AGENTIC AI RAG MODELS IN STRAIGHT-THROUGH UNDERWRITING

ROBERT RICHARDSON - BRIGHAM YOUNG UNIVERSITY

JOSH MEYERS, FCAS – AKUR8

THANKS TO CAS FOR RESEARCH FUNDING!


FEAR NOT! D/C 6:36


FEAR NOT! D/C 6:36

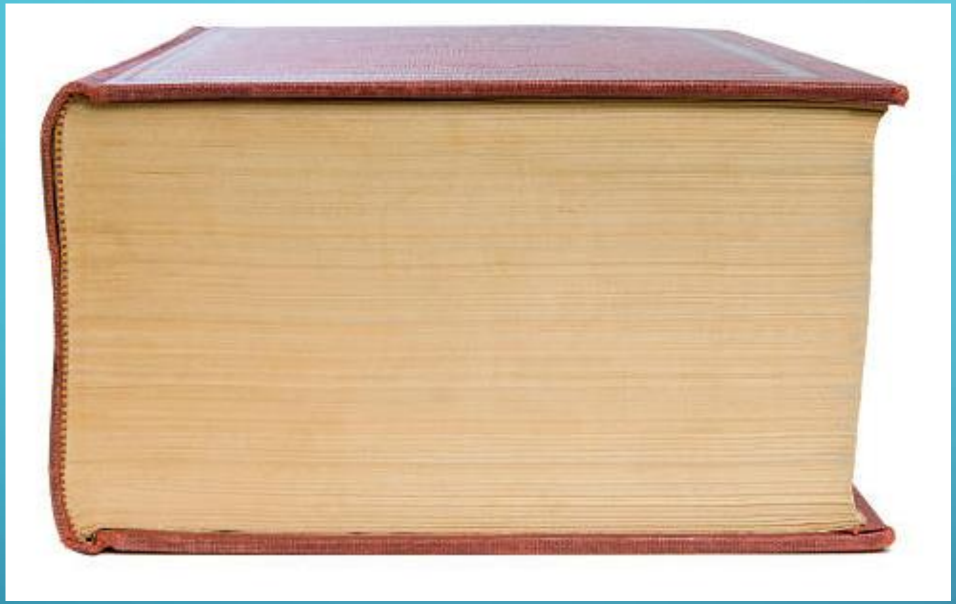

home
sweet
home
DC 6 36

Expectation: AI will increase operational efficiency

Reality:

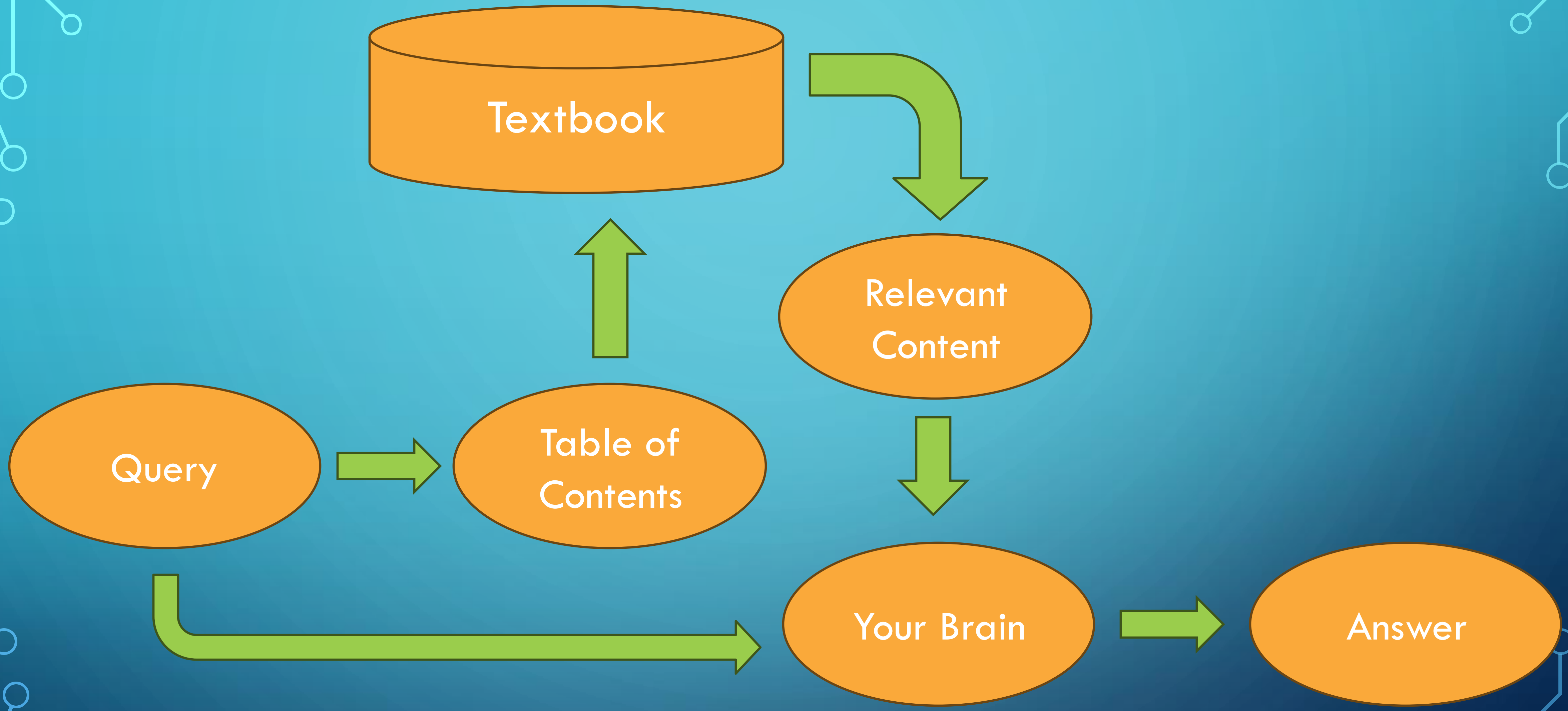
- **NBER (Rapid Adoption of Generative AI, Working Paper 32966)**
  "Users report significant time savings. Among those using it for work, the average time saving is 5.4% of their work hours. Including non-users, this translates to an average of 1.4% across all workers."
- **METR (Randomized Controlled Trial of AI Coding Tools)**
  "Surprisingly, we find that when developers use AI tools, they take 19% longer than without — AI makes them slower."
- **METR (reported in ITPro/Reuters/etc.)**
  "Developers expected AI to speed them up by 24%, and even after experiencing the slowdown, they still believed AI had sped them up by 20%."
- **Accenture (Business Insider reporting CEO remarks)**
  "Simply enabling employees to complete tasks faster might not lead to increased output; instead, it could result in more downtime… for AI to truly boost productivity, companies need to rethink workflows and job roles."
- **Ars Technica (reporting on broader productivity paradox)**
  "Time saved by AI offset by new work created, study suggests" — survey data indicates AI created more tasks for 8.4% of workers

