Algorithms for Text Generation

The awes and mysteries of **generate**

François Yvon

ISIR — CNRS and Sorbonne Université

Aussois, April 2nd, 2025

Goal and Motivations

Understanding Contextual Generation is Key

- LLMs are multitask learners & problem solvers via **prompting**
  contextual generation procedure a critical component
- evaluation of **linguistic abilities** of LLMs requires generation
  wrt. agreement, proper word order, semantic consistency etc.
- **test time inference** and multi-step "reasoning" rely on meta-generation strategies
- detecting **artificial texts** requires deep understanding of generation strategies
- **good artificial data** helps ML
  privacy, confidentiality, distillation / data augmentation, artificial text detection
- computation of expectations $\mathbb{E}_{w_{[1:T]} \sim P}(f(w_{[1:T]}))$ requires **good text samples**
  e.g., to train GANs or PPO

Goal and Motivations

Understanding Contextual Generation is Key

- LLMs are multitask learners & problem solvers via **prompting**
  contextual generation procedure a critical component
- evaluation of **linguistic abilities** of LLMs requires generation
  wrt. agreement, proper word order, semantic consistency etc.
- **test time inference** and multi-step "reasoning" rely on meta-generation strategies
- detecting **artificial texts** requires deep understanding of generation strategies
- **good artificial data** helps ML
  privacy, confidentiality, distillation / data augmentation, artificial text detection
- computation of expectations $\mathbb{E}_{w_{[1:T]} \sim P}(f(w_{[1:T]}))$ requires **good text samples**
  e.g., to train GANs or PPO

Goal and Motivations

Understanding Contextual Generation is Key

- LLMs are multitask learners & problem solvers via **prompting**
  contextual generation procedure a critical component
- evaluation of **linguistic abilities** of LLMs requires generation
  wrt. agreement, proper word order, semantic consistency etc.
- **test time inference** and multi-step “reasoning” rely on meta-generation strategies
- detecting **artificial texts** requires deep understanding of generation strategies
- **good artificial data** helps ML
  privacy, confidentiality, distillation / data augmentation, artificial text detection
- computation of expectations $\mathbb{E}_{w_{[1:T]}\sim P}(f(w_{[1:T]}))$ requires **good text samples**
  e.g., to train GANs or PPO

Goal and Motivations

Understanding Contextual Generation is Key

- LLMs are multitask learners & problem solvers via **prompting**
  contextual generation procedure a critical component
- evaluation of **linguistic abilities** of LLMs requires generation
  wrt. agreement, proper word order, semantic consistency etc.
- **test time inference** and multi-step “reasoning” rely on meta-generation strategies
- detecting **artificial texts** requires deep understanding of generation strategies
- **good artificial data** helps ML
  privacy, confidentiality, distillation / data augmentation, artificial text detection
- computation of expectations $\mathbb{E}_{w_{[1:T]} \sim P}(f(w_{[1:T]}))$ requires **good text samples**
  e.g., to train GANs or PPO

Goal and Motivations

Understanding Contextual Generation is Key

- LLMs are multitask learners & problem solvers via **prompting**
  contextual generation procedure a critical component
- evaluation of **linguistic abilities** of LLMs requires generation
  wrt. agreement, proper word order, semantic consistency etc.
- **test time inference** and multi-step “reasoning” rely on meta-generation strategies
- detecting **artificial texts** requires deep understanding of generation strategies
- **good artificial data** helps ML
  privacy, confidentiality, distillation / data augmentation, artificial text detection
- computation of expectations $\mathbb{E}_{w_{[1:T]} \sim P}(f(w_{[1:T]}))$ requires **good text samples**
  e.g., to train GANs or PPO

Goal and Motivations

Understanding Contextual Generation is Key

- LLMs are multitask learners & problem solvers via **prompting**
  contextual generation procedure a critical component
- evaluation of **linguistic abilities** of LLMs requires generation
  wrt. agreement, proper word order, semantic consistency etc.
- **test time inference** and multi-step "reasoning" rely on meta-generation strategies
- detecting **artificial texts** requires deep understanding of generation strategies
- **good artificial data** helps ML
  privacy, confidentiality, distillation / data augmentation, artificial text detection
- computation of expectations $\mathbb{E}_{w_{[1:T]}\sim P}(f(w_{[1:T]}))$ requires **good text samples**
  e.g., to train GANs or PPO

Goal and Motivations

This class

- ✔ understand the variety of controls for basic text generation
  with 🤗 **generate** as the main tool (**generate** has a similar, less complete interface)
- ✔ learn to generate texts with constraints
- ✔ explore some meta-generation algorithms

  train LLMs for better / adapted / non-autoregressive / test time ... generation

generate

```
model_inputs = tokenizer(["A sequence of numbers: 1, 2"], return_tensors="pt").to("cuda")


generated_ids = model.generate(**model_inputs)
tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]


generated_ids = model.generate(**model_inputs, max_new_tokens=50)
tokenizer.batch_decode(generated_ids, skip_special_tokens=True)[0]
```

Part I

Basics

A Language Model is a Distribution

Language Models (LM)

Assume finite vocabulary $\mathcal{V}$, with $\bar{\mathcal{V}} = \mathcal{V} \cup \{<s>, </s>\}$
A neural language model is a parameterized distribution over complete texts in $<s>\mathcal{V}^*</s>$:

$$<s> w_1 \dots w_T </s> \rightarrow \mathrm{P}(<s> w_1 \dots w_T </s> \,|\, \boldsymbol{\theta})$$

$$\forall T > 0, \forall w_1 \dots w_T, \mathrm{P}(<s> w_1 \dots w_T </s> \,|\, \boldsymbol{\theta}) \geq 0,$$

$$\sum_{T, w_{[1:T]}} \mathrm{P}(<s> w_1 \dots w_T </s> \,|\, \boldsymbol{\theta}) = 1$$

Notations:

- $w_{[1:T]} = w_1 \dots w_T$
- $[w_{[1:T]}$ assumes $w_0 = <s>$, denotes a strict prefix (unless $w_T = </s>$)
- $[w_{[1:T]}]$ assumes $w_{T+1} = </s>$, denotes a complete text
- $w_{<t} = [w_{[1:t-1]} = <s> w_1 \dots w_{t-1}$
- $[w_{-t}] : <s> \dots w_{t-1} \sqcup w_{t+1} \dots w_T </s>$
- for $w_T \neq </s>$, $\mathrm{P}([w_1 \dots w_T \,|\, \boldsymbol{\theta})$ is a prefix probability
- $\sum_{w_{[1:T]}} \mathrm{P}([w_1 \dots w_T \,|\, \boldsymbol{\theta}) = 1$ for same length prefixes

A Language Model is a Distribution

Language Models (LM)

Assume finite vocabulary $\mathcal{V}$, with $\bar{\mathcal{V}} = \mathcal{V} \cup \{\texttt{<s>}, \texttt{</s>}\}$
A neural language model is a parameterized distribution over complete texts in $\texttt{<s>}\mathcal{V}^*\texttt{</s>}$:

$$\texttt{<s>}w_1 \dots w_T\texttt{</s>} \rightarrow \mathrm{P}(\texttt{<s>}w_1 \dots w_T\texttt{</s>} \,|\, \boldsymbol{\theta})$$
$$\forall T > 0, \forall w_1 \dots w_T, \mathrm{P}(\texttt{<s>}w_1 \dots w_T\texttt{</s>} \,|\, \boldsymbol{\theta}) \geq 0,$$
$$\sum_{T, w_{[1:T]}} \mathrm{P}(\texttt{<s>}w_1 \dots w_T\texttt{</s>} \,|\, \boldsymbol{\theta}) = 1$$

Notations:

- $w_{[1:T]} = w_1 \dots w_T$
- $[w_{[1:T]}$ assumes $w_0 = \texttt{<s>}$, denotes a strict prefix (unless $w_T = \texttt{</s>}$)
- $[w_{[1:T]}]$ assumes $w_{T+1} = \texttt{</s>}$, denotes a complete text
- $w_{<\mathbf{t}} = [w_{[1:t-1]} = \texttt{<s>}w_1 \dots w_{t-1}$
- $[w_{-t}]$: $\texttt{<s>} \dots w_{t-1} \text{␣} w_{t+1} \dots w_T\texttt{</s>}$
- for $w_T \neq \texttt{</s>}$, $\mathrm{P}([w_1 \dots w_T \,|\, \boldsymbol{\theta})$ is a prefix probability
- $\sum_{w_{[1:T]}} \mathrm{P}([w_1 \dots w_T \,|\, \boldsymbol{\theta}) = 1$ for same length prefixes

Formalizing Text Generation as Search

Unconditional Text Generation: find “most likely text”

$$[w_1^* \dots w_{T^*}^*] = \operatorname*{argmax}_{T,[w_{[1:T]}]} \mathrm{P}([w_{[1:T]}] \mid \boldsymbol{\theta})$$

Finding T^* is part of the problem

Conditional Text Generation: find “most likely response” given input context / query (MAP)

$$[w_1^* \dots w_{T^*}^*] = \operatorname*{argmax}_{T,[w_{[1:T]}]} \mathrm{P}([w_{[1:T]}] \mid \boldsymbol{C}, \boldsymbol{\theta})$$

$\boldsymbol{C}$: a prefix (text completion), a question (question answering), a source text (translation), a long text (summarization), a speech file (transcription), an image (captioning), ...

A variety of situations between open set generation (many acceptable texts) and near deterministic generation (one single acceptable output)

Formalizing Text Generation as Search

Unconditional Text Generation: Find the Mode

$$
\begin{aligned}
[w_1^* \dots w_{T^*}^*] &= \operatorname*{argmax}_{T,[w_{[1:T]}]} \mathrm{P}([w_{[1:T]}] \,|\, \boldsymbol{\theta}) \\
&= \operatorname*{argmax}_{T,[w_{[1:T]}]} \prod_{t=1}^{T+1} \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) && \text{Chain rule for autoregressive / causal LMs} \\
&= \operatorname*{argmax}_{T,[w_{[1:T]}]} \log \prod_{t=1}^{T+1} \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) && \text{log is monotonous} \\
&= \operatorname*{argmin}_{T,[w_{[1:T]}]} \sum_{t=1}^{T+1} -\log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) && \text{log turns } \prod \text{ into } \sum
\end{aligned}
$$

- $-\log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) > 0$ is the surprisal; upper bounded by $\log |\mathcal{V}|$
  quantifies how much w_t was expected given $w_{<t}$, used in psycholinguistic studies
- $\max_{T,[w_{[1:T]}]} \mathrm{P}([w_{[1:T]}] \,|\, \boldsymbol{\theta})$ equivalently minimizes a summation of $T+1$ surprisals

Formalizing Text Generation as Search

Maximum “a posteriori” (MAP) Text Generation

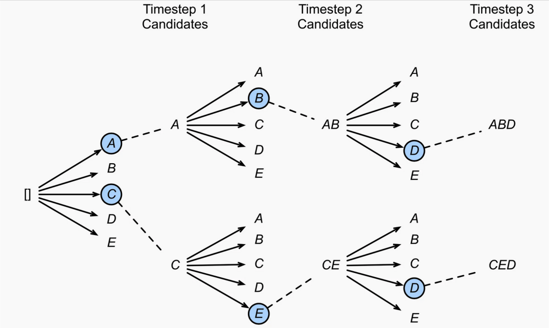
$$\mathrm{P}(w \mid w_{<t}; \boldsymbol{\theta}) = \frac{\exp \operatorname{logit}(w, w_{<t}; \boldsymbol{\theta})}{\sum_{w' \in \mathcal{V}} \exp \operatorname{logit}(w', w_{<t}; \boldsymbol{\theta})}$$

$$\log \mathrm{P}(w \mid w_{<t}; \boldsymbol{\theta}) = \operatorname{logit}(w, w_{<t}; \boldsymbol{\theta}) - \log \sum_{w' \in \mathcal{V}} \exp \operatorname{logit}(w', w_{<t}; \boldsymbol{\theta})$$

$$\begin{aligned}[w_1^* \ldots w_{T^*}^*] &= \operatorname*{argmin}_{T, [w_{[1:T]}]} -\left(\sum_{t=1}^{T+1} \operatorname{logit}(w_t, w_{<t}; \boldsymbol{\theta}) - \log \sum_{w'} \exp \operatorname{logit}(w', w_{<t}; \boldsymbol{\theta})\right) \\ &= \operatorname*{argmin}_{T, [w_{[1:T]}]} - \sum_{t=1}^{T+1} \operatorname{logit}(w_t, w_{<t}; \boldsymbol{\theta})\end{aligned}$$

- $\boldsymbol{X}$ a finite set, $f : \boldsymbol{X} \rightarrow \mathbb{R}$ a real function, $\frac{\exp f(x)}{\sum_{x' \in X} \exp f(x')}$ is the softmax
- $\operatorname{softmax}(x)$ is always > 0; almost 1 for the largest $f(x)$, almost 0 otherwise
- computing the logits requires a full forward pass in Transformers ($O(L \times (T^2 \times d_{model} + T \times d_{model}^2))$)
- normalizer $\sum_{w' \in \mathcal{V}} \exp \operatorname{logit}(w', w_{<t}; \boldsymbol{\theta})$ can be expensive to compute ($\sum$ over $|\mathcal{V}|$ terms)

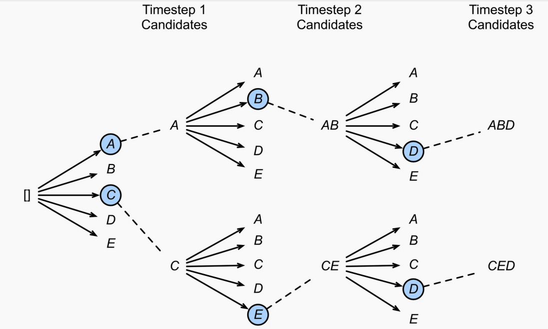
Searching, searching, searching



☞ source https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation

- $-\log \mathrm{P}(w_t \mid w_{<t}; \boldsymbol{\theta})$ factorize / decompose over arcs ⇒ incremental score computation
- $-\log \mathrm{P}(w_t \mid w_{<t}; \boldsymbol{\theta})$ depends on the entire prefix ⇒ no DP solution
- exact search is doable [Stahlberg and Byrne, 2019], yet very costly ⇒ heuristic search (greedy, DFS, Beam, A^*, etc)

Searching, searching, searching



☞ source https://towardsdatascience.com/decoding-strategies-that-you-need-to-know-for-response-generation

Variants and generalizations:

- restrict solutions to a strict subset of <s>$\mathcal{V}^*$</s>: calculability and complexity issues
- use alternative, non-decomposable score functions $F([w_{[1:t]}], \boldsymbol{C}; \theta)$
- use sparse output reward that only evaluate leaf nodes $R([w_{[1:T]}], \boldsymbol{C}; \theta)$

Language model (de)generation

Language Generation is Hard

The promise ☞ OpenAi Website https://openai.com/blog/better-language-models/

GPT-2 generates synthetic text samples in response to the model being primed with an arbitrary input. The model is chameleon-like—it adapts to the style and content of the conditioning text. This allows the user to generate realistic and coherent continuations about a topic of their choosing, as seen by the following select samples.

