UC San Diego

CSE/DSC 234
Data-Centric AI and AI Engineering

Arun Kumar

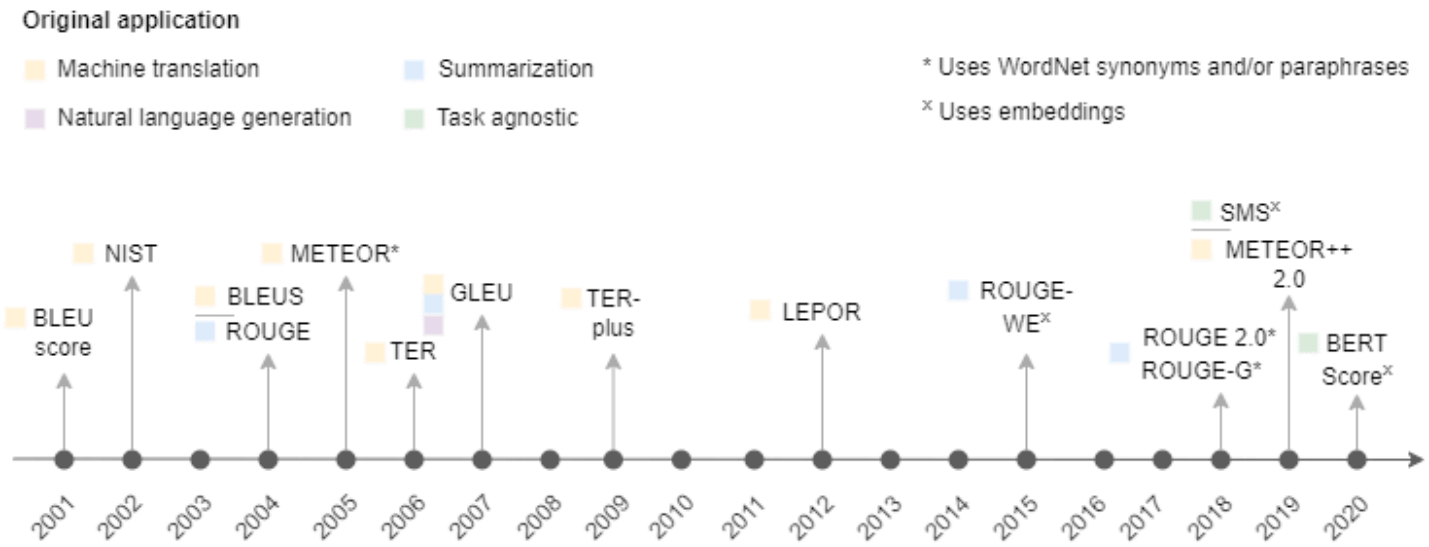
Topic 1: Creating Data and Evals

Outline

- Motivation for Data-Centric AI
- Creating Data
- Creating Evals

Evals: What and Why?

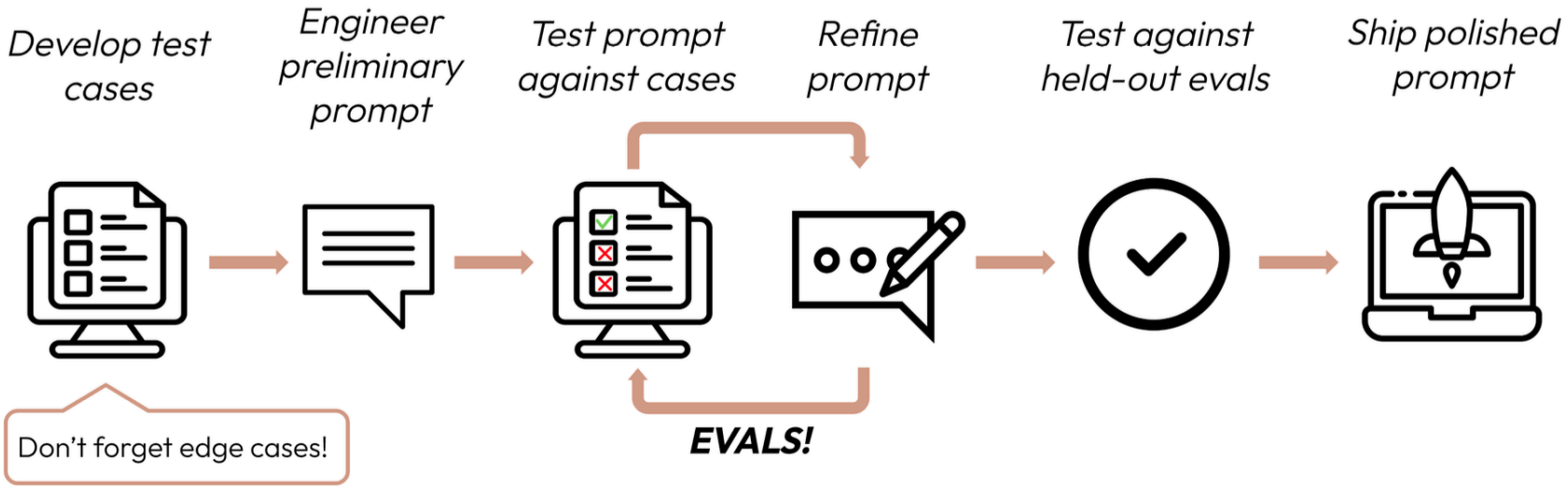
- Traditional AI evaluation came with precise metrics for narrow, well-defined output types/semantics:
  - Classification: F1, precision, recall, AuROC, etc.
  - Regression: MAE, RMSE, etc.
  - Translation/summarization: BLEU, ROUGE, perplexity, etc.



Evals: What and Why?

- But modern LLM applications often produce open-ended, subjective, multi-dimensional outputs
  - Subject Matter Expert (SME) human evaluation is golden but often expensive, slow, inconsistent (low IAA), does not scale
  - Need for *automated evaluation* (evals) that can approximate SME human judgment
- *Creating* eval metrics and data is now a key part of AI engineering
- **NB:** Foundation model benchmarks (MMLU, GSM8K, etc.) are different from *application-specific* evals; doing well on former does not necessarily port to latter!

Evals: What and Why?



