jxnl.co

@jxnlco

Systematically Improving RAG Applications

Session 3

The Art of RAG UX: Turning Design into Data

Jason Liu

Overview

Three goals for today

1. Makes sure we're taking actions to collect feedback
2. Expand on what is possible with streaming
3. Give you a small set of prompting and UX tips to improve satisfaction and quality

Consider this mostly a survey of other techniques that I apply. This does not fit neatly into the other sections this course.

The past two sessions have been around: faking data, creating synthetic data, in hopes that one day user data will supplement the work that we're doing. ''

This session, the goal is to figure out how we can collect that user data and how can we give the users a good experience?

Agenda

Collecting more feedback

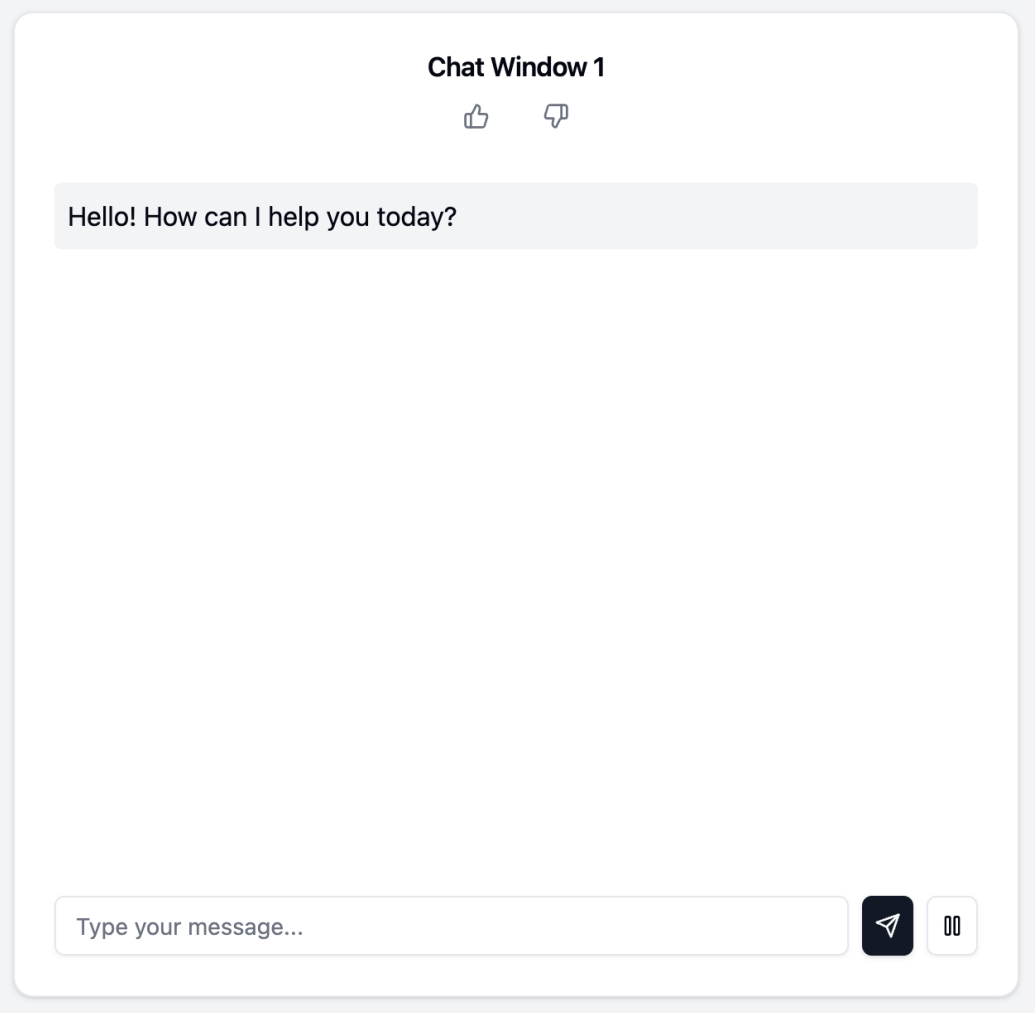
Streaming for better user satisfaction

Prompting and Chain of Thought

Look at your data

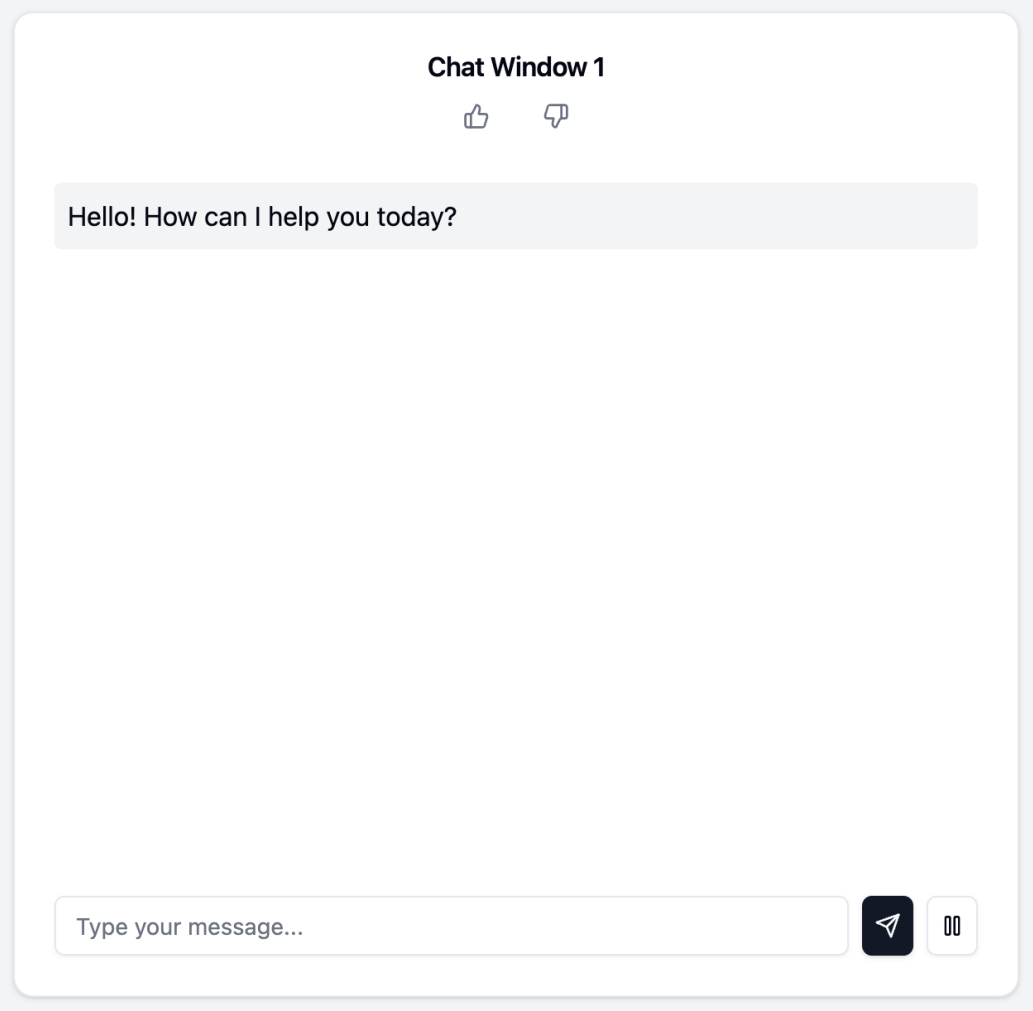
Getting user feedback is the second most important thing you can be doing after looking at your input data

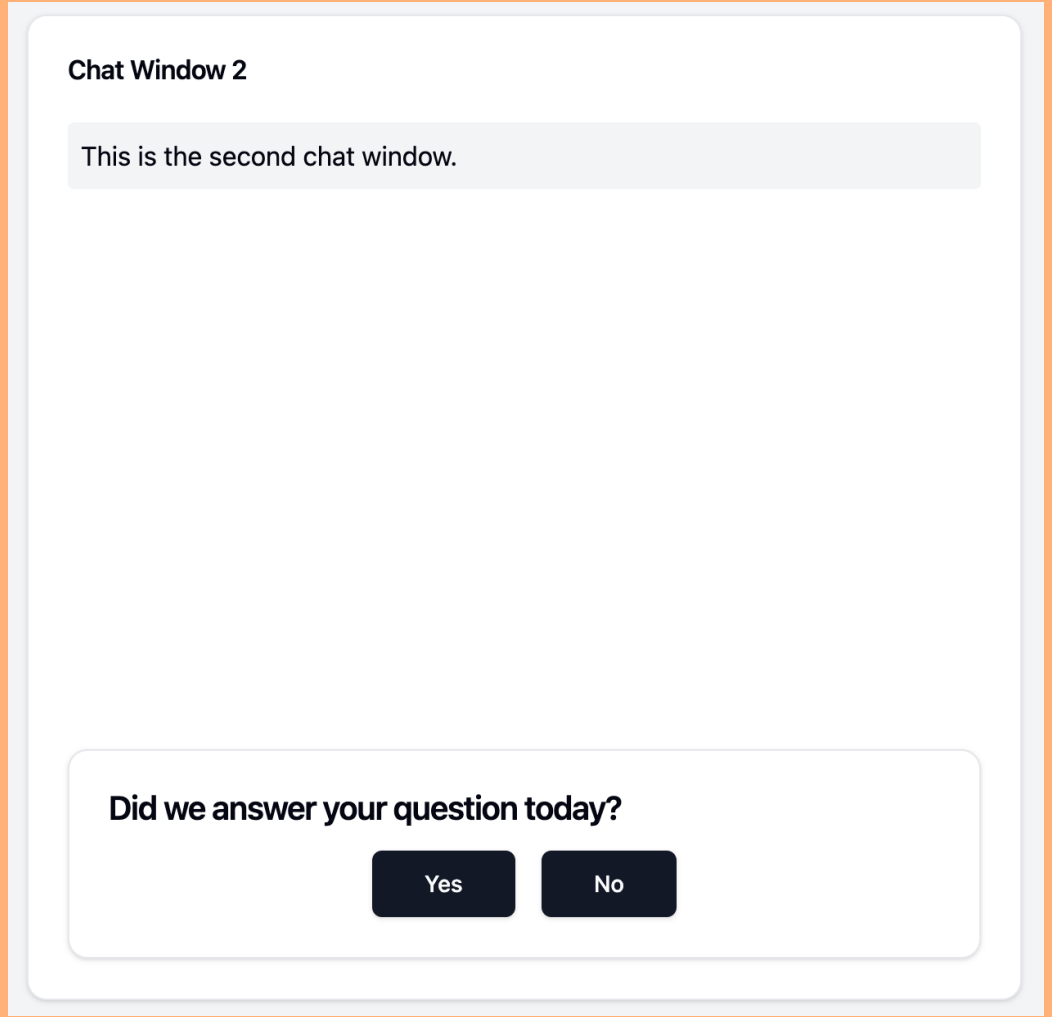
https://claude.site/artifacts/d57936fe-03c1-4815-8511-cbdb507d6d9c



Don’t be subtle

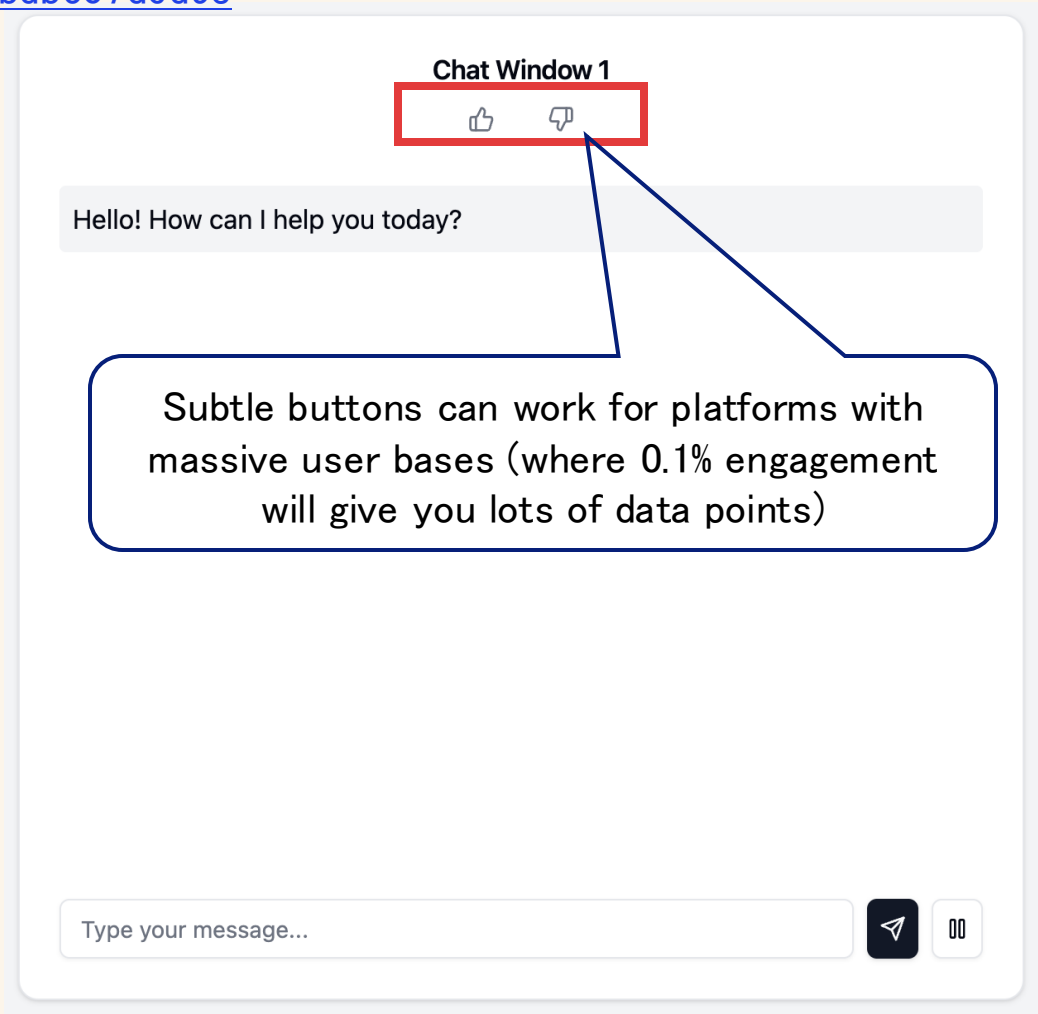
https://claude.site/artifacts/d57936fe-03c1-4815-8511-cbdb507d6d9c

