# Improving RAG Applications

For MIT Generative AI Course

Jason Liu

February 25, 2025

Independent Consultant

# Table of Contents

- About Me
- What are we doing here?
- Practical Implementation Strategies
    - Evaluation Methods
    - Query Segmentation
    - Specialized Retrieval
    - Feedback Collection
    - Implementation Roadmap

# About Me

# About Me

*Goal: To showcase my diverse experience – feel free to ask questions about any area during Q&A*

- Independent Consultant & Staff-level ML Engineer & Educator
  - Meta, Stitchfix, NYU from (2016-2023)
- University of Waterloo
  - B.Math in Mathematical Physics & Computational Mathematics
  - Minor in Statistics (Class of 2017)
- Creator of Instructor - Python library for structured LLM outputs
  - 9500+ GitHub stars
  - 1.5M+ monthly downloads
  - Cited by OpenAI as inspiration for structured output feature
  - Popular ’Pydantic is all you need’ Talk from AI Conference
- a16z Scout & Angel Investor & Startup Advisor

# My Journey

- Transitioned to independent consulting due to RSI (2022)
- Do I want to get 20% of my coding back, or find 100x more leverage?
- Now focused on:
  - Teaching teams to work with AI and be quantitative
  - Writing more 'popular ai' content
  - Advisory work for early stage startups
  - Open source projects, independent research

# Work & Consulting & Advisory Engagements

| Client | Contact | Industry |
|---|---|---|
| Zapier | VP of Product | Automation |
| HubSpot | GM | Sales & Marketing |
| Enterpret | CTO | Analytics |
| Tensorlake | CEO | Data |
| Limitless AI | CTO | AI |
| Trunk Tools | VP Eng | Construction |
| Naro | CTO | Sales & Marketing |

Additionally, I've worked with innovative startups including New Computer, Sandbar, Dunbar, Bytebot, Kay.ai, Raycast, Weights & Biases, Modal Labs, Timescale, and Pydantic on various technical and strategic initiatives.

# Where My Students Come From

| Company | Industry |
|---|---|
| OpenAI | AI Research & Development |
| Anthropic | AI Research & Development |
| Google | Search Engine |
| Salesforce | Customer Relationship Management Software |
| Microsoft | Software, Cloud Computing |
| Amazon | E-commerce, Cloud Computing |
| Zapier | Automation Software |
| Adobe | Software, Creative Tools |
| Accenture | Consulting, Technology Services |
| McKinsey & Company | Management Consulting |
| Bain & Company | Consulting |
| PwC | Professional Services |
| Cisco | Networking Technology |
| Electronic Arts | Gaming |
| Shopify | E-commerce Platform |

# What are we doing here?

# Course Context & Goals

- Learning Objectives
  - Developing durable AI knowledge that outlasts specific technical implementations
  - Understanding ML systems as continuously evolving products rather than ”deploy once and forget”
  - Recognizing the parallels between recommendation systems and retrieval systems
  - Identifying valuable business applications through effective data analysis

# Setting the Context

- Why This Matters Now
  - Democratization of AI tools means the competitive advantage comes from thinking deeply
  - Growing gap between research capabilities and business implementation
  - Science now drives product development, reversing traditional patterns
  - Opportunity for individual contributors to have outsized impact through thoughtful implementation
- Interactive Format
  - This group is quite diverse, so I'll try to keep it broad
  - The goal is to seed you with good questions, rather than dump information
  - We'll leave plenty of time for questions about AI, Business, and Career paths

# Key Questions to Consider

- How has machine learning research and implementation evolved from 2015 to 2025?
- What behavioral practices should teams adopt when working with AI systems?
- How do we identify economically valuable AI applications?
- What's the right balance between research exploration and product implementation?
- How do we design systems that can evolve effectively over time?
- When should you join established labs versus work independently?
- How can individuals and small teams achieve leverage without large resources?
- What skills matter most in the AI era? (Hint: thinking ¿ coding)

# Three Key Arcs We'll Explore

- Technical: From Recommendation to Retrieval Systems
  - The surprising similarities in architecture and challenges
  - Why understanding these parallels helps build better systems
- Organizational: Effective AI Implementation
  - The importance of observability and measurement
  - Balancing unified systems vs. specialized subsystems
- Personal: Career Considerations in AI
  - Information synthesis as a durable skill
  - How AI changes team dynamics and individual contributions

# Practical Implementation Strategies

# Synthetic Data Generation (1/2)

- Use LLMs to generate domain-specific questions from your documents
- Create diverse query types:
  - Factual: "What was the revenue in Q2 2023?"
  - Comparative: "How did Q2 performance compare to Q1?"
  - Analytical: "What factors contributed to the margin decline?"