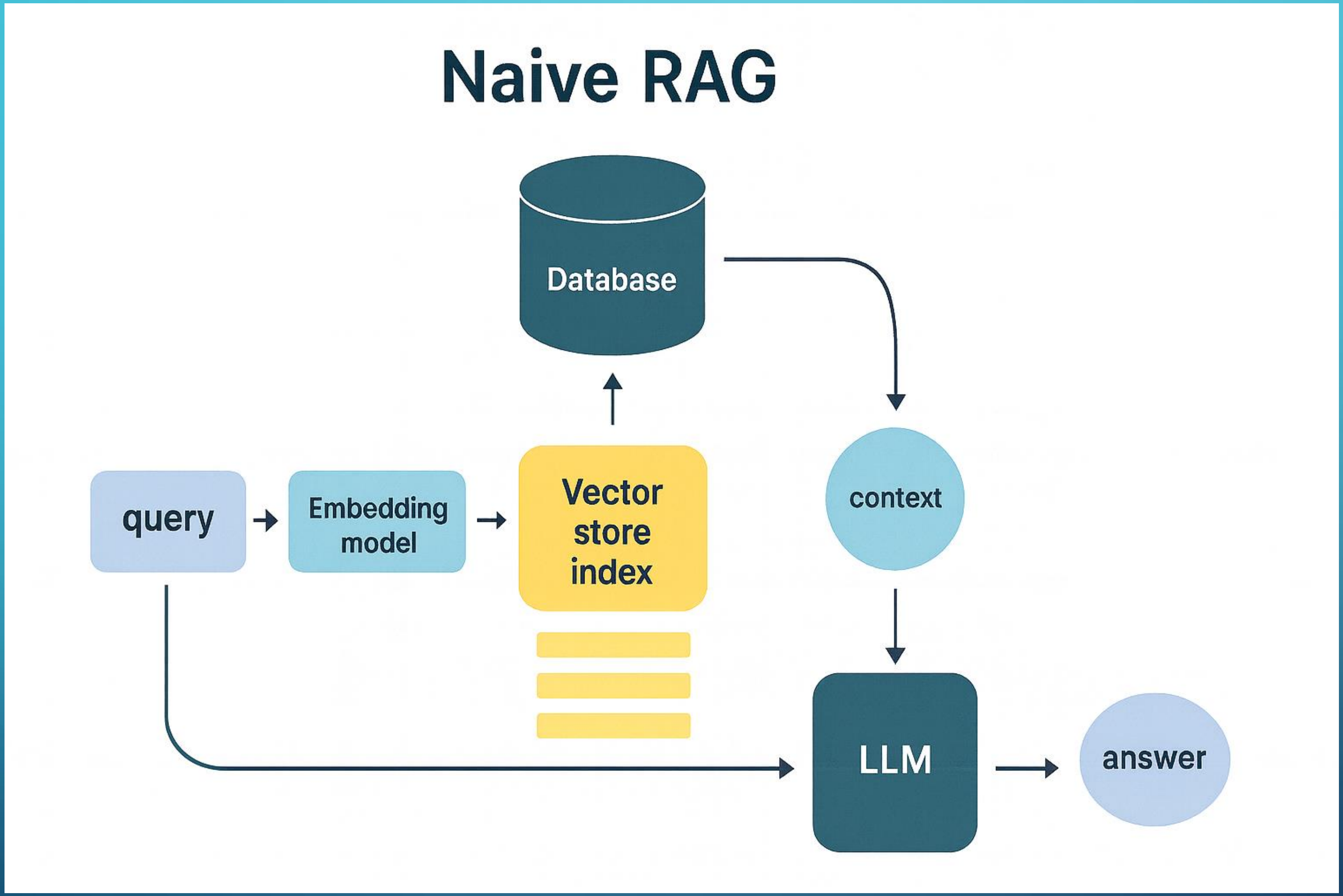
Imagine you need to answer a domain specific question. Here's a textbook.



To make sure you didn't miss anything you need to read the whole book. And there's no table of contents.

Retrieval Automated Generation (RAG) models takes this giant textbook and turns it into something much more manageable.


Textbook
Relevant Content
Query
Table of Contents
Your Brain
Answer


Naive RAG
Database
query
Embedding model
Vector store index
context
LLM
answer

Retrieval-Augmented Generation (RAG) enhances language models by grounding outputs in retrieved documents relevant to a given input query. The core steps include:

- **Vectorization**: The input query is encoded into a dense vector representation using an embedding model such as BERT, OpenAI, or Cohere.
- **Similarity Search**: The query vector is compared (typically using **cosine similarity**) against a **vector store index** built from domain-specific documents (e.g., underwriting manuals, actuarial reports).
- **Context Retrieval**: Top-K documents with the highest similarity scores are selected and passed to the language model along with the original prompt.
- **Augmented Generation**: The LLM (e.g., GPT, Claude, LLaMA) uses both the original input and the retrieved documents (i.e., *context*) to produce a response that is grounded, specific, and less prone to hallucination.

