

Decoding Strategies

Week 9: From Prediction to Generation

NLP Course 2025

Enhanced with Contrastive Search (2025)

November 9, 2025

Two-Tier BSc Discovery

Main: 25 slides — Appendix: 19 slides

From Prediction to Generation



Our Journey:

1. We trained models (Weeks 3-7: RNN → Transformers → BERT/GPT)
2. They learned to predict: $P(\text{word}|\text{context})$
3. They output probability distributions over 50,000+ words
4. **Today:** How do we convert these probabilities into actual text?

Models predict probabilities. Decoding converts probabilities to text.

Today's Challenge: From Probabilities to Text

The Setup: Model gives us probabilities for next word

Example: "The cat ..."

$$P(\text{sat}) = 0.45, \quad P(\text{is}) = 0.30, \quad P(\text{jumped}) = 0.25, \quad \dots \quad (50,000 \text{ words})$$

Naive Approach 1: Pick highest
→ Greedy Decoding

Result:

"The city is a major city in the city..."

Problem: Repetitive, boring

Naive Approach 2: Pick random
→ Pure Sampling

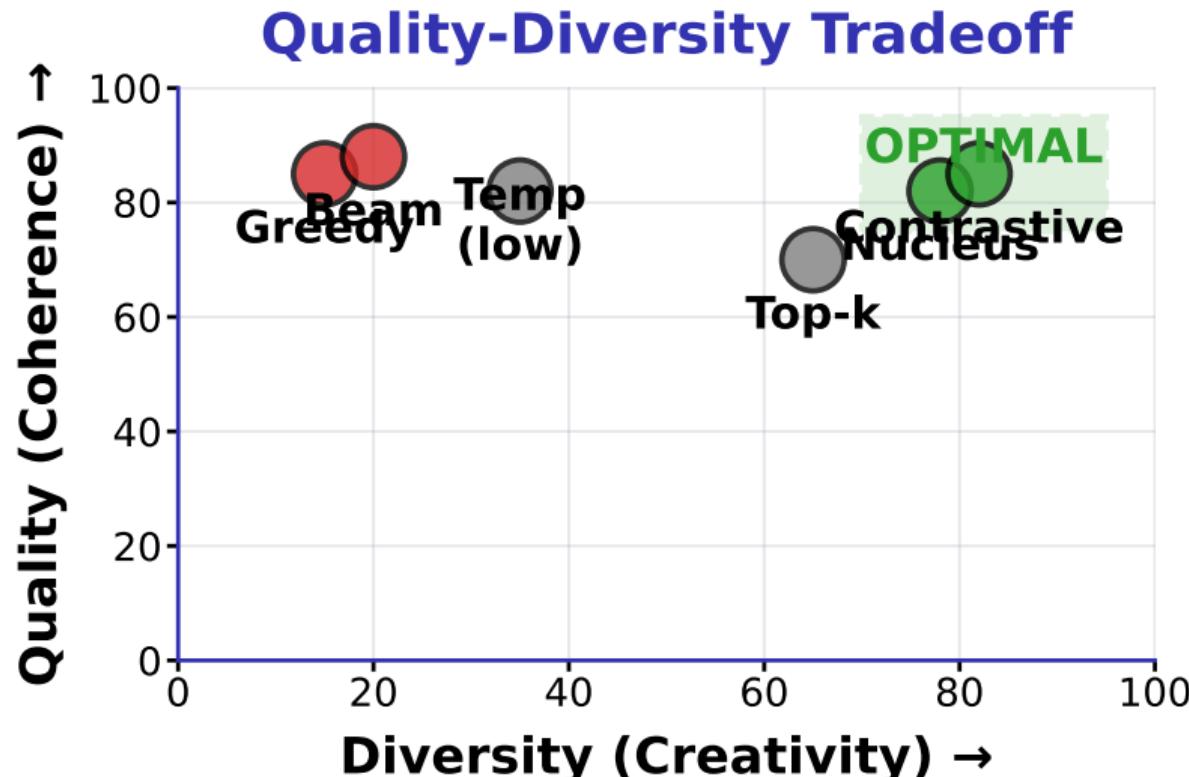
Result:

"The purple flying mathematics..."

Problem: Nonsense, incoherent

Core Challenge: Need text that is *coherent* AND *creative*

Today: 6 strategies to solve this - from simple to sophisticated



Discovery Question: Why is best text boring and creative text nonsense?

