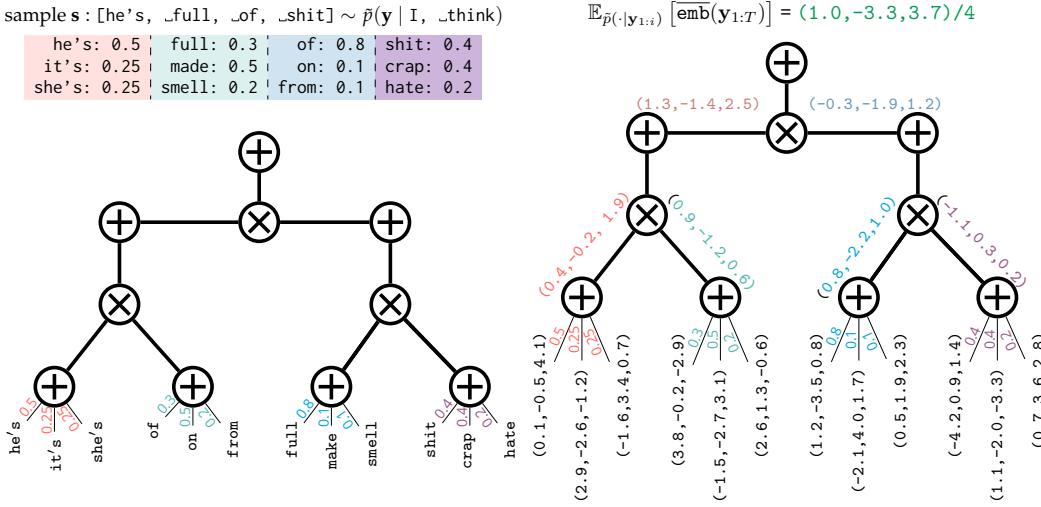


Figure 2: **A technical overview of our approach.** (top left) We start by sampling an approximate generation s using Gibbs sampling \tilde{p} conditioned on the prefix from the model’s marginal conditionals, $p(\mathbf{y}_i \mid \mathbf{y}_{-i}) \forall i$. Conditioned on s , the models marginal conditionals induce a distribution on all generations, assigning higher probabilities to similar sentences and lower probabilities for dissimilar sentences, which we visualize for the top-3 tokens for clarity of exposition. (bottom left) We can parameterize a *circuit* using the above distribution, yielding a closed-form, tractable representation of probability distribution defined in Equation (6), where read left to right, every leaf node corresponds to a categorical distribution on y_i (right) Such a representation enables us to compute the expected embeddings w.r.t. the distribution in the neighborhood of the sample s by substituting token embedding for corresponding embeddings at leaf nodes, computing weighted sums of embeddings at sum nodes, and taking sums at product nodes. This allows us to plug the expected embedding into Equation (8) to yield the constraint probability.

4 Semantic Probabilistic Control

The computational hardness of the expectations that we introduced in Equation (4) can intuitively be attributed to the *lack of structure* along two distinct dimensions.

First, is the *lack of structure to the distribution*. Consider computing the probability that a sequence of length T ends in the word “love”. Computing such a probability under the autoregressive distribution requires that we marginalize over all possible sequences ending in “love”, roughly $O(|V|^T)$. In fact, computing such probability is known to be computationally intractable (Roth, 1993). Contrast that with a fully-independent³ distribution, where we can simply query the network for the probability of a given token in constant time. Clearly there is a tension here: fully-independent distributions, while easier to reason about, are not expressive and therefore do not make for good LMs, whereas autoregressive distributions are harder to reason about, but a lot more expressive, and achieve SoTA language modeling.

The second dimension is the *lack of structure to the constraint*. Recall that we have assumed ϕ_a to be a neural network, which prior work has shown to be computationally intractable to decompose over sequences (Shi et al., 2020)⁴. That is, given $\phi_a(\mathbf{y}_{1:i})$ for a prefix $\mathbf{y}_{1:i}$, we know of no way of efficiently extending $\phi_a(\mathbf{y}_{1:i})$ to $\phi_a(\mathbf{y}_{1:i} \circ \mathbf{y}_{i+1})$ by only processing the new element \mathbf{y}_{i+1} and reusing the result of the previous evaluation $\phi_a(\mathbf{y}_{1:i})$.

³where $p(\mathbf{y}_{1:T}) = \prod_{i=1}^T p(\mathbf{y}_i)$, i.e., the probability of a token is independent from all other tokens.

⁴in fact, the problem remains intractable even assuming ϕ_a is a single neuron (Khosravi et al., 2019).

Algorithm 1 SConE

```

1: Input: Verifier  $\phi_a$ , LM distribution
    $p(y_i \mid y_{1:i})$ , prefix  $y_{1:i}$ , max length  $T$ 
2: Output:  $p(y_{i+1} \mid y_{1:i}, \mathcal{A})$ 
   ▷ Expand the batch to include top-k tokens
3:  $\text{top}_k = \arg \max_k p(y_i \mid y_{1:i})$ 
4:  $y_{1:i+1} = y_{1:i}.\text{expand}(n, \text{top}_k)$ 
   ▷ Get N samples  $\tilde{s}$  from  $p(y_{i+2:T} \mid y_{1:i+1})$ 
5:  $\tilde{s}^1, \dots, \tilde{s}^N \sim \text{GibbsSampler}(y_{1:i+1}, p)$ 
   ▷ Estimate prob  $q$  of satisfying constraint
6:  $q = \text{zeros}(\text{top}_k)$ 
7: for each  $\tilde{s}$  in  $\tilde{s}^1, \dots, \tilde{s}^N$  do
8:    $\tilde{p}_{\text{cond}} = \text{CondMarginals}(p, \tilde{s}_{i+2:T})$ 
9:    $\nabla \phi_a = \text{LinearizeVerifier}(\phi_a, \tilde{s})$ 
10:   $q[\tilde{s}_{i+1}] += \text{EstimateProb}(\tilde{p}_{\text{cond}}, \phi_a, \nabla \phi_a)$ 
11: end for
   ▷ Renormalize  $q$ 
12:  $\log q = q.\text{log\_softmax}()$ 
   ▷ Reweight the LM distribution
13:  $w = \log p(y_{i+1} \mid y_{1:i}) + \log q$ 
14:  $p^* = \text{Categorical}(\text{weights} = w)$ 
15: return  $p^*$ 

```

Algorithm 2 LinearizeVerifier

```

1: Input: Verifier  $\phi_a$ , Sample  $s$ 
2: Output: Gradient of  $\phi_a$  w.r.t.  $s$  embedding
   ▷ Obtain embeddings for  $s$ 
3:  $\text{emb\_layer} = \phi_a.\text{get\_input\_embeddings}()$ 
4:  $\text{emb} = \text{emb\_layer}(s)$ 
   ▷ Collect gradient of  $\phi_a$  w.r.t. to  $\text{emb}$ 
5:  $\text{score} = \phi_a(\text{emb}).\text{sum}()$ 
6:  $\text{grad} = \text{autograd.grad}(\text{score}, \text{emb})$ 
7: return  $\text{grad}$ 

