Machine Programming

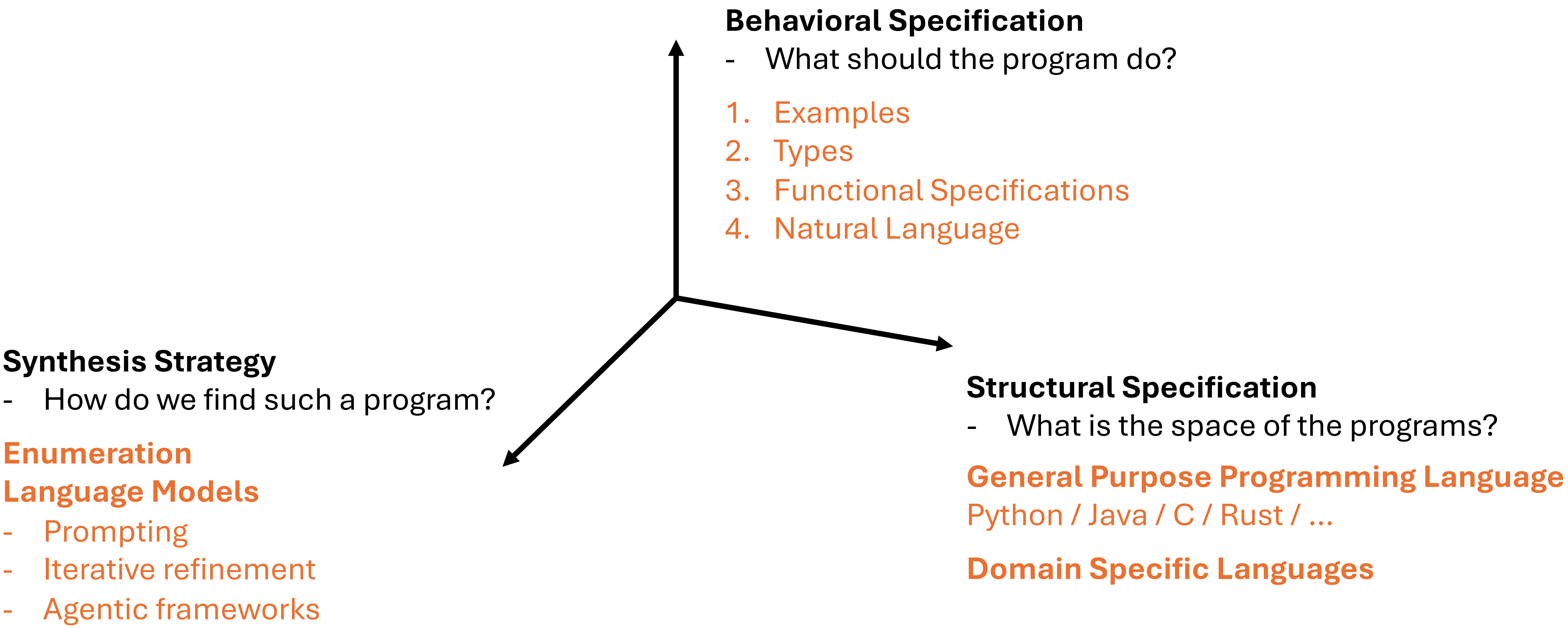
Lecture 12 – Pre-training of Coding Language Models

Ziyang Li

Logistics – Week 7

- Assignment 3
  - https://github.com/machine-programming/assignment-3
  - Releasing tomorrow; due two weeks from now (Oct 23)
- Oral presentation sign up sheet
  - Sent out during the weekend
  - Oral presentation starting on Week 9
- Forming groups for your final projects!
  - Sign up form will be sent out on Thursday
  - Form a group of 2-3 before Next Thursday (Oct 16)

The Course So Far



Behavioral Specification

- What should the program do?

1. Examples
2. Types
3. Functional Specifications
4. Natural Language

Synthesis Strategy

- How do we find such a program?

Enumeration

Language Models

- Prompting
- Iterative refinement
- Agentic frameworks

Structural Specification

- What is the space of the programs?