**Benefits:** Controllable test data, covers edge cases, identifies blind spots

# Synthetic Data Generation (2/2)

- Include edge cases and known failure modes
- Example prompt:

> ”Generate 10 questions a financial analyst might ask about this earnings report, including questions about revenue trends, profitability metrics, and forward guidance.”

- Aim for 100-200 diverse test examples across categories

# Retrieval Metrics (1/2)

- **Recall@k**: Percentage of relevant documents in top k results
  - Critical for RAG - can't generate from missing information
  - Target 80-90% recall before focusing on generation quality
- **MRR (Mean Reciprocal Rank)**: Position of first relevant document
  - Higher weight to documents appearing earlier in results
  - Useful for prioritizing most relevant content

# Retrieval Metrics (2/2)

- **Latency**: Response time for retrieval operations
  - Critical for user experience and scaling
  - Balance between accuracy and speed
- **Coverage**: Percentage of query types that can be answered
  - Identifies gaps in retrieval capabilities
  - Helps prioritize new index or tool development

**Key insight:** Focus on recall first!

Poor recall = ceiling on overall quality

# Week 1: Key Insights

**Evaluation Systems: The Foundation for Improvement**

- **Start with synthetic data** to enable rapid testing cycles
- **Focus on retrieval quality first** before optimizing generation
- **Build fast evaluation pipelines** that run in under a minute
- **Collect 100-200 diverse test examples** across query categories
- **Prioritize recall** as the leading indicator of system quality
- **Log and analyze failures** systematically to identify patterns

**Remember:** You can't generate accurate answers from missing information

# Evaluation Implementation

### Test Size

- 100-200 test examples across categories
- 20-30 examples per major query type
- Include examples from each domain area

### Failure Analysis

- Log all failures for systematic analysis
- Store full context of failed retrievals
- Look for patterns in failure modes

### Automation

- Automate evaluation to run in under 1 minute
- Fast feedback loops enable rapid iteration
- Run after every significant change

# Case Study: Due Diligence Summaries & Missing Experts

### The Challenge

- Consulting firm conducting M&A due diligence
- AI summaries missed half of relevant expert quotes
- Only 3 of 6 experts cited on key topics (50% recall)
- Low recall undermined consultant confidence

### The Solution

- Created "ground truth" set tagging who said what
- Refined chunk boundaries and indexing strategy
- Recall improved from 50% to 90%
- Systematic evaluation led to rapid improvement

**Key Takeaway:** Small "gold set" evaluation + refined chunking dramatically improves system credibility

# Segmentation: The Foundation for Improvement

**Why segment queries?** To identify specific areas for improvement

- Moving beyond basic metrics (recall/precision)
- Identifying patterns in user behavior
- Prioritizing development efforts based on data
- Tracking performance across different query types

**Key insight:** Summary statistics often mask important patterns

# Query Segmentation Approaches (1/2)

- **Intent-based**: What is the user trying to accomplish?
  - Information seeking vs. task completion
  - Exploratory vs. targeted queries
  - Example: "Tell me about X" vs. "How do I do Y?"
- **Domain-based**: Which knowledge area does this touch?
  - Subject matter categories
  - Technical vs. business vs. compliance
  - Example: Financial metrics vs. operational details

# Query Segmentation Approaches (2/2)

- **Complexity-based**: Simple lookup vs. multi-step reasoning
  - Single fact retrieval vs. synthesis across documents
  - Explicit vs. implicit information
  - Example: Direct value lookup vs. trend analysis
- **Data-type**: Text, table, image, code, or multi-modal
  - Different content formats require different handling
  - Example: Narrative text vs. tabular financial data

# Categorizing Issues: Inventory vs. Capabilities

**Inventory Issues**

- Missing content in knowledge base
- Incomplete data sources
- Outdated information
- Gaps in document coverage

**Capability Issues**

- System's functional limitations
- Missing metadata extraction
- Lack of structured filtering
- Insufficient query understanding

