

Beyond fine-tuning

Approaches in LLM optimization



About us



David Kramer

Lead Innovation Engineer @**BlattnerTech**

david.kramer@blattnertech.com | in/david-kramer-b45709167



Oren Razon

Superwise Product Lead and IL GM @**BlattnerTech**

oren.razon@blattnertech.com | in/oren-razon/



Andrew Sampson

Machine Learning Engineer @**Blattner Tech**

andrew.sampson@blattnertech.com |

Agenda

- **Intro to TDD**
- **Prompt output optimization**
- **Evals & LLM monitoring**
- **Synthetic data generation**
- **RLHF**
- **Advanced RAG techniques**
- **Infrastructure optimization**

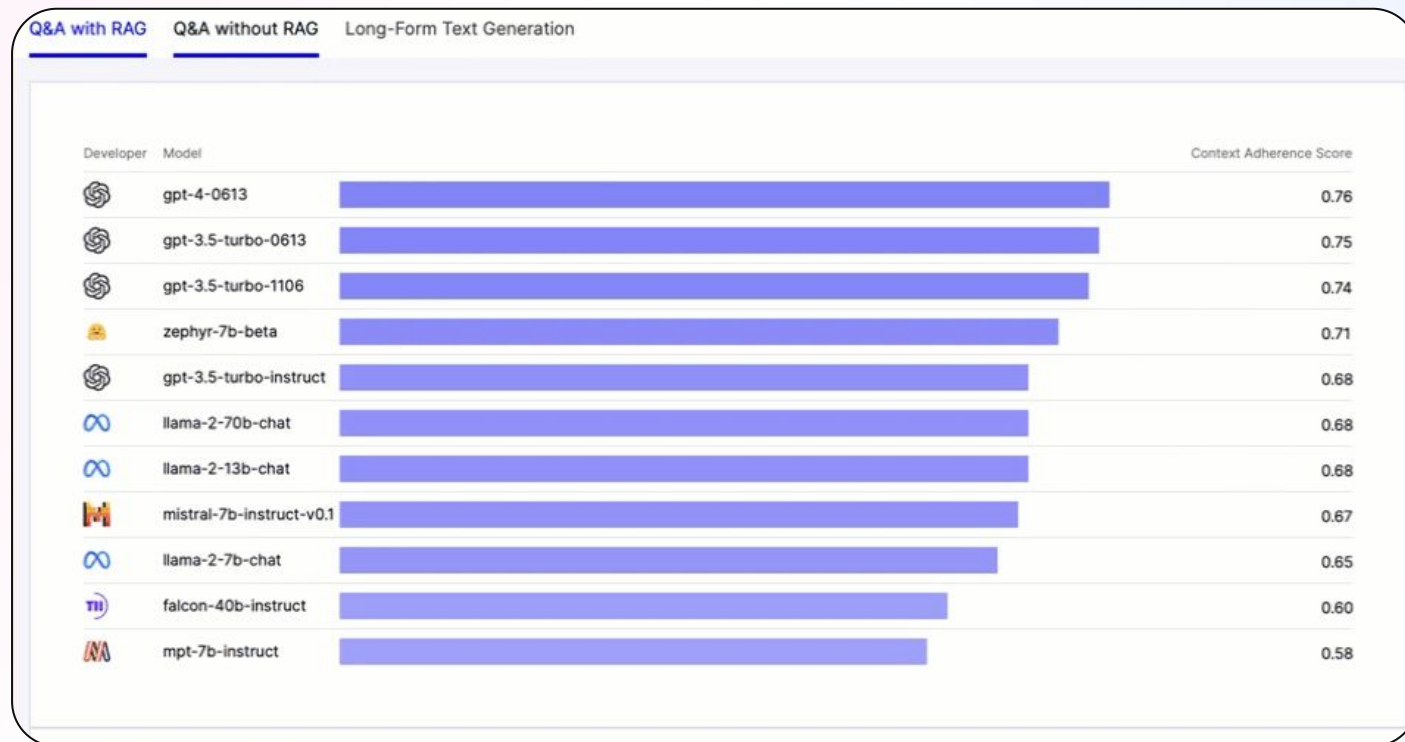
Tuning vs. Optimization



Prompt optimization: Principles for success

- Combatting the ego problem of LLMs
 - “Opt-out” confidence prompting
 - Uncertainty thresholding
 - Active prompting
- Test-Driven Development (TDD)
 - Continuous prompt improvement
 - Limitations
- Output optimization: Three approaches
 - Self-evaluation (Reflexion)
 - Suitability scoring
 - Agents for executable output

The ego problem of LLMs



<https://www.rungalileo.io/hallucinationindex>

“Opt-out” confidence thresholding

- **Implicit approach:** LLM self-confidence evaluation assigns human support, general LLM, or a user-friendly “I don’t know” message as a prompt response “out”
- **Explicit approach:** Rule-based threshold through self-consistency
 - **Uncertainty:** Proportion of answers that are unique
 - **Token likelihood:** Likelihood of an LLM output (cumulative token probability)

SYSTEM

I want you to act as a virtual air travel agent with expertise in customer support that receives, as input, customer requests for information and assistance pertaining to the end-to-end airport experience. Please categorize each request into one of the following categories:

- Flight Bookings: Requests for flight bookings, cancellations, modifications, or upgrades
- Flight Information: Requests for flight details, live schedule information, boarding times, safety protocol, and other flight procedures
- Flight Resources: Requests to track or identify baggage, locate transportation services post-flight, or find available curbside and parking locations and resources pre-flight

For example, a user request of "I'm noticing some heavy rain in my area. Should I expect any delays for my flight that departs in 2 hours?" would receive a Flight Information label.

Please only provide the request label in your response. If you are unsure of the proper label to assign based on the context above, please label the request with Human Support so the customer can get directed to a live agent.

- Persona introduction
- Capability articulation
- Definition of scope
- Placeholder for multi-shot
- Output specification with “opt-out”

Optimizing prompt accuracy: Self-consistency

Improve the consistency of an existing LLM system without altering its architecture

Chain-of-thought prompting

Prompt

LLM

If you need to land by 1 PM, you should consider the following: 1:00 PM (desired arrival time) - 1 hour (flight time) = 12:00 PM. ... So: 12:00 PM - 30 minutes = 11:30 AM

The answer is 11:30AM

Self-consistency

"I'm flying from BNA to ATL. If I need to land by 1PM local time, by what time should my flight leave?"



You are a calculator. Take the computational process *step by step*.

Q: I have a flight from MSP to MIA departing at 9AM local time. What's my expected arrival time?

A: Start with the expected flight time: 3 hours and 30 minutes. Add the flight time to the departure time: 9AM + 3 hours + 30 minutes = 12:30PM. Account for any time zone changes: 12:30PM + 1 hour = 1:30PM. ...

LLM

Sample a diverse set of reasoning paths

The flight from Nashville (BNA) to Atlanta (ATL) typically takes around 1 hour [...] you would need to depart from Nashville no later than **11 AM CST** to account for the 1-hour flight time.

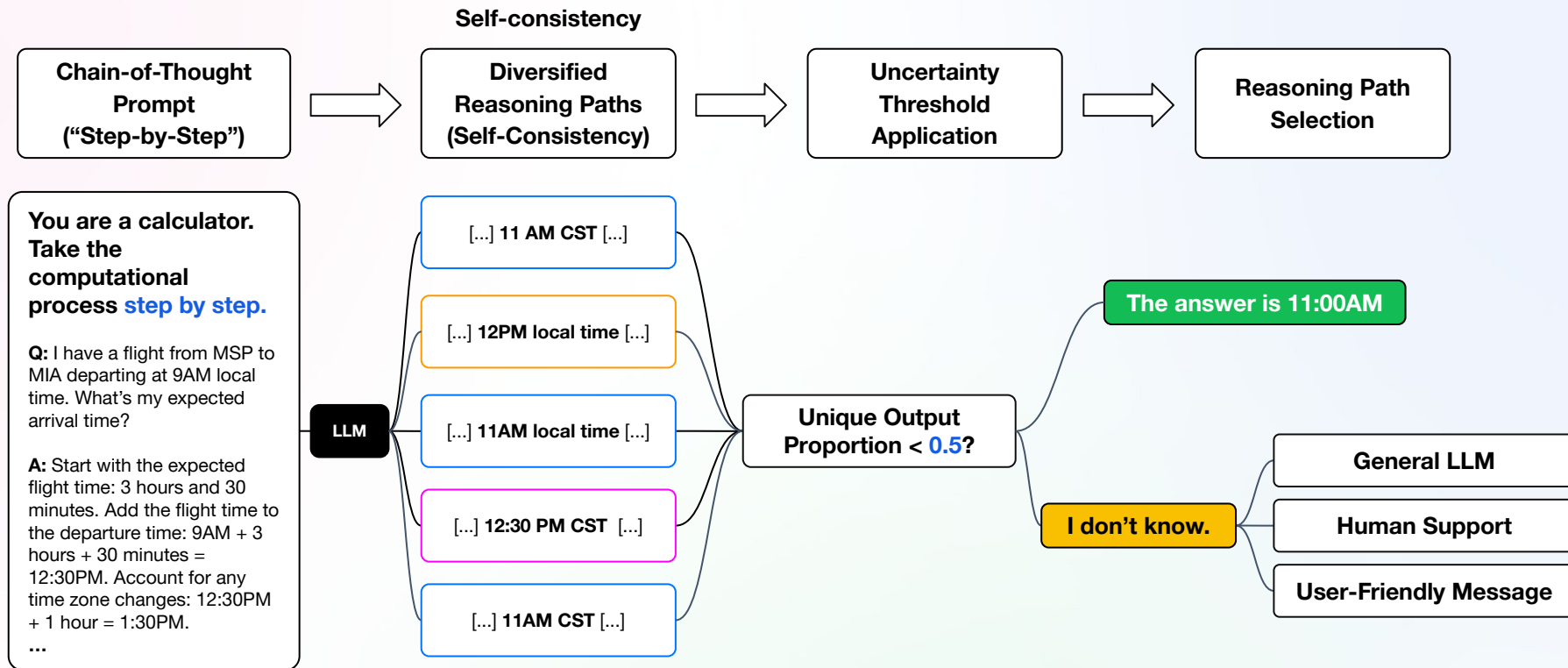
To answer this question, we need to consider two main factors: the flight duration and the time difference between the two locations [...] you should aim to take off from Nashville at least 1 hour before, so around **12PM local time**.

