CS 489: Introduction to Natural Language Processing

Lecture 15: Decoding Strategies

Instructor: Freda Shi

November 10, 2026



UNIVERSITY OF WATERLOO

Today: From Probabilities to Task Outputs

Last lecture: how to evaluate a language model and its generated text.

Today we keep the model's weights fixed and ask:

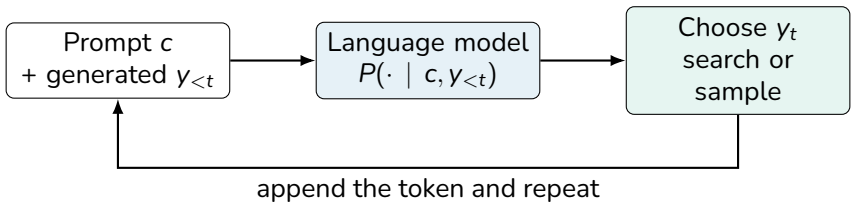
1. How do we choose a continuation from next-token probabilities?
2. Why can high-probability text be repetitive or generic?
3. How do sampling and repetition controls change the output?

The Decoding Problem

Given a prompt c and generated tokens $y_{<t}$, the model supplies

$$P(y_t = v \mid c, y_{<t}), \qquad v \in \mathcal{V}.$$

A **decoding strategy** uses this distribution to choose the next token.

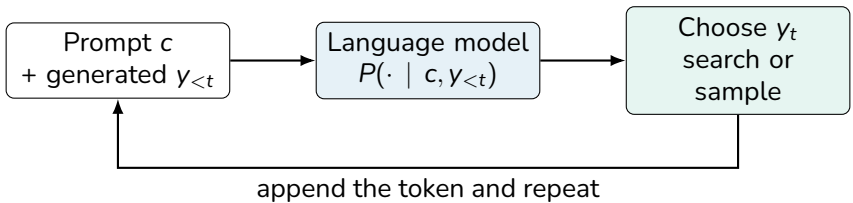


The Decoding Problem

Given a prompt c and generated tokens $y_{<t}$, the model supplies

$$P(y_t = v \mid c, y_{<t}), \qquad v \in \mathcal{V}.$$

A **decoding strategy** uses this distribution to choose the next token.



Stop when the model emits $\langle \text{eos} \rangle$ or a length limit is reached. The model supplies probabilities; the decoder determines how we use them.

Greedy Decoding

At each step, select a highest-probability token:

$$y_t = \arg\max_{v \in \mathcal{V}} P(v \mid c, y_{<t}).$$

Toy model: after the prompt `The`, the next token is

Token	`cat`	`dog`
Probability	0.60	0.40

Greedy Decoding

At each step, select a highest-probability token:

$$y_t = \arg\max_{v \in \mathcal{V}} P(v \mid c, y_{<t}).$$

Toy model: after the prompt `The`, the next token is

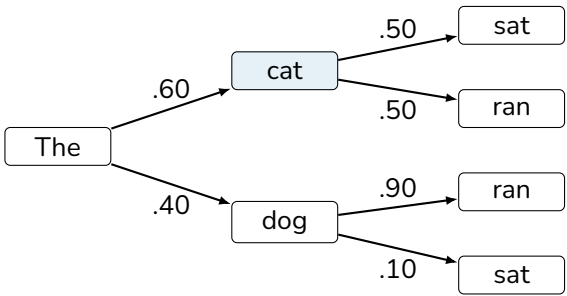
Token	`cat`	`dog`
Probability	0.60	0.40

Greedy selects `cat` and then queries the model again using `The cat`.

- One active prefix; selecting the maximum costs $O(|\mathcal{V}|)$ after the model forward pass.
- With fixed model outputs and tie-breaking, the result is deterministic.

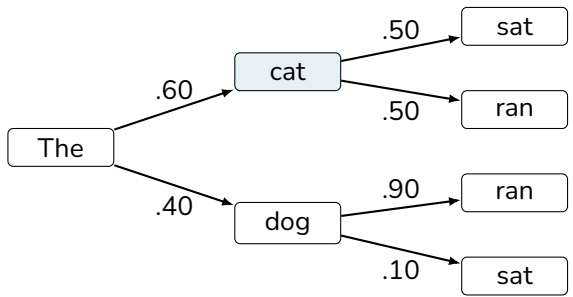
A Local Maximum Can Miss the Best Sequence

Continue the toy model for two generated tokens, then emit ⟨eos⟩ with probability 1.



A Local Maximum Can Miss the Best Sequence

Continue the toy model for two generated tokens, then emit ⟨eos⟩ with probability 1.