General Purpose Programming Language

Python / Java / C / Rust / …

Domain Specific Languages

The Course So Far: Synthesis Strategy

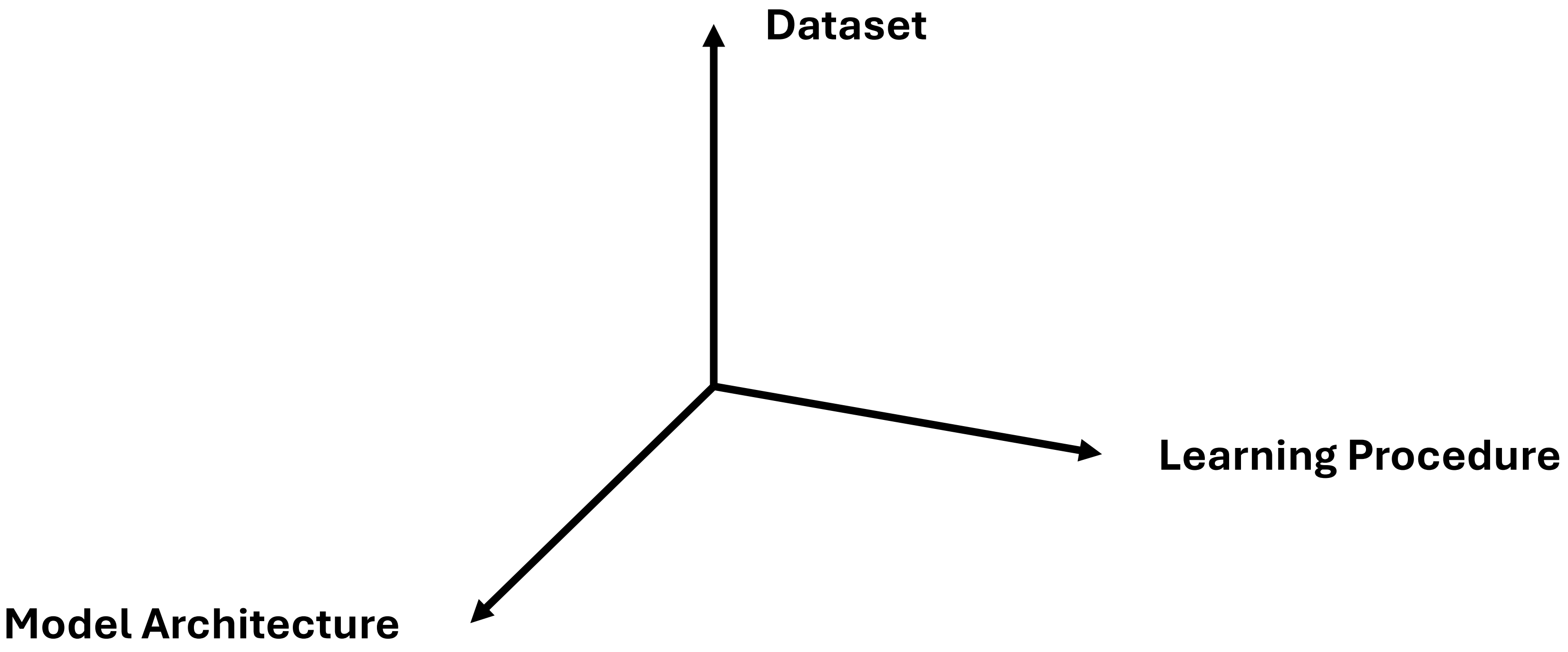
- No prior knowledge
  - Enumerate the entire program space to find the “correct” program
- With prior knowledge: assumes a good enough language model
  - We can query language model to write simple programs
  - We can perform constraint decoding to follow program grammar
  - We can perform prompting strategies to steer language models
  - We can build agentic framework with tools to augment the synthesis

The Course So Far: Synthesis Strategy

- No prior knowledge
  - Enumerate the entire program space to find the “correct” program
- With prior knowledge: assumes a good enough language model
  - We can query language model to write sir
  - We can perform constraint decoding to fo
  - We can perform prompting strategies to steer language models
  - We can build agentic framework with tools to augment the synthesis

How do we obtain a good enough language model?

