LLM-SRBench: A New Benchmark for Scientific Equation Discovery with Large Language Models


*Parshin Shojaee


*Ngoc-Hieu Nguyen


Kazem Meidani


Amir Barati Farimani


Khoa D Doan


Chandan K Reddy

Outline

Background

Motivation

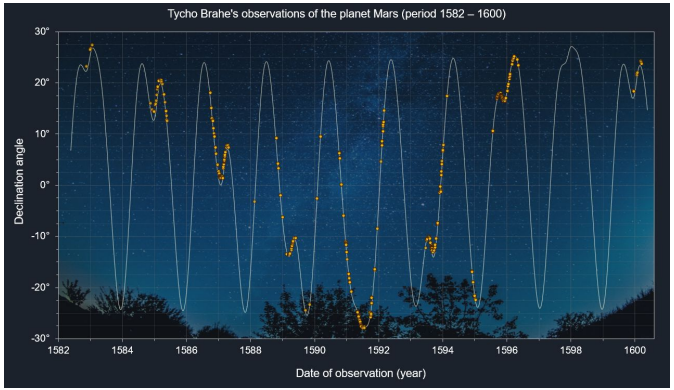
The Proposed Approach

Results

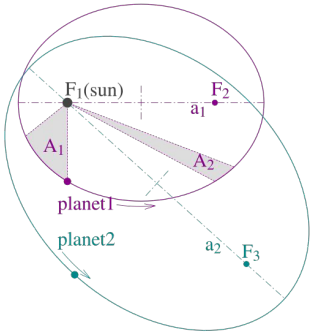
Scientific Equation Discovery (SED)

Find a concise and interpretable mathematical expression that closely models empirical observations.

Tycho’s observatory data



Kepler’s laws of planetary motion



$$\frac{dA}{dt} = \frac{1}{m}\frac{1}{2}(\vec{r} \times \vec{p})$$

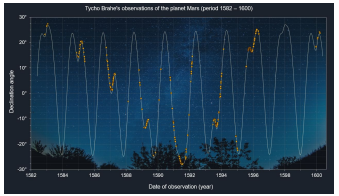
[1] Mathematical discoveries from program search with large language models. Romera-Paredes et al (2024)

Figure sources [1]

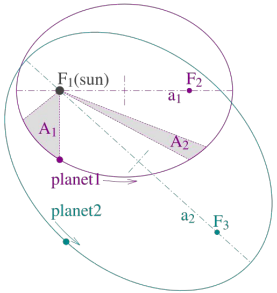
Scientific Equation Discovery (SED)

Find a concise and **interpretable** mathematical expression that closely models empirical observations.

Tycho's observatory data

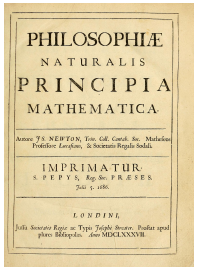


Kepler's laws



$$\frac{dA}{dt} = \frac{1}{m}\frac{1}{2}(\vec{r} \times \vec{p})$$

Newton's laws


PHILOSOPHIÆ
NATURALIS
PRINCIPIA
MATHEMATICA

- Mathematical equations vs Predictive model: provide insights, extrapolation, knowledge transfer

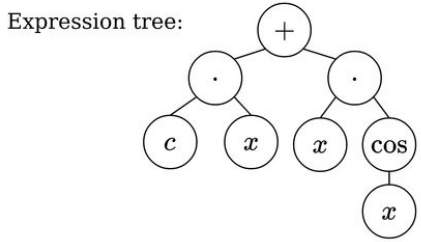
[1] Mathematical discoveries from program search with large language models. Romera-Paredes et al (2024)
[2] ICML 2024 Tutorial: Physics of Language Models. Allen-Zhu et al (2024)

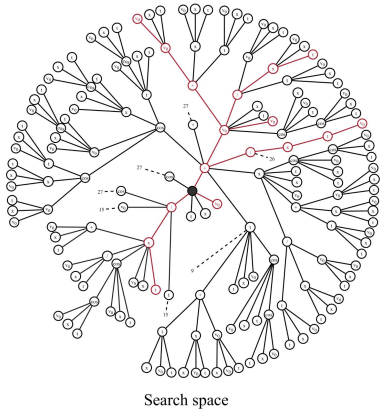
Figure sources [1]