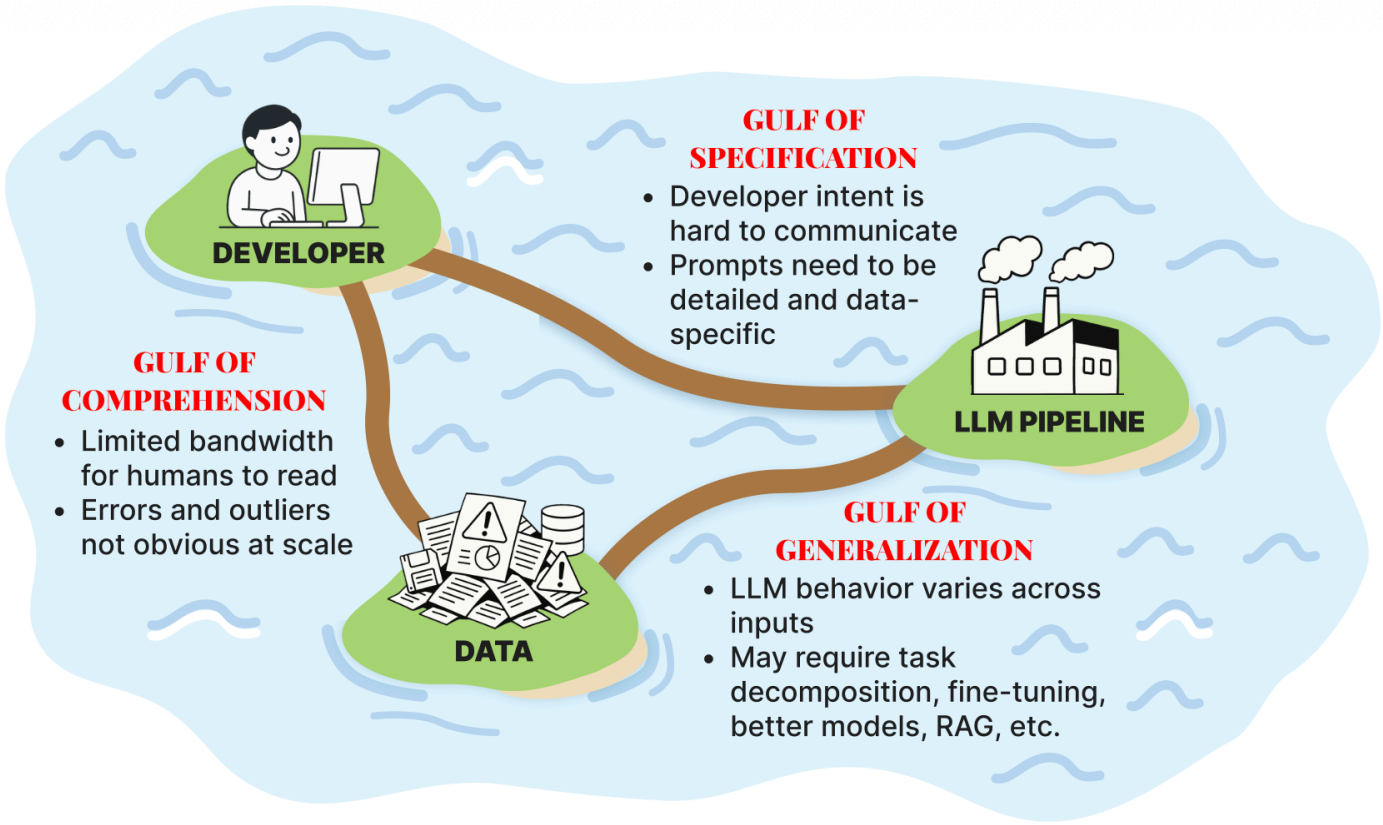
https://platform.claude.com/docs/en/test-and-evaluate/develop-tests

Eval Metrics: Defining “Quality”

- No universal taxonomy for LLM appl. eval metrics:
  - Correctness, completeness, faithfulness, helpfulness, safety, instruction-following, style/tone, etc.
- At a high level, 2 types of eval metrics:
  - **Reference-based:** Compare output to known *correct* ground truth answer, e.g., exact match, SQL execution, code tests
  - **Reference-free:** Evaluate *inherent properties* of output, e.g., No speculation? Follows format? Appropriate tone?
- Reference-based preferred when feasible as it is more objective, but reference-free inevitable when ground truth is hard to create

Credit: “AI Evals For Engineers & PMs” by Hamel Husain and Shreya Shankar
https://maven.com/parlance-labs/evals

Key Concept: “Three Gulfs” in Evals



- Evals is how an AI application can bridge these gulfs

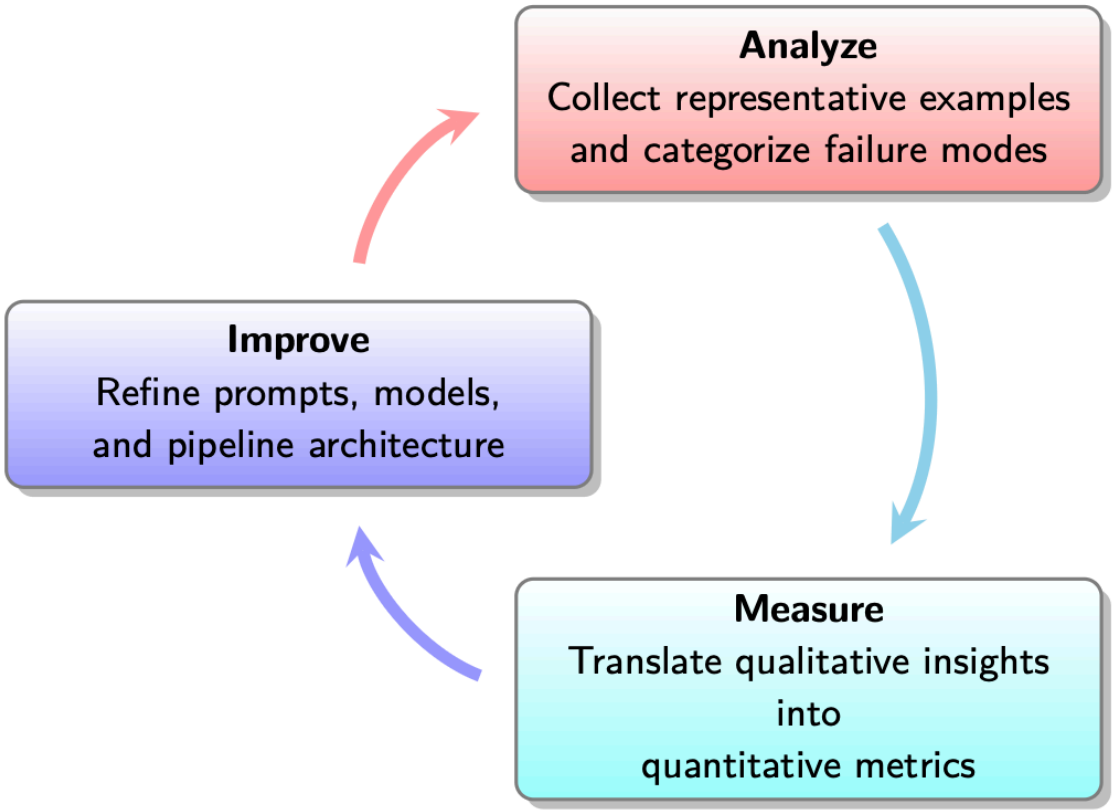
Credit: “AI Evals For Engineers & PMs” by Hamel Husain and Shreya Shankar
https://maven.com/parlance-labs/evals

