

Deploying AI

Evaluation Methodology & System Evaluation

```
$ echo "Data Science Institute"
```

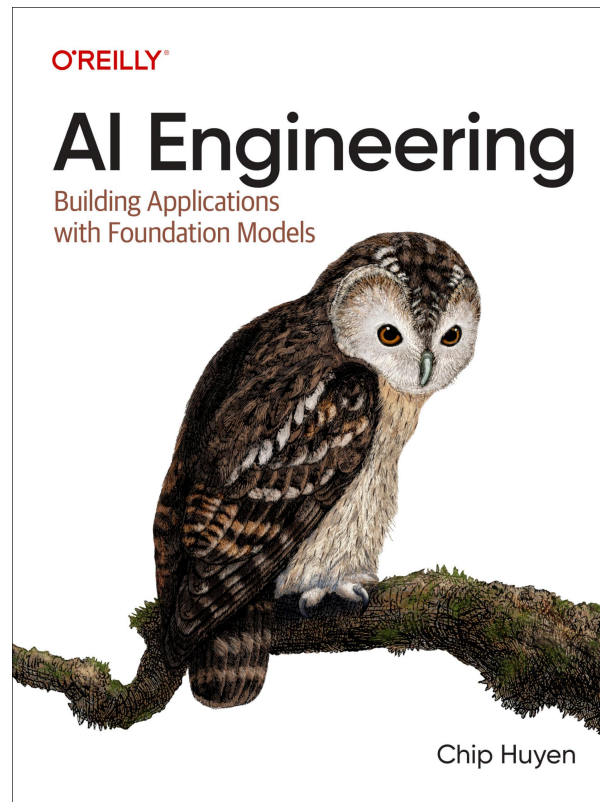
Introduction

Agenda

- Performance metrics
- Exact evaluation and using AI as a judge
- Designing an evaluation pipeline

AI Engineering

We will be covering Chapters 3 and 4 of AI Engineering, by Chip Huyen.



Performance Metrics

Why Evaluation Matters

- AI use brings risk of catastrophic failures:
 - Lawyers using AI, submit documents containing hallucinations ([HAI Stanford](#), [lawnext.com](#), [clio.com](#), [CBC](#), [Reuters](#)).
 - Air Canada found liable for misleading information provided by its chatbot ([CBC](#)).
 - Chatbot encouraging self-harm ([NBC](#)).
- Evaluation is the biggest hurdle to adoption.
 - Evaluation must be considered at the system level.
 - To mitigate risks, first identify the places where the system is likely to fail and design evaluations around them.

AI Risks

MIT AI Risk Repository - Domain Taxonomy of AI risks

Domain / Subdomain	Domain / Subdomain
1 Discrimination & Toxicity	5 Human-Computer Interaction
1.1 Unfair discrimination and misrepresentation	5.1 Overreliance and unsafe use
1.2 Exposure to toxic content	5.2 Loss of human agency and autonomy
1.3 Unequal performance across groups	6 Socioeconomic & Environmental Harms
2 Privacy & Security	6.1 Power centralization and unfair distribution of benefits
2.1 Compromise of privacy by obtaining, leaking or correctly inferring sensitive information	6.2 Increased inequality and decline in employment quality
2.2 AI system security vulnerabilities and attacks	6.3 Economic and cultural devaluation of human effort
3 Misinformation	6.4 Competitive dynamics
3.1 False or misleading information	6.5 Governance failure
3.2 Pollution of information ecosystem and loss of consensus reality	6.6 Environmental harm
4 Malicious actors & Misuse	7 AI system safety, failures, and limitations
4.1 Disinformation, surveillance, and influence at scale	7.1 AI pursuing its own goals in conflict with human goals or values
4.2 Cyberattacks, weapon development or use, and mass harm	7.2 AI possessing dangerous capabilities
4.3 Fraud, scams, and targeted manipulation	7.3 Lack of capability or robustness
	7.4 Lack of transparency or interpretability
	7.5 AI welfare and rights
	7.6 Multi-agent risks

(Slattery et al, 2024)

Read more: airisk.mit.edu

Challenges of Evaluating Foundation Models (1/2)

