jxnl.co

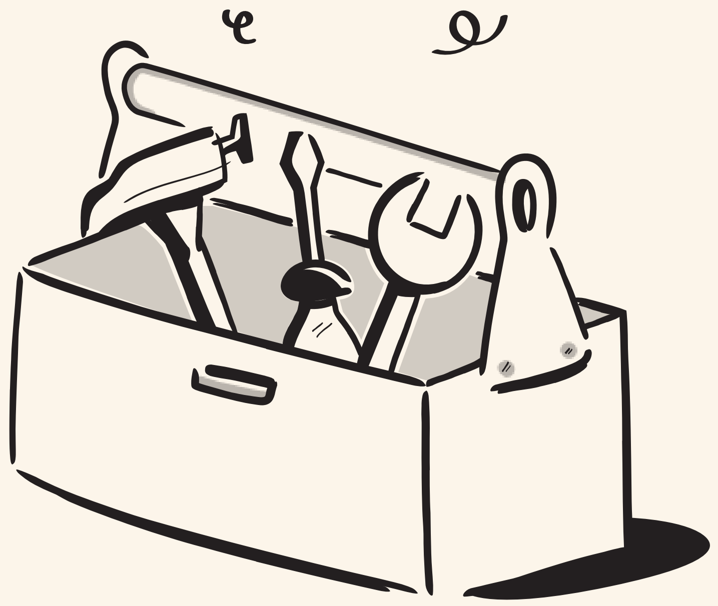
@jxnlco

Systematically Improving RAG Applications

Session 1

Kickstart the Data Flywheel: Fake it till you make it

Jason Liu



Agenda

We need to build high-specificity tooling that users care about

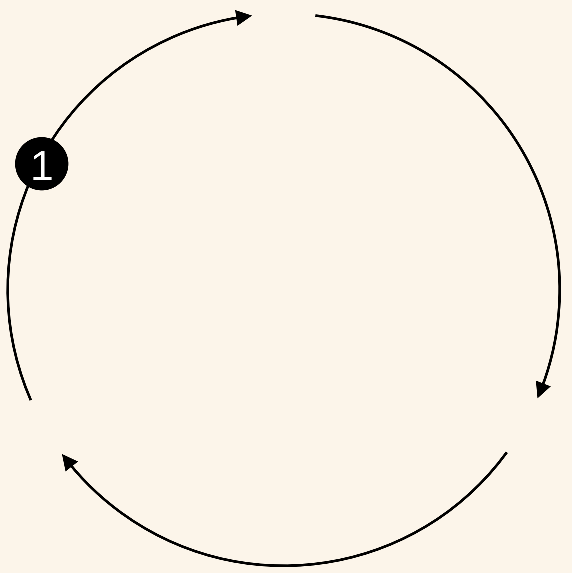
Often when I hear "we need complex reasoning" it often comes from lack of user empathy and not knowing what the user wants.

Agenda

RAG developers unknowingly sabotage their own applications

Vague Metrics

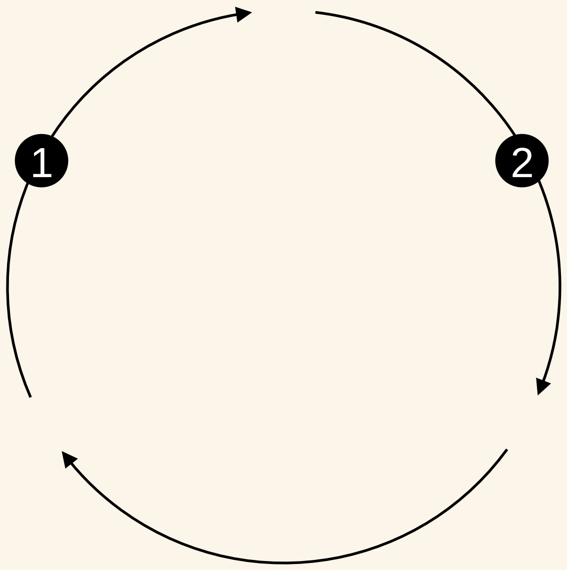
- Thinks "Look better?"
- Thinks "Don't feel right"