Approaches

- Evolutionary search algorithms
- DL methods using Transformers
- Recent works have shown that using **LLMs for SED is a promising approach**

$$c \cdot x + x \cdot \cos(x)$$




Search space

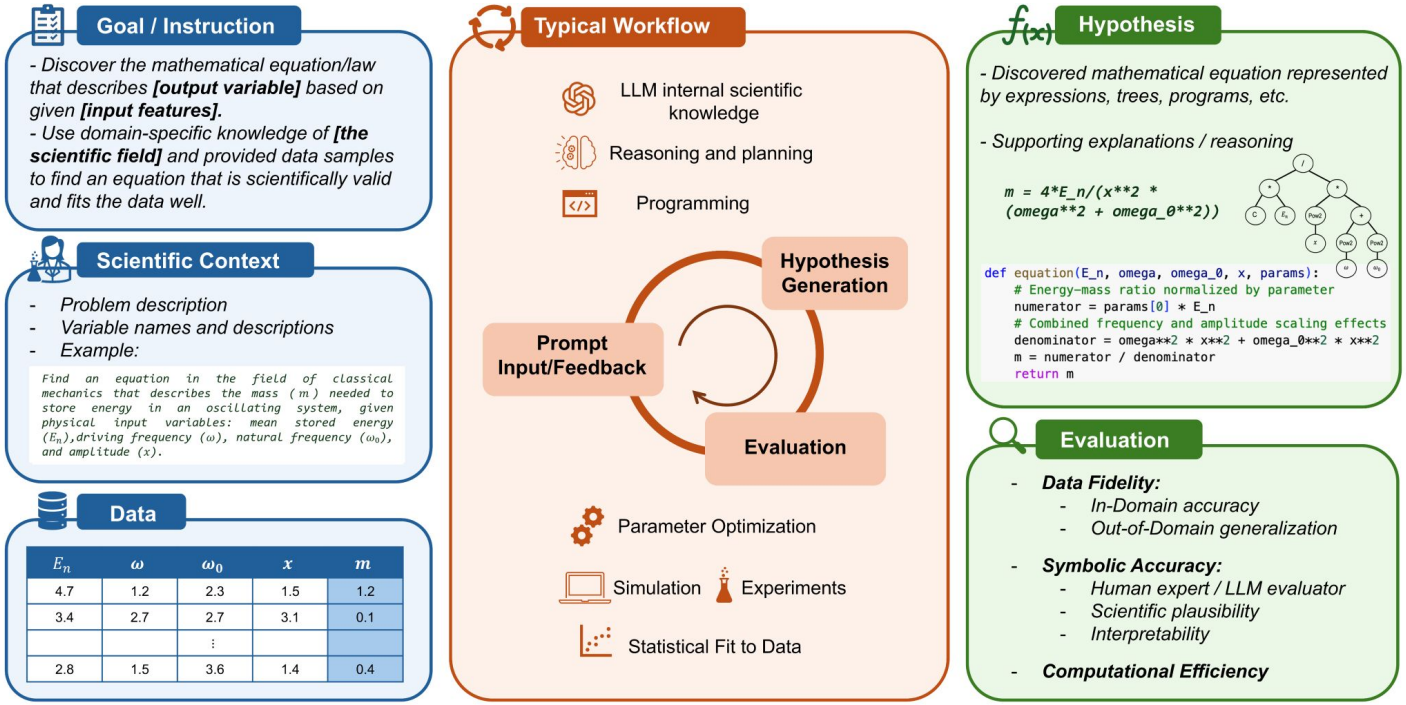
Interpretable machine learning for science with pysr and symbolicregression. Cranmer et al. (2023)
Symbolic regression via deep reinforcement learning enhanced genetic programming seeding. Mundhenk et al (2021)
Contemporary symbolic regression methods and their relative performance. La Cava et al (2021)

Motivation of using LLMs

- LLMs have the ability to
  - process vast amounts of scientific literature
  - extract relevant information
  - generate coherent hypotheses
- LLMs have shown the potential to enhance automatic reasoning and problem-solving capabilities [1, 2]

[1] Large language models for scientific synthesis, inference and explanation. Zheng et al (2023)
[2] Mathematical discoveries from program search with large language models. Romera-Paredes et al (2024)