Three Decoding Families

Deterministic

Methods:

- Greedy
- Beam search

Traits:

- Same output
- High quality
- No creativity

Stochastic

Methods:

- Temperature
- Top-k
- Nucleus

Traits:

- Random
- Creative
- Variable quality

Balanced

Methods:

- Contrastive
- Hybrid

Traits:

- Best of both
- No repetition
- Modern std

Different tasks need different strategies - no single best method

6 methods total: Each optimizes different quality-diversity tradeoff

Greedy Decoding: The Baseline

How It Works:

1. Compute probabilities
2. Pick highest probability
3. Add to sequence
4. Repeat until done

Example:

$P(\text{cat}) = 0.45 \leftarrow \text{Pick!}$

$P(\text{dog}) = 0.30$

$P(\text{bird}) = 0.25$

When to Use:

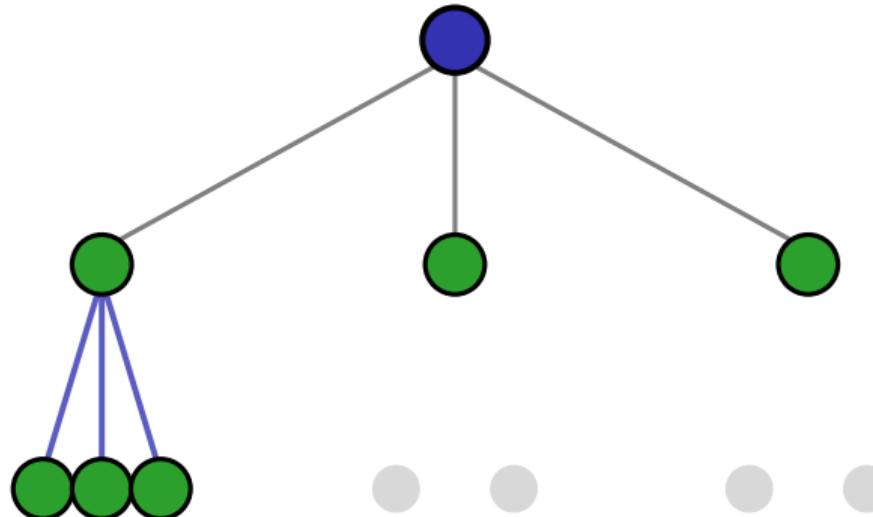
- Code generation
- Short responses
- Speed critical
- Reproducibility

Problem: Degeneration

"The city is a city in a city..."

Model gets stuck in loops

Simplest but prone to repetition



Width=3: Keep top 3 paths at each step

Key Insight: Keep top-k hypotheses at each step to find better sequences

Beam width = 3-5 typical. Balance greedy (1) vs exhaustive (∞)

Beam Example Tree Bsc

(Chart visualization here)

Worked example: “The cat...” with width=3 shows pruning in action

Beam Search: Detail

Algorithm:

1. Start: Keep top-k tokens
2. Expand: All continuations
3. Score: Multiply probs
4. Prune: Keep top-k sequences
5. Repeat until END

Scoring:

$$\text{score} = \prod_{i=1}^t P(y_i | y_{<i})$$

With length normalization:

$$\text{score} = \frac{1}{t} \sum_{i=1}^t \log P(y_i | y_{<i})$$

Best For:

- Machine translation
- Summarization
- Question answering
- "Correct" answer tasks

Parameters:

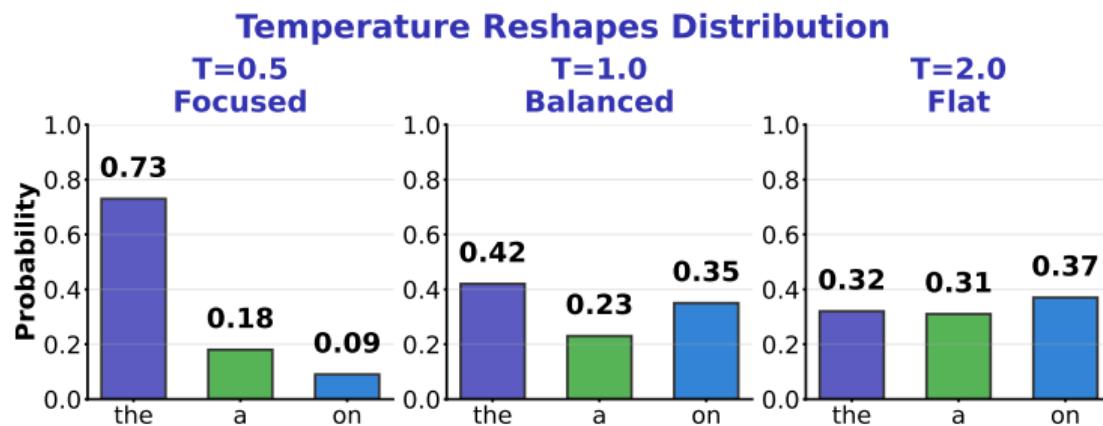
Width = 3-5 (translation)