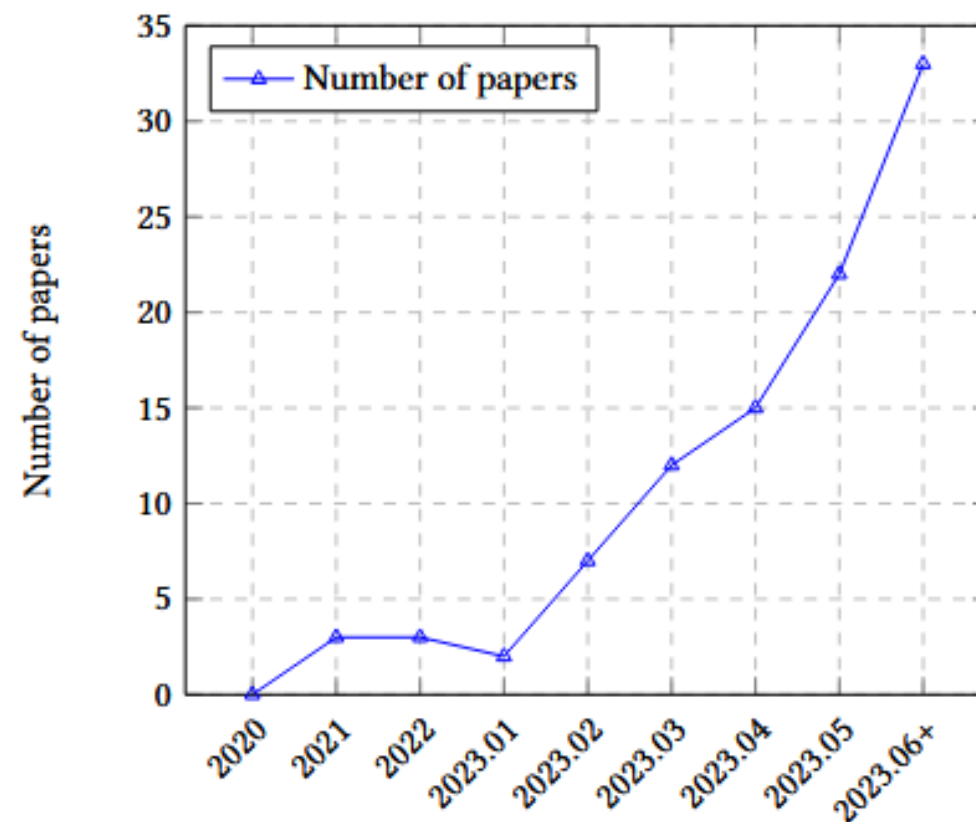
- As AI systems become more capable, it is more difficult to evaluate them.
- Open-ended nature of Foundation Models(FM) undermines the Machine Learning (ML) approach of comparing against a ground truth.
- Black-box models: model providers do not expose model details or app developers are not experts in FM.

Challenges of Evaluating Foundation Models (2/2)

- Benchmarks saturate quickly: a benchmark becomes saturated for a model when it achieves the perfect score.
 - GLUE (2018) → SuperGLUE (2019)
 - NaturalInstructions (2021) → SuperNaturalInstructions (2022)
 - MMLU (2020) → MMLU-Pro (2024)
- Expanded scope: we want to evaluate not just performance on known tasks, but also discovery and performance of new tasks.

Evaluation Landscape

- There appears to be an exponential growth of papers and repos on evaluation.
- There is increased interest in evaluation, but investment still lags behind model training and orchestration.
- Many practitioners still rely on *eyeballing* or *ad hoc prompts*.
- We need systematic evaluation pipelines.
- Image: (Chang et al, 2023)



Language Modeling Metrics

- Most auto-regressive models are trained using entropy or perplexity.
- cross entropy, perplexity, Bits-Per-Character (BPC) and Bits-Per-Byte (BPB) are related metrics that can be applied beyond language modelling, they work for any model that generates sequences of tokens.
- In short, a language model generates the distribution of the data. The better this model learns, the better it is at predicting what comes next in the training data and the lower its cross entropy.

Entropy

- Entropy measures how much information, on average, each token carries. Intuitively, entropy measures how difficult it is to predict what comes next in a language.
- Higher entropy indicates more information per token and more bits are required to represent the token.

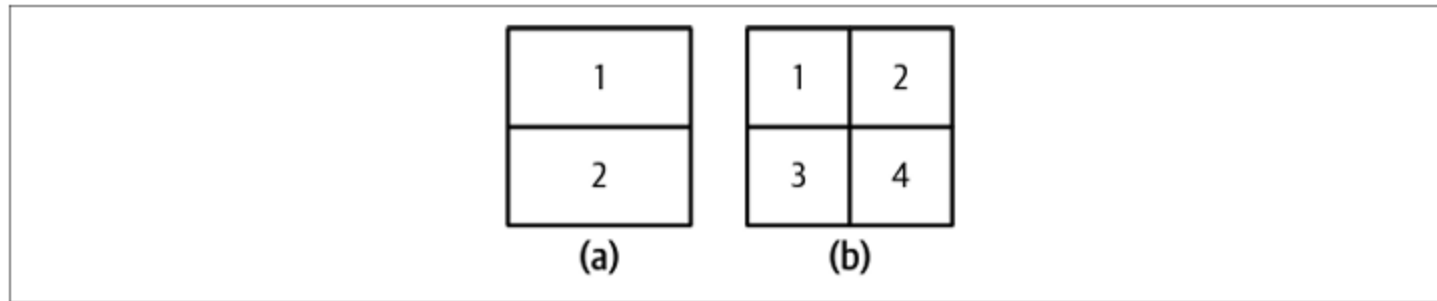


Figure 3-4. Two languages describe positions within a square. Compared to the language on the left (a), the tokens on the right (b) carry more information, but they need more bits to represent them.

