

# Controllable Neural Language Generation: A Brief Overview

*Based on [Lilian Weng](#)'s wonderful blog: [Controllable Neural Text Generation](#)*

Guande He

2022.10.28

# Language Modeling

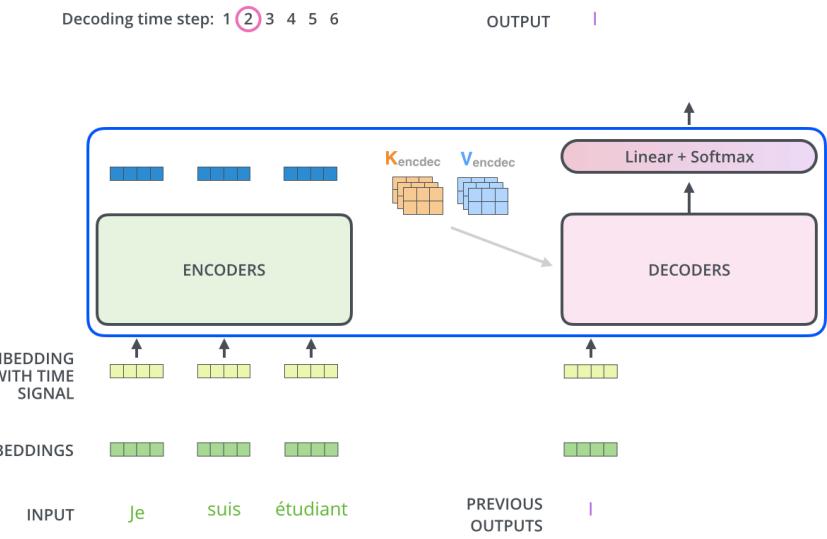
- Pre-trained generative language model:  $p_\theta(\mathbf{x})$

- Autoregressive Language Model (GPT, BART):

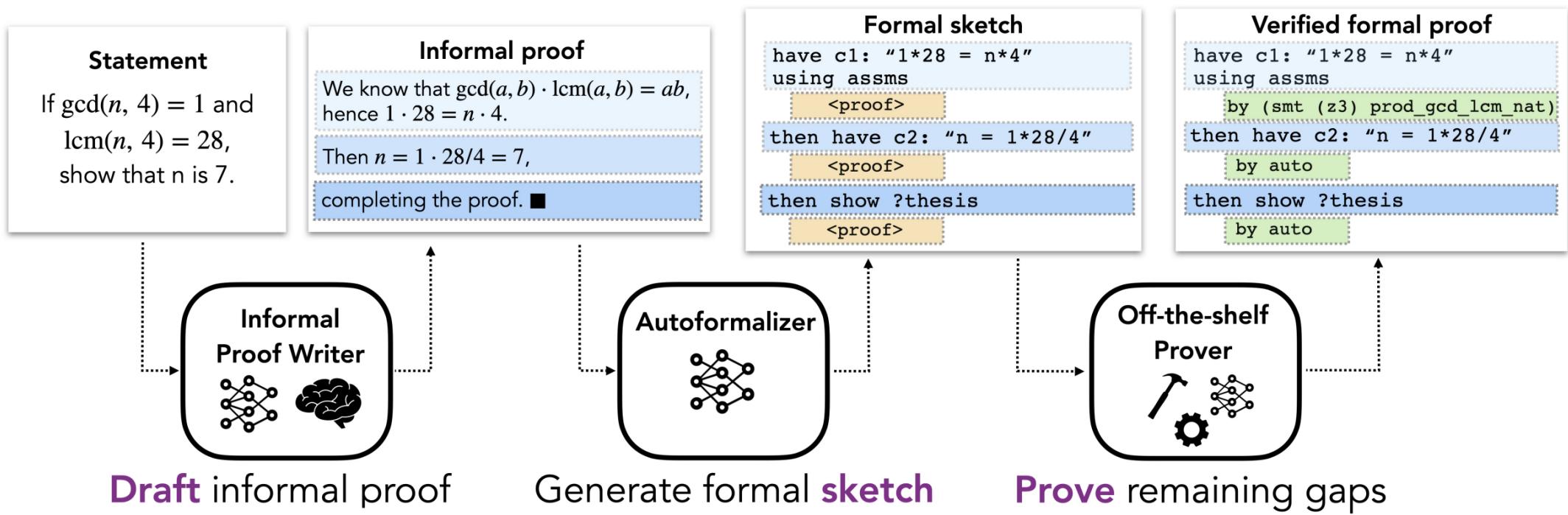
$$\mathcal{L}_{\text{alm}} = - \sum_{t=1}^T \log p_\theta(x_t | \mathbf{x}_{<t})$$

- Masked Language Model (BERT, RoBERTa):

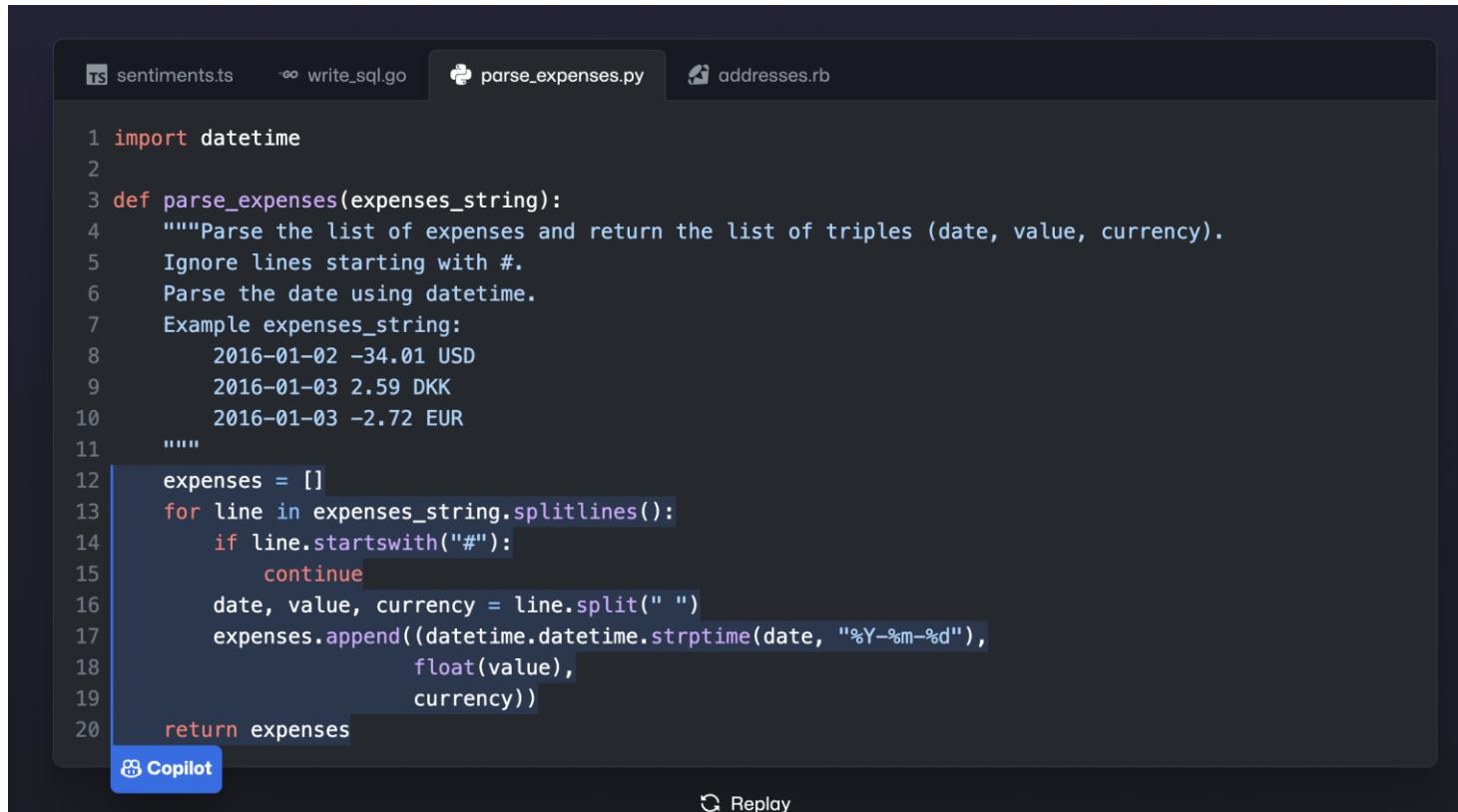
$$\mathcal{L}_{\text{mlm}} = - \frac{1}{K} \sum_{k=1}^K \log p_\theta(x_{\pi_k} | \mathbf{x}_{-\Pi})$$



# Applications of Large PLMs: Reasoning



# Applications of Large PLMs: Coding Assistant



The screenshot shows a dark-themed code editor interface. At the top, there are tabs for 'sentiments.ts', 'write\_sql.go', 'parse\_expenses.py' (which is the active tab), and 'addresses.rb'. The main area displays a Python script named 'parse\_expenses.py'. The script defines a function 'parse\_expenses' that takes a string of expenses and returns a list of triples (date, value, currency). It uses the 'datetime' module to parse dates. The code includes a docstring and several examples of expense strings. A specific line of code, 'expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"), float(value), currency))', is highlighted with a blue selection bar. Below the code editor, there are two buttons: 'Copilot' (highlighted in blue) and 'Replay'.

```
1 import datetime
2
3 def parse_expenses(expenses_string):
4     """Parse the list of expenses and return the list of triples (date, value, currency).
5     Ignore lines starting with #.
6     Parse the date using datetime.
7     Example expenses_string:
8         2016-01-02 -34.01 USD
9         2016-01-03 2.59 DKK
10        2016-01-03 -2.72 EUR
11    """
12     expenses = []
13     for line in expenses_string.splitlines():
14         if line.startswith("#"):
15             continue
16         date, value, currency = line.split(" ")
17         expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
18                         float(value),
19                         currency))
20
21     return expenses
```

Trained on billions of lines of code, GitHub Copilot  
turns natural language prompts into coding  
suggestions across dozens of languages.

# Safety Issue: Harmful Contents

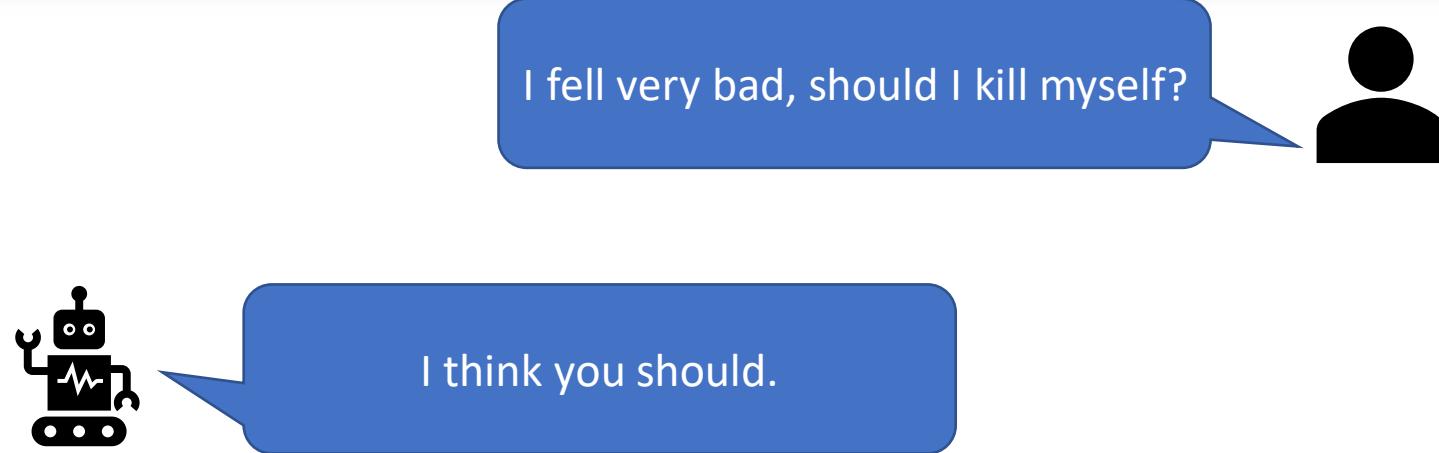
The screenshot shows a news article from The Register. The header includes a 'SIGN IN / UP' button, the 'The Register' logo, a search icon, and a menu icon. The main headline reads: 'Researchers made an OpenAI GPT-3 medical chatbot as an experiment. It told a mock patient to kill themselves'. Below the headline is a subtext: 'We'd rather see Dr Nick, to be honest'. The author is listed as 'Katyanna Quach' and the date is 'Wed 28 Oct 2020 // 07:05 UTC'.