Scientific Equation Discovery & LLMs



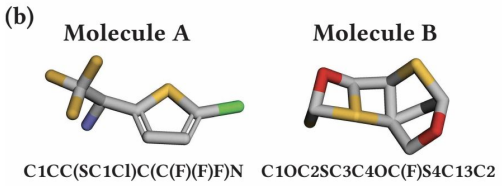
Scientific Equation Discovery & LLMs

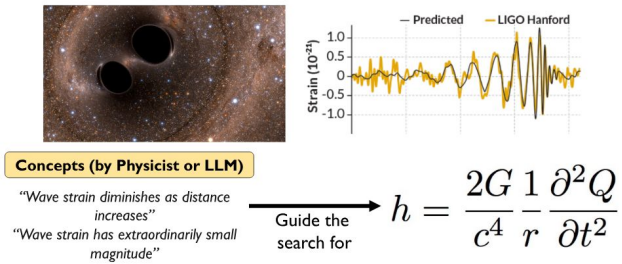
- **LLM-SR: Scientific equation discovery via programming with large language models.** Shojaee et al 2024
- **In-context symbolic regression: Leveraging large language models for function discovery.** Merler et al 2024.
- **LaSR: Symbolic Regression with a Learned Concept Library.** Arya et al (NeurIPS 2024).
- **LLM and Simulation as Bilevel Optimizers: A New Paradigm to Advance Physical Scientific Discovery.** Pingchuan et al (ICML 2024).

(a)

```python
class Physics(nn.Module):
  def __init__(self, youngs_modulus_log: float = 13.03,
                     poissons_ratio_sigmoid: float = -1.99):
    super().__init__()
    self.youngs_modulus_log = nn.Parameter(
      torch.tensor(youngs_modulus_log))  # Log of Young's modulus
    self.poissons_ratio_sigmoid = nn.Parameter(
      torch.tensor(poissons_ratio_sigmoid))  # Sigmoid of Poisson's ratio
  def forward(self, F: torch.Tensor) -> torch.Tensor:
    youngs_modulus = self.youngs_modulus_log.exp()
    poissons_ratio = torch.sigmoid(self.poissons_ratio_sigmoid) * 0.49
    mu = youngs_modulus / (2 * (1 + poissons_ratio))  # Shear modulus
    lam = youngs_modulus * poissons_ratio / (
      (1 + poissons_ratio) * (1 - 2 * poissons_ratio))
    # Deformation gradient determinant J
    J = F.det().view(-1, 1, 1)
    F_invT = F.inverse().transpose(1, 2)
    # Volumetric part
    P_vol = lam * (J - 1) * F_invT
    # Deviatoric part
    P_dev = mu * (F - (1 / J) * F_invT)
    # Compute Kirchhoff stress tensor
    kirchhoff_stress = P_vol + P_dev @ F.transpose(1, 2)
    return kirchhoff_stress
```

(b)





