Decoding Strategies

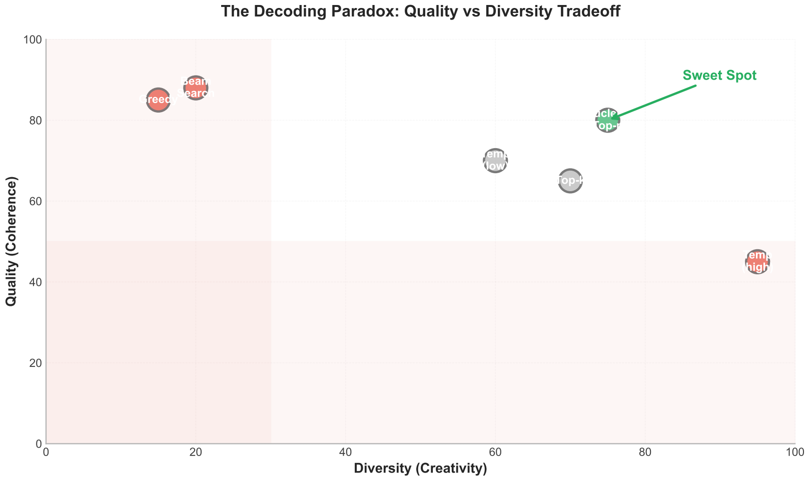
Week 9 - From Greedy to Creative (Enhanced 2025)

NLP Course 2025

November 9, 2025

Two-Tier BSc Discovery — Enhanced with Contrastive Search

The Quality-Diversity Tradeoff



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

The central challenge: How to balance coherence with creativity

Three Decoding Families

Deterministic Methods:

- Greedy
- Beam search

Traits:

- Same output always
- High quality
- No creativity

Stochastic Methods:

- Temperature
- Top-k
- Nucleus (top-p)

Traits:

- Random sampling
- Creative
- Can be chaotic

Balanced Methods:

- Contrastive
- Hybrid

Traits:

- Best of both
- Avoid repetition
- Modern standard

Different tasks need different strategies - no single best method

Greedy Decoding: The Baseline

How It Works:

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

Example:
$P(\text{cat}) = 0.45 \leftarrow$ Pick this!
$P(\text{dog}) = 0.30$
$P(\text{bird}) = 0.25$

When to Use:

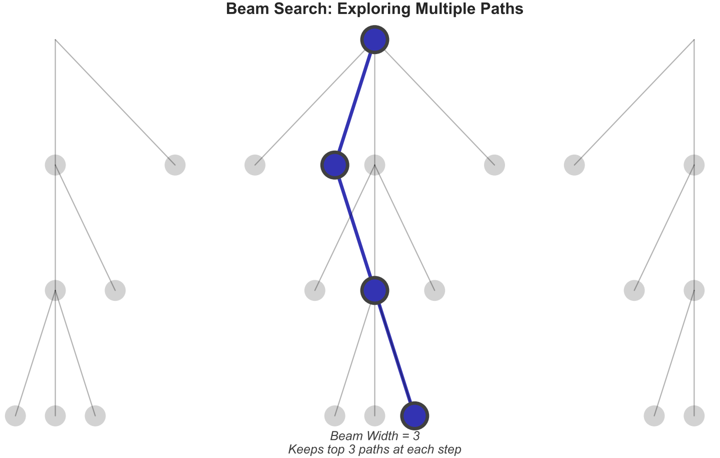
- Code generation (correctness critical)
- Short responses
- Speed is critical
- Need reproducibility

Problem:
Always picks same word $\rightarrow$ repetitive text
"The city is a city in a city..."

