# LLM-Based Summarization

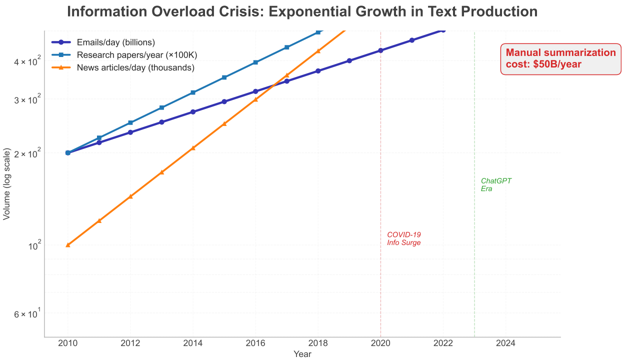
## From Human Effort to LLM Automation

NLP Course 2025

November 15, 2025

**BSc Discovery-Based Presentation - 36 slides**

# The Information Overload Crisis




**Your Daily Reality:**

- 500+ unread emails
- 100-page reports to review
- 50 research papers to analyze

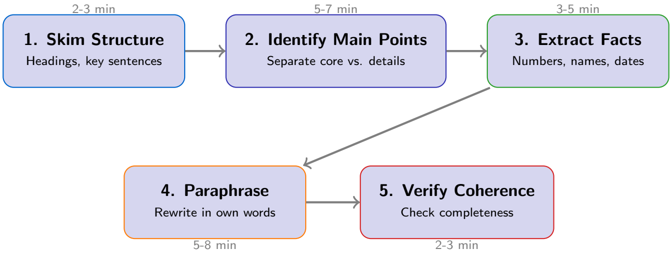
**Industry Scale:**

- Legal firm: 1000 contracts/month
- News agency: 500 articles/day
- Hospital: 50 patient histories/day

**Question:** How long would it take you to process 500 emails manually?

**Information production grows 20%/year, but reading speed doesn't**

# How Humans Summarize - The Cognitive Process




**Which steps are hardest? Which take longest?**

---

**Experts develop mental shortcuts, but still limited by working memory**

# Human Strengths vs. Weaknesses

**Human Strengths**

- Context understanding
  - Detect sarcasm, irony
  - Understand implications
- Audience adaptation
  - Technical vs. general
  - Formal vs. casual
- Judgment calls
  - What's truly important
  - Relevance filtering
- Cross-reference ability
  - Connect disparate ideas
  - Synthesize themes

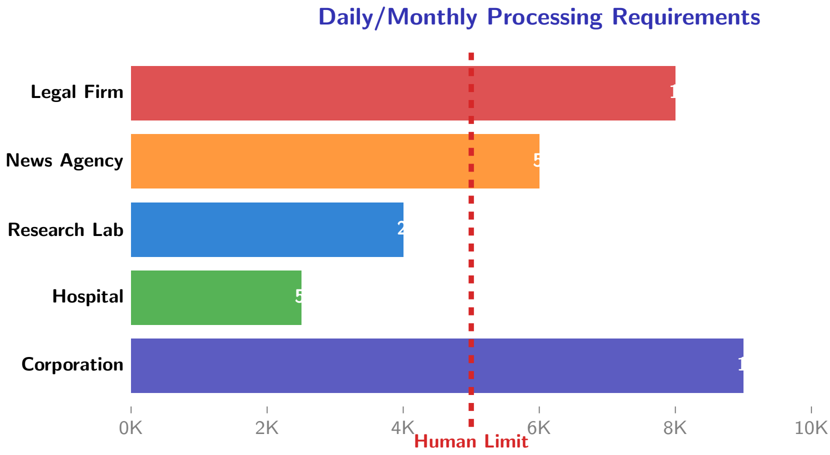
**Human Weaknesses**

- Speed limitations
  - 15-30 min per document
  - Cognitive fatigue
- Consistency issues
  - Quality varies with mood
  - Different styles between people
- Scale problems
  - Can't process 1000/day
  - Bottleneck in workflows
- Bias introduction
  - Personal interpretation
  - Selective attention

**Example:** Legal contract summary - Lawyer ($200/hr) takes 1 hour for 50-page contract.
Firm needs 1000/month = **$200,000/month** in labor costs!