Outline

Background

Motivation

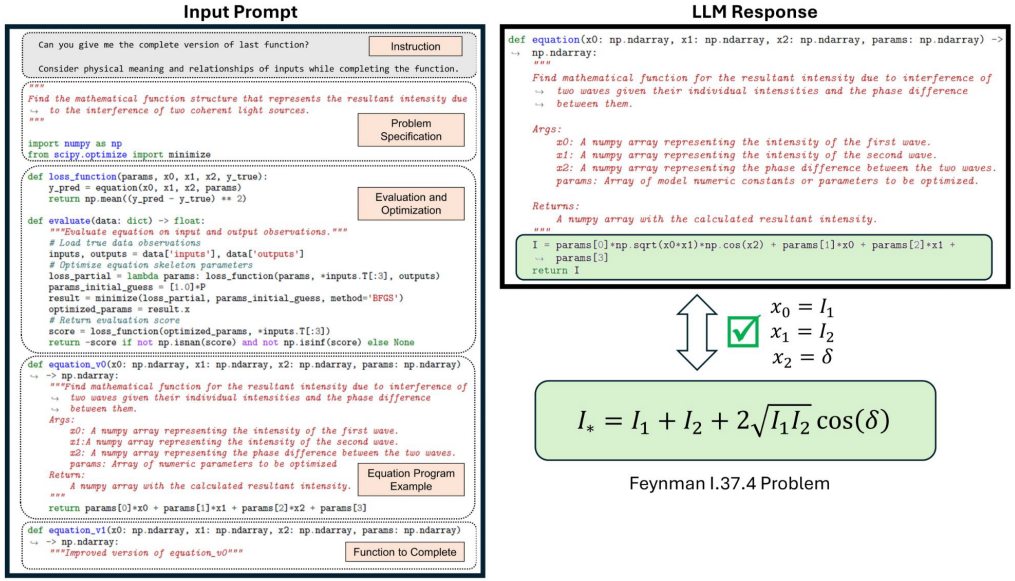
The Proposed Approach

Results

The need of a new SED benchmark

1. Data contamination

A better score might not reflect an improvement but indicating that a model memorised the benchmark data.



[1] LLM-SR: Scientific Equation Discovery via Programming with Large Language Models

Current SR benchmarks

- AI Feynman dataset (Udrescu & Tegmark, 2020)
  - 120 equations derived from the Feynman Lectures on Physics
- Strogatz dataset (Strogatz, 2018)
  - 14 equations modeling nonlinear and chaotic dynamical processes
- SRBench (La Cava et al., 2021)
  - combine AI Feynman dataset with Strogatz dataset, different noise levels
  - 14 SR methods
- SRSD (Matsubara et al., 2022)
  - address some issues in the AI Feynman dataset

[1] ICML 2024 Tutorial: Physics of Language Models
[2] Rethinking Symbolic Regression Datasets and Benchmarks for Scientific Discovery. Matsubara et al 2024

Outline

Background

Motivation

LLM-SRBench

Results

Creating a new benchmark with transformed equations

$$E_n = \frac{1}{4} m(\omega^2 + \omega_0^2)x^2 \longrightarrow \omega = 4\frac{\sqrt{\frac{E_n}{m} - \omega_0^2 x^2}}{x}$$

```python
def equation(m: np.ndarray, omega: np.ndarray, omega_0: np.ndarray, x: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Mathematical function for Energy

    This function represents the energy of a vibrating string as a function of mass, frequency, length, and natural frequency.

    Args:
        m: A numpy array representing observations of Mass.
        omega: A numpy array representing observations of Frequency.
        omega_0: A numpy array representing observations of Natural Frequency.
        x: A numpy array representing observations of Length.
        params: Array of numeric constants or parameters to be optimized (currently unused)

    Return:
        A numpy array representing Energy as the result of applying the mathematical function to the inputs.
    """
    # calculate terms
    squared_omega = omega**2
    squared_omega_0 = omega_0**2
    squared_x = x**2

    # calculate energy
    energy = 0.25 * m * (squared_omega + squared_omega_0) * squared_x

    # return the energy
    return energy
```

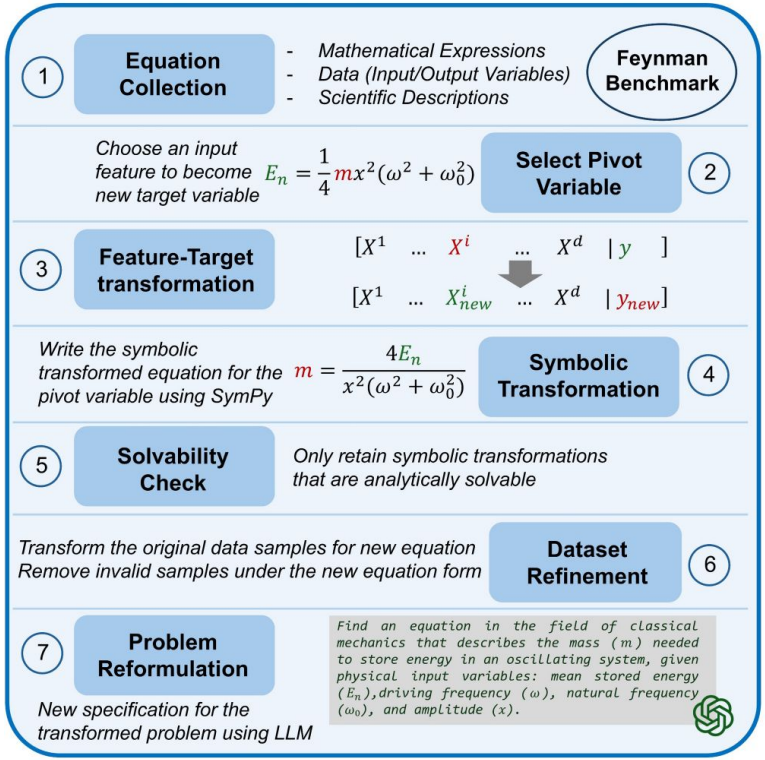
```python
def equation(E_n: np.ndarray, m: np.ndarray, omega_0: np.ndarray, x: np.ndarray, params: np.ndarray) -> np.ndarray:
    """Improved version of `equation_v1`.

    Args:
        E_n: A numpy array representing observations of Energy.
        m: A numpy array representing observations of Mass.
        omega_0: A numpy array representing observations of Frequency (initial frequency or reference frequency).
        x: A numpy array representing observations of Length.
        params: Array of numeric constants or parameters to be optimized.

    Return:
        A numpy array representing Frequency as the result of applying the mathematical function to the inputs.
    """
    # Calculate the expected frequency based on energy and mass
    expected_frequency = np.sqrt(E_n / m)

    # Introduce a scaling factor based on length, with a decay factor in the exponential term
    output = expected_frequency * params[0] * np.exp(-params[1] * x / (x ** 2 + params[2])) * (params[3] + params[4] * np.sin(np.sqrt(x)) + params[5] * np.cos(np.sqrt(x)))

    # Add a reference frequency term with a decay factor
    output += omega_0 * params[6] * np.exp(-params[7] * x)

    return output
```