GPT-2 displays a broad set of capabilities, including the ability to generate conditional synthetic text samples of unprecedented quality, where we prime the model with an input and have it generate a lengthy continuation.

Language model (de)generation

The truth about language model generation

Prefix	Lyrically the song has excerpts of different languages including French , Spanish ,
Greedy	German , Italian , Portuguese , Spanish , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , Portuguese , ...
Top3	German , Italian , Portuguese , Spanish , Portuguese , Portuguese, Italian and Spanish. It is also available in English, French and Spanish. In addition to its lyrics, the album features a number of original songs, ...
Nucleus-0.3	German , Italian , Portuguese , Spanish , Portuguese , Italian , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , Spanish , Portuguese , ...

Prefix	The first pair of pereiopods is armed with a large , asymm
Greedy	etrical, and long-range laser cannon. The second pair is armed with a large , asymmetrical, and long-range laser cannon. The third pair is armed with a large , asymmetrical, and long-range laser cannon. The fourth pair is armed with a large ...
Top3	etrical, and highly mobile head, and the second pair has a small , asymmetrical, and highly mobile head. The second pair has a large and highly mobile head, and the third pair is armed with a large and highly mobile head. The first pair ...
Nucleus-0.3	etrical head and a large body. The first pair of pereiopods is armed with a large , asymmetrical head and a large body. The first pair of pereiopods is armed with a large , asymmetrical head and a large body. The first pair of pereiopods is armed ...

GPT-2 generated examples from [Welleck et al., 2020b].

Language model (de)generation

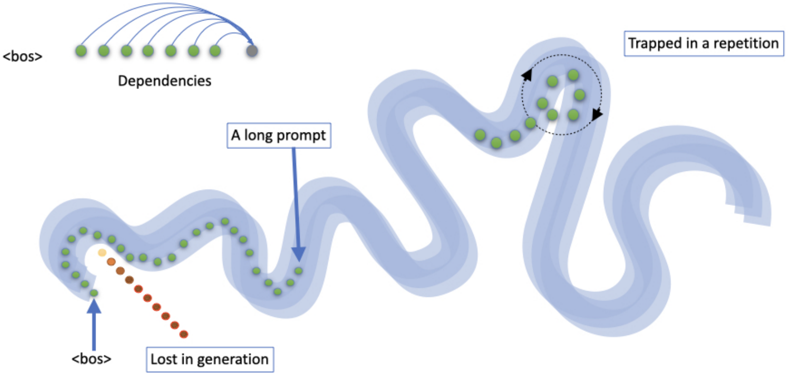
Language Generation is Hard

High probability sentences do not resemble human productions

- too many repetitions
- high frequency tokens over-represented, low frequency tokens under-represented
- lack of lexical diversity
- lack of global consistency
- posterior distribution poorly calibrated

Language model (de)generation

Language Generation is Hard



Action takes place in $\mathbb{R}^d, d \in [500; 10000]$

Evaluating Language Models with Perplexity

Perplexity of a test sequence $[w_{[1:T]}]$ [Brown et al., 1992]

$$\mathrm{PPL}(M_{\boldsymbol{\theta}}) = 2^{\frac{-1}{T}\log_2 \mathrm{P}([w_{[1:T]}]\,|\,\boldsymbol{\theta})} = \mathrm{P}([w_{[1:T]}]\,|\,\boldsymbol{\theta})^{-\frac{1}{T}}$$

Assumes "sufficiently large" T . Alt take: normalizer = T+1.

- The cross-entropy between the source (S) and model $M_{\boldsymbol{\theta}}$:

$$H(S, M_{\boldsymbol{\theta}}) = \lim_{T\to\infty} \frac{-1}{T} \log_2 \mathrm{P}([w_{[1:T]}]\,|\,\boldsymbol{\theta})$$

  $H(S, M_{\boldsymbol{\theta}})$ upper bounds $H(S)$

- PLL() homogeneous to a vocabulary size

$$\mathrm{PPL}(\mathrm{Unif}) = 2^{\frac{-1}{T}\log_2 \mathrm{P}([w_{[1:T]}]\,|\,\boldsymbol{\theta})} = 2^{\frac{-1}{T} T \log_2(1/|\mathcal{V}|)} = |\mathcal{V}|$$

Evaluating Language Models with Perplexity

PPLs are hard to compare

Comparing LMs with different support or tokenizers ?

1. closed-world LMs assume a fixed vocabulary size $|\mathcal{V}|$ - models with different $\mathcal{V}$ cannot be compared.
2. open-world models with different segmentations can be compared, must use a common normalizer
3. typical normalizers when using subwords vocabularies
   - number of chars $\Rightarrow$ bits per char $\equiv \log_2$ of char-normalized PPL
   - number of bytes $\Rightarrow$ bits per byte $\equiv \log_2$ of byte-normalized PPL

Also ? Comparing LMs for different languages?

Evaluating Language Models with Perplexity

Implementing $\sum_{t=1}^{T} \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta})$ with **finite, fixed-length window of size L**?

4 possible implementations

- split in short parts of length $T_i < L$ (lines, paragraphs), average over parts;
- "reshape" text into $\lfloor T/L \rfloor$ sequences of length L, average $\log \mathrm{P}(w_L \,|\, w_{<L})$ over blocks
- "reshape" text into $T - L$ sequences of length L with shift 1, average $\log \mathrm{P}(w_L \,|\, w_{<L})$ over blocks;
- "reshape" text into $\lfloor 2 \times (T-L)/L \rfloor$ sequences of length L with shift $L/2$, average $\sum_{t=L/2}^{L} \log \mathrm{P}(w_t \,|\, w_{<t})$ over blocks;

☞ https://huggingface.co/docs/transformers/perplexity

Evaluating Language Models with Perplexity

Implementing $\sum_{t=1}^{T} \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta})$ with **finite, fixed-length window of size L**?

4 possible implementations

- split in short parts of length $T_i < L$ (lines, paragraphs), average over parts;
- "reshape" text into $\lfloor T/L \rfloor$ sequences of length L, average $\log \mathrm{P}(w_L \,|\, w_{<L})$ over blocks
- "reshape" text into $T - L$ sequences of length L with shift 1, average $\log \mathrm{P}(w_L \,|\, w_{<L})$ over blocks;
- "reshape" text into $\lfloor 2 \times (T - L)/L \rfloor$ sequences of length L with shift $L/2$, average $\sum_{t=L/2}^{L} \log \mathrm{P}(w_t \,|\, w_{<t})$ over blocks;

☞ 🤗 https://huggingface.co/docs/transformers/perplexity

Another Caveat: segmentation ambiguities and exact surprisal computations

$$\mathrm{P}(abcd \,|\, \boldsymbol{\theta}) = \sum \mathrm{P}(a\_bcd \,|\, \boldsymbol{\theta}) + \mathrm{P}(ab\_cd \,|\, \boldsymbol{\theta}) + \ldots \mathrm{P}(abc\_d \,|\, \boldsymbol{\theta})$$

Evaluating LMs with distributional properties

rep/ℓ: a repetition / diversity metric [Welleck et al., 2020b]

Given a set $\mathcal{D}$ of length-T sequences,

$$\text{rep}/\ell = \frac{1}{|\mathcal{D}|T} \sum_{\mathbf{x}\in\mathcal{D}} \sum_{t=1}^{T} \mathbb{I}\left[w_t \in w_{t-\ell-1:t-1}\right].$$

$\mathbb{I}$ the indicator function. Generalizes to repeated n-gram sequences.

Evaluating LMs with distributional properties

rep/ℓ: a repetition / diversity metric [Welleck et al., 2020b]

Given a set $\mathcal{D}$ of length-T sequences,

$$\text{rep}/\ell = \frac{1}{|\mathcal{D}|T} \sum_{\mathbf{x}\in\mathcal{D}} \sum_{t=1}^{T} \mathbb{I}\left[w_t \in w_{t-\ell-1:t-1}\right].$$

$\mathbb{I}$ the indicator function. Generalizes to repeated n-gram sequences.

Global distributional properties [Meister and Cotterell, 2021]

- Zipfian behavior, power-law distribution

  $$\text{P}_{\text{zipf}}(W = w_k) \propto k^{-s}, s \approx 1$$

  w_k is the k^{th} most frequent token
- type-token ratios (TTR) (depend on length)
- proportion of frequency 1 words (*hapax legomena*)
- proportion specific of token classes (punctuation, stopwords, nouns, etc)
- consistency metrics ?

Evaluating zero-shot / few-shot behaviour

Reduce NLP tasks to text generation with appropriate instructions in NL as prompts

Prompts = instructions in Natural Language + [tricks] (from [Brown et al., 2020])

Specifically, we evaluate GPT-3 on over two dozen NLP datasets,(...) For each task, we evaluate GPT-3 under 3 conditions:

- "zero-shot" learning, where no demonstrations are allowed and only an instruction in natural language is given to the model.
  "Evaluate 125 + 12 ="
- "one-shot learning", where we allow only one demonstration, and
  "Evaluate 17 + 301 = 318 </s>Evaluate 125 + 12 = "
- "few-shot learning", or in-context learning, where we allow as many demonstrations as will fit into the model's context window,
  "Evaluate 17 + 301 = 318 </s>Evaluate 48 + 67 = 105 </s>Evaluate 125 + 12 = "

Tricks: *"On tasks with free-form completion, we use beam search with the same parameters as [RSR+19]: a beam width of 4 and a length penalty of $\alpha = 0.6$."* (+ stopping criterion)

Evaluating zero-shot / few-shot behaviour

Reduce NLP tasks to text generation with appropriate instructions in NL as prompts

Task types and their evaluation [Biderman et al., 2024]

Assuming prompt / instruction: $w_1 \dots w_T$.

- Yes / No answers

  Question: [`Question`] True or false? *[prediction]*

  Correct if $\mathrm{P}(\mathrm{True} \mid \mathrm{prompt}) > \mathrm{P}(\mathrm{False} \mid \mathrm{prompt})$.

- Multiple choice answers.

  Question: Which factor will most likely cause a person to develop a fever?

Correct Answer	*a bacterial population in the bloodstream*
Incorrect Answer	*a leg muscle relaxing after exercise*
Incorrect Answer	*several viral particles on the skin*
Incorrect Answer	*carbohydrates being digested in the stomach*

  Correct if $\mathrm{P}(\mathrm{Correct\ answer} \mid \mathrm{prompt}) > \mathrm{P}(\mathrm{Alternative} \mid \mathrm{prompt})$

  Alt. take - index choices with letter or numbers, evaluate the probability of the correct index.

- One word continuation. Correct if $(w_{T+1} == w^*)$

- Multiple word continuation. Measure $\Delta(w_{T+1} \dots w_{T+S}; w_1^* \dots w_L^*)$ with $\Delta()$ task-dependent distance (ROUGE for summarization, BLEU for MT, etc)

Evaluating zero-shot / few-shot behaviour

Reduce NLP tasks to text generation with appropriate instructions in NL as prompts

Understanding “instruction learning” results

Should pay attention to:

- how much effort went into prompting ?
- how many shots is few shots?
- free generation or text infilling or multi-choice answers ?
- how were alternatives selected / generated ?
- how was search performed (greedy or beam) ?
- how does generation stops?

Part II

Algorithms for Text Generation

Deterministic Algorithms for Text Generation

Searching for the Maximum “A Posteriori”

Greedy search (a.k.a argmax)

$$w_0 = \text{<s>}$$
$$\forall t > 0, w_t = \operatorname*{argmax}_{w \in \bar{\mathcal{V}}} \log \mathrm{P}(w \mid w_{<t})$$

$\bar{\mathcal{V}} = \mathcal{V} \cup \{\text{<s>}, \text{</s>}\}$

Generation stops with </s> or when some maximum length $T_{\max}$ is reached.