```

Algorithm 3 EstimateProb

```

1: Input: Conditional marginals  $\tilde{p}_{\text{cond}}$ , Verifier  $\phi_a$ , Gradient  $\nabla_{\text{emb}(s)} \phi_a$ ,  $\text{embs} := [\overline{\text{emb}}(y_{1,1}), \dots, \overline{\text{emb}}(y_{i,\|\mathcal{V}\|})]$ , score  $\phi_a(s)$ ,  $T$ 
2: Output:  $p(\mathcal{A} \mid y_{1:i})$ 
   ▷ Compute expected embedding
3:  $\text{exe} = 0$ 
4: for  $i$  in  $1, \dots, T$  do
5:    $\text{exe} += \text{embs}[\dots, \text{None}] \cdot \tilde{p}_{\text{cond}}[:, i : i + 1, :]$ 
6: end for
7:  $\text{exe} = \text{exe}.\text{mean}(0)$ 
   ▷ First-order Taylor expansion about  $s$ 
8: return  $\phi_a(s) + \nabla_{\text{emb}(s)} \phi_a \cdot (\text{exe} - \overline{\text{emb}}(s))$ 

```

4.1 Locally Contextualized Distribution

To sidestep the hardness of the autoregressive distribution, we move towards the tractability of fully-independent distributions, while retaining as much of the contextual information. Therefore, we consider the *pseudolikelihood* of a sentence (Besag, 1975; Ahmed et al., 2023),

$$p(y_{1:T}) \approx \tilde{p}(y_{1:T}) := \prod_i p(y_i \mid y_{-i}), \quad (5)$$

where y_{-i} denotes $y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n$. Unfortunately, Equation (5) does not ensure tractability, seeing that different sentences would depend on different sets of conditionals. We define the pseudolikelihood of a sentence y in the semantic neighborhood of a sentence \tilde{y}

$$\tilde{p}_{\tilde{y}}(y) := \prod_i p(y_i \mid \tilde{y}_{-i}) \quad (6)$$

which can be thought of as the *contextualized probability* of a sentence y given the context \tilde{y} . That is, Equation (6) calculates the probability of sequence y by taking the product of probabilities of each token y_i , crucially conditioning each token y_i not on the preceding tokens of y , but on the context surrounding position i within \tilde{y} (specifically, \tilde{y} excluding its i -th token, denoted \tilde{y}_{-i}). Therefore, \tilde{y} acts as a contextual anchor for evaluating y under this measure. Intuitively, sentences y that semantically or structurally align well with the specific token-level contexts provided by \tilde{y} are expected to yield a higher pseudolikelihood $\tilde{p}_{\tilde{y}}(y)$.

4.2 Bridging Samples and Expectations: A Tangential View

Next, we turn our attention to address the *hardness of the verifier* ϕ_a . In particular, given an LM sample $s \sim p(y_{i+1:T} \mid y_{1:i})$ and access to a verifier ϕ_a , we leverage gradient information obtained during the evaluation of $\phi_a(s)$, coupled with the contextualized probability distribution in Equation (6), to approximate $\mathbb{E}_{p(\cdot \mid y_{1:i})} [\phi_a(y_{1:T})]$, the constraint probability.

We denote by $\text{emb} : \mathcal{V} \mapsto \mathbb{R}^d$ an embedding function that maps each token onto a d -dimension vector and let $\overline{\text{emb}}(\mathbf{y})$ denote the average token-wise embedding.⁵ Then, we can approximate Equation (4) using a first-order Taylor expansion of ϕ_a about the LM sample \mathbf{s}

$$\mathbb{E}_{\tilde{p}(\cdot|\mathbf{y}_{1:i})} [\phi_a(\mathbf{y}_{1:T})] \approx \mathbb{E}_{\tilde{p}(\cdot|\mathbf{y}_{1:i})} [\phi_a(\mathbf{s}) + \nabla \phi_a(\mathbf{s}) \cdot (\overline{\text{emb}}(\mathbf{y}_{1:T}) - \overline{\text{emb}}(\mathbf{s}))]. \quad (7)$$

Using the linearity of expectation, we can further simplify expression, obtaining

$$\mathbb{E}_{\tilde{p}(\cdot|\mathbf{y}_{1:i})} [\phi_a(\mathbf{y}_{1:T})] \approx \phi_a(\mathbf{s}) + \nabla \phi_a(\mathbf{s}) \cdot (\mathbb{E}_{\tilde{p}(\cdot|\mathbf{y}_{1:i})} [\overline{\text{emb}}(\mathbf{y}_{1:T})] - \overline{\text{emb}}(\mathbf{s})). \quad (8)$$

We have now managed to *reduce the problem of estimating the constraint probability*, given by the expectations in Equation (4) to the problem of computing an average sentence embedding w.r.t. an approximate LM distribution \tilde{p} , followed by simple arithmetic operations. We will next show how we can efficiently compute the expected sentence embedding.

4.3 From Sequence Probabilities to Average Embeddings

We appeal to knowledge compilation, a class of methods that transform, or *compile*, a function into a tractable target form which represents functions as parameterized computational graphs, or *circuits*. By enforcing certain structural properties on the compiled circuits, we can enable the tractable computation of corresponding classes of probabilistic queries. Thus, circuits provide a language for constructing and reasoning about tractable representations.

Formally, a *circuit* p over variables \mathbf{Y} is a parameterized computational graph encoding a function $p(\mathbf{Y})$. Each node n in the graph encodes a parameterized function $p_n(\text{vars}(n))$ over variables $\text{vars}(n) \subseteq \mathbf{Y}$, also known as its *scope*. Each inner node in the graph is a sum or a product node, and each leaf node encodes a tractable input distribution over its scope. Each inner unit n (*i.e.*, product or sum node) receives inputs from other units, denoted $\text{in}(n)$.

A circuit is *decomposable* if the inputs of every product node depends on disjoint sets of variables, *i.e.*, for $n = c_1 \otimes c_2$, $\text{vars}(c_1) \cap \text{vars}(c_2) = \emptyset$. Intuitively, decomposable product nodes encode local factorizations over variables of the function. We assume that decomposable product nodes always have two inputs, a condition that is enforceable on any circuit in exchange for a polynomial increase in its size (Vergari et al., 2015; Peharz et al., 2020).

