Lessons from the Trenches on Reproducible Evaluation of Language Models

Stella Biderman, Hailey Schoelkopf,
Lintang Sutawika, et al.

Presenters: Steven Yuan, Frank Bai

Key Questions

- Why is it hard to evaluate LM’s fairly, consistently, and transparently?

- How can we make scientific progress despite these difficulties?

- What are the best practices for evaluating language models?

- What common infrastructure do researchers need?

The Key Problem

To tell if LM output matches a target, we need to determine semantic equivalence,

but the best tool we have to determine semantic equivalence is a LM…

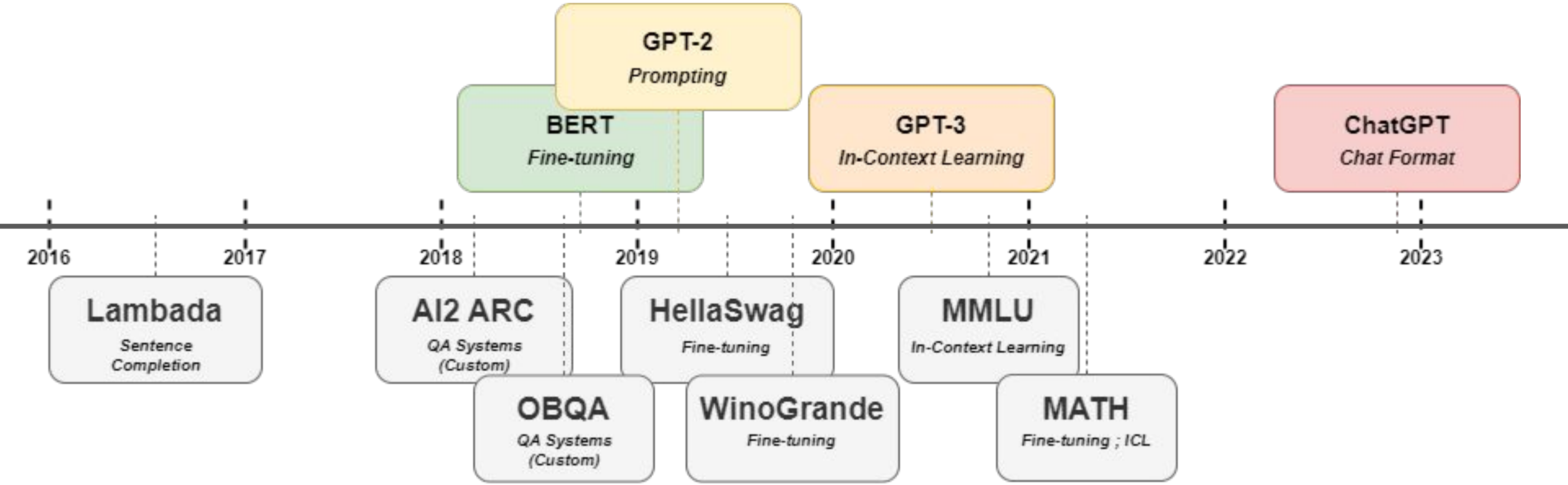
Dealing with the Key Problem

- Expert human annotators?
  - Cost-prohibitive. Doesn’t scale. Humans are biased.
- BLEU/ROUGE score?
  - Inherently flawed. Not construct valid. Implementation differences.
- Ground truth verifier
  - For code generation and mathematics.
- Re-framing as multiple choice
  - Applicable to some use cases.

Problems with Consistency

- Implementations of the same benchmark can have small differences
- Results can be very sensitive to differences in implementation details
- Re-implementing/adapting benchmarks is sometimes unavoidable
- Sometimes because they’re being adapted outside of their original paradigm

Models



Benchmarks

Problems with Fairness

Different models may expect different prompting styles.

ARC Challenge

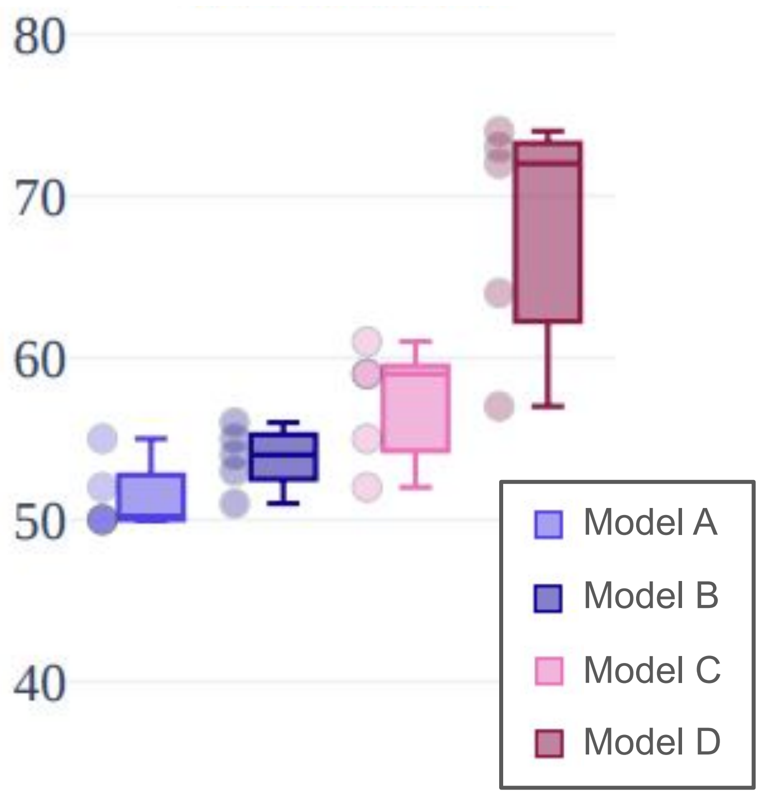
	Cloze	MMLU-style
GPT-NeoX-20B	$38.0 \pm 2.78\%$	$26.6 \pm 2.53\%$
Llama-2-7B	$43.5 \pm 2.84\%$	$42.8 \pm 2.83\%$
Falcon-7B	$40.2 \pm 2.81\%$	$25.9 \pm 2.51\%$
Mistral-7B	$50.1 \pm 2.86\%$	$72.4 \pm 2.56\%$
Mixtral-8x7B	$56.7 \pm 2.84\%$	$81.3 \pm 2.23\%$