**Different solutions:** Inventory → expand corpus
Capabilities → build new functionality

# Detecting Inventory vs. Capability Gaps

## Inventory Gap Signals

- Low cosine similarities
- No results from lexical search
- LLM refusing to answer
- Returned chunks never cited
- Broken data pipelines

## Capability Gap Examples

- Time-based filtering needs
- Comparison across documents
- Structured data extraction
- Missing metadata fields
- Need for specialized indices

**Example:** "Latest contract modifications" requires both inventory (recent docs) and capability (time filtering)

# Implementing Query Classification

- Use a classifier prompt to categorize each query
  - Chain-of-thought prompting improves accuracy
  - Allow multiple categories per query when relevant

**Prompt Example:**

"Analyze this query and determine which category it belongs to. Explain your reasoning before giving your final answer."

# Tracking Segmentation Performance

- Track performance metrics per segment
  - Separate dashboards for each major category
  - Compare performance across segments
- Identify segments with highest volume × lowest performance
  - Focus improvements on high-impact areas
  - Prioritize based on business value

# Prioritization Framework

## Prioritization Formula

- Impact of answering this type of question
- Query volume for the segment
- Likelihood of success (can we solve it?)

$$\boxed{\begin{array}{c}\text{Priority} = \text{Impact} \times \\ \text{Volume} \times \text{P(Success)}\end{array}}$$

## Decision Matrix

- High satisfaction + High volume = Maintain
- High satisfaction + Low volume = Promote
- Low satisfaction + Low volume = Phase out
- Low satisfaction + High volume = Focus here!

# Week 2: Key Insights

**Query Segmentation: The Path to Targeted Improvements**

- **Segment queries** by intent, domain, complexity, and data type
- **Distinguish between inventory and capability gaps**
  - Inventory: Missing content → Add more data
  - Capability: System limitations → Build new features
- **Track performance by segment** to identify specific weaknesses
- **Prioritize improvements** based on impact, volume, and feasibility
- **Focus on high-volume, low-satisfaction segments** first

**Remember:** Summary statistics often mask important patterns

# Financial Query Types (1/2)

- **Numerical extraction**
  - Finding specific values in financial statements
  - Example: "What was the EPS in Q2 2023?"
  - Requires precise table extraction and entity recognition
- **Trend analysis**
  - Identifying patterns over time in financial data
  - Example: "How has the gross margin changed over the last 4 quarters?"
  - Requires time-series data and comparative analysis

# Financial Query Types (2/2)

- **Comparative analysis**
  - Comparing entities, periods, or metrics
  - Example: "How does Company X's ROI compare to industry average?"
  - Requires multi-document retrieval and normalization
- **Risk assessment**
  - Evaluating potential issues or concerns
  - Example: "What are the key risk factors mentioned in the report?"
  - Requires understanding of risk terminology and context

# Real-World Example: Construction Project

## Initial Analysis

- 80
- 20
- Document search had high satisfaction
- Schedule queries had low satisfaction

## Time-Based Analysis

- New users started with schedule queries
- Poor results led to behavior change
- Users learned to use document search as workaround
- Masked the actual problem

**Solution:** Built specialized data extraction for dates and schedules

Highlighted new capability → user behavior changed back

# Case Study: Scheduling & Learned User Behavior

## The Challenge

- Construction management platform
- New users struggled with scheduling queries
- Veteran users learned workarounds
- Only 20% success rate for new users on scheduling queries

## The Solution

- Split "document search" vs. "scheduling" segments
- Created specialized scheduling index
- Extracted due dates and milestones
- Announced new feature to users

**Key Takeaway:** Segmentation + dedicated metadata indices dramatically improve user satisfaction

# Hybrid Search: Combining Approaches

### Lexical Search

- Exact keyword matching
- Based on BM25, TF-IDF
- Great for precise terms
- Example: Elasticsearch

### Semantic Search

- Meaning-based matching
- Uses embeddings
- Great for concepts
- Handles synonyms

**Hybrid Search:** Combines strengths of both approaches

# Hybrid Search: Implementation

- Weight results based on query characteristics
  - Adjust weights dynamically based on query type
  - Use query classifier to determine optimal weights

**When to favor each approach:**

- **Lexical**: Specific terms, codes, IDs, exact phrases
- **Semantic**: Concepts, themes, topics, intentions