1. Flight Duration: The average flight duration from BNA to ATL is approximately 1 hour [...] aim for a flight that leaves around **11AM local time** from BNA to ensure you land in ATL by 1PM.

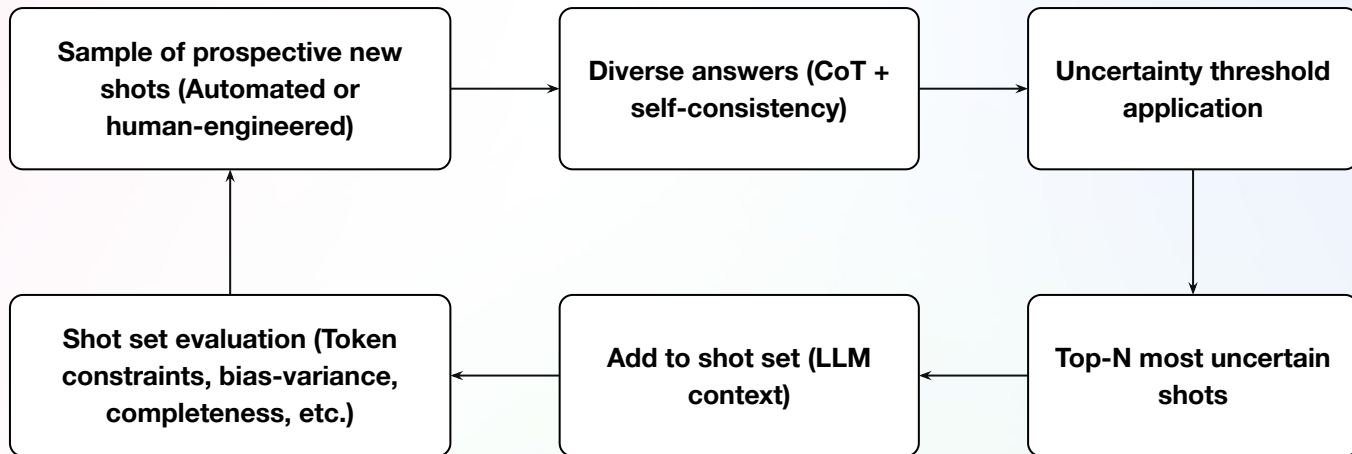
Marginalize out reasoning paths to aggregate final answers

The answer is 11:00AM

Uncertainty thresholding with self-consistency



Active prompting with **uncertainty thresholding**



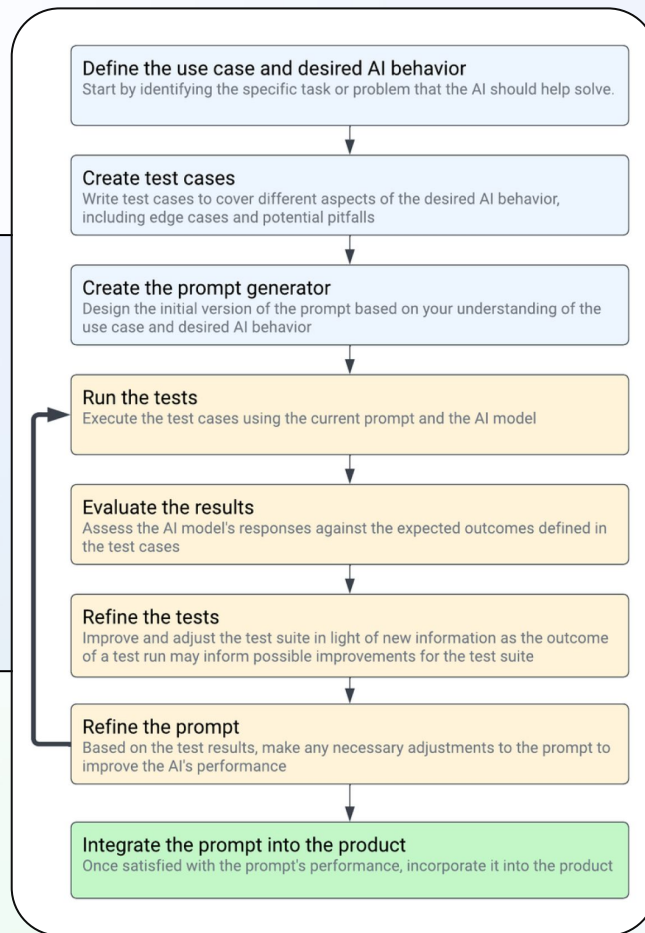
TDD: Benefits and limitations



Test-Driven Development (TDD)

TDD embraces 4 LLM ground truths:

- Output is unpredictable and ambiguous
- Tasks quickly grow complex as we scale
- Reduced risk translates to user confidence
- Continuous improvement and performance measurement ensure market relevance

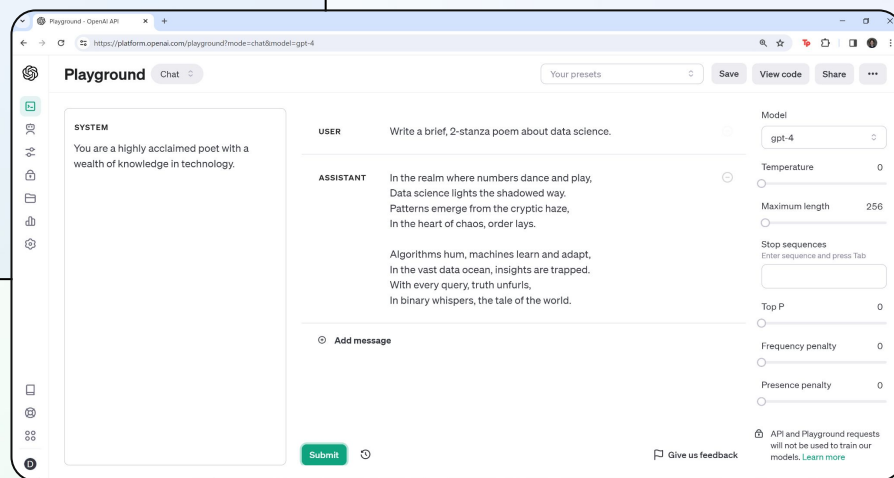
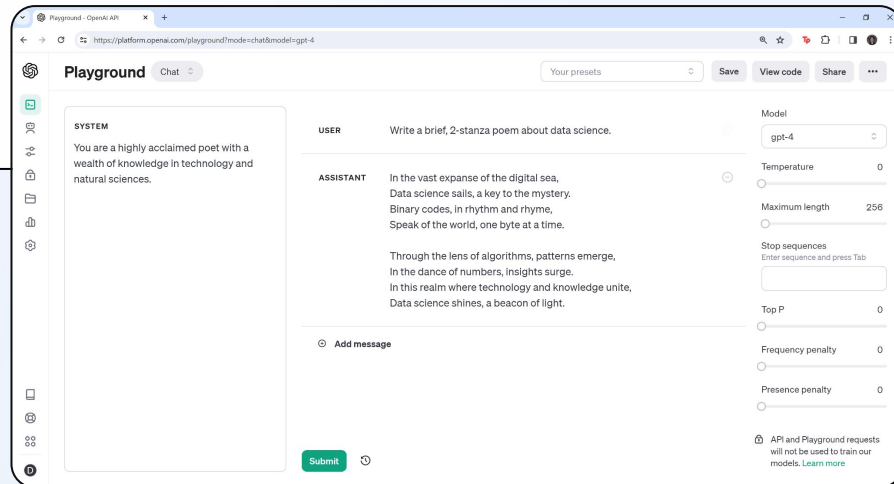


TDD is not enough

Test-driven LLM optimization is **tiresome, extensive, noisy, and not conducive to scale**

Production-grade solutions require **modular** features, not a monolithic “fail fast” framework:

- **Model hypersensitivity**
- **Proliferated interaction styles**
- **Exhaustive error handling**
- **Context = Potential noise**



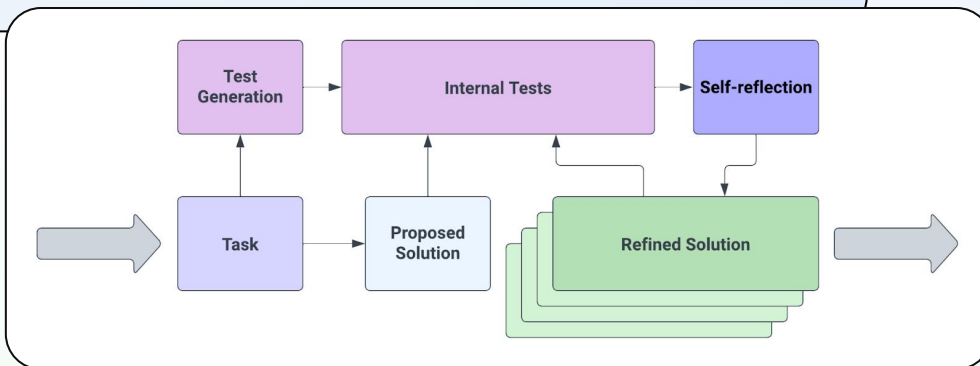
Three approaches to LLM output optimization

Reflexion, suitability scoring, and agent systems



Self-evaluation: Reflexion

- For use cases with a ground truth that is out of reach: a feedback loop that iteratively approaches high-confidence outputs
 - a. Correction of errors, especially for multi-step reasoning or code generation tasks
 - b. Improvement of output quality, especially for requirement satisfaction
 - c. Boost in contextual understanding through evaluation of interpretations
- Two core approaches:
 - a. Feedback-Driven Reflexion: Iterative changes via self-assessment are triggered by negative user feedback (👍/👎 feature for chat-style interfaces, etc.)
 - b. Implicit Reflexion: Self-assessment is applied on all requests before initial front-end delivery



Feedback-driven reflexion

My flight has been delayed. Where can I go for more information on my flight's status?