Evals Schemes and Tradeoffs

	Vibe Checks	Codified Evals (Rules, KBs, SLMs, etc.)	LLM-as-Judge	SME Annotations
Upfront Cost	Very Low	High	Medium	Very High
Recurring Cost	Low	Low-Medium	Medium-High	Very High
Scalability	Low	Very High	Medium-High	Very Low
Reliability	Low	High	Medium-High	Very High
Flexibility	Very High	Medium	High	Very High
Prevalence	Very High	Medium	Very High	Low-Medium

Key Concept: Iterative Cycle of Evals

- Evals in practice is an iterative refinement process, not one-time!



Credit: “AI Evals For Engineers & PMs” by Hamel Husain and Shreya Shankar
https://maven.com/parlance-labs/evals

Outline

- Motivation for Data-Centric AI
- Creating Data
- Creating Evals
  - Analyze
  - Measure
  - Improve

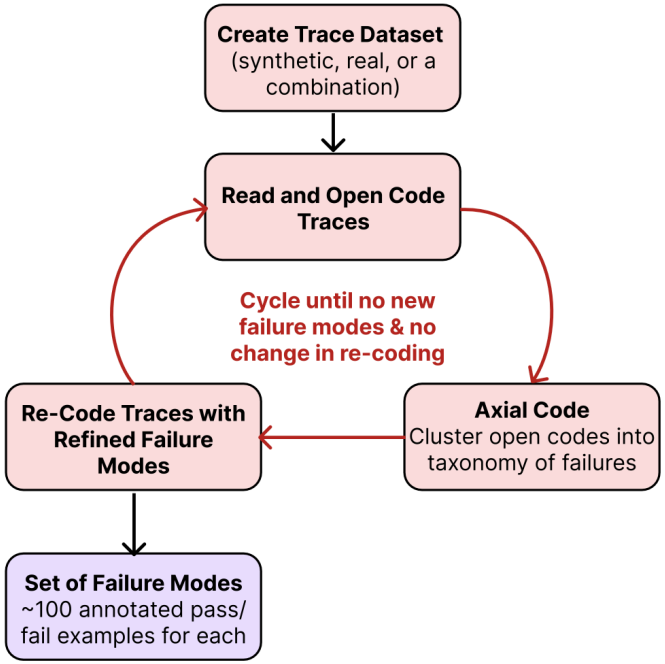
Peer Instruction Activity

(Switch to Google Form)

Error Analysis: From Traces to Taxonomy

- **Trace:** A complete record of all steps for one query, viz., all LLM turns, tool calls, data retrievals, outputs, etc.
- **Error Analysis:** A methodology from grounded theory (used in social sciences)

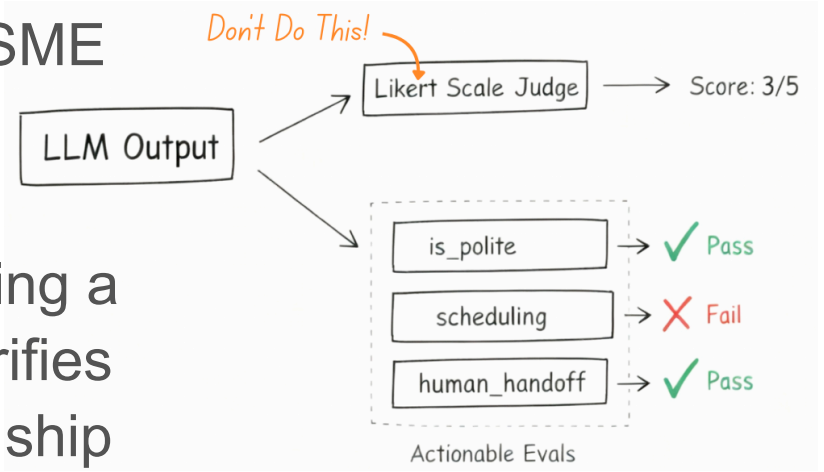
1. **Collect Traces**: gather a diverse sample of 100+ traces from production or your own real/synthetic usage.
2. **Annotate Traces** "aka Open Coding": review each trace and write brief, unstructured notes on the problem (e.g., 'hallucinated a fact', 'misread the user's name', 'failed to use the calculator tool').
3. **Group & Categorize**: Group similar notes into clusters. (ex: tone violation, failed tool call).
4. **Prioritize**: Count the frequency of each category, which informs your priority.