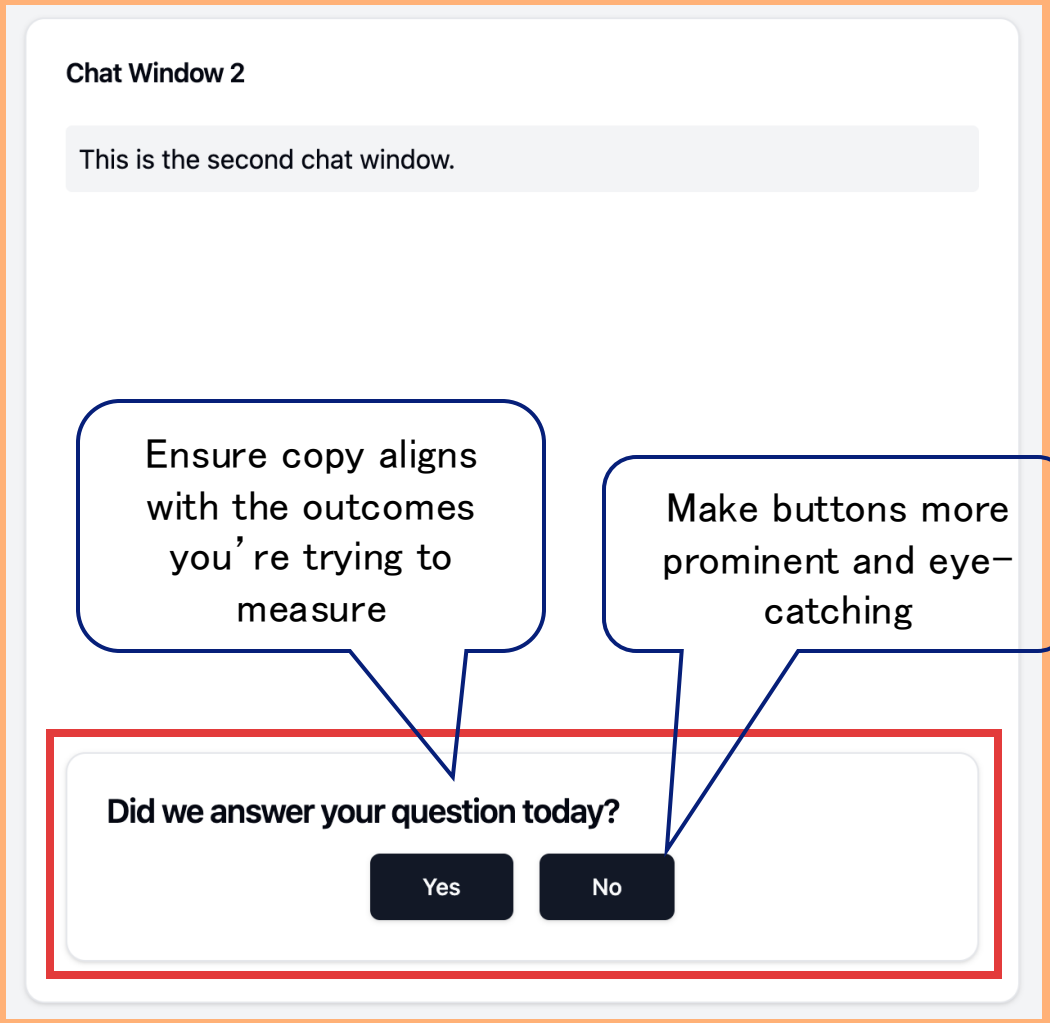




Don't be subtle

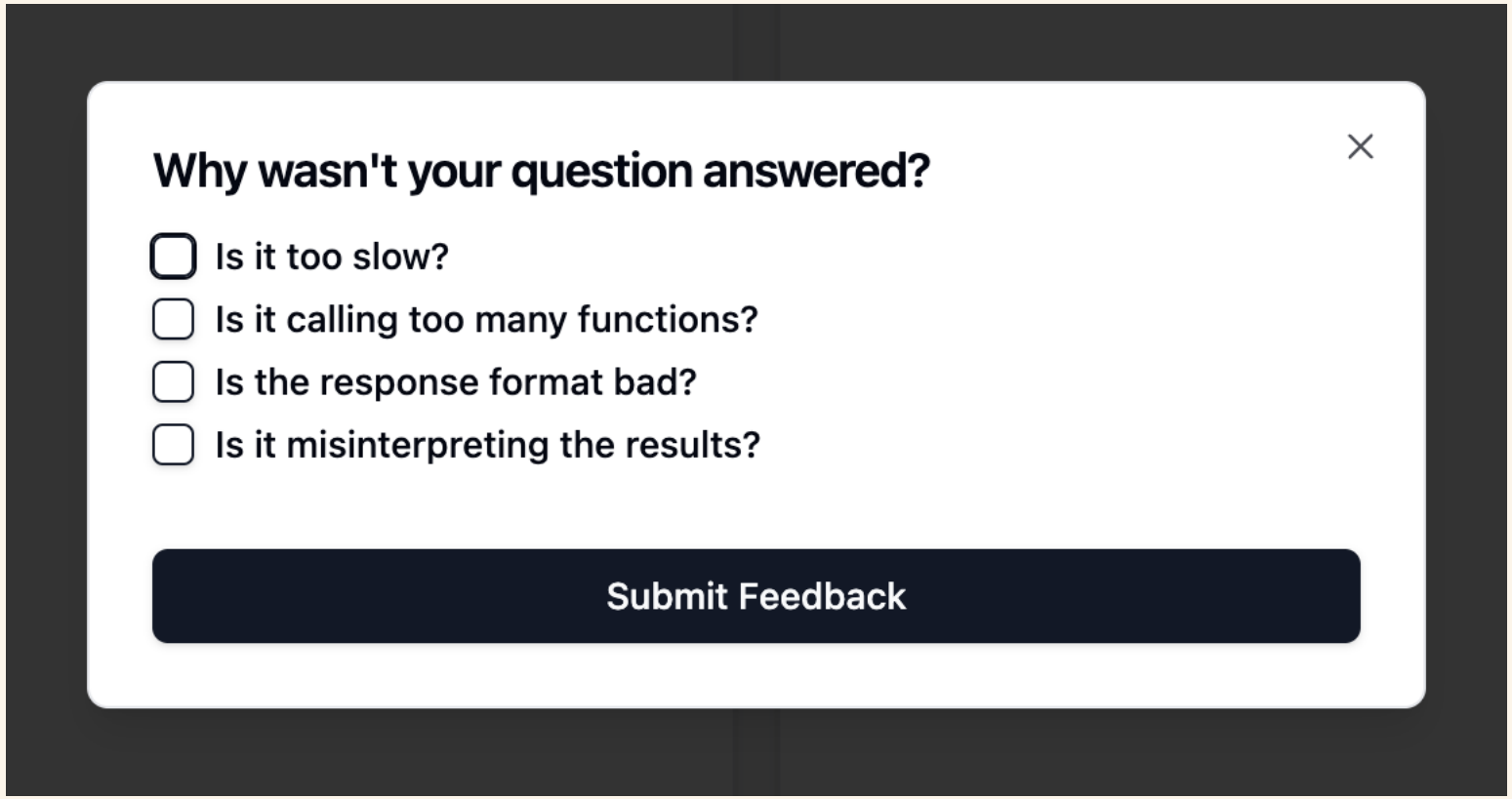
https://claude.site/artifacts/d57936fe-03c1-4815-8511-cbdb507d6d9c





Don't be subtle

https://claude.site/artifacts/d57936fe-03c1-4815-8511-cbdb507d6d9c



All this feedback can be used as part of your segmentation and exploration.

Question:

- Are there certain question segments that fall victim to these failure modes?
- Could we predict if a question might lead to a failure mode?

Feedback from enterprise customers

All of this works well for consumer cases. For enterprise, we'll have to try a lot harder

We need to hear about your negative feedback. It's essential in order for us to improve our application

What you share with us will be discussed:

- In a shared channel with Customer Success
- At our weekly/bi-weekly syncs

We'll bring up this negative feedback, add it to our evaluation framework (set), and report back to you how much they have improved over time

This is how we can drive the volume of feedback for our customers while building trust, collecting data, and building evals

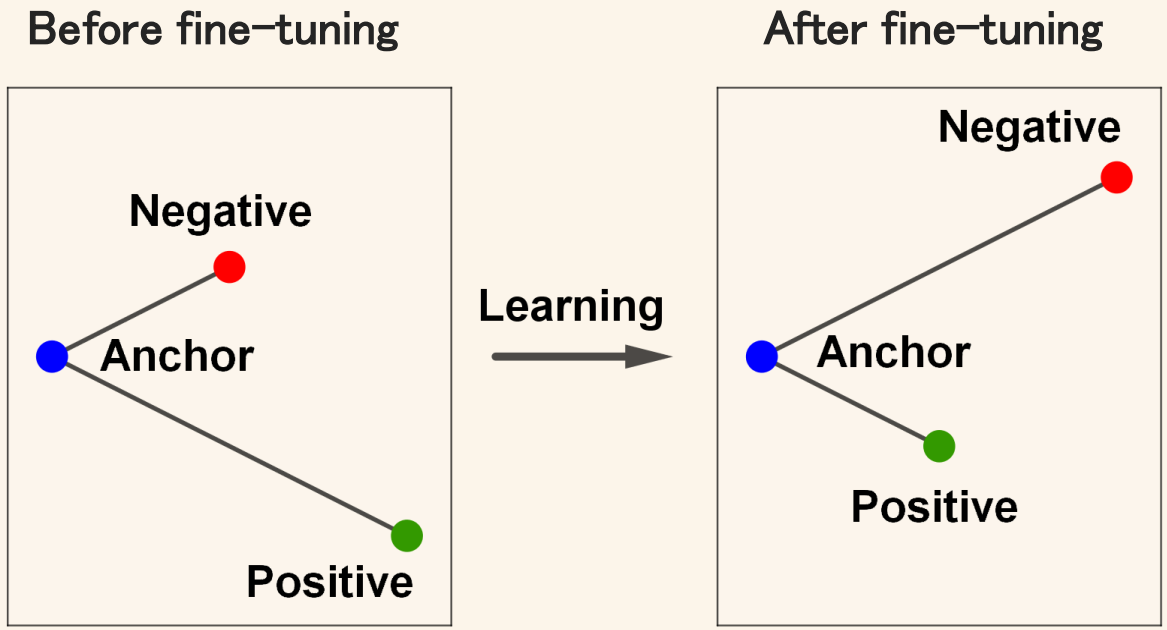
we can also use feedback it to improve our re-rankers and our embedding models (session 2).

For example, re-labeling data if the feedback is about relevance

Recall from our fine tune section the notion of triplets

So what happens when we finetune?

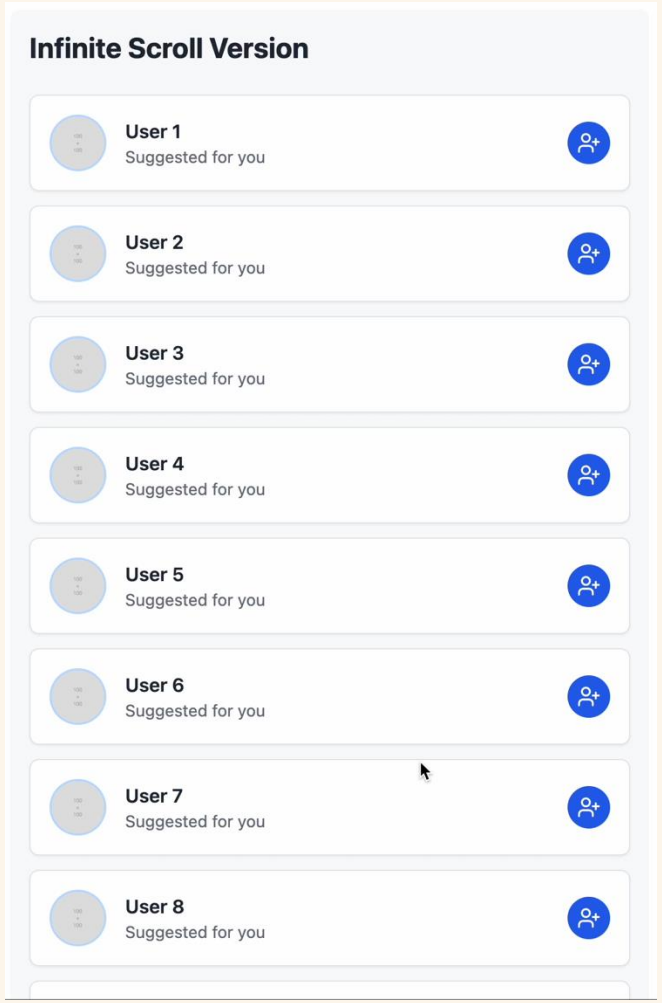
Create triplet examples (anchor and positive have same label, negative has different label)



The hardest part of this task is actually about finding the negative examples.

