

# Evaluating LLMs and LLM Systems

Pragmatic approach

Andrei Lopatenko, PhD, VP AI and Engineering

# About the author

Nearly two decades dedicated to the development and management of cutting-edge Search and Recommendation engines across global and US tech companies, including Google, Apple, WalmartLabs, eBay, and Zillow.

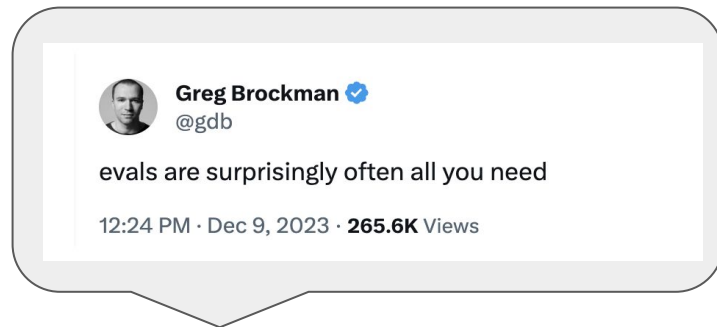
Nearly two decades of integrating language models into large-scale applications, evolving from classical NLP techniques to pioneering work with BERT and the latest advancements in Large Language Models (LLMs) over the past 5 years.

Nearly two decades of the evaluation of AI applications for consumers and businesses, striving to improve usability and satisfaction in large-scale systems.

Leading LLM Science teams, LLM models and evaluation, leading end-to-end Search/Recommendation/Personalization teams, NLP platform, Call Center Intelligence platforms

My academic journey: PhD in Computer Science from the University of Manchester, UK (Logic Based Data Integration, Logic based AI), which laid the groundwork for my career in AI.

# Importance of Evaluation



In the early 2000s, the search engine landscape was marked by fierce competition among several key players, including Google, Yahoo, and Microsoft. Each of these companies boasted a highly talented team of scientists and engineers dedicated to refining their search technologies—encompassing query understanding, information retrieval, and ranking algorithms—according to their specific evaluation metrics. While each company believed their search engine outperformed the competition based on their internal assessments, there emerged one among them that was universally recognized as superior from the perspective of billions of users worldwide. This search engine ultimately rose to prominence, becoming the dominant force in the market. (Google)

**The moral of this story underscores the critical importance of evaluation as a cornerstone of success.**

# Purpose of this presentation

No metrics is good and final, no evaluation is comprehensive

More important to know *principles* of building *good* metrics and *evaluation data sets* and how to apply them to build *evaluation of your systems for your business needs*, rather than to apply a particular metrics and benchmark invented by someone else for a their tasks,

So this presentation is intended to be broad, to show good principles and examples

# Purpose of this presentation

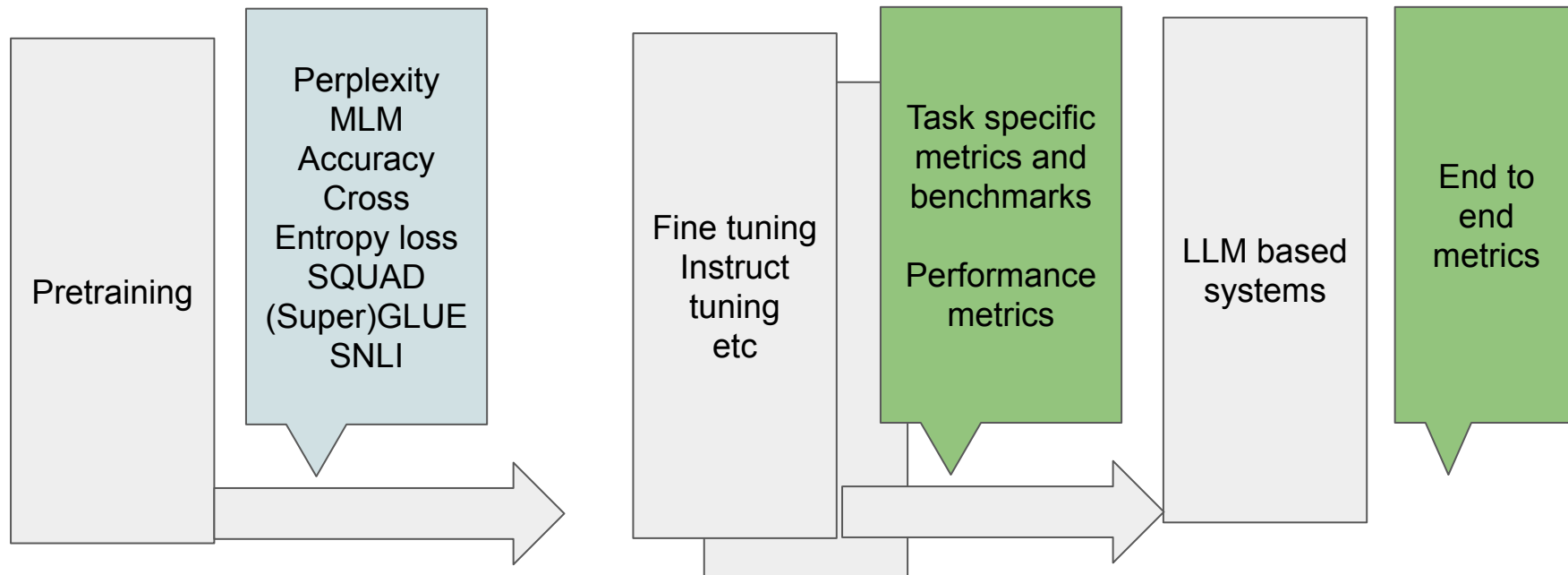
In this presentation, we talk about industrial use of LLM, developing your use cases and products,

