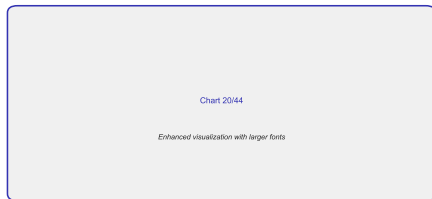


LLM-Based Summarization

NLP Course 2025

November 14, 2025

RAG-Enhanced Summarization



Key insight for rag-enhanced summarization

Retrieval-Augmented Generation

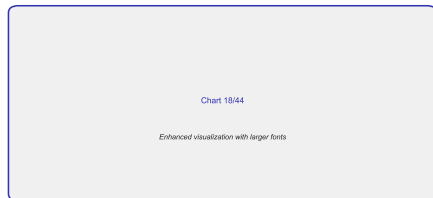
- Ground summaries in retrieved facts
- Reduce hallucinations significantly
- Enable citation tracking
- Better for technical domains

When to Use:

- Medical/legal summaries
- Multi-document synthesis
- Fact-critical applications

RAG ensures factual grounding by retrieving relevant chunks before generation

Hallucination Types Taxonomy



Key insight for hallucination types taxonomy

Failure Types by Frequency:

1. **Extrinsic hallucination**
 - Adding information not in source
 - "FDA-approved" when not mentioned
2. **Factual errors**
 - Wrong numbers, dates, names
 - "100 participants" → "1000"
3. **Missing information**
 - Key findings omitted
 - Context lost in compression
4. **Style mismatch**
 - Too casual/formal
 - Wrong audience level

Understanding failure modes is key to improving summarization quality

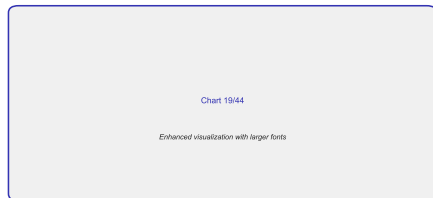
Failure Modes Decision Tree

Chart 21/44

Enhanced visualization with larger fonts

Key insight for failure modes decision tree

Fact Checking Pipeline



Key insight for fact checking pipeline

Prevention Strategies:

1. Parameter Optimization

- Temperature: 0.3-0.5
- Top-p: 0.9
- Repetition penalty: 1.1

2. Prompt Engineering

- Clear instructions
- Length constraints
- Style examples

3. Post-Processing

- Fact checking
- Length validation
- Consistency checks

Combining prevention and detection strategies ensures high-quality output

What We Learned

Core Techniques:

- Prompt strategies (zero/few-shot)
- Parameter control (T, p, repetition)
- Context handling (chunking, RAG)
- Error detection & mitigation

Best Practices:

- Start with $T=0.7$, $p=0.9$
- Use few-shot for consistency
- Implement fact checking
- Monitor for hallucinations

Production Checklist

Model Selection:

- GPT-3.5: Speed/cost
- GPT-4: Quality
- Claude: Long context
- FLAN-T5: Open source

Quality Assurance:

- Automated fact checking
- Human review sampling
- A/B testing
- Error monitoring

Performance:

- Parallel processing
- Caching strategies
- Batch operations

LLM summarization is powerful but requires careful configuration and monitoring

End of Main Presentation

See Appendices for Technical Details & Worked Examples

Technical Appendix

Mathematical Foundations & Worked Examples

Worked Example 1: Temperature Calculation

Problem: Given logits [2.0, 1.0, 0.5], calculate probabilities at $T=0.5, 1.0, 2.0$

Formula: $P(w_i) = \frac{e^{\text{logit}_i / T}}{\sum_j e^{\text{logit}_j / T}}$

T=0.5 (Sharp)

$$\text{logits} / T = [4.0, 2.0, 1.0]$$

$$e^{\text{logits} / T} = [54.6, 7.4, 2.7]$$

$$\sum = 64.7$$

$$P = [0.844, 0.114, 0.042]$$

Most deterministic

T=1.0 (Default)

$$\text{logits} / T = [2.0, 1.0, 0.5]$$

$$e^{\text{logits} / T} = [7.4, 2.7, 1.6]$$

$$\sum = 11.7$$

$$P = [0.632, 0.231, 0.137]$$

Balanced

T=2.0 (Smooth)

$$\text{logits} / T = [1.0, 0.5, 0.25]$$

$$e^{\text{logits} / T} = [2.7, 1.6, 1.3]$$

$$\sum = 5.6$$

$$P = [0.482, 0.286, 0.232]$$

More random

Insight: Higher temperature \rightarrow more uniform distribution \rightarrow more creative output

Temperature directly controls the sharpness of the probability distribution

Worked Example 2: Top-p (Nucleus) Sampling

Problem: Apply top-p=0.9 to vocabulary with these probabilities

Original Distribution:

Word	Probability
"excellent"	0.35
"great"	0.25
"good"	0.15
"amazing"	0.10
"fantastic"	0.08
"wonderful"	0.04
"superb"	0.02
"brilliant"	0.01

Top-p=0.9 Process:

1. Sort by probability
2. Calculate cumulative sum:
 - 0.35 → include
 - 0.60 → include
 - 0.75 → include
 - 0.85 → include
 - 0.93 → **STOP** (> 0.9)
3. Keep top 5 words
4. Renormalize: divide by 0.93

Result: Only sample from ["excellent", "great", "good", "amazing", "fantastic"]

Insight: Top-p dynamically adjusts vocabulary size based on confidence

Nucleus sampling prevents sampling from the long tail of unlikely words

Worked Example 3: Beam Search (width=3)

Problem: Generate next 3 tokens with beam search, starting from "The"

Step 1: From "The"

Token	Score
"cat"	0.4
"dog"	0.3
"bird"	0.2
"fish"	0.1

Keep: cat, dog, bird

Step 2: Expand each

- "The cat" + "sat": $0.4 \times 0.5 = 0.20$
- "The cat" + "ran": $0.4 \times 0.3 = 0.12$
- "The dog" + "barked": $0.3 \times 0.6 = 0.18$
- "The dog" + "ran": $0.3 \times 0.3 = 0.09$
- "The bird" + "flew": $0.2 \times 0.7 = 0.14$

Keep top 3 sequences

Step 3: Final Best sequences:

1. "The cat sat" (0.20)
2. "The dog barked" (0.18)
3. "The bird flew" (0.14)

Winner: "The cat sat"

Total considered: 12 paths

Pruned: 9 paths

Significant reduction

Insight: Beam search balances exploration with computational efficiency

Beam width controls the trade-off between quality and speed

Worked Example 4: Repetition Penalty Application

Problem: Apply repetition penalty $\alpha = 1.3$ after generating "study shows that"

Original Probabilities:

Next Word	$P(w)$
"the"	0.20
"study"	0.15
"research"	0.15
"shows"	0.12
"indicates"	0.10
"reveals"	0.08
Others	0.20

After Penalty ($\alpha = 1.3$):

Words already used: {"study", "shows"}

$$P_{adj}(\text{"study"}) = 0.15 / 1.3 = 0.115$$

$$P_{adj}(\text{"shows"}) = 0.12 / 1.3 = 0.092$$

$$P_{adj}(\text{others}) = \text{unchanged}$$

Renormalize:

- Sum = 0.957
- Divide all by 0.957

Final: "the" = 0.209, "research" = 0.157,
"indicates" = 0.104

Result: Reduced probability for repeated words, encourages variety

Insight: Penalty ≥ 1.0 reduces repetition; < 1.0 encourages it (rare use case)

Repetition penalty is essential for natural-sounding summaries