$$E = 0.25 \cdot m \cdot (\omega^2 + \omega_0^2) \cdot x^2$$

$$\left[\sqrt{\frac{E_n}{m}} \cdot p_0 \cdot e^{-\frac{p_1 x}{x^2 + p_2}} \cdot \left(p_3 + p_4 \sin(\sqrt{x}) + p_5 \cos(\sqrt{x})\right)\right] + \left[\omega_0 \cdot p_6 \cdot e^{-p_7 x}\right]$$

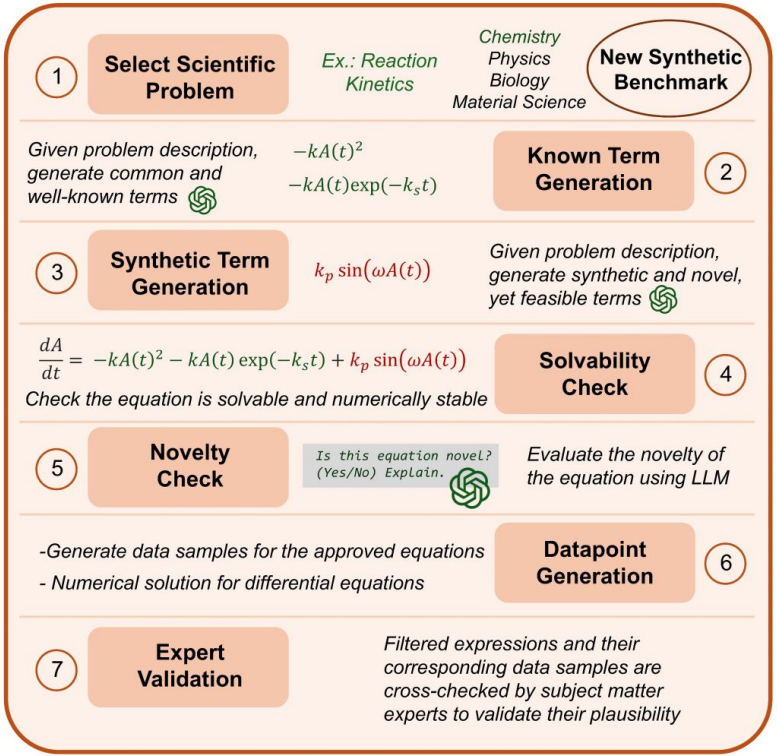
LLM-SR found the correct expression of the original but failed on the transformed equation

Creating a new benchmark with transformed equations



(a) LSR-Transform

Creating a new benchmark with synthetic equations



(b) LSR-Synth

LLM-SRBecnh | Metrics

📊 Data Fidelity + 🎯 Symbolic Accuracy

Data Fidelity: How well discovered equations fit data

- **$Acc_{0.1}$** (accuracy to tolerance) + **NMSE** (normalized error)
- **Out-of-Domain testing**: True scientific equations must generalize