Building LLM products in many companies is about fine tuning, instruction tuning of models, training adapters, building LLM augmented systems and continuous circle of improvements of those systems and models

This talk is a landscape talk: to provide high overview of the evaluation landscape, to survey trends and methods, provide orientation so you can choose what's the best for you (we need a multi-hour evaluation class to go deeply). I'll be a bit fast to survey more

# LLM Training and Evaluation

Focus of this talk because most businesses will need to use it



# Content

Why LLMs Evaluation is different from the classical ML Evaluation

Why to evaluate & Evaluation Driven Process

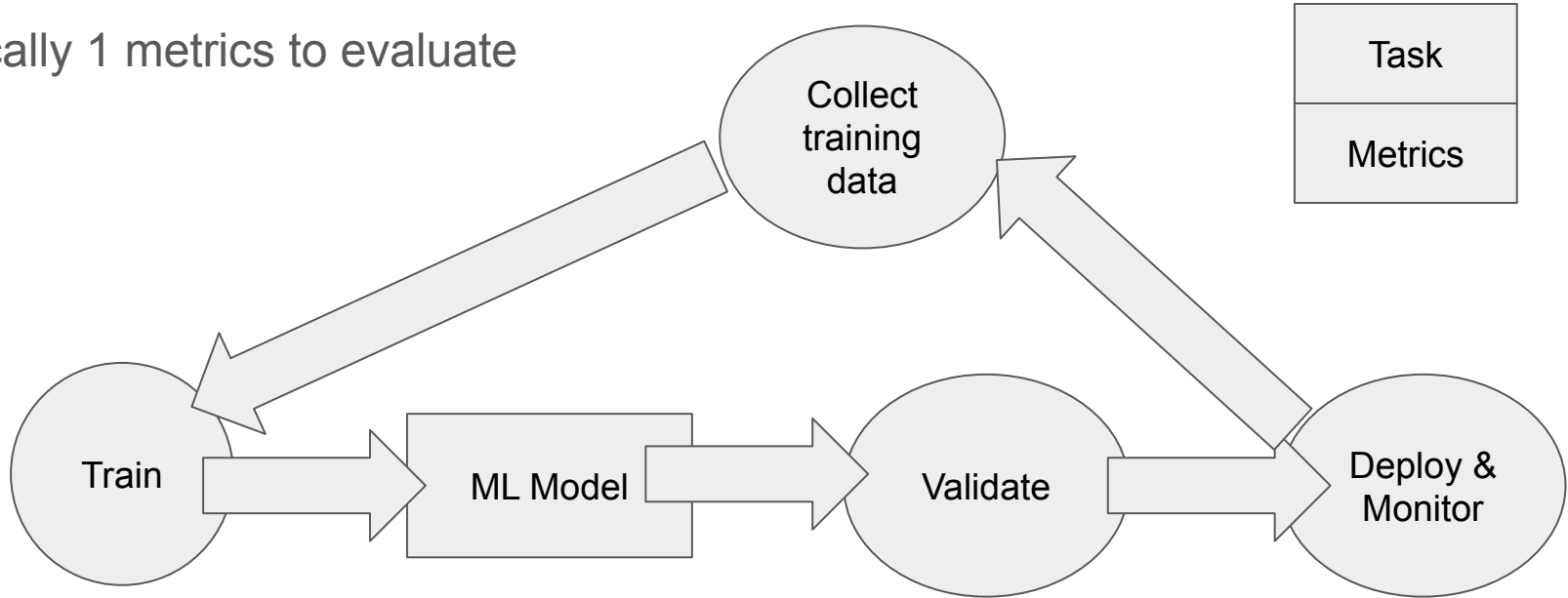
Types of Evaluation

End to End evaluation

Evaluation Harnesses

# Classical ML workflows

Typically 1 metrics to evaluate

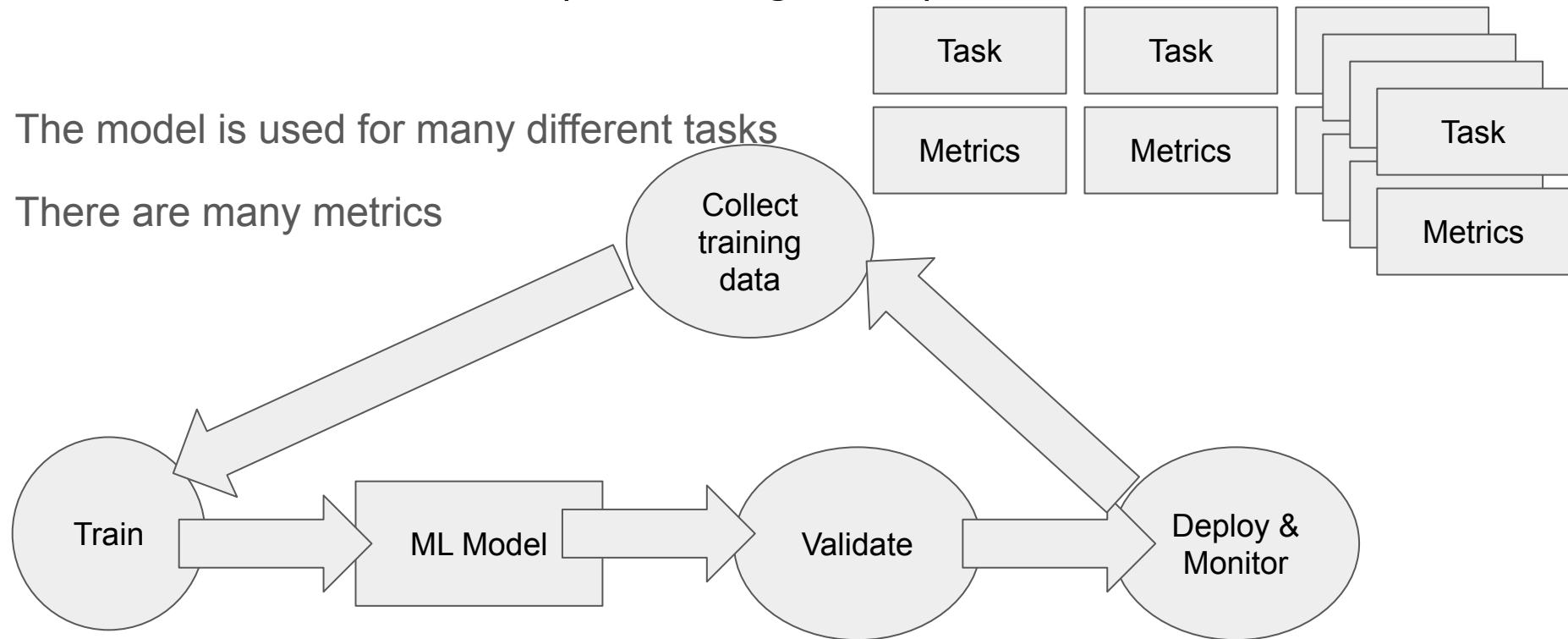




# Foundational Models (including LLM)

The model is used for many different tasks

There are many metrics





# LLM Diversity of semantic areas

LLMs are used for very different semantic areas with the same business . The performance may vary significantly

See for example <https://arxiv.org/pdf/2009.03300.pdf>

Or Fig 3 from

<https://arxiv.org/pdf/2401.15042.pdf>

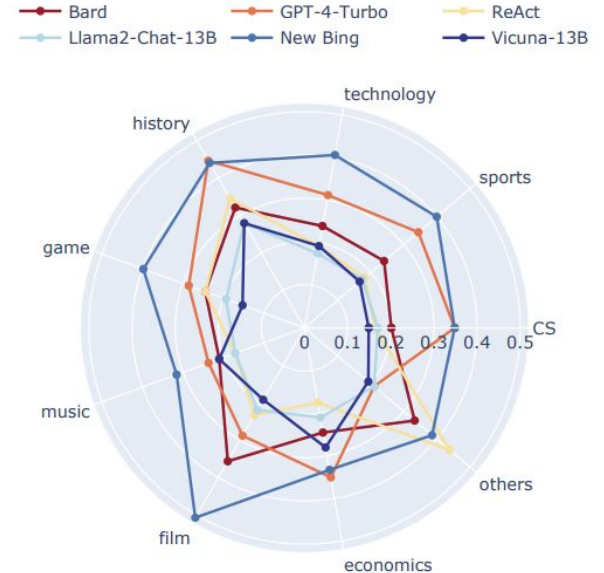


Figure 3: Performance of LLMs on different domains.

# LLM Evaluation / Continuous process

There are multiple steps in training LLM, each of them require own evaluation sets

Typically, early stage , pre-training is evaluated on standard NLP metrics such as SQuAD 2.0, SNLI, GLUE that are good to measure standard NLP tasks

In this deck, we will focus more on late stage evaluation (fine tuning and after), as they train LLM for your specific tasks and needs to be evaluated against specific tasks

