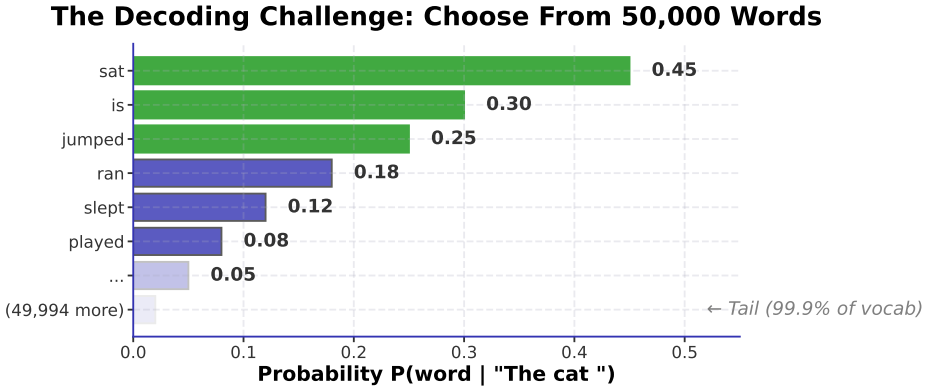
Decoding Strategies

Week 9: From Probabilities to Text

November 2025

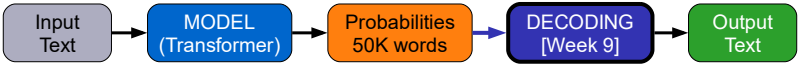
The Decoding Challenge: Choosing From 50,000 Words



The Question: Given these probabilities for "The cat __", which word should we pick?

At each step, model outputs probability distribution over entire vocabulary - how do we choose?

Context: How We Got Here

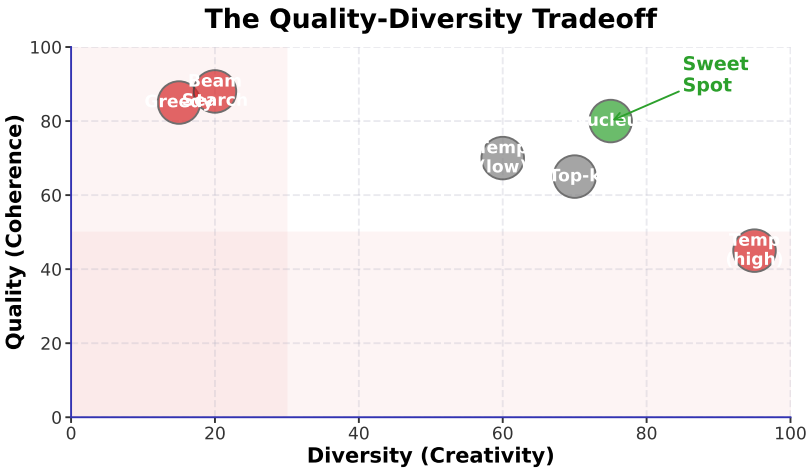


Our Journey:

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.

The Quality-Diversity Tradeoff



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

The central challenge: How to balance coherence with creativity

Bad Output: City Repetition

Framing 4: Real Bad Outputs

What's Wrong: Model gets trapped repeating same pattern
Output: "*The city is a major city in the city is a city...*"

Problem 1 of 6: Greedy decoding creates loops

Bad Output: Nonsense Words

Framing 4: Real Bad Outputs

What's Wrong: Ask 100 times, get identical response every time
Example: Always answers "The weather is nice" even for different contexts

Problem 3: Too Boring or Too Crazy

Bad Output: Always Same

Framing 4: Real Bad Outputs

What's Wrong: Can't balance quality and creativity simultaneously

Too focused:
"It is. It is. It is."

Too random:
"Flying purple elephant mathematics"

Problem 4: Missing Better Paths

Bad Output: Missed Better

Framing 4: Real Bad Outputs

What's Wrong: Greedy early choice blocks better sequences later
Example: Pick "The" (0.45) → stuck with "The cat sat"
But "A" (0.30) → leads to "A beautiful cat played"

Bad Output: Wrong for Distribution

Framing 4: Real Bad Outputs

What's Wrong: Sampling from tail produces nonsense words
Example: With 50K vocabulary, words at position 45,000 have P=0.00001
But they can still be sampled!

Problem 6: Speed vs Quality Tradeoff

Bad Output: Too Slow

Framing 4: Real Bad Outputs

What's Wrong: Methods that prevent repetition are 10-12× slower
Example: Contrastive search computes similarity to ALL previous tokens
For 1000-token text: 1000 comparisons per new token!