**Human expertise is valuable but doesn't scale to modern volumes**

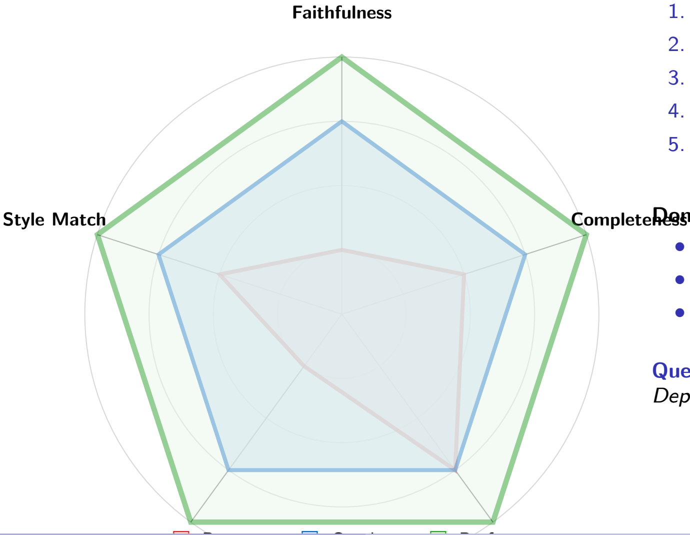
# The Scale Problem - Volume Requirements




**Critical Insight:** News agency needs 500 articles × 15 min each = 125 hours/day.
Would require **16 full-time summarizers** working non-stop!

**The bottleneck isn't quality - it's throughput**

# What Makes a Good Summary?



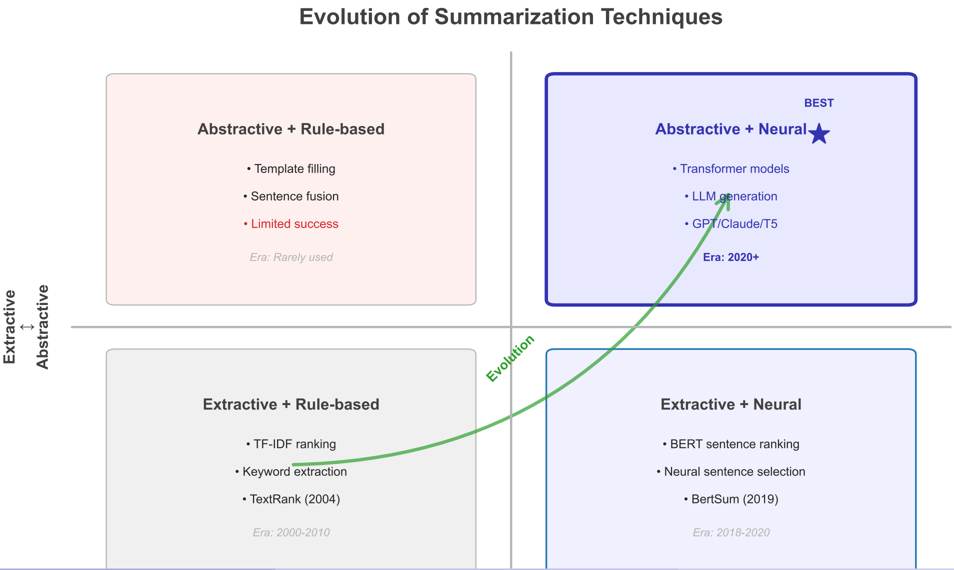

**Five Quality Dimensions:**

1. **Faithfulness**: No added information
2. **Completeness**: Key points included
3. **Conciseness**: Appropriate length
4. **Coherence**: Natural flow
5. **Style Match**: Audience-appropriate