# LLMs as Foundation Models : Example

Typical use of LLM models in the organization:

Many different NLP tasks

- NE/Relation/etc Extraction (search, recommendation)
- Summarization (analysis of documents, business workflows, phone calls)
- Classification (search, business workflows)
- Question Answering (customer and business facing, customer support, copilots)
- Embedding representation (search, recommendation, personalization)
- etc

Each of them is used in multiple use cases and products

# LLM : Difference in eval vs classical ML

Different fundamental capabilities of LLM to be evaluated

- Knowledge
- Reasoning
- more

Higher Risks: LLMs outputs are frequently *directly exposed to customer/business* experience (conversational experiences), risk function is high (think about recent launch announced from bigTech)

There are new types of errors due to emerging behaviours, no evaluation methods for them

*Complexity of the LLM output* makes evaluation more complex (the evaluation requires semantic matching, format, length evaluation – depending on tasks the LLM performs)

# LLM Evaluation Process

# LLM Evaluation - naturally hard

Subjectivity of the textual output for many tasks, typically low level of inter-rater agreement

Evaluation requires expertise, domain knowledge, reasoning

Answers are long, formatting and semantic matching needs to be a part of evaluation

LLM functions to evaluate are very diverse : (examples)

- Reasoning tasks, requires specific evaluation (ex ConceptArc)  
<https://arxiv.org/abs/2311.09247>
- Multi turn tasks, requires a specific evaluation (ex LMRL Gym)  
<https://arxiv.org/abs/2311.18232>
- Instruction following <https://arxiv.org/pdf/2310.07641.pdf>



# Why to evaluate LLMs?

Validation of model changes : Will a new change improve the product? Is it a good users' experience? Will the system become better?