RAG developers unknowingly sabotage their own applications

Vague Metrics

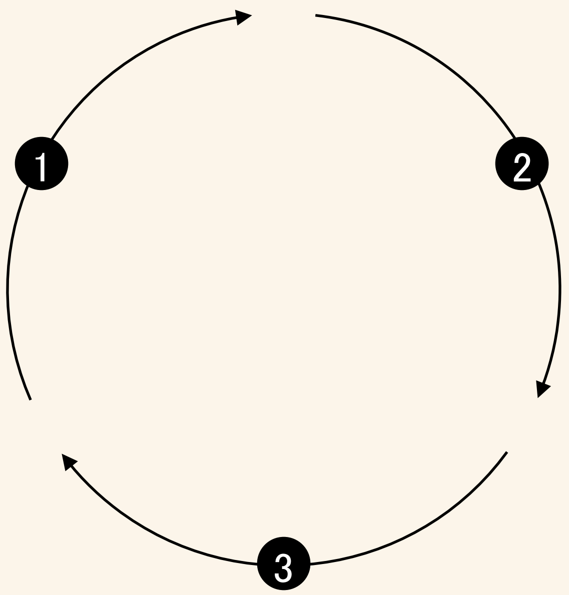
- Thinks "Look better?"
- Thinks "Don't feel right"



No superpowers

- **Build generic solutions for broad problems** (e.g., search all personal data) instead of on specific tasks (e.g., meeting preparation tool)
- Outcomes not features

RAG developers unknowingly sabotage their own applications



Vague Metrics

- Thinks "Look better?"
- Thinks "Don't feel right"

No superpowers

- **Build generic solutions for broad problems** (e.g., search all personal data) instead of on specific tasks (e.g., meeting preparation tool)
- Outcomes not features

Unactionable Feedback

- Reviews focused on generation before establishing search evals
- Looking at data does not result in recommendations for next time
- Vibes translation is generally unsolved

This leads to a disappointed leadership and sad developers

After looking at the data, you're supposed to then take action

Agenda

Common pitfalls made by RAG developers

- Enter into a vicious cycle
- **Overemphasize lagging metrics**
- Fall victim to absence blindness and intervention bias

The RAG playbook

Notebook exercises

Homework for this week

Sneak peak for next week

Difference between leading and lagging metrics

Leading

Easy to improve

- Track and predict future performance
- Provide feedback on when and where to intervene
- Often Inputs to a system

Example

Number of experiments run

Easy Analogy:

Counting Calories, Burning Calories, Working out

Lagging

Difficult to improve

- Measure past outcomes
- Often unresponsive and hard to change
- Often Outputs of a system

Example

Application quality, Churn, Satisfaction

Easy Analogy:

"Be strong, lose weight"

Difference between leading and lagging metrics

Leading

Easy to improve

- Track and predict future performance
- Provide feedback on when and where to intervene
- Often Inputs to a system

Example

Number of experiments run

Set concrete goals for leading metrics

- Focus on increasing the number of hypotheses tested per week
  - Through continual experimentation, build a stronger intuition for better retrieval methods or indices
- It's going to be way more boring, like counting calories

Agenda

Absence blindness:

You don’t fix what you can’t see

Everyone sees:

Generation, Latency

Not everyone sees:

Poor Retrieval

Bad Representations

Bad Chunks

Bad Extraction

Intervention bias:

You try to do things to feel in control

People keep trying to change generation and apply techniques to feel in control

Agenda

Common pitfalls made by RAG developers

The RAG playbook

- **Step 1-4: Initial implementation: focus on retrieval metrics**
- Step 2 Deep dive: Synthetic data generation: fast testing & review processes

Homework for this session

Sneak peak for next session

The RAG Flywheel

Thesis: The principles we’ve applied in search are highly relevant to what we want to do with RAG

🟧 Current steps

Step	Description
1 **Initial implementation**	Start with a basic RAG system setup
2 **Synthetic data generation**	Create synthetic questions to test the system’s retrieval abilities
3 **Build fast evaluations**	Conduct quick, unit test-like evaluations to assess basic retrieval capabilities (e.g., precisio recall, mean reciprocal rank), and explain why each matters
4 **Real-world data collection**	Gather real user queries and interactions. Ensure feedback is aligned with business outcomes or correlated with important qualities that predict customer satisfaction
5 **Classification and analysis**	Categorize and analyze user questions to identify patterns and gaps
6 **System improvements**	Based on analysis, make targeted improvements to the system
7 **Production monitoring**	Implement ongoing monitoring to track system performance
8 **User feedback integration**	Continuously incorporate user feedback into the system

Fundamental mindset shift: Stop using LGTM@K (“Looks good to me”)

In the long term, LLMs are going to improve their ability to synthesize information in context (ICL – in-context learning)   It’s our responsibility to own improving search/retrieval