$P = 0.60 \times 0.50 = 0.30$

$P = 0.60 \times 0.50 = 0.30$

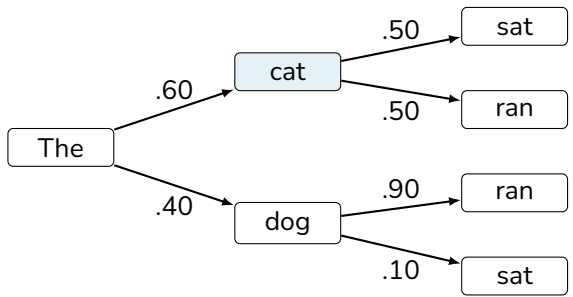
$P = 0.40 \times 0.90 = \mathbf{0.36}$

$P = 0.40 \times 0.10 = 0.04$

Greedy gets probability 0.30 whichever tie it selects after `cat`.

A Local Maximum Can Miss the Best Sequence

Continue the toy model for two generated tokens, then emit ⟨eos⟩ with probability 1.



$P = 0.60 \times 0.50 = 0.30$

$P = 0.60 \times 0.50 = 0.30$

$P = 0.40 \times 0.90 = \mathbf{0.36}$

$P = 0.40 \times 0.10 = 0.04$

Greedy gets probability 0.30 whichever tie it selects after `cat`. The highest-probability continuation is `dog ran` with probability 0.36.

Beam Search

Keep up to B active prefixes so that several early choices can survive.

1. Start with **one** prefix, the prompt, with log-score 0.
2. Expand each active prefix by possible next tokens and add their log-probabilities.
3. Keep the B highest-scoring expansions; move any ending in $\langle \text{eos} \rangle$ to a completed list.
4. Repeat until no active prefixes remain or a length limit is reached.
5. Return the best completed sequence; if none completed, return the best unfinished prefix.

Beam Search

Keep up to B active prefixes so that several early choices can survive.

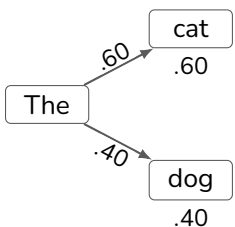
1. Start with **one** prefix, the prompt, with log-score 0.
2. Expand each active prefix by possible next tokens and add their log-probabilities.
3. Keep the B highest-scoring expansions; move any ending in $\langle \text{eos} \rangle$ to a completed list.
4. Repeat until no active prefixes remain or a length limit is reached.
5. Return the best completed sequence; if none completed, return the best unfinished prefix.

Pruning makes this an **approximate search**: a discarded prefix may lead to the best sequence. Each step considers up to $B|\mathcal{V}|$ extensions, plus model and selection costs.

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.

Token 1 Token 2 Token 3



Token 1: expand the prompt

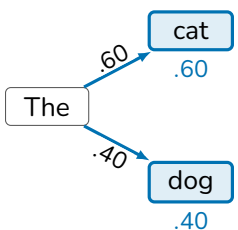
Prefix	Probability
`cat`	.60
`dog`	.40

Which prefixes survive with $B = 2$?

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.

Token 1 Token 2 Token 3



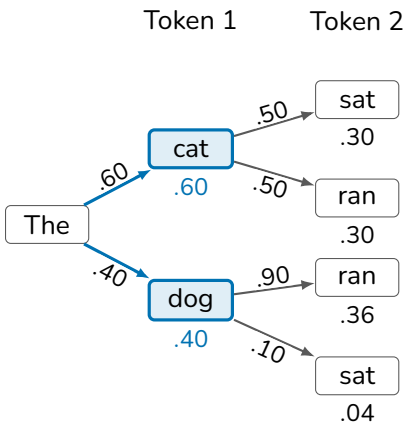
Beam after token 1

Keep	Probability
`cat`	.60
`dog`	.40

Keep both prefixes. Expand both at the next step.

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.



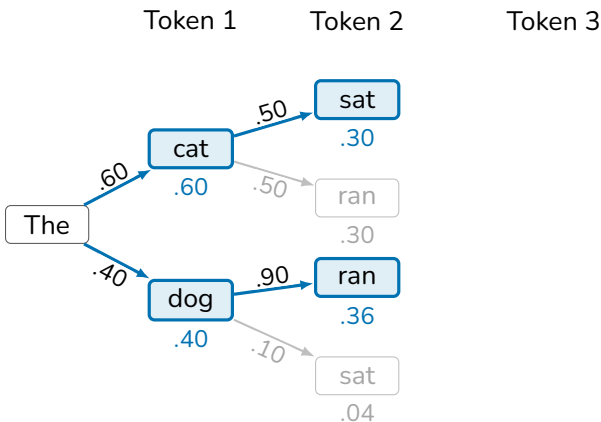
Token 2: expand both prefixes

Prefix	Probability
`cat sat`	.30
`cat ran`	.30
`dog ran`	.36
`dog sat`	.04

For `dog ran`:
$P = .40 \times .90 = .36$. Rank all four candidates together.