New Product Launches: yes/no decisions, can we launch a new product based on the LLM

Continuous Improvement of the users experience <- design of the correct metrics to drive the LLM improvement in the right direction , **evaluation driven development (as test driven development in software engineering)**

# LLM Evaluation

Worst case behaviour vs Average case behaviour vs Best case: all are needed all represent various aspects of the system (many benchmark are focused on average case) - import in real world systems, typically not presented in the academic benchmarks

Risk estimates: evaluation to cover worst risks (depends on industry, regulations, public perception)

As LLMs are used for many different cases, complexity of evaluation by semantic category and type of task : semantic coverage is required (the same LLM might have very different metrics value for different topic)

LLM Evaluation is similar to evaluation of Search Engines such as Google where many metrics are needed evaluating different aspects of behavior of the evaluated object

# LLM Evaluation / Continuous process

LLM will drive many different use cases in the company

Most likely there be several LLMs within the same company due to naturally different requirements on latency, costs, serving scenarios, accuracy for specific tasks driven by different products

But each LLM will be driving many different use cases/products/experiences

Improving each LLM possibly leads to many downstream improvements means very large ROIs as LLM become mature in the company

# LLM Evaluation / Continuous process

Continuous work assumes frequent experimentation and frequent evaluation of new versions of your LLMs

To do it:

1. Evaluation process must be fast and cheap, automated (many scientists will work in parallel, frequency of experimentation must be encouraged, evaluation should not be a bottleneck )
2. Evaluation metrics must be correlated with customer and business metrics (requires work on metrics, open source benchmarks are good to bootstrap efforts, but your metrics must represent your users and your business)
3. Evaluation must cover different aspects of use of the LLM (both by tasks and by semantic category and from NLP metrics to production metrics to end-to-end system metrics)

# Necessary Requirements to the Evaluation

**Bias-Free** - no biases in the evaluation .

**Representation** - a set of measurements represents the whole process (sampling, weighting, aggregation represent the whole population) - datasets

**Alignment** (the atomic measurement of the process is aligned with custom or business quality demand for that process) - measurements

**Systems** - evaluation process does not change the system, cheap, fast, measures the intended measurement

**Statistics:** measurement are done through small sample set, fast, interpretable, confident (statistics, data science)

# LLM Evaluation Methods

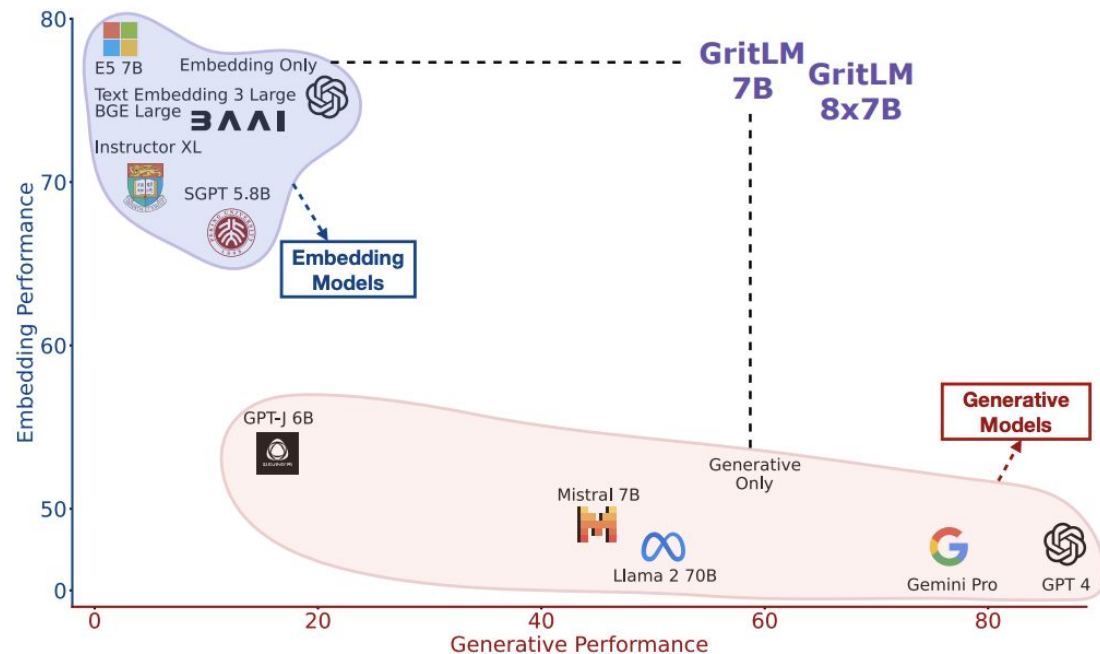
# LLM Embedding Evaluation

Embeddings generation is and will be one of the most important applications of LLM as they are needed for many high impact use cases (search, recommendations, question answering, RAG)

The problem - embedding are to used for many different customer facing tasks, for many different types of entities, for many different modalities, many different fundamental tasks (ranking, retrieval, clustering, text similarity etc)

# Embeddings models vs Generative models

From <https://arxiv.org/pdf/2402.09906.pdf>:





# LLM Embedding evaluation

MTEB ( Massive Text Embedding Benchmark) is a good example of the embedding evaluation benchmark

<https://arxiv.org/pdf/2210.07316.pdf>

<https://huggingface.co/spaces/mteb/leaderboard>

It's relatively comprehensive - 8 core embedding tasks: Bitext mining, classification, clustering, pair classification, reranking, retrieval, semantic text similarity (sts), summarization and open source

# LLM Embedding Evaluation

MTEB evaluation - Learning: no clear winner model across all tasks, the same most probably will be in your case, you'll find different model-winners for different tasks and may need to make your system multi-model

MTEB: Easy to plugin new models through a very simple API (mandatory requirement for model development)

MTEB: Easy to plugging new data set for existing tasks (mandatory requirement for model development, your items are very different from public dataset items)

# LLM Embedding Evaluation

There are standard 'traditional IR' methods to evaluate embeddings such as recall @ k, precision @ k, ndcg @ k that are easy to implement on your own and create dataset representing your data

They are even supported by ML tools such as MLFlow LLM Evaluate

Important learning: find metrics that truly matches customer expectations (for ex NDCG is very popular but it's old and it was built in different setting, one most probably needs to turn it to their system, to represent the way users interact with their system)