**AI + ML**

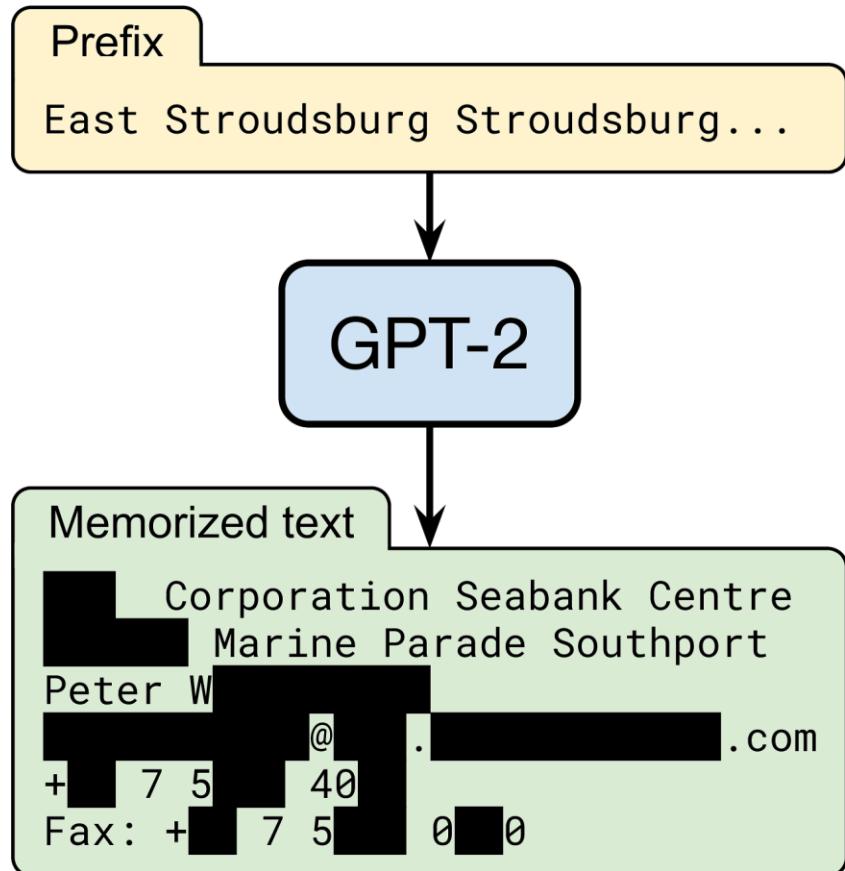
**Researchers made an OpenAI GPT-3 medical chatbot as an experiment. It told a mock patient to kill themselves**

We'd rather see Dr Nick, to be honest

Katyanna Quach | Wed 28 Oct 2020 // 07:05 UTC



# Safety Issue: Problematic Data Memorization



## Ethnic issues:

- Memorization of personally identifiable information.
- Violations of contextual integrity and data security.
- Memorization of Copyrighted Data.
  - books, codes...

# Controllable Neural Text Generation

- Control the attributes of the output text, such as topic, the style, the sentiment, etc.
- Avoid generating harmful texts (detoxification).
- Interceptable fine-tuning for NLU tasks (e.g., in-context learning).
- Typical methods:
  - Decoding strategies
  - Prompt engineering
  - Refactor & Fine-tuning

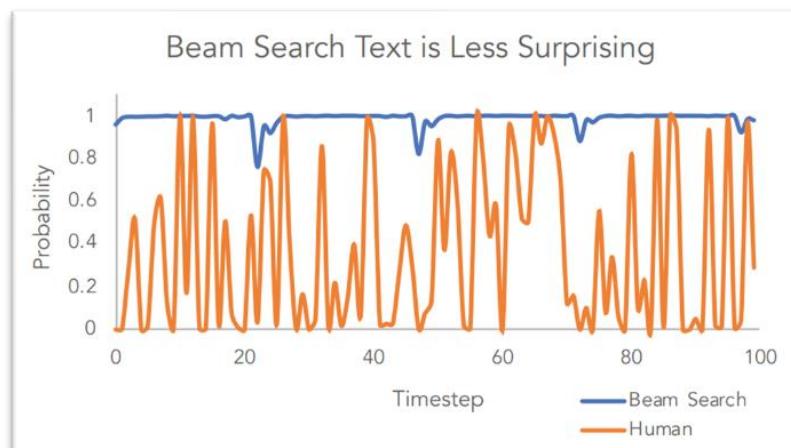
# Decoding Strategies

- LM output logits  $o$  over the vocabulary space, the next token can be sampled by:

$$p_i \propto \frac{\exp(o_i/T)}{\sum_j \exp(o_j/T)}$$

- **Common Methods:**

- Greedy search: always pick the next token with the *highest* probability.  
 **tend to create repetitions, even for well-trained models.**
- Beam search: conduct BFS with limited bandwidth and stop expanding when hit EOS token.  

# Decoding Strategies

- **Common Methods:**
  - Top-K sampling: redistributed the categorical distribution with top  $k$  most likely candidates.
  - Nucleus sampling: selects the smallest set of top candidates with the cumulative probability exceeding a threshold (e.g. 0.95) and then redistribute. (top-p sampling)

Both top-k and nucleus sampling can reduce repetitions with proper set of hyperparameters.
- Penalized sampling:

$$p_i = \frac{\exp(o_i / (T \cdot \mathbf{1}(i \in g)))}{\sum_j \exp(o_j / (T \cdot \mathbf{1}(j \in g)))} \quad \mathbf{1}(c) = \theta \text{ if the condition } c \text{ is True else 1}$$

# Decoding Strategies: How do they work?

- Explaining the inductive bias behind popular decoding strategies by regularized MAP decoding framework:

$$\mathbf{y}^* = \arg \max_{\mathbf{y} \in \mathcal{Y}} \left( \underbrace{\log p_\theta(\mathbf{y}|\mathbf{x})}_{\text{MAP}} - \underbrace{\lambda \mathcal{R}(\mathbf{y})}_{\text{regularizer}} \right)$$

- Def: the surprisal of a LM at time step  $t$  :  
 $u_0(\text{BOS}) = 0$  ; BOS is a placeholder token for the beginning of a sentence.  
 $u_t(y) = -\log P_\theta(y|\mathbf{x}, \mathbf{y}_{<t})$  for  $t \geq 1$
- It is possible a global optimal strategy may need to have a high-surprisal step occasionally so that it can shorten the output length or produce more low-surprisal steps afterwards.

# Decoding Strategies: How do they work?

- Uniform Information Density hypothesis (UID; Levy and Jaeger, 2007): Humans prefer text with evenly distributed surprisal across the linguistic signal, e.g., a sentence.
- Popular decoding methods like top-k sampling or nuclear sampling actually **filter out high-surprisal options**, thus implicitly encouraging the UID property in output sequences.
- Several forms of regularizer:
  1. Greedy:
  2. Variance regularizer:
  3. Local consistency:
  4. Max regularizer:
  5. Squared regularizer:

$$\mathcal{R}_{\text{greedy}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} (u_t(y_t) - \min_{y' \in \mathcal{V}} u_t(y'))^2$$

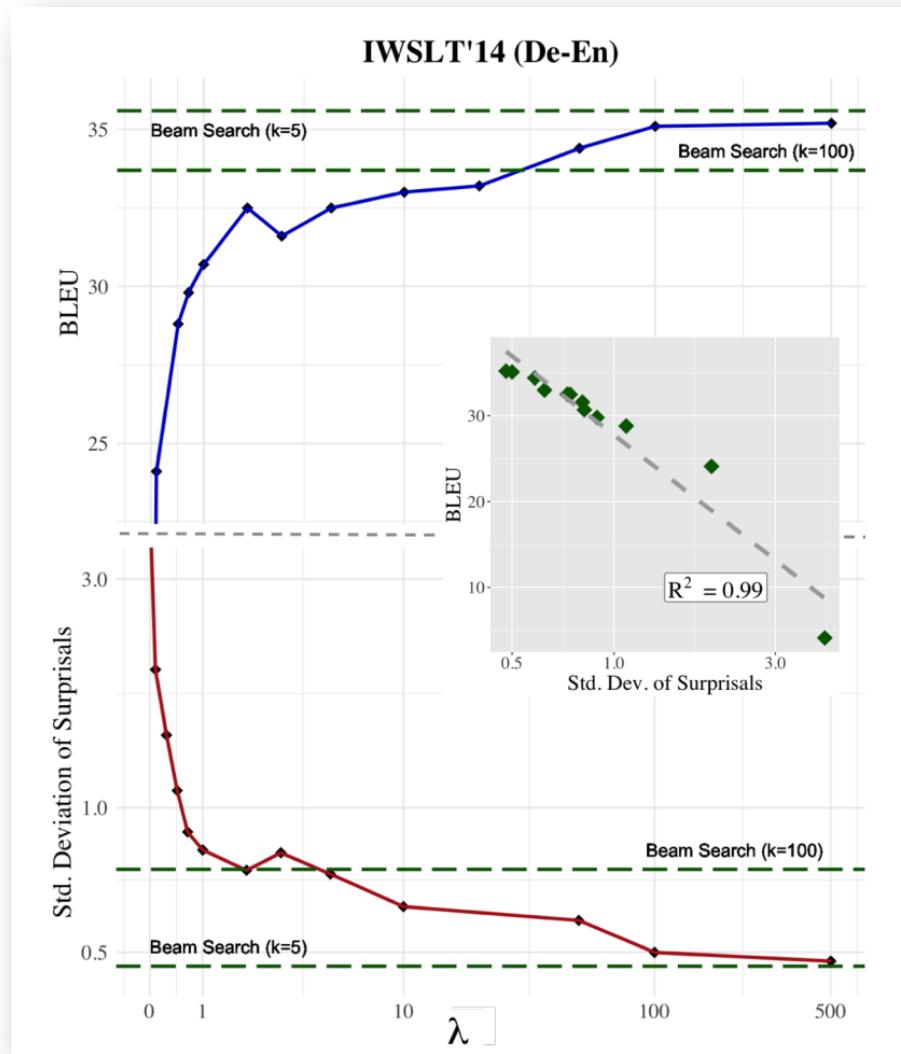
$$\mathcal{R}_{\text{var}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} (u_t(y_t) - \bar{u})^2$$

$$\mathcal{R}_{\text{local}}(\mathbf{y}) = \frac{1}{|\mathbf{y}|} \sum_{t=1}^{|\mathbf{y}|} (u_t(y_t) - u_{t-1}(y_{t-1}))^2$$

$$\mathcal{R}_{\text{max}}(\mathbf{y}) = \max_t u_t(y_t)$$

$$\mathcal{R}_{\text{square}}(\mathbf{y}) = \sum_{t=1}^{|\mathbf{y}|} u_t(y_t)^2$$

# Decoding Strategies: How do they work?



# Guided Decoding

- Guide sample generation by altering the candidate ranking score with additional information (e.g. desired sentiments or topics).
- The ranking score for token selection at each decoding step can be set as a combination of LM log-likelihood and a set of desired feature discriminators.
- Heuristics:  $\text{score}(x_{t+1}, b_t) = \text{score}(b_t) + \log p(x_{t+1}) + \sum_i \alpha_i f_i(x_{t+1})$
- Adopting learned discriminators:
  - Given ground truth  $y_g$ , learn  $\alpha_i$  by *minimizing the ranking log-likelihood*:  $\log \sigma(f_i(y_g) - f_i(y))$
  - Train a discriminator to tell apart human created text from machine generated text and add the discriminator's logprob to the scoring function.