Solution 1 → Beam Search: Explore Multiple Paths

Problem 1 Recap:
Greedy decoding: Trapped in loops
Always picks highest probability
Misses better sequences

Need: Way to explore alternatives

Solution: Beam Search:
Keep top-k paths at each step
Explore k=3-5 hypotheses simultaneously
Pick best complete sequence at end

Result: Finds better sequences than greedy

How it solves Problem 1: Maintains multiple candidates, avoids greedy trap

Solution 1 of 6: Beam search for better quality

Solution 2 → Temperature: Add Controlled Randomness

Problem 2 Recap:
No diversity: Same output always
Deterministic selection
No creativity

Need: Controlled randomness

Solution: Temperature:
Reshape probability distribution
T ¡ 1: More focused
T ¿ 1: More random
Sample from adjusted distribution

Result: Different outputs each time

How it solves Problem 2: Sampling introduces stochasticity, enables diversity

Solution 2 of 6: Temperature for creativity control

Solution 3 → Top-k: Filter Unlikely Words

Problem 3 Recap:
Can't balance quality & creativity
Pure sampling too random
Greedy too boring

Need: Filter bad words, keep good

Solution: Top-k Sampling:
Keep only top-k most likely tokens
Cut tail of distribution
Renormalize probabilities
Sample from filtered set

Result: Diverse but not nonsensical

How it solves Problem 3: Fixed cutoff prevents tail sampling while allowing creativity

Solution 3 of 6: Top-k for controlled sampling

Solution 4 → Nucleus: Dynamic Vocabulary Cutoff

Problem 4 Recap:
Top-k has fixed cutoff
Peaked distribution: Wastes probability
Flat distribution: Still allows junk

Need: Adaptive cutoff

Solution: Nucleus (Top-p):
Choose smallest set with cumulative prob p
Adapts to distribution shape
Peaked → small nucleus (2-3 words)
Flat → large nucleus (50+ words)

Result: Automatic quality-diversity balance

How it solves Problem 4: Dynamic cutoff adapts to each prediction step

Solution 4 of 6: Nucleus for adaptive sampling

Solution 5 → Top-k + Temperature: Hybrid Control

Problem 5 Recap:
Temperature alone doesn't filter tail
Top-k alone doesn't control randomness
Need both filtering AND tuning

Need: Combine strategies

Solution: Hybrid Methods:
Apply temperature THEN top-k
Or: Apply nucleus THEN temperature
Leverages strengths of both
Production systems use combinations

Result: Fine-grained control over generation

How it solves Problem 5: Layered strategies handle multiple issues simultaneously

Solution 5 of 6: Hybrid methods for comprehensive control

Solution 6 → Contrastive: Explicit Degeneration Prevention

Problem 6 Recap:
Repetition even with sampling
Long generation degenerates
Context similarity causes loops

Need: Explicit repetition penalty

Solution: Contrastive Search:
Score = Probability - $\alpha\times$ Similarity
Penalize tokens similar to recent context
Balance quality with diversity
Modern standard for long text

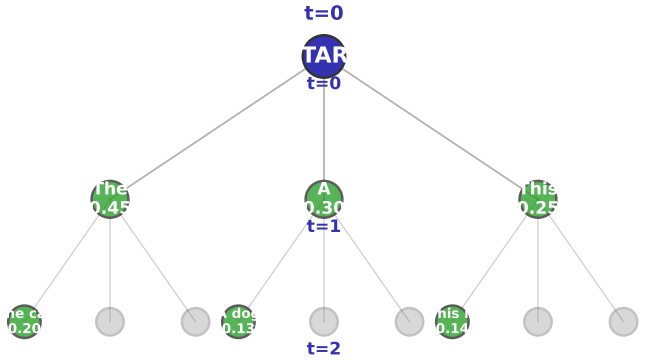
Result: Human-like text without repetition

How it solves Problem 6: Direct similarity penalty prevents copying context

Solution 6 of 6: Contrastive for degeneration-free generation

Beam Search: Explore Multiple Paths

../figures/beam_search_visual_bsc.pdf



Worked example shows why beam search finds better sequences than greedy

Beam Search: Algorithm & Settings

Algorithm:

1. Start: Keep top-k tokens
2. Expand: Generate continuations for each
3. Score: Multiply probabilities
4. Prune: Keep top-k sequences
5. Repeat until END token

Scoring:

$$\text{score}(y_1 \ldots y_t) = \prod_{i=1}^{t} P(y_i | y_{<i})$$

With length normalization:

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

Best For:

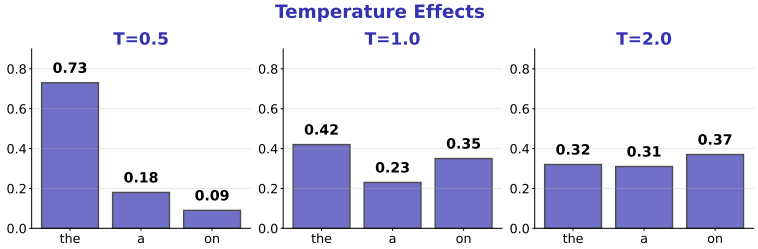
- Machine translation
- Summarization
- Question answering
- Tasks with "correct" answer

Parameters:
Width = 3-5 (translation)
Width = 10 (diverse outputs)

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

Beam search is the workhorse for deterministic tasks

Temperature Sampling: Control Randomness



Key Insight: Temperature reshapes probability distribution

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

Temperature: Worked Example

Temperature: Step-by-Step Calculation

Given: Logits = [2.0, 1.0, 0.5, 0.2]
Tokens = ["cat", "dog", "bird", "fish"]

T=0.5: [4.0, 2.0, 1.0, 0.4] → [0.73, 0.18, 0.07, 0.02]

→ 73% on "cat" (FOCUSED)

T=1.0: [2.0, 1.0, 0.5, 0.2] → [0.42, 0.23, 0.16, 0.13]

→ 42% on "cat" (BALANCED)

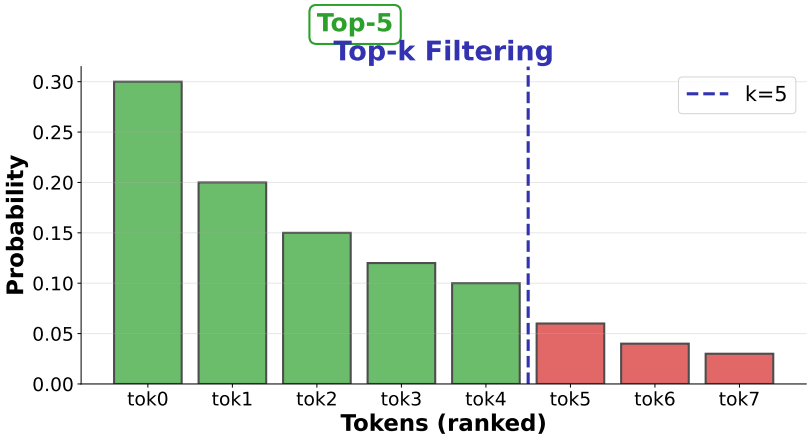
T=2.0: [1.0, 0.5, 0.25, 0.1] → [0.32, 0.26, 0.23, 0.19]

→ 32% on "cat" (FLAT)

Lower T = more peaked | Higher T = more flat

Concrete numbers show how temperature scaling works

Top-k Sampling: Filter the Tail



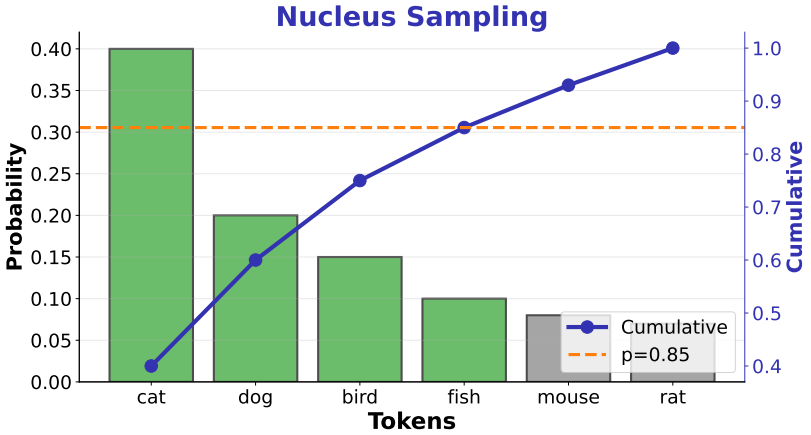
Key Insight: Only sample from top-k most likely tokens

Prevents sampling from long tail of unlikely words

Top-k: Worked Example (k=3)