- Greedy search is deterministic: always produces the same output, given its initial conditions.
- Does not require to compute $\operatorname{softmax}$ normalizer $\log(\sum \exp())$

Deterministic Algorithms for Text Generation

Searching for the Maximum “A Posteriori”

Beam search [with histogram pruning]

$$\mathcal{B}_0 = \{<\mathrm{s}>\}$$

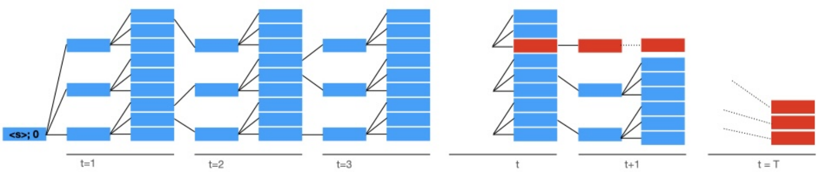
$$\forall t > 0, \mathcal{B}_t = \operatorname*{argmax}_{\substack{\mathcal{B}'_t \subseteq \boldsymbol{H}_t, \\ |\mathcal{B}'_t| = k}} \mathcal{L}(\mathcal{B}'_t)$$

$\mathcal{B}_t$ is the beam, $\boldsymbol{H}_t$ contains all possible extensions of $\boldsymbol{H}_{t-1}$.
$\mathcal{L}$ is a scoring function that operates over sets $\mathcal{B}$, eg. $\mathcal{L}(\mathcal{B}) = \sum_{w_{[1:t]} \in \mathcal{B}} \log \mathrm{P}(w_{[1:t]})$.

- For $k = 1$, beam search is greedy search
- Beam search is also deterministic
- For $k > 1$, does require to compute $\mathrm{softmax}$ normalizer $\log(\sum \exp())$.
- Also: adaptive beam size, with $\mathcal{B}_t$ containing all outputs with score within α % of the current best.
- A faster version borrows ideas from A* search [Meister et al., 2020b]
- 🤗 generate: k = `num_beams`

Deterministic Algorithms for Text Generation

Searching for the Maximum “A Posteriori”



Vanilla Beam stopping condition

$([w^*_{[1:t]}, s^*_t) = \text{argmax}_s \mathcal{B}_t, w^*_t = \text{</s>}$

In words: the top hypothesis in the beam is complete.

Deterministic Algorithms for Text Generation

Flavors of Beam Search - Delivering k solutions [Kasai et al., 2024]

k: beam size, M: maximum length,
$\mathcal{V}$: Vocabulary, $\mathrm{score}(\cdot)$: scoring function.

$B_0 \leftarrow \{\langle 0, \texttt{<s>} \rangle\}$
for $t \in \{1, \ldots, M-1\}$ **do**
  for $\langle s, w_{[1:l]} \rangle \in B_{t-1}$ **do**
    if $w_l = \texttt{</s>}$ **then**
      $H.\mathrm{add}(\langle s, w_{[1:l]} \rangle)$
      continue
    end if
    for $w \in \mathcal{V}$ **do**
      $s \leftarrow \mathrm{score}(w_{[1:l]} \circ w)$
      $H.\mathrm{add}(\langle s, w_{[1:l]} \circ w \rangle)$
    end for
  end for
  $B_t \leftarrow \varnothing$
  while $|B_t| < k$ **do**
    $\langle s, w_{[1:l]} \rangle \leftarrow H.\mathrm{max}()$
    $B_t.\mathrm{add}(\langle s, w_{[1:l]} \rangle)$
    $H.\mathrm{remove}(\langle s, w_{[1:l]} \rangle)$
  end while
  if $\forall w_{[1:l]} \in B_t, w_l = \texttt{</s>}$ **then break**
  end if
end for
return $B_t.\mathrm{max}()$

- Implementing H as a Heap, operations $(\mathrm{add}, \mathrm{remove}, \mathrm{max})$ take $O(\log |\mathcal{V}|)$
- 🤗 generate `num_beams` (k), `num_return_sequences`
- 🤗 stopping condition is `early_stopping = True` (also `False`, `never`)

Deterministic Algorithms for Text Generation

Flavors of Beam Search - Delivering k solutions [Kasai et al., 2024]

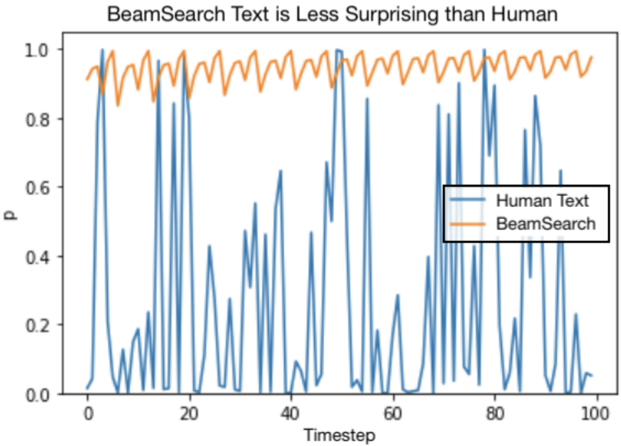
k: beam size, M: maximum length,
$\mathcal{V}$: Vocabulary, $\mathrm{score}(\cdot)$: scoring function.

$B_0 \leftarrow \{\langle 0, \texttt{<s>}\rangle\}$
for $t \in \{1, \ldots, M-1\}$ **do**
  for $\langle s, w_{[1:l]}\rangle \in B_{t-1}$ **do**
    if $w_l = \texttt{</s>}$ **then**
      $H.\mathrm{add}(\langle s, w_{[1:l]}\rangle)$
      continue
    end if
    for $w \in \mathcal{V}$ **do**
      $s \leftarrow \mathrm{score}(w_{[1:l]} \circ w)$
      $H.\mathrm{add}(\langle s, w_{[1:l]} \circ w\rangle)$
    end for
  end for
  $B_t \leftarrow \varnothing$
  while $|B_t| < k$ **do**
    $\langle s, w_{[1:l]}\rangle \leftarrow H.\mathrm{max}()$
    $B_t.\mathrm{add}(\langle s, w_{[1:l]}\rangle)$
    $H.\mathrm{remove}(\langle s, w_{[1:l]}\rangle)$
  end while
  if $\forall w_{[1:l]} \in B_t, w_l = \texttt{</s>}$ **then break**
  end if
end for
return $B_t.\mathrm{max}()$

k: beam size, M: maximum length, p patience
$\mathcal{V}$: Vocabulary, $\mathrm{score}(\cdot)$: scoring function.

$B_0 \leftarrow \{\langle 0, \texttt{<s>}\rangle\}$, $F_0 \leftarrow \varnothing$
for $t \in \{1, \ldots, M-1\}$ **do**
  $H \leftarrow \varnothing, F_t \leftarrow F_{t-1}$
  for $\langle s, w_{[1:l]}\rangle \in B_{t-1}$ **do**
    for $w \in \mathcal{V}$ **do**
      $s \leftarrow \mathrm{score}(w_{[1:l]} \circ w)$,
      $H.\mathrm{add}(\langle s, w_{[1:l]} \circ w\rangle)$
    end for
  end for
  $B_t \leftarrow \varnothing$
  while $|B_t| < k$ **do**
    $\langle s, w_{[1:l]}\rangle \leftarrow H.\mathrm{max}()$,
    if $w_l = \texttt{</s>}$ **then**
      $F_t.\mathrm{add}(\langle s, w_{[1:l]}\rangle)$
    else
      $B_t.\mathrm{add}(\langle s, w_{[1:l]}\rangle)$
    end if
    $H.\mathrm{remove}(\langle s, w_{[1:t]}\rangle)$
  end while
  if $|F_t| = pk$ **then break**
  end if
end for
return $F_t.\mathrm{max}()$

Pitfalls of Beam Search



☞ From https://huggingface.co/blog/how-to-generate

Also [Holtzman et al., 2020]. This can make artificial text detection easy.

Pitfalls of Beam Search

The Beam Search "curse"

Russian–English (medium)	Beam Size					
	10	**50**	**75**	**100**	**150**	**1000**
BLEU	24.9	23.8	23.6	23.3	22.5	3.7
METEOR	30.9	30.0	29.7	29.4	28.8	12.8
length	0.90	0.86	0.85	0.84	0.81	0.31

Results of the Russian–English translation system. We report BLEU and METEOR scores, as well as the ratio of the length of generated sentences compared to the correct translations (length). From [Murray and Chiang, 2018]

Increasing beam width k hurts performance (!)

Pitfalls of Beam Search

The Beam Search "curse"

Russian–English (medium)	Beam Size					
	10	**50**	**75**	**100**	**150**	**1000**
BLEU	24.9	23.8	23.6	23.3	22.5	3.7
METEOR	30.9	30.0	29.7	29.4	28.8	12.8
length	0.90	0.86	0.85	0.84	0.81	0.31

Results of the Russian–English translation system. We report BLEU and METEOR scores, as well as the ratio of the length of generated sentences compared to the correct translations (length). From [Murray and Chiang, 2018]

Increasing beam width k hurts performance (!)

Length issues in beam search

- Increasing k raises the likeliness of inserting a complete hypothesis in $\mathcal{B}_t$
- Complete hypotheses scores do not change;
- Incomplete hypotheses scores only gets worse
- Short sequences are more likely than longer ones

The problem is the MAP not the beam [Eikema and Aziz, 2020] ! Small beams hide this issue

Pitfalls of Beam Search

Better solutions with regularized decoding objectives [Meister et al., 2020a]

$$[w_1^* \dots w_{T^*}^*] = \operatorname*{argmin}_{T, w_{[1:T]}} \sum_{t=1}^{T+1} - \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \lambda \mathcal{R}([w_{[1:T]}])$$

$\mathcal{R}([w_{[1:T]}])$ compensates for length differences, biases towards longer sequences

1. $\mathcal{R}([w_{[1:T]}]) = T + 1$: fixed bonus for each extra word
   ~ score with average surprisal $\frac{1}{T+1} \sum_{t=1}^{T} - \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta})$
2. $\mathcal{R}_{unif}([w_{[1:T]}]) = \frac{1}{T} \sum_t (\log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \mu_t)^2$, with μ_t average surprisal
   enforces uniform information rate
3. $\mathcal{R}_{local}([w_{[1:T]}]) = \frac{1}{T+1} \sum_t (\log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \log \mathrm{P}(w_{t-1} \,|\, w_{<t-1}; \boldsymbol{\theta}))^2$,
   enforces locally uniform information rate
4. $\mathcal{R}_{max}([w_{[1:T]}]) = \frac{1}{T+1} \max_t(- \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}))$,
   enables high surprisal tokens

- 🤗 generate with `length_penalty=`λ to control output length

Pitfalls of Beam Search

Better solutions with regularized decoding objectives [Meister et al., 2020a]

$$[w_1^* \dots w_{T^*}^*] = \operatorname*{argmin}_{T, w_{[1:T]}} \sum_{t=1}^{T+1} - \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \lambda \mathcal{R}([w_{[1:T]}])$$

$\mathcal{R}([w_{[1:T]}])$ compensates for length differences, biases towards longer sequences

1. $\mathcal{R}([w_{[1:T]}]) = T + 1$: fixed bonus for each extra word
   $\sim$ score with average surprisal $\frac{1}{T+1} \sum_{t=1}^{T} - \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta})$
2. $\mathcal{R}_{unif}([w_{[1:T]}]) = \frac{1}{T} \sum_t (\log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \mu_t)^2$, with μ_t average surprisal
   enforces uniform information rate
3. $\mathcal{R}_{local}([w_{[1:T]}]) = \frac{1}{T+1} \sum_t (\log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \log \mathrm{P}(w_{t-1} \,|\, w_{<t-1}; \boldsymbol{\theta}))^2$,
   enforces locally uniform information rate
4. $\mathcal{R}_{max}([w_{[1:T]}]) = \frac{1}{T+1} \max_t (- \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}))$,
   enables high surprisal tokens

- 🤗 generate with `length_penalty=`λ to control output length

Sampling Schemes for Text Generation

Ancestral sampling

$$w_0 = \text{<s>}$$
$$\forall t > 0, w_t \sim \mathrm{P}(w \,|\, w_{<t}; \boldsymbol{\theta})$$

Recursion stops with </s> or when some maximum length $T_{\max}$ is reached.

- Ancestral sampling is non-deterministic: output varies, depending on the sharpness of $\mathrm{P}(w \,|\, w_{<t}; \boldsymbol{\theta})$
- Sampling requires 🤗 **generate** `do_sampling=True`
- softmax is very peaked: increase diversity with temperature τ to "flatten" the distribution with $\exp \frac{\mathrm{logit}(w', w_{<t}; \boldsymbol{\theta})}{\tau}$ (τ is 🤗 **generate** `temperature`)
- better trade-off between likelihood and diversity [Keskar et al., 2019]:

$$\mathrm{P}(w' \,|\, w_{<t}; \boldsymbol{\theta}) \propto \exp \frac{\mathrm{logit}(w', w_{<t}; \boldsymbol{\theta})}{\tau \times \mathbb{I}(w' \in w_{<t})},$$

with $\mathbb{I}(w' \in w_{<t}) = 1$ for "new tokens", $= \lambda > 1$ for "old ones" (`repetition_penalty` for 🤗 generate)

Sampling Schemes for Text Generation

Top-k sampling [Fan et al., 2018]

$$
\begin{aligned}
w_0 &= \texttt{<s>} \\
Q(w_t \mid w_{<t}) &\propto \begin{cases} P(w_t \mid w_{<t}; \boldsymbol{\theta}) \text{ if } w \in \text{top-k}(P(W \mid w_{<t}; \boldsymbol{\theta})) \\ 0 \text{ otherwise} \end{cases} \\
\forall t > 0, w_t &\sim Q(w \mid w_{<t})
\end{aligned}
$$

Sample from a "truncated" distribution containing the k most likely symbols. Generation stops with </s> or when some maximum time step $T_{\max}$ is reached.