Credit: "Evals Flashcards" by Hamel Husain
https://hamel.dev/notes/llm/evals/flashcards/

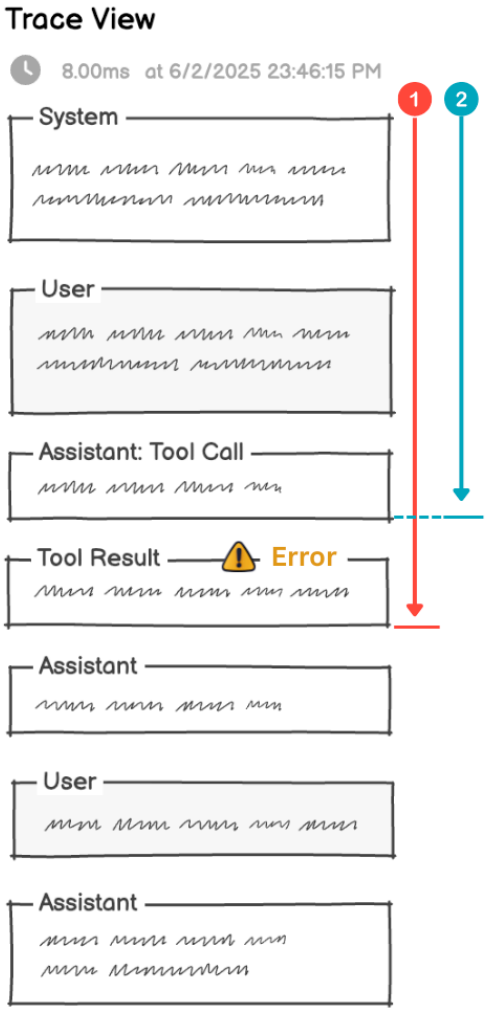
Error Analysis: From Traces to Taxonomy

- Typical desired output: structured taxonomy of failures with ~100 annotated examples
- Quantifying failures:
  - Likert/numeric scales (e.g., 1-5) create ambiguity; hard to align SME judgments
  - Binary decision (Pass/Fail) recommended in practice; “picking a side, even when borderline” clarifies criteria faster; aligns better with ship vs. don’t ship decisions



Credit: “Evals Flashcards” by Hamel Husain
https://hamel.dev/notes/llm/evals/flashcards/

Example: Traces



1. Start with **error analysis**. When reviewing traces, stop at the first (most upstream) error you find. This keeps things tractable (upstream failures tend to be more important). 1

2. Gather traces related to your top failures. For each of these traces, **minimally reproduce the failure in as few turns with as least amount of complexity possible.**

3. N-1: Collect a dataset of minimally reproduced error traces and use the N-1 turns before the error as a test cases. This assumes your system isn't changing rapidly. 2

4. Advanced: For additional test coverage, you can modify traces from #3 in valid ways (rephrasing the user question, etc) with a LLM.

5. Advanced: Simulate a user with another LLM. Doing this well can be challenging.

Credit: "Evals Flashcards" by Hamel Husain
https://hamel.dev/notes/llm/evals/flashcards/

Bootstrapping Evals Dataset

- 3 main sources:
  - **SME-produced:** Traces hand crafted by SMEs
  - **Real data:** Traces from live deployment on alpha testers
  - **(Semi-)Synthetic:** Traces generated using LLMs, alongside code and HIL process
- SME-produced is not scalable; real data unavailable upfront
- Synthesis with SME-produced seed data can bridge the gap

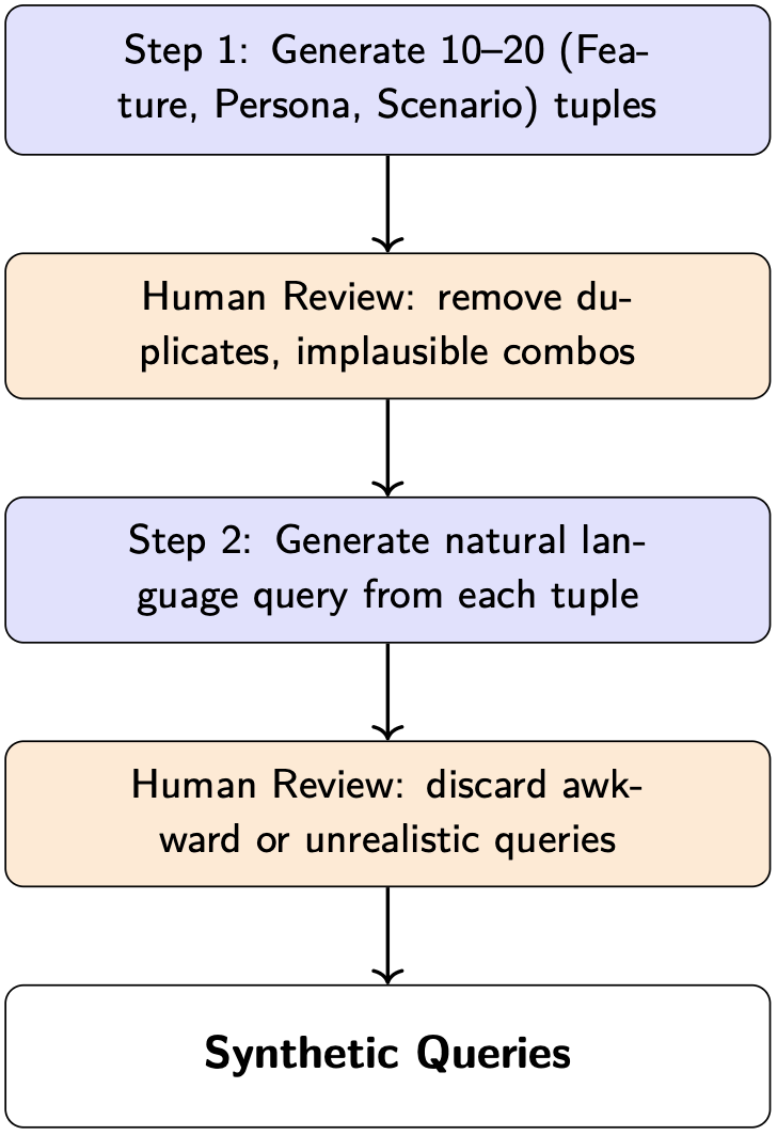
Credit: “AI Evals For Engineers & PMs” by Hamel Husain and Shreya Shankar
https://maven.com/parlance-labs/evals

Bootstrapping Evals Dataset with Synthesis

- **Dimensions:** Mostly independent axes of variation in appl. user queries; useful to ensure *diversity* in synthetic data
- 3 main dimensions:
  - **Feature (the what):** What task the user wants to perform, e.g., property search, scheduling
  - **Persona (the who):** The type of client being served, e.g., first-time buyer, investor
  - **Scenario (the how):** How clearly the user expresses their intent, e.g., well-specified, ambiguous

