

Observing an LLM in production

Phillip Carter, Principal PM @ Honeycombio



LLMs in production are hard.

Can't test and debug LLMs like other software.

Prompt engineering is a subtle art and regressions abound.



We learned the hard way earlier this year!

Launched Query Assistant May 2023.
Much fanfare and rejoicing was had
with marketing.

We iterated until July 1st.

We stumbled our way into best
practices, learned a lot, and made Query
Assistant a core product offering for all
users.

Honeycomb Launches First-of-Kind Natural Language Querying for Observability Using Generative AI

By [Harrison Calato](#) | Last modified on May 3, 2023



And we saw some great results.

Higher product retention for new users.

Higher conversion rates to paid tiers.

Sales spent less time teaching prospects.



So how did we do it?

Observability to the max!

We capture *everything* as traces.

Context-gathering operations

- User input
- User/team ID
- Querying dataset name
- Suggested queries for dataset
- Dataset settings
- Request to embedding model
- Schema subset used for LLM call
- etc.

Actual LLM request/response info

- Dynamic prompt building steps
- Full prompt text
- Full LLM response
- Token usage
- Response parse and validate runs
- Any errors for any step of this process
- User feedback response
- etc.



Anatomy of an LLM trace

- Trace covers 4 primary things
 - User clicks “Get Query” button
 - Full RAG pipeline
 - LLM request
 - Response parsing/validation before submitting to querying engine
- 48 spans per trace to represent “all things LLM feature”
 - Many more spans in the rest of the trace!
 - **LLM operations are a part of a larger user experience!**



We Monitor with SLOs

Latency SLO

- Time it takes from when a user clicks hits “Get Query” until the query is submitted to the query engine should complete in less than 10 seconds
- Based on tracing data → covers everything
- **Monitors the end-user experience**, not just OpenAI latency

Error SLO

- A user should (usually) always get a query that executes for them
- Based on the same tracing data
- Any error in the 48 spans counts against this
- **Monitors the whole end-user experience**, not just calling OpenAI correctly



Ship and monitor daily

- Look at the past 24 hours of activity
 - Look to see if anything regressed the SLO
 - Find something we don't like that's failing the SLO
 - Drill down into a bad user experience, all the way to a specific trace
- Take that data into prompt engineering efforts and improve the problem
 - Use this data as the basis for an evaluation, if needed
- Release that day
- **Repeat the cycle - set up an engineering flywheel!**



Walkthrough

[proj-ml] Natural-language query generation availability

[Configure Burn Alerts](#)
[Delete](#)
[Edit SLO](#)

 Craig Atkinson created Apr 26, 2023

Proportion of requests to generate natural language query that complete without an error. Currently doesn't differentiate between client (e.g. rate limited) and server errors.

80% of eligible events from the  **poodle** column

`sli.nlq_availability` will succeed over a period of 7 days.

Exhaustion time

Status

4 hours

● Normal

Budget Burndown

[Reset Budget](#)

How much of the error budget remains after the last 7 days. Starts at 100% and burns down.



Historical SLO Compliance

For each day of the past 7, how often this SLI has succeeded over the preceding 7 days.