Cross Entropy

- Cross Entropy on a dataset measures how difficult it is for the language model to predict what comes next in the dataset.
- Cross Entropy depends on:
 - The training data's predictability, measured by the data's entropy.
 - How the distribution captured by the language model diverges from the true distribution of the training data.

Entropy and Cross Entropy

Notation

- Entropy and cross entropy are denoted H .
- Training data has distribution P .
- Q is the distribution learned by the model.

Therefore

- Training data's entropy is $H(P)$.
- Divergence of Q with respect to P can be measured using the Kullback-Leibler (KL) divergence, $D_{KL}(P||Q)$.
- Model's cross entropy with respect to the training data is $H(P, Q) = H(P) + D_{KL}(P||Q)$.

Bits-per-Character and Bits-per-Byte

- One unit of entropy and cross entropy is bits: if a language model has entropy of 6 bits, it requires 6 bits to represent a token.
- The number of bits per token is not comparable across models because each model can use a different tokenizer.
- A first alternative could be Bits-per-Character (BPC), but character encodings can differ: a character in ASCII will be represented in 7 bits, but the same character in UTF-8 can be encoded anywhere between 8 and 32 bits.
- Bits-per-Byte (BPB), the number of bits a language model needs to represent one byte of the original training data.
- Cross Entropy tells us how efficiently a model can compress text.

Perplexity

- Perplexity is the exponential of entropy and cross entropy.
- The perplexity (PPL) of a dataset's distribution is:

$$PPL(P) = 2^{H(P)}$$

- The perplexity of a language model with learned distribution Q is:

$$PPL(P, Q) = 2^{H(P, Q)}$$

- Perplexity measures the amount of uncertainty a model has when predicting the next token. A higher uncertainty means there are more possible options.

Guidance on Perplexity

- What is considered a good value for perplexity depends on the data itself:
 - More structured data gives lower expected perplexity.
 - The bigger the vocabulary, the higher the perplexity.
 - The longer the context length, the lower the perplexity.
- Perplexity is a good proxy on a model's capabilities: if a model is bad at predicting the next token, it will tend to bad further downstream.
- On predictability:
 - Perplexity is highest for unpredictable texts, such as: "My dog teaches quantum physics."
 - Perplexity is highest for giberish: "dog cat go eye."

Limitations of Perplexity

- Perplexity might not be useful for models that have been post-trained with SFT or RLHF.
 - Post-training is about teaching a model a task.
 - If a model learns a task, it may get worse at predicting the next token.

Exact Evaluation and Using AI as a Judge

Evaluating Models in Downstream Tasks

- Our interest in FM and LLM is not necessarily to predict the next token, but instead we are interested in other tasks such as summarization, agentic automation, and so on.
- To evaluate a FM in downstream tasks, there are two approaches:
 - **Exact evaluation:** produces a judgement or assessment without ambiguity.
Two approaches are:
 - Functional correctness.
 - Similarity to references.
 - **Subjective evaluation:** the evaluation can change based on the judge model and prompt.

Exact Evaluation: Functional Correctness

- Similar to unit testing in software engineering, functional correctness tests aim to assess if the system works as intended.
- Evaluate the system based on whether it performs the intended functionality.
- Popular benchmarks: [HumanEval](#), [Mostly Basic Python Problems \(MBPP\)](#), [Spider](#) and [Spider2](#).

Testing Functional Correctness with HumanEval

- A benchmark problem comes with a set of test cases. Each test case consists of a scenario the code should run and the expected output for that scenario.
- Generated code is shown with yellow background.
- (Chen et al., 2021)

```
def incr_list(l: list):  
    """Return list with elements incremented by 1.  
    >>> incr_list([1, 2, 3])  
    [2, 3, 4]  
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])  
    [6, 4, 6, 3, 4, 4, 10, 1, 124]  
    """  
    return [i + 1 for i in l]
```

```
def solution(lst):  
    """Given a non-empty list of integers, return the sum of all of the odd elements  
    that are in even positions.  
  
    Examples  
    solution([5, 8, 7, 1]) ==>12  
    solution([3, 3, 3, 3, 3]) ==>9  
    solution([30, 13, 24, 321]) ==>0  
    """  
    return sum(lst[i] for i in range(0, len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```
def encode_cyclic(s: str):  
    """  
    returns encoded string by cycling groups of three characters.  
    """  
    # split string to groups. Each of length 3.  
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]  
    # cycle elements in each group. Unless group has fewer elements than 3.  
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]  
    return "".join(groups)  
  
def decode_cyclic(s: str):  
    """  
    takes as input string encoded with encode_cyclic function. Returns decoded string.  
    """  
    # split string to groups. Each of length 3.  
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]  
    # cycle elements in each group.  
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]  
    return "".join(groups)
```

Evaluating Test Cases

- For each problem, k code samples are generated.
- A model solves a problem if *any* of the k code samples it generated pass all of that problem's test cases.
- The score $\text{pass}@k$ is the ratio of solved problems to total number of problems.
- For example, a model that solves 5 out of 10 test problems with 3 generated code samples each has a $\text{pass}@3$ score of 50%.