- Finding the k most likely tokens is $O(|\mathcal{V}| * \log k)$, the normalizer applies only over k elements.
- 🤗 - generate `top_k`

Sampling Schemes for Text Generation

Nucleus sampling (top p, with variable p) [Holtzman et al., 2020]

$$w_0 = \text{<s>}$$

$$Q(w_t \mid w_{<t}) \propto \begin{cases} P(w_t \mid w_{<t}; \boldsymbol{\theta}) & \text{if } w \in \text{top-p}(P(W \mid w_{<t}; \boldsymbol{\theta})) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall t > 0, w_t \sim Q(w \mid w_{<t})$$

p is the smallest integer such that $\sum_{w \in \text{top-p}} P(w \mid w_{<t}; ; \boldsymbol{\theta}) > \alpha$. Sample from a "truncated" distribution for the p most likely symbols, with variable p (α typically $\in [0.7; 0.9]$).

- α controls the size of the truncated vocabulary ($Q(w \mid w_{<t}) > 0$).
- 🤗 - generate `top_p`

Sampling Schemes for Text Generation

Locally Typical Sampling [Meister et al., 2023]

$$w_0 = \text{<s>}$$

$$Q(w_t \mid w_{<t}) \propto \begin{cases} P(w_t \mid w_{<t}; \boldsymbol{\theta}) & \text{if } w \in \text{LTStop-p}(P(W \mid w_{<t}; \boldsymbol{\theta})) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall t > 0, w_t \sim Q(w \mid w_{<t})$$

$\text{LTStop-p}(P(W \mid w_{<t}; \boldsymbol{\theta}))$ minimize $\sum |H(W|w_{<t}; \boldsymbol{\theta}) + \log P(w \mid w_{<t}; \boldsymbol{\theta})|$ subject to $\sum_{w \in \text{LTStop-p}} P(w \mid w_{<t}; \boldsymbol{\theta}) > \alpha$. Sample from a "truncated" distribution for the p most locally typical symbols, with variable p (α typically $\in [0.7; 0.9]$).

- Locally typical prefers tokens with near average surprisal
- In low uncertainty contexts, prefer high probability tokens
- In high uncertainty contexts, pick token with near average surprisal (=information content)
- 🤗 generate: `typical_p`
- related: Mirostat [Basu et al., 2021], sample with a target perplexity.

Sampling Schemes for Text Generation

Top-k, top-p and typical sample from a truncated distribution $\mathrm{Q}(W \mid <_t; \boldsymbol{\theta})$:

- $\forall t$, select vocabulary $\mathcal{V}_t^+ \subset \mathcal{V}$.
- $\forall t, w \notin \mathcal{V}_t^+, \mathrm{Q}(w \mid <_t; \boldsymbol{\theta}) = 0$

Always sampling high probability words avoids derailing, yet, can be very risky:

1. generation may no longer terminate $\Rightarrow$ probability leakage to infinite strings.
2. may exclude interesting words
   Using top-p, for $p = 0.9$, $\mathrm{P}(\textit{Duck} \mid \textit{Donald}) = 0.95$ may exclude w =Trump
3. may include unlikely words
   Using top-k, $k = 20$ may generate unlikely continuations for low-entropy distributions

Sampling Schemes for Text Generation

Top-k, top-p and typical sample from a truncated distribution $\mathrm{Q}(W \mid <_t; \boldsymbol{\theta})$:

- $\forall t$, select vocabulary $\mathcal{V}_t^+ \subset \mathcal{V}$.
- $\forall t, w \notin \mathcal{V}_t^+, \mathrm{Q}(w \mid <_t; \boldsymbol{\theta}) = 0$

Always sampling high probability words avoids derailing, yet, can be very risky:

1. generation may no longer terminate $\Rightarrow$ probability leakage to infinite strings.
2. may exclude interesting words
   Using top-p, for $p = 0.9$, $\mathrm{P}(\textit{Duck} \mid \textit{Donald}) = 0.95$ may exclude w =Trump
3. may include unlikely words
   Using top-k, $k = 20$ may generate unlikely continuations for low-entropy distributions

Remedies

- solve (1) with **consistent truncated sampling** [Welleck et al., 2020a]: $\mathcal{V}_t^+ \rightarrow \mathcal{V}_t^+ \cup \{$</s>$\}$
- how to mitigate (2) and (3) ? what is the right size for $\mathcal{V}_t^+$?

(P1) never truncate high probability words $\Leftrightarrow$ keep all w such that $\mathrm{P}(w \mid w_{<t}; \boldsymbol{\theta}) > \epsilon$;
(P2) truncate more when entropy is low; truncate less when entropy is high

(P*) sample only w for which the true $\mathrm{P}(w \mid <_t; \boldsymbol{\theta})$ is provably > 0 (with rejection sampling) [Finlayson et al., 2024]

Sampling Schemes for Text Generation

η-Sampling [Hewitt et al., 2022]

$$
\begin{aligned}
w_0 &= \textcolor{blue}{<\text{s}>} \\
Q(w_t \mid w_{<t}) &\propto \begin{cases} P(w_t \mid w_{<t}; \boldsymbol{\theta}) & \text{if } w \in \textcolor{red}{\mathcal{V}_t^+} \\ 0 & \text{otherwise} \end{cases} \\
\forall t > 0, w_t &\sim Q(w \mid w_{<t}) \\
\textcolor{red}{\mathcal{V}_t^+} &= \{w \in \mathcal{V} \mid P(w \mid w_{<t}; \boldsymbol{\theta}) \geq \min(\epsilon, \alpha \exp -H(W_t | w_{<t}; \boldsymbol{\theta})\}
\end{aligned}
$$

Sample from a “truncated” distribution subject to principles (P1) and (P2).

- $\alpha \exp -H(W_t | w_{<t}; \boldsymbol{\theta})$ increases the sampling set when entropy is high
- Yields better samples than typical, greedy, ancestral, nucleus and top-k
- In [Hewitt et al., 2022]'s experiments, $\epsilon = 0.0003, \alpha = \sqrt{\epsilon}$
- 🤗 - generate `epsilon_cutoff`, `eta_cutoff`

Consistent Decoding for Consistent Models

Why we need a maximum decoding length

Consistent model (details in [Welleck et al., 2020a])

A consistent model is such that $\mathrm{P}(|w_{[1:T]}| = \infty \mid \boldsymbol{\theta}) = 0$

A sufficient condition is that hidden states are uniformely bounded.
This implies that $\exists \xi, \forall, t, w_{<t}, \mathrm{P}(\text{</s>} \mid w_{<t}; \boldsymbol{\theta}) > \xi$

$$\mathrm{P}(|w_{[1:T]}| = T \mid \boldsymbol{\theta}) < (1-\xi)^T$$
$$\lim_{T\to\infty} (1-\xi)^T = 0$$

Consistent Decoding for Consistent Models

Why we need a maximum decoding length

Consistent decoding algorithm

A consistent decoding algorithm generates a complete text with probability 1.

Inconsistency of decoding

Ancestral is consistent, greedy, beam, top-k, nucleus, typical, etc. are not consistent.
Argument: no guarantee that </s> will ever appear in the top-k, top-p, etc.

Consistent Decoding for Deterministic Search

$$w_0 = \text{<s>}$$

$$\mathrm{Q}(w_t \mid w_{<t}; \boldsymbol{\theta}) \propto \begin{cases} 1 - \alpha(h_t) & \text{if } w = \text{</s>} \\ \frac{\alpha(h_t) \exp \operatorname{logit}(w, w_{<t}; \boldsymbol{\theta})}{\sum_{w'} \exp \operatorname{logit}(w', w_{<t}; \boldsymbol{\theta})} & \text{otherwise} \end{cases}$$

$$\alpha(h_0) = \sigma(\operatorname{logit}(\text{</s>}, \text{<s>}; \boldsymbol{\theta})) \quad (1)$$

$$\alpha(h_t) = \sigma(\operatorname{logit}(\text{</s>}, w_{<t}; \boldsymbol{\theta}))(1 - \mathrm{P}(\text{</s>} \mid w_{<t}; \boldsymbol{\theta})) \quad (2)$$

With $\sigma : \mathbb{R} \to [0; 1 - \epsilon]$, $\epsilon > 0, \epsilon < 1$. This ensures that $\mathrm{Q}(\text{</s>} \mid w_{<t}; \boldsymbol{\theta})$ is monotonically increasing, meaning that </s> eventually happen.

Promoting Diversity in Text Generation

Diversity promotion has many forms

1. boosting surprisal in open-ended text generation
2. ensuring diversity in a set of solutions
3. mitigating repetition in texts (difficult - repetition can be a good thing)

Promoting Diversity in Text Generation

Boosting surprisal in open-ended text generation

Contrasting Expert and Amateur Models

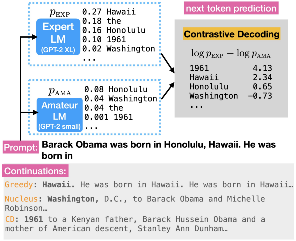
New search objective:

$$w_1^* \dots w_{T^*}^* = \operatorname*{argmax}_{T, w_{[1:T]}} \sum_{t=1}^{T} \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}) - \log \mathrm{P}(w_t \,|\, w_{<t}; \boldsymbol{\theta}_{AMA})$$

$$\text{subject to} \forall t, w_t^* \in \mathcal{V}_t^+$$

$$\mathcal{V}_t^+ = \{w \in \mathcal{V} | \mathrm{P}(w \,|\, w_{<t}; \boldsymbol{\theta}) \geq \alpha \max_{w'} \mathrm{P}(w' \,|\, w_{<t}; \boldsymbol{\theta})\}$$

Select probable words that are unlikely for a weaker amateur model. Constraining the search to high probability words helps handle cases where (a) Expert and Amateur agree on very low probability; (b) Expert and Amateur agree on very high probability. Also respects (P1).



From [Li et al., 2023]

- requires consistent tokenization for expert and amateur
- see also: https://arxiv.org/pdf/2305.12675.pdf

Promoting Diversity in Text Generation

Generating Multiple Diverse Solutions

Ensuring Diversity in Beam Search

Maintains G beams $\mathcal{B}_t^1 \dots \mathcal{B}_t^G$, such that hypotheses in Beam g must be diverse with respect to $\mathcal{B}_t^1 \dots \mathcal{B}_t^{g-1}$

$$\begin{aligned}\text{score}(w_{[1:l]}, g) &= \text{score}(w_{[1:l]}) \text{ if } g = 1 \\ &= \text{score}(w_{[1:l]}) + \lambda \sum_{h=1}^{g-1} \Delta(w_{[1:l]}, \mathcal{B}_t^h), \text{ otherwise} \\ \Delta(w_{[1:l]}, \mathcal{B}_t^h) &= \sum_{w'_{[1:l']} \in \mathcal{B}_t^h} \delta(w_{[1:l]}, w'_{[1:l']}), \text{with } \delta \text{ a similarity function}\end{aligned}$$

- Δ can be any string comparison (set differences for bag-of-words or bag-of-ngrams; Levenshtein distance; neural similarity, etc.)
- beams can run in parallel with a time delay
- 🤗 generate: `num_beam_groups` (G), `diversity_penalty` (λ)

Promoting Diversity in Text Generation

Avoiding Repetitions

Contrastive Search (greedy version) [Su et al., 2022]

$$w_0 = \text{<s>}$$
$$\forall t > 0, w_t = \underset{w \in \bar{\mathcal{V}}}{\operatorname{argmax}} \, (1-\alpha) \log \mathrm{P}(w | w_{<t}) - \alpha \max\{\mathrm{sim}(h_w, h_{w_s}) : 1 \leq s \leq t-1\}$$

h_w is the latent representation associated to w; sim is a similarity function (e.g. cosine). Extra penalty term for repetitions. Generation stops with </s> or when some maximum length T_{max} is reached.

- assumes repetitions can be detected in embedding space
- 🤗 generate: `penalty_alpha=` α ☞ https://huggingface.co/blog/introducing-csearch
- naive version with `no_repeat_ngram_size`: disable n-gram repetition
- 🤗 DoLa contrasts inner vs. outer layers to increase factuality [Chuang et al., 2024]

Combining Beam-Search and Sampling

k: beam size, M: maximum length,
$\mathcal{V}$: Vocabulary, $\mathrm{score}(\cdot)$: scoring function.