**Example:** "Q2 2023 revenue" →
70"Growth strategy reasons" → 20

# Metadata Enhancement: Extraction

- Extract structured data during ingestion

**Temporal**

- Dates
- Time periods
- Fiscal quarters
- Years

**Entities**

- Companies
- People
- Products
- Categories

**Document**

- Doc type
- Sections
- Importance
- Source

**Key insight:** Rich metadata enables powerful filtering

# Metadata Enhancement: Filtering

- Create filters based on document attributes
  - Filter by date range, document type, entity
  - Combine filters with semantic search
  - Improve precision without hurting recall
- Enable faceted search capabilities
  - Allow users to narrow results by metadata
  - Provide context-aware filtering options

**Example:** "Revenue for Q2 2023 in North America region"

```
embedding_search(query) AND
date="Q2 2023" AND region="NA"
```

# Specialized Indices: Content Types

- Create separate indices for different content types

**Text Documents**

- Reports
- Articles
- Narratives
- Analysis

**Tabular Data**

- Financial statements
- Metrics tables
- KPI dashboards

**Visual Content**

- Charts
- Graphs
- Diagrams

**Structured Data**

- SQL
- JSON
- XML
- Code

# Specialized Indices: Chunking Strategies

- Optimize chunking strategy per content type

**Text**

- Semantic paragraphs
- Fixed-size chunks
- Sliding window

**Tables**

- Preserve headers
- Keep context
- Include table titles

**Code**

- Function-level chunks
- Class-level chunks
- Keep imports

**Specialized**

- Entity index
- Time-series index
- KPI index

# Tool-Based Approach: Interfaces

**Key concept:** Define specialized search tools with clear interfaces

## Tool Design Principles

- Define clear interfaces for each specialized retrieval method
  - Consistent input/output formats
  - Well-defined parameters and options
  - Clear documentation and examples

**Example Interface:**

```
table_search(query, date_range,
metrics=["revenue", "margin"])
```

# Week 3: Key Insights

## Specialized Retrieval: Beyond Basic Search

- **Implement hybrid search** combining lexical and semantic approaches
  - Lexical: Exact matches, codes, IDs, specific terms
  - Semantic: Concepts, themes, intentions, synonyms
- **Extract rich metadata** during document ingestion
- **Create specialized indices** for different content types
  - Text, tables, code, images each need different handling
- **Design tool-based interfaces** for specialized retrieval methods

**Remember:** Different content types
require different retrieval strategies

# Tool-Based Approach: Routing

## Query Router

- Create a router that selects appropriate tool(s) for each query
  - Use LLM to classify and route queries
  - Consider confidence scores for tool selection
  - Fall back to general search when uncertain

## Execution Strategy

- Consider parallel execution for better performance
  - Run multiple tools simultaneously when appropriate
  - Merge results with intelligent ranking
  - Balance latency vs. thoroughness

# Implementing Query Routing

### Start Simple

- Begin with 3-5 core tools
- Focus on common query types
- Add more as patterns emerge

### Consistency Matters

- Standardize tool interfaces
- Common parameter formats
- Predictable outputs

**Testing tip:** Create synthetic queries
for each tool to validate router accuracy

# Measuring Router Performance

- Test routing accuracy with synthetic queries
  - Create test cases for each tool
  - Measure routing precision and recall
  - Identify confusion patterns between tools
- Measure tool recall: Is the right tool being selected?
  - Track correct tool selection rate
  - Monitor unnecessary tool calls
  - Improve router prompts based on errors

# Financial Domain Tools (1/2)

table_search

- Finds financial tables
- Preserves row/column context
- Extracts precise metrics
- Example: "Q2 operating margin"

text_search

- Retrieves narratives
- MD&A, risk factors
- Semantic paragraph search
- Example: "Revenue growth factors"

# Financial Domain Tools (2/2)

`entity_lookup`

- Company profiles
- Key metrics
- Industry data
- Example: ”Company X’s position”

`time_series`

- Historical metrics
- Trend analysis
- Period comparisons
- Example: ”Revenue growth over 8 quarters”

**Start with tools that address your most common queries**

# Case Study: Construction Blueprints & Visual Summaries

## The Challenge

- Construction company needed AI to answer blueprint questions
- Simple image captioning gave generic descriptions
- Only  27% recall on specific blueprint questions
- Multimodal retrieval was ineffective

## The Solution

- Prompted for detailed descriptions
- "Count floors, label mechanical rooms, highlight windows"
- Added specialized bounding-box extraction
- Merged text + blueprint data