	Generation evals	Retrieval evals
Testing for…	Factuality **Team lead**: “Looks better now” (based on made up evaluations that don’t capture usage behavior)	Recall, precision@K Lexical search, semantic search, re-rankers
Speed of tests	Slow ~1s to 10s per test	Fast ~10ms to 800ms per test
Cost of tests	$100s per run **Team lead**: “who spent $1000 on OpenAI credits yesterday?”	Negligible per run
Frequency of tests	Infrequent	Frequent
Speed of test iterations	Slow iterations (tests that could take minutes take hours)	Fast iterations
Ease of test scalability	Difficult	Easy

What is the difference between a generation and retrieval eval?

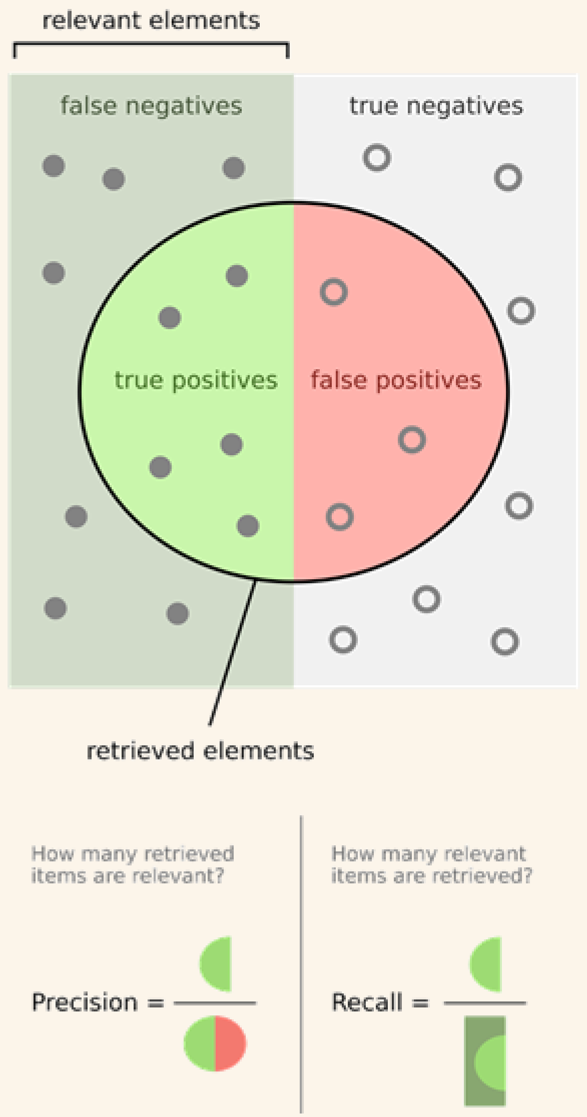
Examples	**Generation eval** (test for generation and factuality)	**Retrieval eval** (test for recall and precision)
Textbook analogy	• **Question:** "What is the powerhouse of the cell?" • **Potential desired answers**: • "Mitochondria[6,9,13]" • "The powerhouse of the cell is mitochondria from pages 6, 9, 13" • Definition of mitochondria + Paragraph	• **Question**: "What page(s) can I find the answer to 'what is the powerhouse of the cell?' " • **Answer**: "6, 9, 13"
Contact information	• **Question**: "How do I contact Jason? • **Answer**: • "Jason's email is jason@jxnl.co" • "Phone number: (408)-555-8796" • "You can contact Jason on X @jxnlco"	• **Question**: "What document has the contact information for Jason?" • **Answer**: "teamcontacts.docx"

When we use generation evals, **we assume there are no issues in the retrieval steps**

Data sets for creating generation evals are expensive to build and verify, so **many retrieval issues may go unnoticed**

Developers don't realize that **building out retrieval may be powerful enough for several initial use cases** to start gathering feedback on (and do generation afterwards)

Precision and recall are key metrics for evaluating Retrieval



	Recall	Precision
Definition	*% of relevant documents that are successfully retrieved* $$\frac{\text{(Relevant retrieved documents)}}{\text{(Total relevant documents)}}$$	*% of retrieved documents relevant to the query* $$\frac{\text{(Relevant retrieved documents)}}{\text{(Total retrieved documents)}}$$
Why is this important to measure?	• High recall means the system finds most of the relevant documents • This is especially important in cases where facts are hard to find across many documents	• High precision means the system retrieves mostly relevant documents • Dumber models might get more confused with irrelevant context