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.



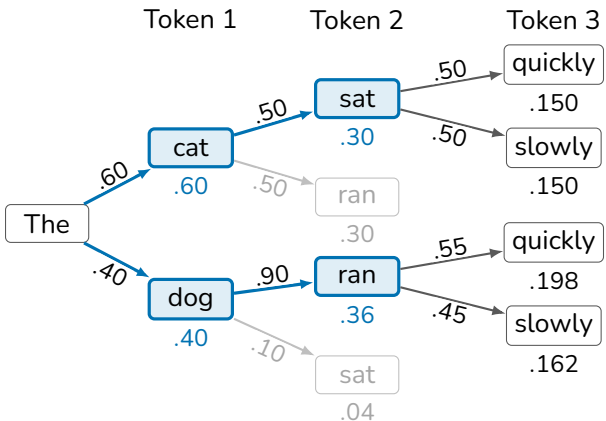
Beam after token 2

Keep	Probability
`dog ran`	.36
`cat sat`	.30

Break the .30 tie in favor of `cat sat`. Expand only these two prefixes next.

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.



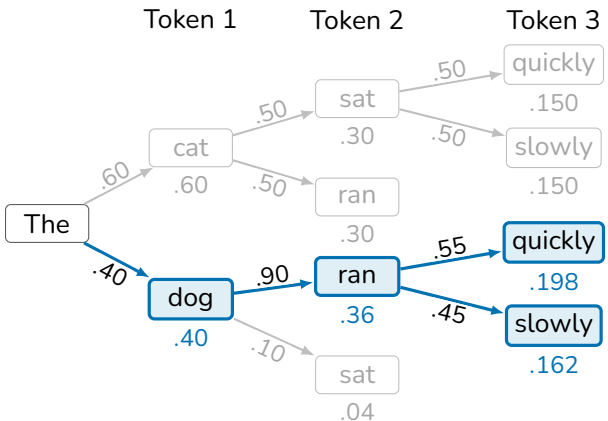
Token 3: expand the survivors

Prefix	Probability
`cat sat quickly`	.150
`cat sat slowly`	.150
`dog ran quickly`	.198
`dog ran slowly`	.162

Multiply each prefix score by the next-token probability.

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.



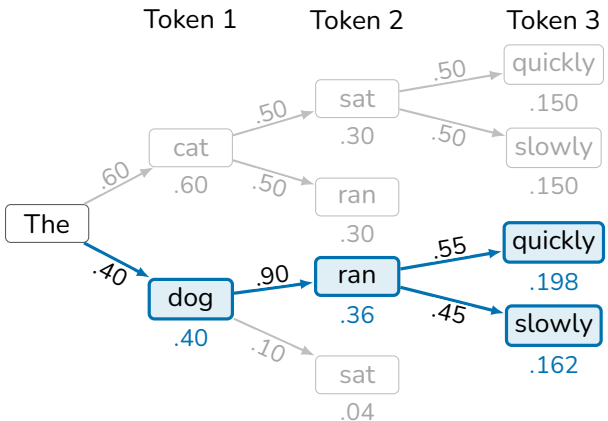
Beam after token 3

Keep	Probability
`dog ran quickly`	.198
`dog ran slowly`	.162

Both surviving prefixes extend `dog ran`. Keep the best two candidates overall.

Beam Search: Three Tokens ($B = 2$)

Toy model: generate three tokens after `The`.



Finish: emit ⟨eos⟩

Both prefixes end with probability 1, so their scores stay the same.

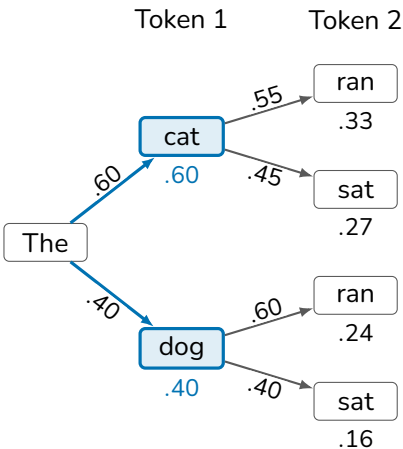
Return the highest-scoring sequence:

`The dog ran quickly`

$P = .40 \times .90 \times .55 = .198.$

A Beam-Search Counterexample ($B = 2$)

Beam search can also miss the highest-probability sequence.