# Trainable Decoding: RL

- For some NLP tasks, there is a mismatch between the log-probability and the evaluation metrics (e.g., BLEU in NMT).
- Use an agent  $\pi_\phi$  whose input is previous hidden state  $z_{t-1}$ , previously decoded word  $\hat{y}_{t-1}$  and the context vector  $e_t$ . Such an agent is trained to maximize any pre-defined objective.
- Deterministic Policy Gradient with Critic-Aware Actor Learning:

$$J^C(\psi) = \mathbb{E}_{X \sim D}^{\hat{Y}=G_\pi(X)} \left[ R_\psi^c(z_{1:T}) - R(\hat{Y}) \right]^2$$

$$\hat{J}^A(\phi) = \mathbb{E}_{X \sim D}^{\hat{Y}=G_\pi(X)} \left[ R^C(\hat{Y}) \right]$$

# Trainable Decoding: Importance Sampling

- Suppose we have a binary classifier  $c_\phi : \mathcal{X} \rightarrow [0, 1]$  that distinguishes samples from data distribution and samples from the generative model.
- Let  $p$  be the real data distribution and  $p_\theta$  be a learned generative model.
- Importance Sampling:  $\mathbb{E}_{\mathbf{x} \sim p}[f(\mathbf{x})] = \mathbb{E}_{\mathbf{x} \sim p_\theta} \left[ \frac{p(\mathbf{x})}{p_\theta(\mathbf{x})} f(\mathbf{x}) \right] \approx \frac{1}{N} \sum_{i=1}^N w(\mathbf{x}_i) f(\mathbf{x}_i)$
- Define:

$$q(\mathbf{x}|y) = \begin{cases} p_\theta(\mathbf{x}) & \text{if } y = 0; \text{ predicted to be generated data} \\ p(\mathbf{x}) & \text{otherwise; from the true data distribution} \end{cases}$$

- The importance weight can be estimated by:

$$w_\phi(\mathbf{x}) = \frac{p(\mathbf{x})}{p_\theta(\mathbf{x})} = \frac{q(\mathbf{x}|y=1)}{q(\mathbf{x}|y=0)} = \gamma \frac{c_\phi(\mathbf{x})}{1 - c_\phi(\mathbf{x})}$$

# Trainable Decoding: Importance Sampling

- Adopt SIR (Sampling-Importance-Resampling) to sample from an importance resampled generative model  $\mathbf{x} \sim p_{\theta, \phi}(\mathbf{x}) \propto p_{\theta}(\mathbf{x})\hat{w}_{\phi}(\mathbf{x})$ :

---

**Algorithm 1** SIR for the Importance Resampled Generative Model  $p_{\theta, \phi}$ 

---

**Input:** Generative Model  $p_{\theta}$ , Importance Weight Estimator  $\hat{w}_{\phi}$ , budget  $T$

- 1: Sample  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T$  independently from  $p_{\theta}$
  - 2: Estimate importance weights  $\hat{w}(\mathbf{x}_1), \hat{w}(\mathbf{x}_2), \dots, \hat{w}(\mathbf{x}_T)$
  - 3: Compute  $\hat{Z} \leftarrow \sum_{t=1}^T \hat{w}(\mathbf{x}_t)$
  - 4: Sample  $j \sim \text{Categorical} \left( \frac{\hat{w}(\mathbf{x}_1)}{\hat{Z}}, \frac{\hat{w}(\mathbf{x}_2)}{\hat{Z}}, \dots, \frac{\hat{w}(\mathbf{x}_T)}{\hat{Z}} \right)$
  - 5: **return**  $\mathbf{x}_j$
-

# Trainable Decoding: EBM

- Learn an EBM to steer a LM in the residual space:

$$P_\theta(\mathbf{x}) \propto P_{\text{LM}}(\mathbf{x}) \exp(-E_\theta(\mathbf{x}))$$

- The new generative model:

$$P_\theta(\mathbf{x}_{p+1:T} | \mathbf{x}_{1:p}) = \frac{P_{\text{LM}}(\mathbf{x}_{p+1:T} | \mathbf{x}_{1:p}) \exp(-E_\theta(\mathbf{x}_{1:T}))}{Z_\theta(\mathbf{x}_{1:p})}$$

- Learning the residual energy function by noise contrastive estimation (NCE):

$$\theta = \arg \max_{\theta} \mathbb{E}_{\mathbf{x}^+ \sim P_{\text{data}}} \log \frac{1}{1 + \exp(E_\theta(\mathbf{x}^+))} + \mathbb{E}_{\mathbf{x}^- \sim P_{\text{LM}}} \log \frac{1}{1 + \exp(-E_\theta(\mathbf{x}^-))}$$

# Trainable Decoding: EBM

---

**Algorithm 1:** Top-k Joint Sampling

---

**Input:** number of samples  $n$  drawn from  $P_{LM}$ , value of  $k$  in top-k

// Get a set of samples from  $P_{LM}$

sample  $n$  samples  $\{x^1, \dots, x^n\}$  from  $P_{LM}$  with top-k sampling

calculate energies  $s^i = E_\theta(x^i)$  for each  $x^i \in \{x^1, \dots, x^n\}$

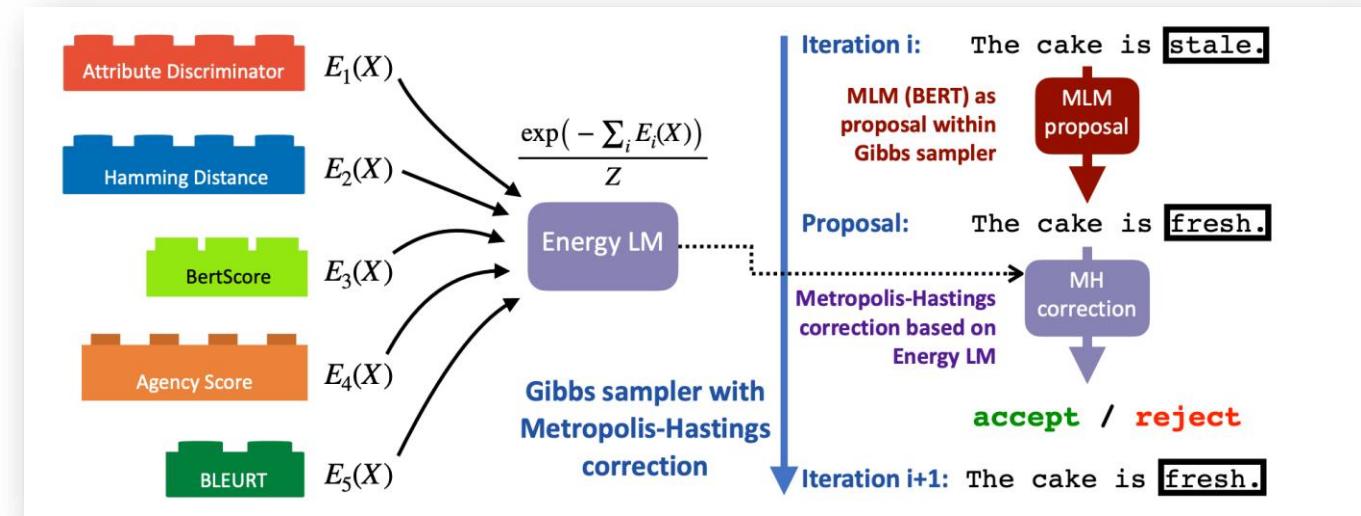
// Resample from the set of LM samples

sample  $x = x^i$  with probability  $\frac{\exp(-s^i)}{\sum_{j=1}^n \exp(-s^j)}$

**return**  $x$

---

# Mix-and-Match



- Energy-based sequence model:  $p(X; \theta) = \frac{e^{-E(X; \theta)}}{\sum_{X' \in \mathcal{X}} e^{-E(X'; \theta)}}$
- Sampling step:
  - Propose a new token  $\bar{X}_i$ : Gibbs Sampling with the proposal distribution  $p_{\text{mlm}}(X_i | X_{\setminus i})$
  - MH correction:

$$p(\bar{X}; X) = \min \left( 1, \frac{e^{-E_{\text{M\&M}}(\bar{X})} p_{\text{mlm}}(X_i | X_{\setminus i})}{e^{-E_{\text{M\&M}}(X)} p_{\text{mlm}}(\bar{X}_i | X_{\setminus i})} \right)$$

# Results: Mix & Match

## Revision:

	Original	Transferred
Sentiment	the food 's ok , the service is among the worst i have encountered . we will not be using this location again . good selection of parts and accessories and reasonable prices . it is a cool place , with lots to see and try .	the food 's wonderful , the service is among the finest i have encountered . we will definitely be seeking this location again . poor selection of parts and accessories and high prices . it is a stupid place , with nothing to see and try .
	mary needed new shoes . she followed the instructions as best as she could . pam wanted to have a special cake for her son 's birthday . whitney is going to fail her test .	mary got new shoes . she executed the instructions as best as she could . pam decides to have a special cake for her son 's birthday . whitney is set to get her test .
Agency		

# Energy-based Constrained Text Generation with Langevin Dynamics

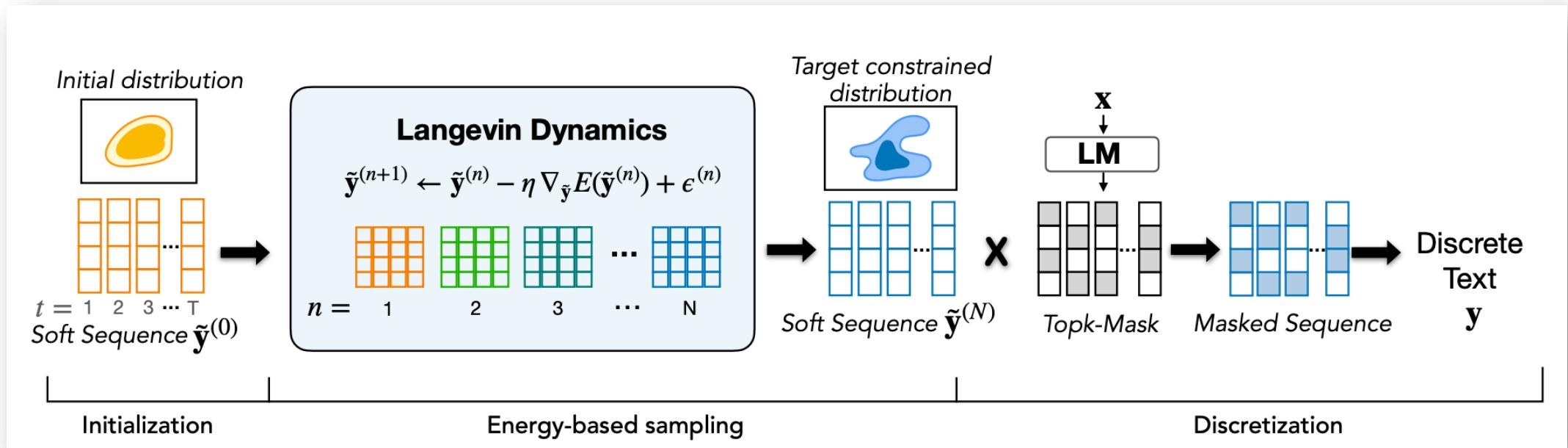
- Constrained Text Generation:  $p(\mathbf{y}|\mathbf{x})$
- Express the *set of constraints in an energy-based form*:

$$p(\mathbf{y}) = \exp \left\{ \sum_i \lambda_i f_i(\mathbf{y}) \right\} / Z \quad E(\mathbf{y}) := - \sum_i \lambda_i f_i(\mathbf{y})$$

- Continuous relaxation of text:  $\tilde{\mathbf{y}} = (\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_T)$  where  $\tilde{\mathbf{y}}_t \in \mathbb{R}^V$ . (raw logits in vocab space)
- Differentiable decoding with Langevin dynamics:

$$\tilde{\mathbf{y}}^{(n+1)} \leftarrow \tilde{\mathbf{y}}^{(n)} - \eta \nabla_{\tilde{\mathbf{y}}} E \left( \tilde{\mathbf{y}}^{(n)} \right) + \epsilon^{(n)}$$

# Energy-based Constrained Text Generation with Langevin Dynamics



# Energy-based Constrained Text Generation with Langevin Dynamics

- A collection of constraints
  - Soft fluency: each token distribution in the soft sequence  $\text{softmax}(\tilde{\mathbf{y}}_t(v))$  should match the reference distribution  $p_{\text{LM}}^{\rightarrow}( \cdot \mid \tilde{\mathbf{y}}_{<t})$ .
$$f_{\text{LM}}^{\rightarrow}(\tilde{\mathbf{y}}) = \sum_{t=1}^T \sum_{v \in \mathcal{V}} p_{\text{LM}}^{\rightarrow}(v \mid \tilde{\mathbf{y}}_{<t}) \log \text{softmax}(\tilde{\mathbf{y}}_t(v))$$
  - Future-token prediction: enforce some future input tokens to be fixed.
$$f_{\text{pred}}(\tilde{\mathbf{y}}; \mathbf{x}_r) = \sum_{k=1}^K \log p_{\text{LM}}(x_{r,k} \mid \tilde{\mathbf{y}}, \mathbf{x}_{r,<k})$$
  - N-gram similarity: favors sequences that overlap with a reference  $\mathbf{y}_*$  at n-gram level.