Case study: AI-generated reports for consultants

Personal anecdote about the importance of the recall metric

Situation

- Consultants do 15-30 research interviews with experts
- The consultants know which experts had specific perspectives about the products

Complication

- The consulting team requested an AI-generated report
- For certain fields, only found 3 out of 6 people quoted
- The consulting team expected the AI report to retrieve 6 quotes
- As a result, the team lost confidence in the system

Approach

- Shifted focus to ensure 100% of recall of all correct citations
  - Manually built Question -> chunk datasets for relevance

Impact

90%+ recall from 50% by doing significant pre-processing work

- Continues the sales processed by building trust and customer specific evaluations

Key takeaway:

- It's essential that the pre-processing that is done aligns with the anticipated queries to improve recall
- Leverage customer interactions to motivate your test suites

Case study: AI for blueprint search at construction company

Personal anecdote about the importance of pre-processing based on unique data sets

Situation

- Construction workers want to ask questions regarding blueprints

Complication

- We used VLLM labeled data we got ~27% recall@75 for image specific search tool
- We need better image summary prompts to be able to recover descriptions beyond image captioning

Approach

- Tested VLLM captioning and search over descriptions
- Used Chain of Thought (CoT) and asked language model to reason about the blue print before describing it
  - "How many rooms…"
  - "Describe the rooms…"
  - "How big was the room…"

Impact

85%+ recall from ~27% through pre-processing of blueprint data sets

4 days only were spent on iterating on summarization prompts

- 12 prompts and 3 models included
- Presented to design partners to obtain real user data with this level of recall
- ~20% of queries were related to counting objects in blueprints
  - Justified an investment to run bounding box models to attempt to count images

Key takeaway:

- Testing the ability of a single subsystem becomes very fast if we have the right baselines to test against
- Experimenting with high-specificity prompts for synthetic summary generation goes a long way in improving recall of images, tables, documents and any other artifacts

Agenda

Common pitfalls made by RAG developers

The RAG playbook

- Step 1-4: Initial implementation: focus on retrieval metrics
- **Step 2 Deep dive: Synthetic data generation: fast testing & review processes**

Homework for this session

Sneak peak for next session

Fake it until you make it

make it ≡ get users

The RAG Flywheel

Thesis: The principles we’ve applied in search are highly relevant to what we want to do with RAG

🟧 Deep dive

Step	Description
❶ **Initial implementation**	Start with a basic RAG system setup
❷ **Synthetic data generation**	Create synthetic questions to test the system’s retrieval abilities
❸ **Fast evaluations**	Conduct quick, unit test-like evaluations to assess basic retrieval capabilities (e.g., precisio recall, mean reciprocal rank), and explain why each matters
❹ **Real-world data collection**	Gather real user queries and interactions. Ensure feedback is aligned with business outcomes or correlated with important qualities that predict customer satisfaction
❺ **Classification and analysis**	Categorize and analyze user questions to identify patterns and gaps
❻ **System improvements**	Based on analysis, make targeted improvements to the system
❼ **Production monitoring**	Implement ongoing monitoring to track system performance
❽ **User feedback integration**	Continuously incorporate user feedback into the system

Procedure overview

If you have no user query data…

- Use existing data to generate evals to get a benchmark for precision and recall
    - question
    - answer
    - chunk_id
- Can be as simple as:

```
assert eval.chunk_id in search(eval.question,
limit=k).chunk_ids
```

If you have some user query data…

- Use query data as few shot examples for generation
- You can even generate chunks from queries and test if they can be retrieved
- Use LLM as a Ranker to produce **Weak Ranking labels**
    - Review the weak labels to get correct labels
    - Continue to test Precision / Recall metrics, or even ranking metrics
- **Ask yourself**: given what I know about the user data, what kinds of questions could I not answer with this method?

Collect everything and track the questions and create evals

- Every demo, every user interview
- Collect thumbs up ratings, allow users to delete sources
- Put it in a google sheet, just collect it, those labels are the gold.

Example prompt for question generation

Try to bake as much domain knowledge into these prompts, change prompts based on document types, be specific

You are tasked with generating relevant questions that a user of a search product might ask. This task is crucial for understanding user needs and improving the product's functionality.