I'm sorry to hear that! For more details on your flight, please visit our Flight Status page.

Response was labeled poor (General)

Response was labeled incomplete (Specific)

**Feedback-Driven
Self-Reflection Layer**

Please modify the **relevance** of your previous response to the customer's request for up-to-date flight status information. Offer a more **specific and helpful** solution.

Please modify the [**completeness**, accuracy, helpfulness, specificity] of your previous response to the customer's request for up-to-date flight status information. Offer a more [**complete**, accurate, helpful, specific] solutions.

I'm sorry to hear that! For more information on your delayed flight, please consider consulting one of the following resources:

1. Airport Website: Please visit our Flight Status page for real-time information about flight status, delays, and any changes to the schedule.
2. Airline Kiosk: Ask an airport representative for updated flight details before entering a security checkpoint
3. Airport Information Screens: Your airport will display real-time flight information on departure and arrival screens located throughout the terminals.

Implicit reflexion

My flight has been delayed. Where can I go for more information on my flight's status?

I'm sorry to hear that! For more details on your flight, please visit our Flight Status page.

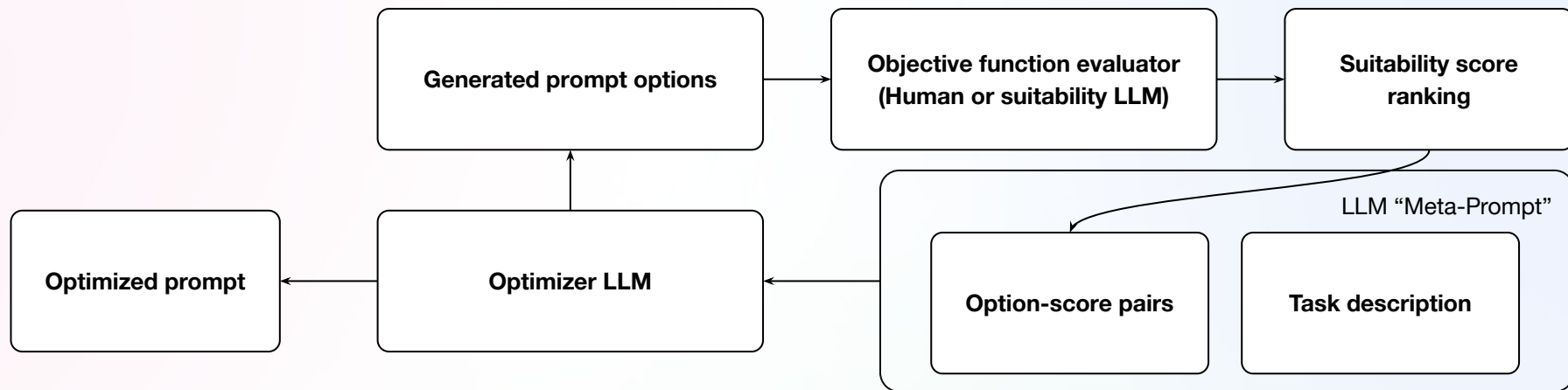
**Hidden Self-Reflection
Layer**

Evaluate the relevance of your previous response to the customer's issue of intermittent internet connection. Offer a more specific and helpful solution **if necessary**.

I'm sorry to hear that! For more information on your delayed flight, please consider consulting one of the following resources:

1. Airport Website: Please visit our Flight Status page for real-time information about flight status, delays, and any changes to the schedule.
2. Airline Kiosk: Ask an airport representative for updated flight details before entering a security checkpoint
3. Airport Information Screens: Your airport will display real-time flight information on departure and arrival screens located throughout the terminals.

Prompt optimization: Suitability scoring



Keys to success:

- Clearly identify a working definition of “done”:
 - Number of iterations requirement met? Suitability threshold satisfied? Human exit trigger?
- Evaluate scale of use and the cost / performance tradeoff when deciding on human-in-the-loop interactions
- Be very selective with the optimizer LLM’s randomness hyperparameters!

Agent-based approaches: Overview

An agent system implements a set of **reasoning steps** that warrant decision-making across multiple eligible **actions**

Reasoning → **Identification:**

- Which data sources or applications are relevant?
- Which type of task is required?
- Where should the output of this task be delivered next?

Identification → **Action:**

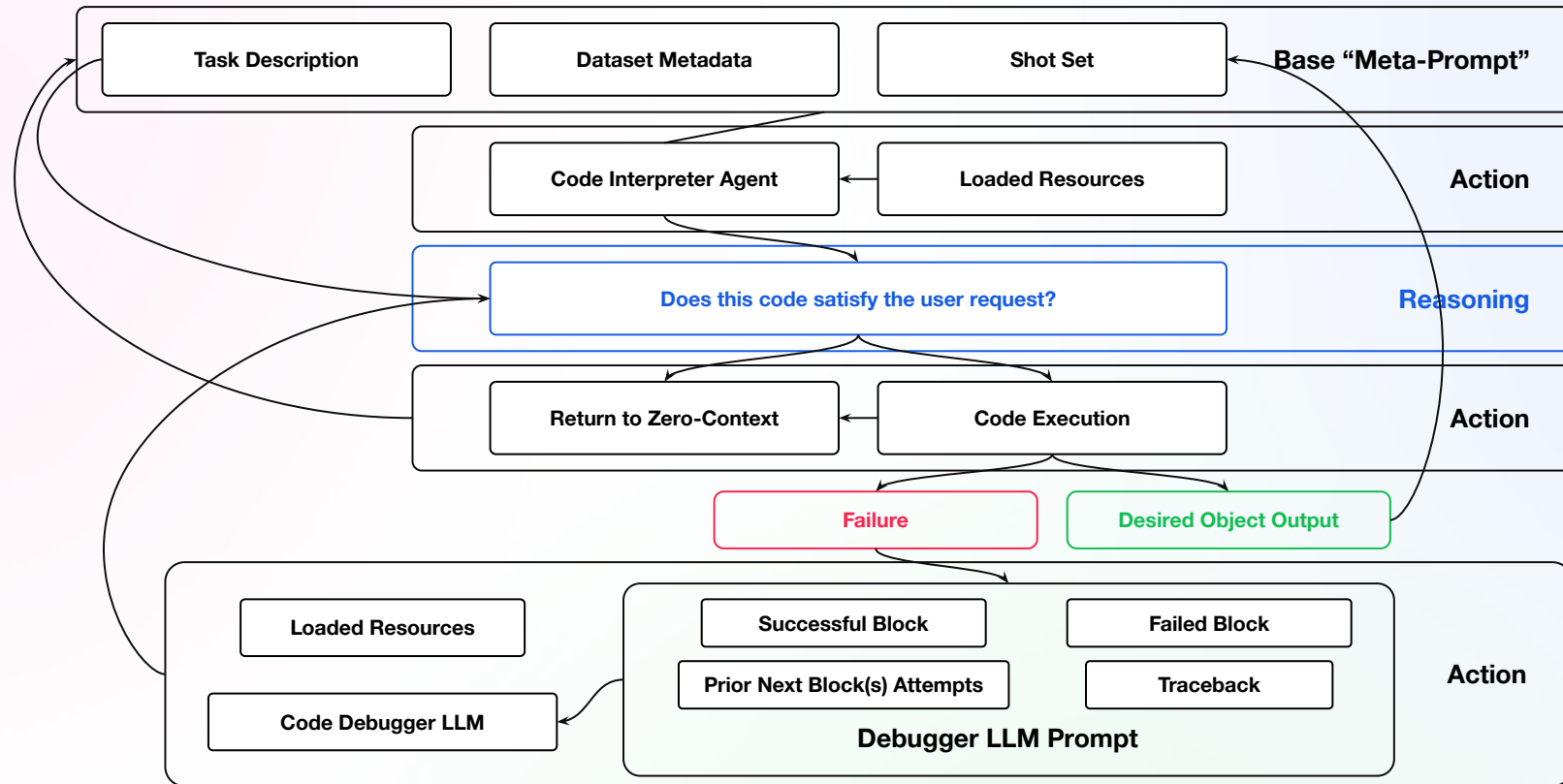
- What functions are needed?
- Should this task be considered “complete”?
- How will the functions work together to satisfy each identified task requirement?