A second property is *smoothness*. A circuit is *smooth* if the inputs of every sum node depend on the same set of variables, *i.e.*, for $n = \bigoplus_i \theta_i \cdot c_i$, $\text{vars}(c_i) = \text{vars}(c_j) \forall i, j$. Decomposability and smoothness are sufficient and necessary for tractable integration over arbitrary sets of variables in a single pass, as they allow larger integrals to decompose into smaller ones. Given a circuit for a distribution \tilde{p} , the expected embedding can then be computed by traversing the circuit bottom-up, substituting token embedding for corresponding embeddings at leaf nodes, computing weighted sums of embeddings at sum nodes, and taking sums (in essence, concatenating embeddings) at product nodes, as can be seen in Figure 2.

4.4 Closing the Loop

Our full algorithm is given in Algorithm 1. We start by truncating the next-token distribution using top-k or top-p, as is common place in modern autoregressive LMs, where we use top-k for clarity of exposition. We then proceed by simulating a continuation for each of the possible top-k tokens, each produced using a masked LM and Hogwild! Gibbs sampling⁶, to avoid expensive autoregressive sampling from the LM. We then proceed by computing the contextualized probability of each sample \mathbf{V}_i and the gradient of the verifier w.r.t. the sample embedding $\nabla_{\text{emb}(\mathbf{s})} \phi_a$, used to estimate the constraint probability. Having computed the constraint probability, we reweigh the next-token distribution to account for the constraint being satisfied, and renormalize to obtain the conditional next-token distribution.

⁵w.l.o.g, we assume this embedding can be extracted directly from the embedding layer of the verifier, *i.e.*, $\phi_a(\mathbf{s}) := \phi_a(\text{emb}(\mathbf{s}_1), \dots, \text{emb}(\mathbf{s}_T))$.

⁶We refer the reader to Appendix D for more details.

5 Related Work

Recent advances in controllable generation with LMs have spurred a wide range of approaches, which we summarize below. These approaches can be roughly classified into three different categories: *training-time*, *prompting*, and *decoding-time* approaches.

Training-time approaches. A subset of the approaches seeks to exert control by fine-tuning or reinforcement learning via some set of data that more closely mirrors the target task, such as via reinforcement learning from human feedback (RLHF) (Ziegler et al., 2020; Stiennon et al., 2020b; Bai et al., 2022; Ouyang et al., 2022) or from symbolic knowledge (Ahmed et al., 2023), but these approaches come with challenges such as hyperparameter sensitivity and distributional collapse (Zheng et al., 2023; Zhu et al., 2023; Xiong et al.). Some of these drawbacks can be mitigated by utilizing on-policy data (Tajwar et al., 2024) and imposing a KL penalty that penalizes shifting an LM too far from its prior distribution, casting optimization as variational inference (Korbak et al., 2022; Amini et al., 2025).

Prompting approaches. Another class of approaches focuses on guiding the distribution implicitly via modifications in the prompt (Ashok & Poczos, 2024). To this end, control can be exerted by either verbally expressing the constraints in the prompt (Chen et al., 2022; Zhou et al., 2023; Ashok & Poczos, 2024), or through the use of examples (Poesia et al., 2022; Zhou et al., 2023). In addition to introducing minimal computation overhead and producing good quality text (Zhou et al., 2023; Ashok & Poczos, 2024), prompting approaches are also more flexible, since complex constraints can be easily integrated in the prompt without further training or expensive data curation. Nonetheless, constraint satisfiability using prompting-based methods is not guaranteed (Zhou et al., 2023) and depends heavily on the instruction following capabilities of the LM (Jiang et al., 2024; He et al., 2024).

Decoding-time approaches. A popular decoding-time approach is to perform token-level modifications at each step and, for that reason, frequently referred to as *locally constrained decoding* (Loula et al., 2025). Methods to locally constrained decoding either mask out specific tokens or heuristically reweigh tokens such that the constraints are more likely to be satisfied. Examples include banning specific words (Gehman et al., 2020), using context-free grammars (Poesia et al., 2022; Geng et al., 2023; Willard & Louf, 2023; Beurer-Kellner et al., 2023; Lundberg et al., 2024; Beurer-Kellner et al., 2024), or through the combination of boolean algebra with search algorithms (Hokamp & Liu, 2017; Anderson et al., 2017; Post & Vilar, 2018; Hu et al., 2019; Lu et al., 2021; 2022; Qin et al., 2022). Note, however, that while setting token-level restrictions can be effective at exerting syntactic control over LMs, these are insufficient to capture the richer and subtler nuances of semantic constraints.

In fact, semantic control approaches resort to attribute “scorers” to estimate how likely the constraint is under a given input, and then use those estimates to reweigh the per-token distribution of the base LM. Previously proposed methods include combining the conditional distributions of different LMs with opposing behaviors, such as a toxic expert and a non-toxic expert (Schick et al., 2021; Liu et al., 2021; Li et al., 2023; Dekoninck et al., 2024), and using an attribute discriminator (*i.e.*, constraint verifier) to reweigh the base LM conditional distribution (Holtzman et al., 2018; Krause et al., 2021). The gradients of attribute discriminators have also been to induce changes the base LM through changes to the LM weights (Dathathri et al., 2020; Liu et al., 2020; Wallace et al., 2019; Zhang et al., 2024b). Although effective, locally constrained decoding approaches often introduce greedy (potentially sub-optimal) approximations that distort the distribution (Loula et al., 2025; Ma et al., 2025). Conversely, sample-reweigh approaches consist of first sampling complete sequences and then reweigh them using a constraint verifier (Stiennon et al., 2020a; Krishna et al., 2022; Sun et al., 2024; Ichihara et al., 2025; Amini et al., 2025). While constraints are imposed globally in sample reweighing approaches, they do not benefit from finer-grained constraint information during generation and, hence, require a larger number of samples to find high-quality generations that comply with the constraints (Loula et al., 2025).

Another line of work performs approximate inference in exact models via sampling (Miao et al., 2019; Zhang et al., 2020; Kumar et al., 2022; Poesia et al., 2022; Qin et al., 2022; Du et al., 2024), and, more recently, via more effective Sequential Monte Carlo (SMC) methods, which maintain a set of samples that evolve through time. The evolution of the samples accounts

Table 1: **Evaluation of the quality and toxicity of Llama-3.2 (1B) generations when steered to be non-toxic and toxic**, respectively. Toxicity is evaluated on 400 prompts RealToxicityPrompts using the toxicity verifier ϕ_{toxicity} (Logacheva et al., 2022). PPL refers to the perplexity of Meta-Llama-3-70B on the model generations. We report **Expected Maximum Toxicity**: the maximum toxicity across generations, and **Toxicity Probability**: the probability of a toxic generation, both computed across 10 generations per prompt. We expect both metrics to be lower (\downarrow) when steering the base LM towards non-toxic generations (**detoxify**) and higher (\uparrow) when steering the base LM towards toxic generations (**toxify**).