<product_description>

{{PRODUCT_DESCRIPTION}}

</product_description>

Now, examine the example text data:

<document>

{{DOCUMENT}}

</document>

Here are some example of other queries users have asked:

<example_questions>

{{EXAMPLE QUESTIONS}}

</example_qustions>

To generate relevant questions:

1. Analyze the product description to understand its purpose and target users.
2. Examine the example text data to identify key information, patterns, or insights it might contain.
3. Consider the types of questions users might ask to extract valuable insight from the product.
4. Generate a diverse set of questions that cover different aspects of the data and product functionality.

When formulating questions, consider the following:

- Information extraction: What specific data points might users want to retrieve?
- Pattern recognition: What trends or relationships in the data might be of interest?
- Comparative analysis: How might users want to compare different elements in the data?
- Contextual understanding: What questions might help users better understand the context of the data?

Present your generated questions in the following format:

Before listing the questions, provide a brief explanation for each question, highlighting why it would be relevant and valuable to a user of this product. Present these explanations in the following format:

Example prompt for ranking generation

You are tasked with evaluating the relevance of text chunks to a given question. Your goal is to determine which chunks contain information that could be useful in answering the question, while preserving the original chunk IDs for later processing.

Here is the question you will be using to evaluate the relevance of the chunks:

<question>

{{QUESTION}}

</question>

You will be provided with a list of text chunks, each with a unique ID. The chunks will be in the following format:

<chunks>

{{CHUNKS}}

</chunks>

For each chunk, you should:

1. Carefully read and understand its content.
2. Evaluate whether the information in the chunk is relevant to answering the question.
3. Provide a brief explanation of why the chunk is or is not relevant.
4. Assign a relevance score from 0 to 1, where 0 is completely irrelevant and 1 is highly relevant.

Present your evaluation for each chunk in the following format:

<evaluation>

<chunk_id>[Insert the original chunk ID here]</chunk_id>

<reasoning>[Your explanation of relevance or irrelevance]</reasoning>

<relevance_score>[Your score from 0 to 1]</relevance_score>

</evaluation>

Important notes:

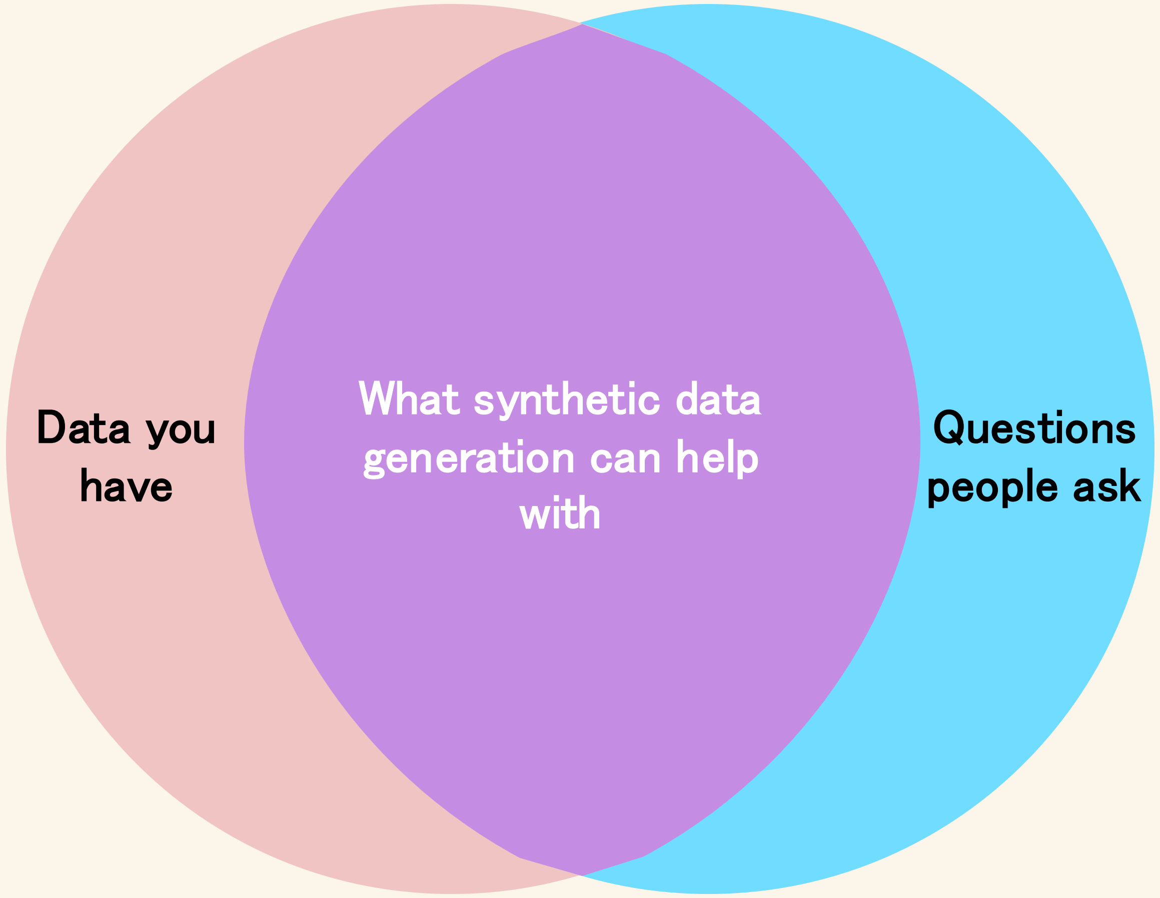
- Maintain objectivity in your evaluations.
- Focus on the content's relevance to the question, not its quality or completeness.
- Be consistent in your scoring across all chunks.
- Do not modify or summarize the original chunk text.
- If a chunk contains partial relevance, explain which parts are relevant and which are not.