Credit: “AI Evals For Engineers & PMs” by Hamel Husain and Shreya Shankar
https://maven.com/parlance-labs/evals

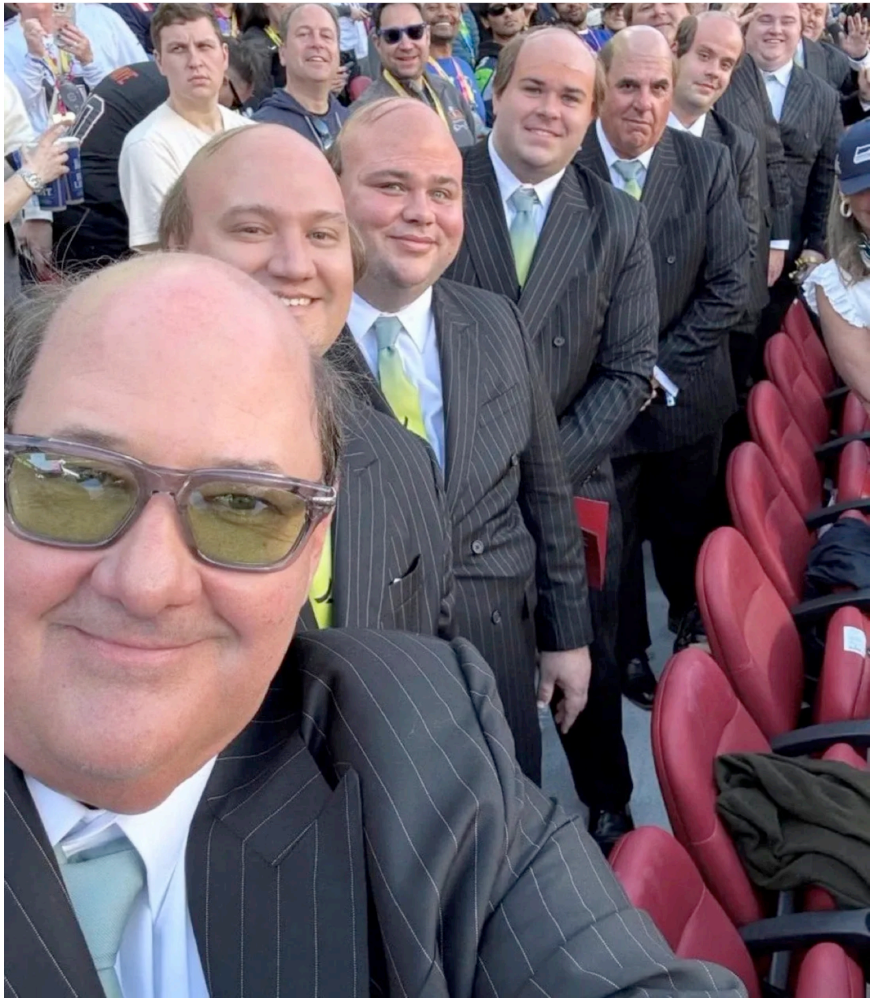
Bootstrapping Evals Dataset with Synthesis



- Do NOT just prompt “give me user queries”—too generic to be useful
- Systematically vary across failure-prone dimensions
- After analysis: fix specification failures first (prompt clarity), then build evaluators for generalization failures

Bootstrapping Evals Dataset with Synthesis

Synthetic eval dataset from prompting
'generate 100 diverse user queries'



Example: Synthesis

Sample Prompt

Generate 10 random combinations of (feature, client persona, scenario) for a real estate CRM assistant.
The dimensions are:
Feature: what task the agent wants to perform. Possible values: property search, market analysis, scheduling, email drafting.
Client persona: the type of client the agent is working with. Possible values: first-time homebuyer, investor, luxury buyer.
Scenario: how well-formed or challenging the query is. Possible values:

- exact match (clearly specified and feasible),
- ambiguous request (unclear or underspecified),
- shouldn't be handled (invalid or out-of-scope).

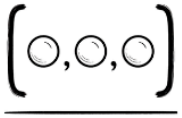
Output each tuple in the format: (feature, client persona, scenario)
Avoid duplicates. Vary values across dimensions. The goal is to create a diverse set of queries for our assistant.

Example: Synthesis

Sample Prompt

```
We are generating synthetic user queries for a real estate CRM
assistant.  The assistant helps agents manage client requests
by searching listings, analyzing markets, drafting emails, and
reading calendars.
Given:
Feature:  Scheduling Client Persona:  First-time homebuyer
Scenario:  Ambiguous request
Write a realistic query that an agent might enter into the
system to fulfill this client's request.  The query should
reflect the client's needs and the ambiguity of the scenario.
Example:
"Find showings for affordable homes with short notice
availability."
Now generate a new query.
```

Bootstrapping Evals Dataset with Synthesis



Use structured input for diversity. Define key dimensions (e.g., Feature, Persona, Scenario) and use them as variables in your prompt.



When possible, **seed your generation with real logs or traces**. Then, ask the model to explicitly inject changes, like a new constraint or a modified variable, to create realistic edge cases.



Enforce output structure & filter. Define a schema for the output. Generate many candidates, then filter to retain the highest-quality, challenging examples.



Increase complexity iteratively. Start with simple queries and incrementally ask the LLM to add constraints, complex formatting, etc.



Do not prompt with zero-shot requests lacking structured input. Example: "Generate 50 test cases". These tend to yield generic, repetitive inputs.

Example: Open Coding of Traces

User Query	Trace Steps (Summary)	First-Pass Annotation
Find 3-bedroom homes under $600k near downtown that allow pets.	▪ Generate SQL query (missing pet constraint) ▪ Return 2 listings ▪ Draft email to client	Missing constraint: pet-friendly requirement ignored.
Set up showings for these two homes this weekend.	▪ Parse dates (Saturday, Sunday) ▪ Calendar API shows agent unavailable Saturday ▪ Propose showings both days	Invalid action: proposed unavailable times.
Send a property list to my investor client in San Mateo.	▪ Search for high ROI properties ▪ Draft casual email about starter homes	Persona mismatch: wrong tone and property type.