This architecture enables dynamic grounding of generative output in ever-changing, organization-specific content without requiring retraining or static fine-tuning.

Multiple peer-reviewed studies and benchmark analyses support the performance and cost-efficiency of RAG models:

- **Lakatos et al. (2025, MDPI)** found RAG models outperformed fine-tuned LLMs: “RAG constructions are more efficient… outperforming fine-tuned models by 16% ROUGE, 15% BLEU, and 53% cosine similarity.”
- **Wang et al. (2024, Tencent R&D)** in a code-completion benchmark: “RAG achieved higher accuracy while consuming significantly fewer compute resources than domain-specific fine-tuning.”

These studies affirm that RAG models are not only **high-performing**, but also **scalable and resource-efficient**, particularly in domain-specific applications.

Businesses across sectors are adopting RAG architectures to unlock productivity gains in content-heavy workflows:

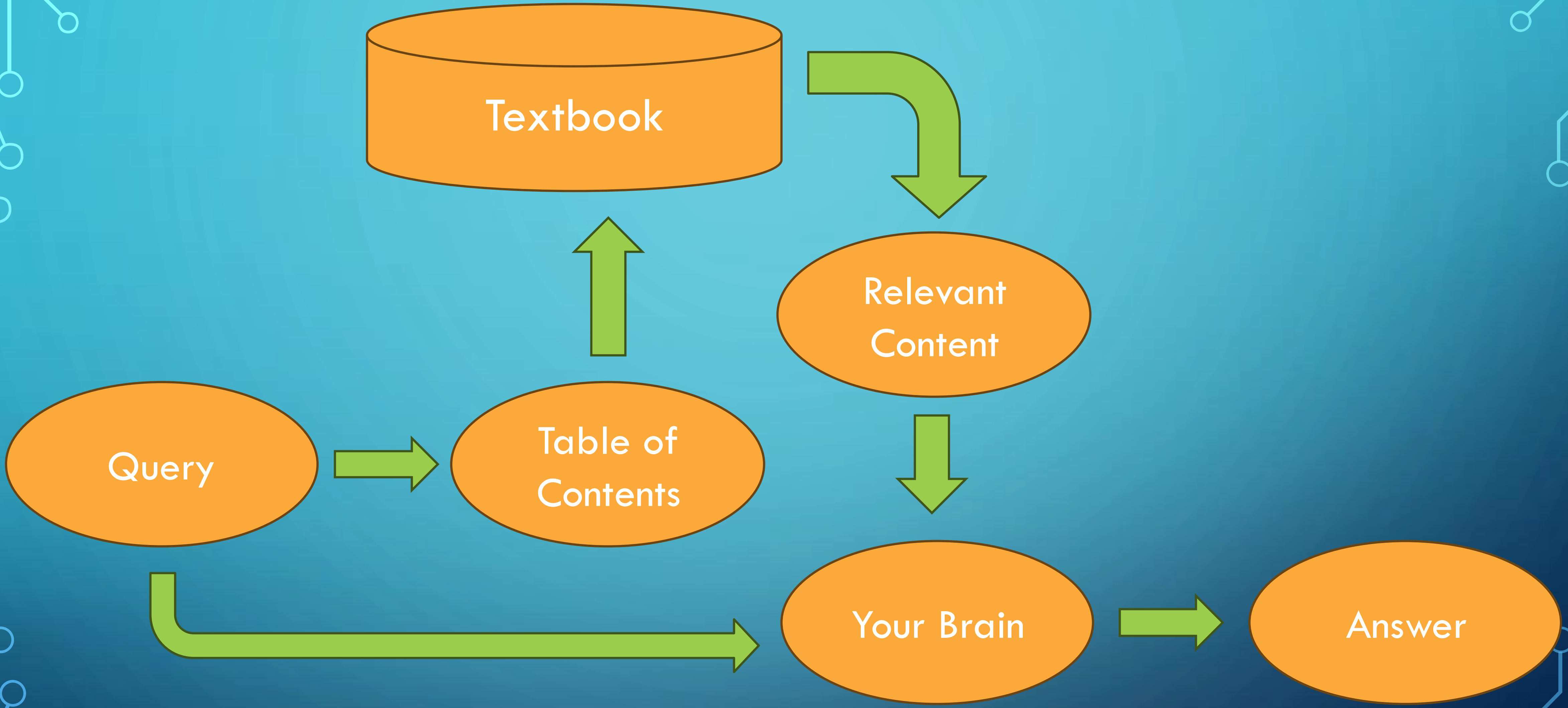
Wall Street Journal (Oct 2024) reports that companies use RAG to automate file tagging and access internal documents faster:
“Firms… are connecting large language models to their data with a technique known as retrieval-augmented generation (RAG)… to organize data more efficiently.”