1: $B_0 \leftarrow \{\langle 0, \text{<s>} \rangle\}$
2: **for** $t \in \{1, \dots, M-1\}$ **do**
3:     **for** $\langle s, w_{[1:l]} \rangle \in B_{t-1}$ **do**
4:         **if** $w_l = \text{</s>}$ **then**
5:             $H.\mathrm{add}(\langle s, w_{[1:l]} \rangle)$
6:             **continue**
7:         **end if**
8:         **for** $i \in \mathrm{range}(k)$ **do**
9:             $< logp, w > \sim \mathrm{P}(W \,|\, w_{[1:l]}; \boldsymbol{\theta})$
10:             $H.\mathrm{add}(\langle s + logp, w_{[1:l]} \circ w \rangle)$
11:         **end for**
12:     **end for**
13:     $B_t \leftarrow \varnothing$
14:     **while** $|B_t| < k$ **do**
15:         $\langle s, w_{[1:l]} \rangle \leftarrow H.\mathrm{max}()$
16:         $B_t.\mathrm{add}(\langle s, w_{[1:l]} \rangle)$
17:         $H.\mathrm{remove}(\langle s, w_{[1:l]} \rangle)$
18:     **end while**
19:     **if** $\forall w_{[1:l]} \in B_t, w_l = \text{</s>}$ **then break**
20:     **end if**
21: **end for**
22: **return** $B_t.\mathrm{max}()$

- 🤗 licences `do_sampling=True` and `num_beams > 0` !
- the Heap H never contains more than k^2 entries
- sampling on line 9 can implement any sampling scheme (top-k, top-p, etc)
- alt take 1: sample from $H_t \propto$ local scores (line 15)
- alt take 2: Kool et al. [2019-06-09/2019-06-15]

Faster Generation with Speculative Sampling

Details in [Leviathan et al., 2023] and [Chen et al., 2023]

Overview

- Sampling algorithms are autoregressive: they return one sample at each timestep.
- At step t speculative sampling uses a simpler model to generate S draft tokens $w_{t+1} \ldots w_{t+S}$ autoregressively, then “validates” the tokens with the large model in parallel with accept / reject procedure.
- Why? Potential to validate multiple tokens in one parallel forward pass.

[START] japan ' s benchmark ~~bond~~ n

[START] japan ' s benchmark nikkei 22 ~~,~~5

[START] japan ' s benchmark nikkei 225 index rose 22 ~~,~~6

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~,~~ points

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or ~~0~~ 1

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 98~~5~~9

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 ~~,~~ in

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in ~~tokyo~~ late

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in late morning trading . [END]

Figure from Leviathan et al. [2023], K > 4

🤗 **generate**: `assistant_model (assistant_tokenizer)`

Faster Generation with Speculative Sampling

Details in [Leviathan et al., 2023] and [Chen et al., 2023]

sample K drafts $[w_{t+i}, q(w_{t+i})], i = 1 \dots K$
evaluate drafts $[w_{t+i}, p(w_{t+i})]$
sample $u_i \sim \mathrm{Unif}[0:1], i = 1 \dots K$
accept $\leftarrow$ **True** ; $i \leftarrow 1$
while accept **and** $i \leq K$ **do**
  if $q(w_{t+i}) < p(w_{t+i})$ **then**
    $i \leftarrow i + 1$ ▷ accept
  else if $u_i < \frac{q(w_{t+i})}{p(w_{t+i})}$ **then**
    $i \leftarrow i + 1$ ▷ accept
  else
    accept $\leftarrow$ **False** ▷ reject
    $\forall w, r(w) \propto (\max(0, p(w) - q(w)))$
    sample $w_{t+i} \sim r(w)$
  end if
end while

Notations:

- $p(w) = \mathrm{P}(W \mid w_{<t}; \boldsymbol{\theta})$, $q(w) = \mathrm{Q}(W \mid w_{<t}; \boldsymbol{\theta}')$
- $\mathcal{V}_+ = \{w \mid q(w) > p(w)\}$ oversampled tokens
- $\mathcal{V}_- = \{w \mid q(w) \leq p(w)\}$; undersampled tokens

Claim: speculative sampling generates tokens under $p(w)$

1. $w \in \mathcal{V}_+$? accept with proba $\frac{p(w)}{q(w)} \Rightarrow p'(w) = q(w) \times \frac{p(w)}{q(w)} = p(w)$
2. $w \in \mathcal{V}_-$? $p'(w) = q(w)$ always accept and there is a second chance:

$$p'(w) += \sum_{v \in \mathcal{V}_+} q(v) \times \left(1 - \frac{p(v)}{q(v)}\right) \times \left(\frac{p(w) - q(w)}{\sum_{w' \in \mathcal{V}_-} p(w') - q(w')}\right) = p(w) - q(w)$$

Part III

Constrained Generation

Constraining Text Generation

Generating with simple constraints

- length constraints (soft and hard) – for beam search
- no repetition (soft and hard penalties)
- with in-text / cross-text diversity (soft and hard penalties)

Constraining Text Generation

Generating with simple constraints

- length constraints (soft and hard) – for beam search
- no repetition (soft and hard penalties)
- with in-text / cross-text diversity (soft and hard penalties)

A smorgasbord of additional constraints

- lexical / terminological choices (positive and negative, hard and soft) [Keskar et al., 2019]
- language, idiom, sociolect (hard)
- style, consistency, toxicity, polarity, stance, etc (soft)
- optimizing other global scores: alignment score, backward model (translation); coverage score (summarization), etc.

Updated search goals: **restricted search space** (hard), **new search objective** (soft)

Guiding Decoding with Soft Constraints

Soft constraints

A **soft or probabilistic constraint** for text $w_{[1:T]}$ is a model $\mathrm{P}(A \mid w_{[1:T]}, \mathbf{C}; \boldsymbol{\lambda})$, where A is a (binary) discrete attribute representing the constraint.

For instance: $A = 1$ for harmfull / toxic texts, 0 for harmless content;

Probabilistic constraints can be learned from supervision:

- "generatively" with $\mathrm{P}(w_{[1:T]} \mid a, \mathbf{C}; \lambda) \forall a$: learns / adapt multiple LMs - potentially costly
- "discriminatively" with $\mathrm{P}(A \mid w_{[1:T]}, \mathbf{C}; \lambda)$: LM + classification head

Generative to discriminative score use Bayes rule

$$\mathrm{P}(A \mid w_{[1:T]}, \mathbf{C}; \lambda) \propto \mathrm{P}(A)\,\mathrm{P}(w_{[1:T]} \mid A, \mathbf{C}; \lambda)$$

Guiding Decoding with Soft Constraints

Soft constraints

A **soft or probabilistic constraint** for text $w_{[1:T]}$ is a model $\mathrm{P}(A \mid w_{[1:T]}, \mathbf{C}; \boldsymbol{\lambda})$, where A is a (binary) discrete attribute representing the constraint.

For instance: $A = 1$ for harmfull / toxic texts, 0 for harmless content;

Decoding with constraints

A LM computes $\mathrm{P}(w_{[1:T]} \mid \boldsymbol{\theta})$, how to generate $w_{[1:T]}$ that simultaneously

- is likely fluent: high $\log \mathrm{P}(w_{[1:T]} \mid \mathbf{C}; \boldsymbol{\theta})$
- likely satisfies constraint: high $\log \mathrm{P}(A \mid w_{[1:T]}, \mathbf{C}; \boldsymbol{\lambda})$?

one requirement is based on the LM prior, one on the class posterior

Guiding Decoding with Soft Constraints

Soft constraints

A **soft or probabilistic constraint** for text $w_{[1:T]}$ is a model $\mathrm{P}(A \mid w_{[1:T]}, \mathbf{C}; \boldsymbol{\lambda})$, where A is a (binary) discrete attribute representing the constraint.

For instance: $A = 1$ for harmfull / toxic texts, 0 for harmless content;

Training-based methods

- fine-tuning, VAEs, GAN – all these methods requires retraining a model
- **[Ctrl]**, a class-conditional models (with class tokens) [Keskar et al., 2019].
  Learns $\boldsymbol{\theta}$ with [ctrl:]$w_1 \ldots w_T$, a model for $\mathrm{P}(w_{[1:T]} \mid [\text{ctrl:}]; \boldsymbol{\theta})$
  - [ctrl:] is generic - represent style or domain or language or even length.
  - Require a finite set of predefined control codes for training
- **GeDi** [Krause et al., 2021] trains [ctrl] with $\{a, \bar{a}\}$ and guide generation with Bayes rule

$$\mathrm{P}(A = a \mid w_{[1:T]}; \boldsymbol{\lambda}) = \frac{\mathrm{P}(a) \prod_t \mathrm{P}(w_t \mid w_{<t}, a; \boldsymbol{\lambda})}{\sum_{a'} \mathrm{P}(a') \prod_t \mathrm{P}(w_t \mid w_{<t}, a'; \boldsymbol{\lambda})}$$

  Soft constraint A is promoted in decoding with $\mathrm{P}(w \mid w_{<t}; \boldsymbol{\theta})\, \mathrm{P}(a \mid w_{[1:t-1]} w; \boldsymbol{\theta}')^{\alpha}$

  The trick is to compute $\mathrm{P}(w_t \mid w_{<t}, A; \boldsymbol{\theta})$ in parallel for $a, \bar{a}$

Generating with Hard Rational Constraints

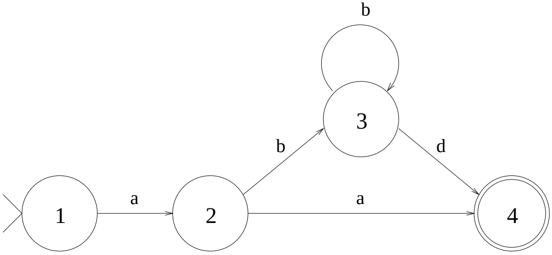
Multiple types of Hard Constraints

1. watch your language 🤗 `bad_words_ids`
2. force words in output (e.g., QA, MT with term constraints): 🤗 `force_words_ids`
3. question answering with fixed choices
4. structured answers (e.g. JSON records or csv tables)
5. code generation

Generating with Hard Rational Constraints

Rational Languages

Rational languages are languages represented by Rational Expressions (a.k.a RegExps), are also languages represented by (Deterministic) Finite Automata (DFAs).



Accomodate finite lists of words and sequences, numerics, http / mail addresses, etc

Generating with Hard Rational Constraints

Implementing Rational Constraints

Requirements:

1. Transitions mapping (states, words) to next states.
   - restrict choice to valid continuations
   - apply transition; update state
2. List of final states: add </s> to valid word list

Caveats

1. increase complexity (one search / state)
2. words are not tokens
3. compatible with beam?
4. generalizes to simple (deterministic) CF grammars

Check it out - with `outlines` library: https://github.com/dottxt-ai/outlines

Part IV

Meta-Generation Strategies

Meta-generation techniques

Motivations

- **complex constraints** in generation
- **generation of long, structured output**:
  justifications, “reasonning” steps, code, etc

Adavanced Search Strategies

1. **parallel search** (combines multiple complete generations)
   - reranking (pick one out-of-N)
   - transform (build a new one out-of-N)
2. **heuristic tree search** (MCTS, A^*)
3. **refinement**, local search, self-critics, self-improvement, etc

+ Hybrid strategies: eg., N tree-search, then aggregate, etc.

Meta-generation techniques

Unifiying terms

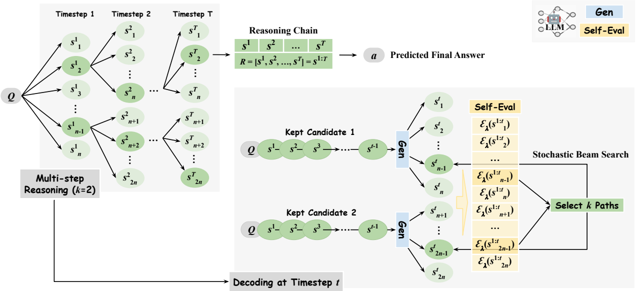
- context $S \equiv$ input prompt (+ critics / refinements) $\mathbf{C}$
- search states $S \equiv$ generated prefix, can be complete or incomplete
- basic action $\equiv$ generation of one token w
- policy $v^{\pi_{\boldsymbol{\theta}}}(S) = w$ means $w = \operatorname{argmax} \mathrm{P}(W|S;\boldsymbol{\theta})$ or $w \sim \mathrm{P}(W|S;\boldsymbol{\theta})$
- final / output reward $R(S, \mathbf{C})$, for S complete state.
  - $R(S)$ boolean: grammaticality test, hard constraint, provable solution
  - $R(S)$ scalar: soft constraint
  - $R(S)$ approximated or learned with confidence estimation $\Rightarrow \hat{R}(S, \mathbf{C})$

  a.k.a verifier model
- state value $v^{\pi}(S) =\equiv \mathbb{E}_{w_{[1:T]} \sim \mathrm{P}(\,|S;\theta)}(R(S \oplus w_{[1:T]}, \mathbf{C}))$, can be estimated $\hat{v}^{\pi}_{\phi}(S)$ or learned $v^{\pi}_{\phi}(S)$
- intermediary **steps** decompose $[w_{[1:T]}]$ as $[w_{[1:T_1]} \ldots w_{[1:T_{K-1}]} w_{[1:T_K]}]$. $w_{[1:T_K]}$ is the final output.
- intermediary steps can be scored too !