Finding Hard Negatives

Consider Facebook's people you may know feature



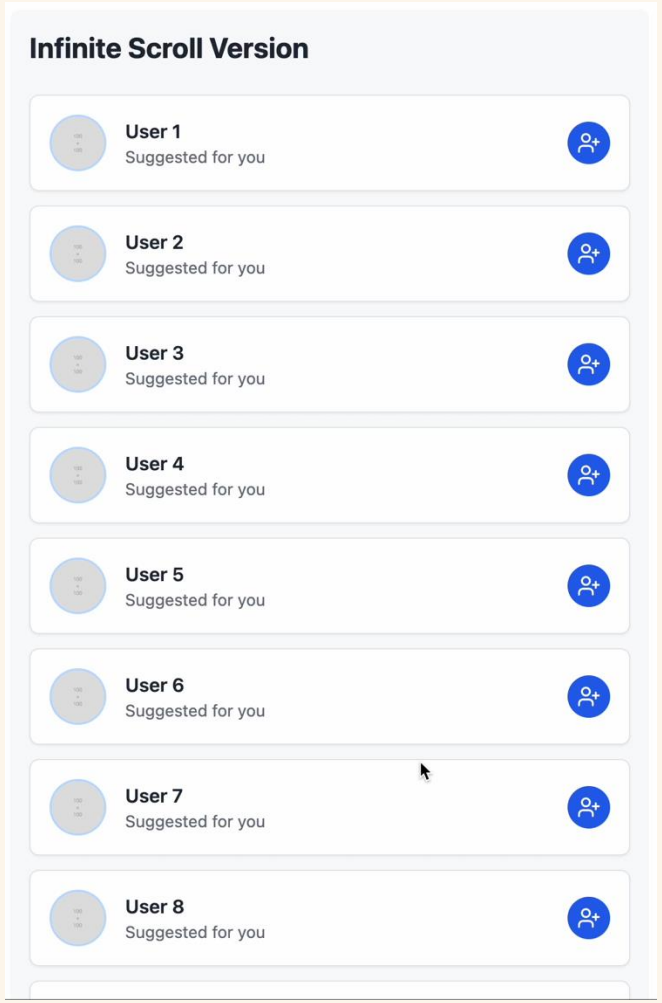
@jxnlco

https://claude.site/artifacts/8bf39407-3a66-46be-84ff-a042a4161485

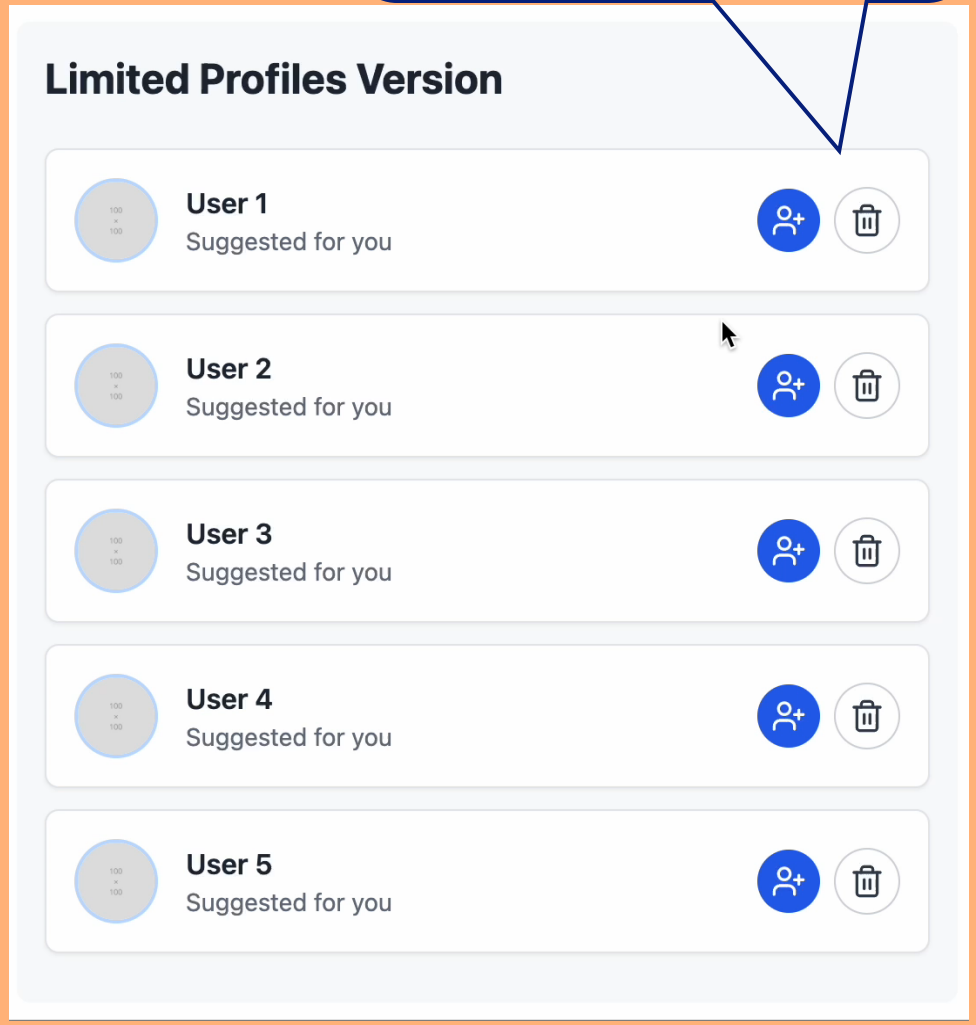
maven.com/applied-llms/rag-playbook

Finding Hard Negatives

Consider Facebook's people you may know feature



Requires deleting a profile to see a new one, naturally increasing the amount of data collected



@jxnlco

https://claude.site/artifacts/8bf39407-3a66-46be-84ff-a042a4161485

maven.com/applied-llms/rag-playbook

There will be numerous ways to collect positive and negative examples.

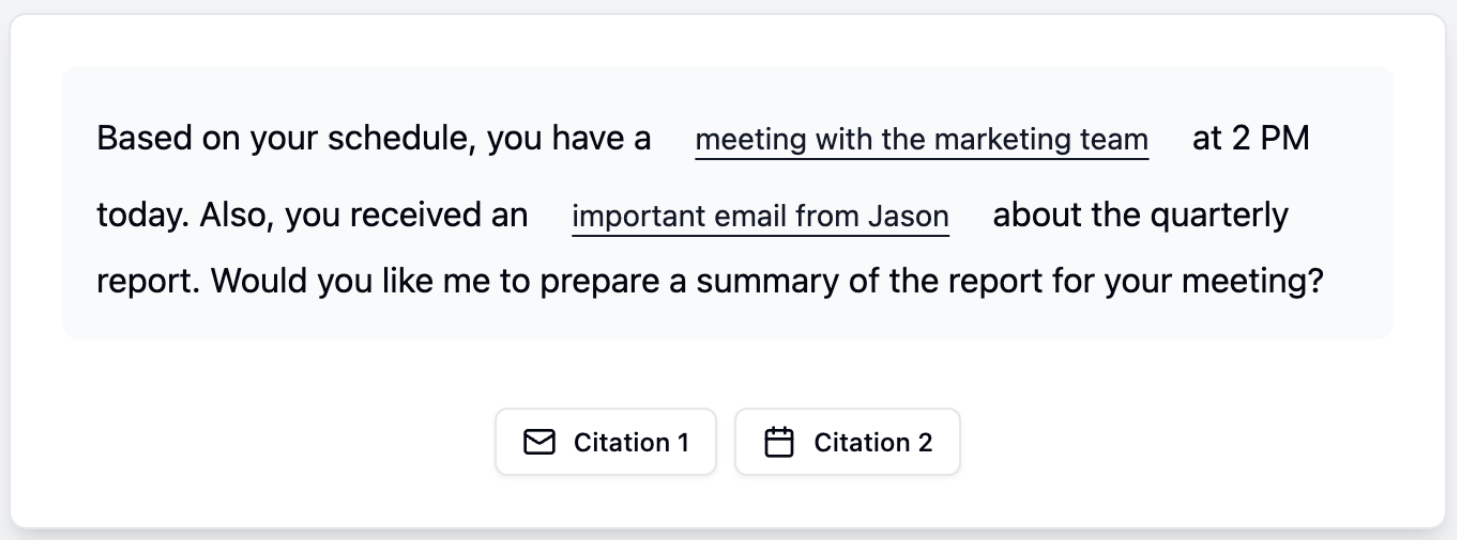
I firmly believe that the success of Tinder and Hinge is rooted in the volume of data they collect and their ability to obtain negatives.

- Lightweight Interaction (Swipe)
- Positive and Negative interactions (Like, Dislike)
- Simple objective (Match)

I believe I can do the same thing with citations, allowing us to collect data, while also building more trust in our system

Building Citations To Improve Trust

https://claude.site/artifacts/14cbd428-0382-44b8-9353-1c06d830b8dc



Building Citations :

- Allow Preview of Citation text
- ***Allow Delete/Regenerate in order to give negative feedback***

Building Citations To Improve Trust

```
<task>
  Your task is to provide a summary of the user's schedule and important
  messages, with citations to the original sources. Use the following
  format for citations: (cited text)[citation number].
</task>

<given_information>
  [1] The user has a calendar event: "Marketing Team Meeting" at 2 PM
  today.</item>
  [2] The user received an email from jason@company.com with the subject
  "Quarterly Report Due".</item>
</given_information>

Generate a brief summary that mentions both the meeting and the email,
using citations
Your response should be in JSON format with a "body" field for the main
text and a "citations" array for the citation details.</step>
```

Generate citations to build satisfaction and trust (cont'd)

```
<task>
  Your task is to provide a summary of the user's schedule and important
  messages, with citations to the original sources. Use the following
  format for citations: (cited text)[citation number].
</task>

<given_information>
  [1] The user has a calendar event: "Marketing Team Meeting" at 2 PM
  today.</item>
  [2] The user received an email from jason@company.com with the subject
  "Quarterly Report Due".</item>
</given_information>

Generate a brief summary that mentions both the meeting and the email,
using citations
Your response should be in JSON format with a "body" field for the main
text and a "citations" array for the citation details.</step>
```

Provide a prompt

Return a structured object

```
{
  "body": "Based on your schedule, you have a (meeting with the marketing team)[2] at 2 PM today.
  Also, you received an (important email from Jason)[1] about the quarterly report. Would you like
  me to prepare a summary of the report for your meeting?",
  "citations": [
    {
      "chunk_id": 1,
      "title": "Email from jason@company.com: Subject: Quarterly Report Due"
    },
    {
      "chunk_id": 2,
      "title": "Calendar event: Marketing Team Meeting, 2 PM"
    }
  ]
}
```

Negative examples: generate citations to build satisfaction and trust (cont'd)

```
<task>
  Your task is to provide a summary of the user's schedule and important
  messages, with citations to the original sources. Use the following
  format for citations: (cited text)[citation number].
</task>

<given_information>
  [1] The user has a calendar event: "Marketing Team Meeting" at 2 PM
  today.</item>
  [2] The user received an email from jason@company.com with the subject
  "Quarterly Report Due".</item>
</given_information>

Generate a brief summary that mentions both the meeting and the email,
using citations
Your response should be in JSON format with a "body" field for the main
text and a "citations" array for the citation details.</step>
```

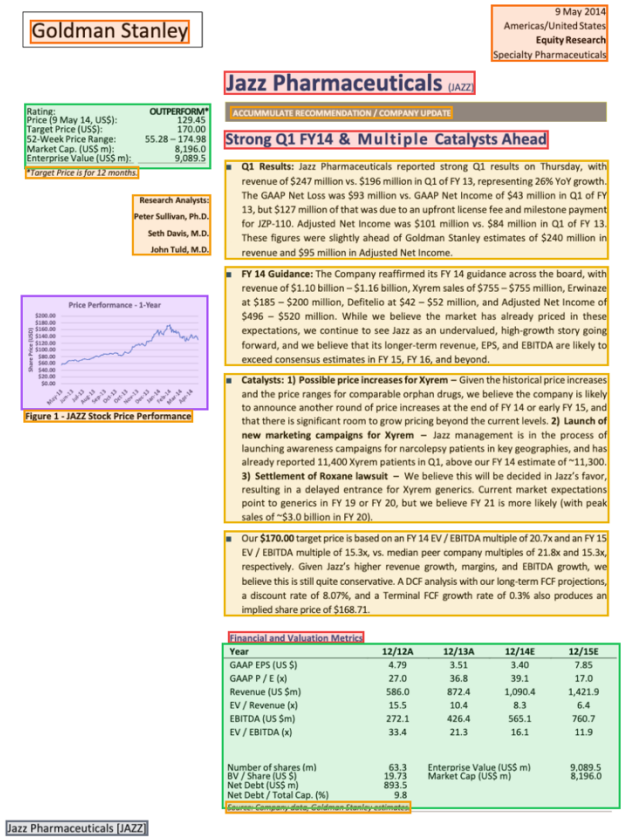
Provide a prompt

Return a structured object

```
{
  "body": "Based on your schedule, you have a (meeting with the marketing team)[2] at 2 PM today.
  Also, you received an (important email from Jason)[1] about the quarterly report. Would you like
  me to prepare a summary of the report for your meeting?",
  "citations": [
    {
      "chunk_id": 1,
      "title": "Email from jason@company.com: Subject: Quarterly Report Due"
    },
    {
      "chunk_id": 2,
      "title": "Calendar event: Marketing Team Meeting, 2 PM"
    }
  ]
}
```

```
{
  "query": ...
  "chunk_ids": ["chunk1", "chunk2", "chunk3"],
  "neg_chunk_ids": ["chunk2"]
}
```

Use consistent formatting for links and employ IDs as pointers



Goldman Stanley

9 May 2014
Americas/United States
Equity Research
Specialty Pharmaceuticals

Jazz Pharmaceuticals (JAZZ)

ACCUMMULATE RECOMMENDATION / COMPANY UPDATE

Strong Q1 FY14 & Multiple Catalysts Ahead

Rating: OUTPERFORM*
Price (9 May 14, US$): 129.45
Target Price (US$): 170.00
52-Week Price Range: 55.28 – 174.98
Market Cap. (US$ m): 8,196.0
Enterprise Value (US$ m): 9,089.5
*Target Price is for 12 months.

Research Analysts:
Peter Sullivan, Ph.D.
Seth Davis, M.D.
John Tuld, M.D.