Width = 10 (more diverse)

- + Better than greedy
- + Diverse hypotheses
- Still deterministic
- 4-5× slower

Beam search: workhorse for deterministic tasks

Temperature Sampling: Control Randomness



Key Insight: Temperature reshapes probability distribution

$T < 1$: focused. $T = 1$: unchanged. $T > 1$: random

Temperature Calculation Bsc

(Chart visualization here)

Concrete numbers show how temperature scaling works

Temperature: Detail

How It Works:

Given logits z_1, \dots, z_V

Scale by temperature T :

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

Sample from p

Effect:

$T \rightarrow 0$: argmax

$T = 1$: standard softmax

$T \rightarrow \infty$: uniform

Practical Settings:

- $T = 0.1\text{-}0.3$: Factual Q&A
- $T = 0.7$: Chatbots
- $T = 0.9\text{-}1.2$: Creative
- $T = 1.5+$: Experimental

+ Simple to implement

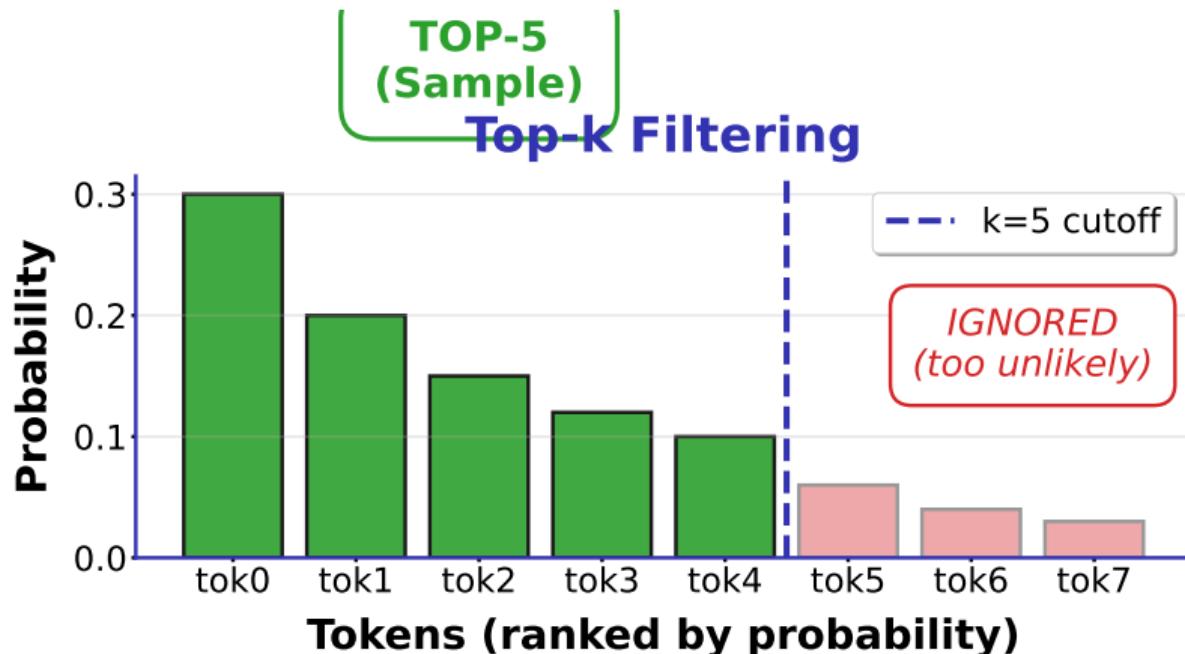
+ Continuous control

- No quality guarantee

- Can be nonsense

Temperature: simplest randomness control

Top-k Sampling: Filter the Tail



Key Insight: Only sample from top-k most likely tokens

Prevents sampling from long tail of unlikely words

Topk Example Bsc

(Chart visualization here)