Degeneration: Model gets stuck in loops

Simplest method but prone to repetition - need better strategies

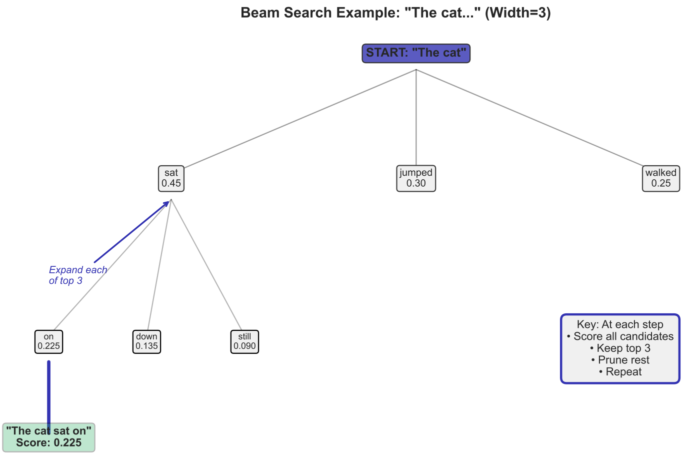
Beam Search: Explore Multiple Paths



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

Beam width = 3-5 typical. Balance between greedy (1) and exhaustive (all)

Beam Search: Step-by-Step Example



Beam Search: Detail

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:

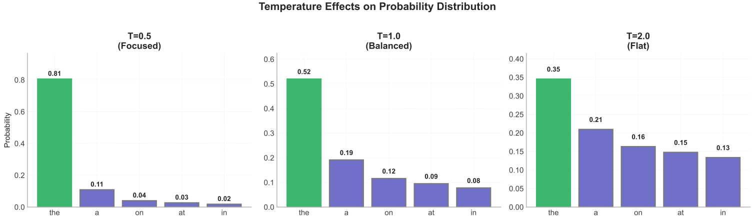
- Machine translation
- Summarization
- Question answering
- Tasks with “correct” answer

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

Tradeoffs:
+ Better quality than greedy
+ Diverse hypotheses
- Still deterministic
- 4-5x 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 Calculation: Step-by-Step

Given: Logits = [2.0, 1.0, 0.5, 0.2]

Tokens = ["cat", "dog", "bird", "fish"]

Step 1: Scale by T=0.5 (MORE PEAKED)

Scaled = [2.0/0.5, 1.0/0.5, 0.5/0.5, 0.2/0.5]

= [4.0, 2.0, 1.0, 0.4]

Softmax → [0.73, 0.18, 0.07, 0.02]

→ 73% on "cat" (VERY FOCUSED)

Step 2: Scale by T=1.0 (UNCHANGED)

Scaled = [2.0, 1.0, 0.5, 0.2]

Softmax → [0.53, 0.19, 0.12, 0.10]

→ 53% on "cat" (BALANCED)

Step 3: Scale by T=2.0 (FLATTER)

Scaled = [1.0, 0.5, 0.25, 0.1]

Softmax → [0.38, 0.23, 0.17, 0.15]

→ 38% on "cat" (MUCH FLATTER)

General Formula:

$$\boxed{p_i = \frac{\exp(\text{logit}_i/T)}{\sum_j \exp(\text{logit}_j/T)}}$$

Lower T → More confident (peaky)
Higher T → More random (flat)

Concrete numbers show how temperature scaling works

Temperature: Detail

How It Works:
Given logits $z_1, z_2, \ldots, z_V$

Scale by temperature T:

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

Sample from p

Effect:
$T \to 0$: argmax (greedy)
$T = 1$: standard softmax
$T \to \infty$: uniform

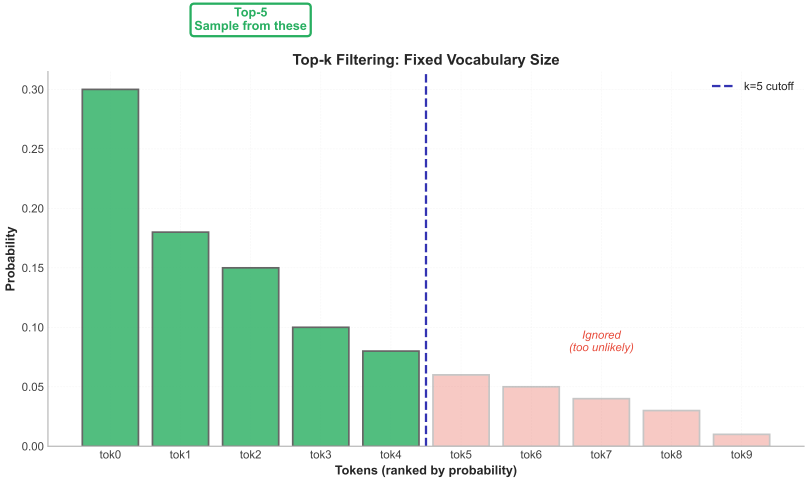
Practical Settings:

- **T = 0.1-0.3**: Factual Q&A
- **T = 0.7**: Chatbots
- **T = 0.9-1.2**: Creative writing
- **T = 1.5+**: Experimental

Tradeoffs:
+ Simple to implement
+ Continuous control
- No quality guarantee
- Can generate nonsense

Temperature is the 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

Top-k: Worked Example (k=3)

Top-k Example: k=3

1. Original Probabilities:

cat: 0.45
dog: 0.18
bird: 0.15
fish: 0.10
mouse: 0.08
...: 0.04

→

2. Keep Top-3:

cat: 0.45
dog: 0.18
bird: 0.15
(discard rest)

→

3. Renormalize:

Sum = 0.45 + 0.18 + 0.15 = 0.78

cat: 0.45/0.78 = 0.58
dog: 0.18/0.78 = 0.23
bird: 0.15/0.78 = 0.19

Result: Sample from {cat: 58%, dog: 23%, bird: 19%}

Prevents sampling from long tail ("mouse" eliminated)

Filtering + renormalization prevents tail sampling

Top-k: Detail

Algorithm:

1. Compute $P(w_i)$ for all tokens
2. Sort by probability
3. Keep only top-k
4. Renormalize: $p_i' = p_i / \sum_{j=1}^{k} p_j$
5. Sample from p'

Example (k=3):
Original: [0.45, 0.18, 0.15, 0.10, ...]
Keep: [0.45, 0.18, 0.15]
Renormalize: [0.58, 0.23, 0.19]

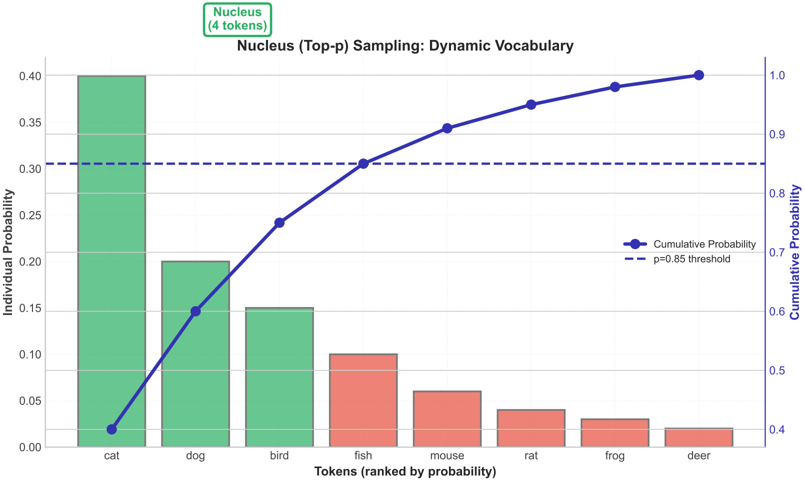
Typical Values:
k = 40-50 (balanced)
k = 10-20 (focused)
k = 100+ (very diverse)

Limitation:
Fixed cutoff regardless of distribution!
Peaked: Wastes probability mass
Flat: Still allows too many bad tokens

Solution: Dynamic cutoff (nucleus)

Top-k improves over temperature but fixed cutoff is inflexible

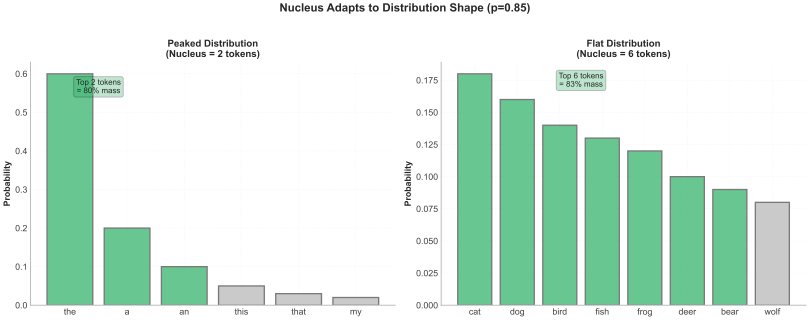
Nucleus (Top-p) Sampling: Dynamic Cutoff