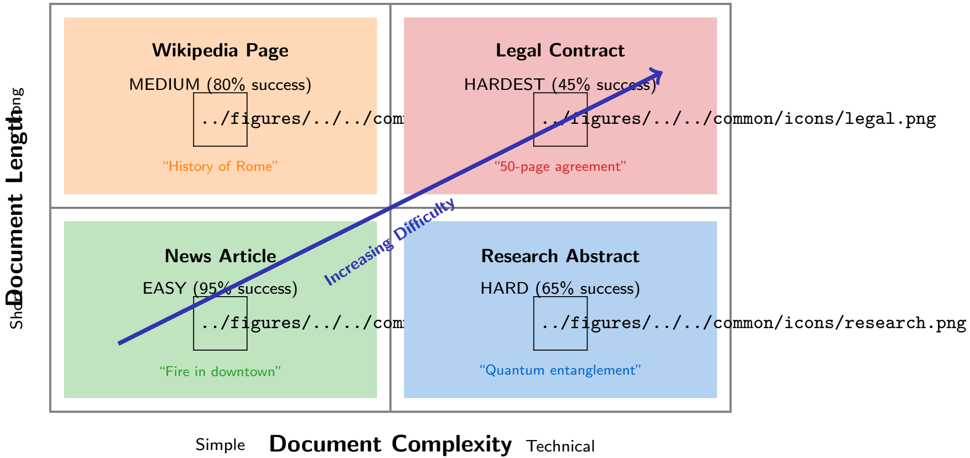
**Domain Priorities:**

- Legal: Faithfulness = 100%
- News: Conciseness = 100%
- Medical: Completeness = 100%

**Question:** Which dimension matters most?
*Depends on use case!*


Evolution of Summarization Techniques
Abstractive + Rule-based
• Template filling
• Sentence fusion
• Limited success
Era: Rarely used
BEST
Abstractive + Neural
• Transformer models
• LLM generation
• GPT/Claude/T5
Era: 2020+
Extractive ↔ Abstractive
Evolution
Extractive + Rule-based
• TF-IDF ranking
• Keyword extraction
• TextRank (2004)
Era: 2000-2010
Extractive + Neural
• BERT sentence ranking
• Neural sentence selection
• BertSum (2019)
Era: 2018-2020

# The Challenge Matrix




**Discovery Question:** What makes some summaries harder than others?

**Different challenges require different prompting strategies**

**Question:**

Which quality dimension is hardest for automated systems to measure?

**A**

Summary length
(word count)

**B**

Faithfulness
(no hallucinations)

**C**

Compression ratio
(input/output)

**D**

Word overlap
(ROUGE score)

## Question:

Which quality dimension is hardest for automated systems to measure?

**A**

Summary length
(word count)

**B**

Faithfulness
(no hallucinations)

**C**

Compression ratio
(input/output)

**D**

Word overlap
(ROUGE score)

## Attention Mechanism Visualization

Chart 15/44

*Enhanced visualization with larger fonts*

Key insight for attention mechanism visualization

# Encoder-Decoder Process

### Encoder (Understanding)

- Input tokens $\rightarrow$ embeddings
- Self-attention layers (6-12)
- Learn document structure
- Identify salient information
- Output: Encoded representations

### Decoder (Generation)

- Start with [CLS] token
- Cross-attend to encoder
- Generate tokens autoregressively
- Stop at [SEP] or max length
- Output: Summary tokens

**Example Input:**
"Complex study findings with statistical significance p¡0.01..."

**Example Output:**
"Study shows significant results (p¡0.01)"

Encoder $\rightarrow$ Decoder flow

**Key:** Decoder can *rephrase*, not just copy!

**Encoder-decoder architecture (BART, T5) vs. decoder-only (GPT, Claude)**

# Text to Tokens Pipeline

**Raw Text:** "The study finds significant results."

↓

**Tokens:** ["The", "study", "finds", "sign", "ificant", "results", "."]

Special: [CLS], [SEP], [PAD]

↓

**Token IDs:** [102, 2025, 4532, 3241, 8475, 3421, 119]

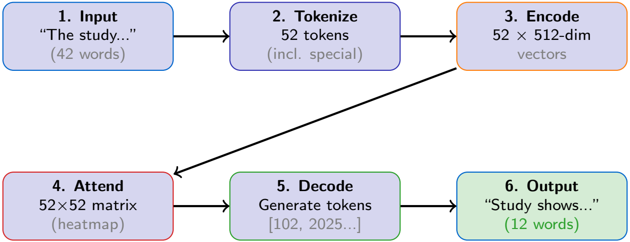
↓

**Embeddings:** 512-dimensional vectors (shown as 3D projection)

**Why split "significant" into "sign" + "ificant"?**
Rare word: "immunotherapy" $\rightarrow$ ["immuno", "therapy"] (subword units)
Benefit: Vocabulary of 30K handles millions of words