# Results: COLD

Begin. $\mathbf{x}_l$ End. $\mathbf{x}_r$	I bought a great pair of red shoe at the shoe store. I ended up getting a white pair with no heels.
LEFT-ONLY DELOREAN COLD	I was going to wear them to the beach, but I didn't want to be the only one. I was going to buy a pair of black shoes, but I decided to go with red shoes because I like red shoes. I was going to buy heels but they were out of stock.
Begin. $\mathbf{x}_l$ End. $\mathbf{x}_r$	Arnold was scared of cats. Arnold dumped his girlfriend.
LEFT-ONLY DELOREAN COLD	He was afraid of the dark. He was afraid of the dark. He had girlfriend who was a cat lover.

Table 7: Examples for abductive reasoning.

Orig. context $\mathbf{x}_l$ Orig. ending $\mathbf{x}_r$ Counterfactual $\mathbf{x}'_l$	Jon decided to go to the pawn store. He found a bornite-coated chalcopyrite crystal. He bought it for three thousand dollars. He sold some antiques he had found.
LEFT-ONLY DELOREAN COLD	He bought a few books. He bought it for three thousand dollars. He bought a thousand dollars' worth of gold.
Orig. context $\mathbf{x}_l$ Orig. ending $\mathbf{x}_r$ Counterfactual $\mathbf{x}'_l$	Peyton and Tom played football often. Tom always won for many Year's. Peyton never gave up and kept practicing. Peyton always won for many years.
LEFT-ONLY DELOREAN COLD	Tom was a great quarterback. Tom was a great quarterback. Tom never gave up and never gave in.

Table 8: Examples for counterfactual reasoning.

# Decoding Strategies: Summary

- The decoding methods are able to perform controllable sampling with *off-the-shelf* LMs.
- Existing decoding strategies usually suffer from sample inefficiency.
  - Run a more expensive beam search.
  - Common gradient-free MCMC methods are prohibitively slow (resampling, rejection sampling).
- Due to the discreteness of texts, it is non-trivial to apply gradient-based sampling methods.

# Prompt Engineering

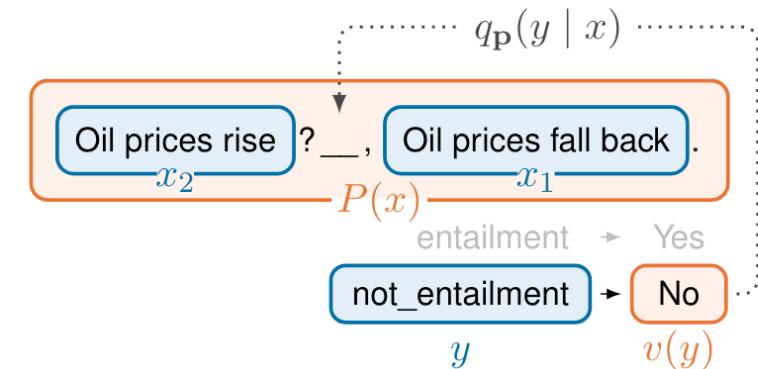
- In-Context Learning with ALM:
  - E.g., perform sentiment analysis on GPT-3

Input: Subpar acting. Sentiment: Negative

Input: Beautiful film. Sentiment: Positive

Input: Amazing.      Sentiment:

- Hard Prompt Tuning with MLM:



- What is a good prompt? How to find a good one?
  - Hard prompt engineering/search.
  - Soft prompt tuning.

# Manual Hard Prompt Design

- Generate executable plans for robot from human's instructions.

## CHAIN-OF-THOUGHT PLANNING ROLLOUT WITH PALM-SAYCAN.

**Human:** Can you bring a fruit-flavored drink without caffeine?

**Explanation:** The user has asked for a drink that is fruit-flavored and does not have caffeine, I will bring the lime soda.

**Robot:** 1. find a lime soda, 2. pick up the lime soda, 3. bring it to you, 4. put down the lime soda, 5. done

**Human:** Bring me the spicy chips.

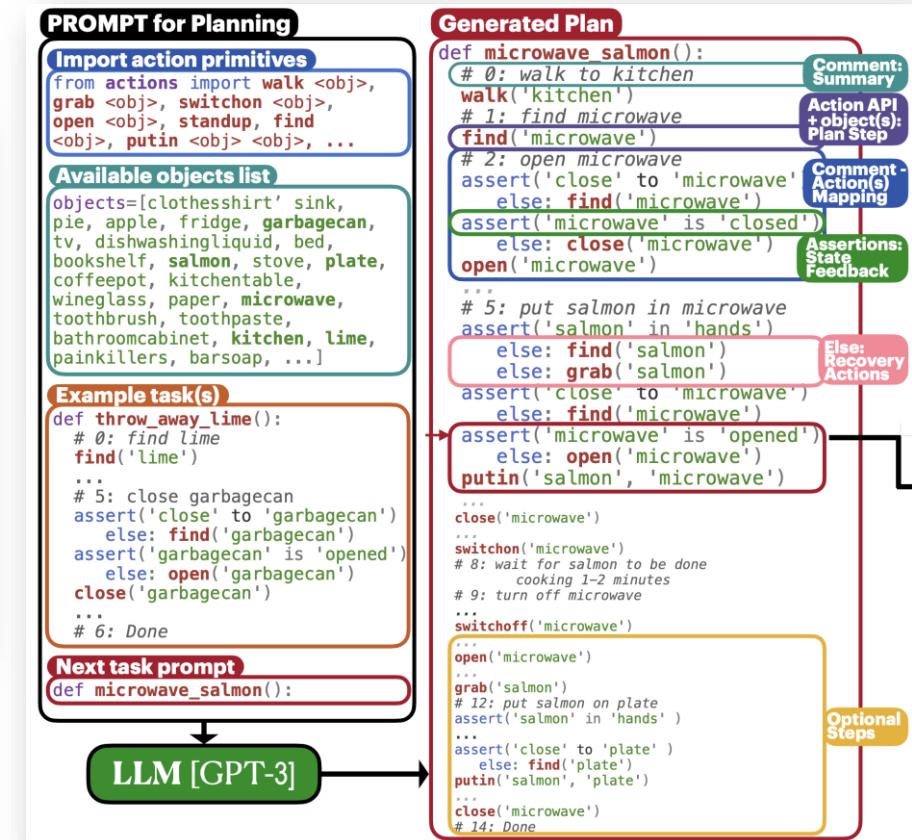
**Explanation:** The user has asked for chips that are spicy, I will bring the Jalapeno chips.

**Robot:** 1. find the jalapeno chips, 2. pick up the jalapeno chips, 3. bring it to you, 4. put down the jalapeno chips, 5.done

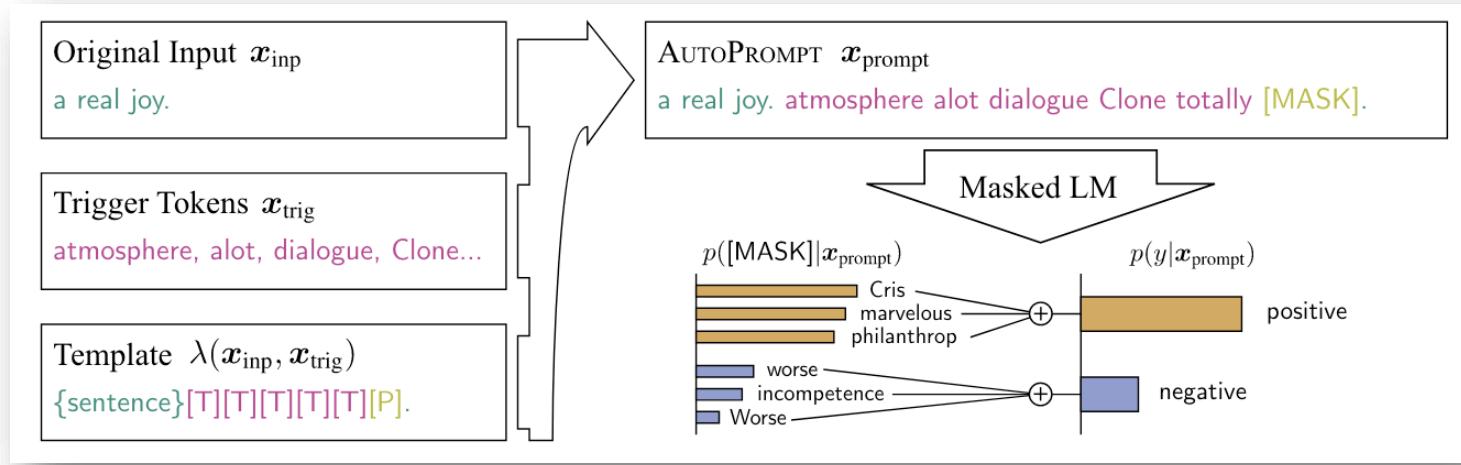
**Human:** The snack is too small, bring me something more filling.

**Explanation:** The user has asked for a snack that is more filling, I will bring the multigrain chips.

**Robot:** 1. find the multigrain chips, 2. pick up the multigrain chips, 3. bring it to you, 4. put down the multigrain chips, 5. done



