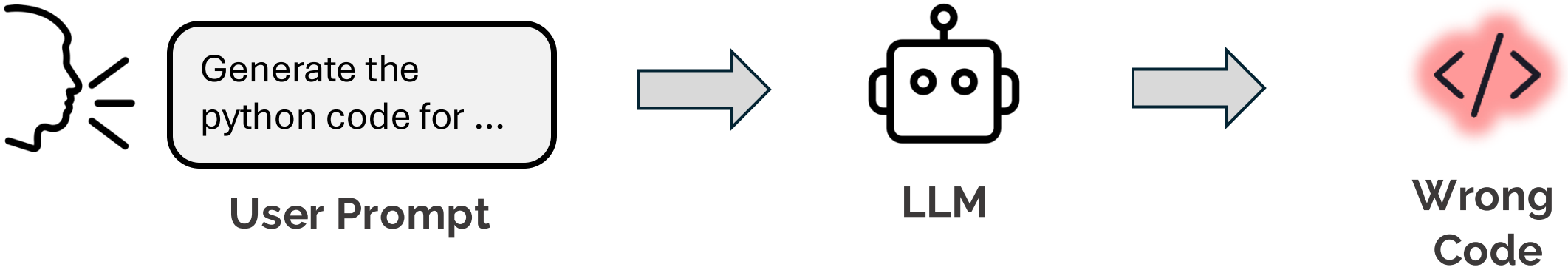
Selective Prompt Anchoring for Code Generation