How to obtain a “good enough” LLM

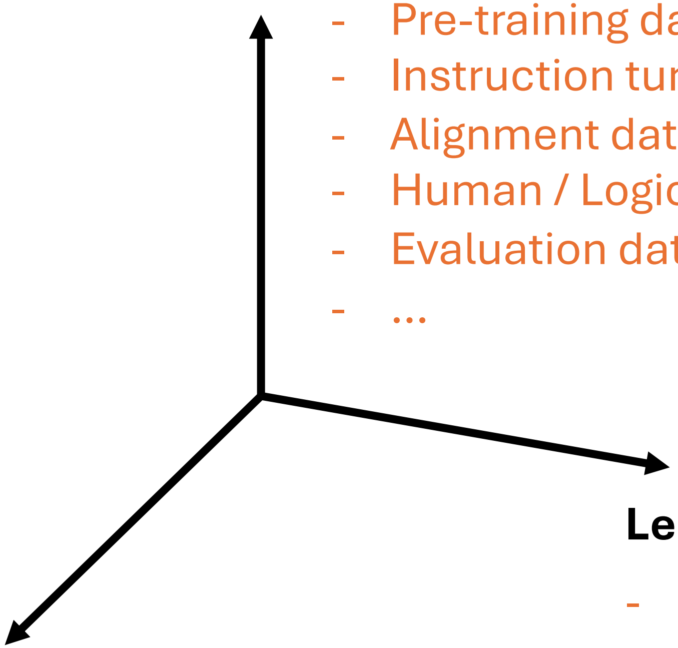


Dataset

Learning Procedure

Model Architecture

How to obtain a “good enough” LLM



Dataset

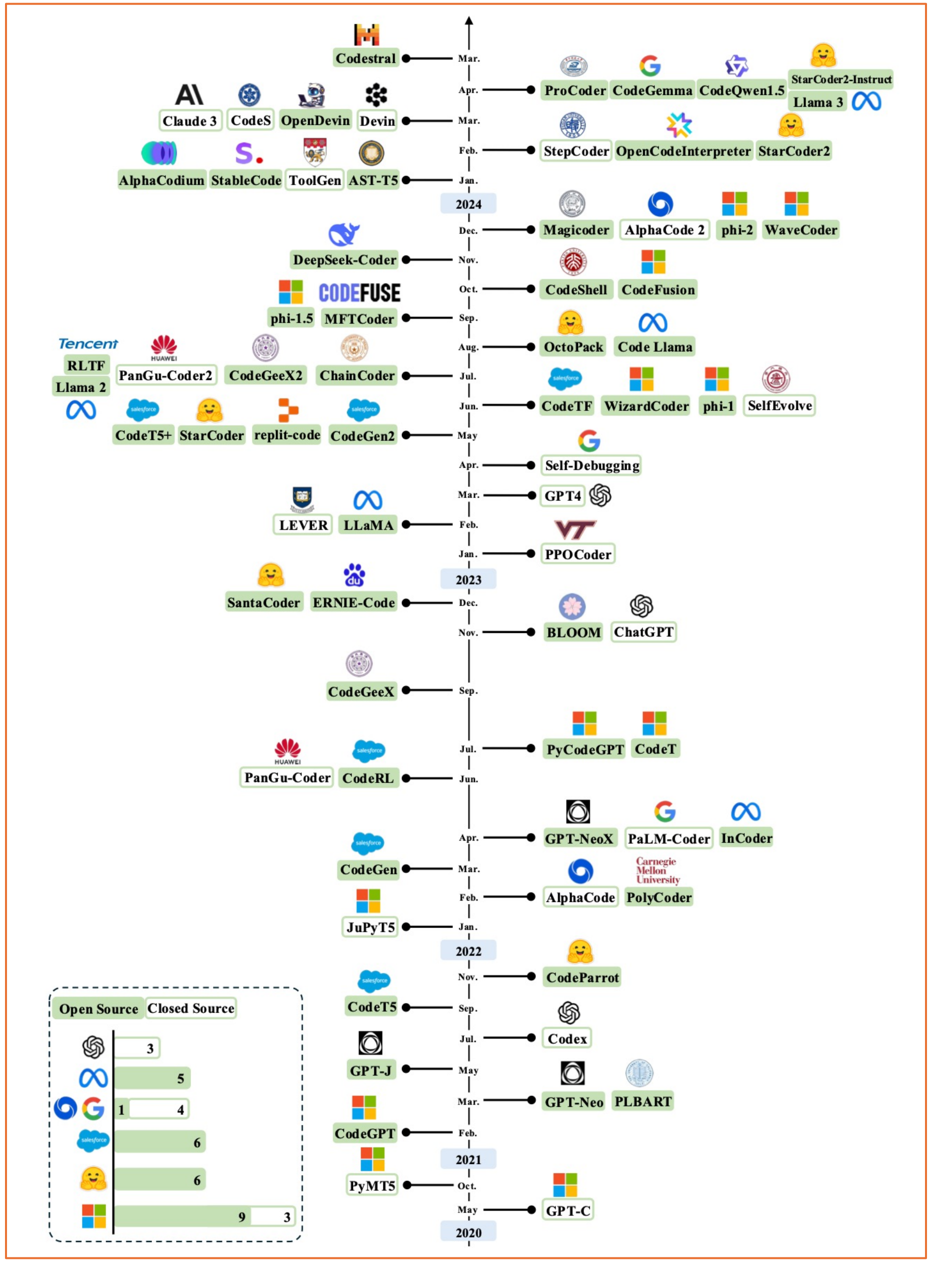
- Pre-training dataset / Fine-tuning dataset
- Instruction tuning dataset
- Alignment dataset
- Human / Logical feedback dataset
- Evaluation dataset
- …