**BPE/WordPiece balance vocabulary size vs. coverage**

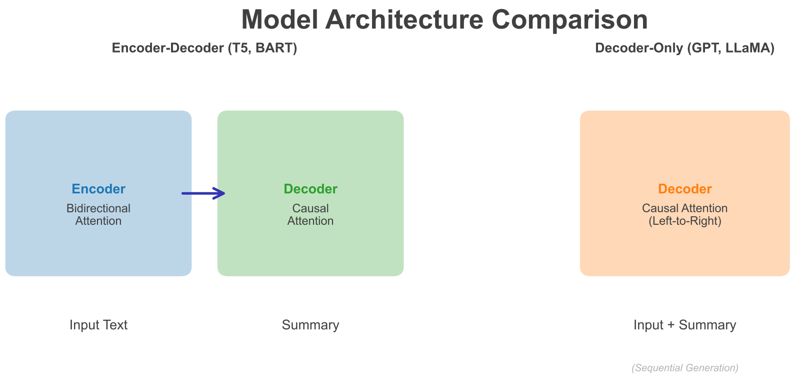
# Complete Pipeline with Concrete Numbers




**Compression: 42 → 12 words (3.5:1)**

**Where does compression happen?**
Attention focuses on: "study", "shows", "significant", "results"
Ignored: "the", "of", "over the course of", etc.

**Each step is learned from data, not programmed - no rules!**



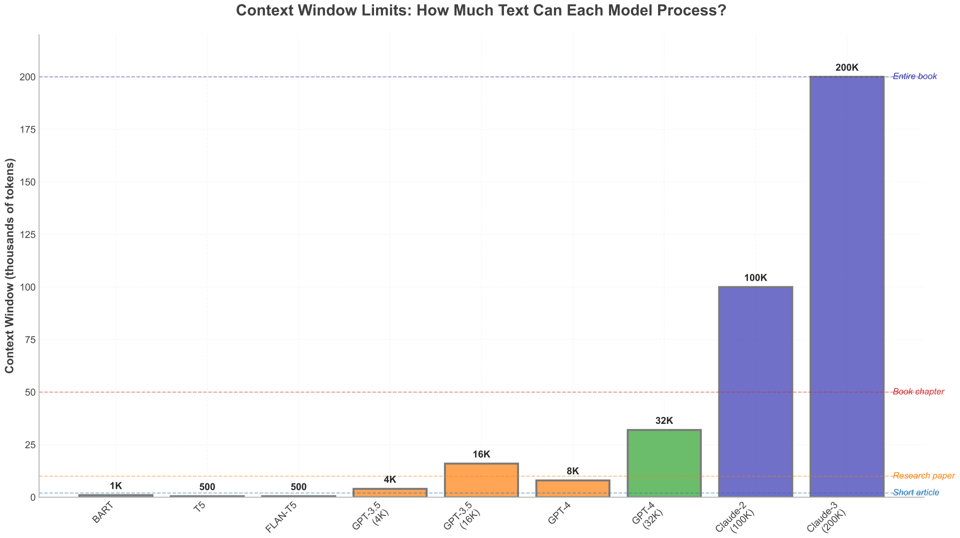

**Example Use Cases:**

News summaries (short, high volume) → FLAN-T5

# Model Selection Decision Tree

../figures/model_selection_decision_tree_bsc.pdf

# Context Window Limits



# Worked Example: Token Flow

**Input (32 words):**
"A recent study examined 1,000 patients with Type 2 diabetes over a 5-year period. The results showed a 30% reduction in complications for those following the new treatment protocol."

**Tokenization:**
Token IDs: [102, 138, 2332, 4521, 1000, ...]
(First 5 of 40 tokens shown)

**Attention Focus:**
Strong weights on:
"1,000", "patients", "5-year",
"30%", "reduction", "complications"

**Generation:**
Output tokens: [102, 2025, 4532, ...]
(Decoded to text below)

**Final Output (17 words):**
"Study of 1,000 diabetes patients found 30% reduction in complications with new treatment over 5 years."

| Metric | Value | Quality |
|---|---|---|
| Compression | 32 $\rightarrow$ 17 words (53%) | ✓ |
| Faithfulness | All numbers correct | ✓ |
| Coherence | Grammatical, flows well | ✓ |

**This is abstractive summarization - paraphrasing, not copying**

**Question:**

You have a 50-page legal document ( 40K words)
and want to use GPT-3.5 (4K token limit).
What should you do?