- **Q1 Results:** Jazz Pharmaceuticals reported strong Q1 results on Thursday, with revenue of $247 million vs. $196 million in Q1 of FY 13, representing 26% YoY growth. The GAAP Net Loss was $93 million vs. GAAP Net Income of $43 million in Q1 of FY 13, but $127 million of that was due to an upfront license fee and milestone payment for JZP-110. Adjusted Net Income was $101 million vs. $84 million in Q1 of FY 13. These figures were slightly ahead of Goldman Stanley estimates of $240 million in revenue and $95 million in Adjusted Net Income.
- **FY 14 Guidance:** The Company reaffirmed its FY 14 guidance across the board, with revenue of $1.10 billion – $1.16 billion, Xyrem sales of $755 – $755 million, Erwinaze at $185 – $200 million, Defitelio at $42 – $52 million, and Adjusted Net Income of $496 – $520 million. While we believe the market has already priced in these expectations, we continue to see Jazz as an undervalued, high-growth story going forward, and we believe that its longer-term revenue, EPS, and EBITDA are likely to exceed consensus estimates in FY 15, FY 16, and beyond.
- **Catalysts: 1) Possible price increases for Xyrem –** Given the historical price increases and the price ranges for comparable orphan drugs, we believe the company is likely to announce another round of price increases at the end of FY 14 or early FY 15, and that there is significant room to grow pricing beyond the current levels. **2) Launch of new marketing campaigns for Xyrem –** Jazz management is in the process of launching awareness campaigns for narcolepsy patients in key geographies, and has already reported 11,400 Xyrem patients in Q1, above our FY 14 estimate of ~11,300. **3) Settlement of Roxane lawsuit –** We believe this will be decided in Jazz's favor, resulting in a delayed entrance for Xyrem generics. Current market expectations point to generics in FY 19 or FY 20, but we believe FY 21 is more likely (with peak sales of ~$3.0 billion in FY 20).
- Our **$170.00** target price is based on an FY 14 EV / EBITDA multiple of 20.7x and an FY 15 EV / EBITDA multiple of 15.3x, vs. median peer company multiples of 21.8x and 15.3x, respectively. Given Jazz's higher revenue growth, margins, and EBITDA growth, we believe this is still quite conservative. A DCF analysis with our long-term FCF projections, a discount rate of 8.07%, and a Terminal FCF growth rate of 0.3% also produces an implied share price of $168.71.

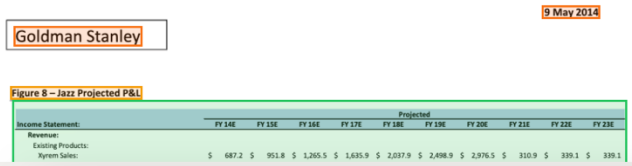
Figure 1 - JAZZ Stock Price Performance

Financial and Valuation Metrics

Year	12/12A	12/13A	12/14E	12/15E
GAAP EPS (US $)	4.79	3.51	3.40	7.85
GAAP P / E (x)	27.0	36.8	39.1	17.0
Revenue (US $m)	586.0	872.4	1,090.4	1,421.9
EV / Revenue (x)	15.5	10.4	8.3	6.4
EBITDA (US $m)	272.1	426.4	565.1	760.7
EV / EBITDA (x)	33.4	21.3	16.1	11.9
Number of shares (m)	63.3	Enterprise Value (US$ m)		9,089.5
BV / Share (US $)	19.73	Market Cap (US$ m)		8,196.0
Net Debt (US$ m)	893.5			
Net Debt / Total Cap. (%)	9.8			

Source: Company data, Goldman Stanley estimates

Jazz Pharmaceuticals (JAZZ)



Goldman Stanley

9 May 2014

Figure 8 – Jazz Projected P&L

Income Statement:	Projected FY 14E	FY 15E	FY 16E	FY 17E	FY 18E	FY 19E	FY 20E	FY 21E	FY 22E	FY 23E
Revenue:										
Existing Products:										
Xyrem Sales:	$ 687.2	$ 951.8	$ 1,265.5	$ 1,635.9	$ 2,037.9	$ 2,498.9	$ 2,976.5	$ 310.9	$ 339.1	$ 339.1

Goldman Stanley

9 May 2014
Americas/United States
Equity Research
Specialty Pharmaceuticals

Jazz Pharmaceuticals (JAZZ)

Reducto AI Table Summary
This table provides an overview of the investment rating and financial information for Jazz Pharmaceuticals (JAZZ). The company is rated as "OUTPERFORM*" by the analysts. The current stock price is $129.45 as of May 9, 2014, and the target price is $170.00. The 52-week price range for the stock is $55.28 to $174.98. The company's market capitalization is $8,196.0 million, and its enterprise value is $9,089.5 million.

Rating:	**OUTPERFORM***
Price (9 May 14, US$):	129.45
Target Price (US$):	170.00
52-Week Price Range:	55.28-174.98
Market Cap. (US$ m):	8,196.0
Enterprise Value (US$ m):	9,089.5

*Target Price is for 12 months.

Research Analysts: Peter Sullivan, Ph.D. Seth Davis, M.D. John Tuld, M.D.

Figure Text:
| Date | Price (USD) | |----------|----------| | Mar '13 | ~$60 | | Jun '13 | ~$80 | | Sep '13 | ~$100 | | Dec '13 | ~$120 | | Mar '14 | ~$160 | | Jun '14 | ~$180 | | Sep '14 | ~$140 | Note: The table provides approximate values at quarterly intervals. The actual chart shows more frequent data points and fluctuations throughout the year.

Figure 1 – JAZZ Stock Price Performance

ACCUMMULATE RECOMMENDATION / COMPANY UPDATE

Strong Q1 FY14 & Multiple Catalysts Ahead

Agenda

Collecting more feedback

Streaming for better user satisfaction

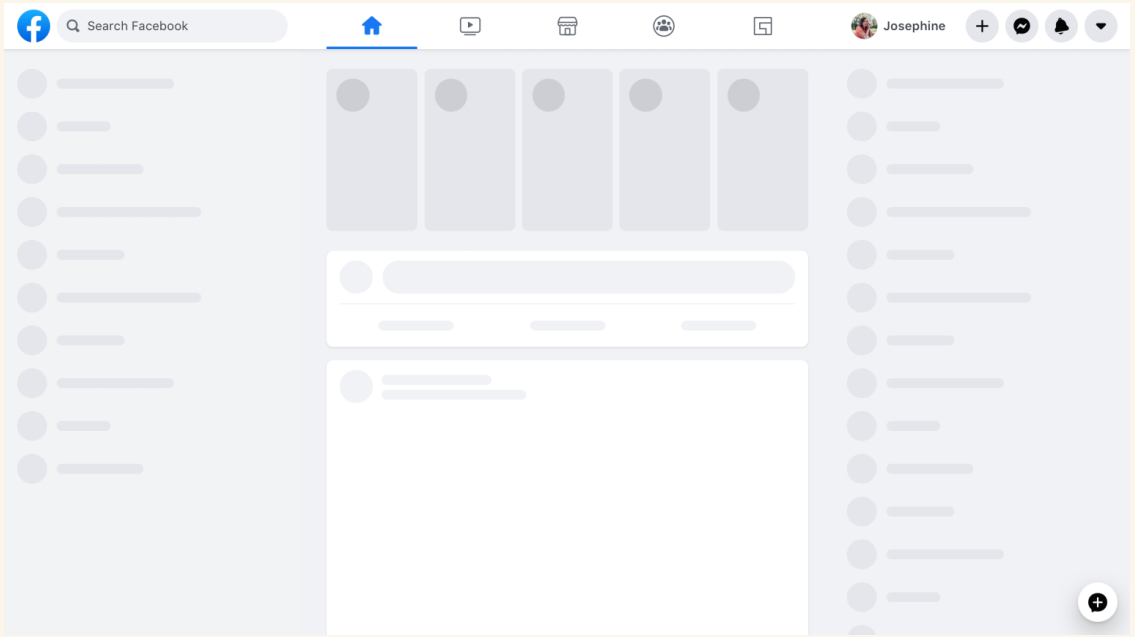
Prompting and Chain of Thought

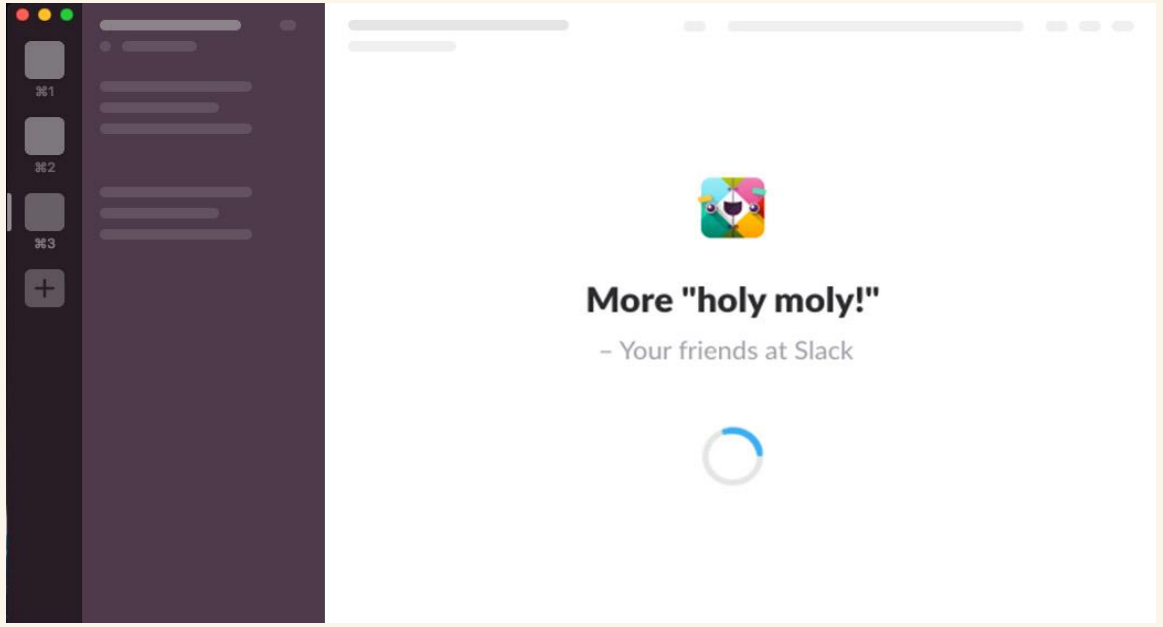
Streaming Overview

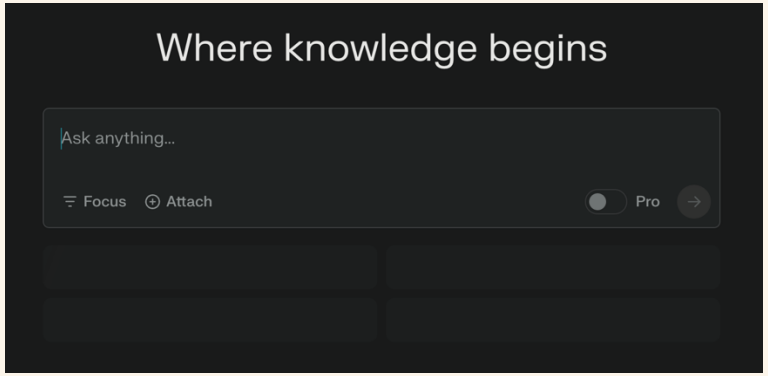
1. Stream interstitials to explain latency
2. Stream results and UI
3. Stream tool calls!

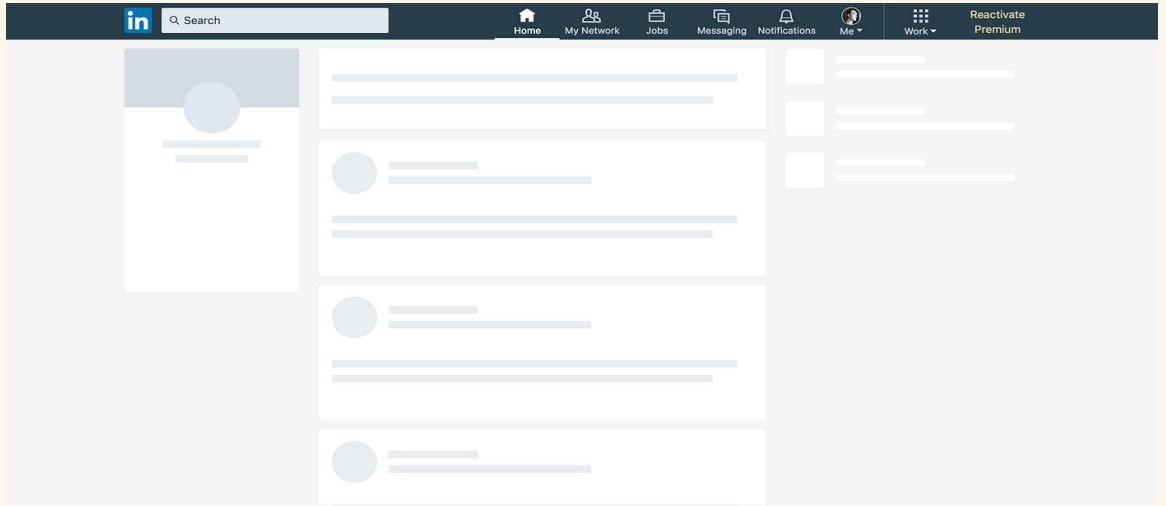
Skeleton screens are everywhere

Once you see them, you cannot unsee them









@jxnlco

maven.com/applied-llms/rag-playbook

Streaming overview

Why we should stream:

Lower perceived wait time

- Users perceive animated progress bars as 11% faster, even with equal wait times (Harrison et al.).

Reduce user churn / disengagement

- Users will tolerate up to 8 seconds of waiting if given visual feedback reducing abandonment (Nah).

Increase satisfaction and trust

- Applications with engaging loading screens often report higher user satisfaction scores.
- Facebook discovered that skeleton screens reduced perceived load times, resulting in better user retention and engagement.