Learning Procedure

- Optimization objectives
- Learning algorithm (SFT, RL, etc.)
- Continual learning, Curriculum learning
- Staged learning
- …

Model Architecture

- Encoder-decoder models
- Decoder-only models
- Hyper-parameter tuning
- …

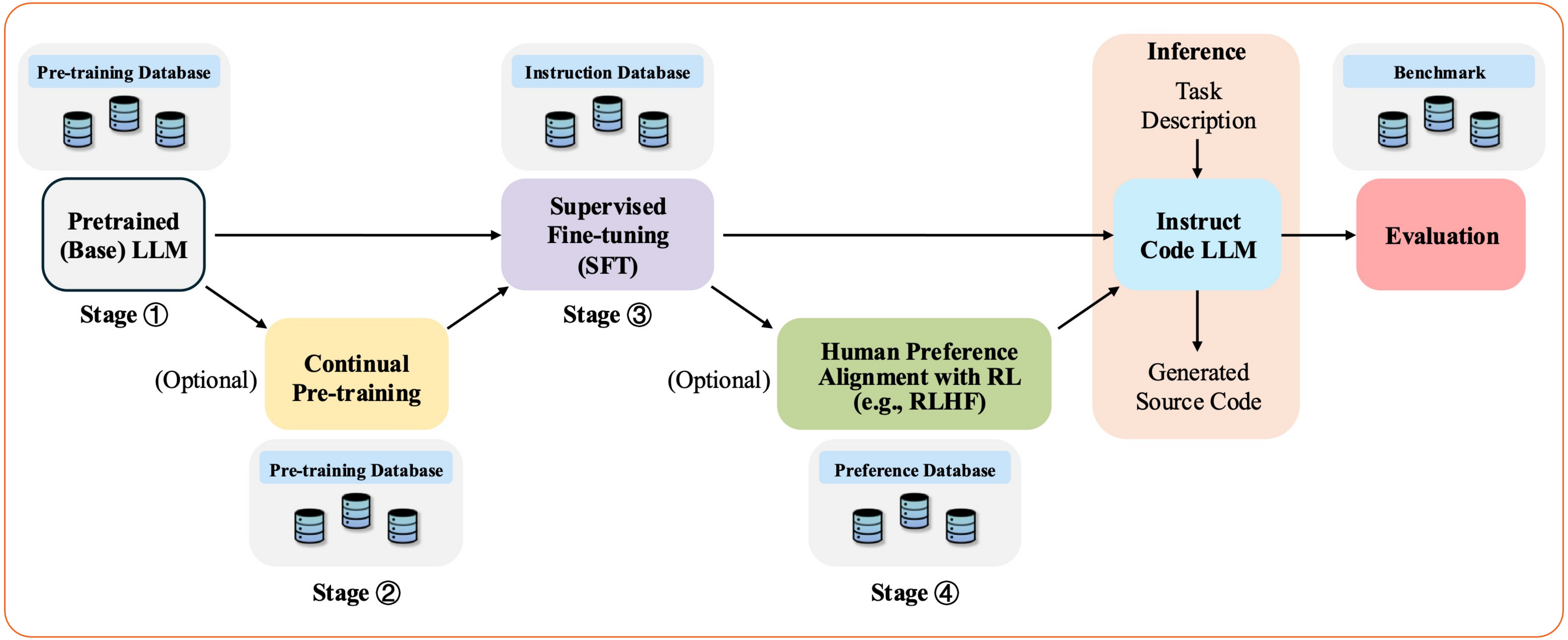


A Survey on Large Language Models for Code Generation, Jiang et al., 2024

Learning Objective

- 🚫 This lecture is NOT about:
  - Memorizing every model name, size, or configuration
  - Ranking models by “who’s best” or “who wins on benchmark X”