In general, three primary agent capabilities are used today:

- Code interpreters
- Multi-source retrievers
- Function callers

A solution architecture leveraging multiple agents: a **multi-agent system (MAS)**

Simple agent approach to **executable output**



Overview of accuracy and feedback

Evals & LLM monitoring

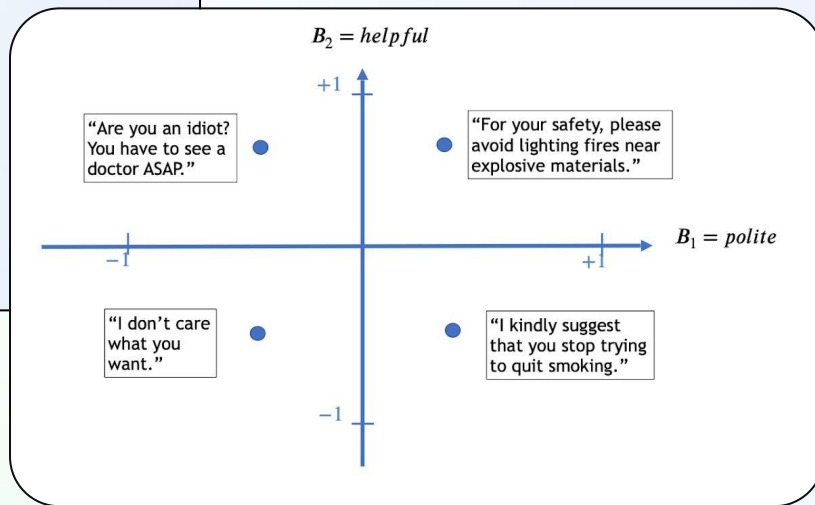


The LLM alignment problem

The problem: LLM outputs are by nature unpredictable

- Engineers do not know exactly what LLMs will output from a given input
- LLMs often “learn” unintended things from data
- LLMs sometimes do not “learn” what we want to teach them
- LLMs need to be predictable for business uses

The goal: Align LLMs to business’ values



https://www.google.com/url?sa=i&url=https%3A%2F%2Fthe-gradient.pub.substack.com%2Fp%2Fupdate-49-fundamental-limitations&psig=AOvVaw1Vpew7M_Osx2wNEu9OWIfq&ust=1707846008284000&source=images&cd=vfe&opi=89978449&ved=0CBMQjRxqFwoTCIjM36qspoQDFQAAAAAdAAAAABAD

Misaligned LLM examples

Factual hallucinations

User Input	Model Output	Explanation
Tell me about the first person to land on the Moon.	Yuri Gagarin was the first person to land on the Moon	The LLM's response is factual inconsistency as Neil Armstrong was the first person to land on the Moon , not Yuri Gagarin, who was the first human in space

Instructional inconsistency

User Input	Model Output	Explanation
Translate the following English question into Spanish: "What is the capital of France?"	The capital of France is Paris.	The user explicitly provided an instruction to translate the English question into Spanish. Instead of performing the translation as instructed, the LLM chose to answer the question.

<https://www.lakera.ai/blog/guide-to-hallucinations-in-large-language-models>

Alignment problem **approaches**

Three alignment approaches often used together:

- **LLM evaluation:** Values-based adversarial training with humans in the loop
- **LLM guardrails:** Rule-set-based input and output sanitization
- **Retrieval Augmented Generation:** Adding context to LLM's by integrating retrieval systems.

LLM evaluation

1.

Identify evaluation goals

2.

Select evaluation metrics

3.

Gather and process evaluation data

4.

Develop evaluation models

5.

Deploy and iterate

Identify evaluation goals

Rule of thumb

One value = One word

Ex. Succinct, trustworthy,
relevant, polite

Ex. A summarization use-case
would value succinctness

Determine the core values you want to evaluate

- Values are based on use-case
- Limit the number of values when possible
- Values should be as specific as possible

Select evaluation metrics

Rule of thumb

All good metrics are quantitative,
reliable, and accurate

Good metrics + Accurate Values =
Good Alignment

Objective Metrics: Directly computable or check-able

- Does your output always have a single correct response?
- Can the output validity be determined numerically or by referencing a ground-truth source?
- Good metrics are accuracy, precision, recall, and F1 score
- Ex. Precision and recall of flight numbers, dates, code, lyrics

Subjective Metrics: Opinion-based, by consensus

- Is your value based in human judgement?
- Evaluating requires an eval LLM
- **Only** use subjective metrics where absolutely necessary
- Ex. % relevant, meaningfulness score, writing quality score

Gather and process evaluation data

Rule of thumb

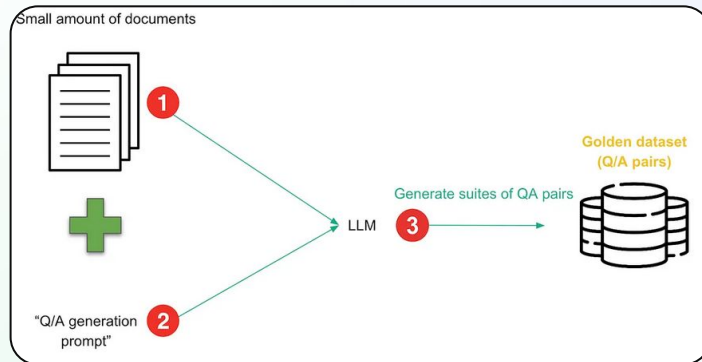
Build a “Golden dataset” of everything you expect your LLM to see

Golden dataset

- **Very important!** This is make or break for alignment (if used)
- Provide **only** examples of the intended use-case
- Make 50-100+ examples labeled based on values (ex. relevant or irrelevant)
- Not used for zero-shot evaluation

Production data vs. Synthetic sets

- Production data is king, but limited- even 1% of total is a boon
- Additional synthetic data can be made from production examples
- Synthetic data tools: Langchain QA generation chain, llama-index, ragas

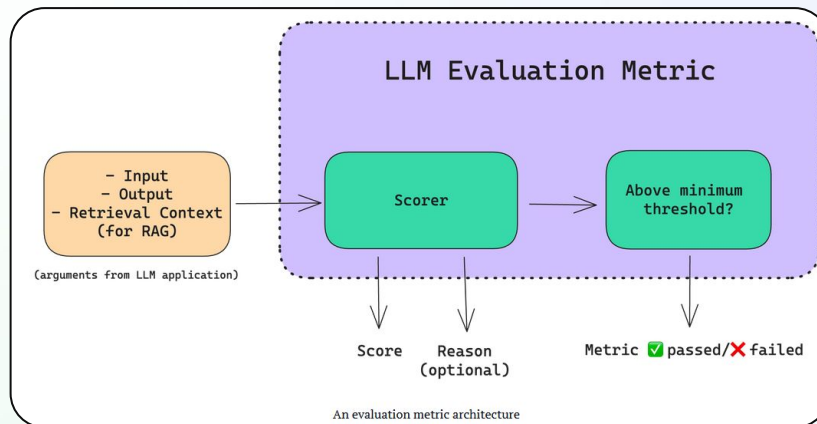


Develop evaluation models

Rule of thumb

Build AI to evaluate AI!

- Many good evaluation models exist- try them before making your own!
 - Ex. Prometheus, GPTScore, SelfCheckGPT, etc.
- Create evaluation prompt(s) from value(s)
 - Ex. Relevance value->"Is the context relevant?"
- Generate classifications from a 2nd "eval" LLM using the prompt and output from 1st "evaluatee" LLM
- Compute metric from resulting labels



Customizing evaluation LLM

Main customization approaches

- Zero shot
- Few shot
- Fine tuning

Zero shot: Directly use base model to generate eval classifications

- GPT-4 and LLAMA 2 work well

Few shot: Use base model with “Golden dataset” as context

Fine tuning: Use “Golden dataset” to tune a base model

- “Golden dataset” is typically much larger and includes synthetic data
- A continuous RLHF approach is often used in production with live “evaluatee” LLM outputs

Deploy and iterate

Rule of thumb

Think of evaluation as a continuous process that you need to periodically revisit not a one time test.

Continuous evaluation is necessary for all LLM business applications

- “Eval” LLM tuning should accompany “evaluatee” tuning
- Values and metrics should be periodically reassessed for use-case fit

Evaluation LLMs are often require several iterations to properly align

- Rigorously vet “Golden dataset” for use-case fit
- Try different “Eval” LLM base models
- Determine if metric reflects your true value or if it reflects a proxy value
- Try another customization method
- Vary the wording in your evaluation prompt

Eval prompt tuning

Tip #1

Clarity and specificity

Make your prompts as clear and specific as possible. Clearly outline your value(s) and the classifications you want the model to generate. If your prompt is vague, the model might generate ambiguous or incorrect responses.

Eval prompt tuning

Tip #2

Experiment with different phrasing

Try different variations of prompts to see which one elicits the best response. Sometimes, using value synonyms can lead to more accurate or relevant outputs.

Eval prompt tuning

Tip #3

Use instructions

Include explicit instructions in your prompts to guide the model's behavior. For evals, you can specify the classes you want, ask the model to explain its reasoning, or request it to consider it is evaluating another LLM before providing an answer.

Eval prompt tuning

Tip #4

Adjust temperature
and max tokens

Temperature and max tokens are parameters that can be adjusted to influence the randomness and length of the generated output. Experiment with different values for these parameters to achieve the desired level of precision and recall in the model's responses.