Objective	Method	Toxic Prob. (\downarrow, \uparrow)			Exp. Max. Toxicity (\downarrow, \uparrow)			PPL (\downarrow)
		Full	Non-toxic	Toxic	Full	Non-toxic	Toxic	
	random	37.25	10.00	64.50	37.11	13.17	61.05	12.18
	beamsearch	17.25	3.00	31.50	18.22	4.34	32.09	8.00
detoxify	BoN	2.75	1.00	4.50	4.90	1.91	7.89	15.46
	SConE (ours)	00.25	00.50	00.00	01.85	1.30	2.40	14.88
toxify	BoN	62.50	37.00	88.00	61.36	39.62	83.11	13.97
	SConE (ours)	93.75	88.00	99.50	91.15	85.75	96.55	23.87

not only for the sample likelihood under the base LM, but also for constraint information that can be provided either by learnable twist functions (Zhao et al., 2024) or by evaluating the constraint verifier on partial sequences (Lew et al., 2023; Loula et al., 2025).

6 Experiments

We empirically evaluate the effectiveness of the proposed method across numerous open-ended generation tasks, including text detoxification, controlled sentiment generation, and topic steering. Section 6 introduces specific details of our methods, baselines, and metrics. Task-specific details, such as dataset and constraint verifiers, and results for the toxicity, sentiment, and topic experiments are described in Sections 6.1, 6.2, C.1, respectively.

Experimental Setup

Baselines. To validate our method, we compare it against two sampling-based baselines: random, which consists of sampling outputs autoregressively from a base LM, and beamsearch, which leverages information about the top K most likely continuations under a base LM to greedily select the next token. Additionally, we evaluate **Best-of-N rejection sampling (BoN)** (Stiennon et al., 2020a), a popular training-free method for language model control which has been shown to be competitive to RLHF-based methods (Amini et al., 2025). Like our proposed method, BoN exploits non-lexical constraint verifiers to exert semantic control on the base LM. However, it does so by first sampling N continuations from the base LM and selecting one that maximizes the verifier.⁷ We refer to Appendix B for more details.

Metrics. In line with prior work (Gehman et al., 2020; Ahmed et al., 2025), we report **Perplexity (PPL)** as a measure of sample quality, specifically, we use Meta-Llama-3-70B. Intuitively, effective control methods should yield generations that satisfy the constraint but that are also high quality, *i.e.*, low perplexity.

The primary constraint satisfaction metric that we report is the **Average ϕ_a score**. This metric can be defined as the average verifier score across all model generations. Intuitively, because this verifier is being used to steer control during generation, it can be interpreted as the *ground truth* measure of the desired semantic attribute a (*e.g.*, toxicity, positive sentiment, topic). As such, we expect effective control methods to achieve high Average ϕ_a scores, especially when compared to uncontrolled baselines like random.

⁷For a fair comparison, we use the same decoding settings as in our method’s initialization.

Table 2: **Evaluation of quality and sentiment of GPT2-IMDB generations when steered using a positive sentiment constraint $\phi_{\text{sentiment}}$.** Sentiment is evaluated on 600 prompts from the IMDB test set using a sentiment verifier (Maas et al., 2011), spanning equal number of positive and negative reviews. Results are discriminated by the **Full** set of prompts, the **Negative** subset, and the **Positive** subset. All metrics are calculated using 10 different generations per prompt. **PPL** refers to the perplexity of Meta-Llama-3-70B on the model generations using 10 different seeds; In line with Rafailov et al. (2023); Amini et al. (2025), we report the average sentiment score, the sentiment score is greater than 0.8 in 9 out 10 generations (**Sentiment Prob.**), and the expected minimum sentiment score (**Exp. Min. Sentiment**).

Method	Avg $\phi_{\text{sentiment}}$ (↑)			Sentiment Prob. (↑)			Exp. Min. Sentiment (↑)			PPL (↓)
	Full	Neg	Pos	Full	Neg	Pos	Full	Neg	Pos	Full
random	57.10	53.16	61.04	95.50	95.33	95.67	12.83	10.78	14.87	21.18
beamsearch	58.83	50.83	66.82	58.83	48.33	69.33	44.46	37.21	51.71	3.96
BoN	60.66	55.17	66.14	95.83	93.33	98.33	15.24	11.70	18.77	10.84
SConE (ours)	93.06	92.73	93.37	100.00	100.00	100.00	84.50	83.18	85.82	20.96

As additional measures of constraint satisfaction, we report metrics that capture the expected worst-case and the empirical probability of constraint satisfaction (Gehman et al., 2020). Assuming that each prompt x is associated with multiple generations, the **expected worst score metric** is calculated by computing the worst constraint score ϕ_a across all generations for x , and, then taking the average over all evaluation prompts. Similarly, the **constraint probability** metric represents the fraction of evaluated prompts for which at least one of its generations satisfies the constraint above a user-defined threshold (*i.e.*, $\mathbb{1}[\phi_a(\mathbf{y}) \geq \tau_a]$).

6.1 Controlled Toxicity Generation

In this section, we compare the performance of different methods in steering the toxicity of a small Llama-3.2 (1B) (Grattafiori et al., 2024). We do so by prompting the LM with 400 natural occurring prompts from RealToxicityPrompts (Gehman et al., 2020). We randomly select 200 *Toxic* and 200 *Non-toxic* prompts from RealToxicityPrompts and use them in both toxification and detoxification settings, sampling 10 generations of up to 25 tokens per prompt. Evaluation and toxicity steerability are both conducted using a RoBERTa-based binary classifier ϕ_{toxicity} , finetuned for toxicity detection (Logacheva et al., 2022). To steer models to generate non-toxic outputs, we set them to maximize $1 - \phi_{\text{toxicity}}$.

Detoxification Task. Table 1 summarizes the results for the detoxification task, discriminated by prompt type. Intuitively, effective semantic control methods should be able to generate non-toxic outputs, *i.e.*, minimize the toxicity metrics, irrespective of the toxicity of the prompt type. Overall, we observe that the uncontrolled baselines random and beamsearch, still lead to toxic continuations even when prompted with non-toxic inputs. While beamsearch seems to lower both toxicity and perplexity, we find that this is explained by degenerate outputs characterized by repetition (Holtzman et al., 2020). Contrastingly, we find that BoN is very effective at detoxifying LM generations: reducing the average worst-case toxicity down 4.90 with minimal penalty in perplexity (3.28 points). While this represents a big improvement over the uncontrolled baselines, we find that our method is able to further achieve a 3-fold reduction in terms of the average worst case toxicity in toxic prompt and reduce the probability of a toxic generation to a negligible amount (up to 0.50).