Common Pitfalls in Practice

- Failing to test on representative data
- Skipping open coding on traces
- False comfort of generic metrics
- Likert/numeric scales of failure annotation instead of binary
- Treating evals as one-time step instead of iterative refinement

Credit: “AI Evals For Engineers & PMs” by Hamel Husain and Shreya Shankar
https://maven.com/parlance-labs/evals

Outline

- Motivation for Data-Centric AI
- Creating Data
- Creating Evals
  - Analyze
  - Measure
  - Improve

The Trap of Generic Eval Metrics

- Generic eval metrics such as “coherence” or “faithfulness” are OK for starters but rarely suffice for production LLM applications

Your AI Evals Dashboard



Application-Specific Eval Metrics

- Create application-specific eval metrics that are *actionable*, i.e., help fix the pipeline and/or source data to improve reliability



Credit: "Evals Flashcards" by Hamel Husain
https://hamel.dev/notes/llm/evals/flashcards/

Application-Specific Eval Metrics

- "Hamel's Checklist" for high-quality eval metrics:

- [ ] Measure an **error you've observed.**
- [ ] Relates to a **non-trivial issue** you will iterate on.
- [ ] Are scoped to a **specific failure**.
- [ ] Has a **binary outcome** (not a 1-5 score).
- [ ] Is **verifiable** (i.e. human labels for LLM-as-a-Judge)?

*This checklist helps you obtain positive ROI from evals (since they aren't free).

Credit: "Evals Flashcards" by Hamel Husain
https://hamel.dev/notes/llm/evals/flashcards/

Codified Eval Metrics

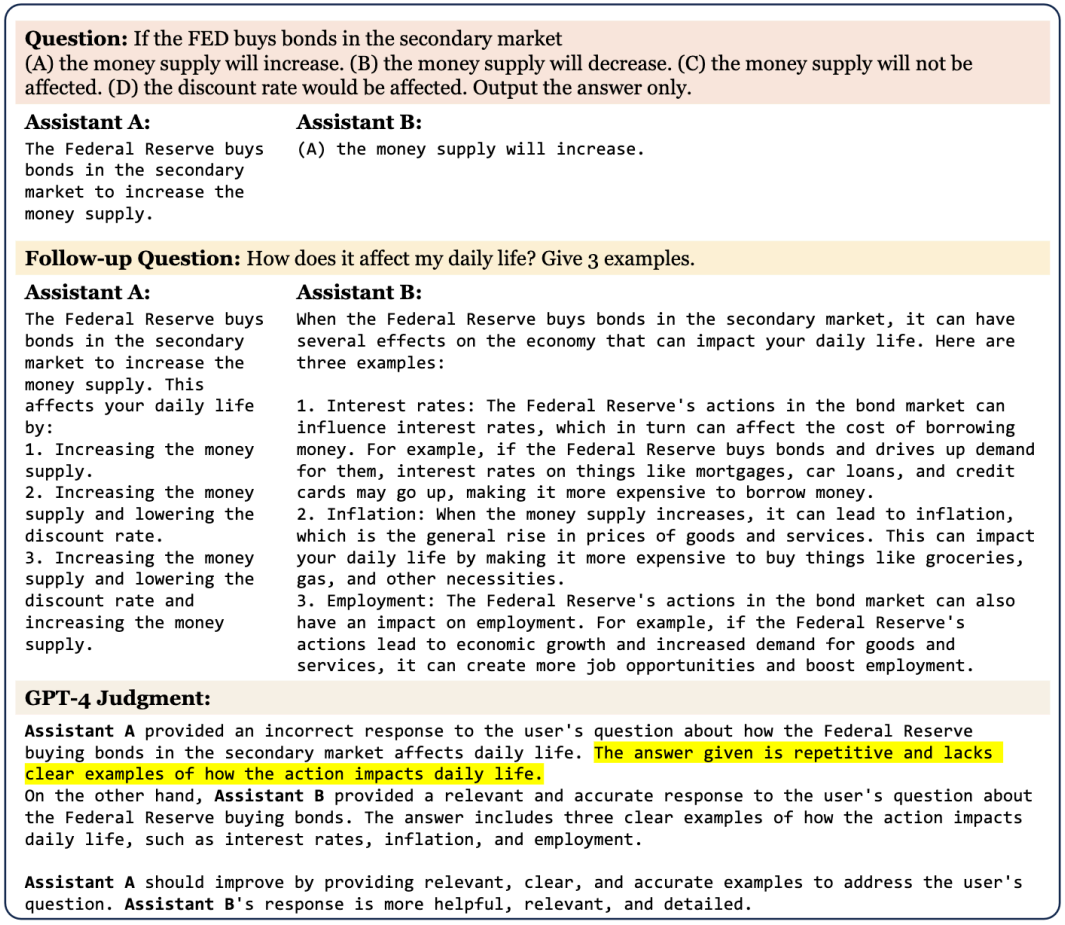
- Bespoke code and/or SLMs to define quality in an application-specific manner
- Increasingly common in practice; difficult but highly valuable
- **Examples:**
  - Parser to check JSON schema validity or SQL syntax
  - Regex or string matching to catch required/forbidden phrases
  - Checking structural properties, e.g., counting bullets
  - Semantic SLM classifiers, e.g., is the tone professional?
  - Tool use to execute some calculations, code, etc.

Pivot Point: LLM-as-Judge

- Very popular evals scheme in practice
- Use (more powerful) judge LLM, possibly aligned with golden human preference data, to judge outputs of application's LLM
- **MT-bench:** A multi-turn question set with 80 high-quality examples
  - 8 common categories of user prompts: writing, roleplay, extraction, reasoning, math, coding, knowledge I (STEM), and knowledge II (HSS)
- **Chatbot Arena:** A crowdsourced "battle" platform for humans to vote on outputs of anon. pairs of LLMs
  - GPT-4 had >80% agreement with humans; same as human-human!

Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. NeurIPS 2023. https://arxiv.org/pdf/2306.05685

Pivot Point: LLM-as-Judge



Question: If the FED buys bonds in the secondary market
(A) the money supply will increase. (B) the money supply will decrease. (C) the money supply will not be affected. (D) the discount rate would be affected. Output the answer only.