After token 2

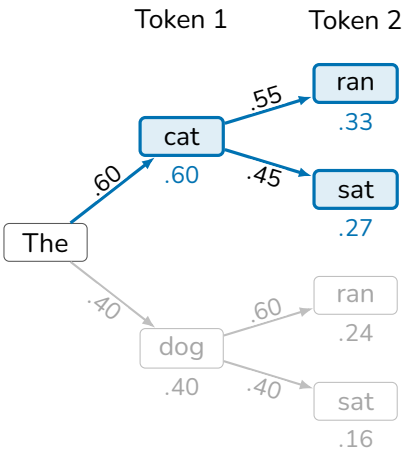
Prefix	Probability
`cat ran`	.33
`cat sat`	.27
`dog ran`	.24
`dog sat`	.16

Which two prefixes survive?

After token 3, ⟨eos⟩ has probability 1.

A Beam-Search Counterexample ($B = 2$)

Beam search can also miss the highest-probability sequence.



Keep the top two

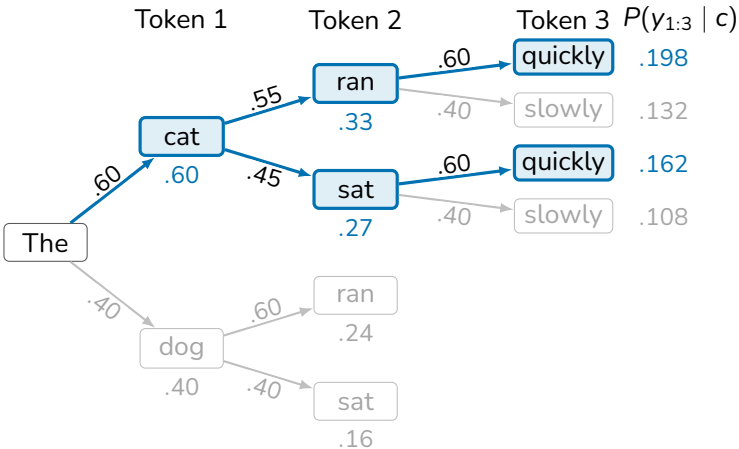
`cat ran`: .33
`cat sat`: .27

Both `dog` prefixes are pruned before token 3.

After token 3, ⟨eos⟩ has probability 1.

A Beam-Search Counterexample ($B = 2$)

Beam search can also miss the highest-probability sequence.



Best sequence found

`cat ran quickly`

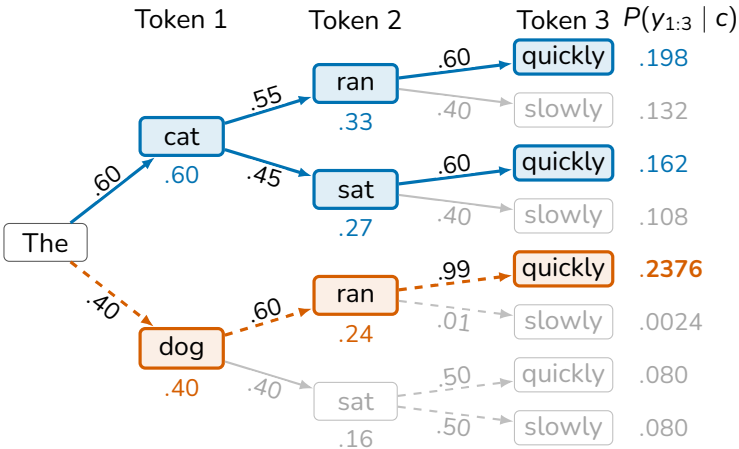
$P = .33 \times .60 = .198$.

The other retained sequence is `cat sat quickly`, with probability .162.

After token 3, ⟨eos⟩ has probability 1.

A Beam-Search Counterexample ($B = 2$)

Beam search can also miss the highest-probability sequence.



The pruned branch

`dog ran quickly`

$P = .24 \times .99 = .2376.$

$.2376 > .198$**.**

This branch was pruned before token 3. Its completion has the highest probability in the tree.

After token 3, ⟨eos⟩ has probability 1.

Scoring Completed Sequences

With a fixed prompt c, the raw log-score of a continuation is

$$L(y_{1:T} \mid c) = \sum_{t=1}^{T} \log P(y_t \mid c, y_{<t}).$$

Here T counts generated tokens, including $\langle \text{eos} \rangle$, and excludes the prompt.