**A**

It will automatically
compress the input

**B**

The model will
fail with an error

**C**

Use chunking
strategies

**D**

Switch to Claude
(100K context)

# Checkpoint Quiz 2

**Question:**

You have a 50-page legal document ( 40K words)
and want to use GPT-3.5 (4K token limit).
What should you do?

**A**

It will automatically
compress the input

**B**

The model will
fail with an error

**C**

Use chunking
strategies

**D**

Switch to Claude
(100K context)

# Beyond Basic Prompting - Error Distribution

**Common Failures:**

- 35% Hallucinations
  Adding fake information
- 20% Missing Info
  Omitting key points
- 15% Length Issues
  Too short/long
- 20% Style Problems
  Wrong formality
- 10% Factual Errors
  Wrong numbers

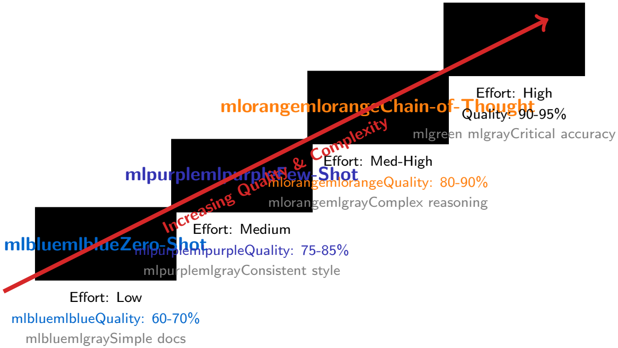
**Example Hallucination:**
Input: "Study of 100 patients..."
Output: "Study of 1,000 patients showed FDA approval..."

**With basic prompting, 35% have hallucinations - Unacceptable!**
Need advanced techniques: few-shot, CoT, RAG to fix these

**Need advanced techniques to handle these failure modes**

# Prompting Strategies - Sophistication Ladder

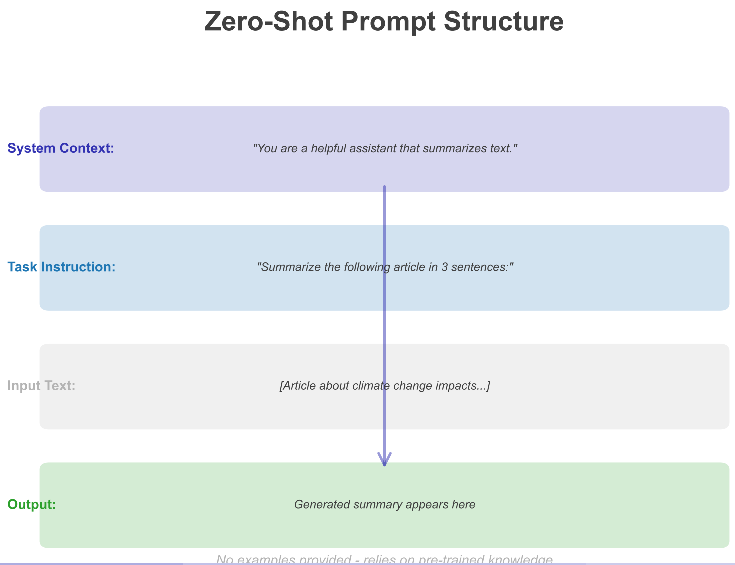
## Start Simple, Add Complexity Only When Needed