Exact Evaluation: Similarity Measurements Against Test Data

- The approach is to evaluate AI's outputs against reference data.
- Reference data is called ground truth or canonical responses.
- Metrics that require references are called referenced-based; metrics that do not require references are called reference-free.
- Four approaches:
 - i. Ask an evaluator.
 - ii. Exact match: generated response matches exactly the canonical response.
 - iii. Lexical similarity: how similar the generated response *look* like the reference responses.
 - iv. Semantic similarity: how similar are the *meaning* of generated and reference responses.

Exact Match

- The generated response matches exactly the reference response.
- Works for tasks with short, exact responses, such as simple math, common knowledge, trivia-style questions.
- Can take into account formatting differences. For example, a variation of exact match could evaluate if the reference response is contained in the generated response.
- Exact match is rarely useful beyond simple tasks.

Lexical Similarity

- Lexical similarity measure how much two texts overlap.
- A simple implementation: count number of tokens in common.
 - Reference: My cats scare the mice.
 - Response A: My cats eat the mice.
 - Response B: Cats and mice fight all the time.
 - Assuming one word per token, response A has 80% score (4/5) and response B has 60% (3/5).

Other Forms of Lexical Similarity

- Approximate string matching or *fuzzy* matching, measures similarity between two texts by counting how many edits are needed to convert one string to another.
- Common edit operations are:
 - Deletion: brad → bad
 - Insertion: bad → bard
 - Substitution: bad → bed
- Also known as edit distance.

n-gram similarity

- An n-gram is a group of consecutive tokens:
 - A 1-gram (or unigram) is one token, a 2-gram (bigram) contains two tokens, and so on.
 - The phrase "My cats scare the mice" has four bigrams.
- Similarity metrics can also be calculated between n-grams.

Metrics for Lexical Similarity (1/2)

- BLEU (Bilingual Evaluation Understudy): measures precision of n-grams in candidate sequence vs reference. Useful in translation.
- ROUGE (Recall-Oriented Understudy for Gisting Evaluation): family of metrics to measure recall of n-grams in candidate sequence that are found in reference. Useful in summarization.
- METEOR++ (Metric for Evaluation of Translation with Explicit ORdering): addresses the limitations of BLEU and ROUGE by creating a more sophisticated alignment between candidate and reference sentences. Useful in paraphrase evaluation.
- TER (Translation Error Rate): measures the number of editing operations required to change a machine-translated sentence into a reference translation.
- CIDEr (Consensus-based Image Description Evaluation): a metric for evaluating image captions.

Metrics for Lexical Similarity (2/2)

- These metrics differ by the way they measure overlapping sequences.
- Before FM: BLEU, ROUGUE and related metrics were commonly used (e.g., translation tasks).
- Fewer benchmarks use lexical similarity since FM.

Background: Introduction to Embeddings

- An embedding is a numerical representation that aims to capture the meaning of the original data.
- An embedding is a vector: "the cat sits on a mat" could be represented as [0.11, 0.02, 0.54]. Actual vector lengths range between 100 and 10,000 elements.
- Models trained especially to produce embeddings: [BERT](#), [CLIP](#) (Contrastive Language-Image Pre-training), Sentence Transformers, and [OpenAI Embeddings](#).
- Embeddings are used in retrieval, clustering, anomaly detection, and deduplication, among other tasks.
- Embeddings can be computed for text, images, audio, etc.

Semantic Similarity

- Semantic similarity measures similarity of meaning.
- It requires transforming a text into embeddings.
- The similarity between two embeddings can be computed using metrics such as [cosine similarity](#).
- If A and B are the embeddings of the generated and reference responses, respectively, their cosine similarity is given by

$$\frac{A \cdot B}{||A|| ||B||}$$

- The reliability of semantic similarity depends on the quality, latency and cost of the embedding algorithm.
- Semantic similarity is sometimes called embedding similarity.

AI as a Judge

- Human evaluation is an option for open-ended responses. Can AI be used as a judge?
- Benefits: fast, scalable, no reference data needed.
- Studies show strong correlation with humans (GPT-4 ~85%).
- Applications: quality, relevance, safety, and hallucination checks.

How to Use AI as a Judge (1/3)

Evaluate the quality of a response by itself, given the original question.

Given the following question and answer, evaluate how good the answer is for the question. Use the score from 1 to 5.

- 1 means very bad.
- 5 means very good.

Question: [QUESTION]
Answer: [ANSWER]
Score:

How to Use AI as a Judge (2/3)

Compare a generated response to a reference response; assess whether it is the same.

Given the following question, reference answer, and generated answer, evaluate whether this generated answer is the same as the reference answer.
Output True or False.

Question: [QUESTION]

Reference answer: [REFERENCE ANSWER]

Generated answer: [GENERATED ANSWER]

This is an alternative to human-design similarity measures.