**Key Takeaway:** Specific extraction prompts improved recall from 27% to 75-85% in just days

# Week 4: Key Insights

**Query Routing: Directing Queries to the Right Tools**

- **Implement a query router** to direct questions to specialized tools
  - Use chain-of-thought prompting for better classification
  - Consider multi-tool routing for complex queries
- **Design clear tool interfaces** with consistent parameters
- **Balance precision and recall** in routing decisions
- **Implement fallback mechanisms** for uncertain classifications
- **Track routing accuracy** as a key performance metric

**Remember:** The right tool for the right job dramatically improves results

# Week 5: Key Insights

## Embeddings & Reranking: Optimizing Relevance

- **Choose embedding models** based on domain and query types
  - General models for broad topics
  - Domain-specific models for specialized fields
- **Implement reranking** to improve precision
  - Cross-encoders for higher accuracy
  - LLM-based reranking for complex relevance judgments
- **Fine-tune embeddings** with domain-specific feedback
- **Optimize chunk size** for your specific content and queries

**Remember:** Better embeddings and reranking can dramatically improve relevance

# User Feedback: Binary Feedback

## Simple Binary Feedback with Categories

### Feedback UI

- Thumbs up/down buttons
- Simple and prominent
- Quick to complete

### Reason Categories

- Irrelevant results
- Incomplete answer
- Incorrect information
- Outdated content

**Key insight:** Even simple feedback is better than none

# User Feedback: Citation & Implicit Signals

## Citation Validation

- "Were these sources helpful?"
- Mark irrelevant citations
- Track valuable sources
- Identify missing citations

## Implicit Signals

- Time spent reviewing results
- Follow-up questions
- Copied/saved content
- Repeated queries

**Pro tip:** Combine explicit and implicit signals for a complete picture

# Feedback UI Design

## UI Elements

- Large, obvious buttons
- Modal dialogs for higher engagement
- 1-2 clicks for initial feedback
- Optional detailed feedback

## Feedback Categories

- Retrieval vs. generation issues
- Missing vs. incorrect info
- Technical errors vs. content gaps
- UI/UX problems

**Did you know?** Modal dialogs get 4-5x more feedback than subtle buttons

# Feedback Storage & Analysis

## Store Complete Context

- Original query and retrieved documents
- Generated response and citations
- User feedback and follow-up actions
- System metadata (latency, model version, etc.)

**Example schema:**

{`query`, `results`, `response`,
`feedback`, `metadata`}

# Feedback Analysis Cycle

### Weekly Review

- Analyze feedback patterns
- Track trends over time
- Identify common failures
- Share insights with team

### Prioritization

- Focus on high-volume issues
- Balance quick wins vs. structural fixes
- Create targeted test cases
- Track impact of fixes

# Continuous Improvement Process

## Creating Test Cases

- Convert real failures into test examples
- Expand test coverage based on user behavior
- Validate fixes against expanded test set

## Model Refinement

- Use feedback to create training pairs
- Fine-tune embeddings or rerankers
- Improve router accuracy with real examples

**Key milestone:** When feedback drives automatic system improvements

# Case Study: Rejected URLs in Sales Follow-Up

## The Challenge

- System wrote post-call follow-up emails with links
- AI kept hallucinating or mistyping certain URLs
- 4% error rate on links in emails
- Damaged credibility with customers

## The Solution

- Introduced URL validator
- Rejected links to non-existent pages or unknown domains
- Asked model to remove/fix invalid links
- Later fine-tuned model to avoid invalid links

**Key Takeaway:** Simple post-processing + feedback loop reduced errors to nearly 0%

# Case Study: Changing Copy to Collect More Feedback

## The Challenge

- Team wanted more thumbs up/down data
- Tiny button labeled "How did we do?" hidden at top
- Very few users provided ratings
- Insufficient data for measuring correctness

## The Solution

- Changed copy to "Did we answer your question?"
- Made feedback UI larger and more central
- Requested brief reason for thumbs down
- Categorized feedback for targeted improvements

**Key Takeaway:** UI changes increased feedback volume 4-5x, providing crucial data for improvements

# Streaming Responses: Improving Perceived Performance

## Benefits of Streaming

- Reduces perceived latency by 11%
- Users tolerate longer wait times
- Allows immediate reading while generation continues
- Provides visual feedback on progress