Symbolic Accuracy: LLM (GPT-4o) as Judge

✅ **Handles any hypothesis representations**: Programs, expressions, strings
✅ **Mathematical equivalence**: Beyond exact matching
✅ **94.6% human agreement** + **Strong OOD correlation**

Outline

Background

Motivation

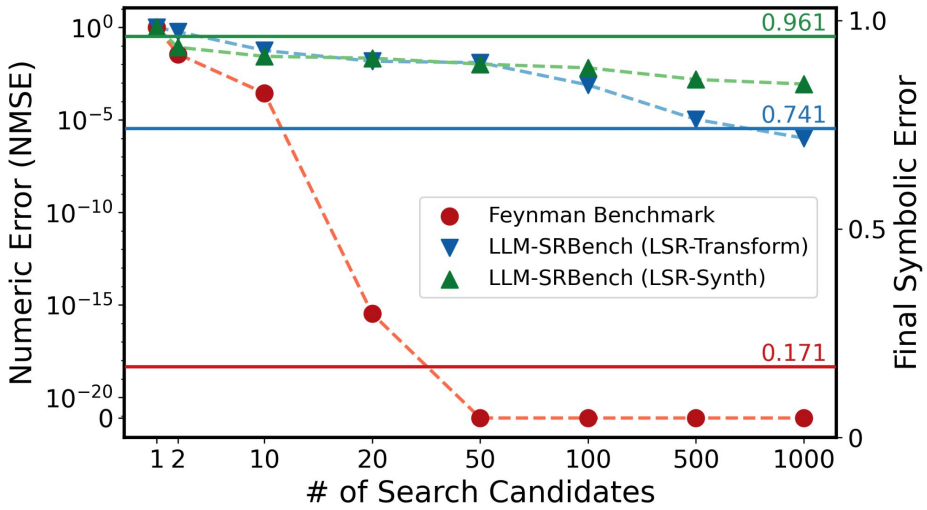
The Proposed Approach

Results

The need of a new SED benchmark

1. Data contamination

A better score might not reflect an improvement but indicating that a model memorised the benchmark data.



[1] LLM-SR: Scientific Equation Discovery via Programming with Large Language Models

LLM-SRBecnh | Results

🔬Key finding: Even state-of-the-art discovery methods using different LLM backbones only achieve ~31% symbolic accuracy on our benchmark, highlighting the challenging nature of scientific equation discovery!