Illustration: Beam Search with Self-Evaluation

Problem statement: improved search for mathematical “reasoning”

•



from [Xie et al., 2023]

Changes to standard beam search

1. generate n **complete steps** for each of k states in beam $\mathcal{B}_{l-1}$
2. **evaluate** $y_{[1:T_l]} : G(y_{[1:T_l]} | [y_{[1:T_1]} \dots y_{[1:T_{l-1}]})$ with **auxiliary model** (nk times)
3. **sample** k best steps in $\mathcal{B}_{l-1}$ according to:

$$\mathcal{E}_\lambda(S) = \log \mathrm{P}([y_{[1:T_1]} \dots y_{[1:T_l]} \,|\, \mathbf{C}; \boldsymbol{\theta}) + G(y_{[1:T_l]} | [y_{[1:T_1]} \dots y_{[1:T_{l-1}]})^\lambda$$

Reranking 101

Picking one out-of-M

Reranking as Meta Generation

1. **generate** M complete solutions $\mathcal{W}_S = \{[w_{[1:T]}]^{(m)}, m = 1 \dots M\}$
   e.g., based on $\log \mathrm{P}([w_{[1:T]}] \,|\, \mathbf{C}; \boldsymbol{\theta})$
2. **evaluate** $[w_{[1:T]}]^{(m)}$ with output reward $R([w_{[1:T]}], \mathbf{C}', \boldsymbol{\theta}')$
3. **return** $[w_{[1:T]}]^* = \mathrm{argmin}_m R([w_{[1:T]}]^{(m)}, \mathbf{C}', \boldsymbol{\theta}')$

Reranking 101

Picking one out-of-M

Reranking as Meta Generation

1. **generate** M complete solutions $\mathcal{W}_S = \{[w_{[1:T]}]^{(m)}, m = 1 \dots M\}$ e.g., based on $\log \mathrm{P}([w_{[1:T]}] \mid \mathbf{C}; \boldsymbol{\theta})$
2. **evaluate** $[w_{[1:T]}]^{(m)}$ with output reward $R([w_{[1:T]}], \mathbf{C}', \boldsymbol{\theta}')$
3. **return** $[w_{[1:T]}]^* = \operatorname{argmin}_m R([w_{[1:T]}]^{(m)}, \mathbf{C}', \boldsymbol{\theta}')$

- Design of **generate** (for M): (diverse) beam-search ? (diverse) sampling ? stochastic beam search ? Multiple models and checkpoints ? Multiple prompts? Impact of M?

  🤗 `num_return_sequences`

- Design of **evaluate**: length control; score of a larger or better model ($\boldsymbol{\theta}'$); increased context (**C'**); use auxiliary models of grammaticality, style, toxicity, stance, polarity; use result of execution (code); also watermarking; privacy; etc.

Reranking 101

Picking one out-of-N

Voting as Meta Generation

1. **generate** M solutions $\mathcal{W}_S = \{[w^{(m)}]_{[1:T]}, m = 1 \ldots M\}$ based on model $\log \mathrm{P}([w^m]_{[1:T]} \mid \mathbf{C}; \boldsymbol{\theta})$
2. **evaluate** $[w^{(m)}]_{[1:T]}$ with output reward $R([w_{[1:T]}], \mathbf{C}', \boldsymbol{\theta}')$
3. Voting procedures:
   1. **return** $[w^*_{[1:T]}] = \operatorname{argmax}_{[w_{[1:T]}]} \sum_m \mathbb{I}([w_{[1:T]}]^{(m)} = [w_{[1:T]}])$ (**simple vote**)
   2. **return** $[w^*_{[1:T]}] = \operatorname{argmax}_{[w_{[1:T]}]} \sum_m \lambda_m \mathbb{I}([w_{[1:T]}]^{(m)} = [w_{[1:T]}])$ with $\lambda_m \propto R([w_{[1:T]}]^{(m)})$ (**weighted vote**)

Reranking 101

Picking one out-of-N

Voting as Meta Generation

1. **generate** M solutions $\mathcal{W}_S = \{[w^{(m)}]_{[1:T]}, m = 1 \dots M\}$ based on model $\log \mathrm{P}([w^m]_{[1:T]} \mid \mathbf{C}; \boldsymbol{\theta})$
2. **evaluate** $[w^{(m)}]_{[1:T]}$ with output reward $R([w_{[1:T]}], \mathbf{C}', \boldsymbol{\theta}')$
3. Voting procedures:
   1. **return** $[w^*_{[1:T]}] = \mathrm{argmax}_{[w_{[1:T]}]} \sum_m \mathbb{I}([w_{[1:T]}]^{(m)} = [w_{[1:T]}])$ (**simple vote**)
   2. **return** $[w^*_{[1:T]}] = \mathrm{argmax}_{[w_{[1:T]}]} \sum_m \lambda_m \mathbb{I}([w_{[1:T]}]^{(m)} = [w_{[1:T]}])$ with $\lambda_m \propto R([w_{[1:T]}]^{(m)}$ (**weighted vote**)

- Design of **generate** (for M): (diverse) beam-search ? (diverse) sampling ? stochastic beam search ? Multiple models and checkpoints ? Multiple prompts? Impact of M?
  🤗 `num_return_sequences`
- Design of **evaluate**: length control; score of a larger or better model ($\boldsymbol{\theta}'$); increased context (**C'**); use auxiliary models of grammaticality, style, toxicity, stance, polarity; use result of execution (code); also watermarking; privacy; etc.

Main compute tradeoff: M vs. cost of one generation

Reranking 101

Picking one out-of-N

Voting as Meta Generation

1. **generate** M solutions $\mathcal{W}_S = \{[w^{(m)}]_{[1:T]}, m = 1 \ldots M\}$ based on model $\log \mathrm{P}([w^m]_{[1:T]} \mid \mathbf{C}; \boldsymbol{\theta})$
2. **evaluate** $[w^{(m)}]_{[1:T]}$ with output reward $R([w_{[1:T]}], \mathbf{C}', \boldsymbol{\theta}')$
3. Voting procedures:
   1. **return** $[w^*_{[1:T]}] = \mathrm{argmax}_{[w_{[1:T]}]} \sum_m \mathbb{I}([w_{[1:T]}]^{(m)} = [w_{[1:T]}])$ (**simple vote**)
   2. **return** $[w^*_{[1:T]}] = \mathrm{argmax}_{[w_{[1:T]}]} \sum_m \lambda_m \mathbb{I}([w_{[1:T]}]^{(m)} = [w_{[1:T]}])$ with $\lambda_m \propto R([w_{[1:T]}]^{(m)}$ (**weighted vote**)

- for $R([w_{[1:T]}], \mathbf{C}', \boldsymbol{\theta}')$ binary, recovers the hard constraint case – akin to **rejection sampling**
- for $[w_{[1:T]}] = [w_{[1:T_r]} \oplus w_{[1:T_a]}]$ comprising "reasoning" and answer part, returning $[w^*_{[1:T]}] = \mathrm{argmax}_{[w_{[1:T]}]} \sum_m \mathbb{I}([w_{[1:T_a]}]^{(m)} = [w_{[1:T]}])$ is **self-consistency**, marginalizes over "reasoning" steps.

Minimum Bayes Risk Decoding

Context and Concepts

$\ell([w_{[1:T]}],[v_{[1:S]}]) : (\text{<s>}\mathcal{V}^*\text{</s>}) \times (\text{<s>}\mathcal{V}^*\text{</s>}) \rightarrow \mathbb{R}^+$ a **global dissimilarity function**

$\ell(x,y)$ small when x and y are "similar"

- $\ell([w_{[1:T]}],[v_{[1:S]}]) = 1 - \mathbb{I}([w_{[1:T]}] = [v_{[1:S]}])$
  one-hot dissimilarity, all (non identical) pairs of sequences have $\ell = 1$
- $\ell([w_{[1:T]}, v_{[1:S]}]) = 1 - \mathrm{NED}([w_{[1:T]}],[v_{[1:S]}])$
  normalized edit distance, normalized minimum number of edits from $w_{[1:T]}$ to $v_{[1:S]}$
- $\ell([w_{[1:T]}],[v_{[1:S]}]) = 1 - \mathrm{BLEU}([w_{[1:T]}],[v_{[1:S]}])$
  reference based metrics - n-gram overlap (BLEU, METEOR for MT, Rouge for summarization)
- $\ell([w_{[1:T]}],[v_{[1:S]}]) = -\cos(\mathrm{Emb}([w_{[1:T]}]),\mathrm{Emb}([v_{[1:S]}]))$:
  cosine dissimilarity in embedding space, generalize to neural metrics (BLEURT, BertScore, COMET) [Suzgun et al., 2023]

Minimum Bayes Risk Decoding

Main idea

For fixed $[w_{[1:t]}]$, the risk of $[w_{[1:T]}]$

$$\begin{aligned} R([w_{[1:T]}]) &= \mathbb{E}_{S,[v_{[1:S]}]\sim P}(\ell([w_{[1:T]}],[v_{[1:S]}))]) \\ &= \sum_{[v_{[1:S]}]} \mathrm{P}(v_{[1:S]})\ell([w_{[1:T]}],[v_{[1:S]}]) \end{aligned}$$

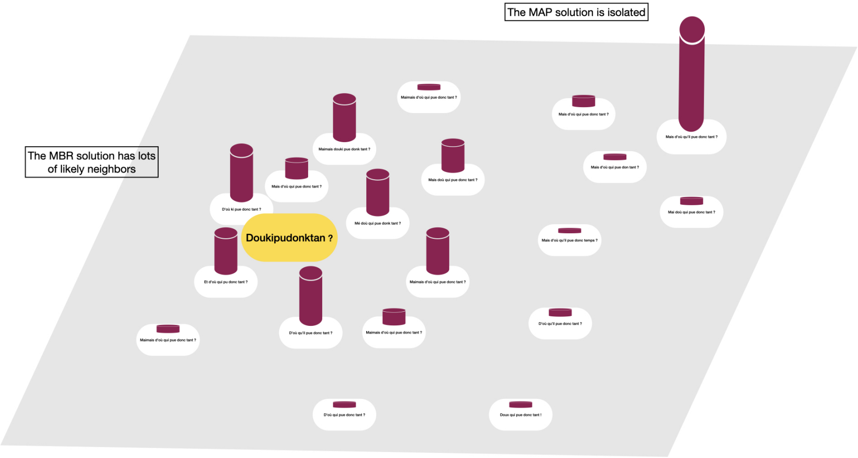
Minimum Bayes Risk decoding seeks

$$\begin{aligned} [w^*_{[1:T^*]}] &= \operatorname*{argmin}_{T,[w_{[1:T]}]} R([w_{[1:T]}]) \\ &= \operatorname*{argmin}_{T,[w_{[1:T]}]} \mathbb{E}_{S,[v_{[1:S]}]\sim P}(\ell([w_{[1:T]}],[v_{[1:S]}])) \\ &= \operatorname*{argmin}_{T,[w_{[1:T]}]} \sum_{S,[v_{[1:S]}]} \mathrm{P}(v_{[1:S]})\ell([w_{[1:T]}],[v_{[1:S]}]) \end{aligned}$$

The optimal sequence is (on average) the closest to all other sequences

Minimum Bayes Risk Decoding

Intuition: why is MBR is a good idea ?



Minimum Bayes Risk Decoding

Intuition: why is MBR is a good idea ?

0- the mode ($\operatorname{argmax} \mathrm{P}([w_{[1:T]}] \mid \boldsymbol{\theta})$) *may be anomalous and risky [Eikema and Aziz, 2020]*

1- If likely solutions (high $\mathrm{P}([w_{[1:T]}] \mid \boldsymbol{\theta})$*) have a good quality, being close to many good solutions* ($[w^*_{[1:T]}]$) *is also likely to have a good quality [smoothness of search space]*

2- For the one-hot dissimilarity: $\ell([w_{[1:T]}], [v_{[1:S]}]) = 1 - \mathbb{I}([w_{[1:T]}] = [v_{[1:S]}])$,

$$\mathbb{E}_{S,[v_{[1:S]}]\sim P}(\ell([w_{[1:T]}], [v_{[1:S]}])) = \sum_{[v_{[1:S]}] \neq [w_{[1:T]}]} \mathrm{P}([v_{[1:S]}] \mid \boldsymbol{\theta})$$

$$= 1 - \mathrm{P}([w_{[1:T]}] \mid \boldsymbol{\theta})$$

Minimizing the risk maximizes the model probability: back to MAP !

3- The MAP maximizes a proxy quality score $\mathrm{P}(w_{[1:t]} \mid \boldsymbol{\theta})$*, MBR directly optimizes the true metric* $\ell()$ *instead*

See also the motivations of Bertsch et al. [2023].