Harrison, C., Yeo, Z., & Hudson, S. E. (2010). Faster progress bars: manipulating perceived duration with visual augmentations. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.
Nah, F. F. (2004). A study on tolerable waiting time: how long are Web users willing to wait?. *Behaviour & Information Technology*.

Streaming overview

Why we should stream:

Lower perceived wait time

- Users perceive animated progress bars as 11% faster, even with equal wait times (Harrison et al.).

Reduce user churn / disengagement

- Users will tolerate up to 8 seconds of waiting if given visual feedback reducing abandonment (Nah).

Increase satisfaction and trust

- Applications with engaging loading screens often report higher user satisfaction scores.
- Facebook discovered that skeleton screens reduced perceived load times, resulting in better user retention and engagement.

Harrison, C., Yeo, Z., & Hudson, S. E. (2010). Faster progress bars: manipulating perceived duration with visual augmentations. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.
Nah, F. F. (2004). A study on tolerable waiting time: how long are Web users willing to wait?. *Behaviour & Information Technology*.

What we can stream:

- Responses (incl. citations or follow-up questions)
- Arguments of function calls
- Interstitials

We should assess:

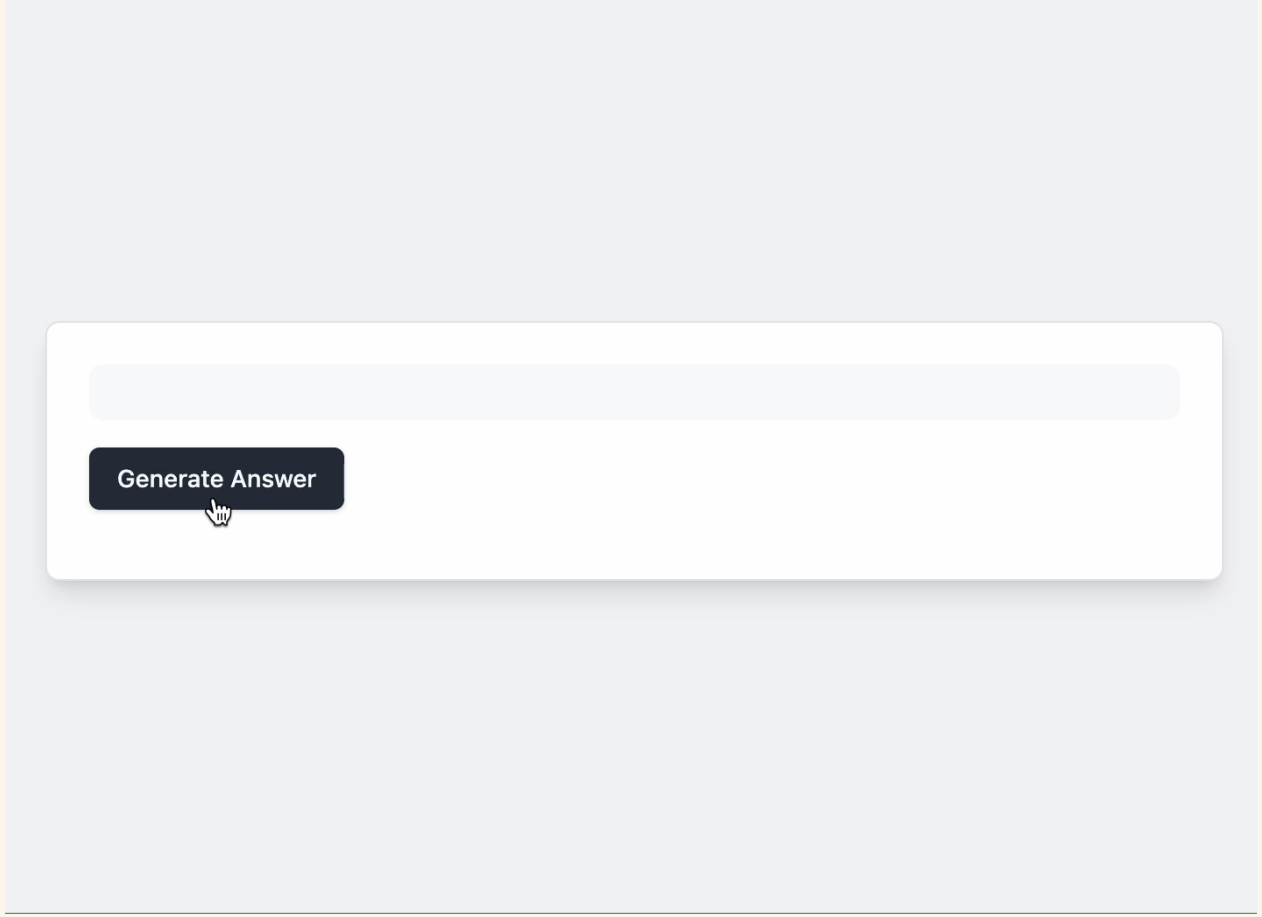
- Track user feedback to understand user demand
- Migrating to streaming is complex and challenging
- Understand how much latency matters in your application

If you're thinking about it...

Migrating from non-streaming to streaming is a pain the ass, you either have to build it from the start or it'll take weeks out of your dev cycle to 'upgrade'

It's worth it.

Stream and parse the response and other attributes (follow-up actions) in real time

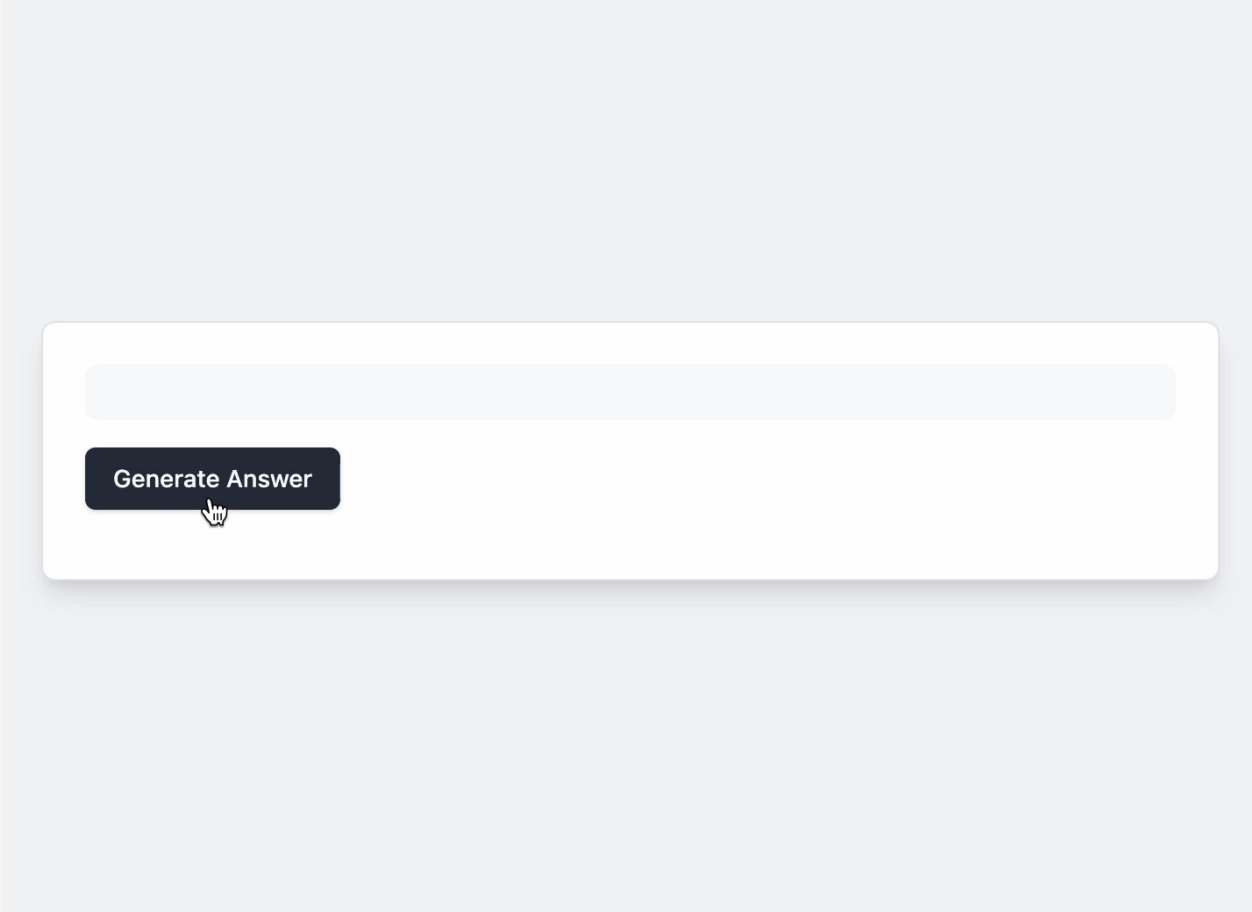


https://claude.site/artifacts/8644e9fc-939a-4520-8fa2-58f589f929d3

@jxnlco

maven.com/applied-llms/rag-playbook

Stream and parse the response and other attributes (follow-up actions) in real time




```python
class Citation(BaseModel):
    id: str
    title: str

class RespondToUser(BaseModel):
    content: str
    follow_actions: List[str]
    sources: List[Citation]
```

```python
client = instructor.from_openai(client)

partial_response = client.create_partial(
    messages=[...]
    model=...
    response_model=RespondToUser
)

for response in partial_response:
    yield response
```

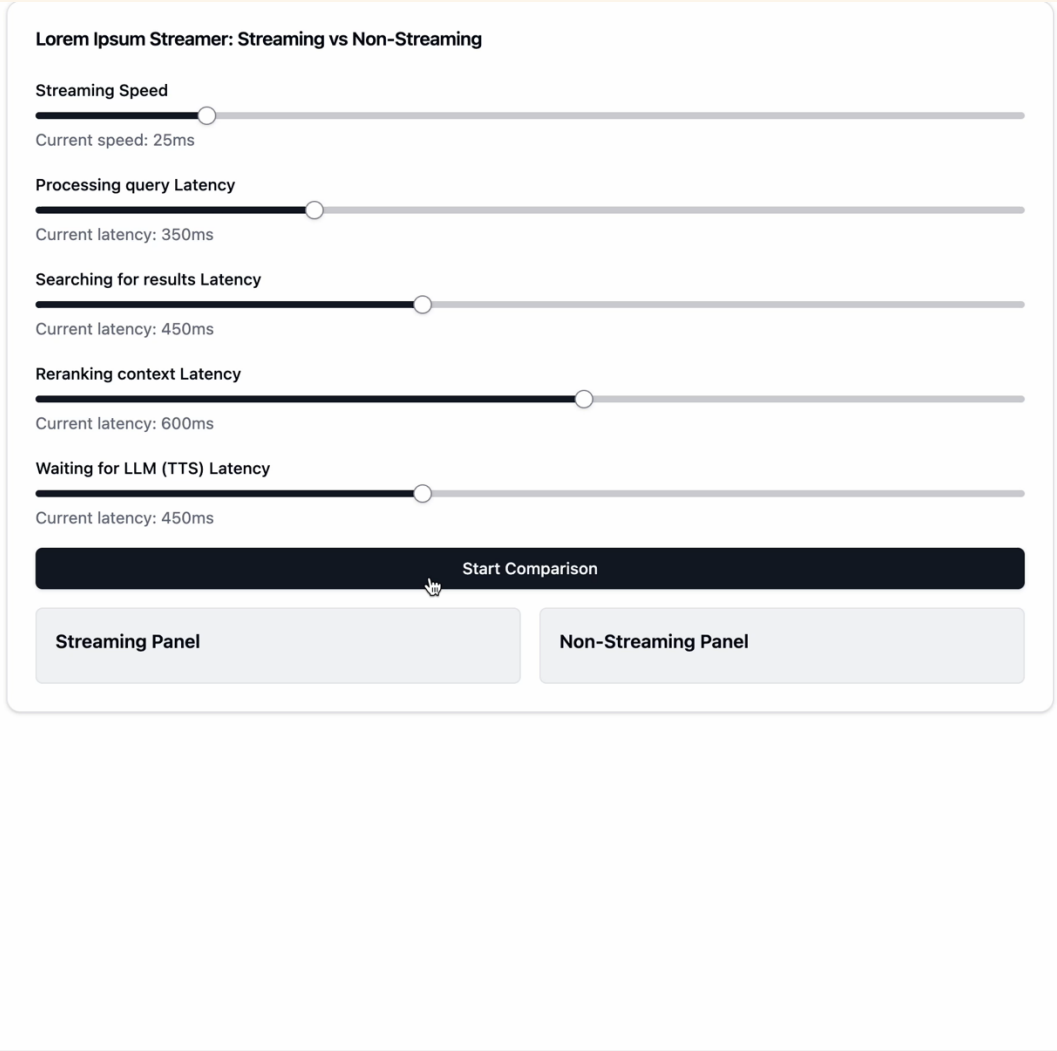
https://claude.site/artifacts/8644e9fc-939a-4520-8fa2-58f589f929d3