Scoring Completed Sequences

With a fixed prompt c, the raw log-score of a continuation is

$$L(y_{1:T} \mid c) = \sum_{t=1}^{T} \log P(y_t \mid c, y_{<t}).$$

Here T counts generated tokens, including $\langle \text{eos} \rangle$, and excludes the prompt.

Every added log-probability is nonpositive, which can favor short outputs. A common heuristic is **length normalization**:

$$S(y_{1:T} \mid c) = \frac{L(y_{1:T} \mid c)}{T^{\alpha}}, \qquad 0 \le \alpha \le 1.$$

- $\alpha = 0$: raw sequence log-probability; $\alpha = 1$: average token log-probability.
- This changes the ranking objective and the appropriate stopping rule.

High Likelihood Does Not Guarantee Good Text

Holtzman et al. (2020) found repetition and generic output with maximization-based decoding in open-ended generation.

Illustrative symptoms:

- Repetition: “The city is a great place to live. The city is a great place to live.”
- Blandness: “I think that is a good idea. It would be good to do that.”

High Likelihood Does Not Guarantee Good Text

Holtzman et al. (2020) found repetition and generic output with maximization-based decoding in open-ended generation.

Illustrative symptoms:

- Repetition: “The city is a great place to live. The city is a great place to live.”
- Blandness: “I think that is a good idea. It would be good to do that.”

Repeating a phrase can make another repetition likely; search can amplify this loop.

High Likelihood Does Not Guarantee Good Text

Holtzman et al. (2020) found repetition and generic output with maximization-based decoding in open-ended generation.

Illustrative symptoms:

- Repetition: “The city is a great place to live. The city is a great place to live.”
- Blandness: “I think that is a good idea. It would be good to do that.”

Repeating a phrase can make another repetition likely; search can amplify this loop.

Why sample? Plausible continuations occupy many paths through the distribution. Sampling can explore more of them.

Simple Sampling

Sample each next token from the full model distribution:

$$y_t \sim P(\cdot \mid c, y_{<t}).$$

One toy distribution, reused in the next examples:

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02

Simple Sampling

Sample each next token from the full model distribution:

$$y_t \sim P(\cdot \mid c, y_{<t}).$$

One toy distribution, reused in the next examples:

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02

In a large vocabulary, many low-probability tokens can have substantial total mass. Sampling from a poorly modeled tail can produce incoherent continuations. We can **reshape** the probabilities or **truncate** the candidate set.

Temperature Reshapes the Distribution

Given logits z_v, sample from the temperature-adjusted distribution

$$Q_\tau(v) = \frac{\exp(z_v/\tau)}{\sum_{u \in \mathcal{V}} \exp(z_u/\tau)}, \qquad \tau > 0.$$

- $\tau < 1$: sharper; more mass on high-probability tokens.
- $\tau = 1$: the original distribution P.
- $\tau > 1$: flatter; more mass on lower-probability tokens.

Temperature Reshapes the Distribution

Given logits z_v, sample from the temperature-adjusted distribution

$$Q_\tau(v) = \frac{\exp(z_v/\tau)}{\sum_{u\in\mathcal{V}} \exp(z_u/\tau)}, \qquad \tau > 0.$$

- $\tau < 1$: sharper; more mass on high-probability tokens.
- $\tau = 1$: the original distribution P.
- $\tau > 1$: flatter; more mass on lower-probability tokens.

For finite logits, $\tau \to \infty$ gives a uniform distribution. As $\tau \to 0$, mass concentrates on the largest logits (greedy if the maximum is unique).
A system's "temperature 0" setting usually selects greedy directly.

Temperature: A Reproducible Example

The same transformation can be computed from probabilities:

$$Q_\tau(v) = \frac{P(v)^{1/\tau}}{\sum_u P(u)^{1/\tau}}.$$

Token	$\tau = 1$: original
cat	.500
dog	.250
fish	.120
bird	.080
hamster	.030
rabbit	.020

Temperature: A Reproducible Example

The same transformation can be computed from probabilities:

$$Q_\tau(v) = \frac{P(v)^{1/\tau}}{\sum_u P(u)^{1/\tau}}.$$

Token	$\tau = 1$: original	$\tau = 0.5$: square
cat	.500	.747
dog	.250	.187
fish	.120	.043
bird	.080	.019
hamster	.030	.003
rabbit	.020	.001

$\sum_u P(u)^2 = .3346$, so $Q_{0.5}(\text{cat}) = .25/.3346 \approx .747$.