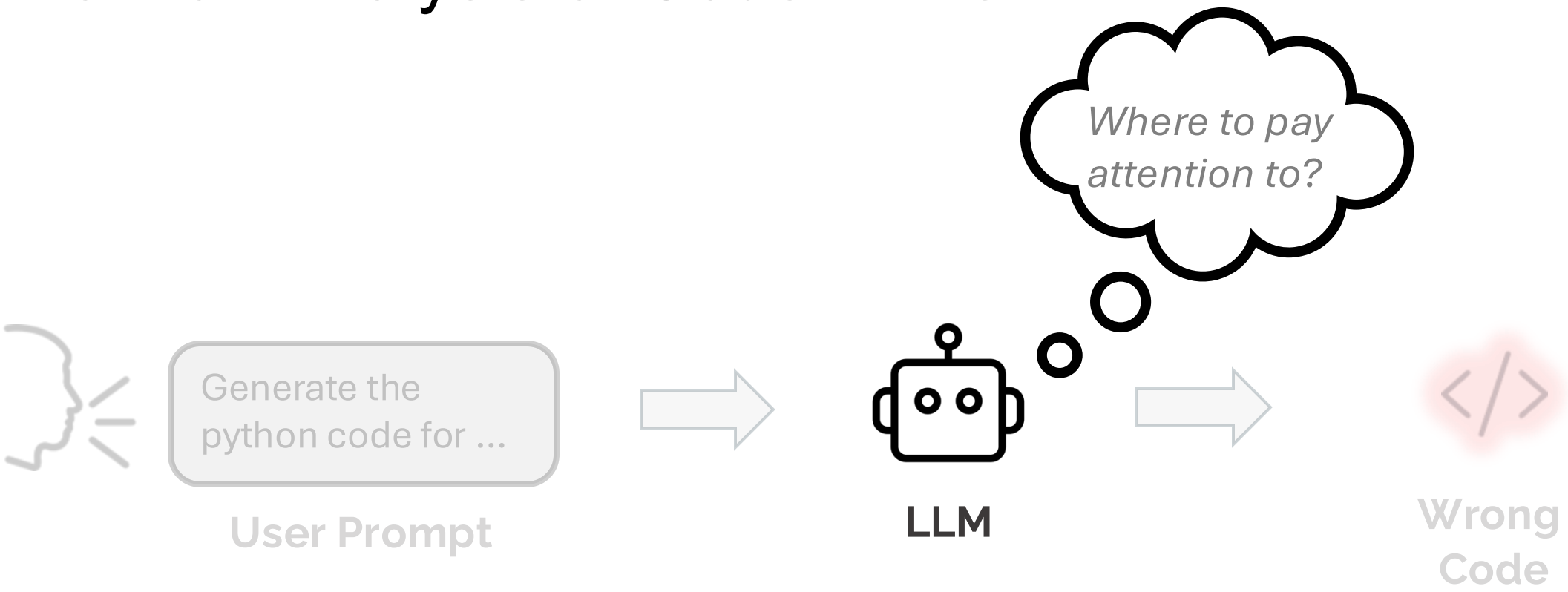
Yuan Tian
Purdue University

Tianyi Zhang
Purdue University

LLMs Make Mistakes in Code Generation



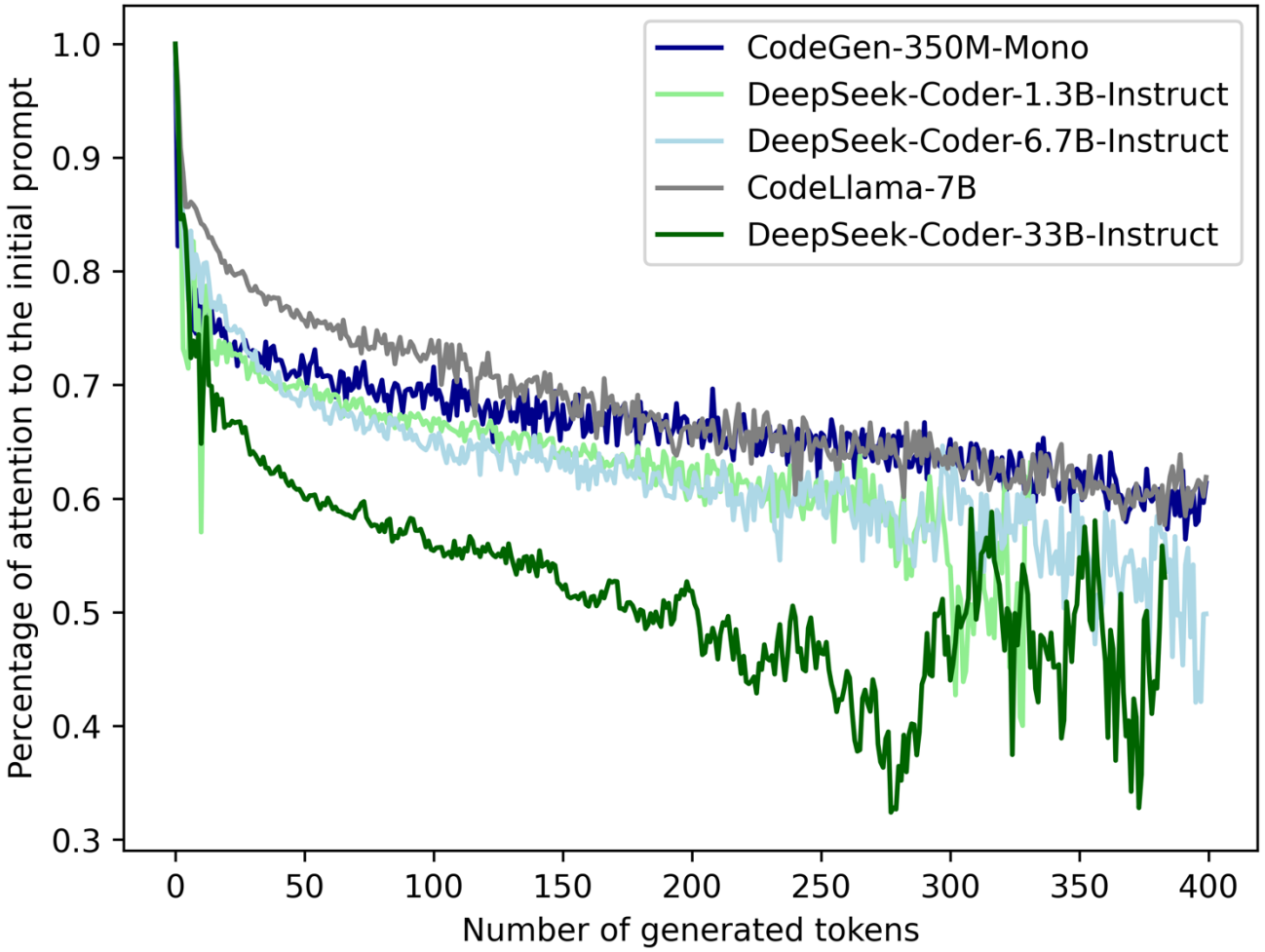
Attention Analysis of Code LLMs



› *Analyze LLMs' **attention shift** during **code generation.***

Attention Dilution

- Attention to the **user prompt** gradually **decreases** as generating more tokens