Distribution of Events

failing SLI by

fit

duration_ms

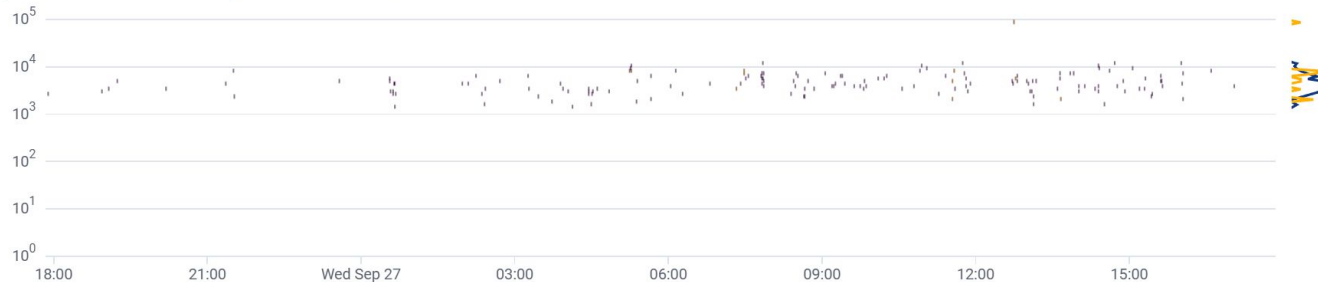


Make Default

Last 24 hours



Sep 26 2023 17:50:50.375 – Sep 27 2023 17:50:50.375 UTC-07:00



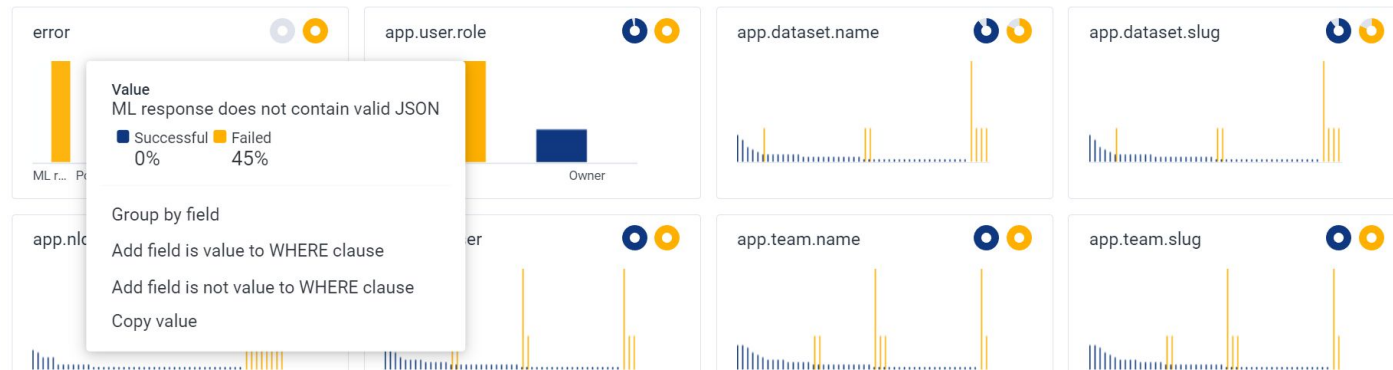
Search BubbleUp results

Clear


Dimensions

Try to `{-} GROUP BY` columns that look most different between the ■ successful and ■ failed.

Showing 1-12 of 156




VISUALIZE	WHERE	GROUP BY
COUNT	name = GenerateQueryFromPrompt sli.nlg_availability = false app.nlg.user_input exists	app.nlg.user_input app.nlg.response
ORDER BY	LIMIT	HAVING
COUNT desc	None	None; include all results

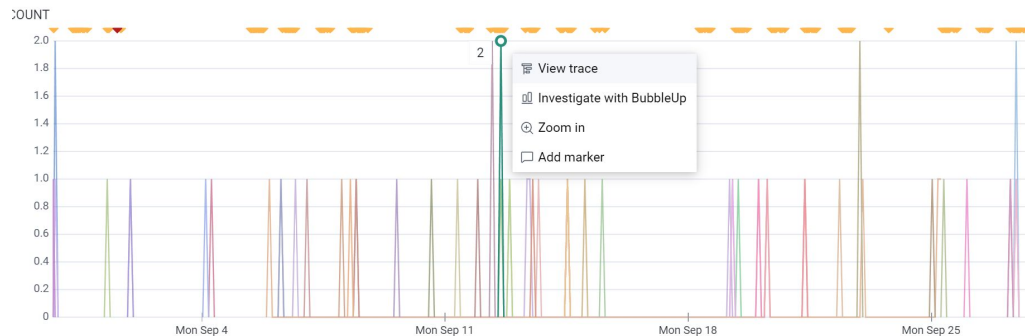
Query Assistant 

Run Query

Query Results

Last 28 days (run 1 min ago)  

Wednesday Aug 30 17:42:27 – Wednesday Sep 27 17:42:27 UTC-07:00 (Granularity: 2 hour)

Overview **BubbleUp** Correlations Traces Events

app.nlg.user_input

app.nlg.response

latency distribution by status code

what are my errors?

SELECT * FROM traces WHERE api_endpoint = '/your-api-endpoint'

number of requests where http.route contains /v3/gateway or /v3/personas or /v3/sensor or /v3/traffic but do NOT contain the words Full or heartbeat

breakdowns: [{"http.route", "http.method"}, {"calculations": [{"op": "COUNT"}]}, {"filters": [{"column": "http.route", "op": "con"}]}



© 2023 Hound Technology, Inc. All Rights Reserved.

Fields

response

`str` app.nlq.response

...

```
{
  "breakdowns": [
    "trace.trace_id",
    "trace.link.trace_id",
    "customer.analytics_id",
    "cook.distance_from_customer_meters",
    "exception.stacktrace",
    "customer.external_id",
    "customer.email",
    "customer.location",
    "customer.loyalty_balance",
    "customer.advocate_referral_code",
    "trace.span_id",
    "cart.start_new_cart",
    "http.url",
    "customer.address_external_id",
    "http.method",
    "rpc.method",
    "cart.credit_applied_cents",
    "trace.parent_id",
    "cart.external_id",
    "http.route",
    "user_agent.original",
    "response.body",
    "cart.cart_id",
    "payment.ExternalId",
    "request.body",
    "http.response_content_length",
    "http.target",
    "firebase.email",
    "cook.location",
    "search.query",
    "query.limit",
    "order.external_"
  ]
}
```



Insight – JSON response too big!

- Our limit on response size is 150 tokens for OpenAI requests
- This user interaction led to an incomplete output truncated after 150 tokens
- This request has all we need to improve via prompt engineering
 - Full requests context
 - Full prompt text sent to OpenAI
 - Known-bad behavior
 - Understanding what good behavior means for us

You can do this today!

It's all powered by OpenTelemetry

- Pick your language, install your SDK
- Add autoinstrumentation to make sure your **whole system** is instrumented
- Add custom spans for relevant LLM operations
 - RAG pipeline steps
 - Any context gathering you do
 - A request to an LLM
 - Output parsing and validation, etc.
- **An engineer can add this instrumentation in a day.**



Expect this to get easier, too!

- Autoinstrumentation packages for modern AI components
 - LLM providers
 - Frameworks like LangChain, LLamaIndex, etc.
 - Vector database calls
 - Etc.
- Standards for data via OpenTelemetry Semantic Conventions proposed
- Observability tools and products morphing to adopt the LLM Observability use case
 - Honeycomb, DataDog already working on this





Questions?



www.honeycomb.io