Filtering + renormalization prevents tail sampling

Top-k: Detail

Algorithm:

1. Compute $P(w_i)$ for all
2. Sort by probability
3. Keep only top-k
4. Renormalize
5. Sample from p'

Example (k=3):

Original: [0.45, 0.18, 0.15, ...]
Keep: [0.45, 0.18, 0.15]
Renorm: [0.58, 0.23, 0.19]

Typical Values:

- k = 40-50 (balanced)
- k = 10-20 (focused)
- k = 100+ (very diverse)

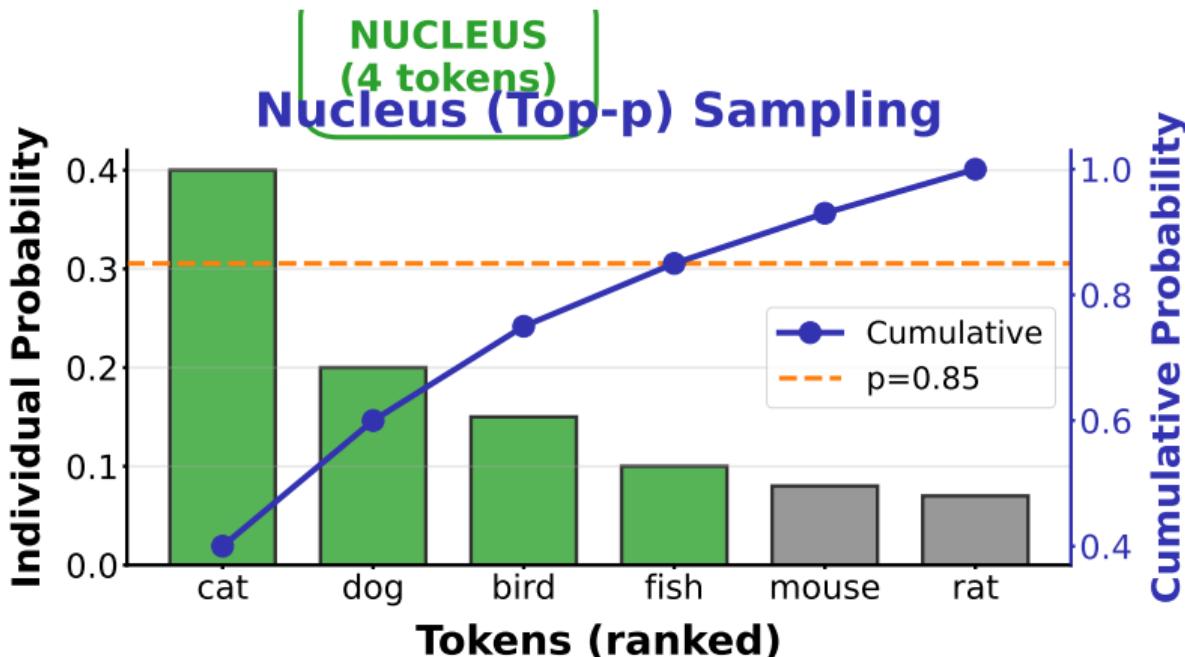
Limitation:

- Fixed cutoff regardless of distribution!
- Peaked: Wastes mass
- Flat: Too many bad tokens

Solution: Dynamic cutoff

Top-k improves over temperature but inflexible

Nucleus (Top-p) Sampling: Dynamic Cutoff

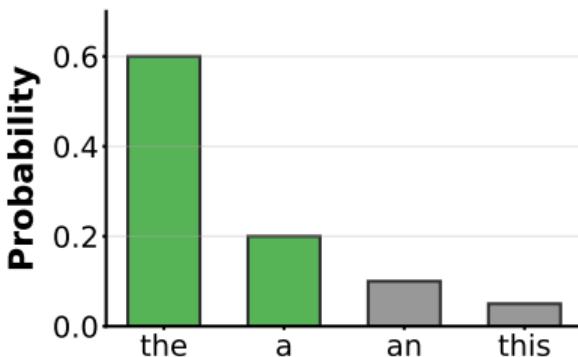


Key Insight: Adapt vocabulary size to distribution shape

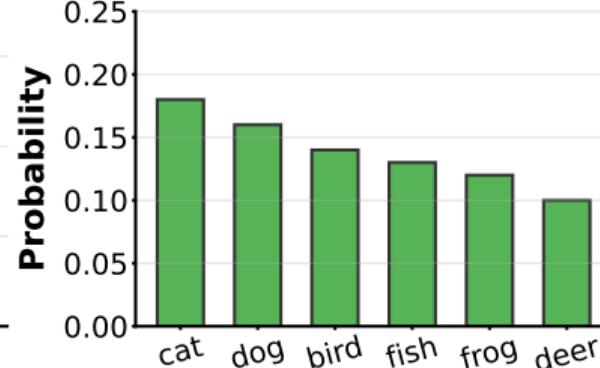
Nucleus size grows/shrinks based on probability spread