## Implementation Approaches

- Stream tokens as they're generated
- Show interstitials during processing steps
- Render skeleton screens during loading
- Display tool execution in real-time

**Did you know?** Users will wait up to 8 seconds longer when given visual feedback

# Streaming Implementation: Interstitials & UI

- **Interstitial messages** during processing steps:
  - "Thinking..." → "Searching documents..." → "Reading results..." → "Generating response..."
- **Structured streaming** for complex responses:
  - Stream partial JSON objects with content, citations, follow-ups
  - Parse and render components as they arrive
- **Tool execution visualization**:
  - Show function calls and parameters in real-time
  - Allow user edits to parameters before execution

**Key insight:** Streaming isn't just about tokens—it's about communicating process

# Rejecting Work: Setting Expectations

## The Problem

- Not all queries can be answered well
- 90% success rate means 10% failures
- Attempting all queries reduces trust
- Users blame system for poor answers

## The Solution

- Identify query types with low success
- Gracefully reject answering these
- Set clear expectations with users
- Collect feedback on rejected queries

**Example:** "I can't confidently answer this question with the information available. Would you like me to try anyway with the understanding that the answer may be incomplete?"

# Showcasing Capabilities

- **Highlight what your system does well**:
  - Suggest example queries users can try
  - Demonstrate different capabilities through UI
  - Show different content types you can handle
- **Implementation approaches**:
  - Categorized example queries on landing page
  - "Focus" buttons to expand capability options
  - Follow-up suggestions after each response
  - Special UI components for different content types

**Key insight:** Users will use the capabilities you highlight and ignore those you don't

# Chain of Thought: Improving Reasoning

## Benefits

- 10% performance improvement
- Makes complex reasoning possible
- Provides transparency into model thinking
- Enables better error analysis

## Implementation

- Structure as XML components
- Stream thinking process separately
- Allow users to expand/collapse
- Collect feedback on reasoning errors

**Key insight:** Chain of thought can be the difference between usable and unusable

# Monologuing: Managing Complex Contexts

- **Reiterate relevant information** before generating responses:
  - Restate key instructions from the prompt
  - Summarize relevant parts of the context
  - Co-locate related information for better attention
- **Multi-stage monologuing** for complex tasks:
  - Identify relevant variables first
  - Extract relevant sections from documents
  - Connect information across sources
  - Reason about options before generating final response

**Case study:** SaaS pricing quotes improved dramatically with 4-stage monologuing

# Week 6: Key Insights

**Product Considerations:**
**UX, Feedback & Prompting**

- **Design prominent feedback mechanisms**
  - Make feedback UI large and obvious
  - Use modal dialogs for higher engagement
- **Implement streaming responses**
  - Reduce perceived latency with visual feedback
  - Use interstitials to communicate processing steps
- **Know when to reject work**
  - Set clear expectations for low-confidence answers
  - Showcase capabilities you excel at
- **Leverage chain of thought & monologuing**
  - Improve reasoning with structured thinking
  - Reiterate key information for better context handling

# Implementation Roadmap: Initial Phase

## Weeks 1-2: Evaluation Foundation

### Data Generation

- Create synthetic test data
- Cover diverse query types
- Include domain-specific examples

### Metrics Setup

- Implement evaluation pipeline
- Establish baseline metrics
- Build performance dashboard

# Implementation Roadmap: Analysis Phase

**Weeks 3-4: Segmentation & Analysis**

### Query Classification

- Define query categories
- Classify existing queries
- Create segment-specific test sets

### Performance Analysis

- Analyze metrics by segment
- Identify performance gaps
- Prioritize improvement areas

# Implementation Roadmap: Development Phase

## Weeks 5-6: Specialized Retrieval

### Index Development

- Build domain-specific indices
- Optimize chunking strategies
- Implement metadata extraction

### Routing System

- Create basic query router
- Design tool interfaces
- Test with synthetic queries

# Implementation Roadmap: Refinement Phase

## Weeks 7-8: Feedback & Refinement

### Feedback Collection

- Add user feedback mechanisms
- Collect structured feedback
- Analyze feedback patterns

### Continuous Improvement

- Implement fine-tuning process
- Create improvement cycle
- Schedule regular reviews