Toxification Task. We now move to the opposite task: given a naturally occurring prompt, are methods able to steer the base LM towards more toxic inputs? Table 6.1 shows the toxicity results for the semantic control methods. While both methods are able to substantially increase both the worst-case toxicity and the likelihood of sampling toxic outputs from Llama-3.2 (1B), we find that SConE systematically is far more effective than BoN with +30% gap toxicity increase across both toxicity metrics. Much of this performance gap appears to stem from the non-toxic subset, for which the base LM is less predisposed to generate toxic outputs. As such, methods like rejection sampling that use the constraint verifier ϕ_{toxicity}

Table 3: Evaluation of quality and topic adherence of Llama-3.2 (1B) generations when controlled for specific topics. Topic adherence is evaluated on 300 prompts spanning 6 topics ϕ_{topic} (Wettig et al., 2025). We report perplexity (PPL), the average ϕ_{topic} score (Avg ϕ_{topic}), the fraction of examples for which the topic score is greater than 0.8 in 90% or more of the generations (Topic Prob.), and the expected minimum topic score (Exp. Min. Topic).

Method	Topic Prob. (\uparrow)	Exp. Min. Topic (\uparrow)	Avg ϕ_{topic} (\uparrow)	PPL
random	86.20	83.91	91.87	6.16
beamsearch	87.47	90.35	91.63	3.78
BoN	95.40	95.18	97.52	8.42
SConE (ours)	98.40	96.71	99.07	7.39

to rerank the base LM generations are less likely to succeed for low probability semantic constraints. This also provides an explanation for the increase in perplexity for **SConE**.

6.2 Controlled Sentiment Generation

Next, we compare the steerability of the different methods when generating reviews with positive sentiment (Rafailov et al., 2023; Zhao et al., 2024; Amini et al., 2025). Focusing on movie reviews, we prompt GPT2-IMDB with 600 arbitrarily chosen prompts from the IMDB test set (Maas et al., 2011). Building on previous work (Rafailov et al., 2023; Amini et al., 2025), we use the original reviews in the IMDB dataset to create the prompts by randomly splitting them into prefixes of 2 to 8 words. We also adopt the same BERT-based classifier as our sentiment verifier $\phi_{\text{sentiment}}$.⁸ Given that this model was fine-tuned on the IMDB training data, we expect it to be a strong and reliable sentiment predictor for this task.

Positive Movie Review Generation Task. In the context of positive movie review generation, we would like to ensure that most of GPT2-IMDB’s generations are positive.⁹ Once more, as observed in Table 2, the uncontrolled baselines—random and beamsearch—struggle to generate positive reviews. Specifically, as emphasized by the worst case metric, Expected Minimum Sentiment, GPT2-IMDB-generated reviews with no control can be fairly negative (< 52 across all prompts), especially in the negative subset (< 38%). BoN drastically improves upon the uncontrolled baselines, increasing the Sentiment Probability to about 70.83% and improving the average lowest sentiment score to 70.79%. Still, we find that **SConE** further improves (about 14% points average improvement in both metrics) the overall worst-case sentiment and the chances of producing positive reviews at least 90% of the time.

6.3 Controlled Topic Generation

Lastly, we evaluate the methods on their ability to control for the topic of LM generations. We choose 6 diverse topics from the recently taxonomy concerning the web structure (Wettig et al., 2025), including frequent (e.g., *Finance & Business* and *Politics*) and less frequent topics (e.g., *History*, *Industrial*). For each topic, we randomly select 50 different examples from the TopicAnnotations-Llama-3.1-405B-FP8 (Wettig et al., 2025) test set, breaking them into prefixes of 8 to 12 words. Each prefix is used to sample a maximum of 60 tokens.

Topic Generation Task. In general, we find that uncontrolled baselines achieve a fairly high average constraint score ($\geq 91\%$), which may be explained by the use of longer prefixes during generation. We find this to be the case for most examples (see examples in Appendix C.1). Nonetheless, the discrepancy between uncontrolled and controlled methods is still visible with the latter achieving 7%-8% higher average constraint scores. Remarkably, we find **SConE** is not only able to improve upon BoN, achieving an average score of 98.89% but also produces higher quality generations as emphasized by the lower perplexity.

⁸<https://huggingface.co/lvwerra/distilbert-imdb>

⁹In line with Maas et al. (2011), we consider a review to be positive iff $\phi_{\text{sentiment}}(\mathbf{y}) \geq 0.8$.

7 Conclusion

In this paper, we introduced a training-free approach to semantic control of autoregressive language models. Our approach uses exact inference on an approximate distribution induced by an LM generation, using first-order information from a verifier to compute the expected constraint satisfaction for each of the possible next tokens. Our approach demonstrated a substantial improvement compared to previous approach on the tasks of controlling the toxicity, sentiment and topic of LM generations.

Ethics Statement

Our work investigates the problem of exerting semantic control over LM generations. While our method can be very societally beneficial, giving us more control over language models, we acknowledge that our method could be misused to produce harmful content. We look forward to exploring future work that places guardrails on LMs to prevent these pitfalls.

Acknowledgments

This work is supported by the DARPA ANSR program FA8750-23-2-0004, an NSF CAREER award number IIS-2046873 and an NSF award number 1900644. The conclusions are of the authors and do not reflect the official policy or position of DARPA or the U.S. Government.

Statement of Author Contributions

Kareem Ahmed: Conceived and developed the core research idea and the proposed approach. Wrote the introduction and technical sections of the paper. Wrote the code for computing the expected embedding and contributed to debugging the overall approach.

Catarina G. Belem: Implemented the primary codebase. Conducted all experiments and wrote the corresponding experimental section of the paper in addition to the related works.

Padhraic Smyth and Sameer Singh: Senior project leadership. Provided mentorship, supervision, and advisory support throughout the project. Offered critical feedback on the methodology and the manuscript. All authors read and approved the final manuscript.

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, et al. GPT4 Technical Report, 2024.
- Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. In *NeurIPS*, 2022.
- Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. A pseudo-semantic loss for autoregressive models with logical constraints. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=hVA1a20730>.
- Kareem Ahmed, Kai-Wei Chang, and Guy Van den Broeck. Controllable generation via locally constrained resampling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=8g4XgC8HPF>.
- Afra Amini, Tim Vieira, Elliott Ash, and Ryan Cotterell. Variational best-of-n alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=W9FZEQj3vv>.
- Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. Guided open vocabulary image captioning with constrained beam search. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 936–945, Copenhagen, Denmark, September 2017.

Association for Computational Linguistics. doi: 10.18653/v1/D17-1098. URL <https://aclanthology.org/D17-1098/>.

Dhananjay Ashok and Barnabas Poczos. Controllable text generation in the instruction-tuning era, 2024. URL <https://arxiv.org/abs/2405.01490>.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai feedback, 2022. URL <https://arxiv.org/abs/2212.08073>.

Julian Besag. Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society. Series D (The Statistician)*, pp. pp. 179–195, 1975.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proc. ACM Program. Lang.*, 7(PLDI), June 2023. doi: 10.1145/3591300. URL <https://doi.org/10.1145/3591300>.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding llms the right way: fast, non-invasive constrained generation. In *Proceedings of the 41st International Conference on Machine Learning*, ICML’24. JMLR.org, 2024.