Nucleus Adapts to Distribution ($p=0.85$)

Peaked
Nucleus = 2



Flat
Nucleus = 6



Same p value → different vocabulary for peaked vs flat

Nucleus (Top-p): Detail

Algorithm:

1. Sort tokens by $P(w_i)$
2. Compute cumulative sum
3. Find smallest set where cum $\geq p$
4. Sample from nucleus

Example ($p=0.85$):

Cumsum: [0.40, 0.60, 0.75, 0.85]

Nucleus: First 4 tokens

Recommended:

- $p = 0.85\text{-}0.90$: Dialogue
- $p = 0.90\text{-}0.95$: Creative
- $p = 0.95\text{-}0.99$: Very diverse

Why Better:

Peaked \rightarrow small nucleus

Flat \rightarrow large nucleus

Adapts automatically!

Current Standard

ChatGPT, Claude, GPT-4

Nucleus: modern standard for high-quality generation

Greedy Decoding Output:

*"The city of New York is a major city.
The city is known for its culture.
The city has many attractions..."*

"the city" appears 3 times!

Solution: Contrastive Search penalizes repetition

Discovery: Why do models repeat themselves?

Greedy/beam maximize probability - but high prob = repeating recent context

Contrastive Mechanism Bsc

(Chart visualization here)

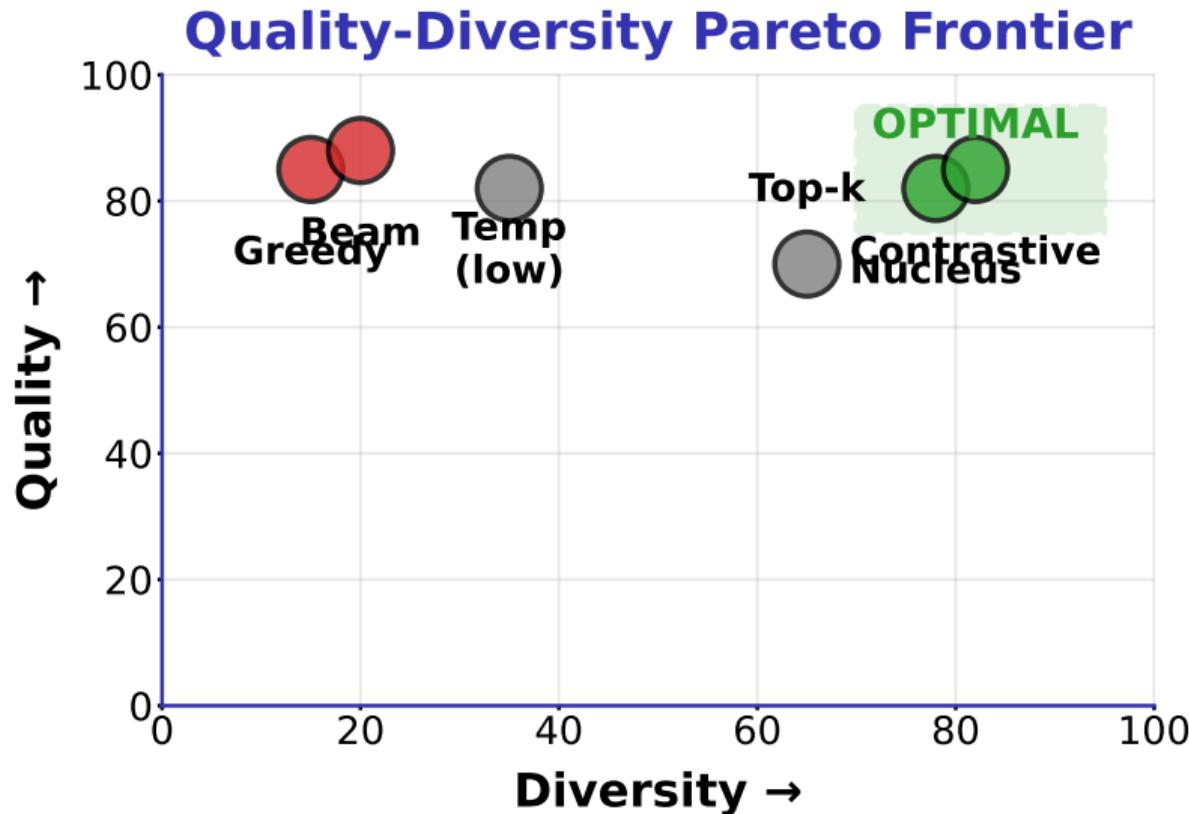
Key Insight: Balance probability with diversity penalty