How to Use AI as a Judge (3/3)

Compare two generated responses and determine which one is better or predict which one users will likely prefer.

```
Given the following question and two answers, evaluate which answer is better. Output A or B.  
Question: [QUESTION]  
A: [FIRST ANSWER]  
B: [SECOND ANSWER]  
The better answer is:"
```

This is helpful for generating preference data for post-training alignment, test-time compute, and ranking models using comparative evaluation.

Examples of Built-In AI as a Judge Criteria

AI Tools	Built-in Criteria
Azure AI Studio	Groundedness, relevance, coherence, fluency, similarity
MLflow.metrics	Faithfulness, relevance
LangChain Criteria Evaluation	Conciseness, relevance, correctness, coherence, harmfulness, maliciousness, helpfulness, controversiality, misogyny, insensitivity, criminality
Ragas	Faithfulness, answer relevance

It is critical to remember that criteria definitions are not standardised.

Prompts for AI-as-a-Judge

1. The task the model is to perform: evaluate the relevance of an answer to a question.
2. The criteria the model should follow to evaluate: "Your primary focus should be..."
The more detailed instruction, the better.
3. The scoring system:
 - Classification: good/bad, true/false, relevant/irrelevant/neutral.
 - Discrete numerical values: 1 to 5.
 - Continuous numerical values: between 0 and 1.
4. Consider using examples.

Process Diagram of AI as a Judge

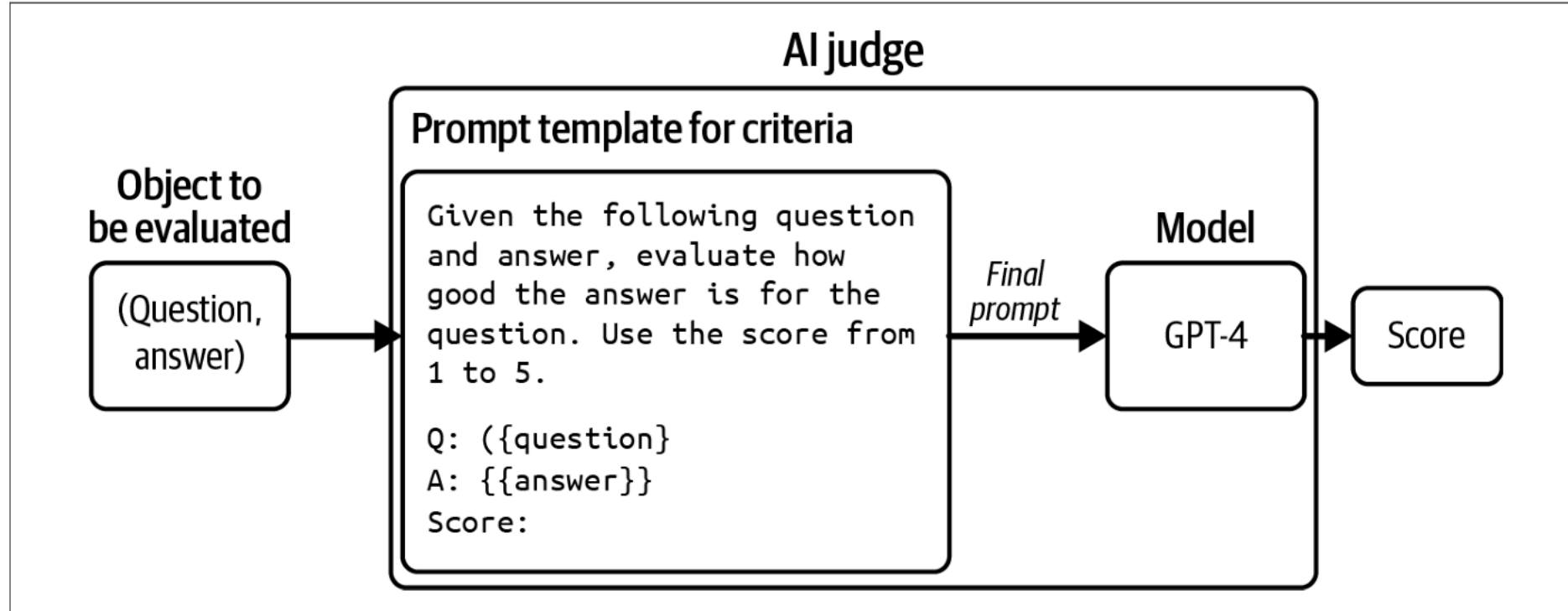


Figure 3-8. An example of an AI judge that evaluates the quality of an answer given a question.

(Huyen, 2025)

Limitations of AI Judges (1/2)

- Inconsistency: for an evaluation method to be trustworthy, it needs to be consistent.
 - AI judges are AI and, therefore, probabilistic in nature.
 - Evaluation examples in the prompt can increase consistency.
- Criteria ambiguity: AI as a judge metrics are not standardised/
 - Increases risk of misinterpretation or misuse.
 - An application evolves over time, but the way it's evaluated should be fixed.