Howard Chen, Huihan Li, Danqi Chen, and Karthik Narasimhan. Controllable text generation with language constraints, 2022. URL <https://arxiv.org/abs/2212.10466>.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic modeling, 2020.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=H1edEyBKDS>.

Jasper Dekoninck, Marc Fischer, Luca Beurer-Kellner, and Martin Vechev. Controlled text generation via language model arithmetic. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=SLw9fp4yI6>.

Li Du, Afra Amini, Lucas Torroba Hennigen, Xinyan Velocity Yu, Holden Lee, Jason Eisner, and Ryan Cotterell. Principled gradient-based MCMC for conditional sampling of text. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11663–11685. PMLR, 21–27 Jul 2024. URL <https://proceedings.mlr.press/v235/du24a.html>.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3356–3369, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.301. URL <https://aclanthology.org/2020.findings-emnlp.301/>.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 10932–10952, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.674. URL [https://aclanthology.org/2023.emnlp-main.674/](https://aclanthology.org/2023.emnlp-main.674).

Saibo Geng, Hudson Cooper, Michał Moskal, Samuel Jenkins, Julian Berman, Nathan Ranchin, Robert West, Eric Horvitz, and Harsha Nori. Jsonschemabench: A rigorous benchmark of structured outputs for language models, 2025. URL <https://arxiv.org/abs/2501.10868>.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models, 2024. URL <https://arxiv.org/abs/2407.21783>.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10864–10882, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.637. URL [https://aclanthology.org/2024.findings-emnlp.637/](https://aclanthology.org/2024.findings-emnlp.637).

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1141. URL <https://aclanthology.org/P17-1141/>.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1638–1649, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1152. URL <https://aclanthology.org/P18-1152/>.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL <https://openreview.net/forum?id=rygGQyrFvH>.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. Improved lexically constrained decoding for translation and monolingual rewriting. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 839–850, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1090. URL [https://aclanthology.org/N19-1090/](https://aclanthology.org/N19-1090).

Yuki Ichihara, Yuu Jinnai, Tetsuro Morimura, Kenshi Abe, Kaito Ariu, Mitsuki Sakamoto, and Eiji Uchibe. Evaluation of best-of-n sampling strategies for language model alignment. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL <https://openreview.net/forum?id=H4S4ETc8c9>.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.257. URL <https://aclanthology.org/2024.acl-long.257/>.

Pasha Khosravi, Yitao Liang, YooJung Choi, and Guy Van Den Broeck. What to expect of classifiers? reasoning about logistic regression with missing features. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI’19, 2019.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Terry Koo, Frederick Liu, and Luheng He. Automata-based constraints for language model decoding, 2024.

Tomasz Korbak, Ethan Perez, and Christopher Buckley. RL with KL penalties is better viewed as Bayesian inference. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, 2022.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. GeDi: Generative discriminator guided sequence generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4929–4952, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.424. URL <https://aclanthology.org/2021.findings-emnlp.424/>.

Kalpesh Krishna, Yapei Chang, John Wieting, and Mohit Iyyer. RankGen: Improving text generation with large ranking models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 199–232, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.15. URL <https://aclanthology.org/2022.emnlp-main.15/>.

Sachin Kumar, Biswajit Paria, and Yulia Tsvetkov. Gradient-based constrained sampling from language models. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 2251–2277, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.144. URL <https://aclanthology.org/2022.emnlp-main.144/>.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Alexander K. Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K. Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs, 2023. URL <https://arxiv.org/abs/2306.03081>.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12286–12312, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.687. URL <https://aclanthology.org/2023.acl-long.687/>.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *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)*, pp. 6691–6706, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.522. URL <https://aclanthology.org/2021.acl-long.522/>.

Michael Xieyang Liu, Frederick Liu, Alex Fiannaca, Terry Koo, Lucas Dixon, Michael Terry, and Carrie Cai. “we need structured output”: Towards user-centered constraints on large language model output. pp. 9, 2024.

Ruibo Liu, Guangxuan Xu, Chenyan Jia, Weicheng Ma, Lili Wang, and Soroush Vosoughi. Data boost: Text data augmentation through reinforcement learning guided conditional

generation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9031–9041, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.726. URL <https://aclanthology.org/2020.emnlp-main.726/>.

Varvara Logacheva, Daryna Dementieva, Sergey Ustyantsev, Daniil Moskovskiy, David Dale, Irina Krotova, Nikita Semenov, and Alexander Panchenko. ParaDetox: Detoxification with parallel data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6804–6818, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL <https://aclanthology.org/2022.acl-long.469>.

João Loula, Benjamin LeBrun, Li Du, Ben Lipkin, Clemente Pasti, Gabriel Grand, Tianyu Liu, Yahya Emara, Marjorie Freedman, Jason Eisner, Ryan Cotterell, Vikash Mansinghka, Alexander K. Lew, Tim Vieira, and Timothy J. O’Donnell. Syntactic and semantic control of large language models via sequential monte carlo. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=xoXn62FzD0>.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. NeuroLogic decoding: (un)supervised neural text generation with predicate logic constraints. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4288–4299, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.339. URL <https://aclanthology.org/2021.naacl-main.339/>.

Ximing Lu, Sean Welleck, Peter West, Liwei Jiang, Jungo Kasai, Daniel Khashabi, Ronan Le Bras, Lianhui Qin, Youngjae Yu, Rowan Zellers, Noah A. Smith, and Yejin Choi. NeuroLogic a*esque decoding: Constrained text generation with lookahead heuristics. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 780–799, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.57. URL <https://aclanthology.org/2022.naacl-main.57/>.

Scott Lundberg, Marco Ribeiro, Richard Edgar, and Harsha-Nori. Guidance: a guidance language for controlling large language models., 2024.

Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. Non-myopic generation of language models for reasoning and planning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=OoNazl6T7D>.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL <https://aclanthology.org/P11-1015/>.

Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Cgmh: constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI’19/IAAI’19/EAAI’19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33016834. URL <https://doi.org/10.1609/aaai.v33i01.33016834>.

Nguyen Nhat Minh, Andrew Baker, Clement Neo, Allen G Roush, Andreas Kirsch, and Ravid Shwartz-Ziv. Turning up the heat: Min-p sampling for creative and coherent LLM outputs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=FBkpCyujTS>.

Feng Niu, Benjamin Recht, Christopher Re, and Stephen J. Wright. Hogwild! a lock-free approach to parallelizing stochastic gradient descent. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, 2011.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference of Machine Learning*, 2020.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*, 2022. URL <https://openreview.net/forum?id=KmtVD97J43e>.

Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1314–1324, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1119. URL <https://aclanthology.org/N18-1119/>.

Lianhui Qin, Sean Welleck, Daniel Khashabi, and Yejin Choi. COLD decoding: Energy-based constrained text generation with langevin dynamics. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL <https://openreview.net/forum?id=TiZYrQ-mPup>.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL <https://openreview.net/forum?id=HPuSIXJaa9>.