Multi-Prompt Evaluation

Any single prompting style can be biased.

Solution: Perform the same benchmark with many prompt styles and compare models by their accuracy distributions w.r.t. style.

Distribution of Model Accuracy Across Prompts



Problems with Transparency

- Some researchers in industry labs don’t release the models
- Some models are only available through an API or chatbot interface
- APIs may non-transparently modify the model or become deprecated
- Chatbots have additional layers of product features that add complications
- Access to proprietary models can be expensive

Best Practices for LM Evaluation

- Always share your code and exact prompts
- Always provide model outputs and artifacts
- Do not compare against results from other works without reproducing them
- Do statistical significance testing
- Perform qualitative analyses

The Language Model Evaluation Harness

Motivation

- Centralizes evaluation tasks & reduces duplication
- Ensures consistent prompts & metrics
- Eases reproducibility across different models

Core Design Philosophy

- Orchestration Problem: Single codebase to evaluate **any** benchmark on **any** model
- Modularity: Separate “Tasks” (benchmarks) from “LM” (model interface)
- Focus on Best Practices: Automatic logging, versioning, standard error reporting

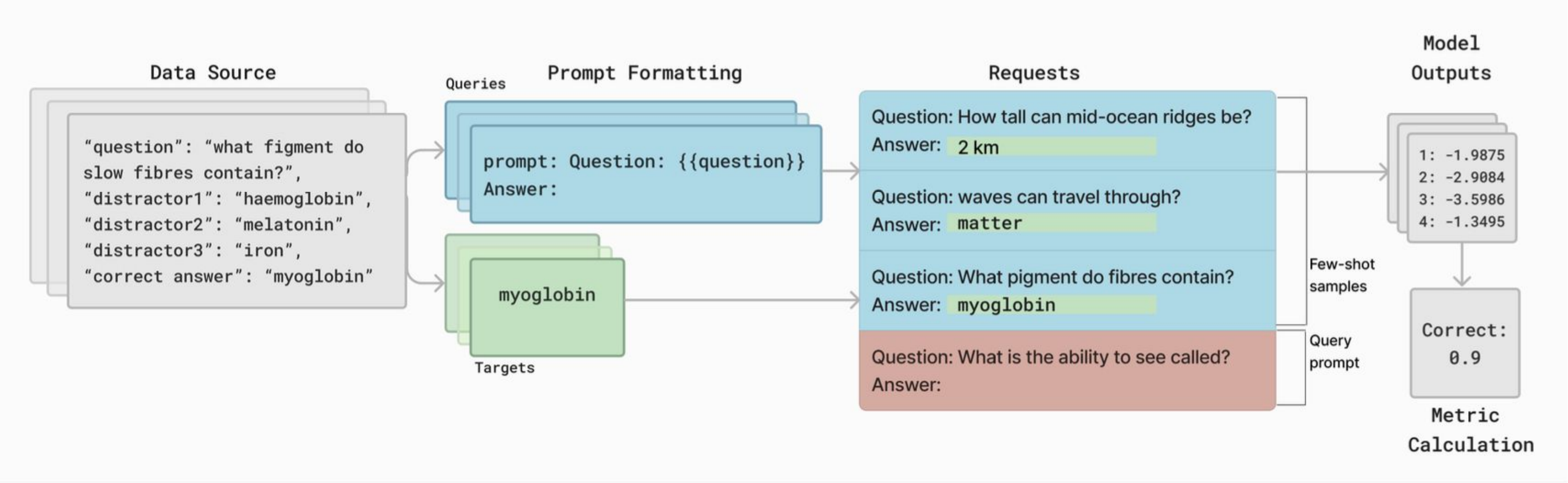
Tasks Overview

- Implemented via a standardized `Task` class

- YAML or Python subclass for flexible setup

- Common methods: data loading, prompt formatting, metric computation

Tasks Overview



The LM Interface

- Three main “Request” types:
    - Loglikelihood (multiple choice)
    - Rolling Loglikelihood (perplexity)
    - Generation (free text)

- Tokenization is abstracted away

- Supports flexible model backends

Handling Minor Implementation Details

- Small prompt differences can alter scores significantly
- Harness ensures identical formatting, tokenization rules
- Task versioning records changes over time

Broader Impact

- Encourages better reporting (code, prompts, outputs)
- Aids new benchmark development and adoption
- Empowers deeper analysis of LLM behaviors

Limitations

- Focus on Implementation Consistency
- Benchmark Validity
- Resource & Cost Barriers
- Closed-Source Model Constraints
- Ongoing Rapid Evolution


FIN