# LLM Embedding Evaluation

Another critical part of LLM evaluation for embedding evaluation is software performance/operations evaluation of the model

Cost, latency, throughput

In many cases, embeddings must be generate for every item on every update (ex: 100M+ updates per day), or for every query (1000000+ qps with latency limits such as 50ms)

The number of model calls and the latency/throughput requirements are different for embedding tasks rather than other LLM tasks. Most embedding tasks are high load tasks

# LLM Embedding evaluation

All traditional search evaluation data set requirements are still valid

Your evaluation set must represent users queries or documents (what you measure) with similar distribution (proper sampling for documents, query logs) , represent different topics, popular, tail queries, languages, types of queries (with proper metrics)

Queries and documents change over time, the evaluation set must reflect these changes

Take into account raters disagreement (typically high in retrieval and ranking and use techniques to diminish subjectivity (pairwise comparison, control of the number of raters etc))

# LLM Embeddings

In most cases, core tasks are serving several tasks facing customers/business. For example, text similarity might be a part of discovery/recommendation engine (text similarity of items as a one of features for the similarity of items ) or ranking (query similarity as if historical performance one of query is applicable as click signals for another query).

Evaluation is not only text similarity LLM output, but the whole end-to-end rank, recommendation etc output

# In Context Learning (ICL) Evaluation

ICL is a hard task as it requires complex reasoning from LLMs (understand the task, examples, figure how it works, apply to new tasks using only contextual clues) Hard to evaluate since it's hard to define a loss function

ICL has a practical value (code generation, fast learning of classifiers, etc )

Many data sets for ICL Eval: PIQA, HellaSwag, LAMBADA

ICL tasks typically require to solve a wide range of linguistic and reasoning phenomena,

See examples: <https://rowanzellers.com/hellaswag/>,

OpenICL Framework: <https://arxiv.org/abs/2303.02913>

# ICL - Evaluation

**PIQA** Selection from multiple answers, only one is correct (PIQA tests commonsense reasoning, physical interactions)

## a. Shape, Material, and Purpose

[Goal] Make an outdoor pillow

[Sol1] Blow into a **tin can** and tie with rubber band ✗

[Sol2] Blow into a **trash bag** and tie with rubber band ✓

[Goal] To make a hard shelled taco,

[Sol1] put seasoned beef, cheese, and lettuce **onto** the hard shell. ✗

[Sol2] put seasoned beef, cheese, and lettuce **into** the hard shell. ✓

[Goal] How do I find something I lost on the carpet?

[Sol1] Put a **solid seal** on the end of your vacuum and turn it on. ✗

[Sol2] Put a **hair net** on the end of your vacuum and turn it on. ✓

## b. Commonsense Convenience

[Goal] How to make sure all the clocks in the house are set accurately?

[Sol1] Get a solar clock for a reference and place it just outside ✗  
a window that gets lots of sun. Use a system of call and response once a month, having one person stationed at the solar clock who yells out the correct time and have another person move to each of the indoor clocks to check if they are showing the right time. Adjust as necessary.

[Sol2] Replace all wind-ups with digital clocks. That way, you set them once, and that's it. Check the batteries once a year or if you notice anything looks a little off. ✓

## LAMBADA

Predict the last word of multi sentence passage (Lambada tests: deep comprehension, context dependent language tasks )

- (10) *Context:* The battery on Logan's radio must have been on the way out. So he told himself. There was no other explanation beyond Cygan and the staff at the White House having been overrun. Lizzie opened her eyes with a flutter. They had been on the icy road for an hour without incident.

*Target sentence:* Jack was happy to do all of the -----

*Target word:* driving



# Ethical AI evaluation

Many different cases of what makes AI unethical require different evaluation metrics

Many practical case are classifiers that have to to classify the LLM output (toxicity, racism) . It's still very open area and very domain dependent

Beliefs encoded in the LLMs - detection and measurement see <https://arxiv.org/pdf/2307.14324.pdf>

Deceptive behaviour <https://arxiv.org/pdf/2308.14752.pdf>

(see chapter 4.3 as a survey on detection techniques -> evaluation )

# LLM as a judge

Human ratings - expensive, slow,

Can LLM evaluate other LLM (typically evaluated is much larger model)

Pros:

- Automated evaluation, flexible, complex, interpretable

Cons:

- Expensive (both API/computation costs and time)

- No 100% confidence in the results, they need to be verified

Ref: <https://arxiv.org/abs/2306.05685>

# LLM as a judge

Create a template grading task using LLM as a judge providing examples for grading scores

Create the first training set

Verify the quality of evaluation (by humans)

Iterate until getting a good agreement between LLM and human evaluation

Run your evaluation (either on training set, or live in production or in other scenarios).

Advantage of LLM as a judge - can be run on live traffic

Grading LLM typically is more complex than the graded LLM (the cost of evaluation can be controlled by sampling)

# LLM as judge

The LLM as a judge may agree with human grading but it requires work on designing and tuning grading prompt templates

Costs can be saved by using cheaper LLMs (but requires experimentation and tuning prompt templates and examples) or through sampling techniques

The judge model can provide interpretation and explanation assuming the right template

For alignment with human scores, important to measure interhuman alignment (this theme is frequently neglected in academic community, some tasks are inherently subjective)

# Evaluation of RAG as example of LLM App evaluation

RAG architectures will be one of the most frequent industrial pattern of LLM usage

- Correctness
- Comprehensiveness
- Readability
- Novelty/Actuality
- Quality of information
- Factual answering correctness
- Depth
- Other metrics

Similar to a traditional search engine evaluation as we evaluate a requested information but there is a substantial difference as we evaluate generate response rather than external documents