Dan Roth. On the hardness of approximate reasoning. In *IJCAI*, pp. 613–619. Morgan Kaufmann, 1993.

Christopher De Sa, Chris Re, and Kunle Olukotun. Ensuring rapid mixing and low bias for asynchronous gibbs sampling. In *Proceedings of The 33rd International Conference on Machine Learning*.

Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424, 2021. doi: 10.1162/tacl_a_00434. URL <https://aclanthology.org/2021.tacl-1.84/>.

Weijia Shi, Andy Shih, Adnan Darwiche, and Arthur Choi. On tractable representations of binary neural networks. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2020.

Alexander Smola and Shravan Narayananurthy. An architecture for parallel topic models. *Proceedings of VLDB Endow.*, 2010.

Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NeurIPS '20, Red Hook, NY, USA, 2020a. Curran Associates Inc. ISBN 9781713829546.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, 2020b.

Hanshi Sun, Momin Haider, Ruiqi Zhang, Huitao Yang, Jiahao Qiu, Ming Yin, Mengdi Wang, Peter Bartlett, and Andrea Zanette. Fast best-of-n decoding via speculative rejection, 2024. URL <https://arxiv.org/abs/2410.20290>.

Jiao Sun, Yufei Tian, Wangchunshu Zhou, Nan Xu, Qian Hu, Rahul Gupta, John Frederick Wieting, Nanyun Peng, and Xuezhe Ma. Evaluating large language models on controlled generation tasks, 2023.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Simplifying, regularizing and strengthening sum-product network structure learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 343–358. Springer, 2015.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *NeurIPS*, 2021.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2153–2162, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1221. URL <https://aclanthology.org/D19-1221/>.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Nathan Cooper, Griffin Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference, 2024. URL <https://arxiv.org/abs/2412.13663>.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation, 2025. URL <https://arxiv.org/abs/2502.10341>.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *ArXiv*, abs/2307.09702, 2023.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Honghua Zhang, Po-Nien Kung, , Masahiro Yoshida, Nanyun Peng, and Guy Van den Broeck. Adaptable logical control for large language models. In *NeurIPS*, 2024a.

Maosen Zhang, Nan Jiang, Lei Li, and Yexiang Xue. Language generation via combinatorial constraint satisfaction: A tree search enhanced Monte-Carlo approach. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1286–1298, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.115. URL <https://aclanthology.org/2020.findings-emnlp.115/>.

Yuansen Zhang, Xiao Wang, Tianze Chen, Jiayi Fu, Tao Gui, and Qi Zhang. P4: Plug-and-play discrete prompting for large language models personalization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9129–9144, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.541. URL <https://aclanthology.org/2024.findings-acl.541/>.

Stephen Zhao, Rob Brekelmans, Alireza Makhzani, and Roger Grosse. Probabilistic inference in language models via twisted sequential monte carlo. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Rui Zheng, Shihuan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, Limao Xiong, Lu Chen, Zhiheng Xi, Nuo Xu, Wenbin Lai, Minghao Zhu, Cheng Chang, Zhangyue Yin, Rongxiang Weng, Wensen Cheng, Haoran Huang, Tianxiang Sun, Hang Yan, Tao Gui, Qi Zhang, Xipeng Qiu, and Xuanjing Huang. Secrets of rlhf in large language models part i: Ppo, 2023. URL <https://arxiv.org/abs/2307.04964>.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. Controlled text generation with natural language instructions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 42602–42613. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/zhou23g.html>.

Banghua Zhu, Michael Jordan, and Jiantao Jiao. Principled reinforcement learning with human feedback from pairwise or k-wise comparisons. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL <https://arxiv.org/abs/1909.08593>.

Table 4: **Breakdown of the average ϕ_{topic} , Topic Prob, and Exp. Min. Topic for 6 topics when steering Llama-3.2 (1B) generations to adhere to each given topic.** Topics are ordered left-to-right according to their reported frequency in Wettig et al. (2025).

Metric	Method	Politics	Finance & Business	Science & Tech	Food & Dining	History	Industrial
ϕ_{topic}	random	90.89	95.79	91.21	89.83	92.13	91.40
	beamsearch	90.94	97.54	86.02	90.18	91.14	93.95
	BoN	97.40	98.98	98.64	94.36	98.30	97.46
	SConE	98.99	99.70	99.42	97.14	99.60	99.56
Topic Prob	random	84.00	92.80	84.80	84.40	84.40	86.80
	beamsearch	83.60	95.60	77.60	86.80	89.20	92.00
	BoN	96.00	98.00	97.20	89.60	97.20	94.40
	SConE	98.40	100.00	99.20	93.60	99.60	99.60
Exp. Min. Topic	random	82.51	92.03	81.55	82.46	82.48	82.41
	beamsearch	88.51	97.08	84.61	87.64	90.36	93.89
	BoN	94.93	97.74	95.58	91.52	96.21	95.13
	SConE	96.42	99.08	95.60	93.37	97.47	98.23

A Experiment Details

SConE. As a trade-off between efficiency and performance, we perform exact inference over the top-10 tokens of the base LM. For each prefix, we run 2 independent, non-blocking Gibbs Sampling chains for 20 iterations, applying a thinning factor of 5. Each chain starts by sampling 25 tokens from the base LM using a combination of nucleus and min-p sampling ($\text{top_p}=0.9$, $\text{min_p}=0.1$) (Holtzman et al., 2020; Minh et al., 2025). A BERT-based model (Warner et al., 2024) is used to efficiently approximate the conditionals \tilde{p}_{cond} .

B Hyperparameters Configurations

In this section, we describe the hyperparameters used for each of the decoding algorithms and baselines used in this work. Except where explicitly mentioned we rely on the HuggingFace’s implementation¹⁰ and the default configurations.

- **Random Search** (random): `do_sample=True`
- **Beam Search** (beamsearch): `do_sample=True`, `num_beams=5` and `temperature=0.3`.
- **Best-of-N** (BoN) (BoN): We implement a custom best-of-n rejection sampling approach (Stienon et al., 2020a), that independently generates $N = 10$ sequences using HuggingFace’s `generate` method, parameterized with `do_sample=True`, `top_p=0.9`, `min_p=0.1`. A verifier ϕ_a is used to choose the final generation, picking the generation out of the N that maximizes the constraint verifier. For the detoxification experiments where the goal is to minimize toxicity as measured by ϕ_{toxicity} we chose the generation that minimizes ϕ_{toxicity} (in practice, we maximize $1 - \phi_{\text{toxicity}}$).

Experiments were run on RTX A6000 (48GB RAM) GPUs using HuggingFace and PyTorch.