Temperature: A Reproducible Example

The same transformation can be computed from probabilities:

$$Q_\tau(v) = \frac{P(v)^{1/\tau}}{\sum_u P(u)^{1/\tau}}.$$

Token	$\tau = 1$: original	$\tau = 0.5$: square	$\tau = 2$: square root
cat	.500	.747	.329
dog	.250	.187	.232
fish	.120	.043	.161
bird	.080	.019	.131
hamster	.030	.003	.081
rabbit	.020	.001	.066

$\sum_u P(u)^2 = .3346$, so $Q_{0.5}(\text{cat}) = .25/.3346 \approx .747$.

Top-k Sampling

Fan et al. (2018): restrict sampling to the k most probable tokens.

1. Find the set V_k of the top k tokens (with a fixed tie-break).
2. Set probabilities outside V_k to zero.
3. Renormalize the retained probabilities and sample.

Top-k Sampling

Fan et al. (2018): restrict sampling to the k most probable tokens.

1. Find the set V_k of the top k tokens (with a fixed tie-break).
2. Set probabilities outside V_k to zero.
3. Renormalize the retained probabilities and sample.

$$Q_k(v) = \begin{cases} \dfrac{P(v)}{\sum_{u \in V_k} P(u)} & v \in V_k, \\ 0 & v \notin V_k. \end{cases}$$

$k = 1$ gives greedy selection; $k = |\mathcal{V}|$ gives full-distribution sampling.

Top-k: Retain, Then Renormalize

Use $k = 3$ on the same six-token distribution.

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02

Top-k: Retain, Then Renormalize

Use $k = 3$ on the same six-token distribution.

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02
Retained?	✓	✓	✓	×	×	×

Retained mass: $.50 + .25 + .12 = .87$.

Top-k: Retain, Then Renormalize

Use $k = 3$ on the same six-token distribution.

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02
Retained?	✓	✓	✓	×	×	×
$Q_k(v)$	.575	.287	.138	0	0	0

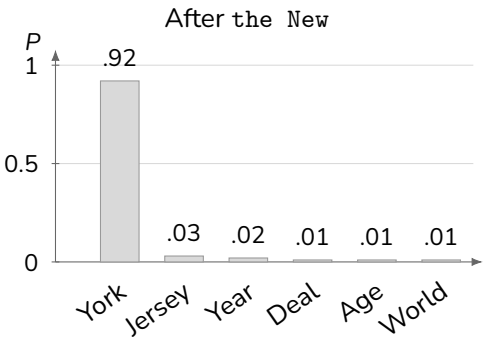
Retained mass: $.50 + .25 + .12 = .87$.

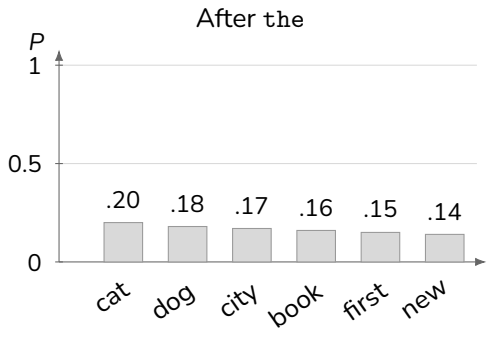
$$Q_k(\text{cat}) = \frac{.50}{.87} \approx .575.$$

The candidate count stays at k even when the next-token distribution becomes sharper or flatter.

Top-*k*: Different Prefixes, Different Distributions

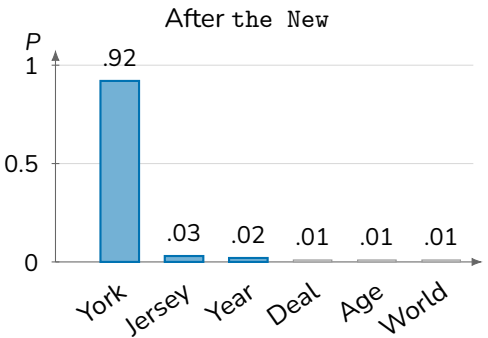
Illustrative distributions: next-token probabilities depend on the prefix.



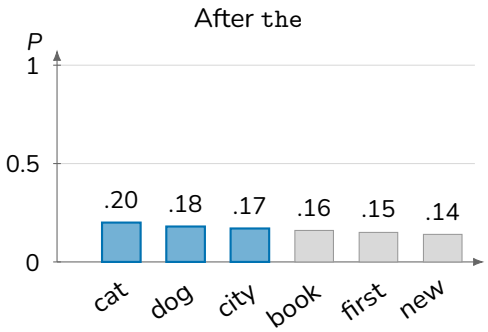


Top-k: Different Prefixes, Different Distributions

Illustrative distributions: next-token probabilities depend on the prefix.