@jxnlco

maven.com/applied-llms/rag-playbook

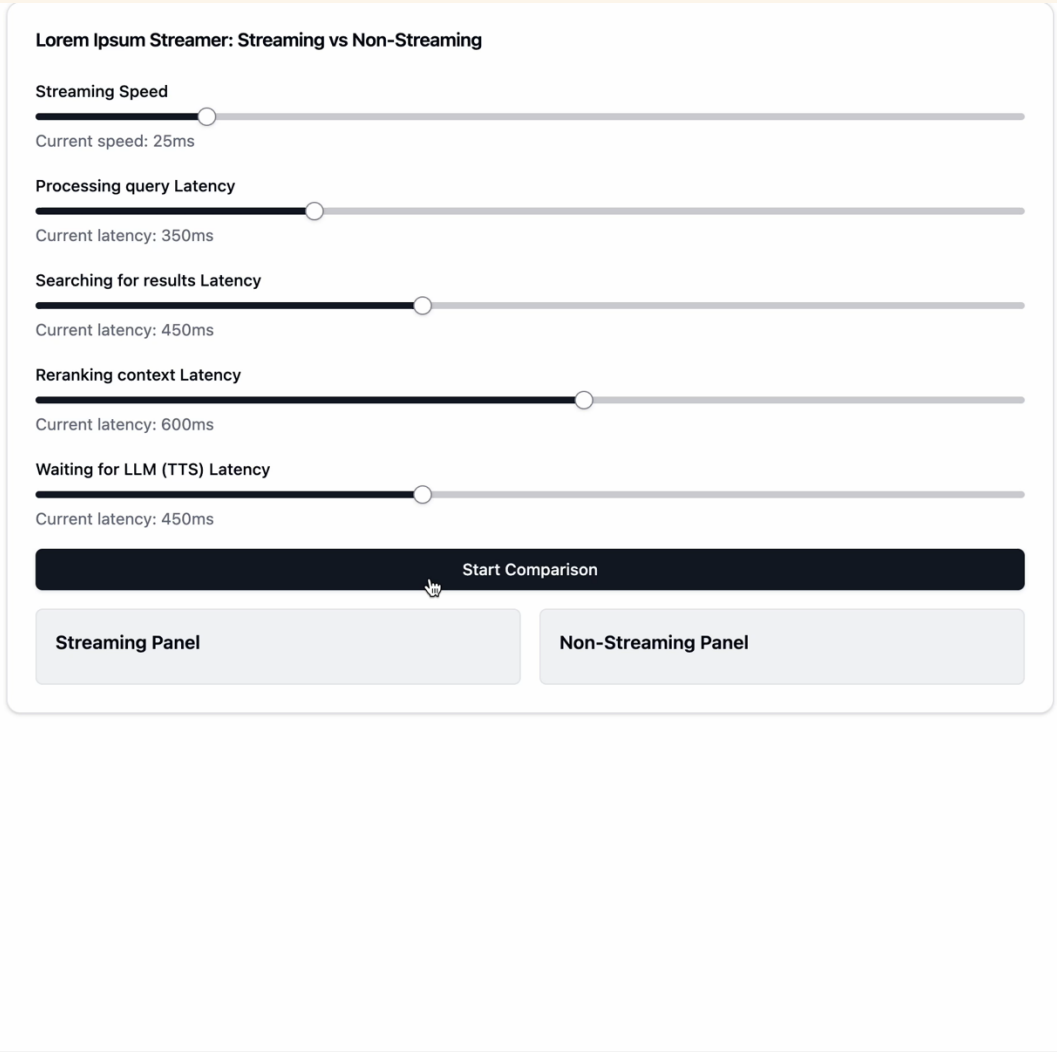
If you build a lot of follow-up action UI, you can build datasets and again use retrieval to few-shot your prompts to take follow-up actions

Interstitials for explainability



@jxnlco https://claude.site/artifacts/bba5efe9-1d41-49ff-a43f-05d3d349e193

Interstitials for explainability



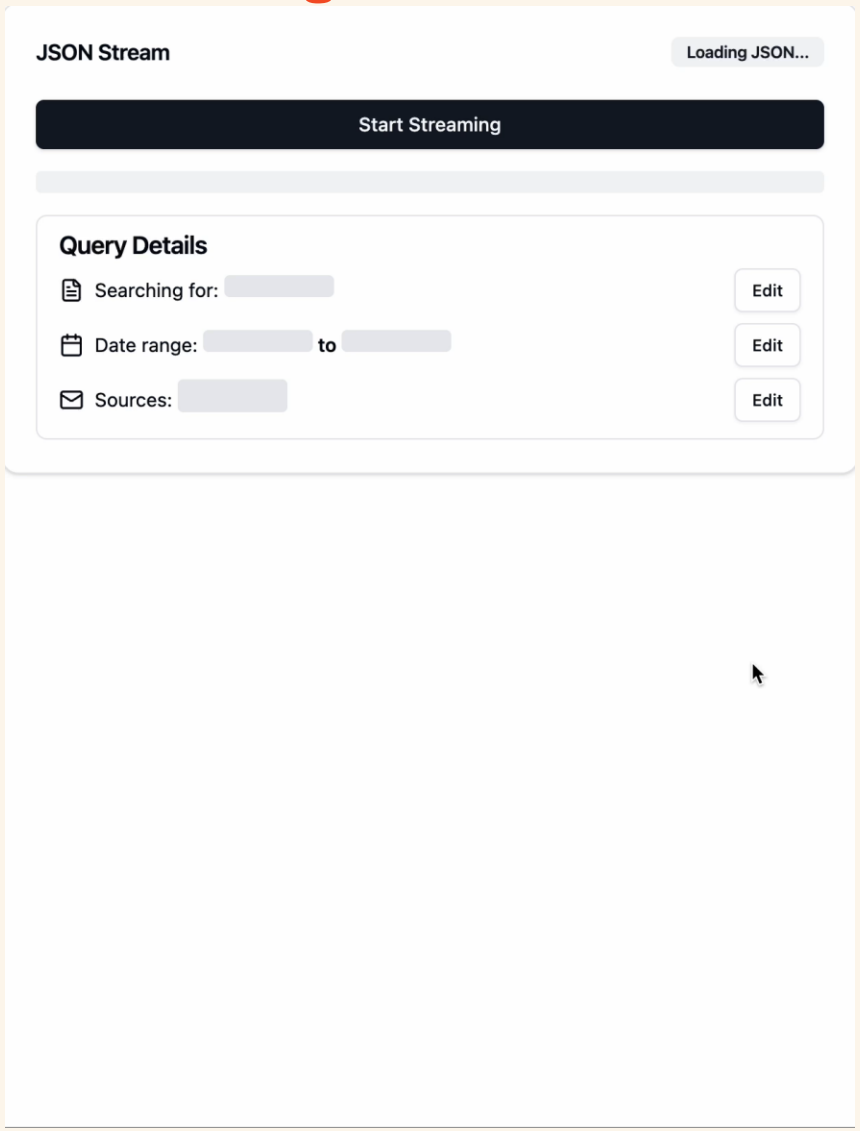
@jxnlco https://claude.site/artifacts/bba5efe9-1d41-49ff-a43f-05d3d349e193

Use webhooks, server-sent events, or a generator to stream

```python
from fastapi import FastAPI, Request
from fastapi.responses import StreamingResponse
import random
import asyncio
import json

app = FastAPI()

...

async def sse_stream(request: Request, query: str):
    async def event_generator():
        try:
            # Understand query
            yield json.dumps({"event": "thinking", "data": "..."})
            search = await understand_query(query)
            yield json.dumps({"event": "tool", "data": query_understanding.model_dump()})

            # Search
            yield json.dumps({"event": "searching", "data": "..."})
            search_results = await search.execute()
            yield json.dumps({"event": "search_results", "data": search_results})

            # Generate answer
            answer_stream = await generate_answer(query, search_results)
            for chunk in answer_stream
              yield json.dumps({"event": "generating", "data": chunk.model_dump()})

            yield json.dumps({"event": "complete", "data": "[DONE]"})
        except Exception as e:
            yield json.dumps({"event": "error", "data": str(e)})

    return EventSourceResponse(event_generator())

@app.get("/query")
async def process_query(request: Request, query: str):
    return StreamingResponse(sse_stream(request, query), media_type="text/event-stream")
```

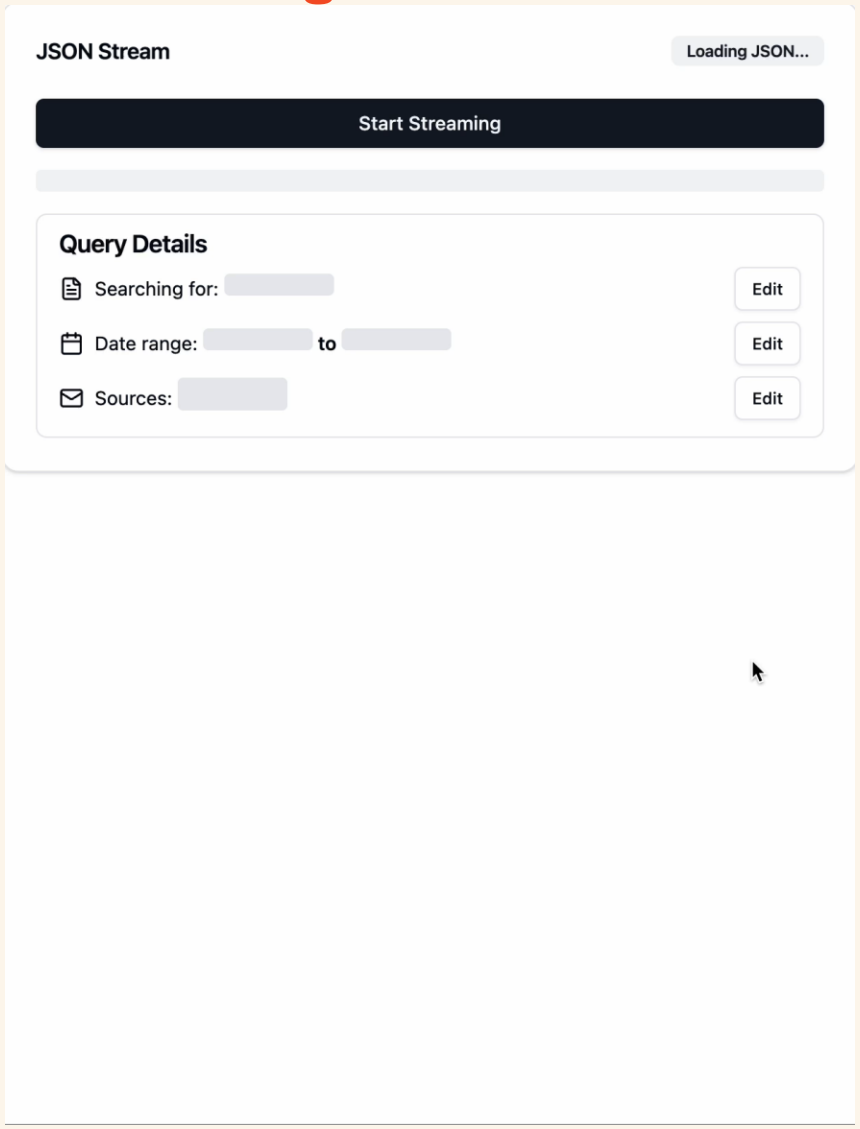
maven.com/applied-llms/rag-playbook

Stream tool arguments and render in UI



https://claude.site/artifacts/1a2a1e9a-e5d7-4511-a299-211d70fdeb3b

Stream tool arguments and render in UI



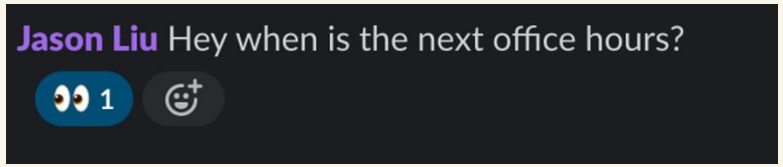
Benefits

- Create UI that allows users to edit and rerun tools
  - This becomes data collection that we can leverage to create better few-shot examples or fine-tuning data.
- Enables collection of better feedback data
- Supports analytics on what questions have additional attributes

https://claude.site/artifacts/1a2a1e9a-e5d7-4511-a299-211d70fdeb3b

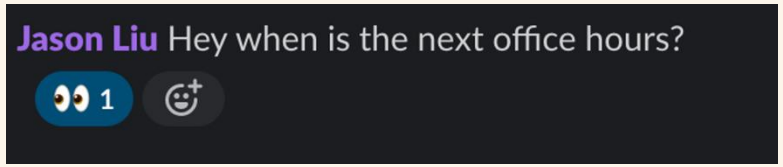
Simple Example: Slack Bots

As an interstitial, the **slack integration can react with the eyes emoji** to communicate it has seen the user's message

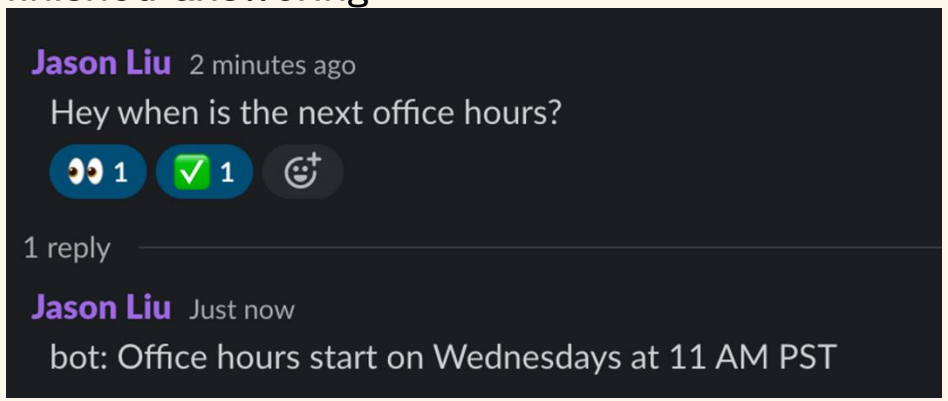


Simple Example: Slack Bots

As an interstitial, the **slack integration can react with the eyes emoji** to communicate it has seen the user's message