Explicitly avoid copying recent context - prevents degeneration

Contrastive Vs Nucleus Bsc

(Chart visualization here)

Contrastive prevents repetition better for long generation



Pareto Frontier: No method dominates all others

Task Method Decision Tree Bsc

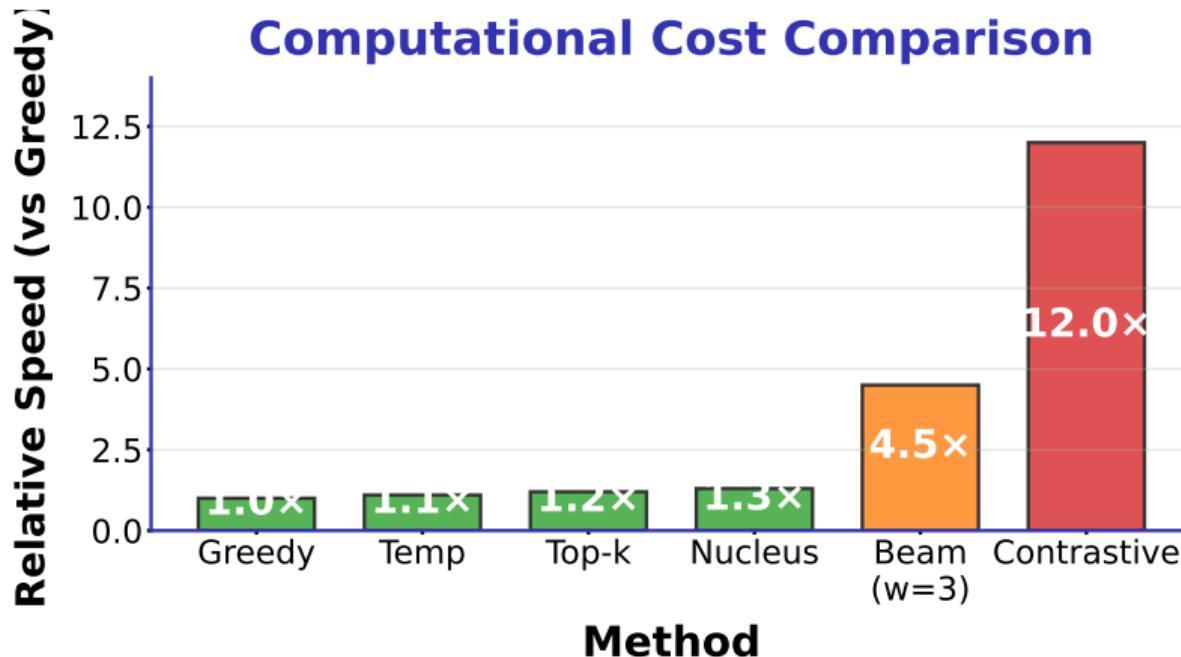
(Chart visualization here)

Start with task requirements, follow tree to recommended method

Task Recommendations Table Bsc

(Chart visualization here)

Comprehensive mapping: 8 tasks → optimal strategies



Contrastive: best quality-diversity but 12 \times slower. **Nucleus: best balanced**

Production tradeoff: Quality vs speed vs diversity

Key Takeaways

1. **Deterministic** (Greedy, Beam): High quality, no diversity → factual tasks
2. **Temperature**: Simple randomness control → universal but crude
3. **Top-k**: Fixed vocabulary filter → prevents tail sampling
4. **Nucleus (Top-p)**: Dynamic cutoff → modern standard, adapts
5. **Contrastive (NEW)**: Degeneration prevention → long creative text
6. **Task matters**: Translation → Beam — Dialogue → Nucleus — Stories → Contrastive

