Evaluation of LLMs and LLM Applications

Pragmatic approach

Andrei Lopatenko, PhD, VP Engineering

Content

Intent of this presentation

My point of view based on experience building AI systems (Search, Recommendations etc) in last 16 years

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*, rather than to apply a particular metrics invented for a particular system,

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

Intent of this presentation

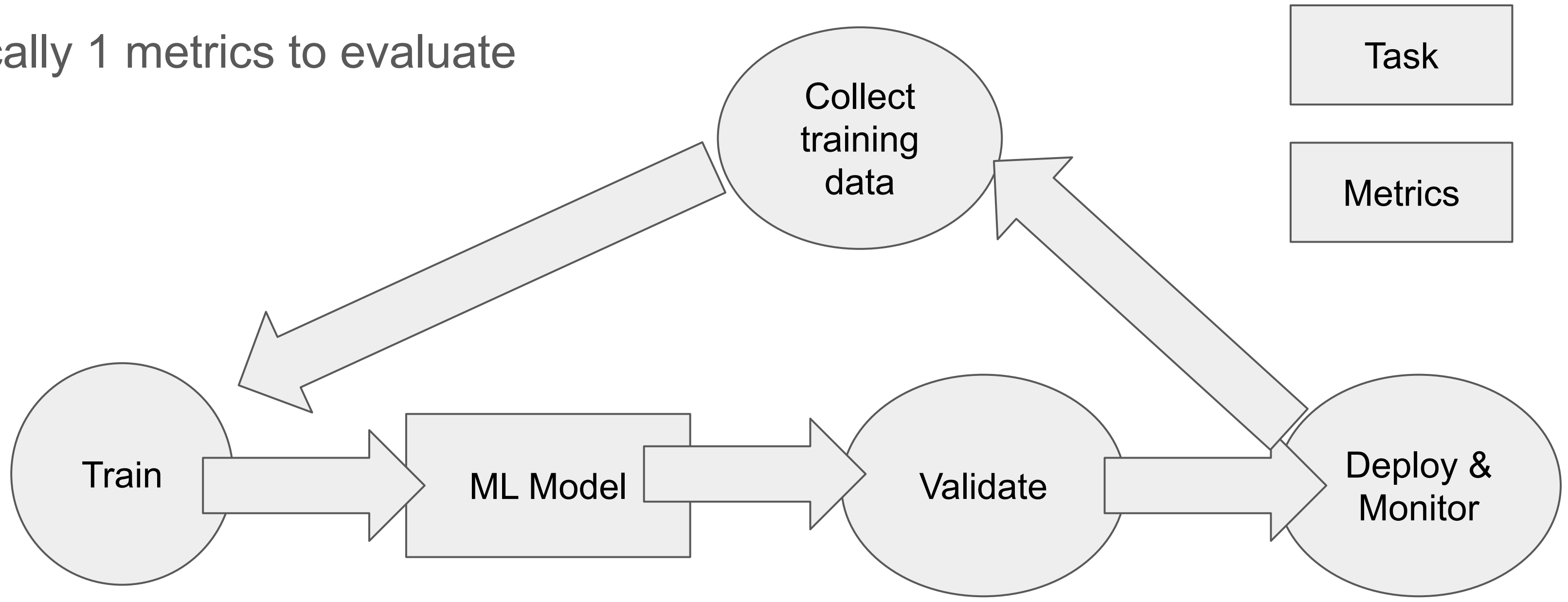
In this presentation, we talk about industrial use of LLM, developing your use cases, in most cases most companies will not do pre-training and will use open source LLM as start of their journey.