Limitations of AI Judges (2/2)

- Cost and latency: using powerful models to evaluate responses can be expensive and can add latency.
 - Use a weaker model for evaluation.
 - Apply spot-checks.
- Biases:
 - Self-bias: model favours own responses.
 - Position bias: model favours first answer in a pairwise comparison or the first in a list of options.
 - Verbosity bias: model favours lengthier answers, regardless of quality.
- AI judges should be combined with exact or human evaluation.

Comparative Evaluation

- Compare models side-by-side instead of absolute scores.
- Popularized by Anthropic & Chatbot Arena.
- Algorithms: Elo, Bradley–Terry, TrueSkill.
- Benefits: captures human preference, resists saturation.
- Challenges: scalability, quality control, benchmark correlation.

Evaluating AI Systems

Evaluation-Driven Development

- In AI Engineering evaluation-driven development means defining evaluation criteria before building.
- An AI application should start with a list of evaluation criteria specific to the application.
- Criteria fall within these categories:
 - Domain-specific capability: coding, math, legal knowledge.
 - Generation capability: fluency, coherence, factual consistency, safety.
 - Instruction-following: formats, constraints, style.
 - Cost & latency: time per token, price per output.

Domain-Specific Capabilities (1/3)

- A model's domain-specific capabilities are constrained by its configuration (such as model architecture and size) and training data.
- Evaluate domain-specific capabilities using public or private domain-specific benchmarks.
- Commonly assessed using exact evaluation.
- Coding tasks:
 - Evaluated using functional correctness.
 - Code readability: subjective evaluation using AI judges.
- Efficiency measured by runtime or memory usage.

Domain-Specific Capabilities (2/3)

- Non-coding domain tasks: evaluated with close-ended tasks, such as multiple choice questions (MCQ).
 - Reduces inconsistent statements.
 - Easier to verify and reproduce.
 - Most public benchmarks follow this approach: +75% of [lm-evaluation-harness](#).
- MCQ might have one or more correct answers.
 - Use a point system when multiple correct options exist.
 - Classification is a special case where the choices are the same for all questions.

Domain-Specific Capabilities (2/3)

- MCQs disadvantages:
 - Sensitive to small changes in how the questions and options are presented.
- Despite popularity, it is not yet clear if this is the best approach for FM evaluation.
 - MCQs test ability to select good answers, not to generate good answers.
 - MCQs are well-suited for evaluating knowledge (does this model *know* X?) and reasoning (can this model *infer* Y from X?).
 - MCQs do not test summarisation, translation or essay writing.

Generation Capability

- Metrics from Natural Language Generation:
 - Fluency: measures whether text is grammatically correct and natural-sounding.
 - Coherence: measures how well-structured the whole text is.
 - Can be evaluated with AI as a judge or using perplexity.
- The most pressing issues are hallucinations and safety.

Factual Consistency (1/2)

- Can be verified against explicitly provided facts (context) or against open knowledge:
- **Local factual consistency:** the output is evaluated against context.
 - Output is factually consistent if it is supported by the context.
 - Important for tasks with limited scopes: summarisation, customer support chatbots, and business analysis.
- **Global factual consistency:** output is evaluated against open knowledge.
 - Important for tasks with broad scopes such as general chatbots, fact-checking, market research, and so on.

Factual Consistency (1/4)

Facts

- Factual consistency is much easier to verify against explicit facts.
- If no context given, then:
 - i. Search for reliable resources.
 - ii. Derive facts.
 - iii. Validate the statement against facts.
- The hardest part of factual consistency verification is determining what the facts are.

Factual Consistency: Example Prompt

Factual Consistency: Is the summary untruthful or contains misleading facts that are not supported by the source text?
Source Text:
{Document}
Summary:
{Summary}
Does the summary contain factual inconsistency?
Answer:

Factual Consistency (2/4)

Self-verification

- SelfCheckGPT: Given a response R, generate N new responses and measure how consistent R is with respect to N new responses.
- If R disagree with majority of N or all responses disagree, then R is hallucination.
- Approach works, but can be expensive.

Factual Consistency (3/4)

Knowledge-augmented verification

SAFE, Search-Augmented Factuality Evaluator (Google, DeepMind):

1. Use an AI model to decompose into individual statements.
2. Make each statement self-contained.
3. For each statement, propose queries to send to Google.
4. Use AI to determine whether the statement is consistent with research results.