Assistant A:

```
The Federal Reserve buys
bonds in the secondary
market to increase the
money supply.
```

Assistant B:

```
(A) the money supply will increase.
```

Follow-up Question: How does it affect my daily life? Give 3 examples.

Assistant A:

```
The Federal Reserve buys
bonds in the secondary
market to increase the
money supply. This
affects your daily life
by:
1. Increasing the money
supply.
2. Increasing the money
supply and lowering the
discount rate.
3. Increasing the money
supply and lowering the
discount rate and
increasing the money
supply.
```

Assistant B:

```
When the Federal Reserve buys bonds in the secondary market, it can have
several effects on the economy that can impact your daily life. Here are
three examples:

1. Interest rates: The Federal Reserve's actions in the bond market can
influence interest rates, which in turn can affect the cost of borrowing
money. For example, if the Federal Reserve buys bonds and drives up demand
for them, interest rates on things like mortgages, car loans, and credit
cards may go up, making it more expensive to borrow money.
2. Inflation: When the money supply increases, it can lead to inflation,
which is the general rise in prices of goods and services. This can impact
your daily life by making it more expensive to buy things like groceries,
gas, and other necessities.
3. Employment: The Federal Reserve's actions in the bond market can also
have an impact on employment. For example, if the Federal Reserve's
actions lead to economic growth and increased demand for goods and
services, it can create more job opportunities and boost employment.
```

GPT-4 Judgment:

Assistant A provided an incorrect response to the user's question about how the Federal Reserve buying bonds in the secondary market affects daily life. The answer given is repetitive and lacks clear examples of how the action impacts daily life.
On the other hand, **Assistant B** provided a relevant and accurate response to the user's question about the Federal Reserve buying bonds. The answer includes three clear examples of how the action impacts daily life, such as interest rates, inflation, and employment.

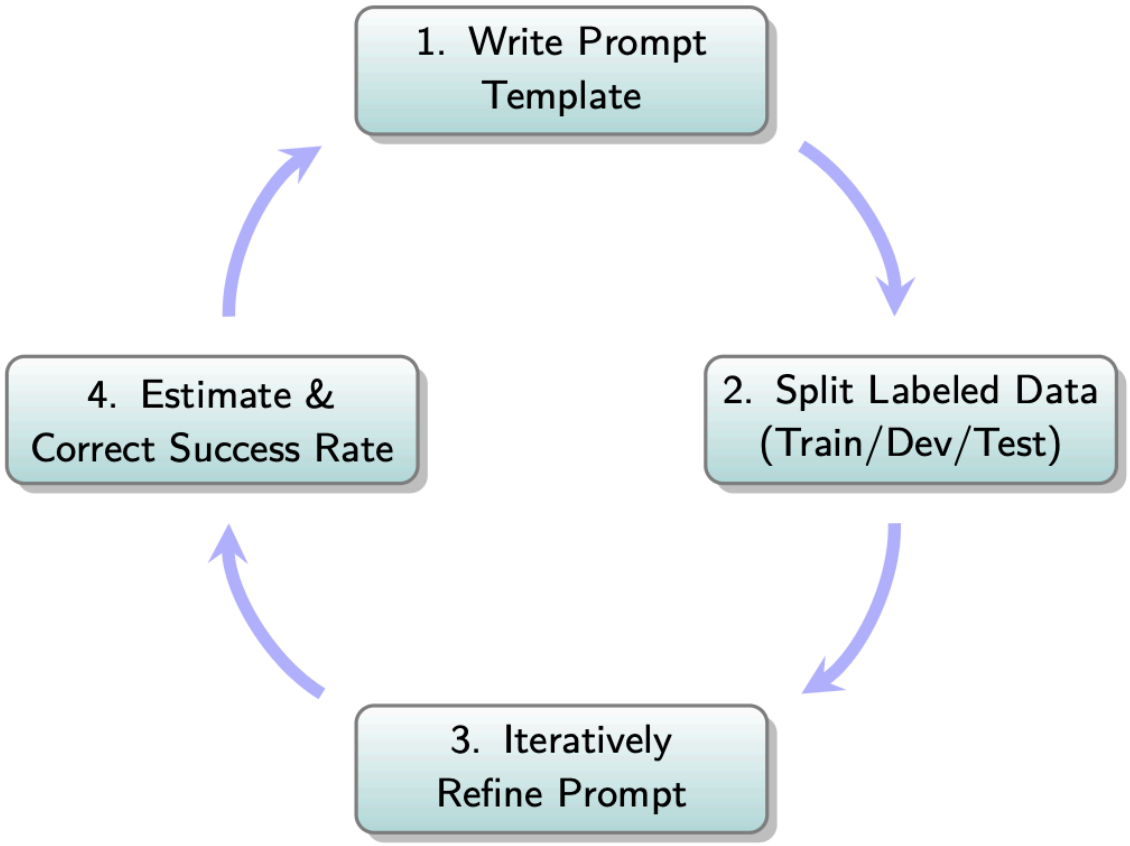
Assistant A should improve by providing relevant, clear, and accurate examples to address the user's question. **Assistant B**'s response is more helpful, relevant, and detailed.

Figure 1: Multi-turn dialogues between a user and two AI assistants—LLaMA-13B (Assistant A) and Vicuna-13B (Assistant B)—initiated by a question from the MMLU benchmark and a follow-up instruction. GPT-4 is then presented with the context to determine which assistant answers better.

Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. NeurIPS 2023.
https://arxiv.org/pdf/2306.05685

LLM-as-Judge Eval Metrics

- Typical workflow for creating LLM-as-judge eval metrics:



LLM-as-Judge Eval Metrics

- The judge prompt matters enormously! Key elements:
  - **Task description** + clear Pass/Fail definitions
  - **Few-shot examples** from a training set with both Pass/Fail
  - **Chain-of-thought:** “Explain reasoning, then give verdict”
  - **Structured output**: JSON with “reasoning”, “answer” fields
- **Caveats:**
  - Without rubric, LLM judges tend to default to 60%-70% score
  - Binary Pass/Fail is more reliable than Likert/numeric scales

Example: LLM-as-Judge Prompt

Sample Prompt

You are an expert evaluator assessing outputs from a real estate assistant chatbot.
Your Task: Determine if the assistant-generated email to a client uses a tone appropriate for the specified client persona.
Evaluation Criterion: Tone Appropriateness
Definition of Pass/Fail:

- Fail: The email's tone, language, or level of formality is inconsistent with or unsuitable for the described client persona.
- Pass: The email's tone, language, and formality align well with the client persona's expectations.