Attention Dilution Leads to Errors

- Attention dilution leads to code generation errors

```python
import pandas as pd

def compute_total_sales(csv_path):

    try:
        df = pd.read_csv(csv_path)
    except Exception as e:
        print(f"Error reading file: {e}")
        return None

    if 'Date' not in df.columns or 'Price' not in df.columns:
        print("Error: Missing required columns.")
        return None

    df['Date'] = pd.to_datetime(df['Date'], errors='coerce')

    df = df.dropna(subset=['Date', 'Price'])

    total_sales = df['Price'].sum()
    total_sales = df.groupby('Date')['Price'].sum()

    return total_sales
```

"Write a Python function to compute **daily** total sales from a CSV with Date and Price columns."

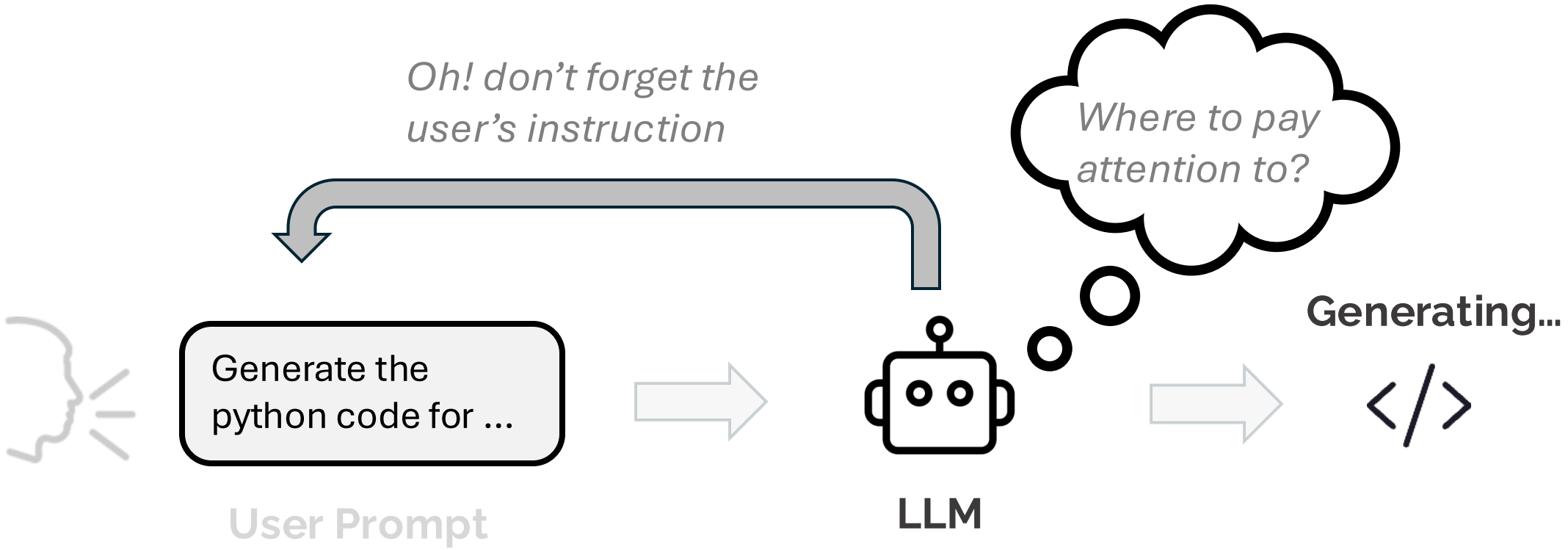
Attention Dilution Leads to Errors

- **Longer** generated code leads to **lower accuracy**
- Confirm that attention dilution is detrimental

	Passed	Failed
Easy	294	418
Medium	475	664
Hard	400	784
Overall	390	622

Addressing Attention Dilution

- ***Attention steering** by **logit arithmetic***
  - ***Model-agnostic***
  - ***Training-free***



Selective Prompt Anchoring (SPA)

1. Influence of other tokens
2. Positional encoding
3. Random noises

"Write a Python function to compute daily total sales from a CSV with Date and Price columns."