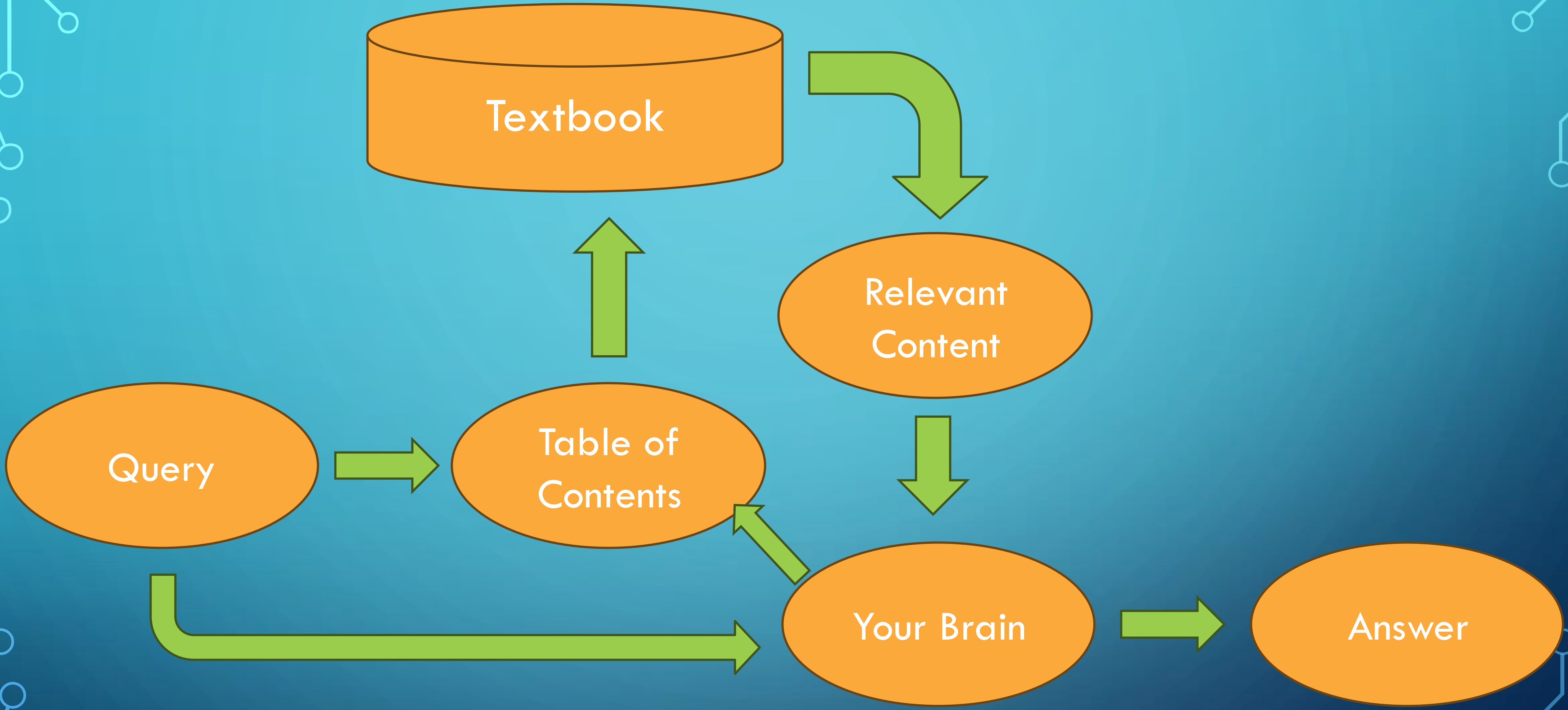
Society of Actuaries – Generative AI Roundtable (June 2025)
“Several panelists reported using RAG to enhance the capabilities of AI chat interfaces by integrating access to large repositories of internal documents. This improves accuracy and contextual relevance in responses.”

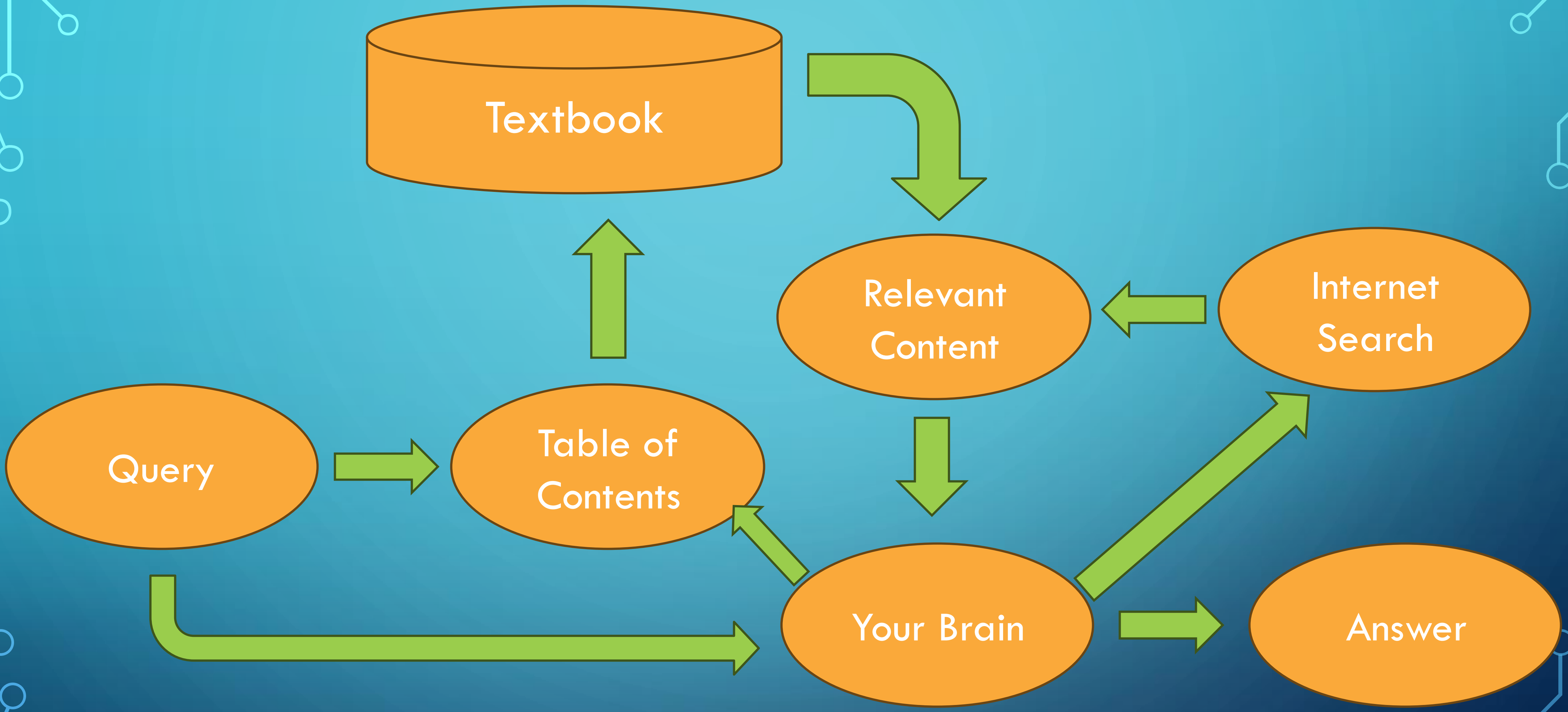
Possible Issues:

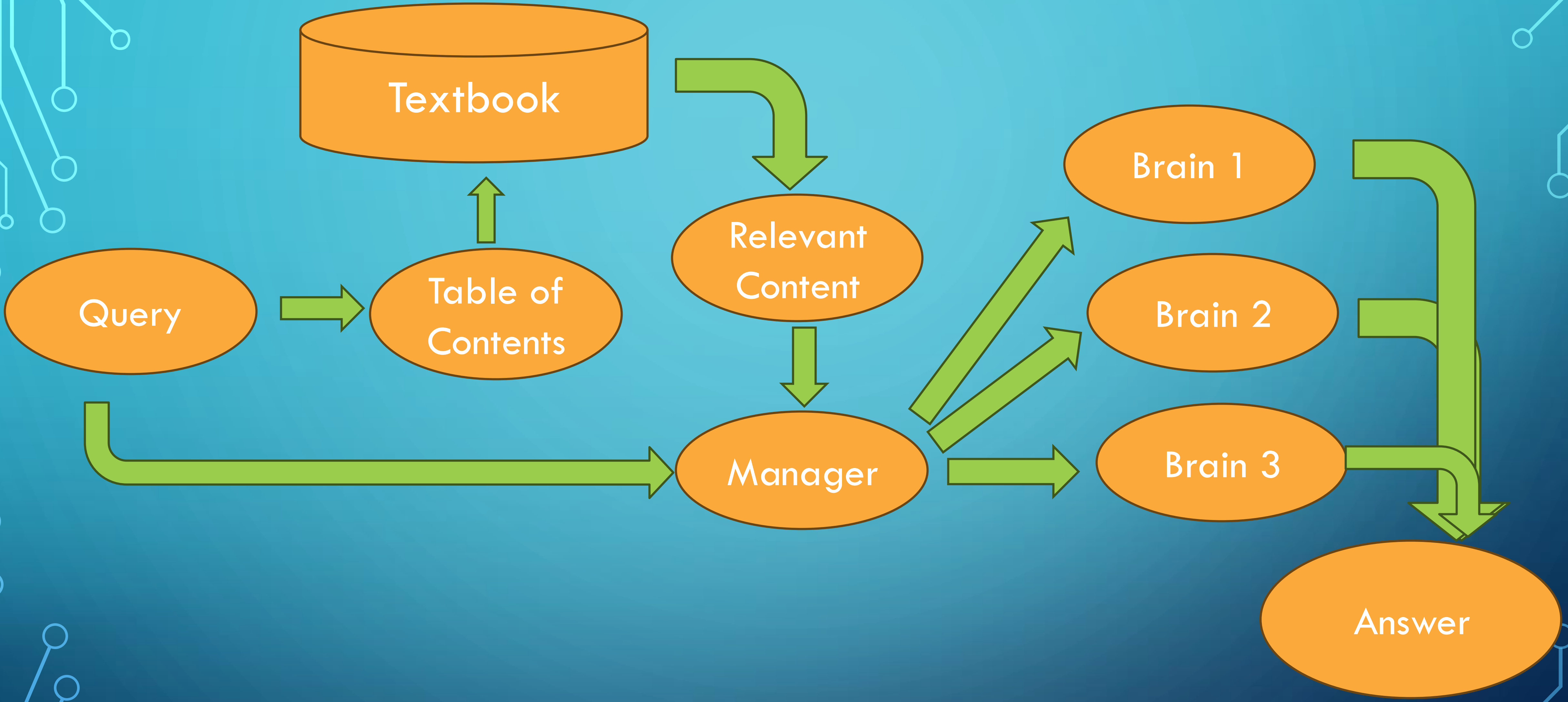
- What if it misses the right context needed to answer the question
- What if the context needed exists somewhere else
- What if there are multiple ways to address the question

RAG models reduce error and increase accuracy, but still also require simple straightforward tasks.


Textbook
Relevant Content
Query
Table of Contents
Your Brain
Answer


Textbook
Relevant Content
Query
Table of Contents
Your Brain
Answer


Textbook
Relevant Content
Internet Search
Query
Table of Contents
Your Brain
Answer


Textbook
Query
Table of Contents
Relevant Content
Manager
Brain 1
Brain 2
Brain 3
Answer

Agentic AI

Agentic AI systems go beyond single-turn prompting by exhibiting autonomous, goal-directed behavior. These models:

- **Reflect** on intermediate outputs and adjust their reasoning
- **Use tools** like retrievers, calculators, or APIs when needed
- **Route tasks** dynamically based on uncertainty or missing data
- **Maintain memory/state** across interactions to track progress

Instead of answering a prompt once, **agentic systems orchestrate multiple steps**, incorporating feedback, iteration, and tool invocation to pursue objectives — much like human agents do.

Agents are the current BIG DEAL in AI

- **Prashant D. Sawant et al. (2025) — *Agentic AI: A Quantitative Analysis of Performance***

"Agentic AI systems completed tasks 34.2 % faster than traditional AI, with an accuracy rate of 94.2 % compared to 86.5 %, and improved resource utilization by 13.6 %."

- **Maxime Robeyns et al. (April 2025) — "A Self-Improving Coding Agent"**

"We find performance gains from 17 % to 53 % on SWE-Bench Verified benchmarks when the agent autonomously edits its own code."

- **Microsoft Research — AutoGen v0.4 (2025)**

"Scalable, modular, multi-agent orchestration… enabling robust coordination for complex workflows," is reported to **outperform single-agent designs on tasks requiring planning and execution.**

Research Goal: Examine Agentic AI in Actuarial Applications

AGENTIC AI IN STRAIGHT-THROUGH UNDERWRITING

- Using Business Owner Policy Insurance
- Provide a dense policy guide book with global requirements and business specific requirements
- Have an option to retrieve third party data
- Use a logistic regression model for final decision

THE DATA

In order to test this framework, we need applications and a policy guide

Generated a 256-page policy guide book using a chain of prompts.