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

Nucleus (Top-p): Detail

Algorithm:

1. Sort tokens by $P(w_i)$ (descending)
2. Compute cumulative sum: $\text{cum}_j = \sum_{i=1}^{j} p_i$
3. Find smallest set where cum $\geq p$
4. Sample from this nucleus

Example (p=0.85):
Probs: [0.40, 0.20, 0.15, 0.10, ...]
Cumsum: [0.40, 0.60, 0.75, 0.85, ...]
Nucleus: First 4 tokens ($0.85 \geq 0.85$)

Recommended Settings:

- **p = 0.85-0.90**: Dialogue
- **p = 0.90-0.95**: Creative writing
- **p = 0.95-0.99**: Very diverse

Why Better:
Peaked distribution → small nucleus (2-3 tokens)
Flat distribution → large nucleus (10+ tokens)
Adapts automatically!

Current Standard: ChatGPT, Claude, GPT-4 all use nucleus

Nucleus sampling is the modern standard for high-quality generation

The Degeneration Problem

The Degeneration Problem: Model Repetition

Real Output from Greedy Decoding:

"The city of New York is a major city in the United States. The city is known for its diverse culture and the city has many tourist attractions. The city is also home to the city's financial district..."

Problem: "the city" appears 6 times in 4 sentences!

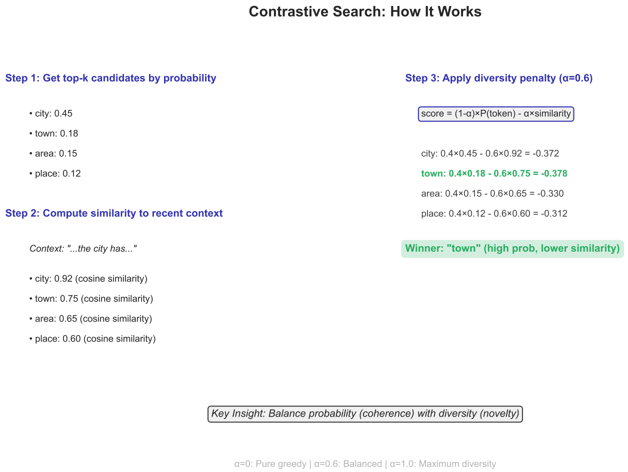
Why? Always picking argmax → same patterns repeated

Solution: Penalize tokens similar to recent context (Contrastive Search)

Discovery Question: Why do models repeat themselves?

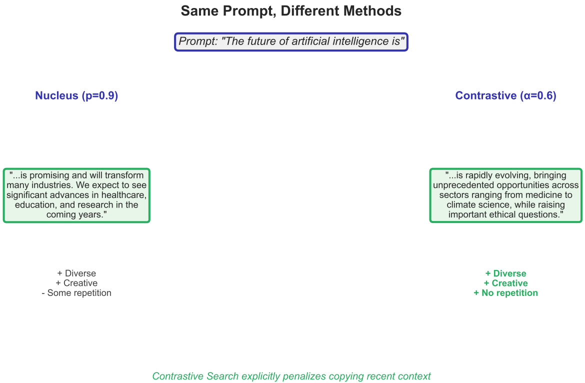
Greedy and beam search maximize probability - but high probability = repeating recent context