  - Treating architecture, objective, or training stage as absolute recipes
  - Chasing transient leaderboard scores or buzzwords
- ✅ This lecture IS about:
  - Grasping the conceptual framework behind how LLMs are trained
  - Developing the skill to read new papers, extract the key ideas, and connect them to broader trends
  - Recognizing trade-offs and design rationales, not just final numbers
  - Building intuition to anticipate and interpret future developments

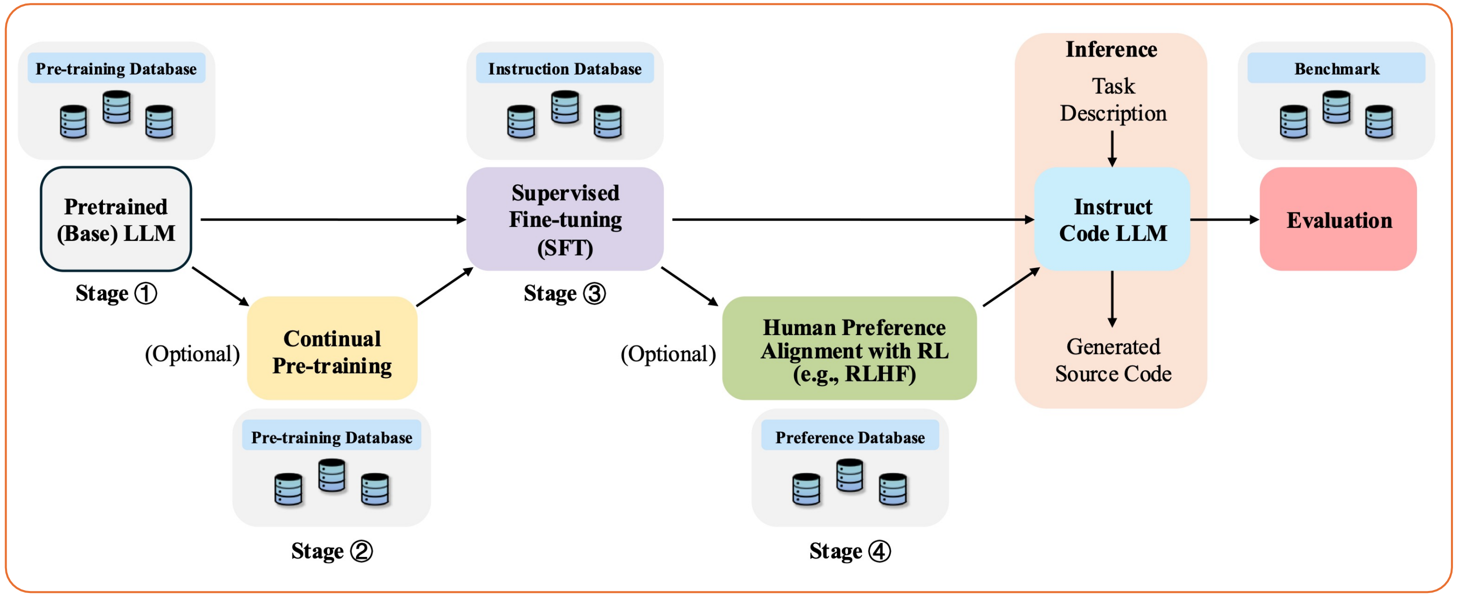
High-level Training, Inference, and Evaluation