Embed → LLM → $f_\theta(\mathbf{E}_i)$

Logits given original input

Pay more attention to "***daily***"

Mask

"Write a Python function to compute daily total sales from a CSV with Date and Price columns."

Embed → LLM → $f_\theta(\mathbf{E}_i^{mask})$

Logits given masked input

⊖ *Cancel out noises*

=

Logit difference

⊗ ***Strength***

*Semantic influence of "**daily**"*

⊕ → New logits

*Logits with more influence caused by "**daily**"*

$$\omega \cdot f_\theta(\mathbf{E}_i) + (1 - \omega) \cdot f_\theta(\mathbf{E}_i^{mask})$$

Prompt Embeddings

Anchoring Strength (ω)

Anchored text

"Write a Python function to compute daily total sales from a CSV with Date and Price columns."

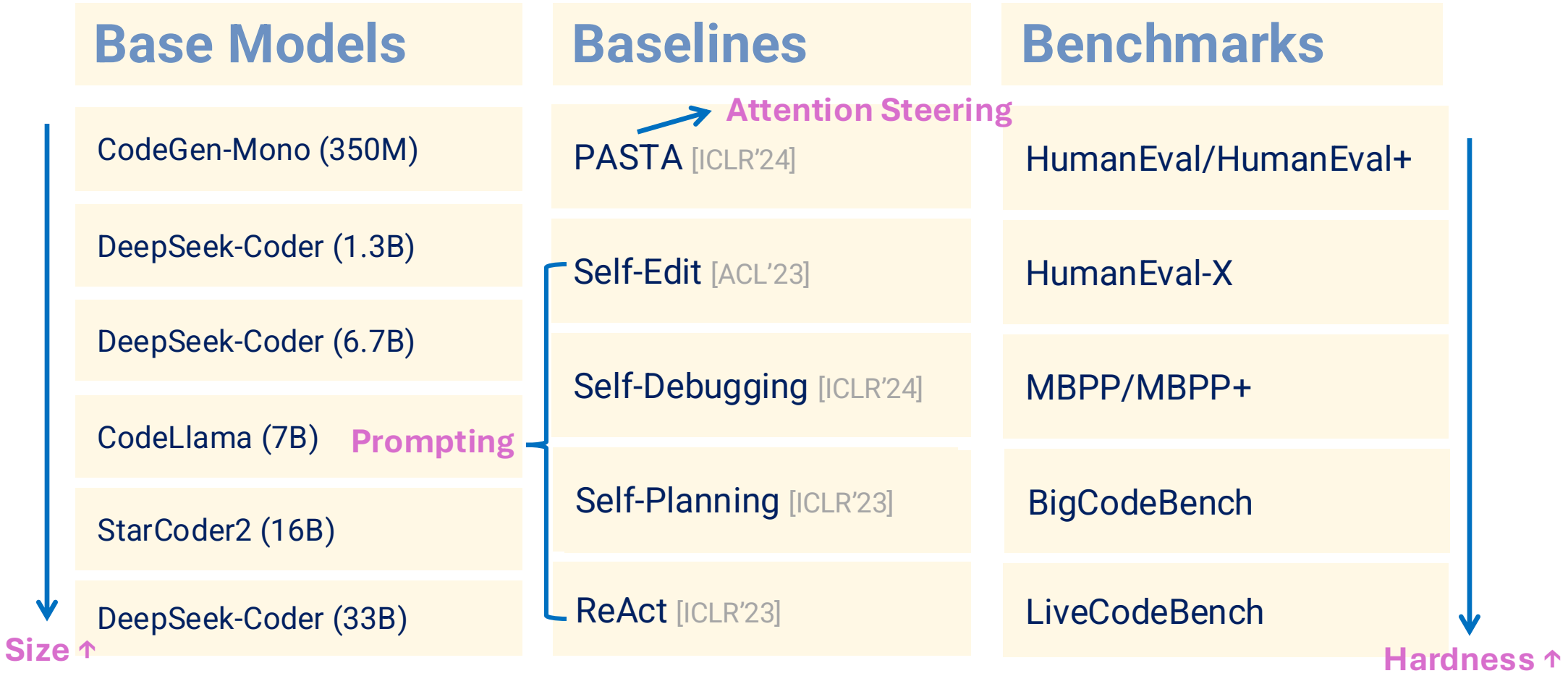
Anchoring Strength

$$\omega \cdot f_\theta(\mathbf{E}_i) + (1 - \omega) \cdot f_\theta(\mathbf{E}_i^{mask})$$

- ω > 1: Pay more attention to the anchored text
- ω = 1: Original output
- 0 < ω < 1: Pay less attention to the anchored text
- ω = 0: Totally ignore anchored text
- ω < 0: Reverse the influence of anchored text
  - For example, if the anchored text is "uppercase", the model will view it as "lowercase".