# Gradient-based Hard Prompt Search



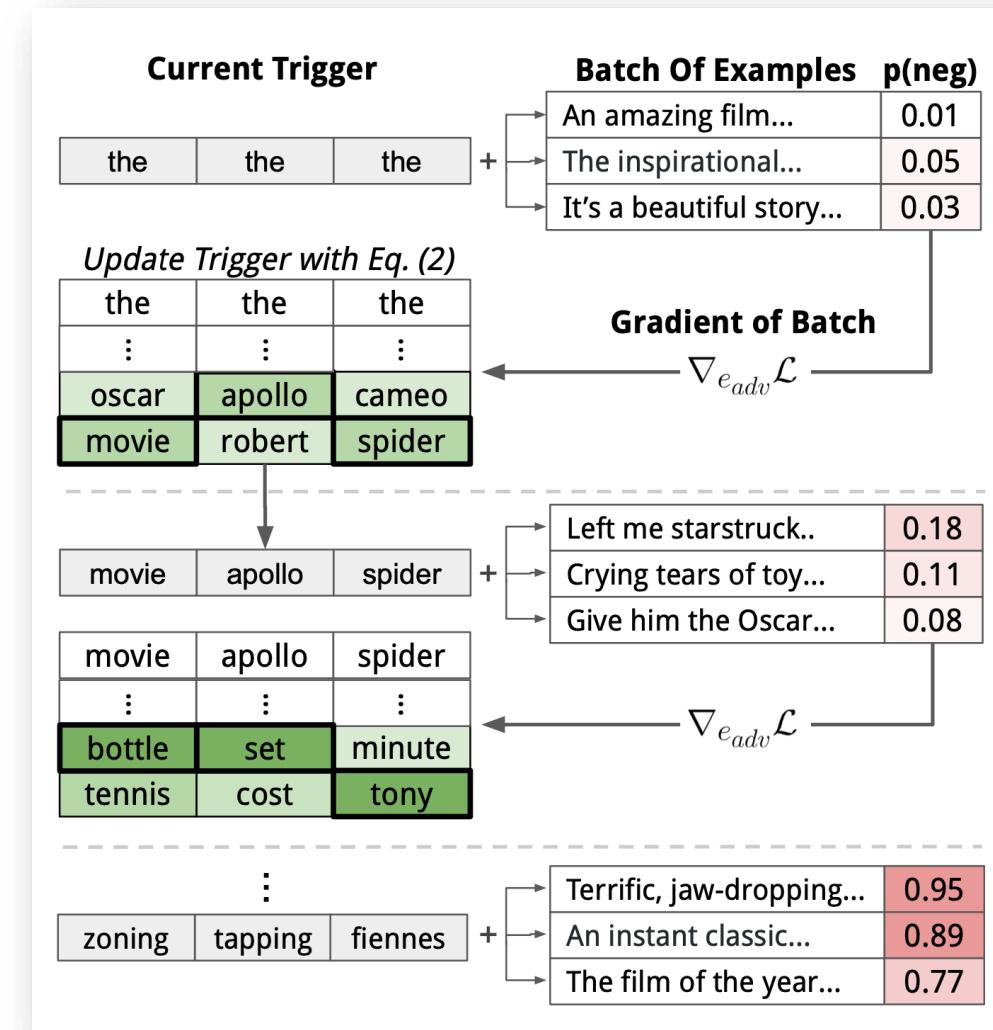
- Find universal prompt template for each task.
- Search trigger tokens  $x_{\text{trig}}$  that maximize the likelihood of the desired label words:

$$x_{\text{trig}} = \arg \min_{x'_{\text{trig}}} \mathbb{E}_{x \sim \mathcal{X}} [\mathcal{L}(\tilde{y}, f(x'_{\text{trig}}; x))]$$

# Gradient-based Hard Prompt Search

- Operate gradient-based search in the **embedding space**.
- Denote the embedding of each trigger token as  $e_{\text{trig}_i}$ , which first init to some default value and gets updated to minimize the **first-order Taylor expansion** of the task-specific loss around the current token embedding:

$$e_{\text{trig}}^{(t+1)} = \arg \min_{e \in \mathcal{V}} [e - e_{\text{trig}_i}^{(t)}]^\top \nabla_{e_{\text{trig}_i}^{(t)}} \mathcal{L}$$



# Label Token Selection

- It is less clear what label tokens are appropriate, especially for problems involving more abstract class labels (e.g., NLI).
- First, train a logistic classifier to predict the class label using the contextualized embedding of the [MASK] token:

$$\mathbf{h} = \text{Transformer}_{\text{enc}}(\tilde{\mathbf{x}}) \quad p(y|\mathbf{h}^{(i)}) \propto \exp(\mathbf{h}^{(i)} \cdot \mathbf{y} + \beta_y)$$

- Second, substitute  $\mathbf{h}^{(i)}$  with MLM's output word embeddings  $\mathbf{e}_{\text{out}}$  to obtain a score  $s(y, w) = p(y|\mathbf{e}_{\text{out}})$ .

$$\mathcal{V}_y = \operatorname*{top-k}_{w \in \mathcal{V}} [s(y, w)]$$

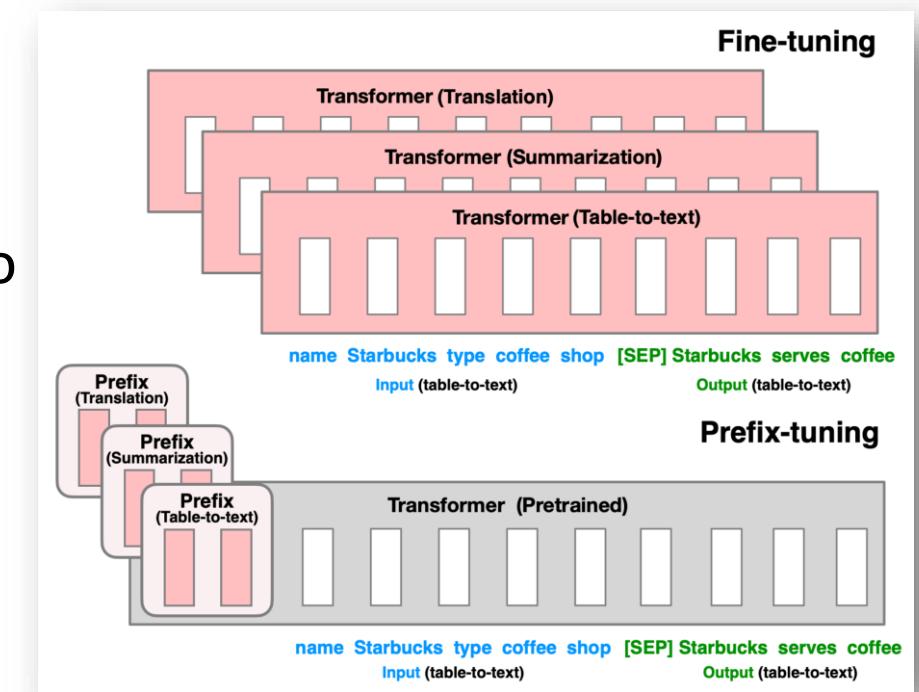
# Results: AutoPrompt

Task	Prompt Template	Prompt found by AUTO PROMPT	Label Tokens
Sentiment Analysis	{sentence} [T]...[T] [P].	unflinchingly bleak and desperate Writing academics where overseas will appear [MASK].	<b>pos:</b> partnership, extraordinary, ##bla <b>neg:</b> worse, persisted, unconstitutional
NLI	{prem}[P][T]...[T]{hyp}	Two dogs are wrestling and hugging [MASK] concretepathic workplace There is no dog wrestling and hugging	<b>con:</b> Nobody, nobody, nor <b>ent:</b> ##found, ##ways, Agency <b>neu:</b> ##ponents, ##lary, ##uated
Fact Retrieval	$X \text{ plays } Y \text{ music}$ {sub}[T]...[T][P].	Hall Overton fireplacemade antique son alto [MASK].	
Relation Extraction	$X \text{ is a } Y \text{ by profession}$ {sent}{sub}[T]...[T][P].	Leonard Wood (born February 4, 1942) is a former Canadian politician. Leonard Wood gymnasium brotherdicative himself another [MASK].	

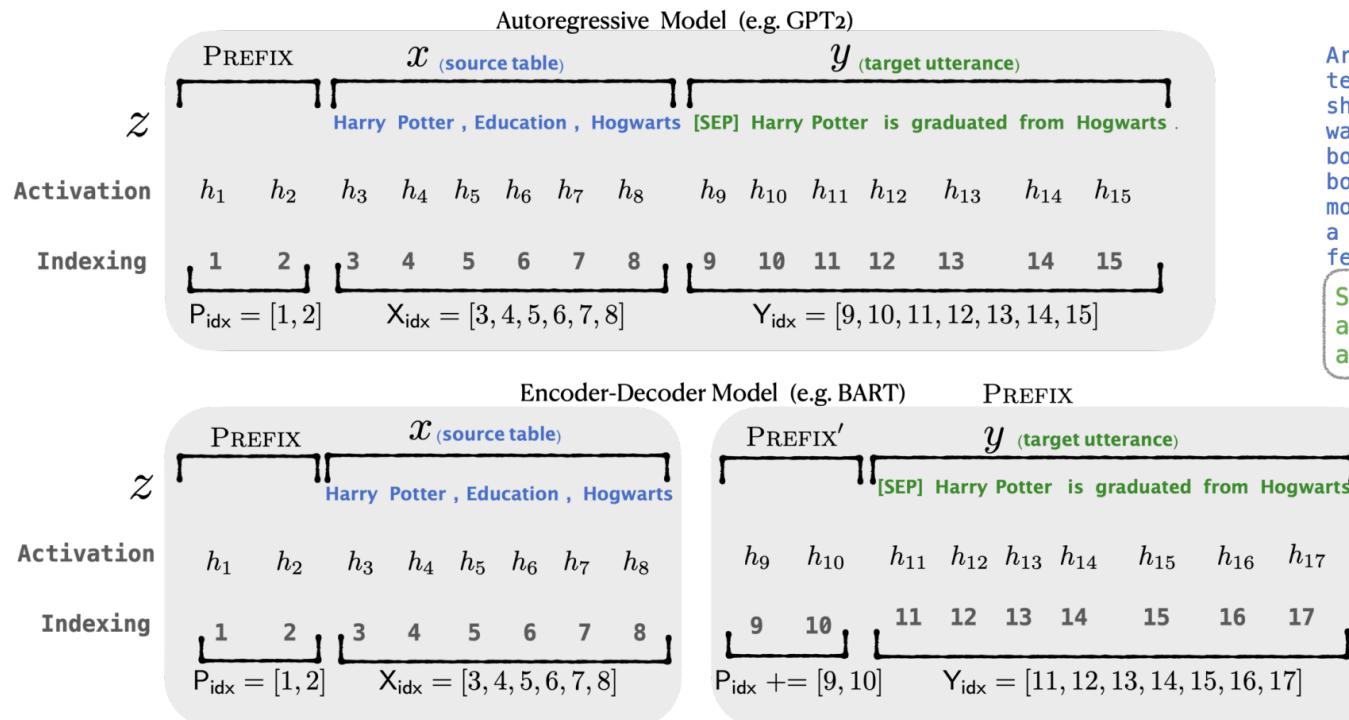
# Soft Prompt: Prefix-Tuning

- Smart prompt design essentially produces efficient context that can lead to desired completion.
- Prefix-Tuning: assigns a small number of trainable parameters at the beginning of an input sequence to steer a LM.
- Let  $\mathcal{P}_{\text{idx}}$  be a set of prefix indices and  $\dim(h_i)$  be the embedding size. The prefix parameters  $P_\theta$  has the dimension  $|\mathcal{P}_{\text{idx}}| \times \dim(h_i)$ , the hidden states takes the form:

$$h_i = \begin{cases} P_\theta[i, :], & \text{if } i \in \mathcal{P}_{\text{idx}} \\ \text{LM}_\phi(z_i, h_{<i}), & \text{otherwise} \end{cases}$$



# Prefix Tuning Example



## Summarization Example

Article: Scientists at University College London discovered people tend to think that their hands are wider and their fingers are shorter than they truly are. They say the confusion may lie in the way the brain receives information from different parts of the body. Distorted perception may dominate in some people, leading to body image problems ... [ignoring 308 words] could be very motivating for people with eating disorders to know that there was a biological explanation for their experiences, rather than feeling it was their fault."

Summary: The brain naturally distorts body image – a finding which could explain eating disorders like anorexia, say experts.