A Survey on Large Language Models for Code Generation, Jiang et al., 2024

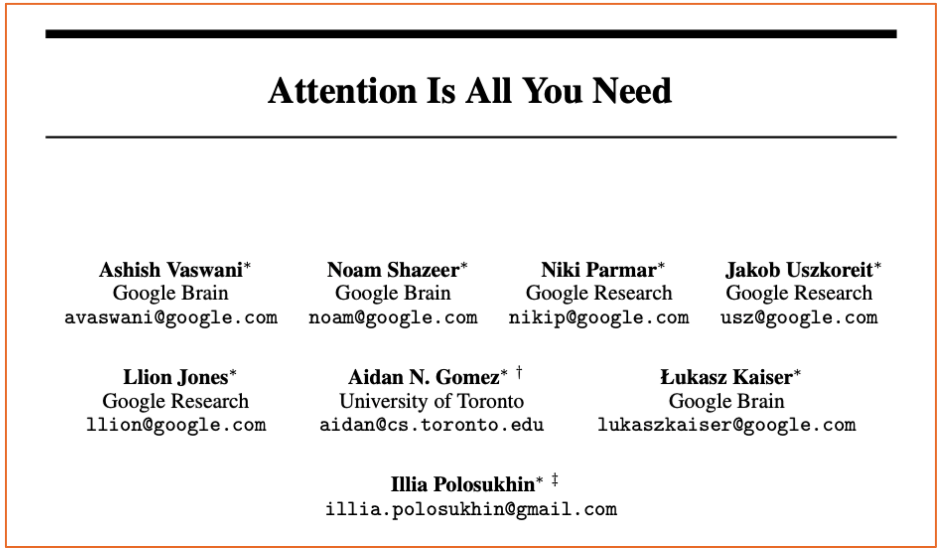
Today’s Agenda

- Pre-training stage
  - Model architecture
  - Pre-training dataset
  - Learning objectives
  - Evaluation dataset
- Special topics
  - Post-training staging
  - Scaling law
  - Hallucination