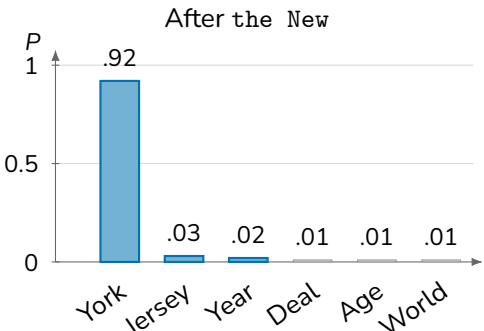
$k = 3$: retained mass .**97**.



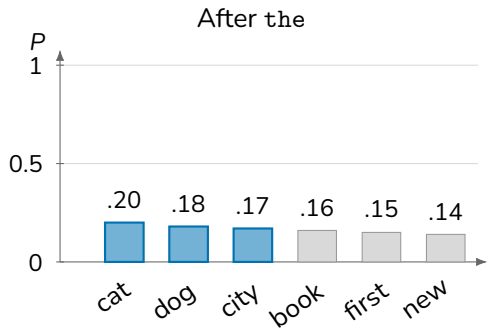
$k = 3$: retained mass .**55**.

Top-k: Different Prefixes, Different Distributions

Illustrative distributions: next-token probabilities depend on the prefix.



$k = 3$: retained mass .**97**.



$k = 3$: retained mass .**55**.

Fixed k, different retained probability mass.
Can we adapt the number of candidates to the distribution?

Nucleus (Top-p) Sampling

Holtzman et al. (2020): retain enough of the most probable tokens to reach a target mass p, with $0 < p \leq 1$.

1. Sort tokens so that $P(v_1) \geq P(v_2) \geq \cdots$.
2. Find the shortest **sorted prefix** reaching the threshold:

$$m = \min \left\{ j : \sum_{i=1}^{j} P(v_i) \geq p \right\}, \qquad V_p = \{v_1, \ldots, v_m\}.$$

3. Renormalize over V_p and sample.

Nucleus (Top-p) Sampling

Holtzman et al. (2020): retain enough of the most probable tokens to reach a target mass p, with $0 < p \leq 1$.

1. Sort tokens so that $P(v_1) \geq P(v_2) \geq \cdots$.
2. Find the shortest **sorted prefix** reaching the threshold:

$$m = \min \left\{ j : \sum_{i=1}^{j} P(v_i) \geq p \right\}, \qquad V_p = \{v_1, \ldots, v_m\}.$$

3. Renormalize over V_p and sample.

Include the token that crosses the threshold. The retained mass may exceed p, and the candidate count can change at every step.

Nucleus: Find the Cutoff, Then Renormalize

Use $p = 0.90$ on the same sorted distribution.

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02
Cumulative	.50	.75	.87	.95	.98	1.00

Nucleus: Find the Cutoff, Then Renormalize

Use $p = 0.90$ on the same sorted distribution.

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02
Cumulative	.50	.75	.87	.95	.98	1.00
Retained?	✓	✓	✓	✓	×	×

Three tokens cover $.87 < .90$; four cover $.95 \geq .90$.

Nucleus: Find the Cutoff, Then Renormalize

Use $p = 0.90$ on the same sorted distribution.

Token	cat	dog	fish	bird	hamster	rabbit
$P(v)$	.50	.25	.12	.08	.03	.02
Cumulative	.50	.75	.87	.95	.98	1.00
Retained?	✓	✓	✓	✓	×	×
$Q_p(v)$	.526	.263	.126	.084	0	0

Three tokens cover $.87 < .90$; four cover $.95 \geq .90$.

$$Q_p(\text{cat}) = \frac{.50}{.95} \approx .526.$$

Divide by the **actual retained mass** .95, not the threshold .90. Displayed probabilities are rounded.

The Nucleus Adapts to the Distribution

Two illustrative six-token distributions, each sorted from largest to smallest:

Rank	1	2	3	4	5	6
Peaked	.92	.03	.02	.01	.01	.01
Diffuse	.20	.18	.17	.16	.15	.14

The Nucleus Adapts to the Distribution

Two illustrative six-token distributions, each sorted from largest to smallest:

Rank	1	2	3	4	5	6
Peaked	.92	.03	.02	.01	.01	.01
Diffuse	.20	.18	.17	.16	.15	.14

- **Top-k, $k = 3$:** keeps three tokens in either case, retaining mass .97 or .55.
- **Top-p, $p = .90$:** keeps one token in the peaked case, and all six in the diffuse case (the first five total .86).