C Additional Results

C.1 Controlled Topic Generation

Lastly, we evaluate the methods on their ability to control for the topic of LM generations. We choose 6 diverse topics from the recently taxonomy concerning the web structure (Wettig et al., 2025), including frequent (e.g., *Finance & Business* and *Politics*) and less frequent topics (e.g., *History*, *Industrial*). For each topic, we randomly select 50 different examples from the TopicAnnotations-Llama-3.1-405B-FP8 (Wettig et al., 2025) test set, breaking them into prefixes of 8 to 12 words. Each prefix is used to sample a maximum of 60 tokens.

¹⁰<https://huggingface.co/> (version 4.49.0)

Topic Generation Task. In general, we find that uncontrolled baselines achieve a fairly high average constraint score ($\geq 91\%$), which may be explained by the use of longer prefixes during generation. We find this to be the case for most examples. Nonetheless, the discrepancy between uncontrolled and controlled methods is still visible with the latter achieving 7%-8% higher average constraint scores. Remarkably, we find that `SConE` is not only able to improve upon BoN, achieving an average score of 98.89% but also produces higher quality generations as emphasized by the lower perplexity.

D Efficient Lookahead Generation via Approximate Gibbs Sampling

Our approach requires access to plausible future continuations, or lookahead samples, $\mathbf{y}_{i+1:T}$, given a prefix $\mathbf{y}_{1:i}$. However, we would like to avoid expensive autoregressive sampling, especially since we are happy to trade off sample quality for efficiency. Intuitively, we are only interested in a crude projection of where the current trajectory might lead us, as opposed to a perfectly coherent natural language sentence.

Taking cue from speculative decoding (Leviathan et al., 2023), given a prefix $\mathbf{y}_{1:i}$ we start with a guess for the continuation $\mathbf{y}_{i+1:T}$, either by padding with [MASK] tokens or crudely sampling $p(\mathbf{y}_j \mid \mathbf{y}_{1:i})$ for $j = i + 1$ to T . We can then refine these crude continuations using Gibbs Sampling (Koller & Friedman, 2009), a Markov chain Monte Carlo (MCMC) approach that stochastically samples each token in the sequence, asymptotically converging to the true distribution. Therefore, by setting a *cutoff*, or a maximum number of iterations, we can control how crude of a lookahead sample we desire. Unfortunately, this introduces a multitude of computational challenges. First, the Gibbs sampler assumes efficient access to the full conditionals $p(\mathbf{y}_i \mid \mathbf{y}_{-i}) \forall i$, which requires $O(|V|)$ forward passes of the LM for a single position i , which is untenable given the vocabulary size of modern LMs. Second, in its most basic form, Gibbs sampling requires many iterations through the sentence, computing the conditional and resampling a single token per iteration, which is quite slow.

Algorithm 4 Hogwild! Gibbs Sampling

```

1: Input: ModernBert, prefix  $\mathbf{y}_{1:i}$ , lookahead  $\Delta$ ,  
   block size  $B$ , num workers  $W$ , iterations  $N$   
2: Output:  $\tilde{\mathbf{y}}_{1:T}$  drawn approximately from  $p$   
3:  
4:  $\triangleright$  Randomly initialize continuation  $\mathbf{y}_{i+1:T}$   
5:  $\mathbf{s} \leftarrow \text{InitializeSequence}(\mathbf{y}_{1:i}, \Delta)$   
6:  $\triangleright$  Launch  $W$  workers for  $N/W$  updates  
7: for all workers  $w = 1$  to  $W$  in parallel do  
8:   for  $\text{iter} = 1$  to  $\lceil N/W \rceil$  do  
9:      $\triangleright$  Sample block start  $j$  in continuation  
10:     $j \sim \mathcal{U}(i + 1, T - B + 1)$   
11:     $\text{blk\_idx} \leftarrow [j : j + B - 1]$   
12:     $\triangleright$  Read (potentially stale) state  $\mathbf{s}_{local}$   
13:     $\mathbf{s}_{local} \leftarrow \text{ReadSharedState}(\mathbf{s})$   
14:     $\triangleright$  Get approximate block conditionals  
15:     $p_{blk} \leftarrow \text{ModernBert}(\mathbf{s}_{local}, \text{blk\_idx})$   
16:     $\triangleright$  Sample new tokens for the block  
17:     $\mathbf{y}'_{blk} \leftarrow \text{SampleFromBlockDist}(p_{blk})$   
18:     $\triangleright$  Update shared sequence (Hogwild!)  
19:     $\text{WriteSharedState}(\mathbf{s}, \text{blk\_idx}, \mathbf{y}'_{blk})$   
20:   end for  
21: end for  
22:  $\text{WaitForAllWorkers}()$   
23:  $\tilde{\mathbf{y}}_{1:T} \leftarrow \text{ReadSharedState}(\mathbf{s})$   
24: return  $\tilde{\mathbf{y}}_{1:T}$ 
```

To overcome these challenges and enable efficient generation, we utilize several strategies:

Approximate Conditionals with Masked Language Models (MLMs) In place of analytically computing the conditionals computation, we leverage efficient pretrained MLMs to approximate the conditional probability $p(\mathbf{y}_i \mid \mathbf{y}_{-i})$. These models are inherently designed to predict masked tokens given their bidirectional context, providing a fast approximation of the required conditional distributions without expensive analytical marginalization.

Parallel and Asynchronous Updates (Hogwild! Style) Standard Gibbs sampling updates tokens sequentially. In a bid to accelerate sampling, we employ parallel, potentially asynchronous updates inspired by Hogwild! (Smola & Narayananurthy, 2010; Niu et al., 2011) approaches. Multiple token positions j can be updated simultaneously, possibly using slightly stale context information \mathbf{y}_{-j} . This trades off the unbiasedness of Gibbs sampling (Sa et al.) for substantial gains in wall-clock time that are crucial for inference-time applications.

Blocked Gibbs Sampling Rather than sampling individual tokens one at a time, we can update contiguous blocks of tokens simultaneously. This reduces the number of sampling iterations required for convergence of the chain while allowing us to better leverage the parallel processing capabilities of modern hardware, especially when combined with MLM-based approximate conditionals that excel at processing multiple positions efficiently.

Controlling the Efficiency-Accuracy Trade-off The use of approximate conditionals introduces a natural dial to balance efficiency and sample quality. In very much a Hogwild! fashion, the frequency at which we re-compute or synchronize these approximate conditionals using the latest context influences this trade-off. Less frequent updates lead to faster sampling using potentially more outdated contextual information, while more frequent updates improve fidelity to the target distribution at the cost of increased computation.

By combining Gibbs sampling with these efficiency-focused techniques—approximating conditionals via MLMs, parallelizing updates Hogwild! style, and employing blocked sampling—we can rapidly generate diverse and plausible lookahead samples $\mathbf{y}_{i+1:T}$ suitable for our inference-time algorithm, effectively transforming the computationally demanding task of sampling from the joint distribution into a manageable and efficient procedure.

The pseudocode for the approach elucidated above can be seen in Algorithm 4. Furthermore, an efficient PyTorch implementation will be made available in our GitHub Repository.