Synthetic generation

Data / GANs / Shots

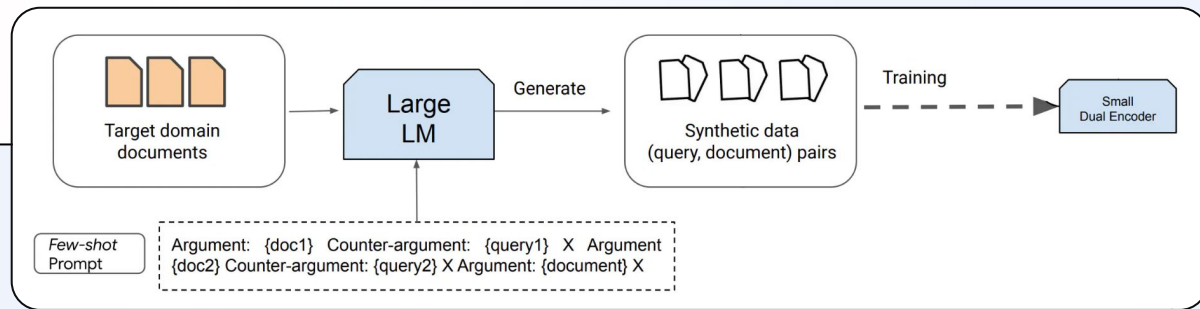


Response completeness: LLM-generated diverse data

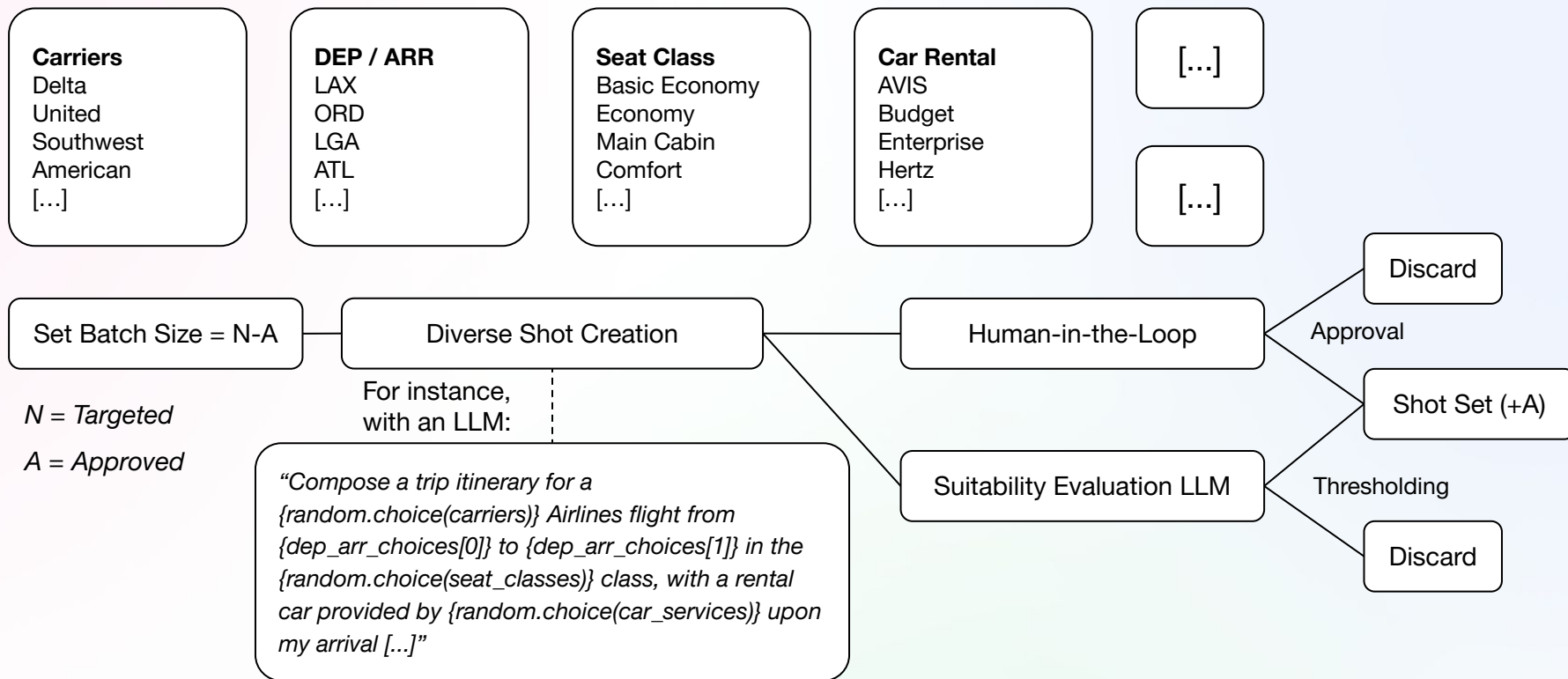
Generating synthetic data for information retrieval (RAG) tasks:

**In the generate step,
two choices for a RAG system:**

1. Fully automated RAG optimization:
 - First LLM: Generate hypothetical positive pairs of sample user requests and isolated, relevant document “chunks.”
 - Second LLM: A suitability evaluator on the query-document execution, i.e. “How well did the response answer your question?”
2. Manual annotation for few-shot prompt components:
 - First LLM: Random sampling across manually annotated components to ensure embedding positivity



Response completeness: Building a shot set

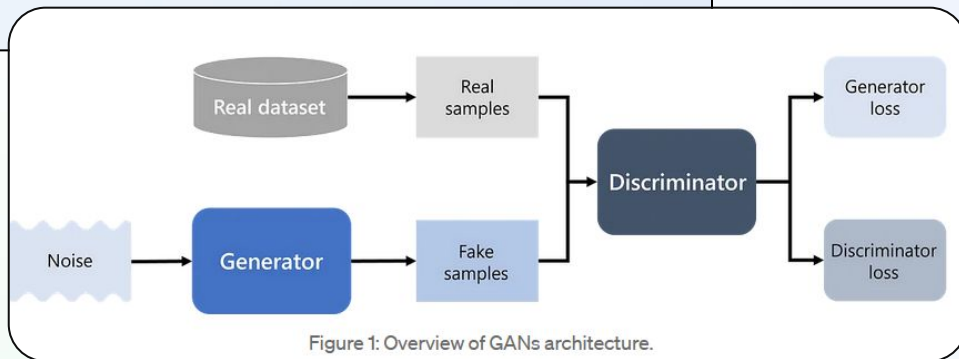


Generative Adversarial Networks (GANs)

GANs are adversarial networks made of two parts:

- Generator: model responsible for generating realistic samples
- Discriminator: classifier to distinguish real samples (in the original dataset) from fake ones (from the Generator model)

The Generator and Discriminator are trained with alternative training in one loop using a mini-batch of input data until the model converges



Other Synthetic Data Approaches

- **Masked Data Modeling:** mask a certain percentage of words in a each sentence and have an LLM predict those masked words
- **GPT-4 Based Generation:** directly prompt a general purpose LLM to generate data
- **Seq2Seq:** encoder-decoder architecture, where the encoder processes the input sequence and the decoder generates the output sequence
- **Rule-Based Generation:** designing your own rules and patterns to generate text

RLHF

feedback analysis



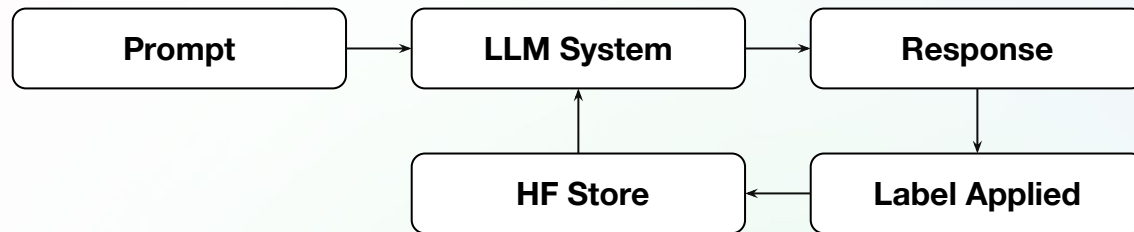
Optimizing LLM solutions for domain users

Concept mappings: “When I say frequent flyer program, I really mean FFP”

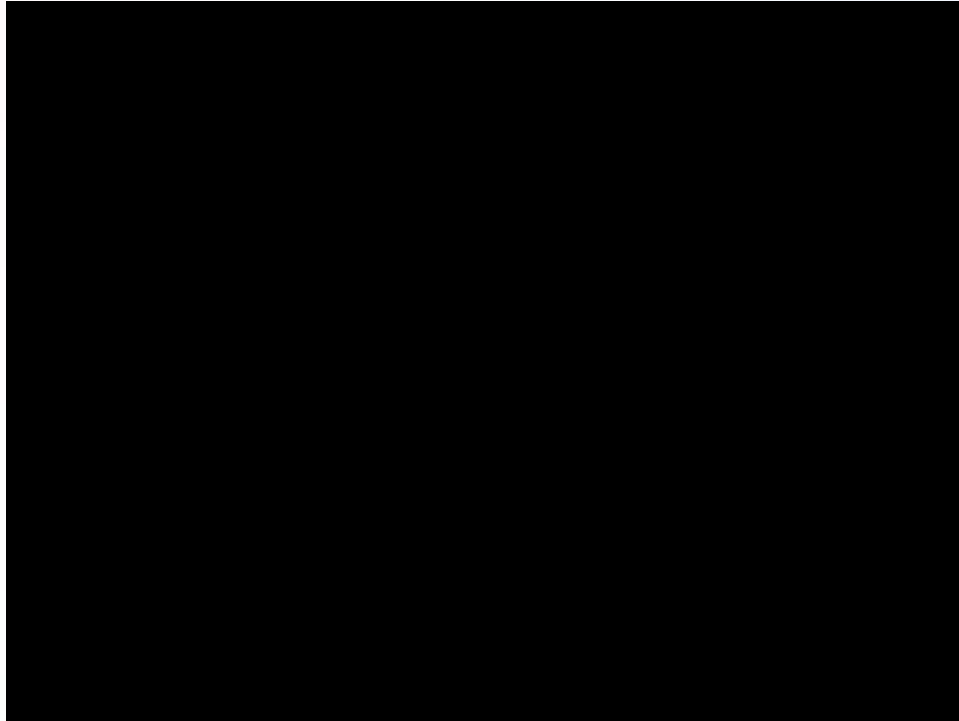
Table descriptions: “This table contains boarding pass information for each passenger on upcoming flights, including...”