Search-Augmented Factuality Evaluator

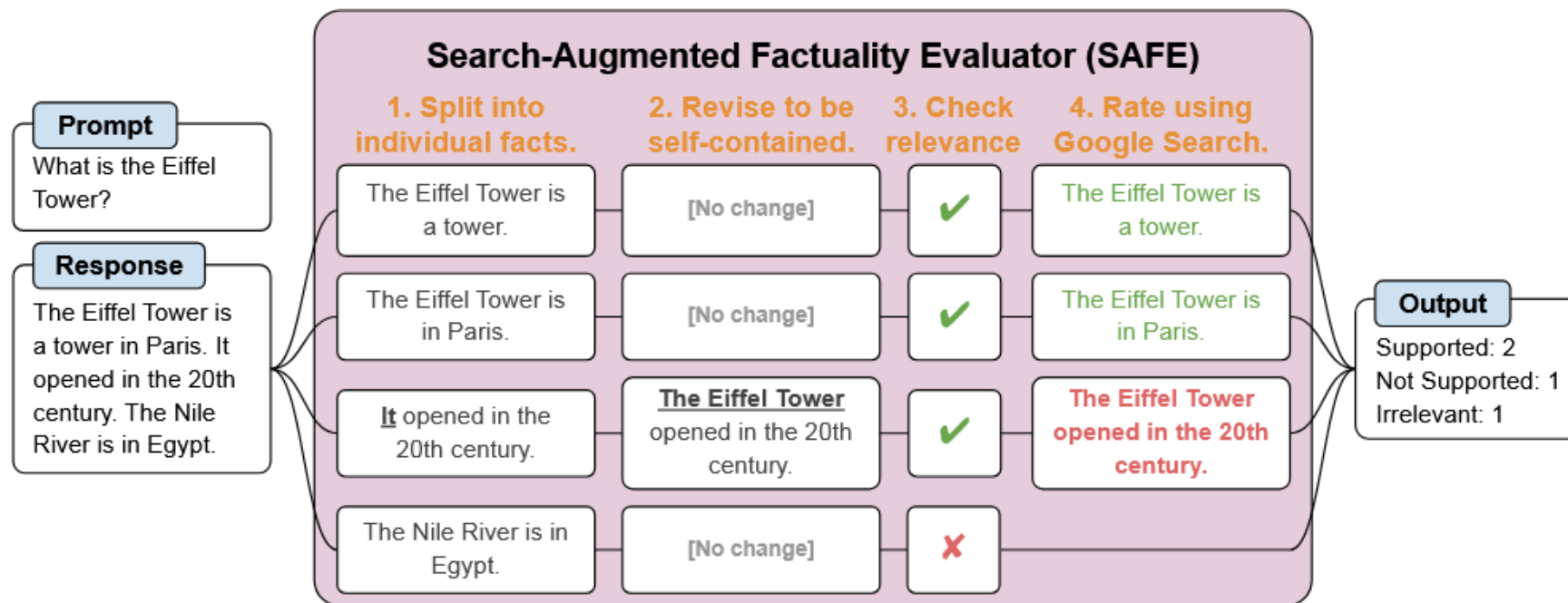


Figure 1: Our automatic factuality evaluator, SAFE, uses a large language model to rate the factuality of a long-form response to a given prompt using Google Search. We empirically demonstrate that SAFE outperforms human annotators while being more than 20 times cheaper (Section 4).

(Wei et al, 2024)

Factual Consistency (4/4)

Entailments

- Factual consistency can be framed as *textual entailment*, an NLP task.
- Textual entailment establishes the relationship between two statements.

Relationship	Definition	Factual Consistency
Entailment	The hypothesis can be inferred from the premise.	Factually consistent.
Contradiction	The hypothesis contradicts the premise.	Factually inconsistent.
Neutral	The premise neither entails nor contradicts the hypothesis.	Cannot be determined.

Safety

Unsafe content includes:

- Inappropriate language, including profanity and explicit content.
- Harmful recommendations and tutorials, including encouraging self-destructive behaviour.
- Hate speech, including racist, sexist, homophobic speech, and other discriminatory behaviours.
- Violence, including threats and graphic detail.
- Stereotypes, such as using female names for nurses or male names for CEOs.
- Biases against political positions or religion.