Next: Lab - Implement all 6 methods, measure quality-diversity tradeoffs

Decoding strategy matters as much as model architecture

Technical Appendix

19 slides: Complete mathematical treatment

A1-A5: Beam Search Mathematics

A6-A10: Sampling Mathematics

A11-A14: Contrastive Search & Degeneration

A15-A19: Advanced Topics & Production

A1: Beam Search Formulation

Objective: Find $y^* = \operatorname{argmax} P(y|x)$

Decomposition:

$$P(y|x) = \prod_{t=1}^T P(y_t|y_{<t}, x)$$

Log-probability:

$$\log P(y|x) = \sum_{t=1}^T \log P(y_t|y_{<t}, x)$$

Beam Search Approximation:

Instead of V^T sequences, maintain top-k hypotheses

Complexity:

Time: $O(k \cdot V \cdot T)$

Space: $O(k \cdot T)$

Tractable approximation to exact search

Length Normalization Bsc

(Chart visualization here)

Solution: Normalize by length

$$\text{score}(y) = \frac{1}{|y|^\alpha} \log P(y) \quad \text{where } \alpha \in [0.5, 1.0]$$

Length normalization essential for fair comparison

A3: Beam Search Variants

Diverse Beam:

Partition beams into groups
Penalize within-group similarity

Constrained Beam:

Force certain tokens
Keywords, entities

Stochastic Beam:

Sample beams (not argmax)
More diverse

Block n-gram:

Penalize n-gram repetition
"city is a city" prevention

Many variants for specific requirements

A4: Beam Search Stopping

When to stop?

1. **Fixed length:** Stop at T_{\max} (simple but rigid)
2. **END token:** Special token (most common)
3. **Score threshold:** When best cannot improve
4. **Timeout:** Computational budget

Stopping criterion affects output length

A5: Beam Search Limitations

Fundamental Issues:

1. **Exposure bias:** Train with teacher forcing, test with own outputs
2. **Label bias:** Cannot compare different prefixes fairly
3. **Repetition:** Still can loop
4. **Bland outputs:** Maximizes probability \neq interestingness
5. **Search errors:** May miss better sequences

When Beam Fails:

Open-ended generation, long-form text, creative tasks

→ Need sampling-based methods

Beam optimizes wrong objective for creative tasks

A6: Sampling as Inference

Goal: Sample $y \sim P(y|x)$ not $\text{argmax } P(y|x)$

Ancestral Sampling:

For $t = 1$ to T :

 Compute $P(y_t|y_{<t}, x)$

 Sample $y_t \sim P(\cdot|y_{<t}, x)$

Variants:

Temperature: Reshape before sample

Top-k: Truncate before sample

Nucleus: Dynamic truncation

Sampling enables diversity but loses quality guarantees

A7: Temperature Mathematics

Softmax with Temperature:

$$p_i(T) = \frac{\exp(z_i/T)}{\sum_{j=1}^V \exp(z_j/T)}$$

Limiting Cases:

$T \rightarrow 0$: deterministic (greedy)

$T \rightarrow \infty$: uniform ($1/V$)

Entropy Analysis:

$H(p) = -\sum p_i \log p_i$ measures randomness

H increases with T

Low T (<0.5): $H \approx 0$

High T (>2.0): $H \approx \log V$

Temperature controls distribution entropy

A8: Top-k Mathematics

Formal Definition:

$$V_k = \{w_{\sigma(1)}, \dots, w_{\sigma(k)}\}$$

(top k by prob)

Truncated distribution:

$$p'(w) = \begin{cases} \frac{p(w)}{\sum_{w' \in V_k} p(w')} & w \in V_k \\ 0 & \text{otherwise} \end{cases}$$

Information Loss:

$H(p') < H(p)$ (entropy decreases)

Loss \approx tail information

Top-k sacrifices tail probability for quality

Formal Definition:

$$V_p = \min \left\{ V' : \sum_{w \in V'} p(w) \geq p \right\}$$

Smallest set with cumulative mass $\geq p$

Dynamic Size:

$$|V_p| = \min \{k : \sum_{i=1}^k p_{\sigma(i)} \geq p\}$$

Peaked: Small $|V_p|$ (2-5)

Flat: Large $|V_p|$ (50+)

Why Nucleus > Top-k:

Top-k: Fixed k

Nucleus: Adapts to distribution

Nucleus adjusts vocabulary to distribution

Quality Metrics Bsc

(Chart visualization here)