So we do not discuss SQuaD 2.0, SNLI, GLUE and other LLM evaluation methods used at the early stage of the LLM development (pre-training)

We assume, that most work on building LLM product in average company is about building adapters, fine tuning, instruction tuning and building continuous circle of improvements at the post pre-training and we will focus on the evaluation during those stages

Classical ML

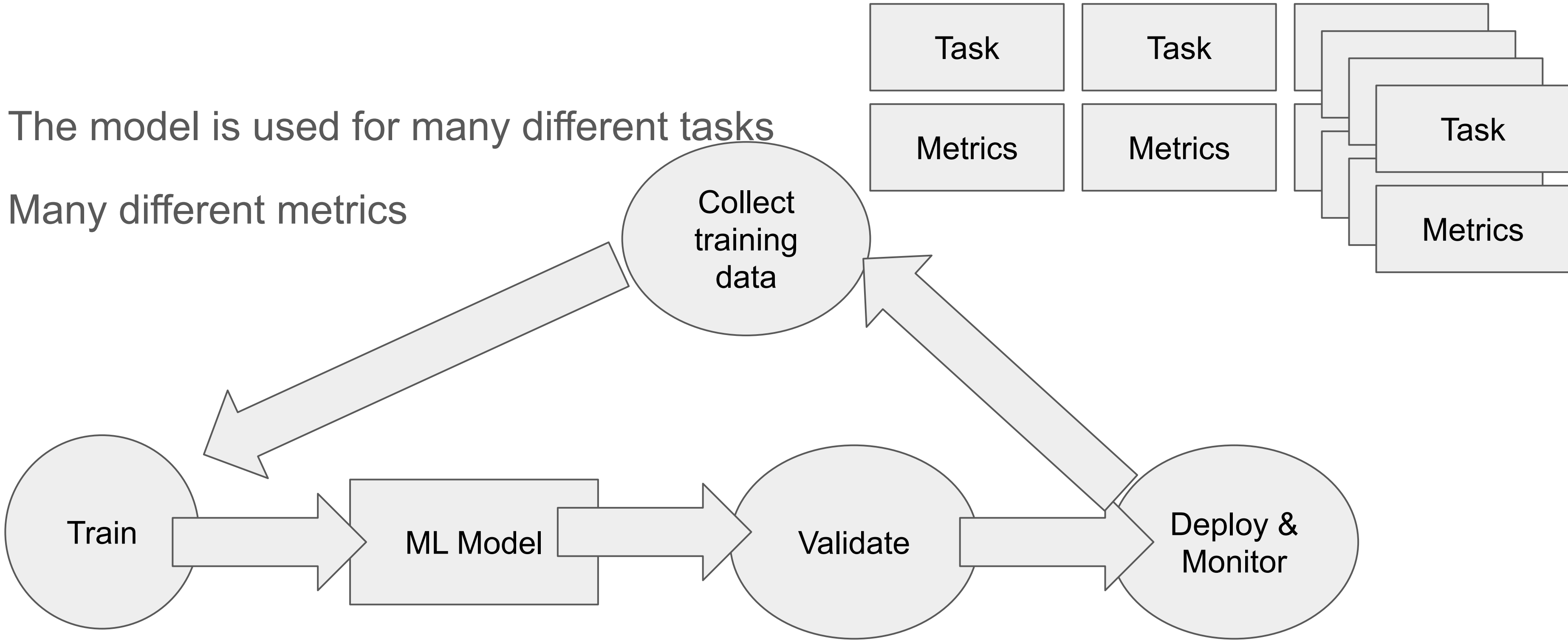
Typically 1 metrics to evaluate



Foundational Models (including LLM)