The Nucleus Adapts to the Distribution

Two illustrative six-token distributions, each sorted from largest to smallest:

Rank	1	2	3	4	5	6
Peaked	.92	.03	.02	.01	.01	.01
Diffuse	.20	.18	.17	.16	.15	.14

- **Top-k, $k = 3$:** keeps three tokens in either case, retaining mass .97 or .55.
- **Top-p, $p = .90$:** keeps one token in the peaked case, and all six in the diffuse case (the first five total .86).

When combining temperature and truncation, apply temperature first, then compute the cutoff from the adjusted probabilities. A high-probability candidate can still be factually wrong.

Mitigating Repetition: A Logit Penalty

A sign-aware repetition penalty lowers logits for tokens already in the prefix. With penalty factor $\rho > 1$:

$$z'_v = \begin{cases} z_v/\rho & v \text{ has appeared and } z_v > 0, \\ \rho z_v & v \text{ has appeared and } z_v < 0, \\ z_v & \text{otherwise.} \end{cases}$$

Mitigating Repetition: A Logit Penalty

A sign-aware repetition penalty lowers logits for tokens already in the prefix. With penalty factor $\rho > 1$:

$$z'_v = \begin{cases} z_v/\rho & v \text{ has appeared and } z_v > 0, \\ \rho z_v & v \text{ has appeared and } z_v < 0, \\ z_v & \text{otherwise.} \end{cases}$$

Example: with $\rho = 2$, a repeated token's logit 4 becomes 2, while -4 becomes -8. Both changes lower its score before softmax.

This can also suppress legitimate repetition, such as a person's name. The prefix used for the penalty may include the prompt, depending on the implementation.

Sign-aware form used in Hugging Face implementations; cf. Keskar et al. (2019).

Frequency, Presence, and n-gram Penalties

Let n_v be the number of occurrences of token v in the chosen prefix.

$$\begin{aligned} \text{Frequency penalty:} \quad & z'_v = z_v - \lambda_f n_v, \\ \text{Presence penalty:} \quad & z'_v = z_v - \lambda_p \mathbf{1}[n_v > 0], \end{aligned}$$

where $\lambda_f, \lambda_p \geq 0$. Frequency depends on how often a token occurs; presence depends only on whether it occurs.

Frequency, Presence, and n-gram Penalties

Let n_v be the number of occurrences of token v in the chosen prefix.

$$\begin{aligned} \text{Frequency penalty:} \quad & z'_v = z_v - \lambda_f n_v, \\ \text{Presence penalty:} \quad & z'_v = z_v - \lambda_p \mathbf{1}[n_v > 0], \end{aligned}$$

where $\lambda_f, \lambda_p \geq 0$. Frequency depends on how often a token occurs; presence depends only on whether it occurs.

n-gram blocking: set a candidate's logit to $-\infty$ if it would repeat an existing n-gram.

These heuristics alter the decoding distribution. They trade off repetition avoidance against fluency and faithful reuse of names or phrases.

Choosing a Decoder Means Choosing an Objective

- **Greedy / beam:** useful when a high-scoring, repeatable output is desired; beam search explores more prefixes at higher cost.
- **Sampling:** useful when multiple acceptable continuations or diverse candidates are desired.
- **Temperature and truncation:** control concentration and candidate support; tune them for the model and task.

Choosing a Decoder Means Choosing an Objective

- **Greedy / beam:** useful when a high-scoring, repeatable output is desired; beam search explores more prefixes at higher cost.
- **Sampling:** useful when multiple acceptable continuations or diverse candidates are desired.
- **Temperature and truncation:** control concentration and candidate support; tune them for the model and task.

Evaluate **generated outputs** using the task's criteria (Lecture 14). Low temperature does not guarantee factual accuracy, and sampling does not guarantee quality.

Choosing a Decoder Means Choosing an Objective

- **Greedy / beam:** useful when a high-scoring, repeatable output is desired; beam search explores more prefixes at higher cost.
- **Sampling:** useful when multiple acceptable continuations or diverse candidates are desired.
- **Temperature and truncation:** control concentration and candidate support; tune them for the model and task.

Evaluate **generated outputs** using the task's criteria (Lecture 14). Low temperature does not guarantee factual accuracy, and sampling does not guarantee quality.

Next: supervised fine-tuning and parameter-efficient adaptation.