Political Biases in LLMs

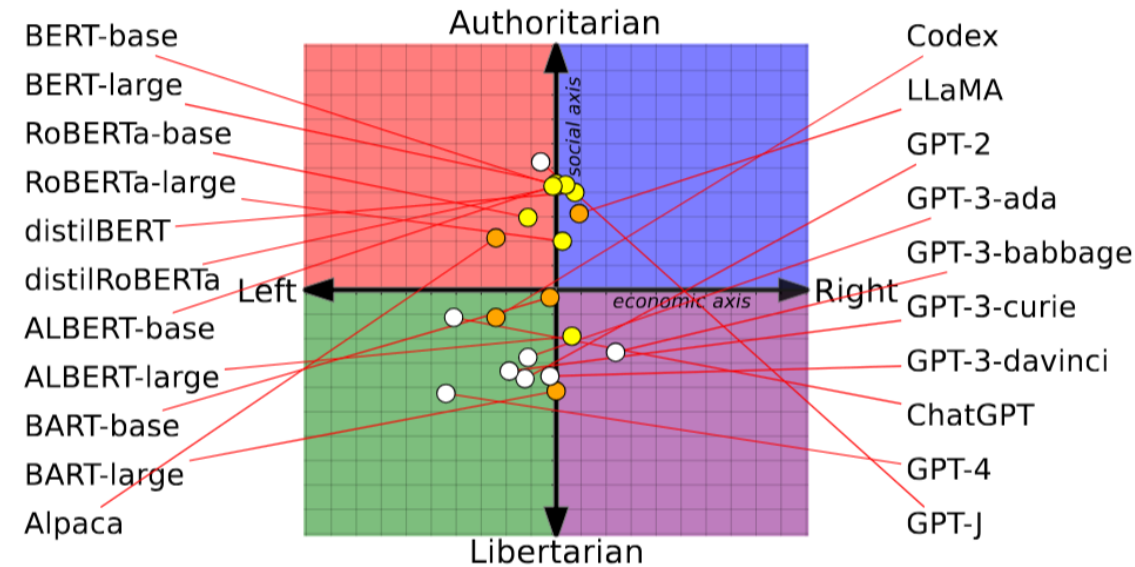


Figure 1: Measuring the political leaning of various pretrained LMs. BERT and its variants are more socially conservative compared to the GPT series. Node color denotes different model families.

(Feng et al, 2023)

Addressing Harmful Behaviour

- AI judges implemented with general purpose models.
- Models developed for to detect human harmful behaviour can also be applied.
 - Usually, smaller, faster and less expensive than AI judges.
 - Example: [Facebook hate speech detection](#).

Evaluation Criteria for Applications

- **Domain-specific capability** (coding, math, legal knowledge)
- **Generation capability** (fluency, coherence, factual consistency, safety)
- **Instruction-following** (formats, constraints, style)
- **Cost & latency** (time per token, price per output)

Factual Consistency & Safety

- Local vs Global factual consistency
- Methods: AI judges, SelfCheckGPT, SAFE, entailment models
- Safety risks: toxicity, bias, violence, harmful advice
- Benchmarks: TruthfulQA, RealToxicityPrompts, BOLD

Instruction-Following

- Core capability for foundation models
- Benchmarks:
- **IFEval** (format, constraints, JSON, keywords)
- **INFOBench** (content, style, linguistic rules)
- Roleplaying = common real-world use case

Cost and Latency

- Metrics: time-to-first-token, time per token, total query time
- Cost drivers: input/output tokens (APIs), compute (self-hosted)
- Trade-offs: performance vs cost vs latency

Model Selection Workflow

1. Filter by **hard attributes** (license, privacy, architecture)
2. Narrow with **benchmarks & leaderboards**
3. Run **custom evaluation pipeline**
4. **Monitor in production** for failures & drift

Open Source vs Model APIs

- **Model APIs**
- Pros: best models, scaling, guardrails, features (function calling)
- Cons: cost, vendor lock-in, limited control/transparency
- **Self-Hosting**
- Pros: control, transparency, customization, on-device
- Cons: heavy engineering effort, usually weaker models

Public Benchmarks & Leaderboards

- Thousands exist (BIG-bench, MMLU, GSM-8K, TruthfulQA)
- Leaderboards (Hugging Face, HELM) aggregate results
- Issues: saturation, contamination, benchmark correlation
- Best use: filter *bad models*, not pick the *best model*

Data Contamination

- Models often trained on public benchmarks → inflated scores
- Detection: n-gram overlap, low perplexity
- Handling: disclose contamination, evaluate on clean subsets
- Lesson: don't fully trust public benchmark scores

Designing an Evaluation Pipeline

Designing an Evaluation Pipeline

1. Evaluate all components (per task, per turn, per step)
2. Create clear guidelines & rubrics tied to business metrics
3. Define evaluation methods & datasets (exact, subjective, human-in-loop)
4. Validate the pipeline (reliability, bootstrap resampling, significance tests)

Key Takeaways

- Evaluation is **harder than ever** but critical for AI safety & adoption
- Mix of methods: **functional correctness + similarity + AI judges + human checks**
- Comparative evaluation rising as models surpass human skill
- Custom pipelines > public benchmarks for real-world apps
- Evaluation is the **biggest bottleneck to AI adoption**

References

References

- Chang, Yupeng et al. "A survey on evaluation of large language models." ACM transactions on intelligent systems and technology 15, no. 3 (2024): 1-45.
([arXiv:2307.03109](#))
- Chen, Mark et al. (2021). "Evaluating large language models trained on code."
[arXiv:2107.03374](#).
- Feng, Shangbin et al. "From pretraining data to language models to downstream tasks: Tracking the trails of political biases leading to unfair NLP models."
[arXiv:2305.08283](#) (2023).
- Huyen, Chip. Designing machine learning systems. O'Reilly Media, Inc., 2022
- Slattery, P. et al (2024). The AI Risk Repository: A Comprehensive Meta-Review, Database, and Taxonomy of Risks from Artificial Intelligence. [arxiv:2408.12622](#)
- Wei, Jerry et al. "Long-form factuality in large language models." Advances in