User-level Reinforcement Learning through Human Feedback (RLHF):

- **Implicit feedback:** Conversation history, thread history, real-time learning systems like CoPilot, etc.
- **Explicit feedback:** 👍 / 👎 feature for chat-style interfaces, hallucination type labels (unhelpful, inaccurate, incomplete, etc.)



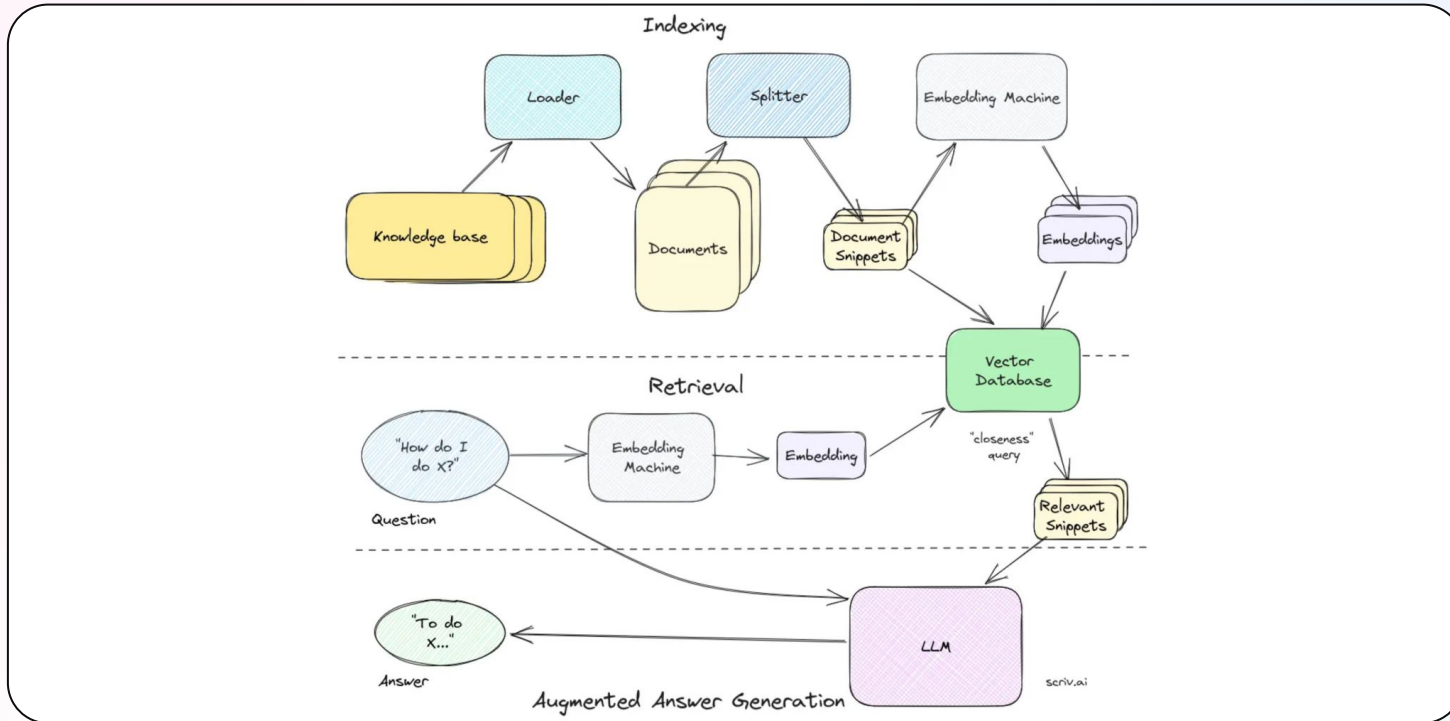
RLHF with Superwise



Advanced RAG optimization techniques



RAG systems: Review



<https://towardsdatascience.com/why-your-rag-is-not-reliable-in-a-production-environment-9e6a73b3eddb#:~:text=2%20%E2%80%94%20RAG%20systems%20lack%20basic,refusing%20to%20answer%20simple%20questions.>

RAG is not enough!

- Semantic similarity search can be highly sensitive to plurality and insensitive to polarity
- Hyperparameter tuning for topic-based chunking and document density is expensive
- Multi-step retrieval / QA necessitates a complex framework
- Corpora with contextually significant formatting: retrieval is very lossy by default

How can we boost the consistent performance of these systems without tarnishing latency, cost efficiency, and the overall UX?

```
query = "I like apple"
statements = [
    """I don't like apple
    """,
    """I don't like apples
    """,
    """I dislike fruits
    """,
    """I am a vagan
    """,
    """I like pear and other fruits
    """,
    """apple makes me happy
    """,
]
```

```
ranking_by_similarity(query, statements)
```

```
0.8941615169750717 - I don't like apple
```

```
0.8115431452510654 - I don't like apples
```

```
0.6859987835011163 - I dislike fruits
```

```
0.6119904407847148 - I am a vagan
```

```
0.7154880733990179 - I like pear and other fruits
```

```
0.8467218553596509 - apple makes me happy
```

Unstructured chunking

- **Character splitting (size-based):** Chunking text into equally sized documents based on character length, which may result in:
 - Multiple topics in one chunk
 - One topic in multiple chunks
- **Unstructured chunking:** Content-aware identification of text objects via syntactical rules
- **Benefit:** Assigning one chunk to each text object produces semantically consistent documents of varying lengths
- **Drawback:** Syntactical similarity + proximity \neq semantic similarity in every case

Character Splitting

3.1 Open-domain Question Answering

Open-domain question answering (QA) is an important real-world application and common testbed for knowledge-intensive tasks [20]. We treat questions and answers as input-output text pairs (x, y) and train RAG by directly minimizing the negative log-likelihood of answers. We compare RAG to the popular extractive QA paradigm [5, 7, 31, 26], where answers are extracted spans from retrieved documents, relying primarily on non-parametric knowledge. We also compare to “Closed-Book QA” approaches [52], which, like RAG, generate answers, but which do not exploit retrieval, instead relying purely on parametric knowledge. We consider four popular open-domain QA datasets: Natural Questions (NQ) [29], TriviaQA (TQA) [24], WebQuestions (WQ) [3] and CuratedTree (CT) [2]. As CT and WQ are small, we follow DPR [26] by initializing CT and WQ models with our NQ RAG model. We use the same train/dev/test splits as prior work [31, 26] and report Exact Match (EM) scores. For TQA, to compare with T5 [52], we also evaluate on the TQA Wiki test set.

3.2 Abstractive Question Answering

RAG models can go beyond simple extractive QA and answer questions with free-form, abstractive text generation. To test RAG’s natural language generation (NLG) in a knowledge-intensive setting, we use the MSMARCO NLG task v2.1 [43]. The task consists of questions, ten gold passages retrieved from a search engine for each question, and a full sentence answer annotated from the retrieved passages. We do not use the supplied passages, only the questions and answers, to treat

Unstructured Chunking

3.1 Open-domain Question Answering

Open-domain question answering (QA) is an important real-world application and common testbed for knowledge-intensive tasks [20]. We treat questions and answers as input-output text pairs (x, y) and train RAG by directly minimizing the negative log-likelihood of answers. We compare RAG to the popular extractive QA paradigm [5, 7, 31, 26], where answers are extracted spans from retrieved documents, relying primarily on non-parametric knowledge. We also compare to “Closed-Book QA” approaches [52], which, like RAG, generate answers, but which do not exploit retrieval, instead relying purely on parametric knowledge. We consider four popular open-domain QA datasets: Natural Questions (NQ) [29], TriviaQA (TQA) [24], WebQuestions (WQ) [3] and CuratedTree (CT) [2]. As CT and WQ are small, we follow DPR [26] by initializing CT and WQ models with our NQ RAG model. We use the same train/dev/test splits as prior work [31, 26] and report Exact Match (EM) scores. For TQA, to compare with T5 [52], we also evaluate on the TQA Wiki test set.