**Key milestone:** Complete end-to-end system with feedback loop

# Implementation Roadmap: Key Insights

## Systematic Implementation: Building the Flywheel

- **Start with evaluation infrastructure** (Weeks 1-2)
  - Create synthetic test data
  - Establish baseline metrics
- **Analyze and segment queries** (Weeks 3-4)
  - Classify query types
  - Identify performance gaps
- **Build specialized retrieval** (Weeks 5-6)
  - Develop domain-specific indices
  - Implement query routing
- **Implement UX improvements & feedback loops** (Weeks 7-8)
  - Design prominent feedback mechanisms
  - Implement streaming responses
  - Add chain of thought reasoning
  - Set clear expectations with users

# RAG Implementation: The Complete Picture

## From Evaluation to Continuous Improvement

### Technical Foundation

- Evaluation metrics & test data
- Query segmentation & analysis
- Specialized retrieval methods
- Hybrid search & metadata

### User Experience

- Feedback collection mechanisms
- Streaming & interstitials
- Chain of thought reasoning
- Setting clear expectations

## Key Success Factors:

- Start with evaluation infrastructure before building features
- Segment queries to identify specific improvement areas
- Build specialized tools for different content types
- Create feedback loops to drive continuous improvement

# Key Resources & References

## Further Learning & Implementation Resources

### Technical Resources

- LangChain - Framework for RAG applications
- LlamaIndex - Data framework for LLM applications
- RAGAS - Evaluation framework for RAG systems
- LangChain Hub - Community prompts and chains

### Research & Best Practices

- RAG Survey Paper - Comprehensive overview
- Self-RAG - Self-reflective retrieval
- Chain-of-Thought - Reasoning techniques
- Instructor - Structured outputs

**Contact:** jxnl.co — @jxnlco — github.com/jxnl

# Conclusion & Next Steps

# Key Takeaways

- **Measurement First, Always**
  - Start with clear metrics before making changes
  - Focus on retrieval recall as a leading indicator
  - Build fast evaluation cycles with synthetic data
- **Systematic Over Ad-hoc**
  - Segment queries to focus improvements
  - Build specialized tools for different content types
  - Create feedback loops for continuous improvement
- **Domain Expertise Matters**
  - Generic embeddings aren't enough for specialized domains
  - Fine-tune with domain-specific feedback
  - Balance technical implementation with domain knowledge

# Common Pitfalls to Avoid

- **Intervention Bias**
  - Adding more prompt engineering without measurement
  - Accumulating technical debt through unproven additions
  - Focusing on generation quality before fixing retrieval
- **Absence Blindness**
  - Missing retrieval problems because they're less visible
  - Not measuring what you can't directly observe
  - Assuming generation issues when retrieval is the problem
- **Premature Optimization**
  - Fine-tuning models before collecting enough data
  - Building complex architectures before proving value
  - Optimizing for edge cases before handling common cases

# Resources & Tools

- **Evaluation Frameworks**
  - RAGAS: Comprehensive RAG evaluation toolkit
  - LangChain Evaluation: Built-in metrics for RAG systems
  - TruLens: Feedback-driven evaluation for LLM applications
- **Retrieval & Embedding Tools**
  - Sentence Transformers: Fine-tunable embedding models
  - Cohere Rerank: Powerful reranking capabilities
  - Weaviate/Pinecone/Qdrant: Vector databases with hybrid search
- **Feedback Collection**
  - Promptfoo: Prompt testing and evaluation
  - Argilla: Data collection and annotation platform
  - Weights & Biases: Experiment tracking and visualization

# Questions to Guide Your Implementation

1. What are your most critical query types and how will you measure success for each?
2. How will you generate synthetic test data that matches your domain?
3. What specialized retrieval methods do you need for your content types?
4. How will you collect and incorporate user feedback?
5. What's your plan for continuous improvement over time?

- Remember: The goal is to build a system that gets better with every interaction
- Start simple, measure thoroughly, and improve systematically
- Focus on the highest impact areas first based on data, not intuition

# Contact & Follow-up

- **Office Hours**
  - Available for project-specific questions
  - Schedule via email: jason@jxnl.co
- **Additional Resources**
  - Course materials and code examples: github.com/jxnl/mit-rag
  - Reference implementations for different domains
  - Recommended reading and tutorials: jxnl.co/writing
- **Community**
  - Join the discussion group for ongoing support
  - Share your progress and learnings
  - Connect with others working on similar challenges

Thank you! Questions?