../figures/topk_example_bsc.pdf

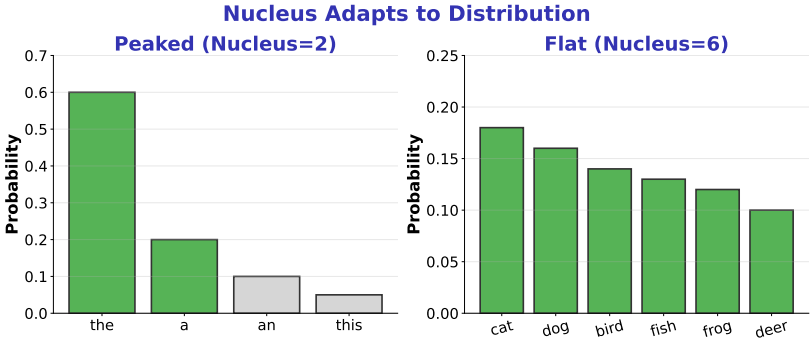
Nucleus (Top-p) Sampling: Dynamic Cutoff



Key Insight: Adapt vocabulary size to distribution shape

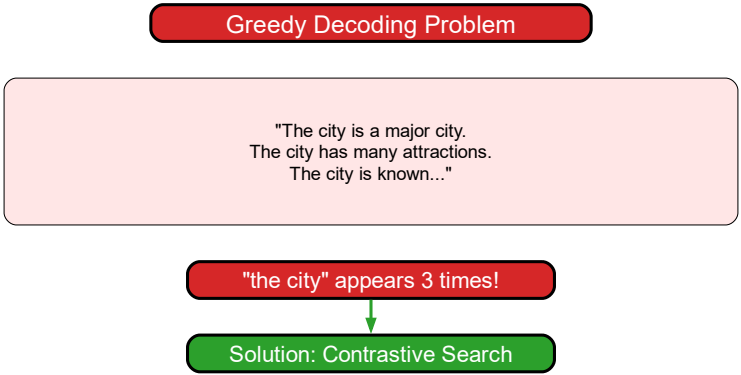
Nucleus size grows/shrinks based on probability spread

Nucleus: How Distribution Shape Matters



Same p value gives different vocabulary sizes for peaked vs flat distributions

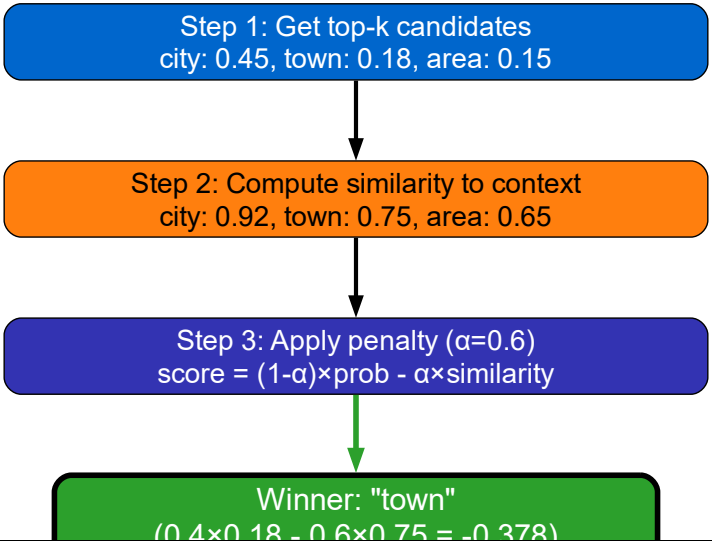
The Degeneration Problem



Discovery Question: Why do models repeat themselves?

Greedy and beam search maximize probability - but high probability = repeating recent context