**Examples:** News article → Level 1-2 sufficient — Medical record → Level 4 required

**Don't over-engineer - match technique to requirements**

# Zero-Shot Prompting



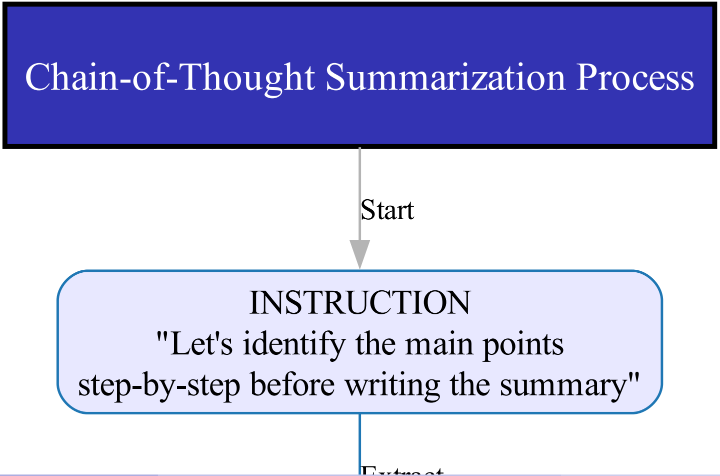
## Few-Shot Prompt with Examples

**System:** *"You are a helpful summarizer"*

**Example 1:**
*Input: [Long text...]*
*Output: [Summary...]*

**Example 2:**
*Input: [Long text...]*
*Output: [Summary...]*

**Task:**
*Input: [New article to summarize]*


Chain-of-Thought Summarization Process
Start
INSTRUCTION
"Let's identify the main points
step-by-step before writing the summary"

# Chain-of-Thought Variants - Latest Research

## Chain-of-Thought Variants Comparison (2024-2025 Research)

| Variant | Key Feature | Best For | Example Prompt |
|---|---|---|---|
| **Standard CoT** | "Let's think step-by-step" | General reasoning<br>All summaries | "Identify main points, then summarize" |
| **Contrastive CoT** | Show wrong example too | Avoiding specific errors | "Good: factual summary<br>Bad: hallucinated facts<br>Now summarize correctly" |
| **Thread-of-Thought** | Multi-part processing | Long RAG contexts | "Walk through document in parts, summarizing as we go" |
| **Faithful CoT** | Verify each step | Critical accuracy (medical/legal) | "Extract claim.<br>Verify in source.<br>Then summarize." |

*Note: These variants emerged from recent research on improving LLM reasoning*

# RAG-Enhanced Summarization

Chart 20/44

*Enhanced visualization with larger fonts*

Key insight for rag-enhanced summarization

# When and How to Use RAG

**When to Use RAG:**

- Multi-document summarization
- Factual domains (medical, legal)
- Citation tracking needed
- Very long documents (¿context)
- Query-focused summaries

**RAG Components:**

- Embedding model (sentence-transformers)
- Vector database (FAISS, Pinecone)
- Retriever (BM25, dense retrieval)
- Reranker (Cross-encoder)
- Generator (GPT-4, Claude)

**Example - Medical Paper:**

**Input:** 30-page paper + 50 cited papers

**Without RAG:**
"Paper discusses treatment..." (generic)

**With RAG:**
"Paper shows 30% improvement (Smith 2020), confirmed by Jones 2021, contradicts Brown 2019..."

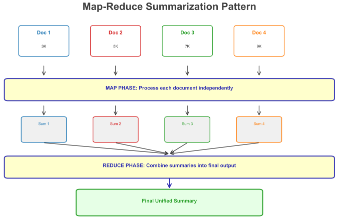
Value: Verifiable, grounded

**Trade-off:** RAG adds complexity
Worth it for critical applications

**Requires infrastructure (vector DB) but essential for critical applications**

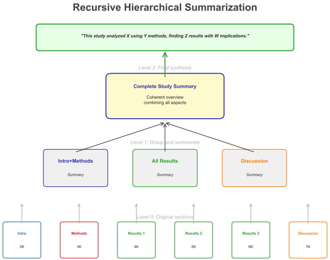
# Three Strategies for Long Documents

## Map-Reduce



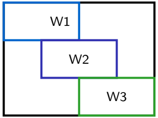

- ✓ Parallelizable
- ✓ Preserves structure
- × May lose cross-chunk connections

## Recursive Hierarchical




- ✓ Handles any length
- ✓ Maintains hierarchy
- × Slow, iterative

## Sliding Window




- ✓ Catches cross-boundary
- ✓ Flexible coverage
- × Complex to implement

**Which strategy for a 100-page report?**

Answer: Map-Reduce for speed (parallel) — Risk: Miss connections between chapters

**Choice depends on document structure - narrative vs. sectioned**

# Worked Example: Map-Reduce in Action

**Input:** 20-page research report

**Stage 1: Split**

- Pages 1-4: Intro
- Pages 5-8: Methods
- Pages 9-12: Results
- Pages 13-16: Discussion
- Pages 17-20: Conclusion

**Stage 2: Map (Summarize)**

$\rightarrow$ "Report introduces..."
$\rightarrow$ "Methods include RCT..."
$\rightarrow$ "Results show 30%..."
$\rightarrow$ "Discussion highlights..."
$\rightarrow$ "Conclusion recommends..."