Traditional IR architecture: retrieval -> ranker, RAG architecture : retrieval -> generator, different type of evaluation

# Evaluation of RAG

A problem, comparison of generated text (answer) with the reference answer. Semantic similarity problem

Old lexical metrics (BLEU, ROUGE) are easy to compute but give little usable answers

BERTScore, BARTScore, BLEURT and other text similarity functions

Calls to external LLMs

# RAG Evaluation 3 typical metrics

Context Relevance

Answer Relevance

Groundedness (precision)

(common across multiple frameworks RAGAs, ARES, RAG Triad of metrics)

but

# Evaluation of RAG - RAGAs

Zero Shot LLM Evaluation

4 metrics:

- Faithfulness,
- Answer relevancy
- Context precision
- Context recall

Important to enhance to what represents your RAG intents for your customers

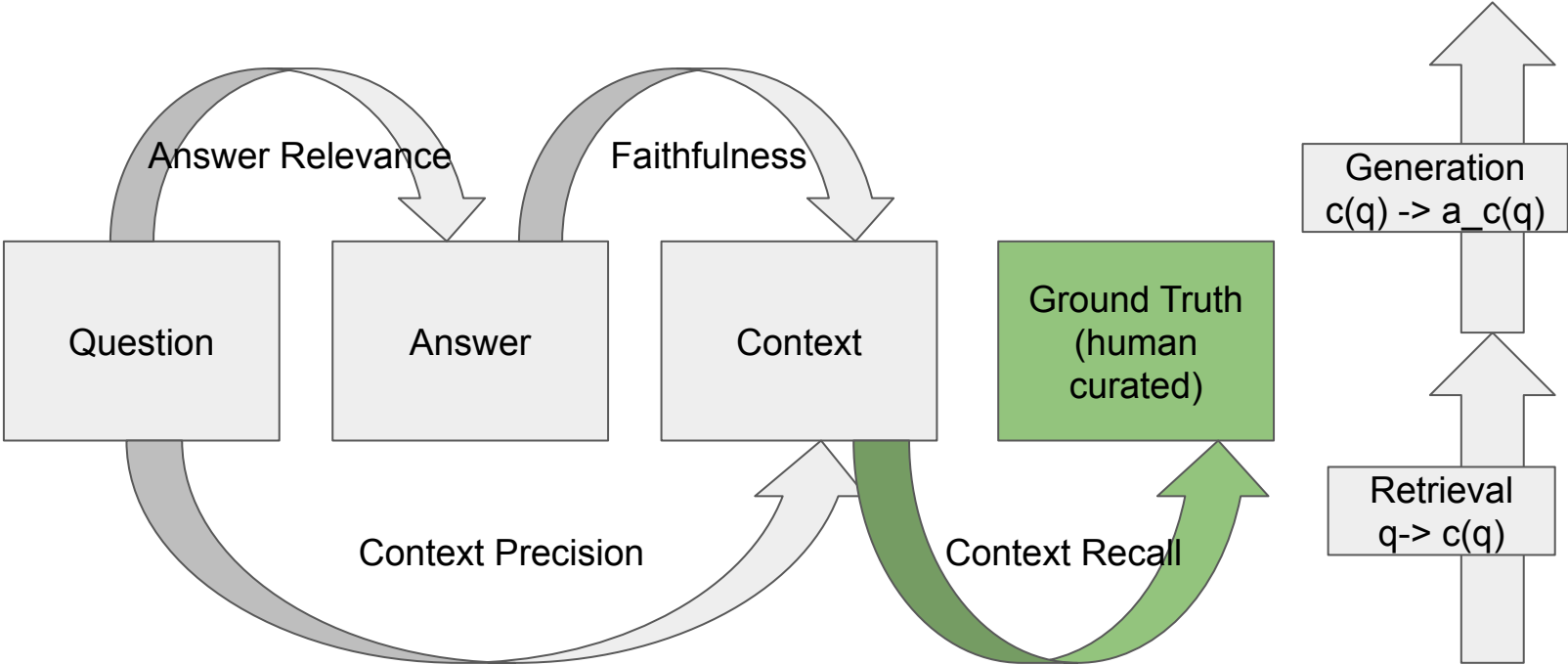
<https://arxiv.org/abs/2309.15217>

<https://docs.ragas.io/en/stable/>

RAGAs framework integrated with llamaindex and LangChain



# RAGAs



# Evaluation of RAG - RAGAs

<p><b>Faithfulness</b> consistency of the answer with the context (but no query!) Two LLM calls, get context that was used to derive to answer, check if the statement supported by the context</p>	<p><b>Context Relevance</b> how is the retrieved context “focused” on the answer , the amount of relevant info vs noise info , uses LLM to compute relevance of sentences / total number of retrieved sentences</p>
<p><b>Answer Relevancy</b> is the answer relevant to the query , LLM call, get queries that may generate answer, verify if they are similar to the original query</p>	<p><b>Context Recall</b> (ext, optional) if all relevant sentences are retrieved , assuming existence of ground_truth answer</p>

# RAGAs

<p><b>Faithfulness</b> : LLM prompt to decompose answer and context into statements F = supported statements / total statements</p>	<p><b>Context Relevance</b>: LLM Prompt to decompose contexts into sentences and evaluate if the sentences are relevant to the question CR = sentences in the context / relevant sentences</p>
<p><b>Answer Relevance</b>: LLM prompt to generate questions for the answer. For each question generate embedding. Compute the average semantic similarity score between original query and all generated queries AR = <math>1/n \sum \text{sim}(q, q_i)</math></p>	<p><b>Context Recall</b>: CR = GT sentences attributed to recall / GT sentences</p>

# Evaluation of RAGs - RAGAs

Prompts should be well tuned, hard to move to another context, or LLM, requires a lot of work on tuning of prompts

Each metrics: faithfulness, answer relevancy, context relevance, context recall can be dependent on your domain/business. It requires tuning to measure that your business depends upon

Available in open source : <https://docs.ragas.io/en/stable/> integrated with key RAG frameworks

More metrics in new version (Aspect Critique, Answer Correctness etc )

# Evaluation of RAG - ARES (Automated RAG Evaluation System)

<https://arxiv.org/abs/2311.09476>

Focused on trained few shot prompter rather than zero shot prompting

Demonstrates accurate evaluation of RAG systems using few hundred human annotations

Shows accurate evaluation despite domain shifts

Evaluates LLM for the metrics: context relevance, answer faithfulness, and answer relevance.