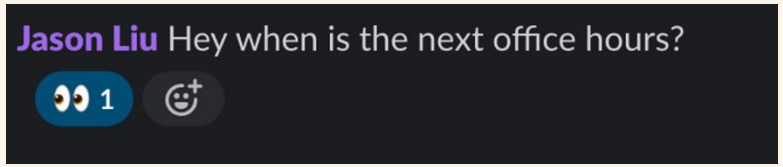


Use a checkmark to communicate the bot has finished answering

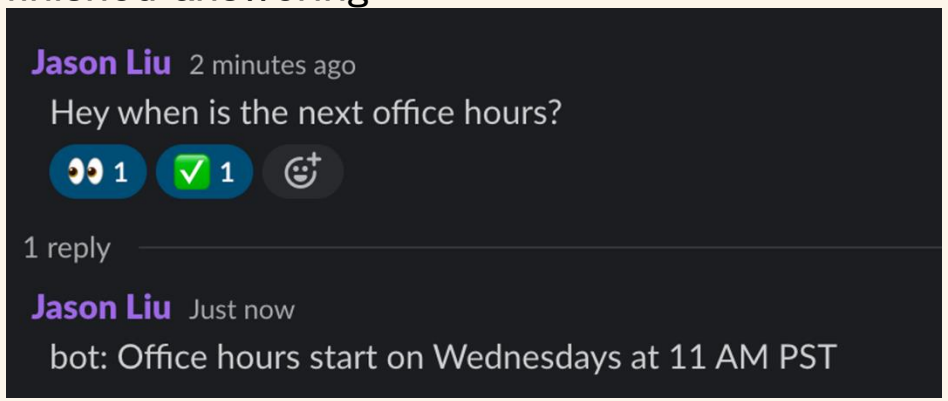


Simple Example: Slack Bots

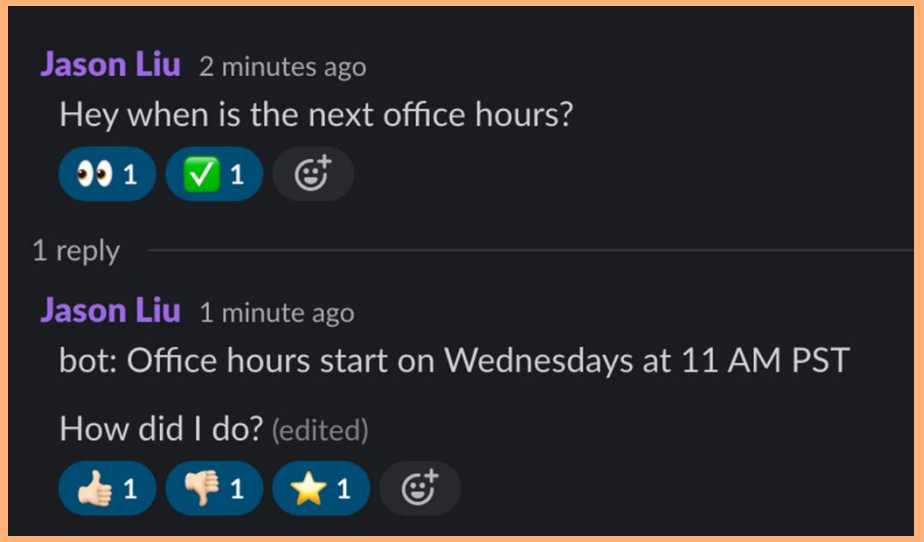
As an interstitial, the **slack integration can react with the eyes emoji** to communicate it has seen the user's message



Use a checkmark to communicate the bot has finished answering



Pre-fill emoji reactions (thumbs-up, thumbs-down, star) to communicate to the user there are alternative ways to provide feedback



This would be saved as an approved question-answer pair which could be used in few-shot examples

@jxnlco

maven.com/applied-llms/rag-playbook

If you see great examples of using interstitials and streaming I would love to see them on Slack in #random

Agenda

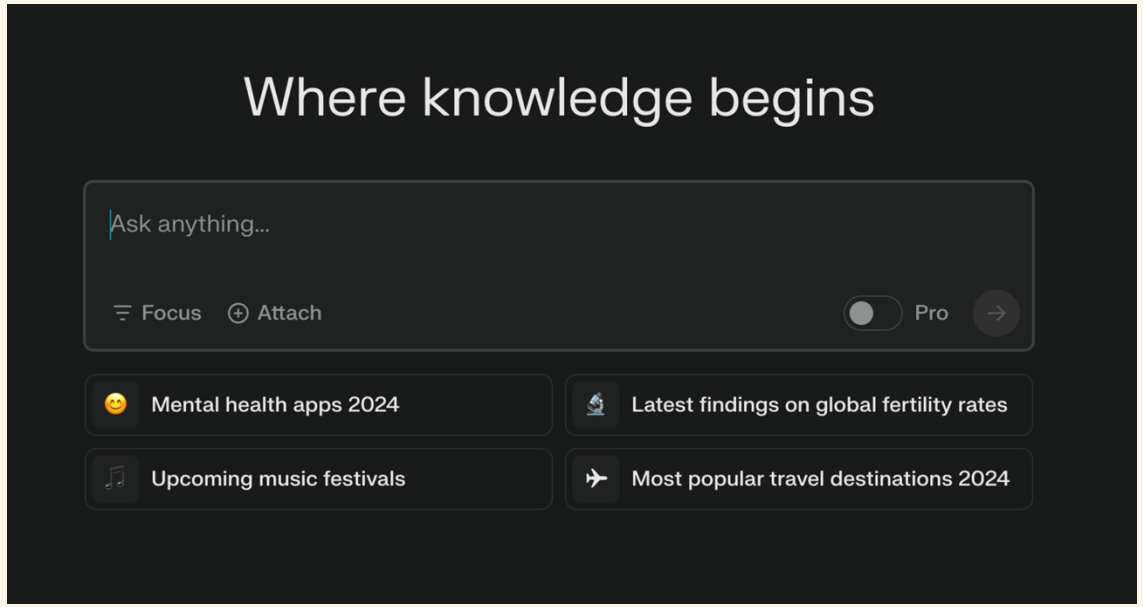
Collecting more feedback

Streaming for better user satisfaction

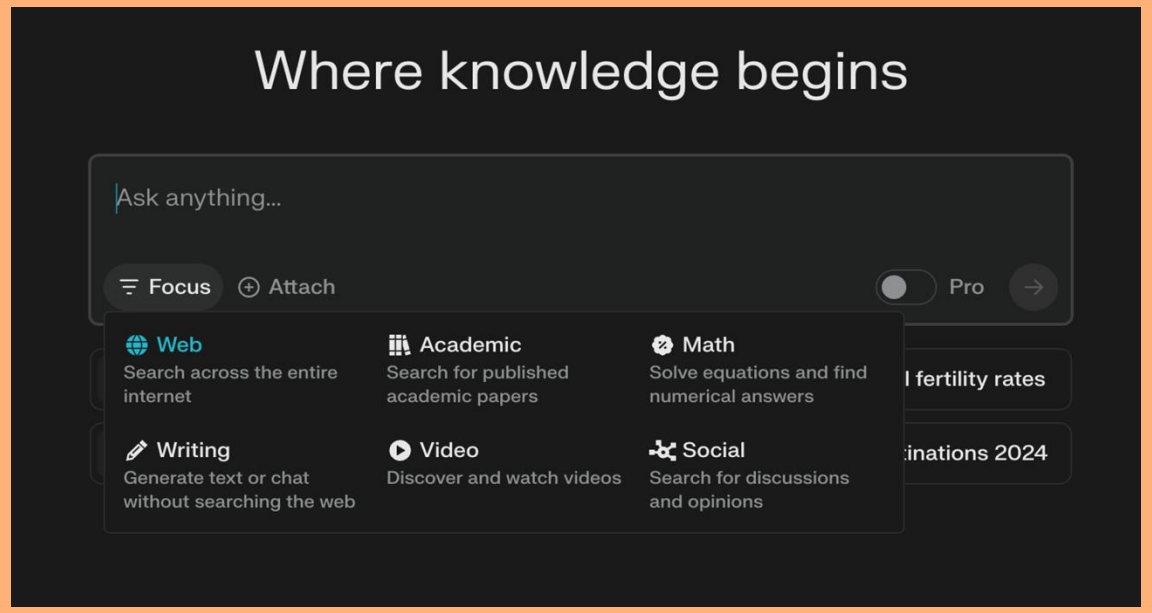
Prompting and Chain of Thought

Prompting: Showcase capabilities

Perplexity is always showing off capaibilites trying to guide the user to behavior we perform well in



Example queries



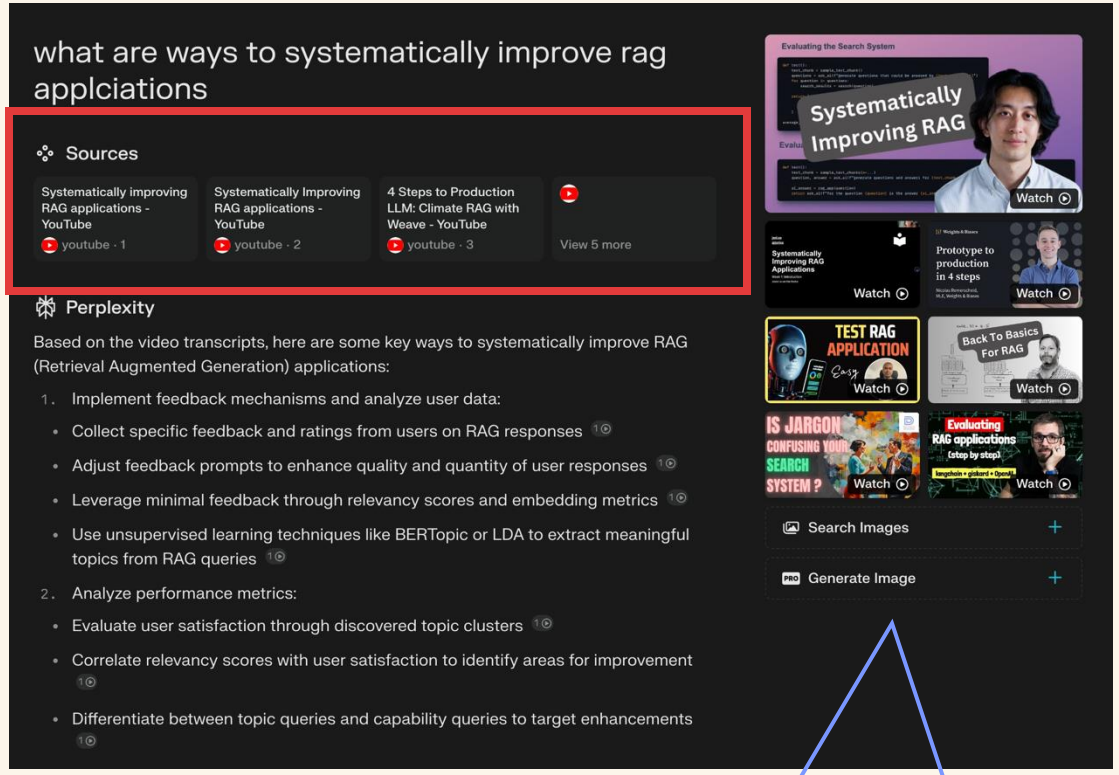
Literally a list of capabilities, which likely have different prompts

In session 4 we'll talk about how to discover them through data analysis

maven.com/applied-llms/rag-playbook

Provide example and related queries

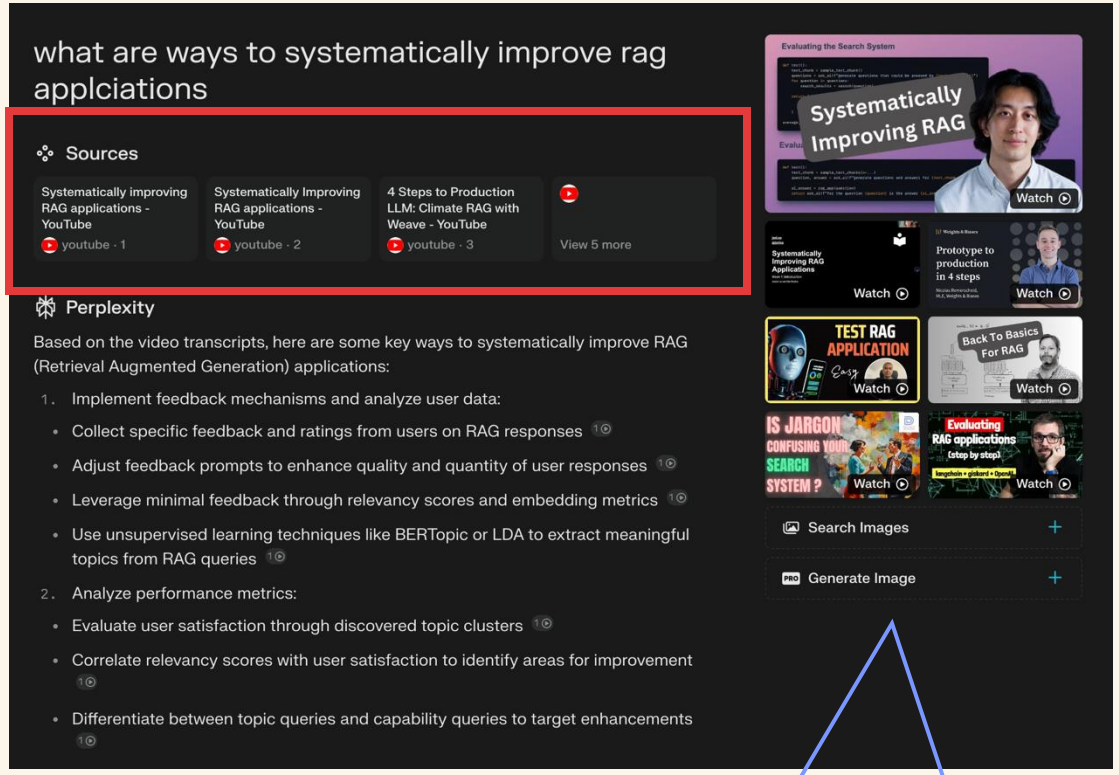
Focus on capabilities available and dynamically change prompting based on specific use case



Special UI elements for sources by type (e.g., video), related queries, etc.