- Use a tool called LangChain in Python that helps building pipelines and workflows where LLMs are used.
- Found public online underwriting guidelines and developed general and business specific guidelines for 127 businesses

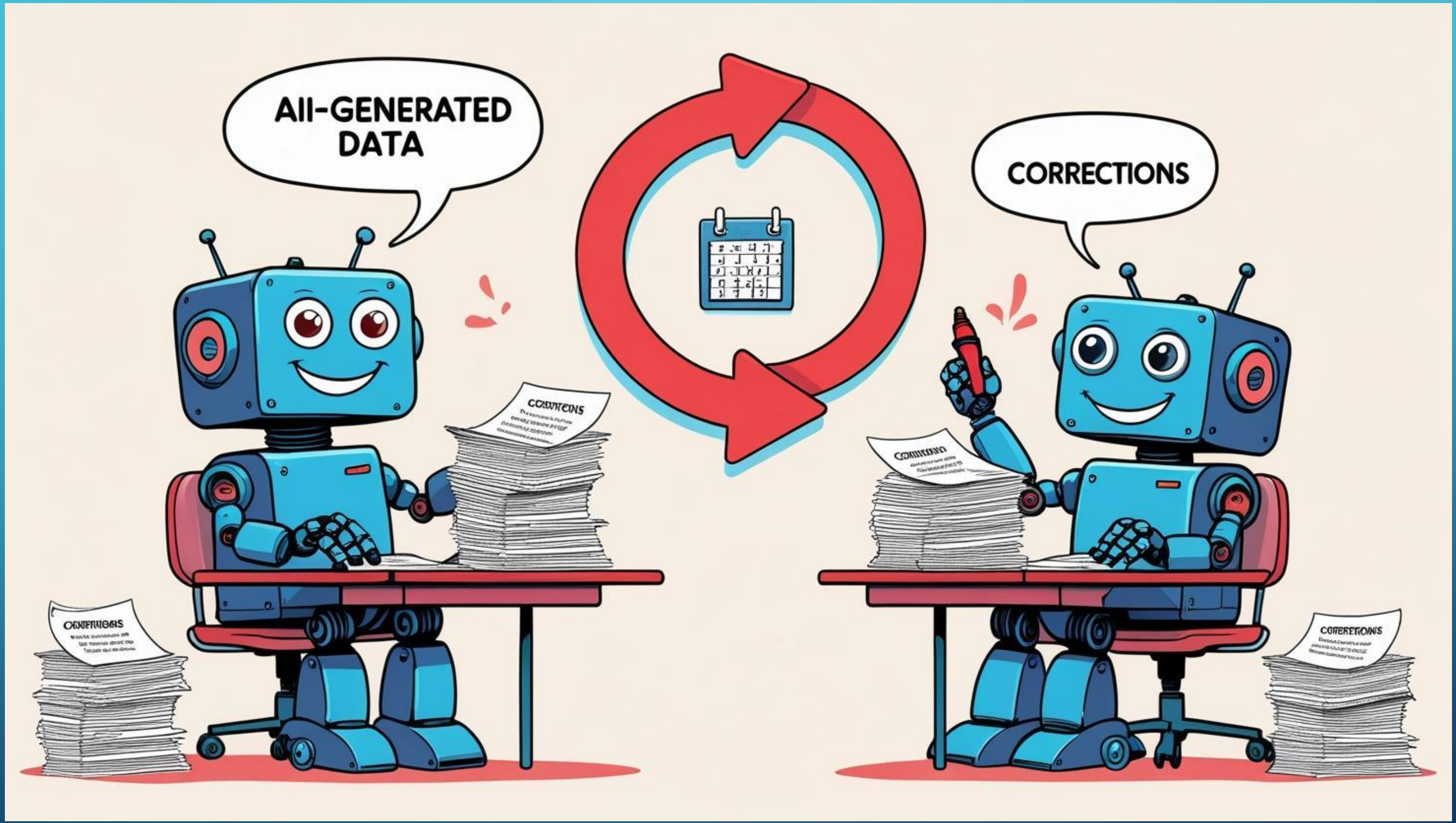
Example LangChain code

```
    # Step 1: Thoroughly describe business
    prompt_step1 = PromptTemplate(
        input_variables=["business_type"],
        template="""
        Thoroughly describe the type of work done by small businesses
classified as {business_type}. Include where the work is performed,
typical customers, and a typical workday.
        """
    )
    steps["step1"] = (prompt_step1 | llm).invoke({"business_type":
business_type}).content

    # Step 2: Property risks
    prompt_step2 = PromptTemplate(
        input_variables=["business_description", "coverage_text"],
        template="""
        Using the description of the business:
        {business_description}

        Identify property insurance risks for this class, referencing this
coverage guide:
        {coverage_text}
        """
    )
    steps["step2"] = (prompt_step2 | llm).invoke({"business_description":
steps["step1"], "coverage_text": coverage_text}).content
```

For each business type we then generate applications under 5 different scenarios

1. Include nothing that would make it out of appetite and include values that would make it pass the logistic regression criteria. SHOULD ACCEPT.
2. Include one and only one detail in the application that would make it out of appetite. SHOULD REJECT.
3. Include nothing that would make it out of appetite and include values that would make it fail the logistic regression criteria. SHOULD REJECT.
4. Leave a crucial detail out for making it in or out of appetite, add that detail it to the third party data that would make it out of appetite. SHOULD REJECT.
5. Leave a crucial detail out for making it in or out of appetite, do not add it to the third party data. SHOULD REFER TO HUMAN REVIEW.

Currently we have a team of BYU students working to validate these applications, otherwise we’d have AI creating data that AI would be evaluating.