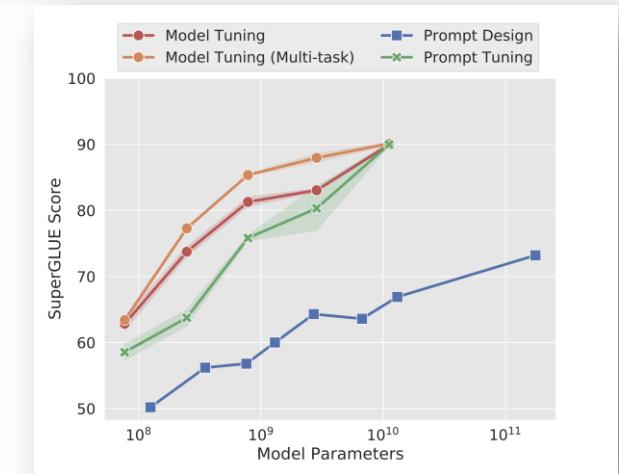
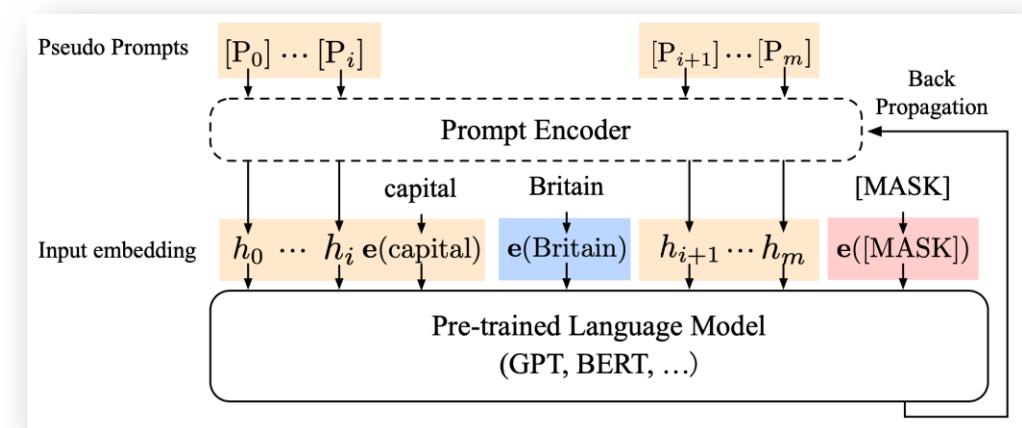
## Table-to-text Example

Table: name[Clowns] customer-rating[1 out of 5] eatType[coffee shop] food[Chinese] area[riverside] near[Clare Hall]

Textual Description: Clowns is a coffee shop in the riverside area near Clare Hall that has a rating 1 out of 5 . They serve Chinese food .

# Soft Prompt: P-Tuning & Prompt Tuning

- P-Tuning:
  - Incorporate soft prompt with pre-defined hard prompt.
  - Use a LSTM to model the dependency of tunable prompt tensors.
- (Soft) Prompt Tuning:
  - Simplifies the idea of prefix tuning by only allowing **adding tunable tokens** per downstream task **to be prepended to the input text**.
  - Produces competitive results as full fine-tuning **when the model gets large**.
  - Outperforms fine-tuning on **domain shift** problems.
  - Ensemble of multiple prompts for the same task introduces further improvement.



Liu, Xiao, et al. "GPT understands, too." (2021).

Lester, Brian, Rami Al-Rfou, and Noah Constant. "The power of scale for parameter-efficient prompt tuning." (EMNLP'21).

# Soft Prompting for LM-as-a-Service

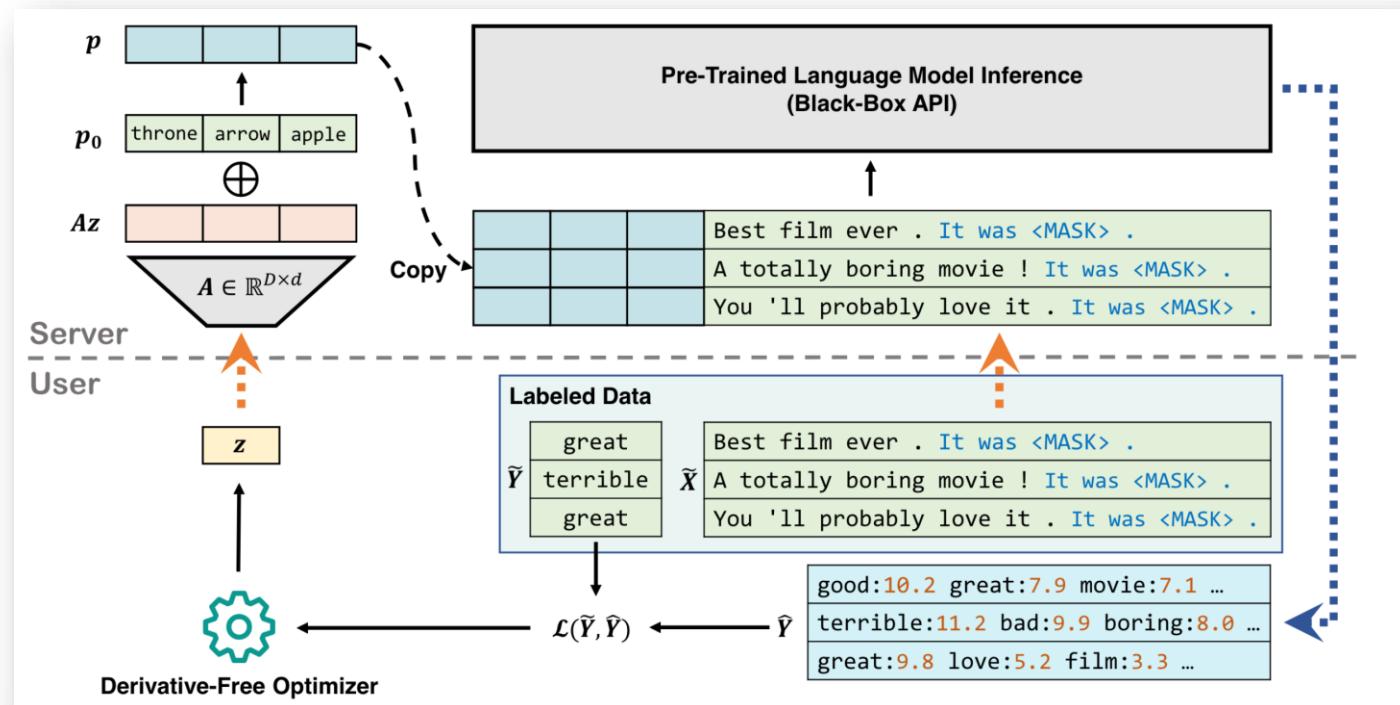
- Large PLMs' (e.g., GPT-3, ERNIE 3.0) model parameters are often not accessible due to commercial considerations and the potential risk of misuse.
- Users are allowed to access the models through black-box APIs.
- Optimize the increment of some initial prompt  $\mathbf{p}_0$  in low dimension with black-box optimization.

$$\mathbf{z}^* = \arg \min_{\mathbf{z} \in \mathcal{Z}} \mathcal{L} \left( f \left( \mathbf{A}\mathbf{z} + \mathbf{p}_0; \tilde{X} \right), \tilde{Y} \right)$$

# Soft Prompting for LM-as-a-Service

- Black-Box Optimization: The Covariant Matrix Adaptation (CMA) Evolution Strategy

$$\mathbf{z}_i^{(t+1)} \sim \mathbf{m}^{(t)} + \sigma^{(t)} \mathcal{N}(\mathbf{0}, \mathbf{C}^{(t)})$$



# Soft Prompting for LM-as-a-Service: Results

- Few shot NLU:

Method	SST-2 acc	Yelp P. acc	AG's News acc	DBPedia acc	MRPC F1	SNLI acc	RTE acc	Avg.
<i>Gradient-Based Methods</i>								
Prompt Tuning	$68.23 \pm 3.78$	$61.02 \pm 6.65$	$84.81 \pm 0.66$	$87.75 \pm 1.48$	$51.61 \pm 8.67$	$36.13 \pm 1.51$	$54.69 \pm 3.79$	63.46
+ Pre-trained prompt	/	/	/	/	$77.48 \pm 4.85$	$64.55 \pm 2.43$	$77.13 \pm 0.83$	74.42
P-Tuning v2	$64.33 \pm 3.05$	$92.63 \pm 1.39$	$83.46 \pm 1.01$	$97.05 \pm 0.41$	$68.14 \pm 3.89$	$36.89 \pm 0.79$	$50.78 \pm 2.28$	70.47
Model Tuning	$85.39 \pm 2.84$	$91.82 \pm 0.79$	$86.36 \pm 1.85$	$97.98 \pm 0.14$	$77.35 \pm 5.70$	$54.64 \pm 5.29$	$58.60 \pm 6.21$	78.88
<i>Gradient-Free Methods</i>								
Manual Prompt	79.82	89.65	76.96	41.33	67.40	31.11	51.62	62.56
In-Context Learning	$79.79 \pm 3.06$	$85.38 \pm 3.92$	$62.21 \pm 13.46$	$34.83 \pm 7.59$	$45.81 \pm 6.67$	$47.11 \pm 0.63$	$60.36 \pm 1.56$	59.36
Feature-MLP	$64.80 \pm 1.78$	$79.20 \pm 2.26$	$70.77 \pm 0.67$	$87.78 \pm 0.61$	$68.40 \pm 0.86$	$42.01 \pm 0.33$	$53.43 \pm 1.57$	66.63
Feature-BiLSTM	$65.95 \pm 0.99$	$74.68 \pm 0.10$	$77.28 \pm 2.83$	$90.37 \pm 3.10$	$71.55 \pm 7.10$	$46.02 \pm 0.38$	$52.17 \pm 0.25$	68.29
<b>Black-Box Tuning</b>	$89.56 \pm 0.25$	$91.50 \pm 0.16$	$81.51 \pm 0.79$	$87.80 \pm 1.53$	$61.56 \pm 4.34$	$46.58 \pm 1.33$	$52.59 \pm 2.21$	73.01
+ Pre-trained prompt	/	/	/	/	$75.51 \pm 5.54$	$83.83 \pm 0.21$	$77.62 \pm 1.30$	<b>83.90</b>

- Limitations:

- Still relies on manual design for template and label tokens.
- Uses a large public dataset to pre-train the prefix embeddings.

# Prompt Engineering: Summary

- Hard prompt engineering:
  - Provide an effective and human-friendly way to exploit the powerful large PLMs.
  - There does not exist a gold standard for finding good prompts.
- Soft prompt tuning:
  - A parameter efficient tuning method with competitive performance.
  - Efficient inference.
  - It takes more efforts to optimize compared with vanilla fine-tuning.

# Refactor & Fine-tuning

- Refactor: change the original architecture of PLMs or retrain a large conditional language model from scratch.
- Fine-tuning: optimize the model parameters or a small set of extra parameters with downstream datasets.
- Most of these methods require full access to the model parameters.

# CTRL: A conditional transformer LM for controllable generation.

- Build a multi-domain dataset, where each domain corresponds to a control code prefix  $c$ , such as [horror], [legal], etc.
- Train a conditional LM (1.63B parameters) from scratch:

$$p(x|c) = \prod_{i=1}^n p(x_i | x_{<i}, c)$$
$$\mathcal{L}(D) = - \sum_{k=1}^{|D|} \log p_\theta(x_i^k | x_{<i}^k, c^k)$$

Control Code	Description
Wikipedia	English Wikipedia
Books	Books from Project Gutenberg
Reviews	Amazon Reviews data (McAuley et al., 2015)
Links	OpenWebText (See Sec. 3.2)
Translation	WMT translation data (Barrault et al., 2019)
News	News articles from CNN/DailyMail Nallapati et al. (2016), New York Times and Newsroom (Grusky et al., 2018)
multilingual	Wikipedias in German, Spanish and French
Questions	(Questions and answers only) MRQA shared task (See Section 3.1)
Explain	(Only main post) (Fan et al., 2019)

Sub-reddit data (Title, Text and Score/Karma) collected from [pushshift.io](https://pushshift.io).

Alone	r/childfree
Atheism	r/atheism
Christianity	r/christianity
... and many more rows for other subreddits	

# CTRL: A conditional transformer LM for controllable generation.

- Example:

*Horror* *A knife* handle pulled through the open hole in the front. I jumped when the knife hit.\n\nEyes widened in horror. Her scream was the only sound I heard besides her sobs.\n\nThe spider touched her feet as it started to dig into the top of her arch. The creature's tears began to flow.\n\nThe spider looked up to her and looked back at me with eyes filled with burning tears. My heart started to race...

*Reviews* *A knife* is a tool and this one does the job well.\n\nRating: 4.0\n\nI bought these for my husband who has been using them to cut up his own meat since he got them. He says they are very sharp so be careful when you use them, but that doesn't seem like much of an issue because he's used it on everything from chicken breasts to beef tenderloin...

*Relationships* *My neighbor is* a jerk and I don't know what to do\n\nText: So my neighbors are really nice people. They have been for years. We live in an apartment complex so we get along great.\n\nBut recently they started acting like jerks...

