

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:

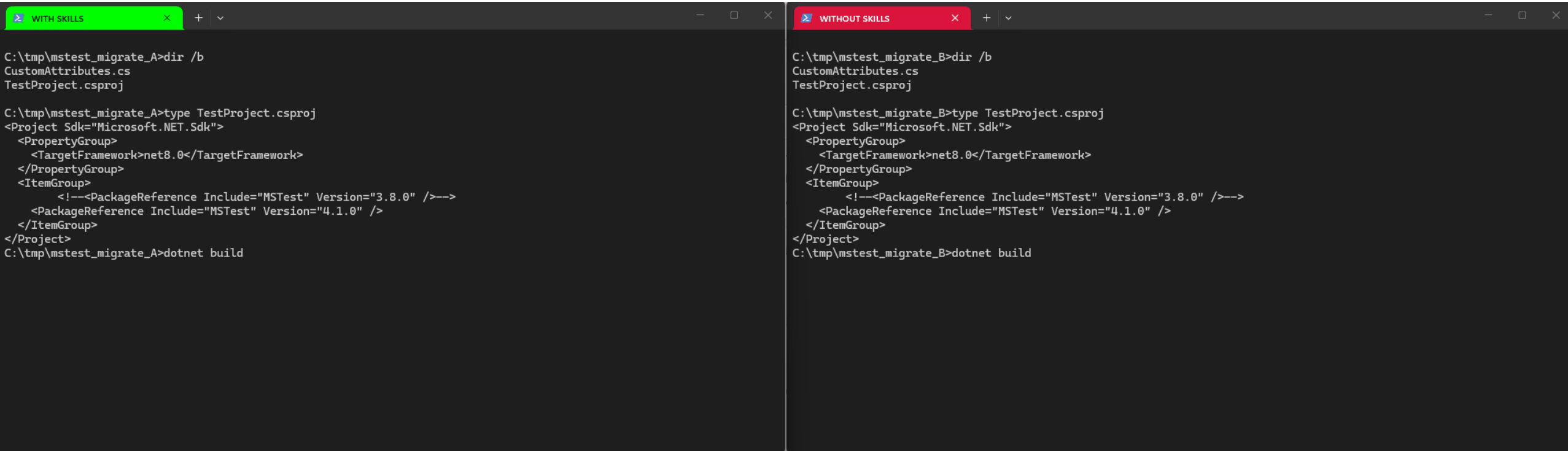
- <https://www.anthropic.com/engineering/demystifying-evals-for-ai-agents>
- <https://microsoft.github.io/vally>
- <https://github.com/dotnet/skills>

Motivation

How do we know we are adding value?

How do we know impact of changes?

Motivation - running skills/agents



The image shows two terminal windows side-by-side, illustrating a build process. The left window, titled 'WITH SKILLS', shows a successful build. The right window, titled 'WITHOUT SKILLS', shows the same process but with a failure to build the project.

```
C:\tmp\mstest_migrate_A>dir /b
CustomAttributes.cs
TestProject.csproj

C:\tmp\mstest_migrate_A>type TestProject.csproj
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <!--<PackageReference Include="MSTest" Version="3.8.0" />-->
    <PackageReference Include="MSTest" Version="4.1.0" />
  </ItemGroup>
</Project>
C:\tmp\mstest_migrate_A>dotnet build
```

```
C:\tmp\mstest_migrate_B>dir /b
CustomAttributes.cs
TestProject.csproj

C:\tmp\mstest_migrate_B>type TestProject.csproj
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
  </PropertyGroup>
  <ItemGroup>
    <!--<PackageReference Include="MSTest" Version="3.8.0" />-->
    <PackageReference Include="MSTest" Version="4.1.0" />
  </ItemGroup>
</Project>
C:\tmp\mstest_migrate_B>dotnet build
```

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

```
1 scenarios:
2   - name: "Analyze MSBuild evaluation performance issues"
3     prompt: >-
4       Analyze this project for MSBuild evaluation performance issues.
5       What patterns are causing slow project evaluation?"
6     setup:
7       copy_test_files: true
8     assertions:
9       - type: "output_contains"
10         value: "import"
11       - type: "output_matches"
12         pattern: "(glob|\\*\\*/\\*)"
13       - type: "output_matches"
14         pattern: "(evaluation|evaluate)"
15     rubric:
16       - "Identified deep import chain (level1→level2→level3) and explained evaluation overhead from nested imports"
17       - "Identified overly broad glob pattern and explained it scans large directories like node_modules and .git"
18       - "Suggested restricting globs or using DefaultItemExcludes"
19       - "Identified property function performing file I/O during evaluation phase"
20       - "Explained difference between MSBuild evaluation phase and execution phase"
21     timeout: 120
```