How does this help me?

Troubleshoot early: By stress-testing your system with diverse synthetic queries, you can identify and address potential issues before they impact real users

Iterate quickly: With a well-structured synthetic dataset, you can quickly evaluate changes to your system, enabling rapid experimentation and improvement

Establish common ground: Synthetic data and its results can serve as a common reference point for discussions with team members, leadership, clients about system capabilities and goals



Agenda

Common pitfalls made by RAG developers

The RAG playbook

Homework for this session

Sneak peak for next session

Homework

- ☐ **Create an evaluation data set** by leveraging synthetic and user queries to improve precision and recall
- ☐ **Establish a baseline to run experiments on**
  - Check BM25, Embeddings, Chunkers, and Rerankers (LanceDB)
  - With baselines, we can "do nothing" if experiments don't pan out
- ☐ **Review generated question sets** with subject matter experts
  - LOOK at questions that generate low scores

Homework if you already have advanced automations in production:

- ☐ **Review user queries**: Sample some % of user traffic and run the LLM Ranker over retrieved text chunks and monitor questions that have low precision

  Ask yourself: "Could my system possibility answer these questions?"
- ☐ **Start to build process and flywheel:** Establish regular cadence to review questions or documents that get poor recall or even just low cosine or re-ranker scores

Experiments to run

- ☐ Try different embedding models
- ☐ Test how Rerankers improves retrieval, experiment with the top K with cosine and top N with reranker to see how to get better recall where N << K
- ☐ Test how Hybrid search improves retrieval
- ☐ Check if embedding additional metadata attached to text chunks helps
- ☐ Compare latency trade-offs for retrieval methods

If you anticipate common questions from users, use these questions as few-shot examples to generate more aligned questions!

If you run out of ideas lets chat in #show-your-work!!!

Common pitfalls to avoid

- **Oversimplification**: Don't create only simple, easily answerable questions, when your recall metrics are too high (> 85%), focus on blending in user data
- **Static datasets**: Avoid treating your synthetic data as a one-time creation. It should evolve with your system as you continue to learn about your user needs
- **Neglecting synthetic or real data**: As you accumulate real user data, don't abandon synthetic data. Instead, use both in complementary ways
- **Misalignment**: Make sure that the search implementation you are testing are the same ones in production, and that there is no misalignment in how things are configured or specified

Agenda

Common pitfalls made by RAG developers

The RAG playbook

Homework for this session

Sneak peak for next sessions

Sneak peek for next week

- **Focus for this week**:
  - Generate a single evaluation set and review the questions line by line, potentially with subject matter experts
  - Start logging user queries so we can use them to augment synthetic data generation
  - Eventually, we will run into limitations with this method. As we get more data, more opinions, and more diverse users, we will need to be able to "see the forest from the trees"…
- **Focus for next week**:
  - Finetuning, Embeddings, and Representations
- **We're still learning who you are!**
  - As we learn everyone's technical level we'll dig deeper during office hours and calibrate office hours to everyone's level
- **Remember**:
  - In this session, we' re just trying to establish baselines to have a null hypothesis. With this, we can start testing different interventions to see which make a real difference in our metrics
    - By using benchmarks, we can also **reject** interventions. We don' t have to just add superstitious interventions that become tech debt
    - As you collect more data, you can slowly transition your evals dataset to a training / fine-tuning dataset