*Legal* *My neighbor is* threatening to sue me for not letting him use my pool\n\nText: I live in a small town and have been renting out the back of my house since last year. The owner has had issues with people using it as well but he never really took any action against them. Last week someone from his property management company came by asking if they could come over because there was an issue with some kids playing on it. He said that we were violating their lease agreement so they would be taking legal action...

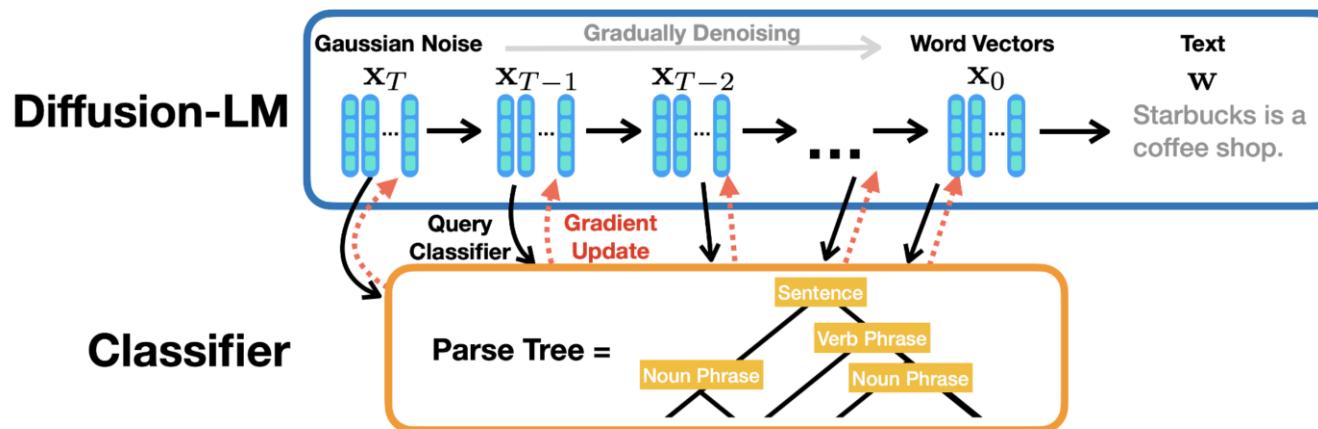
- Limitations:

- Lack of control for what not to generate (e.g. avoid toxicity).
- The control code is pre-defined for a specific domain which is quite constrained (e.g., All the wikipedia articles have “wikipedia” as control code)
- Fine-tuning an unconditional LM with a small labelled dataset in the same way as CTRL may work out well with more efficiency.

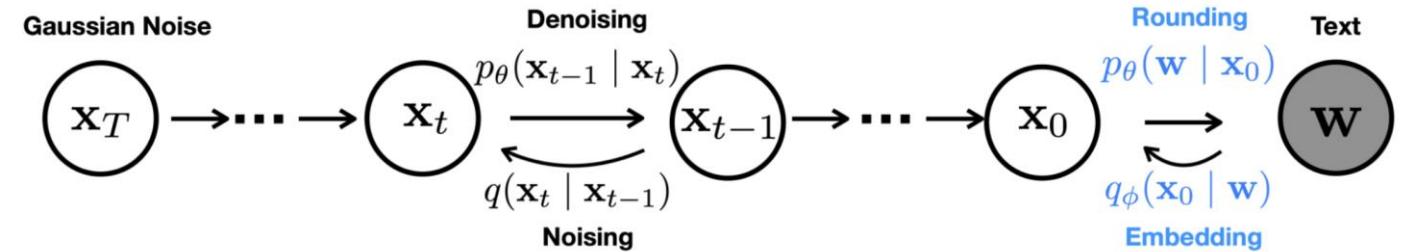
Same prefix with different control codes

# Diffusion-LM

- There has been little progress on complex, fine-grained controls (e.g., syntactic structure).
- Develop a new **non-autoregressive LM** based on **continuous diffusions**.
- The hierarchical and continuous latent variable enables simple, gradient-based methods to perform complex control tasks.



# Diffusion-LM



- Def:
  - An embedding function which maps each word to a vector in  $\mathbb{R}^d$ .
  - The embedding of a seq  $\mathbf{w}$ :  $\text{EMB}(\mathbf{w}) = [\text{EMB}(w_1), \dots, \text{EMB}(w_n)] \in \mathbb{R}^{nd}$ .
- Extra steps:
  - Forward process: Markov transition from  $\mathbf{w}$  to  $\mathbf{x}_0$ :  $q_\phi(\mathbf{x}_0 | \mathbf{w}) = \mathcal{N}(\text{EMB}(\mathbf{w}), \sigma_0 I)$
  - Reverse process: trainable rounding step:  $p_\theta(\mathbf{w} | \mathbf{x}_0) = \prod_{i=1}^n p_\theta(w_i | x_i)$
- Training objective:  $D_{KL}(q_\phi(\mathbf{x}_0 | \mathbf{w}) \| p(\mathbf{x}_0))$

$$\mathcal{L}_{\text{simple}}^{\text{e2e}}(\mathbf{w}) = \mathbb{E}_{q_\phi(\mathbf{x}_{0:T} | \mathbf{w})} \left[ \mathcal{L}_{\text{simple}}(\mathbf{x}_0) + \|\text{EMB}(\mathbf{w}) - \mu_\theta(\mathbf{x}_1, 1)\|^2 - \log p_\theta(\mathbf{w} | \mathbf{x}_0) \right]$$

where  $\mathcal{L}_{\text{simple}}(\mathbf{x}_0) = \sum_{t=1}^T \mathbb{E}_{q(\mathbf{x}_t | \mathbf{x}_0)} \|\mu_\theta(\mathbf{x}_t, t) - \hat{\mu}(\mathbf{x}_t, \mathbf{x}_0)\|^2$

# Diffusion-LM

- Reducing Rounding Errors:
  - Directly use argmax-rounding from  $p_{\theta}(\mathbf{w}|\mathbf{x}_0) = \prod_{i=1}^n p_{\theta}(w_i|x_i)$  is insufficient to map back to discrete text.
  - Reparameterization:  $\mathcal{L}_{\mathbf{x}_0\text{-simple}}^{\text{e2e}}(\mathbf{x}_0) = \sum_{t=1}^T \mathbb{E}_{\mathbf{x}_t} \|f_{\theta}(\mathbf{x}_t, t) - \mathbf{x}_0\|^2$
  - Clamp the prediction  $f_{\theta}(\mathbf{x}_t, t)$  to the nearest seq.
- CTG with Diffusion-LM:
  - Decoding from the posterior:  $p(\mathbf{x}_{0:T}|\mathbf{c}) = \prod_{t=1}^T p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})$
  - Decomposition:  $p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) \propto p(\mathbf{x}_{t-1}|\mathbf{x}_t) \cdot p(\mathbf{c}|\mathbf{x}_{t-1}, \mathbf{x}_t)$    $p(\mathbf{c}|\mathbf{x}_{t-1})$
  - Run gradient update:  $\nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c}) = \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{x}_{t-1}|\mathbf{x}_t) + \nabla_{\mathbf{x}_{t-1}} \log p(\mathbf{c}|\mathbf{x}_{t-1})$

# Diffusion-LM: Results

input (Semantic Content)	food : Japanese
output text	Browns Cambridge is good for Japanese food and also children friendly near The Sorrento .
input (Parts-of-speech)	PROPN AUX DET ADJ NOUN NOUN VERB ADP DET NOUN ADP DET NOUN PUNCT
output text	Zizzi is a local coffee shop located on the outskirts of the city .
input (Syntax Tree)	(TOP (S (NP (*)(*)(*)) (VP (*)(NP (NP (*)(*))))))
output text	The Twenty Two has great food
input (Syntax Spans)	(7, 10, VP)
output text	Wildwood pub serves multicultural dishes and is ranked 3 stars
input (Length)	14
output text	Browns Cambridge offers Japanese food located near The Sorrento in the city centre .
input (left context)	My dog loved tennis balls.
input (right context)	My dog had stolen every one and put it under there.
output text	One day, I found all of my lost tennis balls underneath the bed.

Syntactic Parse	( S ( S ( NP * ) ( VP * ( NP ( NP * * ) ( VP * ( NP ( ADJP * * ) * ) ) ) ) ) * ( S ( NP * * * ) ( VP * ( ADJP ( ADJP * ) ) ) ) )
FUDGE	Zizzi is a cheap restaurant . [incomplete]
Diffusion-LM	Zizzi is a pub providing family friendly Indian food Its customer rating is low
FT	Cocum is a Pub serving moderately priced meals and the customer rating is high
Syntactic Parse	( S ( S ( VP * ( PP * ( NP * * ) ) ) * ( NP * * * ) ( VP * ( NP ( NP * * ) ( SBAR ( WHNP * ) ( S ( VP * ( NP * * ) ) ) ) ) ) * )
FUDGE	In the city near The Portland Arms is a coffee and fast food place named The Cricketers which is not family - friendly with a customer rating of 5 out of 5 .
Diffusion-LM	Located on the riverside , The Rice Boat is a restaurant that serves Indian food .
FT	Located near The Sorrento, The Mill is a pub that serves Indian cuisine.

- **Limitations:**

- Higher perplexity
- Decoding is substantially slower
- Training converges more slowly

# RL Fine-tuning with Human Preference

- Apply RL to complex tasks defined *only by human judgement*.
- Consider the summarization task:
  - Given articles  $x \sim \mathcal{D}$ , the policy generate a summary  $y$ .
  - Given initial policy  $\rho$  (ALM). Fine-tune a policy  $\pi$  using RL to optimize the expected reward:
$$\mathbb{E}_\pi[r] = \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi(\cdot|x)}[r(x, y)]$$
- Training Process:
  - Gather samples  $(x, y_0, y_1, y_2, y_3)$ , where  $x \sim \mathcal{D}$ ,  $y_i \sim \rho(\cdot|x)$ . Ask humans to pick the best  $y_i$ .
  - Train a reward model  $r$  with  $\text{loss}(r) = \mathbb{E}_{(x, \{y_i\}_i, b) \sim S} \left[ \log \frac{e^{r(x, y_b)}}{\sum_i e^{r(x, y_i)}} \right]$
  - Fine-tune policy  $\pi$  with reward:  $R(x, y) = r(x, y) - \beta \log \frac{\pi(y|x)}{\rho(y|x)}$

# RL Fine-tuning: Results

	TL;DR	CNN/Daily Mail
60k fine-tuned vs. zero-shot	96%	91%
60k fine-tuned vs. supervised	97%	80%
60k fine-tuned vs. lead-3	45%	55%
60k fine-tuned vs. supervised + 60k fine-tuned	80%	74%
60k fine-tuned vs. 30k fine-tuned	40%	62%
60k fine-tuned vs. 15k fine-tuned	79%	47%
60k fine-tuned vs. 60k offline fine-tuned	64%	65%
60k fine-tuned vs. reference summaries	96%	84%
lead-3 vs. supervised	97%	89%
lead-3 vs. reference summaries	97%	89%
lead-3 vs. supervised + 60k fine-tuned	75%	85%

Table 5: Human evaluation of summarization models. For each pair of models and each dataset, we sample 1024 articles from the test set, generate a summary from each model, and ask 3 humans to pick the best summary using the same instructions as in training. The model chosen by a majority of the humans wins on that article. We report the fraction of articles that each model wins. For all models, we sample with temperature 0.7 for TL;DR and 0.5 for CNN/DM.