Client Personas Overview:

- Luxury Buyers: Expect polished, highly professional, and deferential language. Avoid slang or excessive casualness.
- First-Time Homebuyers: Benefit from a friendly, reassuring, and patient tone. Avoid overly complex jargon.
- Investors: Prefer concise, data-driven, and direct communication. Avoid effusiveness.

Output Format: Return your evaluation as a JSON object with two keys:

1. reasoning: A brief explanation (1-2 sentences) for your decision.
2. answer: Either "Pass" or "Fail".

Examples:

Input 1:
Client Persona: Luxury Buyer
Generated Email: "Hey there! Got some truly awesome listings for you in the high-end district. Super views, totally posh. Wanna check 'em out ASAP?"
Evaluation 1: "reasoning": "The email uses excessive slang ('Hey there', 'awesome', 'totally posh', 'ASAP') and an overly casual tone, which is unsuitable for a Luxury Buyer persona.", "answer": "Fail"

Input 2:
Client Persona: First-Time Homebuyer
Generated Email: "Good morning! I've found a few properties that seem like a great fit for getting started in the market, keeping your budget in mind. They offer good value and are in nice, welcoming neighborhoods. Would you be interested in learning more or perhaps scheduling a visit?"
Evaluation 2: "reasoning": "The email adopts a friendly, reassuring tone ('great fit for getting started', 'nice, welcoming neighborhoods') suitable for a First-Time Homebuyer, and clearly offers next steps.", "answer": "Pass"

Now, evaluate the following:
Client Persona: {{CLIENT_PERSONA_HERE}}
Generated Email: {{GENERATED_EMAIL_HERE}}
Your JSON Evaluation:

LLM-as-Judge: Labeled Data Splits

- **Intuition:** Treat LLM-as-Judge prompt development akin to regular ML classifier training
- **Key Measures: TPR** (fraction of real Passes correctly judged Pass) and **TNR** (fraction of real Fails correctly judged Fail)
- Typical recommended splits:
  - **"Training"** (10-20%): Pool for few-shot examples
  - **Dev/Validation** (40-45%): Refine prompt, measure TPR/TNR after each edit
  - **Test** (40-45%): Held-out, final unbiased accuracy estimate
  - NEVER put dev or test examples in the prompt (leakage)!

LLM-as-Judge: Typical Biases

- **Position bias:** Tend to prefer first (or last) response
  - This alone can flip model rankings on benchmarks!
- **Verbosity bias:** Often, longer response = higher score
- **Self-enhancement:** LLMs tend to rate their own outputs higher
- **Limited reasoning:** Sometimes struggle to verify math/logic
- **Typical Mitigations:**
  - Swap positions of examples + average scores
  - Pairwise > pointwise comparisons
  - Provide reference answers
  - Use multiple LLM judges (ensembles)

LLM-as-Judge: Practical Limits

- **Tends to work for:** checking style, fluency, helpfulness, instruction following, tone, comparative ranking
- **Tends to fail for**: math correctness, code correctness, factual accuracy on obscure topics, subtle logical errors.
- For “verifiable” tasks, different complementary methods being pursued, e.g., post-training techniques

Outline

- Motivation for Data-Centric AI
- Creating Data
- Creating Evals
  - Analyze
  - Measure
  - Improve

Peer Instruction Activity

(Switch to Google Form)

From Measurement to Improvement

- Menu of fixes, from cheapest to most expensive:
  - **Clarify the spec:** Fix Specification gulf (repeat failures) before Generalization gulf (edge cases). Rewrite prompt, add constraints, add few-shot examples from failure taxonomy.
  - **Grow the eval set:** Synthetic traces for underrepresented failures, more labels, refine judge rubric, re-check TPR/TNR.
  - **Restructure pipeline:** decompose prompt, add tool use, etc.
  - Improve **retrieval engineering** -> Topic 2.
  - **Last resort:** Customize the model itself with SFT, post-training, and/or continued pretraining -> Topic 4.
- After any major change: re-run evals to test for *regressions*. Rule of thumb: cheapest sufficient fix first.

Detailed Example

- **Use case:** Chatbot to summarize support tickets for ITS

Failures from error analysis (say, n=100)	Diagnosis	Improvement	Time taken
22%: Summaries too long	Spec gulf (consistent)	Add length constraint + few-shot	Minutes
14%: Missing customer's key ask	Spec gulf	Add rubric "lead with primary request"	Minutes
6%: Hallucinated account details	Generalization gulf (intermittent)	Retrieve order history, ground summary	Few days

- **Result:** 36% of failures fixed before touching pipeline structure. Rerun evals -> identify new dominant failure modes -> iterate.

Review Questions

1. What are the 3 “gulfs” in the context of LLM evals? Explain the difference between any 2 of them with a concrete application example.
2. Why is binary Pass/Fail considered preferable over a 1-5 scale for error analysis and application-specific LLM judge metrics?
3. Suppose you are a developer tasked with launching an LLM-based feature for your company’s website. Which evals scheme would you suggest for starters?
4. Briefly explain one pro and one con of vibe checks as an evals scheme.
5. Which evals scheme discussed in class is the most scalable? Why?
6. Give an example each of reference-based vs. reference-free evals. Which is preferred when feasible, and why?
7. Briefly explain open vs. axial coding in constructing an LLM failure taxonomy.
8. What is “dimension-based” synthesis of evals data?
9. Name three common biases in LLM judges. Propose a mitigation for one.
10. What is the analog of “label leakage” from classical ML for an LLM judge?
11. Your Project 1 teammate proposes to send every RAG query post-deployment to Claude Opus as judge. Is this a good idea? Why or why not?

Recommended Readings

- Application-Centric AI Evals for Engineers and Technical Product Managers. Husain and Shankar. 2026.
- Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. Zheng et al. NeurIPS 2023.
- Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences. Shankar et al. UIST 2024.