Need both quality AND diversity metrics

A11: Degeneration Problem (Formal)

Definition: Unnatural repetitions in generated text

Why It Happens:

1. Model trained on natural text (low repetition)
2. Generation maximizes $P(y_t|y_{<t})$
3. Recent context influences P
4. Feedback loop: high prob → context → same high prob

Quantifying:

Greedy repetition: 15-30%

Human text: 2-5%

Gap = degeneration

Examples: “*I think that I think that I think...*”

Maximizing probability \neq natural text

A12: Contrastive Search Objective

Scoring Function:

$$\text{score}(w_t) = (1 - \alpha) \times P(w_t | y_{<t}) - \alpha \times \max_{w_i \in y_{<t}} \text{sim}(w_t, w_i)$$

where $\alpha \in [0, 1]$ controls tradeoff

Similarity: Cosine similarity using token embeddings

$$\text{sim}(w_i, w_j) = \frac{h_i \cdot h_j}{\|h_i\| \cdot \|h_j\|}$$

Algorithm:

1. Get top-k by probability
2. Compute similarity to all $y_{<t}$
3. Apply penalty: $\text{score} = \text{prob} - \alpha \times \text{max_sim}$
4. Select highest score

Explicit penalty for copying context

A13: Contrastive Parameters

Alpha (α):

- $\alpha = 0$: Pure greedy
- $\alpha = 0.6$: Balanced (default)
- $\alpha = 1.0$: Max diversity

Typical Settings:

- Short (<100): $\alpha = 0.4 - 0.5$
- Medium (<500): $\alpha = 0.5 - 0.6$
- Long (500+): $\alpha = 0.6 - 0.7$

Top-k (candidates):

- $k = 4$: Fast, focused
- $k = 6$: Balanced (default)
- $k = 10$: Diverse

Cost: $O(k \times T^2)$

12× slower than greedy
Only for quality-critical

Hugging Face default: $\alpha=0.6$, $k=4$

Research Findings:

- Greedy: 18-25% repetition (GPT-2), 12-18% (GPT-3)
- Nucleus: 8-12% (still above human 3-5%)
- Contrastive: 4-7% (closest to human)

Why Probability Fails:

Training: Next token prediction

Generation: Global coherence

Mismatch: Local optimum \neq global quality

Solutions Hierarchy:

1. Temp/Top-k/Nucleus: Reduce greedy determinism
2. Contrastive: Explicit penalty
3. RLHF/DPO: Align with humans (Week 10)

Contrastive addresses fundamental limitation

A15: Hybrid Methods

Combining Strategies:

Nucleus + Temperature:

$$p_i(T) = \text{softmax}(z/T), \text{ then } V_p \leftarrow \text{nucleus}(p_i)$$

Used by GPT-3, ChatGPT

Beam + Sampling:

Beam search + stochastic selection

Contrastive + Nucleus:

Best of both worlds

Hybrid methods leverage complementary strengths

A16: Constrained Decoding (2025)

Goal: Force tokens/patterns

Lexically Constrained:

Must include keywords

Beam variant tracks satisfaction

Format Constraints:

JSON output, code syntax

Use Cases:

Structured extraction, controllable summarization, code generation

Constrained decoding enables controllable generation

A17: Computational Complexity

Method	Time/token	Total	Speed
Greedy	$O(V)$	$O(VT)$	1.0×
Temperature	$O(V)$	$O(VT)$	1.1×
Top-k	$O(V)$	$O(VT)$	1.2×
Nucleus	$O(V \log V)$	$O(V \log V \cdot T)$	1.3×
Beam (k=5)	$O(kV)$	$O(kVT)$	4.5×
Contrastive	$O(kT)$	$O(kT^2)$	12×

Practical (1000 tokens):

Greedy: 2.5s — Nucleus: 3.2s — Beam: 11s — Contrastive: 30s

Computational cost matters for production

Production Settings Bsc

(Chart visualization here)

Real settings from major production systems

A19: Future Directions (2025)

Active Research:

1. Quality-diversity optimization
2. Learned decoding (RLHF, DPO)
3. Speculative decoding (4–8× faster)
4. Adaptive methods
5. Energy-based decoding

Open Problems:

- Auto-select best parameters for new task?
- Balance fluency + factuality + creativity?
- Efficient 100K+ token decoding?

Trend: Hand-tuned → learned decoding strategies

Active research area with many open questions