Application data example


```
{
  "Application Data": {
    "NAME": "WealthGuard Financial Advisors",
    "FEIN OR SOC SEC #": "12-3456789",
    "BUSINESS TYPE": "Corporation",
    "MAILING ADDRESS": "1234 Wealth Ave Suite 200, Finance City, CA 90210-1234",
    "CONTACT FOR INSPECTION": "John Smith",
    "GL CODE": "8742",
    "SIC": "523930",
    "NATURE OF BUSINESS": "Financial Services",
    "DESCRIPTION OF OPERATIONS": "WealthGuard Financial Advisors specializes in
providing comprehensive financial planning, investment management, and advisory
services to individuals and businesses. Our team of certified financial planners and
investment advisors focuses on asset allocation, retirement planning, tax
optimization, and wealth management strategies tailored to meet the unique goals of
each client. We are committed to upholding the highest standards of fiduciary
responsibility and maintaining robust data security practices to protect client
information. Our services also include ongoing portfolio monitoring and regular
performance reviews to ensure alignment with clients' financial objectives.",
    "DATE BUSINESS STARTED": "01/15/2015",
    "EFFECTIVE DATE": "10/01/2023",
    "EXPIRATION DATE": "09/30/2023",
    "NEW/RENEWAL": "NEW",
    "PAYMENT PLAN": "Annual",
    "TOTAL PREMIUM": "$\"12500.00",
```

Third party data example


```
 "Third-Party Data": {
   "Credit Score": 718,
   "Credit Rating": "Good",
   "Google Review Count": 79,
   "Average Review Rating": 4.9,
   "Sample Reviews": [
     "I recently had the pleasure of working with [Business Name], and I couldn't be happier with the experience. Their team of financial planners provided personalized advice that truly aligned with my goals, and their expertise in investment strategies gave me confidence in my financial future. I highly recommend them for anyone looking to take control of their financial planning!",
     "The team at [Business Name] provided exceptional guidance and personalized investment strategies that truly helped me feel confident about my financial future.",
```

Decision example: Because of how we generated the data, this was generated first

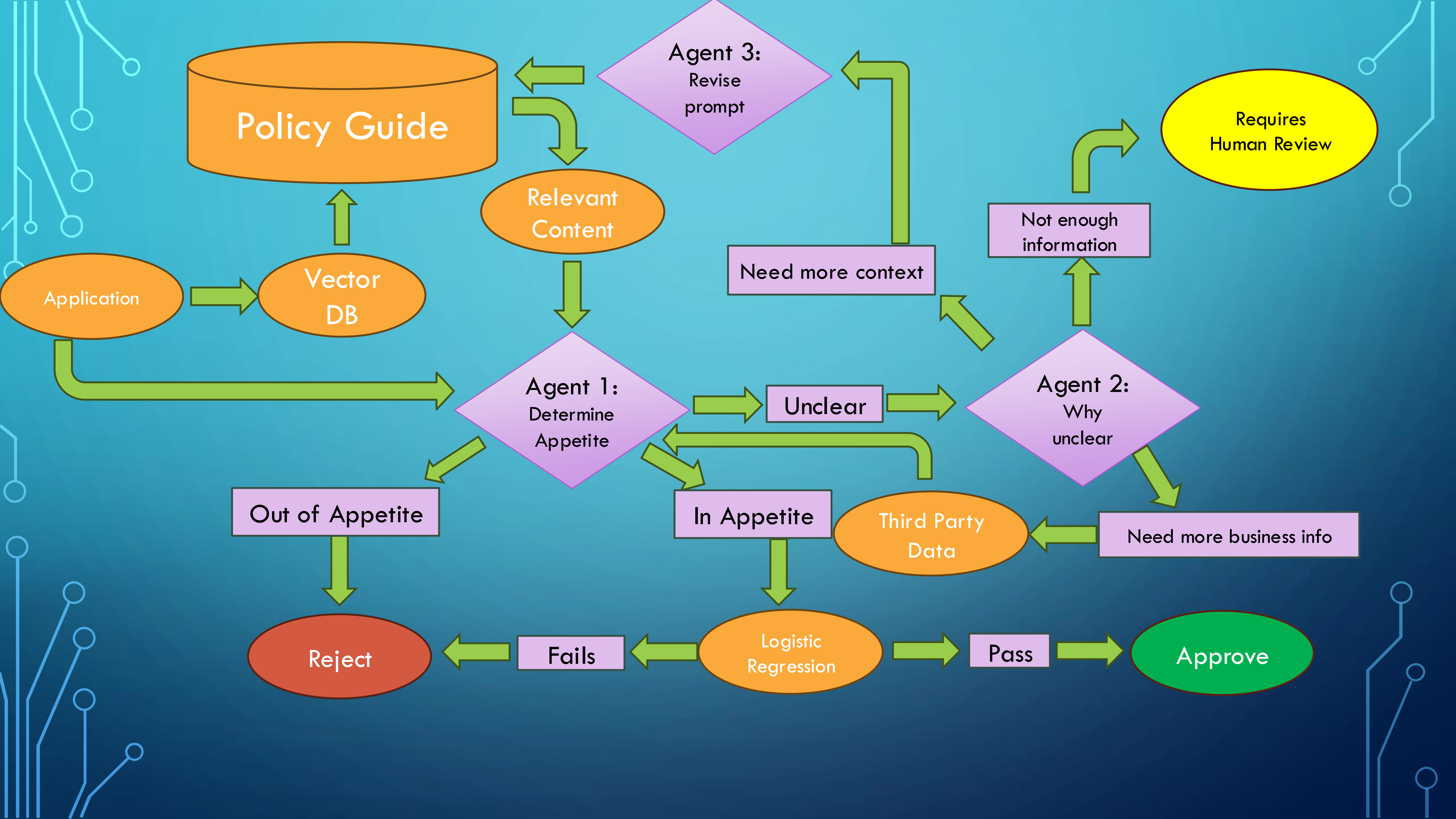

```
  "Final Decision": "REJECT",
  "Final Reason": "This business is rejected due to a previous loss related to a data breach incident resulting in a claim of $10,000, which violates the underwriting guidelines for acceptable loss history.",
  "Underwriting Guidelines": "3\nIndustry-Specific Guidelines\n3.1\nAccounting and Financial Services\nOverview\nThis subsection addresses underwriting considerations for businesses operating within\nthe accounting and financial services sectors. These businesses typically include certified public\naccountants (CPAs), bookkeeping services, tax preparers, financial planners, investment advisors,\nand other related professional service providers. Given the nature of their work\u2014primarily involving\nadvisory, fiduciary, and record-keeping functions\u2014these businesses face unique risks that differ from\ntraditional retail or manufacturing operations. The Business Owners Policy (BOP) for these entities\nmust be carefully tailored to address exposures related to professional liability, data security, and\nregulatory compliance, while also covering standard property and general liability exposures.\nTypical Risks and Exposures\n\u2022 Professional Errors and Omissions: Although the BOP does not typically include pro-"
}
```