5 summaries $\times$ 100 words $=$ 500 words total

**Stage 3: Reduce (Combine)**

"Report on treatment effectiveness. RCT of 1,000 patients showed 30% reduction. Despite limitations, adoption recommended."

Final: 50 words

| Stage | Words | Tokens |
|---|---|---|
| Input | 10,000 | 13,000 |
| After Map | 500 | 650 |
| After Reduce | 50 | 65 |

**Compression ratio:** 13,000 $\rightarrow$ 65 $=$ **200:1**

**Preserves structure but may lose connections between sections**

## Question:

You need to summarize a 100-page legal contract where clauses reference each other across sections. Which technique should you use?

**A**

Zero-shot prompting
(simple)

**B**

Few-shot prompting
(with examples)

**C**

RAG with
citation tracking

**D**

Map-reduce with
overlap + CoT

## Question:

You need to summarize a 100-page legal contract where clauses reference each other across sections. Which technique should you use?

**A**

Zero-shot prompting
(simple)

**B**

Few-shot prompting
(with examples)

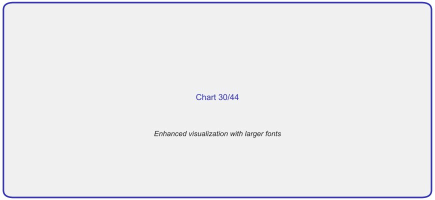
**C**

RAG with
citation tracking

**D**

Map-reduce with
overlap + CoT

Evaluation Metrics (ROUGE, BERTScore)




Key insight for evaluation metrics (rouge, bertscore)

**The Evaluation Pyramid**

**Level 1: Surface Metrics**

- Length compliance
- Format checking
- Speed/cost

**Level 2: Content Metrics**

- ROUGE scores (overlap)
- BERTScore (semantic)
- Factuality checks

**Level 3: Quality Metrics**

- Human evaluation
- LLM-as-judge
- Task-specific measures

**Good evaluation requires multiple metrics at different levels**

# ROUGE: The Classic Metric

**ROUGE-N Calculation**
**Reference:**
"The cat sat on the mat"
**Summary:**
"The cat on mat"

**ROUGE-1 (unigrams):**

- Overlap: {the, cat, on, mat}
- Precision: $4/4 = 1.0$
- Recall: $4/6 = 0.67$
- F1: $2\times(1.0\times0.67)/(1.0+0.67) = 0.80$

**ROUGE-2 (bigrams):**

- Reference: {the cat, cat sat, sat on, on the, the mat}
- Summary: {the cat, cat on, on mat}
- Overlap: {the cat}
- Recall: $1/5 = 0.20$

**ROUGE Variants**

| Metric | Measures |
|---|---|
| ROUGE-1 | Unigram overlap |
| ROUGE-2 | Bigram overlap |
| ROUGE-L | Longest sequence |
| ROUGE-S | Skip-bigrams |

**Pros:**

- Fast to compute
- Language-agnostic
- Interpretable

**Cons:**

- Surface-level only
- Ignores semantics
- Prefers extractive

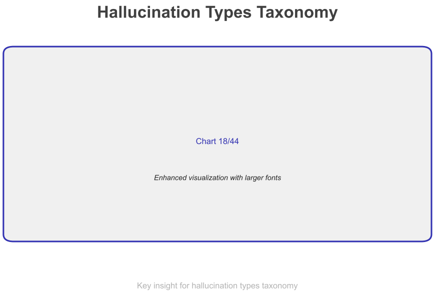
**ROUGE remains popular despite limitations - always combine with other metrics**

# Modern Evaluation Landscape

## Modern Evaluation Metrics Comparison (2024-2025)

| Metric | What it Measures | Human Correlation | Cost | Best For |
|---|---|---|---|---|
| **ROUGE** | Word overlap | **0.45** | Free | Baseline screening |
| **BLEU** | N-gram precision | **0.38** | Free | Translation (not summarization) |
| **BERTScore** | Semantic similarity | **0.72** | Low ($0.01/eval) | Quality filtering |
| **METEOR** | Stemming + synonyms | **0.55** | Free | Improved ROUGE |
| **G-eval** | LLM rates quality (multi-aspect) | **0.85** | Medium ($0.10/eval) | Detailed evaluation |
| **GPT-4 Judge** | Overall quality assessment | **0.92** | High ($0.30/eval) | Final validation |
| **Faithfulness** | Fact verification against source | **0.88** | High ($0.25/eval) | Critical applications |
| **Human Eval** | Gold standard (definition) | **1.00** | Very High ($5-20/eval) | Ground truth |