Focus on increasing attention

Experiments



Improvements over Base Models

SPA consistently improves code generation performance

	Size	HumanEval	HumanEval +	MBPP	MBPP+	BigCodeBench	LiveCodeBench
CodeGen-Mono	350M	15.3	12.2	19.6	15.9	1.1	1.1
+SPA		+4.8(31%↑)	+4.9(40%↑)	+7.8(31%↑)	+6.7(42%↑)	+0.5(45%↑)	+0.4(36%↑)
DeepSeek-Coder	1.3B	66.4	61.8	58.2	52.4	2.5	6.5
+SPA		+3.7(6%↑)	+5.9(10%↑)	+2.7(5%↑)	+0.0(0%-)	+0.9(36%↑)	+2.7(42%↑)
DeepSeek-Coder	6.7B	75.6	70.2	67.0	58.5	12.7	7.8
+SPA		+12.9(17%↑)	+9.7(14%↑)	+4.0(6%↑)	+2.2(4%↑)	+3.7(29%↑)	+3.0(38%↑)
CodeLlama	7B	33.6	28.2	50.9	40.8	3.4	3.8
+SPA		+10.4(31%↑)	+7.8(28%↑)	+3.4(7%↑)	+3.2(8%↑)	+0.7(5%↑)	+0.2(5%↑)
StarCoder2	16B	67.7	60.4	78.0	65.1	13.3	7.0
+SPA		+7.9(12%↑)	+5.2(+9%↑)	+4.0(5%↑)	+4.0(6%↑)	+1.0(8%↑)	+1.2(17%↑)
DeepSeek-Coder	33B	81.7	77.1	73.1	63.2	18.9	11.9
+SPA		+4.5(6%↑)	+2.2(3%↑)	+6.0(8%↑)	+7.1(11%↑)	+2.6(14%↑)	+3.9(33%↑)

Comparison with SOTA Baselines

SPA outperforms existing SOTA code generation approaches.

	ΔPass@1	Latency (Sec)
Base Model	15.3	12.2
PASTA [ICLR'24]	+1.2	48.8
Self-Debugging [ICLR'24]	+4.2	27.3
Self-Edit [ACL'23]	+1.8	26.4
Self-Planning [ICLR'23]	+3.6	21.6
ReAct [ICLR'23]	+1.3	28.8
SPA (Ours)	**+7.7**	**9.8**

Evaluate with Different Programming Languages

Consistently improves in multilingual programming

	Size	Python	Java	JavaScript	C++	Go
CodeGen-Mono	350M	15.3	9.8	13.4	9.8	6.7
+SPA		+4.8	+3.0	+4.3	+3.7	+6.7
DeepSeek-Coder	1.3B	66.4	42.7	57.3	43.3	40.2
+SPA		+6.7	+3.0	+3.0	+2.4	+1.8
DeepSeek-Coder	6.7B	75.6	48.8	65.2	49.4	45.7
+SPA		+12.9	+8.5	+11.6	+1.8	+7.3
CodeLlama	7B	33.6	22.0	29.3	22.0	20.1
+SPA		+10.4	+6.1	+8.5	+6.1	+6.7
StarCoder2	16B	67.7	22.0	29.3	22.0	20.1
+SPA		+7.9	+5.5	+7.9	+5.5	+6.1
DeepSeek-Coder	33B	81.7	53.0	70.7	53.7	49.4
+SPA		+4.5	+3.0	+3.7	+2.4	+2.4

Open-source

- Install with PyPI

  pip install anchoring
- Fully integrate with HuggingFace API
- Directly support **ALL** HuggingFace models

```python
from transformers import pipeline
import anchoring

pipe = pipeline(
    "selective-prompt-anchoring",
    model="meta-llama/Llama-3.1-8B",
)

output = pipe(
    "How is the weather today?",
    anchors=['today']
)
print(output["generated_text"])
```

Now you can assign importance values to prompt!

Thank you!