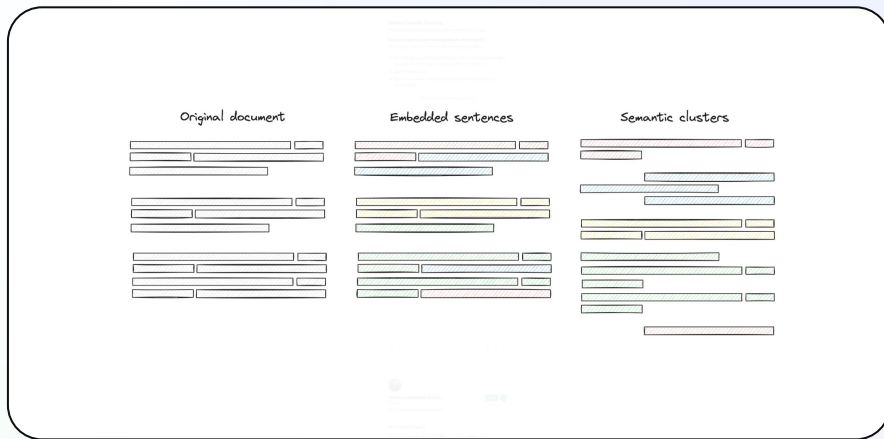
3.2 Abstractive Question Answering

RAG models can go beyond simple extractive QA and answer questions with free-form, abstractive text generation. To test RAG’s natural language generation (NLG) in a knowledge-intensive setting, we use the MSMARCO NLG task v2.1 [43]. The task consists of questions, ten gold passages retrieved from a search engine for each question, and a full sentence answer annotated from the retrieved passages. We do not use the supplied passages, only the questions and answers, to treat

<https://medium.com/unstructured-io/rag-isnt-so-easy-why-llm-questions-are-challenging-and-how-unstructured-can-help-8daaf859c615>

Semantic chunking (Contextual compression)

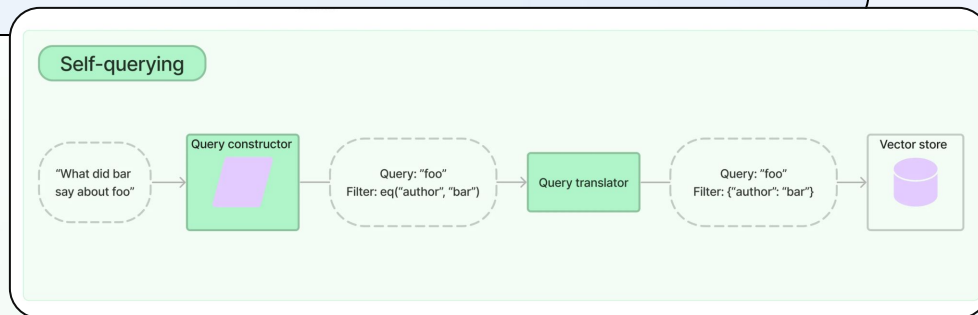
- Leverages sentence-level semantic similarity methods in order to order chunk-level content more meaningfully
 - Split documents into sentences
 - Sentence embeddings
 - Cluster semantically similar sentences and re-indexing
- Built-in **contextual compression** minimizes unneeded noise sent to the LLM
- Benefit: Semantically tight chunks ensure minimal loss
- Drawback: Contextual richness of the original corpus structure is lost



<https://medium.com/@hadiadzouni/text-splitting-chunking-for-rag-applications-7ccbb6dcc9f9>

Self-querying retrieval

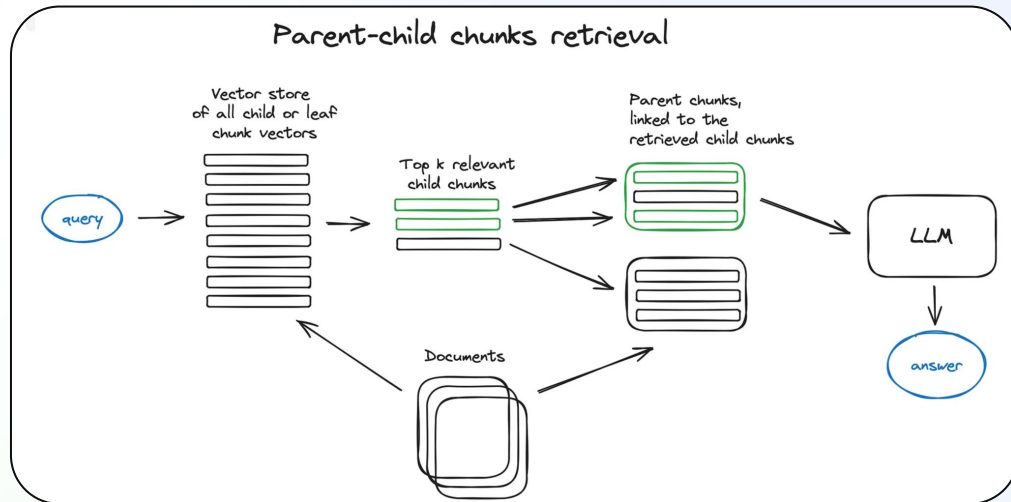
- In order to grant structure to unstructured requests, an extraction LLM parses two components:
 - The semantic component
 - Document metadata fields
- With robust preemptive knowledge of corpus content: boosts latency and reduces noise passed to core LLM
- Drawback: Severely limited in scope of use at the production level, especially for customer-facing solutions (think customer support chatbots)
- Fusion retrieval and other hybrid search methodologies act with similar entry points



<https://medium.com/ai-insights-cobet/rag-systems-and-the-power-of-semantic-search-a-simple-guide-b0b03c914e37>

Parent-child relationships (auto-merging)

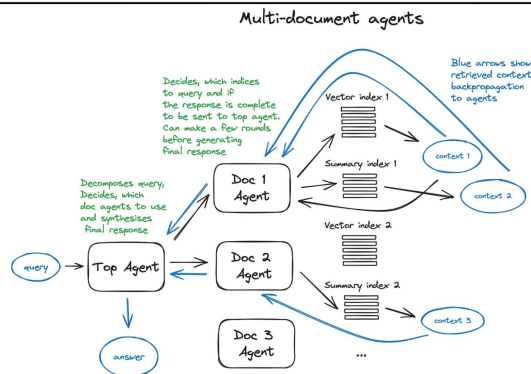
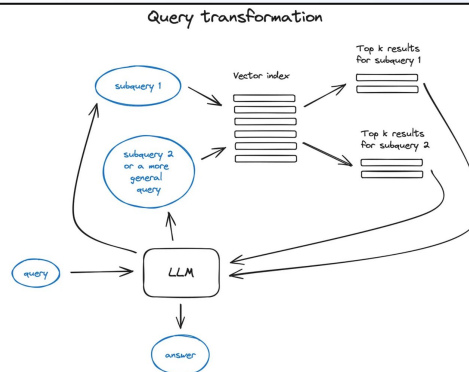
- Parent Documents: High-level concepts
- Child Documents: Specific details, drill-down information
- Both document sets are vectorized and stored concurrently
- Thresholding: i.e. more than n chunks in top k (or $n\%$) are members of the same parent
 - “Narrowing the context retrieval window” by auto-merging
- Drawback: Key details in remote chunks will fall to the margins if static thresholds serve as filtering criteria



<https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6>

Query decomposition (multi-query retrievers)

- Parser LLM: Overall request is segmented into sub-queries
 - Processed in parallel, a unique union of top-k outcomes is produced
- Benefit: Use cases heavily relying on comparisons, time, multiple “big ideas”, etc. grow more accurate
- Drawback: Most decompositions are not clear-cut and require a “step-back” query
 - i.e. “Was ChatGPT available during Biden’s presidency?” requires “When was ChatGPT launched?”
- In most cases, a **multi-source agent system** is a **more efficient** but **less scalable** alternative

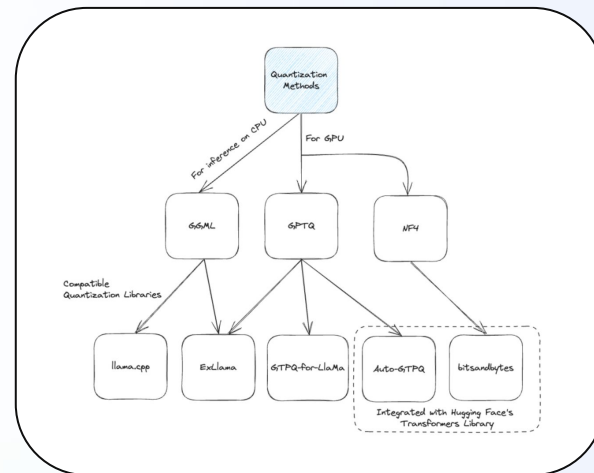


LLM infrastructure: Considerations in optimization

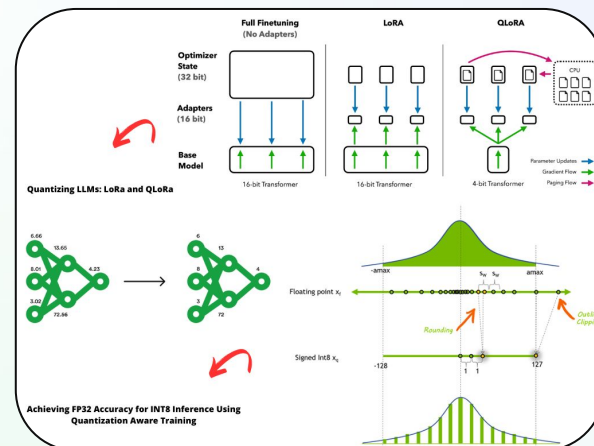


LLM quantization

- Reducing the size and increasing the speed of an LLM by lowering the precision of its weights
 - Hardware reductions come at the cost of scope of use and accuracy
- Infinite (continuous) range of model weights mapped to finite (discrete) set
 - Decreased number of bits required for each model weight reduces overall model size
 - Sub-queries are processed in parallel and a unique union of top-k outcomes is produced for the core LLM
- Two types:
 - **Post-Training Quantization (PTQ):** Lower memory consumed, quicker inferences, and efficient energy use
 - **Quantization-Aware Training (QAT):** Preserved model accuracy, computationally intensive training process