Stimulus
(the prompt)

Deterministic Grader
(assertions)

LLM as Judge Grader
(rubric)

Grader/Judge categories

Deterministic / rule-based

- Code/script assertions

- Cheap, fast (usually 😊), deterministic

- Ex: string match, file existence, command execution return

LLM based

- Usually “rubric”

- Expressive and versatile

- Cheaper than 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)*

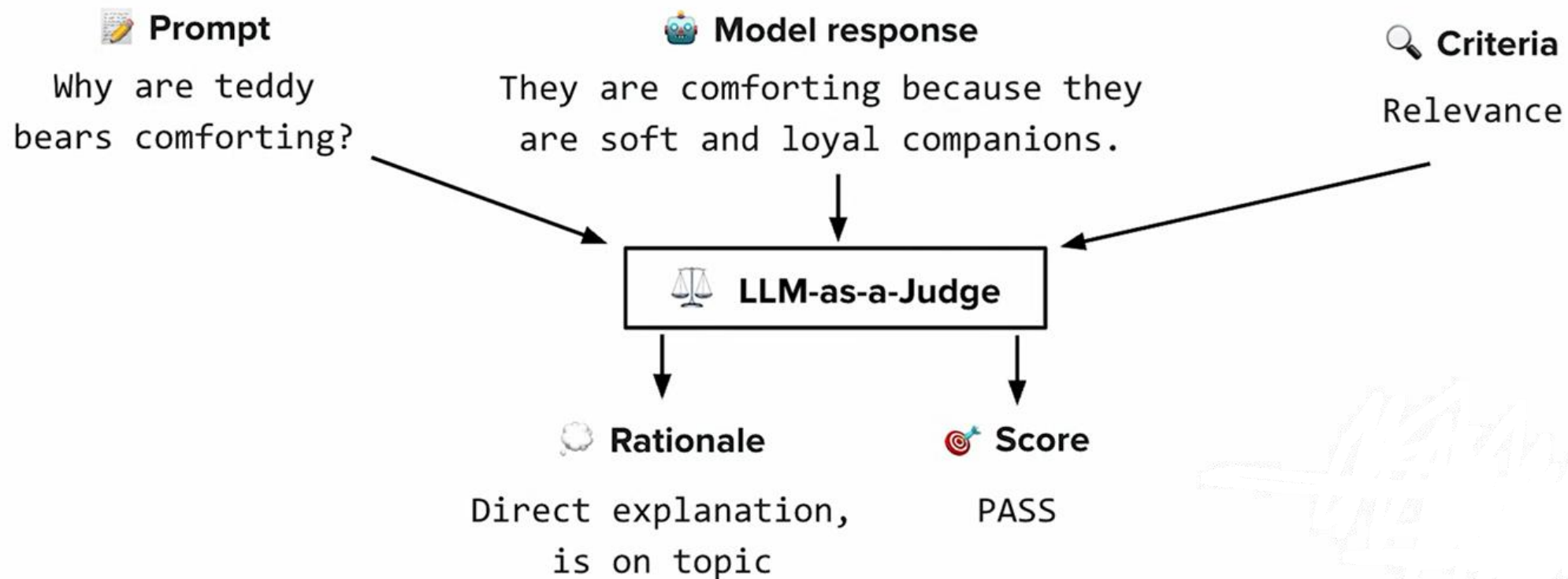
- Don't allow stylistic variations

- Correlation with human ratings is poor

- Still requires human ratings

LLM as a Judge (2)

Model response input to another LLM





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

$(1.00 \times .40) + (.80 \times .35) + (.60 \times .25) = 0.83$



Final verdict

$0.83 \geq 0.70 \rightarrow \text{PASS}$

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 😊

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

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)

MCP / CLI mocking

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

Assorted

- Overfitting
- Skills linting
- Dashboards
- Harbor, isolation

Thank you

(questions?)