The model is used for many different tasks

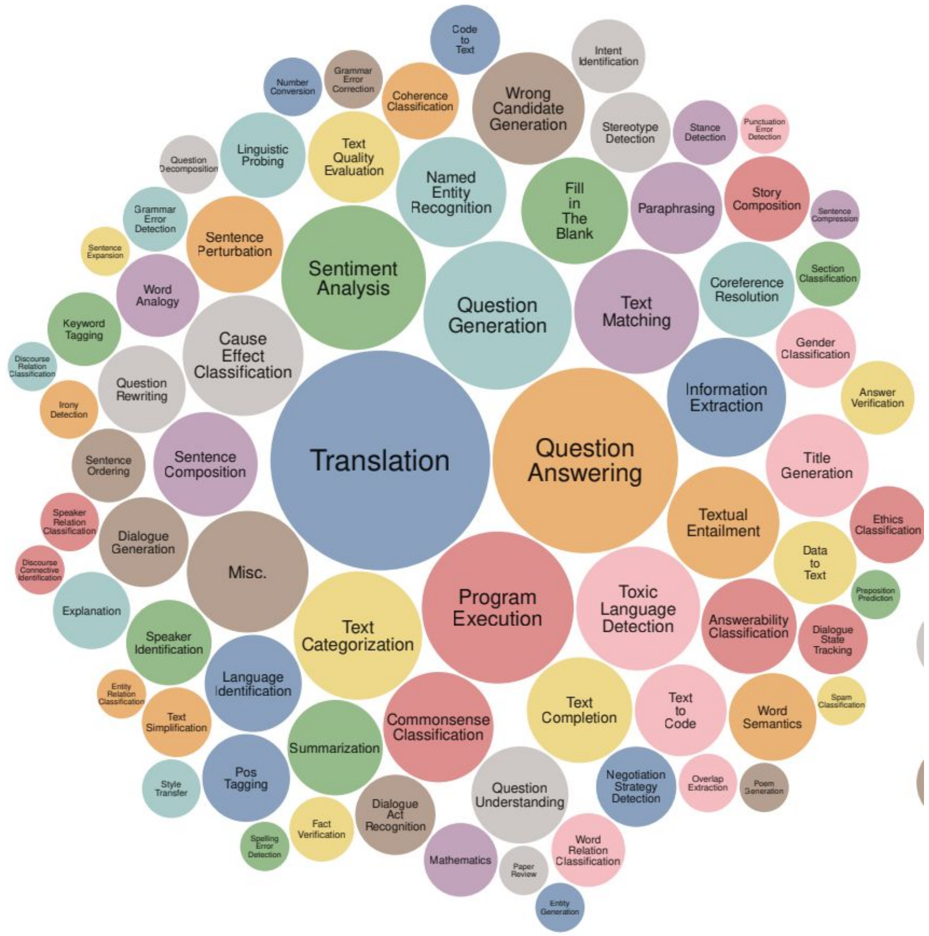
Many different metrics



LLM Tasks

Wide variety of tasks is a typical for use patterns of LLMs in every business

https://aclanthology.org/2022.emnlp-main.340.pdf



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

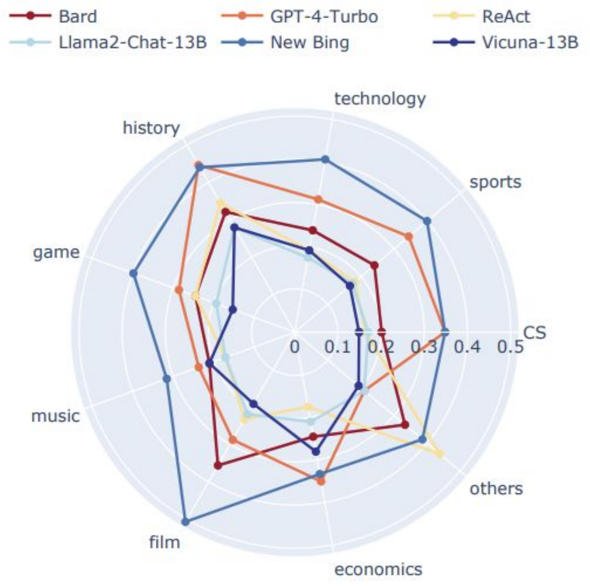


Figure 3: Performance of LLMs on different domains.

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

- Extraction
- Summarization
- Classification
- Conversational experiences

Each of them is used in multiple use cases and products

LLM : Difference in eval vs classical ML

Other differences in the evaluation

Different fundamental capabilities of LLM to be evaluated

LLM Knowledge

LLM Reasoning

Risks: LLM are frequently directly exposed to customer/business experience (conversational experiences), error price is high

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 - naturally hard

Hard to rate/subjectivity, scores are subjective.

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 :

- 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

LLM Why to Evaluate?

Validation : Is it a good users experience? Is the current system good enough?