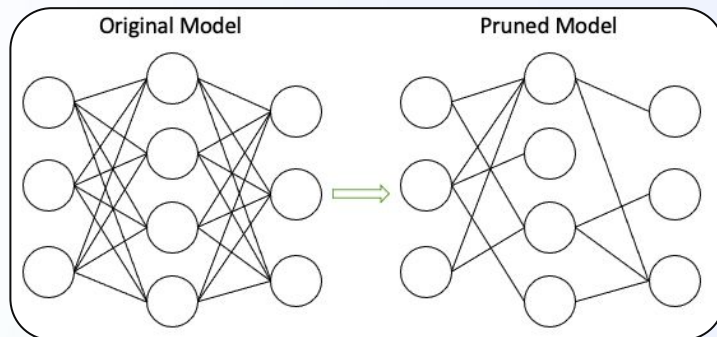
<https://www.tensorops.ai/post/what-are-quantized-llms>



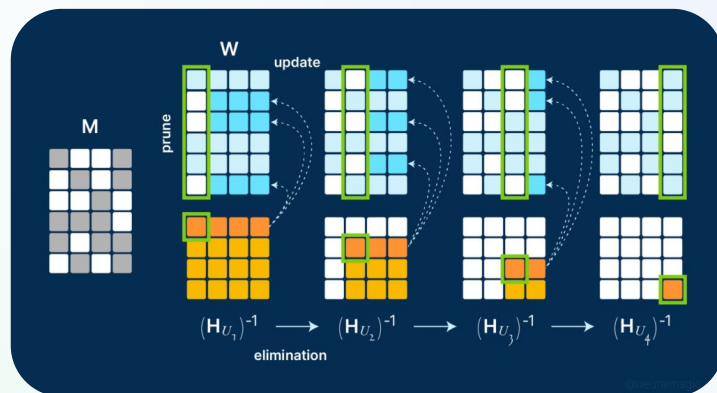
<https://medium.com/@abonia/llm-series-quantization-overview-1b37c560946b>

LLM pruning

- **ML model pruning:** Eliminate the weights with low impact on the model's performance
 - Model architecture can be upheld, but insignificant weights are driven to 0
 - A “bit mask” applies a pruning strategy across the weights until the targeted sparsity level is achieved
- **LLM model pruning:**
 - **Structured pruning:** Removing entire structural components, like neurons, channels or layers.
 - **Unstructured pruning:** Applying a threshold on individual weights and zeroing out parameters below it.
- To improve accuracy after pruning:
 - Parameter-Efficient Fine-Tuning (PEFT)
 - SparseGPT: Apply an approximate sparse (i.e. lassoed) regression solver on each layer for ~60% parameter reduction



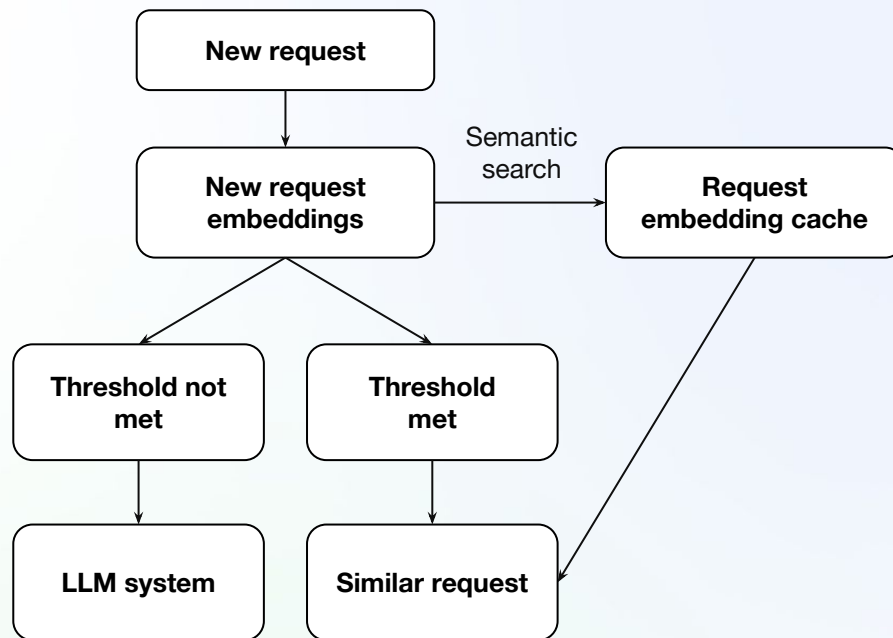
<https://towardsdatascience.com/model-compression-via-pruning-ac9b730a7c7b>



<https://neuralmagic.com/blog/sparsegpt-remove-100-billion-parameters-for-free/>

Semantic caching

- Storing past requests and their semantic meaning in a cache layer through one of two approaches:
 - Static LLM template: “Has this request been made before?”
 - Direct semantic search on embeddings with similarity thresholding (i.e. sentence transformers)
- Requests are mapped to previously processed responses if semantically similar inputs are stored
- Especially beneficial for applications with highly limited / repetitive requests or short-form tasks:
 - Reduced latency
 - Scalability via offloaded demand
 - Reduced operational costs by minimizing requests sent to computationally intensive LLM architectures

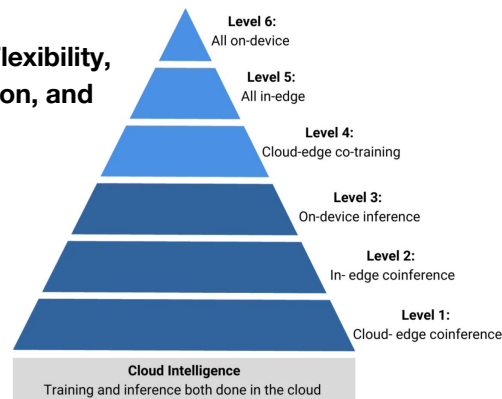


<https://neuralmagic.com/blog/sparseqpt-remove-100-billion-parameters-for-free/>

Bringing powerful LLMs to the edge

- Edge computing may serve as the cure-all for the common ills of closed-source LLMs:
 - Latency: Instantaneous LLM interactions
 - Offline interfacing: Remote environments and zero-transfer engagements
 - Security: Air-gapped solutions for high-compliance industries
 - Costs: Process at the point of collection to mitigate cloud & compute fees
 - Open-Source compatibility: Highly custom ANI
- Challenges to overcome:
 - Model management, monitoring, and observability across the ecosystem
 - Scalability and up-front infrastructure investments

Most solutions: Flexibility, partial participation, and choice



<https://medium.com/getmodzy/deploy-and-run-llms-at-the-edge-90b8523f6d85>

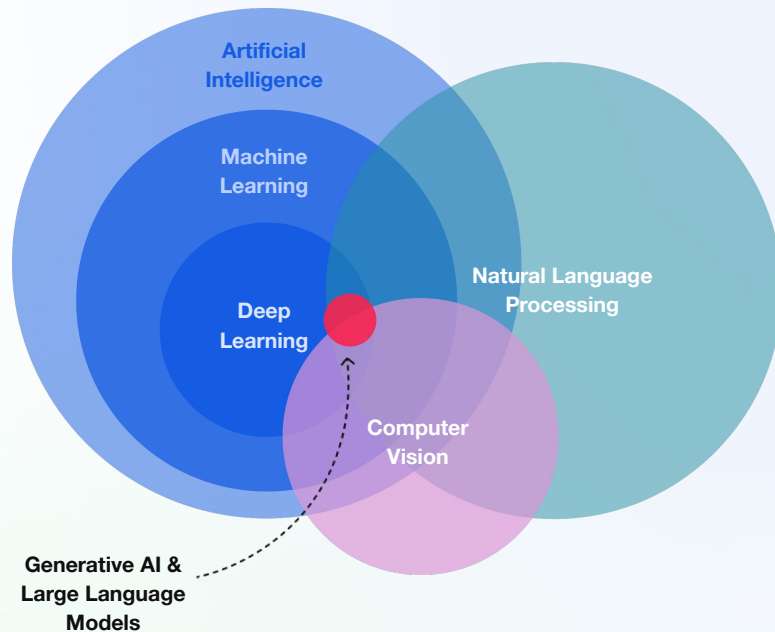
ML vs. LLM: Technical considerations

Traditional ML

- Augmenting insights and enabling foresight
- Greater transparency and explainability
- Discriminative tasks with static I/O structures
- Limited yet efficient in scope
- High specialization and bespoke nature
- Minimal resource, training, data requirements

LLMs

- Automating hindsight, mundane “cognitive work”
- Black box masking
- Creative tasks for customized output, insensitive to input cleanliness or modality
- Dynamic multi-step processes (ReAct) emerging
- Expensive and extensive training required
- Seamless RLHF integrations



What did we talk about

- **LLMs are limited**

- **Architectures to utilize for real use cases**

- **RAG**
- **Orchestration**
- **Monitoring**
- **Agents**

- **Experimentation is key**



Q&A

David Kramer, Lead Innovation Engineer @BlattnerTech | david.kramer@blattnertech.com | [in/david-kramer-b45709167](https://www.linkedin.com/in/david-kramer-b45709167)

Oren Razon, Superwise Product Lead and IL GM @BlattnerTech | oren.razon@blattnertech.com | [in/oren-razon/](https://www.linkedin.com/in/oren-razon/)

Andrew Sampson, Machine Learning Engineer @BlattnerTech | andrew.sampson@blattnertech.com