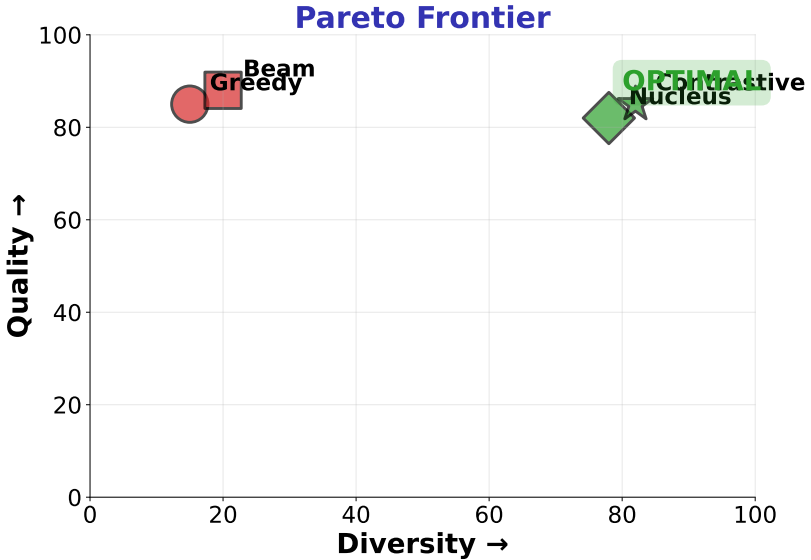
Contrastive Search: Penalize Repetition



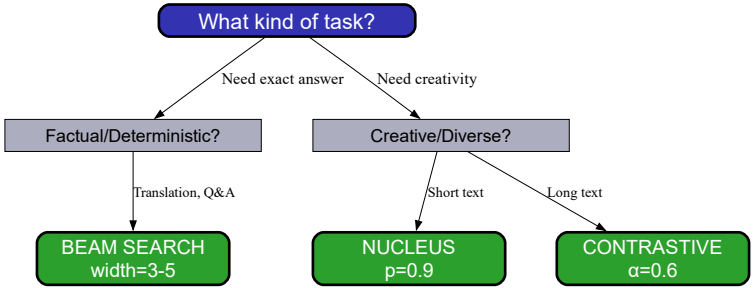
Contrastive vs Nucleus: Comparison

../figures/contrastive_vs_nucleus_bsc.pdf

All Methods on Quality-Diversity Space



Choosing the Right Method: Decision Tree



Start with task requirements, follow tree to recommended method

Task-Specific Recommendations

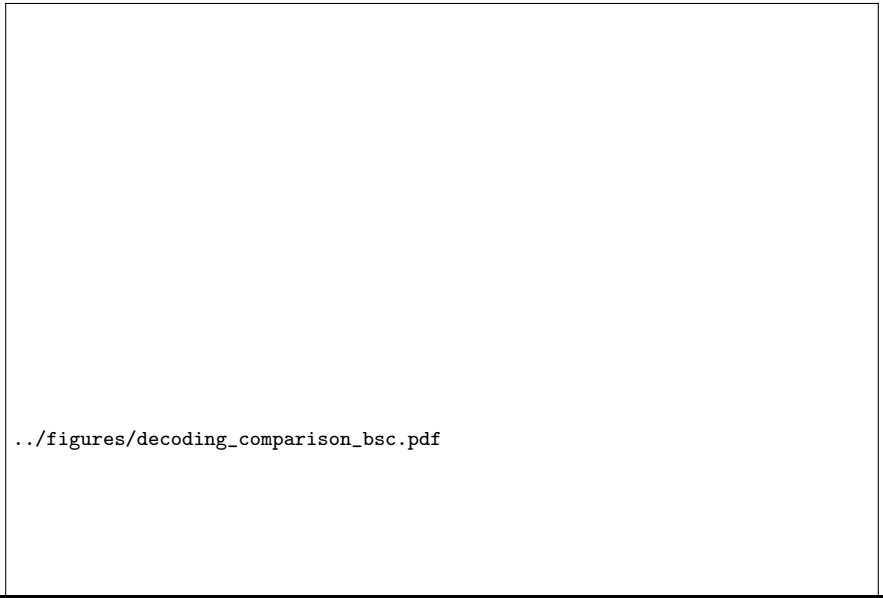
Task	Method	Parameters
Translation	Beam	w=3-5
Factual QA	Greedy	T=0.3
Code	Greedy	T=0
Dialogue	Nucleus	p=0.9
Creative	Nucleus	p=0.95
Long Stories	Contrastive	α=0.6

Comprehensive mapping from 8 common tasks to optimal decoding strategies

Computational Costs Matter

../figures/computational_cost_comparison_bsc.pdf

Method Comparison Overview



Deterministic vs Stochastic: The Fundamental Split

../figures/greedy_vs_sampling_bsc.pdf

Key Takeaways

1. **6 Problems → 6 Solutions**: Each method solves specific failure mode
2. **Deterministic** (Greedy, Beam): High quality, no diversity - factual tasks
3. **Stochastic** (Temperature, Top-k, Nucleus): Diverse but variable quality
4. **Balanced** (Contrastive): Explicit degeneration prevention
5. **Task matters**: Translation → Beam — Dialogue → Nucleus — Stories → Contrastive
6. **Tradeoffs**: Speed vs Quality, Diversity vs Coherence

Modern Standard: Nucleus (top-p=0.9) + Temperature (T=0.7) for most applications

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

Decoding strategy matters as much as model architecture

From Probabilities to Text: The Complete Journey

What We Learned:

- Models give us probability distributions (Week 3-7)
- Converting to text has 6 fundamental challenges
- Each decoding method addresses specific problems
- No universal best - choose based on task requirements
- Production systems use hybrid methods (Nucleus + Temperature)

Complete pipeline from model training to text generation

Technical Appendix

25 slides: Complete mathematical treatment

A1-A5: Beam Search Mathematics
A6-A10: Sampling Mathematics
A11-A14: Contrastive Search & Degeneration
A15-A19: Advanced Topics & Production
A20-A25: The 6 Problems - Technical Analysis (NEW)

A1: Beam Search Formulation

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

Decomposition:

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

Log-probability (more stable):

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

Beam Search Approximation:
Instead of exploring all V^T sequences, maintain top-k hypotheses at each step

Complexity:
Time: $O(k \cdot V \cdot T)$ where k = beam width, V = vocabulary, T = length
Space: $O(k \cdot T)$ to store hypotheses

Beam search is tractable approximation to exact search

A2: Length Normalization

Problem: Longer sequences have lower probabilities (more terms multiplied)

$$P(y_1, y_2, y_3, y_4) = \underbrace{0.5}_{y_1} \times \underbrace{0.5}_{y_2} \times \underbrace{0.5}_{y_3} \times \underbrace{0.5}_{y_4} = 0.0625$$

$$P(y_1, y_2) = 0.5 \times 0.5 = 0.25 > 0.0625$$

Bias toward shorter sequences!

Solution: Length normalization

$$\text{score}(y) = \frac{1}{|y|^{\alpha}} \log P(y)$$

where $\alpha \in [0.5, 1.0]$ (typically 0.6-0.7)

Effect:
Without: Beam search heavily biases toward short outputs
With: Fair comparison across different lengths

Length normalization is essential for beam search quality

A3: Beam Search Variants

Diverse Beam Search:
Partition beams into groups
Penalize within-group similarity
Result: More diverse hypotheses

Constrained Beam Search:
Force certain tokens to appear
Useful for: Keywords, entities
Applications: Controllable generation

Stochastic Beam Search:
Sample beams instead of argmax
Combines beam + sampling
More diverse than standard beam

Block n-gram Beam:
Penalize n-gram repetition
Prevents "the city is a city" loops
Common in summarization

Many beam search variants exist for specific requirements

A4: Beam Search Stopping Criteria

When to stop expanding beams?

Method 1: Fixed length
Stop at T_{max} tokens (simple but rigid)

Method 2: END token
Stop when beam generates special token (most common)

Method 3: Score threshold
Stop when best score cannot improve enough

$$\frac{\text{best\_incomplete}}{\text{best\_complete}} < \text{threshold}$$

Method 4: Timeout
Computational budget exceeded (production systems)

Choice of stopping criterion affects output length distribution

A5: Beam Search Limitations

Fundamental Issues:

1. **Exposure bias**: Trained with teacher forcing, tested with own outputs
2. **Label bias**: Cannot compare sequences of different prefixes fairly
3. **Repetition**: Still can loop ("the city is a major city")
4. **Bland outputs**: Maximizes probability, not interestingness
5. **Search errors**: May miss better sequences outside beam

When Beam Search Fails:
Open-ended generation (dialogue, stories)
Long-form text (repetition accumulates)
Creative tasks (probability $\neq$ quality)

$\rightarrow$ Need sampling-based methods

Beam search optimizes wrong objective for creative tasks

A6: Sampling as Inference

Goal: Sample $y \sim P(y|x)$ instead of argmax $P(y|x)$

Ancestral Sampling:
For $t = 1$ to T:
Compute $P(y_t|y_{<t}, x)$
Sample $y_t \sim P(\cdot|y_{<t}, x)$

Properties:
Stochastic: Different output each time
Explores full distribution (in expectation)
Can generate low-probability sequences

Variants:
Temperature: Reshape distribution before sampling
Top-k: Truncate distribution before sampling
Nucleus: Dynamic truncation before sampling

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:

$T \to 0$: $p_i \to \begin{cases} 1 & \text{if } i = \text{argmax}\, z \\ 0 & \text{otherwise} \end{cases}$ (greedy)

$T \to \infty$: $p_i \to 1/V$ (uniform)

Entropy Analysis:

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

H increases monotonically with T

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

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

Temperature provides continuous control over distribution entropy

A8: Top-k Mathematics

Formal Definition:
Let σ = permutation sorting probabilities descending

$$V_k = \{w_{\sigma(1)}, w_{\sigma(2)}, \ldots, w_{\sigma(k)}\}$$

Truncated distribution:

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

Information Loss:
Original entropy: $H(p) = -\sum_{i=1}^{V} p_i \log p_i$
After top-k: $H(p') = -\sum_{i=1}^{k} p'_i \log p'_i < H(p)$
Loss $\approx \sum_{i=k+1}^{V} p_i \log(1/p_i)$ (tail information)

Top-k sacrifices tail probability mass for sampling quality

A9: Nucleus (Top-p) Mathematics

Formal Definition:

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

Smallest set with cumulative mass $\geq p$

Dynamic Vocabulary Size:

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

Adapts to distribution shape:
Peaked: Small $|V_p|$ (2-5 tokens)
Flat: Large $|V_p|$ (50+ tokens)

Why Nucleus > Top-k:
Top-k: Fixed k regardless of $p(w)$ distribution
Nucleus: Adapts k to achieve consistent probability mass

Nucleus automatically adjusts vocabulary to distribution characteristics

A10: Sampling Quality Metrics

Quality Metrics:
Perplexity: $\exp(-\frac{1}{T}\sum \log p(y_t))$
Lower = better

BLEU (translation):
N-gram overlap with reference
0-100 scale

Human evaluation:
Fluency (1-5)
Relevance (1-5)

Diversity Metrics:
Distinct-n: $\frac{\text{unique n-grams}}{\text{total n-grams}}$
Higher = more diverse

Self-BLEU:
BLEU of output vs other outputs
Lower = more diverse

Repetition Rate:
$\frac{\text{repeated n-grams}}{\text{total n-grams}}$
Lower = less repetitive

Need both quality AND diversity metrics to evaluate decoding

A11: The Degeneration Problem (Formal)

Definition: Model-generated text with unnatural repetitions

Why It Happens:

1. Model trained on natural text (low repetition)
2. But generation maximizes $P(y_t|y_{<t})$
3. Recent context $y_{<t}$ influences P
4. Creates positive feedback: high prob word $\rightarrow$ context $\rightarrow$ same high prob word

Quantifying Degeneration:
Repetition rate in greedy: 15-30% (depending on domain)
Repetition rate in human text: 2-5%
Gap = degeneration problem

Examples:
"*The city is a major city in the United States. The city...*"
"*I think that I think that I think...*"

Maximizing probability does not equal natural text

A12: Contrastive Search Objective

Scoring Function:

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

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

Similarity Function:

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

(cosine similarity)
using token embeddings h

Algorithm:

1. Get top-k candidates by probability
2. For each candidate, compute similarity to all tokens in $y_{<t}$
3. Apply penalty: score = prob - $\alpha \times$ max_similarity
4. Select candidate with highest score

Contrastive search explicitly penalizes copying recent context

A13: Contrastive Search Parameters

Alpha (α):
$\alpha = 0$: Pure greedy (no penalty)
$\alpha = 0.6$: Balanced (recommended)
$\alpha = 1.0$: Maximum diversity (risky)

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

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

Computational Cost:
For each step:

- Compute similarities: $O(k \times t)$
- t grows with generation

Total: $O(k \times T^2)$
12× slower than greedy

Hugging Face default: α=0.6, k=4

A14: Degeneration Analysis

Research Findings (2024-2025):

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

Why Probability Maximization Fails:
Training objective: Next token prediction
But generation requires: Global coherence
Mismatch: Local optimum $\neq$ global quality

Solutions Hierarchy:

1. Temperature/Top-k/Nucleus: Reduce greedy's determinism
2. Contrastive: Explicit degeneration penalty
3. RLHF/DPO: Align model with human preferences (different lecture)

Contrastive search addresses fundamental limitation of likelihood-based decoding

A15: Hybrid Decoding Methods

Combining Strategies:

Nucleus + Temperature:
Apply temperature THEN nucleus

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

Used by GPT-3 API, ChatGPT

Beam + Sampling:
Beam search with stochastic selection
Keep top-k, sample from them (not argmax)

Contrastive + Nucleus:
Nucleus for candidate generation
Contrastive scoring for selection
Best of both worlds

Hybrid methods leverage complementary strengths

A16: Constrained Decoding (2025)

Goal: Force certain tokens/patterns to appear

Lexically Constrained:
Must include keywords: {"AI", "ethics", "safety"}
Beam search variant: Track constraint satisfaction

Format Constraints:
JSON output: Force structure {"key": "value"}
Code: Force syntactic validity

NeuroLogic Decoding (2021):
Beam search + constraint satisfaction
Optimal for: Keyword-based generation

Production Use Cases:
Structured data extraction (force JSON)
Controllable summarization (force keywords)
Code generation (force syntax)