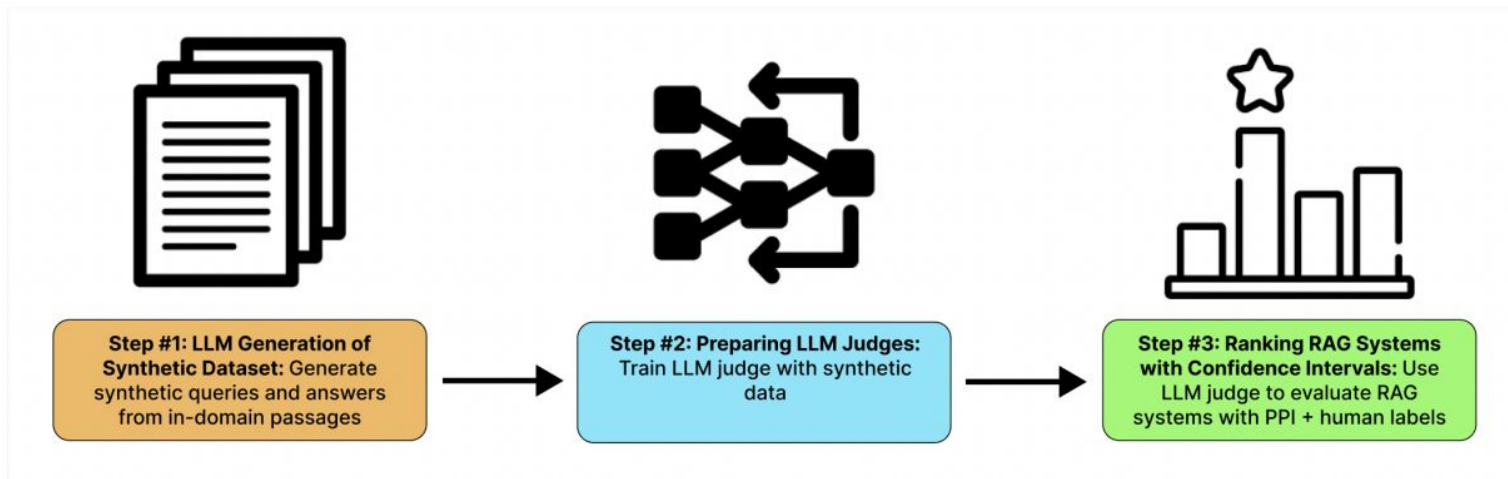
Statistical techniques to get confident answers from few hundred measurements

# ARES

## Inputs

- a set of passages from the target corpus
- a human preference validation set of 150+ annotated datapoints
- five few-shot examples of in-domain queries and answers, which are used for prompting LLMs in synthetic data generation.

# RAG Evaluation ARES



From <https://arxiv.org/pdf/2311.09476.pdf>

# ARES

- Generation of synthetic data. Generate synthetic question answers from the input corpus passages using generative LLM (including negative examples, techniques are presented in the paper)
- For Faithfulness, Answer Relevance, Context Relevance, fine tune LLMs using binary classifier as a training objective. Human preference annotation set is used to evaluate model improvement
- Sample in domain Query - Answer - Context triple and classify them using LLMs

Authors report that

- Ranking of RAG systems is more reliable than pointwise scoring of them due to noise in the synthetic data
- More accurate measures than RAGAs by leveraging domain adaptive techniques for prompting and using PPI method



# RAS and ARES have to be customized

For your particular system:

- Metrics
- Prompts
- Datasets and annotations

# Code Generating LLMs : Evaluation

Problem: the generated code is not used on its own, it's a part of code writing process managed by code.

How to evaluate generated code to be used as a part of development work

Can human recognize bugs in the generated code?

Is generated code easy to use / modify / integrate?

Problems of evaluation are similar to many other LLM evaluation scenarios

# Code Generating LLMs: HumanEval

A popular approach to evaluate code generating LLMs is HumanEval (by OpenAI), an eval set for code generation

<https://arxiv.org/abs/2107.03374>

The program is considered correct if it passed unit tests for the problem (in average 7. Unit tests for the problem)

pass@k correctness is one of the top k program passes unit tests

Not every organization will develop code generation tools, but almost every will use LLM for imperative/action applications: scheduling, routing, home automation, shopping etc and will use LLM to generate action calls

And CodeXGLUE <https://arxiv.org/abs/2102.04664>

# Coding Co-pilot Evaluation:

Co-pilot Evaluation at Microsoft <https://arxiv.org/pdf/2402.14261.pdf>

Clearly definition of core tasks of the co-Pilot

- Documentation Generation
  - Syntax correctness, format correctness
- Bug fixing
  - Syntax correctness, fix rate
- Code generation from Natural Language
  - Syntax correctness, test pass rate
- Test case Generation
  - Syntax correctness, test pass rate
- Workspace understanding and Query resolution
  - Mean reciprocal rank, end to end keyword detection