# When Things Go Wrong



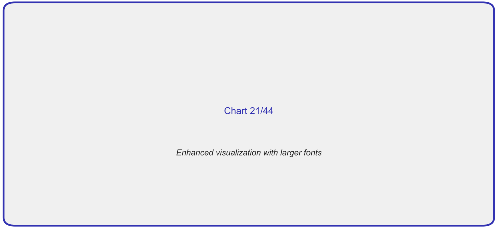

**Common Failure Patterns**

1. **Hallucination (35%)**
   - Adding facts not in source
   - Example: “FDA-approved” when not mentioned
2. **Omission (25%)**
   - Missing key information
   - Example: Ignoring limitations
3. **Distortion (20%)**
   - Changing meaning
   - Example: “may” $\rightarrow$ “will”
4. **Repetition (20%)**
   - Redundant content
   - Example: Saying same thing 3 ways

**Understanding failure modes helps you design better mitigation strategies**

# Debugging Flowchart

**Failure Modes Decision Tree**




Key insight for failure modes decision tree

**Quick Fixes**

**Too Creative?**

- Lower temperature (0.3-0.5)
- Reduce top_p (0.9)
- Add "stick to facts" prompt

**Too Short?**

- Add min_length constraint
- Prompt: "comprehensive summary"
- Use few-shot examples

**Hallucinating?**

- Add RAG verification
- Lower temperature to 0.3
- Post-process fact checking

**Repetitive?**

- Increase repetition_penalty
- Use diverse beam search
- Post-process deduplication

**Systematic debugging saves hours of frustration - follow the flowchart!**

# Complete Pipeline Example

**Task:** Summarize research paper (8 pages)

```
import openai
from rouge import Rouge

def smart_summarize(text, max_tokens=4000):
    # 1. Check length
    if len(text) > max_tokens:
        # 2. Use map-reduce
        chunks = chunk_with_overlap(
            text,
            chunk_size=3000,
            overlap=500
        )

        # 3. Summarize chunks
        summaries = []
        for chunk in chunks:
            summary = openai.chat.completions.create(
                model="gpt-3.5-turbo",
                messages=[{
                    "role": "system",
                    "content": "Summarize:"
                }, {
                    "role": "user",
                    "content": chunk
                }],
                temperature=0.5,
                max_tokens=300
            )
            summaries.append(summary)

        # 4. Combine summaries
        final_text = "␣".join(summaries)
    else:
```

**Configuration Checklist**

**Model Selection:**

- GPT-3.5: Speed/cost priority
- GPT-4: Quality priority
- Claude: Long context (100k+)

**Parameters:**

- temperature: 0.5 (balanced)
- top_p: 0.9 (some creativity)
- max_tokens: 300 per chunk
- repetition_penalty: 1.1

**Quality Checks:**

- Length: 20-30% of original
- ROUGE-L: ¿ 0.35
- No hallucinations (fact check)
- Coherent flow

**Result:**

8 pages → 1.5 pages

ROUGE-L: 0.42

# What We've Learned

**Foundations**

- Information explosion problem
- LLMs vs traditional methods
- Transformer architecture
- Attention is all you need

**Key Techniques**

- Zero/few-shot prompting
- Chain-of-thought
- RAG enhancement
- Map-reduce for scale

**Best Practices**

- Start simple (zero-shot)
- Test systematically
- Monitor for failures
- Use multiple metrics

**Parameters**

- Temperature: 0.3-0.7
- Top-p: 0.9
- Repetition penalty: 1.1
- Chunk overlap: 10-20%

**Next Steps**

1. Try lab notebook
2. Experiment with prompts
3. Build your pipeline
4. Test on your data

**Resources**

- Lab: `week_summarization.ipynb`
- Charts: 87 PDFs in `figures/`
- Scripts: `python/*.py`

**Remember:** LLM summarization is powerful but requires careful configuration. Start simple, iterate based on metrics, and always validate output quality!

**You now have all the tools to build production-ready summarization systems**

# Thank You!

Questions & Discussion

Lab notebooks available in `NLP_slides/summarization_module/lab/`