# Evals 101

Jan Křivánek

https://github.com/JanKrivanek/

Intro

https://dotutils.net/wug-talk

# Agenda

- Evals glossary
- Graders/Judges types and scoring mechanisms
- Examples

Useful links:

- https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents
- https://microsoft.github.io/vally
- https://github.com/dotnet/skills

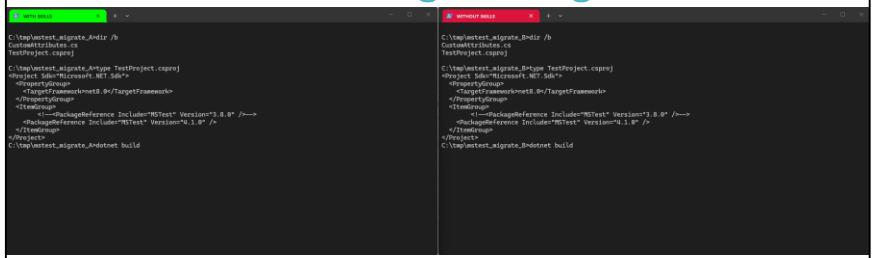
## Motivation

How do we know we are adding value?

How do we know impact of changes?

Job security

# Motivation - running skills/agents



# Evaluations and LLM as a judge

# Glossary of key terms (context: LLMs)

| TERM | DEFINITION |
|---|---|
| **Stimulus** | Input (e.g., prompt, keywords, examples) guiding model output |
| **Trajectory** | Step-by-step task path: thoughts, actions, observations. Example: jsonl data emitted by gh copilot cli |
| **Grader** | Function that evaluates a trajectory or stimulus and returns a result. Example: Regex matching or LLM judge. |
| **Score** | Aggregated numeric result from combined grader results, weighted |
| **Threshold** | Score cutoff required to pass |
| **Eval Verdict** | Pass/fail outcome based on score vs. threshold |

# Glossary of key terms (contd.)

| TERM | DEFINITION |
| --- | --- |
| **Agent harness** | System wrapping foundational model enabling its interactions (e.g. copilot cli) |
| **Eval harness** | Infra for exercising Agent and performing Evals (e.g. vally) |
| **Golden scenario** | Concrete stimuli with known expected results. Used for smoke testing eval |
| **Oracle** | Reference result for golden scenario |
| **Agreement rate** | Comparison of results of various graders (e.g. human vs LLM; Fable vs Sol) |

# Glosary example

incremental-build/eval.yaml

tests > dotnet-msbuild > eval-performance > ! eval.yaml

```yaml
scenarios:
  - name: "Analyze MSBuild evaluation performance issues"
    prompt: >-
      Analyze this project for MSBuild evaluation performance issues.
      What patterns are causing slow project evaluation?"
    setup:
      copy_test_files: true
    assertions:
      - type: "output_contains"
        value: "import"
      - type: "output_matches"
        pattern: "(glob|\\*\\*/\\*)"
      - type: "output_matches"
        pattern: "(evaluation|evaluate)"
    rubric:
      - "Identified deep import chain (level1→level2→level3) and explained evaluation overhead from nested imports"
      - "Identified overly broad glob pattern and explained it scans large directories like node_modules and .git"
      - "Suggested restricting globs or using DefaultItemExcludes"
      - "Identified property function performing file I/O during evaluation phase"
      - "Explained difference between MSBuild evaluation phase and execution phase"
    timeout: 120
```

**Stimulus** (the prompt)

**Deterministic Grader** (assertions)

**LLM as Judge Grader** (rubric)

https://github.com/dotnet/skills/blob/main/tests/dotnet-msbuild/incremental-build/eval.yaml

# Grader/Judge categories

Deterministic / rule-based
- Code/script assertations
- Cheap, fast (usually ☺), deterministic
- Ex: string match, file existence, command execution return

LLM based
- Usually “rubric”
- Expressive and versatile
- Cheaper then human

Human
- Golden scenario(s) / oracle(s) for calibration

## LLM as a Judge (1)

- Limitations of human ratings
  - Subjectivity of evaluation task (agreement rate)
  - Slow
  - Expensive
- Rule-based metrics
  - Ask human to write the ideal output of a given prompt (reference)
  - Compare the LLM output with the fixed references
  - Natural language => need to make comparisons flexible
    - *METEOR (Metric of Evaluation of Translation with Explicit Ordering)*
    - *BLEU (BiLingual Evaluation Understudy)*
    - *ROGUE (Recall-Oriented Understudy for Gisting Evaluation)*
  - Don't allow stylistic variations
  - Correlation with human ratings is poor
  - Still requires human ratings

Rule-based metrics => Only ask human once to judge and compare with LLM output changes

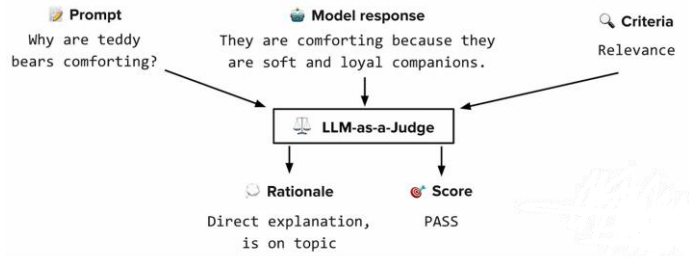
Use rule-based metrics to make these comparisons more flexible:

- METEOR: Compare reference and predicted, penalize cases when words are not in the same order
- BLEU: Precision focused metric. Look at number of matching unigrams in reference with prediction. Penalizes translation too short.
- ROUGE: Recall-Oriented Understudy for Gisting Evaluation

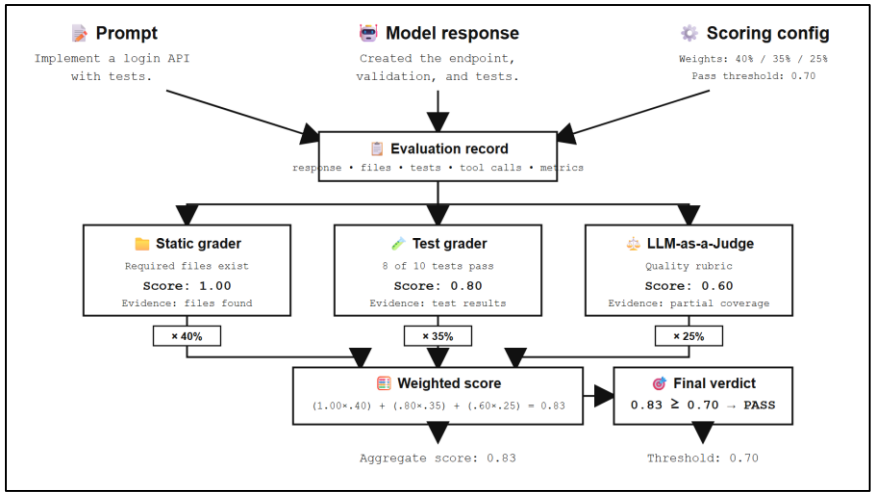
They all compare the output with the reference. They don't allow stylistic variations.

# LLM as a Judge (2)

Model response input to another LLM




Term introduced two years ago in a paper
Score: i.e. binary
Rationale: Explanation on why they graded sth with given score


Prompt
Implement a login API with tests.
Model response
Created the endpoint, validation, and tests.
Scoring config
Weights: 40% / 35% / 25%
Pass threshold: 0.70
Evaluation record
response • files • tests • tool calls • metrics
Static grader
Required files exist
Score: 1.00
Evidence: files found
Test grader
8 of 10 tests pass
Score: 0.80
Evidence: test results
LLM-as-a-Judge
Quality rubric
Score: 0.60
Evidence: partial coverage
× 40%
× 35%
× 25%
Weighted score
(1.00×.40) + (.80×.35) + (.60×.25) = 0.83
Final verdict
0.83 ≥ 0.70 → PASS
Aggregate score: 0.83
Threshold: 0.70

## LLM as a Judge (3)

First ask for rationale before score
  Allows model to externalize the thought process

How to parse response (score / rationale)?
  Force structured response (text_format)
  Try, catch, retry ☺

Benefits of LaaJ:
- No need for a reference text. No need for human ratings
- Already has knowledge from pretraining, etc.

# LLM as a Judge (4)

Pointwise Judge -> single output

Pairwise Judge -> A vs B

- Position bias: LLM prefers the first given value => Counter by swapping the order
- Verbosity bias: Longer response is preferred
- Self-enhancement bias: Use different model for judge

Remedy for verbosity bias: Explicit guidelines, few shots, and/or penalty on output length

## Selecting judge model

- Cross family with executor
- Judge *usually* performs noncomplex text based operations (from trajectories)
- Prepare sample execution results and rejudge with various models
- Ideally include human judge as well
- (Pareto) Optimize cost (or/and time) over agreement rate and MAD

# Selecting judge model

PASS/FAIL AGREEMENT

| Judge | Family | gpt-exec agree | haiku-exec agree | Verdict |
|---|---|---|---|---|
| **GPT-5.5** | GPT | **100%** | **100%** | Matches Opus everywhere |
| **Sonnet-4.6** | Claude | **100%** | **100%** | Matches Opus everywhere |
| Haiku-4.5 | Claude | 85.7% | 100% | 1 borderline flip |
| MAI-Flash | MAI | 71.4% | 100% | 2 borderline flips (lenient) |
| **Sonnet-5** | Claude | **71.4%** | 100% | 2 borderline flips (lenient), worst |
| Opus-4.8 | Claude | reference | reference | — |

MEAN ABSOLUTE DIFFERENCE

| Judge | gpt-exec | haiku-exec |
|---|---|---|
| **GPT-5.5** | **0.021** | 0.099 |
| Sonnet-4.6 | 0.068 | 0.058 |
| MAI-Flash | 0.090 | 0.063 |
| Haiku-4.5 | 0.096 | 0.084 |
| **Sonnet-5** | 0.099 | **0.157** (worst) |

https://gist.github.com/AbhitejJohn/b81c854045aaf24794453830e11be736

# MCP / CLI mocking

https://www.nuget.org/packages/MockMcp.Tool

## Assorted

- Overfitting
- Skills linting
- Dashboards
- Harbor, isolation

Sample eval with command graders:
https://github.com/dotnet/skills/blob/main/tests/dotnet-test-migration/migrate-xunit-to-mstest/eval.yaml

# Thank you

(questions?)