Minimum Bayes Risk Decoding

Theory and Practice of MBR

Two sources of intractability

$$[w^*_{[1:T]}] = \underset{T,[w_{[1:T]}]}{\operatorname{argmin}} \sum_{S,[v_{[1:S]}]} \mathrm{P}([v_{[1:s]}])\ell([w_{[1:T]}],[v_{[1:S]}])$$

1. $\operatorname{argmin}_{T,[w_{[1:T]}]}$: argmin in a very very large set
2. $\mathbb{E}_{S,v_{[1:S]}\sim P}() = \sum_{S,[v_{[1:S]}]}$: $\sum$ over many many terms

Minimum Bayes Risk Decoding

Theory and Practice of MBR

Two sources of intractability

$$[w^*_{[1:T]}] = \underset{T,[w_{[1:T]}]}{\operatorname{argmin}} \sum_{S,[v_{[1:S]}]} \mathrm{P}([v_{[1:s]}])\ell([w_{[1:T]}],[v_{[1:S]}])$$

1. $\operatorname{argmin}_{T,[w_{[1:T]}]}$: argmin in a very very large set
2. $\mathbb{E}_{S,v_{[1:S]}\sim P}() = \sum_{S,[v_{[1:S]}]}$: $\sum$ over many many terms

Two practical remedies

1. argmin in a very very large set $\Rightarrow$ restrict search to $\mathcal{W}_s$
2. $\sum$ over many many terms $\Rightarrow$ replace $\mathbb{E}()$ by Monte-Carlo approximation of size $|\mathcal{W}_{MC}|$

$$[w^*_{[1:t]}] = \underset{T,[w_{[1:T]}]\in\mathcal{W}_s}{\operatorname{argmin}} \sum_{[v_{[1:S]}]\in\mathcal{W}_{MC}} \ell([w_{[1:T]}],[v_{[1:S]}])$$

Minimum Bayes Risk Decoding

MBR: a meta-generation algorithm

$\ell()$: Dissimilarity ℓ, model $\mathrm{P}(W \mid \mathbf{C}; \boldsymbol{\theta})$

1: $\mathcal{W}_{MC} \leftarrow$ **generate**$(\mathrm{P}(\mid \mathbf{C}; \boldsymbol{\theta}), N, \dots)$
2: $\mathcal{W}_S \leftarrow$ **generate**$(\mathrm{P}(\mid \mathbf{C}; \boldsymbol{\theta}), M, \dots)$
3: $mins \leftarrow +\infty$
4: **for** $[w_{[1:T]}] \in \mathcal{W}_S$ **do**
5: $s \leftarrow 0$, $mbr \leftarrow$ <s></s>
6: **for** $[v_{[1:S]}] \in \mathcal{W}_{MC}$ **do**
7: $s \leftarrow s + \ell([w_{[1:T]}], [v_{[1:S]}])$
8: **end for**
9: **if** $s < mins$ **then**
10: $mins \leftarrow s$, $mbr \leftarrow [w_{[1:T]}]$
11: **end if**
12: **end for**
13: **return**$(mins, mbr)$

- **generate** 1: is for MC estimates: prefer sampling with replacement, unbiased (ancestral)
- **generate** 2: is to identify promising solutions: prefer beam-seach, if possible diverse
- Alternative for $\mathcal{W}_S$: reuse $\mathcal{W}_{MC} \Rightarrow$ back to reranking
- Alternative for $\mathcal{W}_S$: use multiple models, multiple checkpoints, multiple prompts, etc.
- Run-time is sampling time + $O(MN)$; larger N yields better MC estimates; larger M yields better exploration

Minimum Bayes Risk Decoding

MBR: a meta-generation algorithm

$\ell()$: Dissimilarity ℓ, model $\mathrm{P}(W \mid \mathbf{C}; \boldsymbol{\theta})$

$\mathcal{W}_{MC} \leftarrow$ **generate**$(\mathrm{P}(\mid \mathbf{C}; \boldsymbol{\theta}), N, \ldots)$
$\mathcal{W}_{S} \leftarrow$ **generate**$(\mathrm{P}(\mid \mathbf{C}; \boldsymbol{\theta}), M, \ldots)$
$mins \leftarrow +\infty$
for $[w_{[1:T]}] \in \mathcal{W}_S$ **do**
    $s \leftarrow 0$, $mbr \leftarrow$ <s></s>
    for $[v_{[1:S]}] \in \mathcal{W}_{MC}$ **do**
        $s \leftarrow s + \ell([w_{[1:T]}], [v_{[1:S]}])$
    end for
    if $s < mins$ **then**
        $mins \leftarrow s$, $mbr \leftarrow [w_{[1:T]}]$
    end if
end for
return$(mins, mbr)$

- **generate** 1: is for MC estimates: prefer sampling with replacement, unbiased (ancestral)
- **generate** 2: is to identify promising solutions: prefer beam-seach, if possible diverse

- Alternative for $\mathcal{W}_S$: reuse $\mathcal{W}_{MC} \Rightarrow$ back to reranking
- Alternative for $\mathcal{W}_S$: use multiple models, multiple checkpoints, multiple prompts, etc.

- Run-time is sampling time $+ O(MN)$; larger N yields better MC estimates; larger M yields better exploration

Better Searching for Good Solutions with MCTS

Monte-Carlo Tree Search [Kocsis and Szepesvári, 2006]

The problem with output reward $R([w_{[1:T]}], C)$

- searching with $\mathrm{P}([w_{[1:T]}] \,|\, \mathbf{C}; \boldsymbol{\theta})$ may yield poor / unappropriate solutions
- ensemble-based methods (best-out-of-N, MBR) require multiple inferences, no garantee of improvement

MCTS delivers solutions with a high output reward, based on a estimates of $R([w_{[1:T]}], \mathbf{C})$ for partial sequences $[w_{[1:t]}]$.

Better Searching for Good Solutions with MCTS

Monte-Carlo Tree Search [Kocsis and Szepesvári, 2006]

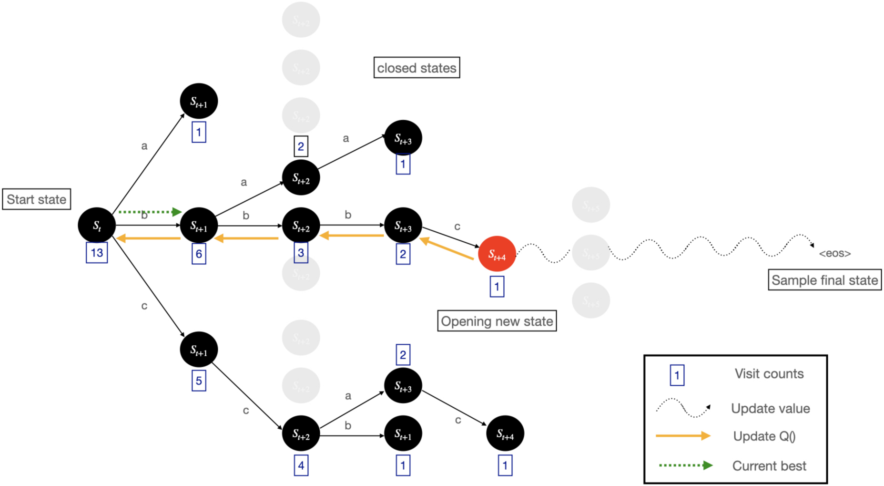
Concept and Terminology (adapted from RL / POMDP)

- state: $S_t \Leftrightarrow$ context + current prefix $\mathbf{C}, [w_{[1:t]}$; $\mathcal{S}$ is the set of states (prefixes).
  States can be complete (w_t = </s>) or incomplete.
- actions: pick next possible token $w_{t+1} \in \mathcal{V}$
- using action w in state S_t: yields new state $S_t \oplus w \equiv w_{[1:t+1]} = w_{[1:t]}w$
- policy $\pi_{\boldsymbol{\theta}}$: $\mathcal{S}_t \rightarrow \mathcal{V}$; next action selection rule. For instance:
  - $\pi_{\boldsymbol{\theta}}(S_t) = \operatorname{argmax}_w \mathrm{P}(w|S_t;\boldsymbol{\theta})$: greedy policy (deterministic)
  - $\pi_{\boldsymbol{\theta}}(S_t) = w \sim \mathrm{P}(w|S_t;\boldsymbol{\theta})$: sampling policy (non-deterministic) - also top-k, top-p etc.
- value (of a state, given policy): $v_\pi : \mathcal{S} \rightarrow \mathbb{R}$; $v_\pi(S_t)$ estimates the best score $F()$ attainable from S_t.

Use state values to obtain MC samples of local subtrees that guide the generation policy towards leaf nodes with large scores.

Better Searching for Good Solutions with MCTS

Monte-Carlo Tree Search [Kocsis and Szepesvári, 2006]



Generating one token with MCTS

Better Searching for Good Solutions with MCTS

Monte-Carlo Tree Search [Kocsis and Szepesvári, 2006]

procedure MCTS(K : int)
  $S \leftarrow S_0 (\equiv \mathbf{C}, \texttt{<s>})$
  while $!\,\mathrm{complete}(S)$ **do**
    for K iterations **do**
      MCTS-Explore(S)
    end for
    $w^* \leftarrow \mathrm{argmax}_{w \in \mathcal{V}}\, \mathrm{cnt}(S \oplus w)$
    $S \leftarrow S \oplus w^*$
  end while
  return S
end procedure

procedure PUCT-SCORE(S,w)
  $U \leftarrow Q(S \oplus w)$
  $U \leftarrow U + c_{puct}\, \mathrm{P}(w \mid S; \boldsymbol{\theta}) \frac{\sqrt{\mathrm{cnt}(S)}}{1+\mathrm{cnt}(S,w)}$
  return U
end procedure

procedure MCTS-EXPLORE(S : state)
  $\mathrm{cnt}(S) \leftarrow \mathrm{cnt}(S) + 1$
  $w^* \leftarrow \mathrm{argmax}_w$ PUCT-Score(S, w)
  if $\mathrm{open}(S \oplus w^*) \wedge !\,\mathrm{complete}(S \oplus w^*)$ **then**
    $Q \leftarrow$ MCTS-Explore($S \oplus w^*$)
    $Q(S) \leftarrow \max(Q(S), Q)$
  else if $!\,\mathrm{complete}(S \oplus w^*)$ **then**
    $\mathrm{open}(S \oplus w^*) \leftarrow$ **true**
    estimate$v_\pi(S \oplus w^*)$
    $Q \leftarrow \mathrm{argmax}_{\mathrm{open}(S \oplus w)} v_\pi(S \oplus w)$
      ▷ aggregate with max or avg
  else
    $Q \leftarrow F(S \oplus w^*)$
  end if
  return Q
end procedure

PUCT-SCORE **trades-off high scores** (Q) **and likely, unvisited states**

Better Searching for Good Solutions with MCTS

Monte-Carlo Tree Search [Kocsis and Szepesvári, 2006]

Computing state values

In state S, how to estimate $v_\pi(S)$?

1. **sampling based**: apply sampling using roll-out policy $\mathrm{P}(\,|S;\boldsymbol{\theta})$ (e.g. [Chaffin et al., 2022])
   return underestimates, as costly as a complete generation for each simulation.
2. **learning based**: learns to predict $v_\pi(S;\lambda)$ using an auxilary network [Leblond et al., 2021]
   get complete (complete) samples $[w_{[1:T]}]$ and associated scores; learns to predict scores for incomplete states; this can be hard.
3. **repurpose** value networks trained with reinforcement learning (PPO) during LLM alignment step [Liu et al., 2024]
   show improvements even when using PPO-tuned language models.

Better Searching for Good Solutions with MCTS

A compute-effective approach: REBASE [Wu et al., 2025]

Motivations

- MCTS empirically dominated by simpler alternatives eg., best-of-N
- exploration costly and inefficient
- main idea: use trained reward $\hat{R}(S;\lambda)$ to improve search
- return a target number N of solutions $\Rightarrow$ best-of-N
- $\mathrm{sample}(S,K)$ samples K times $w \sim \mathrm{P}(W \mid S;\boldsymbol{\theta})$, returns sample

procedure REBASE(N : int)
    $S \leftarrow S_0(\equiv \mathbf{C}, \texttt{<s>})$
    $\mathcal{C} \leftarrow \varnothing, t \leftarrow 1$
    $\mathcal{S}_1 \leftarrow \mathrm{sample}(S, M(S))$
    while $|\mathcal{C}| < N$ **do**
        for $S \in \mathcal{S}_t$ **do**
            if $\mathrm{complete}(S)$ **then**
                $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$
                $N \leftarrow N - 1$
            end if
        end for
        $\mathcal{S}_{t+1} \leftarrow \varnothing$
        for $S \in \mathcal{S}_t \setminus \mathcal{C}$ **do**
            $M(S) \propto (N - |\mathcal{C}|) \exp \hat{R}(S;\lambda)$
            $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{t+1} \cup \mathrm{sample}(S, M(S))$
        end for
        $t \leftarrow t + 1$
    end while
end procedure

Local search and reformulation

Principles of local search

Intuition:

1. **generate** an initial solution $[w^{(0)}_{[1:T]}]$,
2. **hill-climb** neighour solutions guided by output reward model $R(,)$

neighbours are defined by simple operators: replace a word, insert / delete a word, swap two words, etc

Generating Texts Non-Auto-Regressively

Parallel Text Generation

- standard left-to-right / right-to-left decoding is slow
- decoding in arbitrary order does not solve this [Welleck et al., 2019]
- alternative: generate multiple words simultaneously

How ? **Parallel Unmasking**.