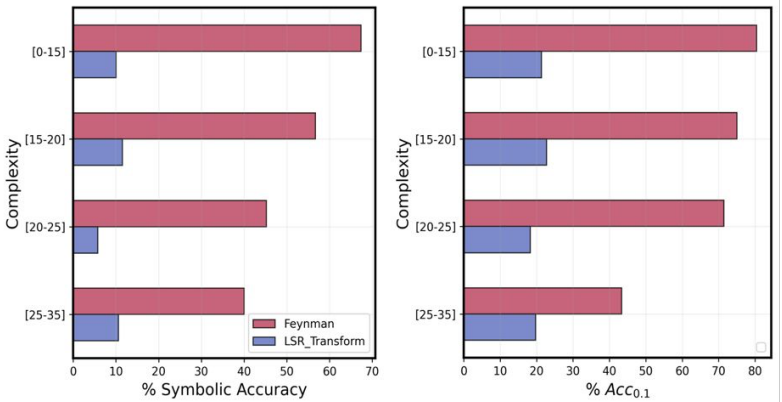
Models	LSR-Transform			LSR-Synth											
				Chemistry			Biology			Physics			Material Science		
	SA (%)↑	$Acc_{0.1}$(%)↑	NMSE↓	SA (%)↑	$Acc_{0.1}$(%)↑	NMSE↓	SA (%)↑	$Acc_{0.1}$(%)↑	NMSE↓	SA (%)↑	$Acc_{0.1}$(%)↑	NMSE↓	SA (%)↑	$Acc_{0.1}$(%)↑	NMSE↓
Direct Prompting (DataBlind)															
Llama-3.1-8B-Instruct	3.61	1.801	0.3697	0.0	0.0	0.0644	0.0	0.0	0.5481	0.0	0.0	0.0459	0.0	0.0	0.0826
GPT-3.5-turbo	2.10	1.801	0.3553	0.0	8.33	**0.0023**	0.0	4.16	0.5990	0.0	2.27	**0.0274**	0.0	0.0	**0.0277**
GPT-4o-mini	**7.21**	**6.306**	**0.2631**	0.0	**13.88**	0.0221	0.0	**4.16**	**0.4648**	4.54	**9.09**	0.0647	0.0	0.0	0.0484
SGA (Ma et al., 2024)															
Llama-3.1-8B-Instruct	2.70	0.909	0.3519	0.0	8.33	0.0458	0.0	0.0	0.2416	0.0	2.27	0.1549	0.0	12.12	0.0435
GPT-3.5-turbo	0.0	0.909	0.3465	0.0	8.33	0.0071	0.0	8.33	0.1279	2.27	4.54	**0.0249**	0.0	28.10	0.0019
GPT-4o-mini	**9.91**	**8.11**	**0.2321**	0.0	**16.66**	**5.46e-4**	**4.16**	**12.51**	**0.0128**	**4.54**	**9.09**	0.0511	0.0	**36.11**	**6.02e-4**
LaSR (Grayeli et al., 2024)															
Llama-3.1-8B-Instruct	5.41	45.94	0.0021	0.0	27.77	2.77e-4	4.16	16.66	2.73e-4	4.54	25.02	0.0018	8.21	64.22	7.44e-5
GPT-3.5-turbo	**12.61**	47.74	0.0015	0.0	38.89	1.51e-4	0.0	16.66	2.31e-4	6.81	22.71	0.0011	20.66	64.09	3.77e-5
GPT-4o-mini	6.31	**50.45**	**0.0011**	**2.77**	**38.92**	**9.11e-5**	**8.33**	**20.83**	**1.53e-4**	**9.91**	**31.81**	**9.94e-4**	**28.12**	**72.04**	**9.23e-6**
LLM-SR (Shojaee et al., 2024b)															
Llama-3.1-8B-Instruct	30.63	38.55	0.0101	8.33	**66.66**	8.01e-6	**25.30**	**58.33**	**1.04e-6**	6.97	34.09	1.23e-4	4.10	88.12	1.15e-7
GPT-3.5-turbo	10.81	10.81	0.1449	0.0	50.22	2.87e-5	0.0	25.03	2.33e-5	0.0	25.12	8.84e-4	12.42	82.14	2.75e-8
GPT-4o-mini	**31.53**	**39.64**	**0.0091**	**11.11**	52.77	**4.12e-6**	16.66	29.16	3.06e-6	**9.91**	**36.36**	**7.62e-5**	**20.24**	**88.28**	**3.21e-9**

LLM-SRBecnh | Results

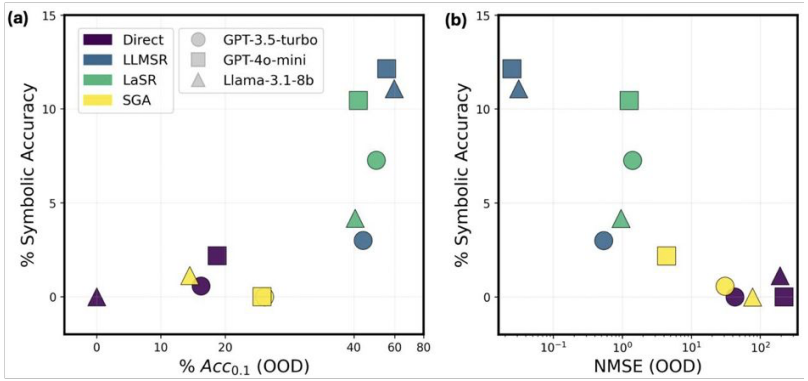
⚠️**Same complexity**

⇩

Considerably different performance



📈**Strong correlation between <u>Symbolic Accuracy</u> & <u>OOD generalization</u>**



Conclusion

- Introduced **LLM-SRBench**: A new benchmark addressing data contamination in SED
- Proposed Symbolic Accuracy evaluated by an LLM-as-a-Judge
- Highlighted SED Challenge: SOTA methods achieve ~31% symbolic accuracy on our benchmark

Thank you