Contrastive Search: Penalize Repetition



Key Insight: Balance probability with diversity penalty

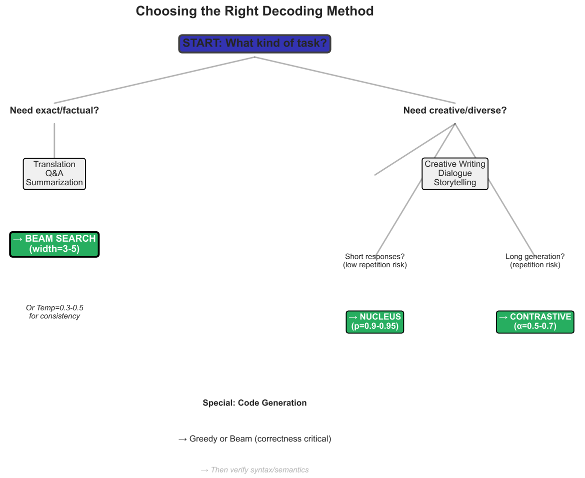
Contrastive vs Nucleus: Comparison



Contrastive search prevents repetition better than nucleus for long generation

All Methods on Quality-Diversity Space

Choosing the Right Method: Decision Tree



Start with task requirements, follow tree to recommended method

Task-Specific Recommendations (2025)

Task-Specific Decoding Recommendations (2025)

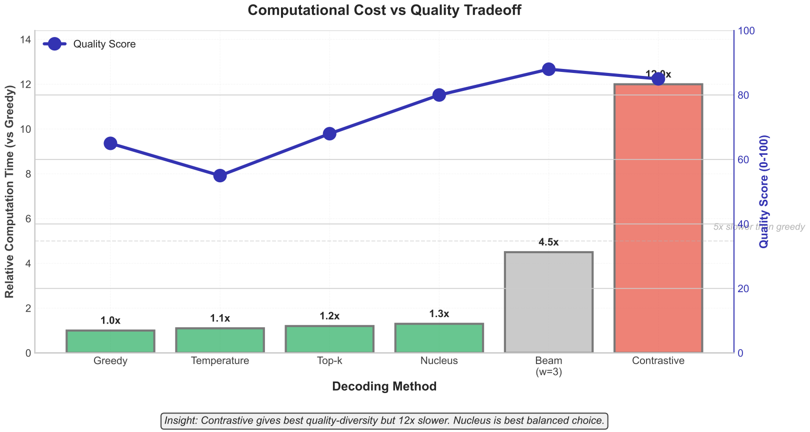
Deterministic Tasks

Task	Recommended Method	Parameters	Why?
Machine Translation	Beam Search	width=3-5	Deterministic, quality critical
Factual Q&A	Greedy / Low Temp	T=0.1-0.3	Single correct answer needed
Summarization	Beam Search	width=4	Balance coverage + conciseness
Code Generation	Greedy	T=0	Syntax errors costly
Creative Writing	Nucleus / Contrastive	p=0.9, α=0.6	Diverse but coherent
Dialogue Systems	Nucleus	p=0.85-0.95	Natural variation needed
Story Generation	Contrastive	α=0.5-0.7	Avoid repetition in long text
Long-form Articles	Contrastive	α=0.6, p=0.9	Degeneration prevention

Creative Tasks

Comprehensive mapping from 8 common tasks to optimal decoding strategies

Computational Costs Matter



Tradeoff: Contrastive gives best quality-diversity but 12× slower

Nucleus is the best balanced choice for most applications

Key Takeaways

1. **Deterministic** (Greedy, Beam): High quality, no diversity - for 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 to distribution
5. **Contrastive**: Explicit degeneration prevention - best for long creative text
6. **Task matters**: Translation $\rightarrow$ Beam — Dialogue $\rightarrow$ Nucleus — Stories $\rightarrow$ 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 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 $\arg\max P(y|x)$

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:

$T \to 0$: $p_i \to \begin{cases} 1 & \text{if } i = \arg\max z \\ 0 & \text{otherwise} \end{cases}$ (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 $\sigma =$ 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)

Production Decoding Settings (Real Systems 2024-2025)

System (2024-2025)	Method	Parameters	Goal
GPT-3 API (2024)	Nucleus	T=0.7, p=1.0	Balanced default
ChatGPT	Nucleus + Temp	T=0.8, p=0.95	Creative but controlled
Google Translate	Beam Search	width=4	Quality critical
GitHub Copilot	Greedy	T=0	Code correctness
Claude	Nucleus	T=1.0, p=0.9	High quality generation
Hugging Face Default	Greedy	T=1.0	Deterministic baseline

Real-world settings from major production systems

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$\times$ 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