Our straight through underwriting pipeline:

1. Agent 1 is a routing agent. Checks the application against the underwriting guidelines
   a. If it is fully in appetite, it goes to logistic regression model
   b. If it is out of appetite it ends in a decision of "Reject"
   c. If it is unclear it goes to Agent 2
2. Agent 2 is another routing agent.
   a. If it is unclear because it needs more context from the guide book it goes to Agent 3.
   b. If it is unclear because it needs more business information it collects third party data
   c. If it has collected all available data and decides more would not change anything, it ends in a decision of "Requires Human Review"
3. Agent 3 is a reflection agent. It creates a prompt based on material in the application it needs to find clarification on from the guidebook and returns to step 1.

4. If it goes to third party data, the third party data is included along with the application data and context and it returns to Agent 1

5. If it makes it to the logistic regression function, it tests certain variables. If it exceeds a certain threshold it ends with a decision of “Accept”. Otherwise it ends in a decision of “Reject”.


Policy Guide
Agent 3:
Revise
prompt
Requires
Human Review
Relevant
Content
Not enough
information
Need more context
Application
Vector
DB
Agent 1:
Determine
Appetite
Unclear
Agent 2:
Why
unclear
Out of Appetite
In Appetite
Third Party
Data
Need more business info
Reject
Fails
Logistic
Regression
Pass
Approve

LangGraph is like LangChain but is particularly useful for building agents with LLMs as core tools.

```
flow = StateGraph(UnderwritingState)

# Nodes (already defined by you)
flow.add_node('SIC_CHECK', check_sic_node)
flow.add_node('GUIDELINES_EVAL', guidelines_eval_node)
flow.add_node('THIRD_PARTY_EVAL', third_party_eval_node)
flow.add_node('REFLECTION_NODE', reflection_node)
flow.add_node('LOGISTIC_EVAL', logistic_eval_node)
flow.add_node('CONCERNING_DETAILS_CHECK', concerning_details_check_node)
flow.add_node('REFLECT_CONCERNS', reflect_concerns_node)
flow.add_node('HUMAN_REVIEW', human_review_node)
flow.add_node('FINAL_REJECT', final_reject_node)

# Entry Point
flow.set_entry_point('SIC_CHECK')

# SIC_CHECK always goes to GUIDELINES_EVAL unless immediately rejected
flow.add_edge('SIC_CHECK', 'GUIDELINES_EVAL')

# GUIDELINES_EVAL will go to different nodes depending on the router
flow.add_conditional_edges('GUIDELINES_EVAL', guidelines_router)
```

All the code including the application data and policy guidebook in its current form is available on GitHub:

https://github.com/drbob-richardson/Actuarial_Agentic_AI/tree/main/bop_agentic_rag

However, this is an active project, we'll be updating this periodically, so probably not quite worth going to it yet.

SCENARIO	MODEL	ACCEPT	REVIEW (RHR)	REJECT
1. Guidebook Compliance	Agentic RAG	109	16	0
	RAG	105	10	10
	LLM-only	79	30	16
2. Single-Issue Violation	Agentic RAG	0	32	93
	RAG	4	60	61
	LLM-only	12	88	25
3. Logistic Failure	Agentic RAG	0	0	125
	RAG	0	17	108
	LLM-only	0	30	95
4. Missing Info (Recoverable)	Agentic RAG	5	42	88
	RAG	40	59	26
	LLM-only	45	41	39
5. Missing Info (Unrecoverable)	Agentic RAG	9	101	15
	RAG	29	66	30
	LLM-only	29	80	16

I also have each model provide a reason why they are rejecting and can evaluate accuracy based on cosine similarity. This will measure how often they are rejecting for the exact reason the data is generated to be rejected.

Example:

Actual reason: Reject due to significant losses due to a security breach

Answer 1: Reject due to losses that could have been avoided due to weak security protocols exceeded a given threshold

Would score better than

Answer 2: Reject due to distance from the fire hydrant being too far.

Based on Cosine similarity, the results were

- Agentic RAG – 21%
- Naïve RAG – 15%
- Standard LLM – 11%

The Agentic RAG outperforms Naïve RAG which outperforms non-RAG models

Things to do still

- Finish human editing
- Test different design patterns, other modeling choices
  - Retrieval strategies
  - Foundation model choices
- Write the paper – including looking carefully at ethics
- Tighten up the GitHub, make data available – Allow other people to test different AI structures on my evaluation data

Extensions

- Different actuarial scenarios
  - Detecting changes over time in the sic codes
  - Claims processing
  - Company procedure and government compliance in pricing / reserving models
- New things are always coming out
  - Cache Automated Generation (CAG) is supposedly faster and more accurate than RAG
  - Model Context Protocol (MCP) is the biggest new thing since Agentic AI
- Economic Adaptive Agents
  - The agent decision making has a utility function, balances costs of getting third party data against potential gains for streamlined decision making.
  - Agent includes a learned model: how often did it make a routing decision that changed the outcome

THANK YOU!!