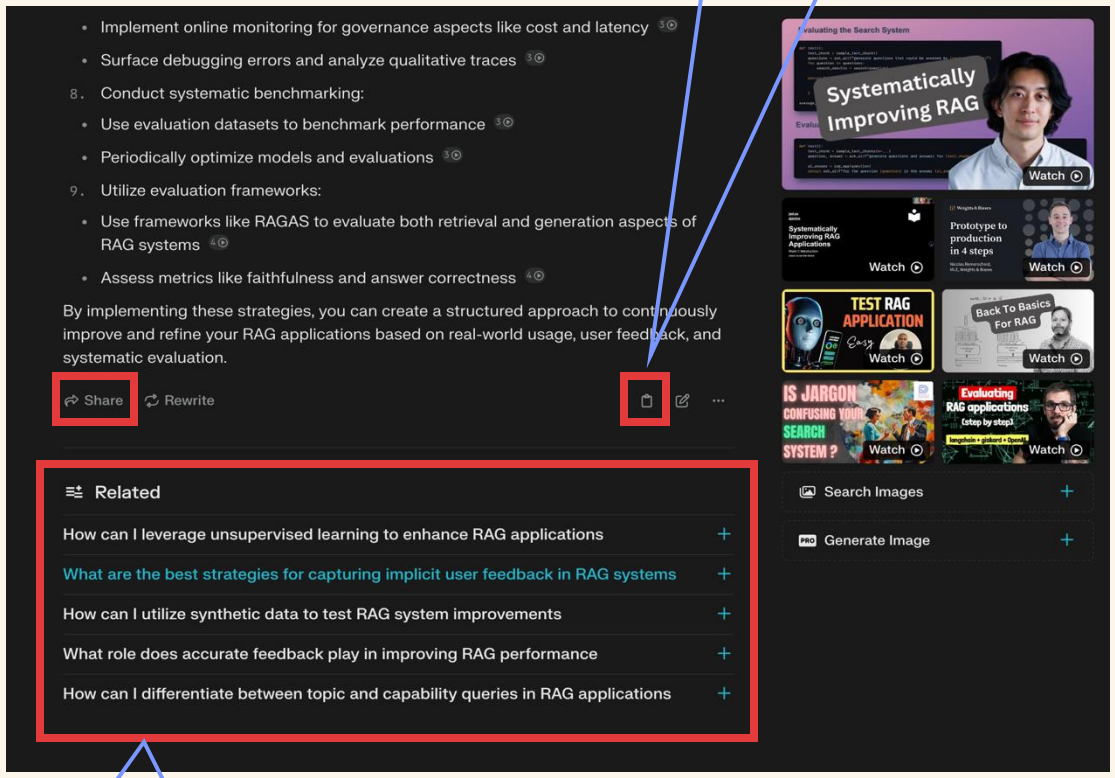
Provide example and related queries

Focus on capabilities available and dynamically change prompting based on specific use case



Special UI elements for sources by type (e.g., video), related queries, etc.

Share and Copy buttons for feedback



Related queries

Once you start looking for interactions that help collect data, you'll see them everywhere

Prompting: Reject Work

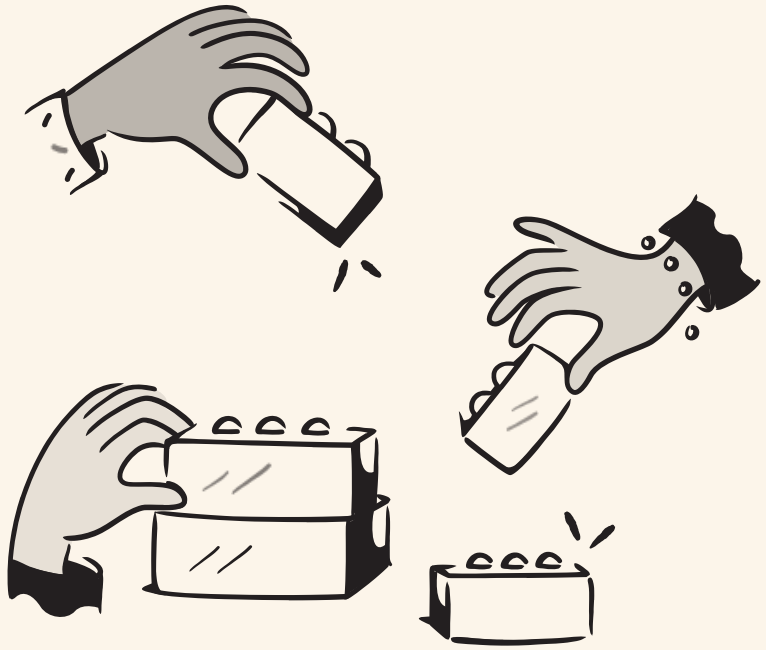
Reject work.

- Just like how we had to perform segmentations, I believe it's crucial to consider ways to enable the language model to reject work
    - **Explore if few shotting works well "Here are examples of similar questions we cannot answer..."**
- Give it permission to say no, but follow up, and set expectations
- If we model the success of this rejection system it's just another precision / recall trade off

Monologues and Chain of Thought

I still see many companies underestimate the power of Chain of Thought

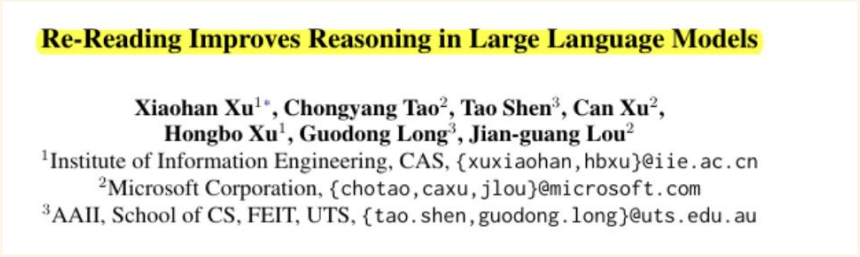
- Chain of Thought is often a 10% bump in performance
- It can often make the difference between something that is usable vs. unusable
- If we wrapped Chain of Thought in either XML or in streaming, we can:
  - Build a dynamic UI that renders the Chain of Thought as separate data
  - Treat the Chain of Thought as some kind of loading interstitial too!



Monologues and Chain of Thought

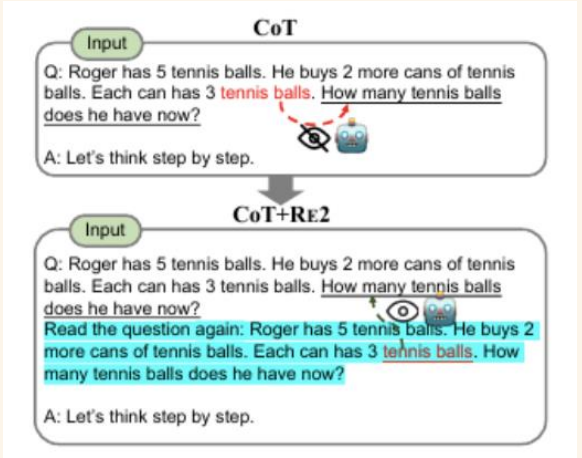
Overview:

- Leverage the monologue for multiple purposes
- When dealing with lengthy contexts, the LLM may struggle with recall or fully process instructions
- Try to prompt the model to reiterate relevant instructions and key text chunks before response generation
  - This is like training an intern who you'd naturally ask them to review and summarize important info
  - Consider including a "re-reading" prompt to improve reasoning



Re-Reading Improves Reasoning in Large Language Models

Xiaohan Xu[1]*, Chongyang Tao[2], Tao Shen[3], Can Xu[2],
Hongbo Xu[1], Guodong Long[3], Jian-guang Lou[2]
[1]Institute of Information Engineering, CAS, {xuxiaohan,hbxu}@iie.ac.cn
[2]Microsoft Corporation, {chotao,caxu,jlou}@microsoft.com
[3]AAII, School of CS, FEIT, UTS, {tao.shen,guodong.long}@uts.edu.au




```python
from pydantic import BaseModel

class Response(BaseModel):
    relevant_instructionrs: str
    relevant_context: str
    chain_of_thought: str
    content: str
```

Case study: Generating Quotes for SaaS Pricing

Overview: We built a capability focused on generating pricing quotes based on a call transcript

- Detect that a pricing quote was being requested for a call
  - LLM Classifier was measured w/ recall
- In the prompt, we included the entire one-hour transcript and the entire pricing document
- By using monologues and Chain of Thought, we asked the LLM to:
  - Reiterate what variables determine the quotes
  - Reiterate relevant parts of the transcript
  - Reiterate parts of the pricing document that were relevant
- As a result, the LLM reasoned what pricing options might be available before generating a response. With a single prompt, we were able to get our pricing questions answered without complex multi-stage reasoning
- This allow us to make sure our follow ups were A+ and we had citations for our sales reps to verify the generated quotes. (We paid them to correct quotes, more data!)

Monologues before responses dramatically improve tonality and quality, which can be fine-tuned later without monologues.

Monologues and Chain of Thought

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

You are an AI assistant tasked with answering queries based on given context.

Before generating a response, you must use <monologue></monologue> tags to reiterate

the relevant instructions and the relevant text chunks involved in answering the query.

Here is the context you will be working with:

<context>

{{CONTEXT}}

</context>

When answering a query, follow these steps:

1. Use <monologue></monologue> tags to:

   a. Reiterate the relevant instructions under <relevant_instructions> tags

   b. Include the relevant parts of the context under <relevant_context> tags

2. After the monologue, generate your response and enclose it in <response></response> tags.

Monologues and Chain of Thought

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

You are an AI assistant tasked with answering queries based on given context.

Before generating a response, you must use <monologue></monologue> tags to reiterate

the relevant instructions and the relevant text chunks involved in answering the query.

Here is the context you will be working with:

<context>

{{CONTEXT}}

</context>

When answering a query, follow these steps:

1. Use <monologue></monologue> tags to:

   a. Reiterate the relevant instructions under <relevant_instructions> tags

   b. Include the relevant parts of the context under <relevant_context> tags

2. After the monologue, generate your response and enclose it in <response></response> tags.

Here is the query you need to answer:

<query>

{{QUERY}}

</query>

Your output should follow this format:

<monologue>

<relevant_instructions>

[Reiterate the relevant instructions here]

</relevant_instructions>

<relevant_context>

[Include the relevant parts of the context here]

</relevant_context>

</monologue>

<response>

[Your answer to the query goes here]

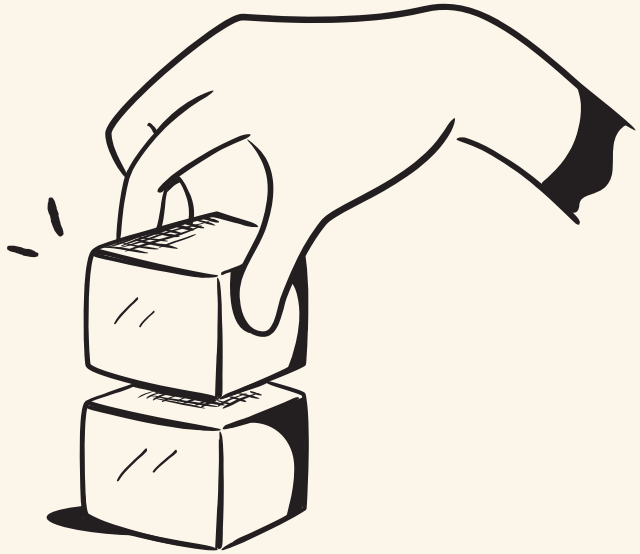
</response>

Remember to always use the monologue tags before generating your response and ensure that your answer is based on the information provided in the context.

A single step is often not enough.

I would consider a validation pattern before going into full-blown multi-stage agents.

As our LLMS get more powerful, we're going to be able to do more and more within a single prompt. What might take an agent now might be possible with a single prompt in the future.



Incorporating validators

- In latency-insensitive applications, incorporating validators can help **increase user trust and satisfaction** in your product
- **Just use evals / tests within the production workflow**
- When we have components like reasoning, citations, and text chunks, we can utilize them in a secondary prompt:
  - Use an external system to evaluate whether the reasoning, citations, and generated response **effectively answer the question**
  - If they don't, the system provides detailed feedback on what's incorrect, unreasonable, or needs revision

- Note!
  - These tests could be language models but it could also be unit tests or calls to external APIs

Incorporating validators: Referencing Content without Hallucinations

Problem:

- We wanted a language model to respond to emails with references to case studies and marketing material.
- However, we wanted to ensure that every single link is from the company namespace and that there were no hallucinations or invalid links since our links included UUIDs in the URL.

```python
from pydantic import BaseModel, field_validator
import re
from typing import List

class Email(BaseModel):
    subject: str
    body: str

    @field_validator('body')
    def validate_urls(cls, v, context):
        allowed_domains = context.get('allow_list', [])
        urls = re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', v)
        for url in urls:
            if not any(allowed_domain in url for allowed_domain in allowed_domains):
                raise ValueError(f"URL {url} is not in the allow list")
        return v
```

Incorporating validators: Referencing Content without Hallucinations

Solution:

- Our validator used a regular expression to find all urls
- Checked domains for our allow list
- We made a GET request to each URL to verify 200 status
- If there were any issues, we would send an error and request regeneration

- We initially had 4% failure rate, after 1 retry, it was 0%, after finetuning gpt-4o, 0% in a single pass

```python
from pydantic import BaseModel, field_validator
import re
from typing import List

class Email(BaseModel):
    subject: str
    body: str

    @field_validator('body')
    def validate_urls(cls, v, context):
        allowed_domains = context.get('allow_list', [])
        urls = re.findall(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', v)
        for url in urls:
            if not any(allowed_domain in url for allowed_domain in allowed_domains):
                raise ValueError(f"URL {url} is not in the allow list")
        return v
```

Even as of Feb 2025, Deep Research will include fake links to example.com/slug-1-2-3 when given the opportunity

Food for thought: try this at work or in your own projects

- [ ] **Work on food for thought from last few sessions**
  - Generate synthetic data to test your system
  - Improve representations for each sub-task. Consider preparing triplet data sets, using Cohere re-rankers, or finetuning an embedding model (with sufficient data)
  - Implement user feedback mechanisms

- [ ] **Questions to ask yourself**
  - Am I being too subtle with collecting feedback on my product.
  - Could building better citations help me gain user trust and satisfaction?
    - Is there any way for me to leverage the citations to collect more relevancy data?
  - Could I implement better streaming, interstitials, and follow-up actions to make my application feel faster?
  - How can I better promote capabilities and reject other work
  - Can I include monologues and chains of thought to reiterate parts of the prompt and improve reasoning in my system.