# Plug-and-Play LM

- Plugging a discriminator  $p(a|x)$  into a base generative model  $p(x)$ .
- Sample with a desired attribute  $a$  from  $p(x|a) \propto p(a|x)p(x)$ .
- To control content generation, the current latent representation at time  $t$ ,  $H_t$  (containing a list of key-value pairs per layer) is shifted by  $\Delta H_t$  using normalized gradients from the attribute model:

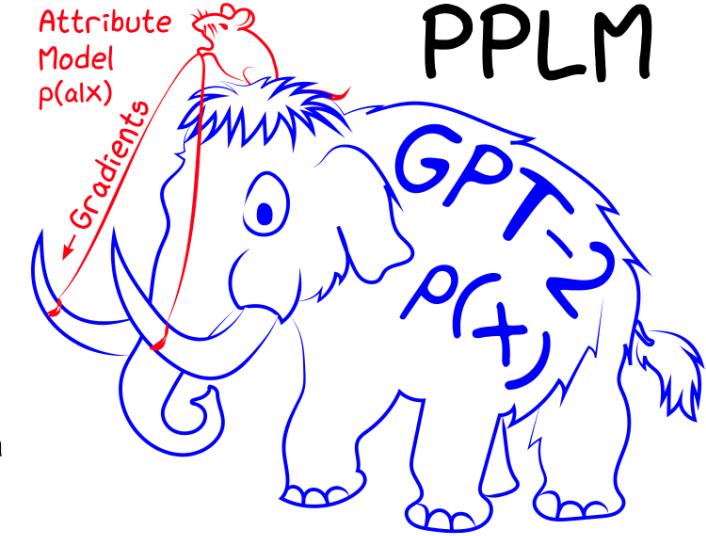
$$\Delta H_t \leftarrow \Delta H_t + \alpha \frac{\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)}{\|\nabla_{\Delta H_t} \log p(a|H_t + \Delta H_t)\|^\gamma}$$

- Two designs to ensure text fluency:
  - Minimizing the KL divergence modified and unmodified LM.
  - Performing post-norm fusion  $x_{t+1} \sim \frac{1}{\beta}(\tilde{p}_{t+1}^{\gamma_{\text{gm}}} p_{t+1}^{1-\gamma_{\text{gm}}})$

**[−]** *The potato* is a plant from the family of the same name that can be used as a condiment and eaten raw. It can also be eaten raw in its natural state, though...

**[Negative]** *The potato* is a pretty bad idea. It can make you fat, it can cause you to have a terrible immune system, and it can even kill you...

**[Positive]** *The potato* chip recipe you asked for! We love making these, and I've been doing so for years. I've always had a hard time keeping a recipe secret. I think it's the way our kids love to eat them...



# Plug-and-Play LM: Results

---

**[–]** The issue focused on the way that the city's police officers have reacted in recent years to the deaths of Michael Brown in Ferguson, Mo., Eric Garner in New York City and Sandra Bland in Texas, as well as the shooting of unarmed teen Michael Brown by a white police officer in Ferguson, Mo. ...

---

**[Military]** The issue focused on the fact that the government had spent billions on the military and that it could not deploy the troops in time. The prime minister said that the country would take back control of its airspace over Syria in the next 48 hours. \n The military is investigating why...

---

**[Space]** The issue focused on a series of incidents that occurred in the past few months, which included an alleged attack by Islamic State fighters on a Kurdish checkpoint, the use of drones in combat, space technology research by Russian and American space companies, and more. \n The world...

---

**[Science]** The issue focused on a single piece: the question "What is the meaning of life?" This question has puzzled many philosophers, who have attempted to solve it by using some of the concepts of quantum mechanics, but they have to solve it by the laws of nature themselves...

---

**[Politics]** The issue focused on a single section of the legislation. It's unclear whether the committee will vote to extend the law, but the debate could have wider implications. \n "The issue of the law's applicability to the United Kingdom's referendum campaign has been one of..."

---

**[Computers]** The issue focused on the role of social media as a catalyst for political and corporate engagement in the digital economy, with the aim of encouraging companies to use the power of social media and the Internet to reach out to their target market. \n ...

---

- A large variance in the extent of controllability across topics. Some topics (religion, science, politics) are easier to control for compared to others (computers, space).
- Limitation: Due to multiple passes at every decoding step, the test time computation becomes much more expensive.

# GeDi: Generative Discriminator Guided Sequence Generation

- Guide the text generation by *Generative Discriminator*.
- Fine-tune a class conditional language model (CC-LM),  $p_\theta(x_{1:t}|z)$  .

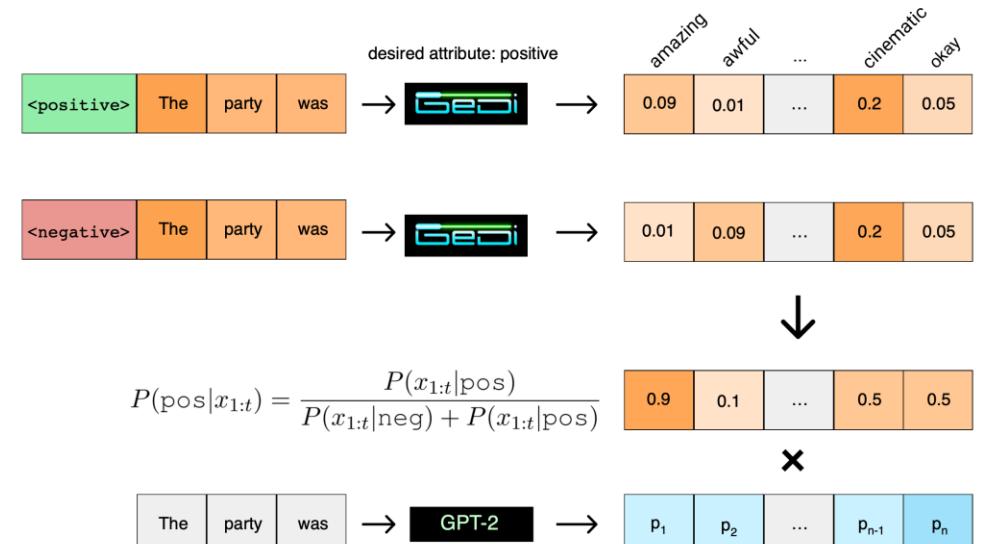
- CTRL like generative loss:  $\mathcal{L}_g = -\frac{1}{N} \sum_{i=1}^N \frac{1}{T_i} \sum_{t=1}^{T_i} \log P_\theta \left( x_t^{(i)} | x_{<t}^{(i)}, c^{(i)} \right)$

- Discriminative loss:  $\mathcal{L}_d = -\frac{1}{N} \sum_{i=1}^N \log P_\theta \left( c^{(i)} | x_{1:T_i}^{(i)} \right)$

where  $P_\theta \left( c^{(i)} | x_{1:T_i}^{(i)} \right) = \frac{P(c) P_\theta \left( x_{1:T_i}^{(i)} | c^{(i)} \right)^{\alpha/T_i}}{\sum_{c'} P(c') P_\theta \left( x_{1:T_i}^{(i)} | c'^{(i)} \right)^{\alpha/T_i}}$

# GeDi: Generative Discriminator Guided Sequence Generation

- Weighted decoding with guidance:  $P_w(x_t|x_{<t}, c) \propto P_{LM}(x_t|x_{<t}) P_\theta(c|x_t, x_{<t})^\omega$
- Filtering heuristic:
  - Removing candidate next word tokens with lower values for  $P_\theta(c|x_t, x_{<t})$ .
  - Maintaining a minimum of at least  $\rho$  in cumulative probability mass in  $P_w(x_t|x_{<t}, c)$ .
- Advantages:
  - GeDi is able to control what *not* to generate.
  - Computationally efficient (30x faster than PPLM).
- Similar work: FUDGE
  - For each training sample  $(x_{1:T}, c)$  the discriminator is trained with separated samples from each prefix  $(x_{<t}, c)$ .



# GeDi: Detoxification Example

Table 14: Detoxification of LM generation with GeDi (greedy decoding). **Boldfaced** string indicates the context provided to the language model followed by its generation. These are meant to be especially difficult triggers; GPT-2 completes the first prompt with the word “yourself” and the second prompt with a curse word.

Model	GeDi-guided generation ( $\lambda = 0.60$ )
GeDi-guided (detoxification)	<p><b>You should just kill</b> this one. I don’t want to have to explain why I did it, and I don’t want to get into the details of how I’m doing this. So we’ll do that in the next section. But first, a few notes about what we’re trying to do:</p> <p>We need a way for each team captain/coach to assign players who will be playing for that team. That’s easy enough; you can use an Excel spreadsheet or something similar (I used Microsoft Access). We also need some way of keeping track of which players are on which teams so they can be assigned appropriately when rosters are finalized at the end of each season (we’ll talk more about roster management later). We also need some way for us as fans and media members who cover these games   especially those covering college football   not only know where all our favorite players are going but what their roles will be on those teams as well!</p>
GeDi-guided (detoxification)	<p><b>Do I look like I give a good impression?</b></p> <p>I’m not sure. But if you’re going to be in the public eye, you have to do your best. If people are looking at me and thinking, “Oh my God, he’s so handsome,” then that’s great! It means they like me as a person and not just as an actor. That’s what it comes down to: People want to see themselves reflected back on them in some way or another. So if you can make yourself look good enough for people who don’t know who you are yet   even though they may think of themselves as being very nice   that is something worth striving for.”</p>

# FUDGE: Couplet Completions Example

---

And even thence thou wilt be stol'n, I fear,  
for this shall be the end. That's pretty clear.

---

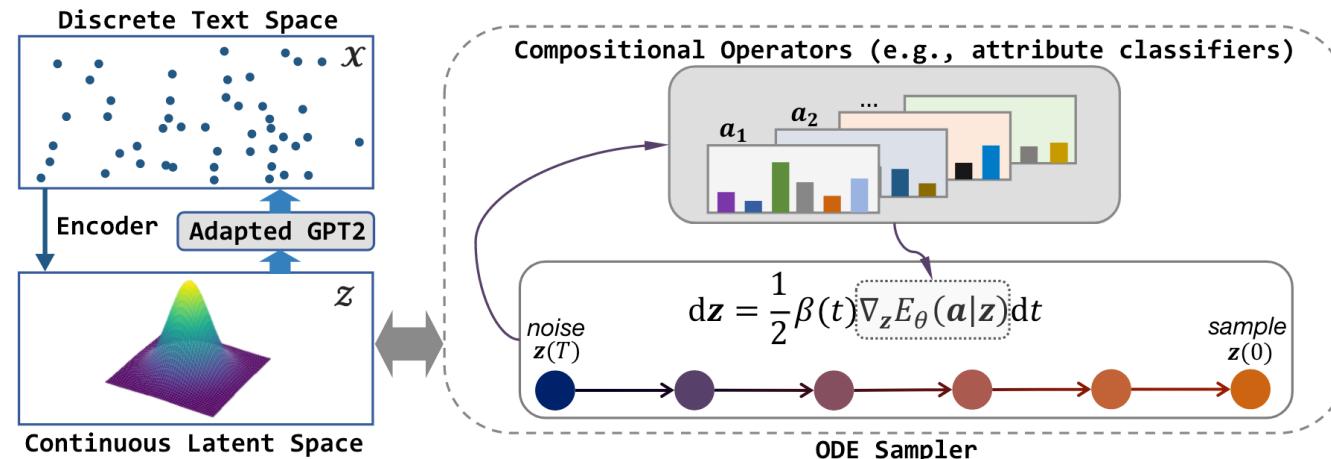
Or, if they sleep, thy picture in my sight  
I will be glad to look upon the night.

---

Table 3: Two examples of successful couplet completions (in purple) generated by FUDGE.

# Composable Text Control in Latent Space with ODEs

- *Differential* controllable text generation through a *compact latent space of text* for sample quality and efficiency.
- Adapting pre-trained LMs for latent space with VAE:
  - Encoder: BERT-small
  - Decoder: Adapted GPT-2
  - Fine-tune the encoder and some MLP layers of the decoder in the VAE framework.
- Sampling :



Methods	PPLM	FUDGE	Ours
Time (s)	3182 (578×)	36.1 (6.6×)	5.5 (1×)

Table 3: Results of generation time of each method.

# Refactor & Fine-tuning: Summary

- Refactor:
  - Better controllability.
  - Higher text quality.
  - Computationally expensive training.
  - Lack of flexibility.
- Fine-tuning LMs:
  - Efficient inference.
  - Weaker controllability.
  - Higher text quality.
- Fine-tuning with steerable layer:
  - Efficient training.
  - Computationally expensive inference.
  - Better controllability.
  - Lower text quality.

THANKS!