Product Launch: 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

Risk estimates

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

Semantic coverage is required, as you may discover that your LLM behaves different for different part of your business domain (for example, home department vs electronics vs apparel in e-Commerce) and must be thoroughly tested and monitored

See an example of different of performance for various semantic categories in https://arxiv.org/pdf/2009.03300.pdf Figure 6

LLM Evaluation / Continuous process

LLM most probably will drive many different use cases in the company

(there might be several LLMs within the same company due to naturally different requirements on latency, costs, (->possible size in the number of parameters) , serving scenarios, needed accuracy or tasks) but each of them will be driving many different use cases/products/experiences

Improving each of them possibly leads to many downstream improvements means very large ROIs as LLM become mature in the company (used widely)

LLM Evaluation / Continuous process

Continuous work assumes frequent experimentation and frequent evaluation of new version of LLMs

Lesson: Eval driven development is successful way to develop AI systems – compare test driven development for development software systems

To do it:

1. Evaluation process must be fast and cheap, automated (many scientists work in parallel, frequency of experimentation must be encourage, evaluation should not be a bandwidth)
2. Evaluation metrics must be correlated with customer and business metrics (required work on metrics, open source benchmarks are good to bootstrap efforts, but their utility value is small)
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)

LLM Embedding Evaluation

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

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)

LLM Embedding evaluation

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

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 winners 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

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 (100000+ 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 requirement 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 document are change over time, the evaluation set must reflect these changes

Take into account rater 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, the core task is serving higher task. 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).

Important to understand and build evaluation of not only text similarity LLM output, but the whole end-to-end rank,recommendation etc output and rune your notion of similarity to the system

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 PIQA, HellaSwag, LAMBADA

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

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

Framework: https://arxiv.org/abs/2303.02913

Ethical AI evaluation

Very different cases , many practical case are classifiers that have to to classify the LLM output (toxicity, racism)

And such cases as beliefs encoded in the LLMs

https://arxiv.org/pdf/2307.14324.pdf

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

https://arxiv.org/pdf/2308.14752.pdf

LLM as a judge

Pros:

Automated, flexible, complex, interpretable,

Cons:

Expensive (both API/computation costs and time)

No 100% confidence in the results, need verification

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

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

Grading LLM can be more complex than the graded LLM (the cost is 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 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 but also call to external LLM.

Evaluation 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

RAGAs framework integrated with llamaindex and LangChain

Evaluation RAG - RAGAs

Faithfulness 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	**Context Relevance** 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
Answer Relevancy is the answer relevant to the query , LLM call, get queries that may generate answer, verify if they are similar to the original query	**Context Recall** (ext, optional) if all relevant sentences are retrieved , assuming existence of ground_truth answer

Evaluation of RAGs - RAGAs

Weak point

Prompt should be well tuned, hard to move to another context, LLM, required a lot of work on tuning of prompts

Evaluation of RAG - ARES

https://arxiv.org/abs/2311.09476

Focused on trained few short prompter rather than zero short 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.

RAG Evaluation ARES

Step 1 LLM Generation of synthetic data

High quality model is required, query - passage - answer triplets including negative answers

Step 2 Preparing LLM Judges (training smaller LLMs)

Step 3 Ranking RAG (with confidence intervals) (new method of prediction power inference to compute inference intervals https://arxiv.org/abs/2311.01453)

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), a popular eval set for code generation that became very popular since 2021

Before, there existed many other methods, typically based on textual similarity of code

https://arxiv.org/abs/2107.03374

https://paperswithcode.com/sota/code-generation-on-humaneval

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

Other domain specific LLM Evaluation tasks

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

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

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, zero and few short tasks

OpenAI Evals

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

MLFlow LLM evaluation for some metrics, embedding tasks

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

Ragas for LLM based evaluation https://docs.ragas.io/en/latest/

Evaluation Harnesses

EleutherAI - excellent

HF leaderboard uses it

Well integrated wirth many popular LLM software (OS)

Supports many default metrics and evaluation set , 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 short evaluation tasks

LLM In production

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

Besides NLP Evaluation metrics, it’s important if LLMs can 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 100000+ qps and 50 millisecond limit for query understanding

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,

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