Constrained decoding enables controllable generation

A17: Computational Complexity Comparison

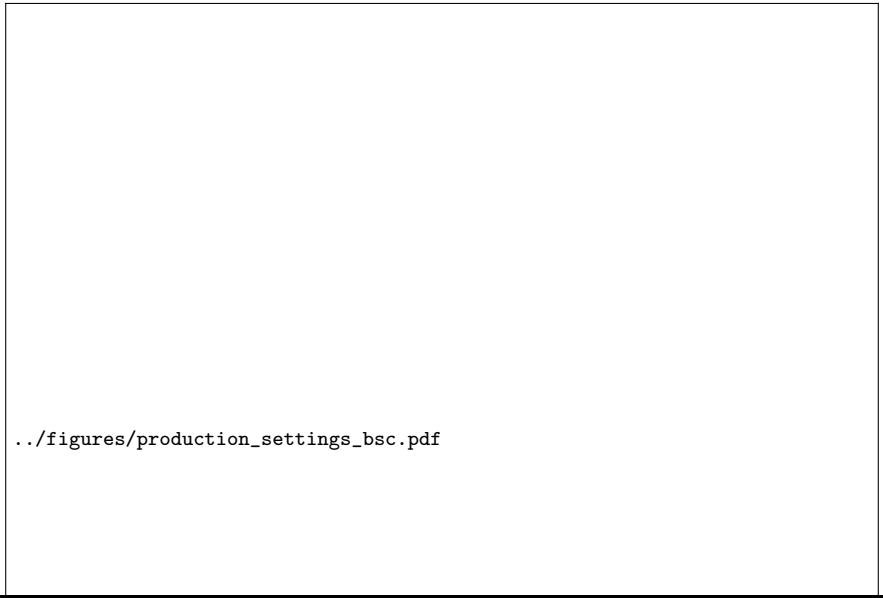
Method	Time per token	Total complexity	Relative speed
Greedy	$O(V)$	$O(V \times T)$	1.0× (baseline)
Temperature	$O(V)$	$O(V \times T)$	1.1× (softmax overhead)
Top-k	$O(V)$	$O(V \times T)$	1.2× (sorting)
Nucleus	$O(V \log V)$	$O(V \log V \times T)$	1.3× (sort + cumsum)
Beam (k=5)	$O(k \times V)$	$O(k \times V \times T)$	4.5× (k=5)
Contrastive	$O(k \times T)$	$O(k \times T^2)$	12× (similarity)

Key Insight: Contrastive's T^2 term makes it expensive for long sequences

Practical Impact (1000-token generation):
Greedy: 2.5 seconds
Nucleus: 3.2 seconds (best choice)
Beam: 11 seconds
Contrastive: 30 seconds (only if quality critical)

Computational cost matters for production deployment

A18: Production Deployment Settings (2024-2025)



A19: Future Directions & Open Problems

Active Research Areas (2025):

1. **Quality-diversity optimization**: Multi-objective search methods
2. **Learned decoding**: Train models to decode better (RLHF, DPO)
3. **Speculative decoding**: Parallel generation for speed (4-8× faster)
4. **Adaptive methods**: Choose strategy dynamically during generation
5. **Energy-based decoding**: Score sequences globally (not token-by-token)

Open Problems:
How to automatically select best T, p, k, α for new task?
How to balance fluency + factuality + creativity simultaneously?
How to decode efficiently for 100K+ token outputs?

Trend: Moving from hand-tuned parameters to learned decoding strategies

---

Decoding is an active research area with many open questions

Appendix B: The 6 Problems - Technical Deep Dive

Technical Analysis of Each Failure Mode

Next 6 slides: Detailed technical analysis pairing:

Top: Technical level explanation (framing3)
Bottom: Concrete output failure (framing4)

B1: Problem 1 Technical Analysis

Technical Issue (Level):

Level 1: Too Deterministic

Framing 3: Progressive Sophistication

B2: Problem 2 Technical Analysis

Technical Issue (Level):

Level 2: Too Random

Framing 3: Progressive Sophistication

B3: Problem 3 Technical Analysis

Technical Issue (Level):

Level 3: Inflexible Filtering

Framing 3: Progressive Sophistication

Technical Issue (Level):

Level 4: Missing Better Paths

Framing 3: Progressive Sophistication

B5: Problem 5 Technical Analysis

Technical Issue (Level):

Level 5: Still Repetitive

Framing 3: Progressive Sophistication

B6: Problem 6 Technical Analysis

Technical Issue (Level):

Level 6: Need Balance

Framing 3: Progressive Sophistication