# Other domain specific LLM Evaluation tasks

Health-LLM (Health prediction based on wearable sensor data)

See <https://arxiv.org/pdf/2401.06866.pdf>

As an example of using LLM for various health prediction tasks and building LLM evaluation sets and tools to evaluate LLM performance for those tasks

Evaluation of explanation of recommendations

Chapter 4 in <https://arxiv.org/pdf/2304.10149.pdf>

Conversational LLM based experiences in healthcare: (i highly recommend this paper for understanding of evaluation of practical conversational systems)

<https://www.nature.com/articles/s41746-024-01074-z.epdf>

# Hallucination Evaluation

HaluEval <https://aclanthology.org/2023.emnlp-main.397.pdf>

3 tasks : question answering, knowledge grounded dialog, summarization

Responses from the LLM labeled by human annotators

The set is focused on understanding what hallucinations can be produced by LLM and the LLM is asked to produce to wrong answer through hallucination patterns (one pass instruction and conversation schema )

- four types of hallucination patterns for question answering (i.e., comprehension, factualness, specificity, and inference)
- three types of hallucination patterns for knowledge grounded dialogue (i.e., extrinsic-soft, extrinsic-hard, and extrinsic-grouped)
- three types of hallucination patterns for text summarization (i.e., factual, non-factual, and intrinsic)

Focus: Understanding Hallucination patterns and understanding if LLM can detect hallucinations

# Hallucination Evaluation

Search-Augmented Factuality Evaluator (SAFE). An evaluation framework from Google DeepMind and Stanford

Available in open source

- Send a request, get an answer
- Break answer into atomic facts
- Check each fact with the verifier (in their case Google Search, and it can be your system)

<https://arxiv.org/pdf/2403.18802.pdf>

# Pairwise comparison - Arenas

How to select 'the best' model out of many one? How to rank a set of models by their quality

Game: model vs model on performing a certain task (can be very flexible such as an arbitrary question from the user).

A set of evaluation tasks: at each step two models 'play' against each other, human rates the best answer, the model with the best answer 'wins' the game

The task: predict which model wins more frequently



# Arena

Chatbot arena for generic user tasks

<https://chat.lmsys.org/>

Negotiation area for negotiation tasks

<https://arxiv.org/pdf/2402.05863.pdf>

A paper from Cohere about model ranking based on tournaments

<https://arxiv.org/pdf/2311.17295.pdf>

# Arena

Such arena/tournament setting are useful for ranking many very different models vs generic tasks - open source community,

But they are applicable in your setting too

1 evaluation of many models and \*version of models\*

2 results of evaluation are easy to interpret

3 possible to modify tasks (sub domain/ sub tasks) , make results more interpretable

# Model vs Model evaluation

Comparative study is useful for incremental improvements of the same model

Model vs model evaluation

- interpretation/deeper understanding of model change,
- measures both improvement and degradation
- useful in case of small model changes

Supported by some cloud providers (Google Vertex AI AutoSxS) but in most cases requires to build your own evaluation

# Evaluation harnesses

Tasks, how many different types of tasks can harness support

Speed / Scalability, how fast is it to run evaluation tasks. Does it scale with the number of GPUs/compute power for the evaluation system, scalability by the data sets

Easy to use code base, is it easy to modify at the code level, adopt to new tasks, support of various coding scenarios, integration with other open source tools (fast inference with vLLM, hugging face transformers)

Evolvability, easy to add new models, new tasks, new data sets, new metrics

Harness are very complicated products, hard to develop on your own, open source harnesses are very valuable

# Evaluation Harnesses

EleutherAI/lm-evaluation-harness, very powerful, for zero and few short tasks

OpenAI Evals

bigcode-project/bigcode-evaluation-harness, code evaluation tasks

MLFlow LLM Evaluate , integrated with ML Flow,

MosaicML Composer, icl tasks, superfast, scaling to multi gpu

RAGAs for LLM based evaluation <https://docs.ragas.io/en/latest/>

TruLens

# Evaluation Harnesses

EleutherAI -

HuggingFace leaderboard uses it

Well integrated with many OS LLM software libraries

Supports many metrics and evaluation sets, supports third party models

Many other things (LORA adapters evaluation)

Widely used and tested (hundreds papers ), Nvidia, MosaicML, Cohere etc use it

Scales well for multi GPU

vLLM integration

Easy to create a new zero or few shot evaluation tasks

# LLM In production

LLM are used in production, in online serving, streaming, batch scenarios

Besides NLP Evaluation metrics, LLMs should support the required load and support required experience from software performance point of view

For example, one might have LLM generating embedding representing the query or supporting semantic parsing or classification of the search query with the requirement to handle 1000000+ qps and 50 millisecond latency limit (online serving) or process thousands of phone call transcripts every several minute (in batch processing) or continuously process crawled web pages (~ 1B per day, )

# LLM In production

## GPU Utilization metrics

Total latency (to produce the full output), throughput as the total number of tokens per sec generated across all users and queries

Time to produce the first token, time to produce the whole response, time per token for each user (here a lot of control points, for example, latency can be reduced by prompt producing shorter output without change of the model)

Operational metrics such as the percentage/number of 'Model is overloaded' responses,

Costs: (a lot of optimization tricks, more about systems rather than models per se)



# LLM In production End to end evaluation

User engagement metrics: engagement rate, response rate, negative response rate

Response quality: length of response, time between reading and responding, explicit quality response (thumbs up and down etc), garbage responses (by stage in the LLM funnel that typically involved other systems)

Session metrics: length, time of engagement, number of engagement per session or “Average Size Basket, return rate

Q&A

**Thank you**