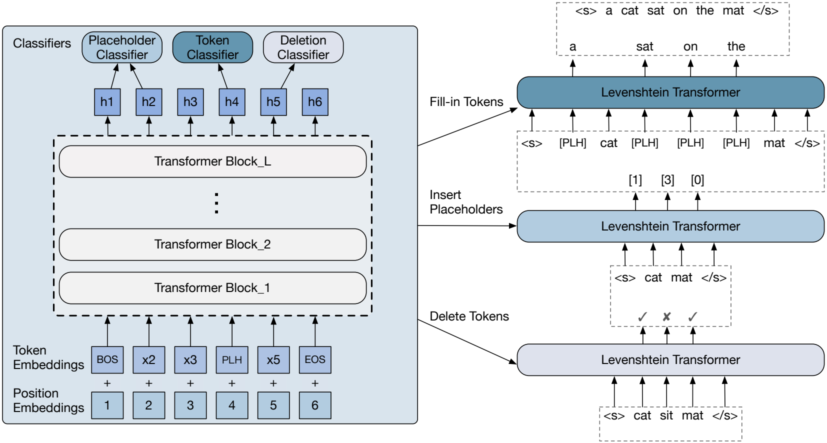
Generating Texts Non-Auto-Regressively

Mask-Predict by Ghazvininejad et al. [2019]

1: **procedure** MASK-PREDICT
   Input: $\mathbf{C}$: Context, T: Target Length
   Output: Generated Sequence
2: $w_0 = [\,, w_{T+1} =], \forall t \in [1:T], w_t \sim \mathrm{Unif}(\mathcal{V})$
3: **for** K iterations **do**
4: ToMask $\leftarrow$ top-k$_t(-\log \mathrm{P}(w_t \,|\, C, w_{-t}; \boldsymbol{\theta}))$
5: **for** ($t \in$ ToMask) **do**
6: $w_t \leftarrow$ MASK
7: **end for**
8: **for** ($t \in$ ToMask) **do**
9: $w_t \leftarrow$ **unmask**(w_t)
10: **end for**
11: **end for**
12: **return**$([w_{[1:T]}])$
13: **end procedure**

- a better initialization samples independently given $\mathbf{C}$
- **unmask** (l9) can be argmax or obtained via sampling
- T is unknown ? Generate with multiple lengths in parallel
- masking and generation can be performed in parallel
- K and k trade-off speed and fluency
- recover Gibbs sampling with $k = 1$ and iterative masking (instead of top-k)

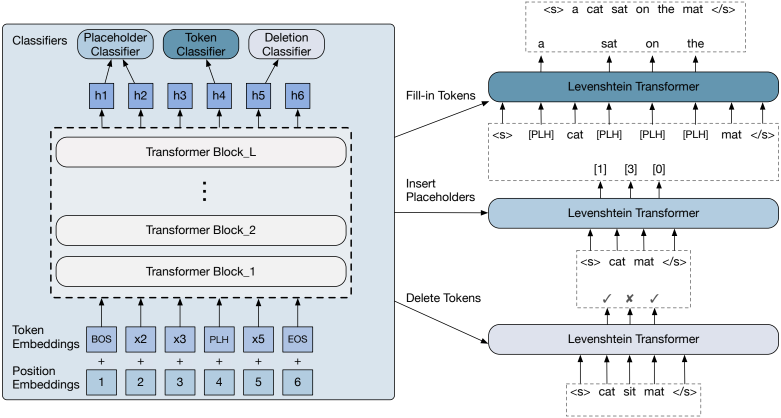
Generating Texts Non-Auto-Regressively



The Levenvshtein Transformer [Gu et al., 2019]

- "Multimodality" problem and solutions (latent alignments, KD, etc) [Xiao et al., 2023]
- Mostly used for standard translation tasks (also: term constraints [Xu and Carpuat, 2021])
- Decoding starts from scratch or initial solution [Xu et al., 2023]

Generating Texts Non-Auto-Regressively



LevT uses

3 classifiers to predict **D**eletions and **I**nsertions

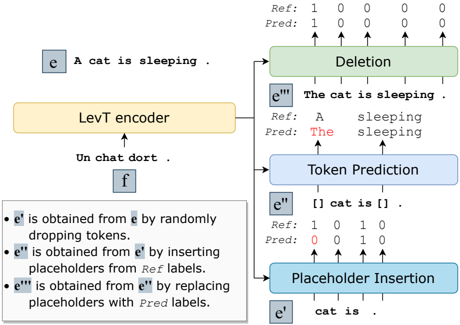
- D deletion classifier predicts $y \in \{0, 1\}$
- I placeholder classifier predicts $y \in [0 : N]$
- I token classifier predicts $y \in [1 : |V|]$

Generating Texts Non-Auto-Regressively

Training with parallel sentences (**f**, **e**)

1. erase random words in **e**, yields **e'**
2. train placeholder & token prediction with samples (**e'**, **e''**), (**e''**, **e**)
3. generate output **e'''**
4. train deletion prediction with (**e**, **e'''**)

Decode with $\mathbf{e}^{(0)} = [e_{[1:T]}]$:
PLH - TOK - DEL + repeat in iterative refinement



Dual Policy learning with:

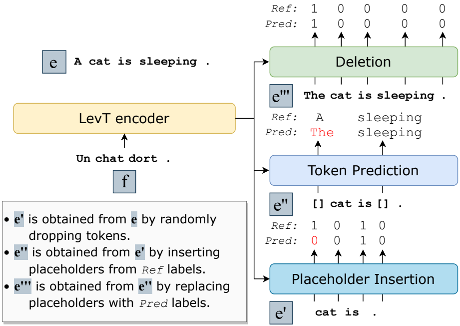
- roll-in policy π_{ins} for [I]nsertion: empty string or random deletion from **e**
- roll-in policy π_{del} for [D]eletion: model's Insertions
- expert policy π^* from the optimal alignment ⇔ Edit Distance

Generating Texts Non-Auto-Regressively

Training with parallel sentences (**f**, **e**)

1. erase random words in **e**, yields **e**’
2. train placeholder & token prediction with samples (**e**’, **e**”), (**e**”, **e**)
3. generate output **e**”’
4. train deletion prediction with (**e**, **e**”’)

Decode with $\mathbf{e}^{(0)} = [e_{[1:T]}]$:
PLH - TOK - DEL + repeat in iterative refinement



Dual Policy learning with:

- roll-in policy π_{ins} for [I]nsertion: empty string or random deletion from **e**
- roll-in policy π_{del} for [D]eletion: model’s Insertions
- expert policy π^* from the optimal alignment ⇔ Edit Distance

An effective model for NAR Machine Translation

Self-Refinements with Prompting

Require: input $\mathbf{C}_x$, model $\mathrm{P}(W \mid \mathbf{C}, \boldsymbol{\theta})$, prompts $\{\mathbf{C}_g, \mathbf{C}_f, \mathbf{C}_r\}$, stop condition $\mathrm{stop}()$

$S_0 = \mathbf{generate}(\pi^{\boldsymbol{\theta}}, \mathbf{C}_x)$ ▷ Initial generation
for iteration t $\in 0, 1, \ldots$ **do**
  $F_t = \mathbf{generate}(\pi^{\boldsymbol{\theta}}, \mathbf{C}_x \oplus \mathbf{C}_f(S_t))$ ▷ Feedback
  if $\mathrm{stop}(F_t, t)$ **then** ▷ Stop condition
    break
  else
    $S_{t+1} = \mathbf{generate}(\pi^{\boldsymbol{\theta}}, \mathbf{C}_x \oplus \mathbf{C}_\mathbf{r}(S_0 \oplus F_0 \cdots \oplus \mathbf{C}_f(S_t)))$ ▷ Refine
  end if
end for
return S_t

[Madaan et al., 2023]

- prompts are task-dependent
- prompts can include few-shot examples

Self-Refinements with Prompting

```
I have some code. Can you give one suggestion to improve readability.
Don't fix the code, just give a suggestion.

{code}
```

Prompting for Feedback C_F - Readability task

```
I have some code. Can you give one suggestion to improve readability.
Don't fix the code, just give a suggestion.

{code}

{suggestion}

Now fix the code.
```

Prompting for Self-Refinement C_R - Readability task

Conclusions

Generation is Tricky

- implementation details matter in generation
- generation parameters matter both for quality and speed
- there is much more than temperature, top-k and top-p

A call for better documenting text generation parameters in evaluations

Generation is not Solved

- generation with refinement and self-critics
- training multi-step generation and planing
- finding compute optimal generation policies?

Conclusions

Generation is Tricky

- implementation details matter in generation
- generation parameters matter both for quality and speed
- there is much more than temperature, top-k and top-p

A call for better documenting text generation parameters in evaluations

Generation is not Solved

- generation with refinement and self-critics
- training multi-step generation and planing
- finding compute optimal generation policies?

Part V

References

Bibliography I

Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. Mirostat: A neural text decoding algorithm that directly controls perplexity. In *Proc. International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=W1G1JZEIy5_.

Amanda Bertsch, Alex Xie, Graham Neubig, and Matthew Gormley. It's MBR all the way down: Modern generation techniques through the lens of minimum Bayes risk. In Yanai Elazar, Allyson Ettinger, Nora Kassner, Sebastian Ruder, and Noah A. Smith, editors, *Proceedings of the Big Picture Workshop*, pages 108–122, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.bigpicture-1.9. URL https://aclanthology.org/2023.bigpicture-1.9.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, Julen Etxaniz, Benjamin Fattori, Jessica Zosa Forde, Charles Foster, Jeffrey Hsu, Mimansa Jaiswal, Wilson Y. Lee, Haonan Li, Charles Lovering, Niklas Muennighoff, Ellie Pavlick, Jason Phang, Aviya Skowron, Samson Tan, Xiangru Tang, Kevin A. Wang, Genta Indra Winata, François Yvon, and Andy Zou. Lessons from the Trenches on Reproducible Evaluation of Language Models, May 2024.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Jennifer C. Lai, and Robert L. Mercer. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18 (1):31–40, 1992.

Bibliography II

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Antoine Chaffin, Vincent Claveau, and Ewa Kijak. PPL-MCTS: Constrained textual generation through discriminator-guided MCTS decoding. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2953–2967, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.215. URL https://aclanthology.org/2022.naacl-main.215.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023. URL https://arxiv.org/abs/2302.01318.

Bibliography III

Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Th6NyL07na.

Bryan Eikema and Wilker Aziz. Is MAP Decoding All You Need? The Inadequacy of the Mode in Neural Machine Translation. In Donia Scott, Nuria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 4506–4520, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.398. URL https://aclanthology.org/2020.coling-main.398.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Matthew Finlayson, John Hewitt, Alexander Koller, Swabha Swayamdipta, and Ashish Sabharwal. Closing the curious case of neural text degeneration. In *Proceedings of the International Conference on Representation Learning*, ICLR, 2024. URL https://openreview.net/forum?id=dONpC9GL1o.

Bibliography IV

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-Predict: Parallel Decoding of Conditional Masked Language Models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633.

Jiatao Gu, Changhan Wang, and Junbo Zhao. Levenshtein Transformer. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 11179–11189, 2019.

John Hewitt, Christopher Manning, and Percy Liang. Truncation sampling as language model desmoothing. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3414–3427, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.249. URL https://aclanthology.org/2022.findings-emnlp.249.

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.

Bibliography V

Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. A call for clarity in beam search: How it works and when it stops. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 77–90, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.7.

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

Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45375-X. doi: 10.1007/11871842_29.

Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The Gumbel-top-k trick for sampling sequences without replacement. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3499–3508. PMLR, 2019-06-09/2019-06-15.

Bibliography VI

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, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.424.

Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislar, Lespiau Jean-Baptiste, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. Machine translation decoding beyond beam search. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8410–8434, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.662. URL https://aclanthology.org/2021.emnlp-main.662.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Bibliography VII

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, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 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.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. Don't throw away your value model! Generating more preferable text with Value-Guided Monte-Carlo Tree Search decoding. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/pdf?id=kh9Zt2Ldmn.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Clara Meister and Ryan Cotterell. Language model evaluation beyond perplexity. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5328–5339, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.414. URL https://aclanthology.org/2021.acl-long.414.

Bibliography VIII

Clara Meister, Ryan Cotterell, and Tim Vieira. If beam search is the answer, what was the question? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185. Association for Computational Linguistics, 2020a. doi: 10.18653/v1/2020.emnlp-main.170. URL https://aclanthology.org/2020.emnlp-main.170.

Clara Meister, Tim Vieira, and Ryan Cotterell. Best-first beam search. *Transactions of the Association for Computational Linguistics*, 8:795–809, 2020b. doi: 10.1162/tacl_a_00346.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *Transactions of the Association for Computational Linguistics*, 11:102–121, 2023. doi: 10.1162/tacl_a_00536. URL https://aclanthology.org/2023.tacl-1.7.

Kenton Murray and David Chiang. Correcting length bias in neural machine translation. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Christof Monz, Matteo Negri, Aurélie Névéol, Mariana Neves, Matt Post, Lucia Specia, Marco Turchi, and Karin Verspoor, editors, *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 212–223, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-6322. URL https://aclanthology.org/W18-6322.

Bibliography IX

Felix Stahlberg and Bill Byrne. On NMT search errors and model errors: Cat got your tongue? In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1331. URL https://aclanthology.org/D19-1331.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. A contrastive framework for neural text generation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 21548–21561. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/871cae8f599cb8bbfcb0f58fe1af95ad-Paper-Conference.pdf.

Mirac Suzgun, Luke Melas-Kyriazi, and Dan Jurafsky. Follow the wisdom of the crowd: Effective text generation via minimum Bayes risk decoding. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4265–4293, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.262. URL https://aclanthology.org/2023.findings-acl.262.

Sean Welleck, Kianté Brantley, Hal Daumé Iii, and Kyunghyun Cho. Non-monotonic sequential text generation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6716–6726. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/welleck19a.html.

Bibliography X

Sean Welleck, Ilia Kulikov, Jaedeok Kim, Richard Yuanzhe Pang, and Kyunghyun Cho. Consistency of a recurrent language model with respect to incomplete decoding. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5553–5568, Online, November 2020a. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.448. URL https://aclanthology.org/2020.emnlp-main.448.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020b. URL https://openreview.net/forum?id=SJeYe0NtvH.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VNckp7JEHn.

Yisheng Xiao, Lijun Wu, Junliang Guo, Juntao Li, Min Zhang, Tao Qin, and Tie-Yan Liu. A survey on non-autoregressive generation for neural machine translation and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(10):11407–11427, October 2023. ISSN 0162-8828. doi: 10.1109/TPAMI.2023.3277122. URL https://doi.org/10.1109/TPAMI.2023.3277122.

Bibliography XI

Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 41618–41650. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/81fde95c4dc79188a69ce5b24d63010b-Paper-Conference.pdf.

Jitao Xu, Josep Crego, and François Yvon. Integrating translation memories into non-autoregressive machine translation. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*, pages 1326–1338, Dubrovnik, Croatia, 2023. URL https://aclanthology.org/2023.eacl-main.96.

Weijia Xu and Marine Carpuat. EDITOR: An Edit-Based Transformer with Repositioning for Neural Machine Translation with Soft Lexical Constraints. *Transactions of the Association for Computational Linguistics*, 9:311–328, 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00368.