A Survey on Large Language Models for Code Generation, Jiang et al., 2024

Pre-training: Model Architecture



Attention Is All You Need

Ashish Vaswani[*]
Google Brain
avaswani@google.com

Noam Shazeer[*]
Google Brain
noam@google.com

Niki Parmar[*]
Google Research
nikip@google.com

Jakob Uszkoreit[*]
Google Research
usz@google.com

Llion Jones[*]
Google Research
llion@google.com

Aidan N. Gomez[*][†]
University of Toronto
aidan@cs.toronto.edu

Łukasz Kaiser[*]
Google Brain
lukaszkaiser@google.com

Illia Polosukhin[*][‡]
illia.polosukhin@gmail.com

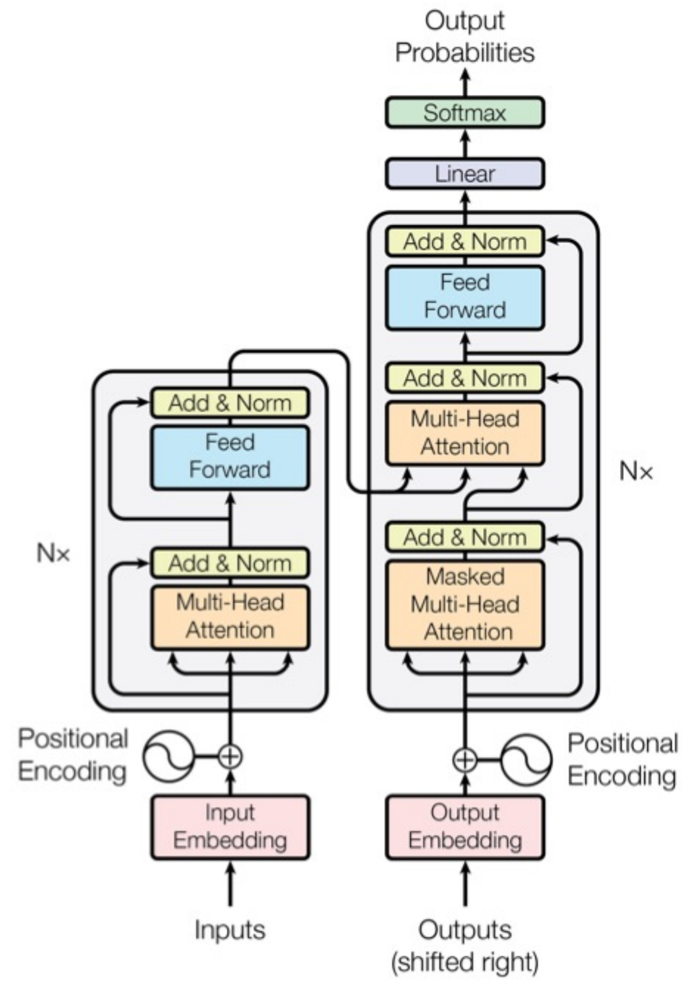


Figure 1: The Transformer - model architecture.

Attention is all you need, Vaswani et al., 2017

Pre-training: Model Architecture

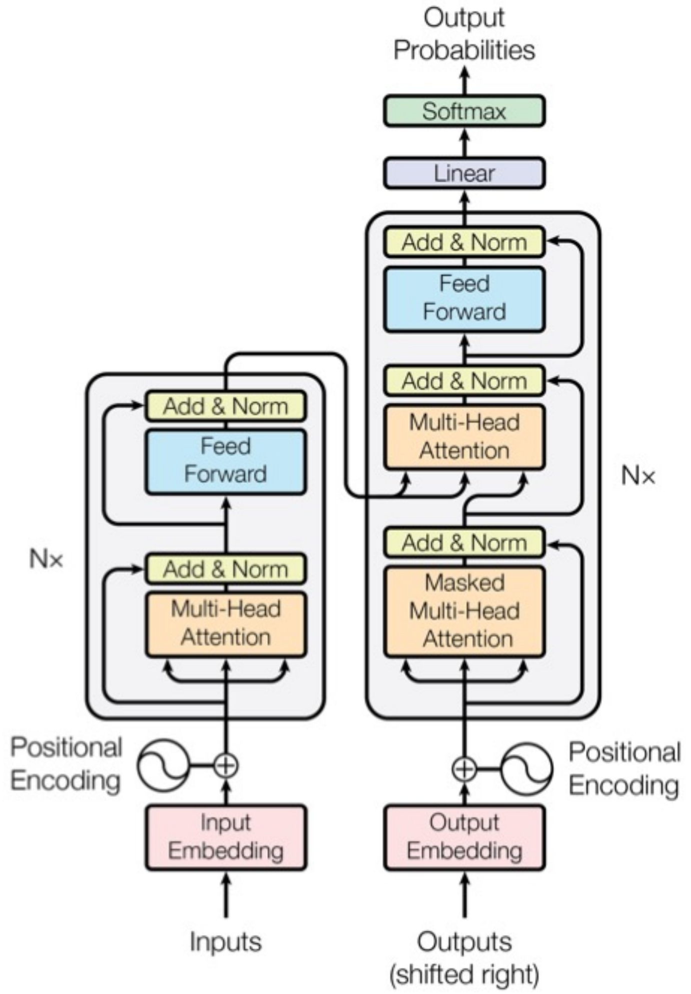
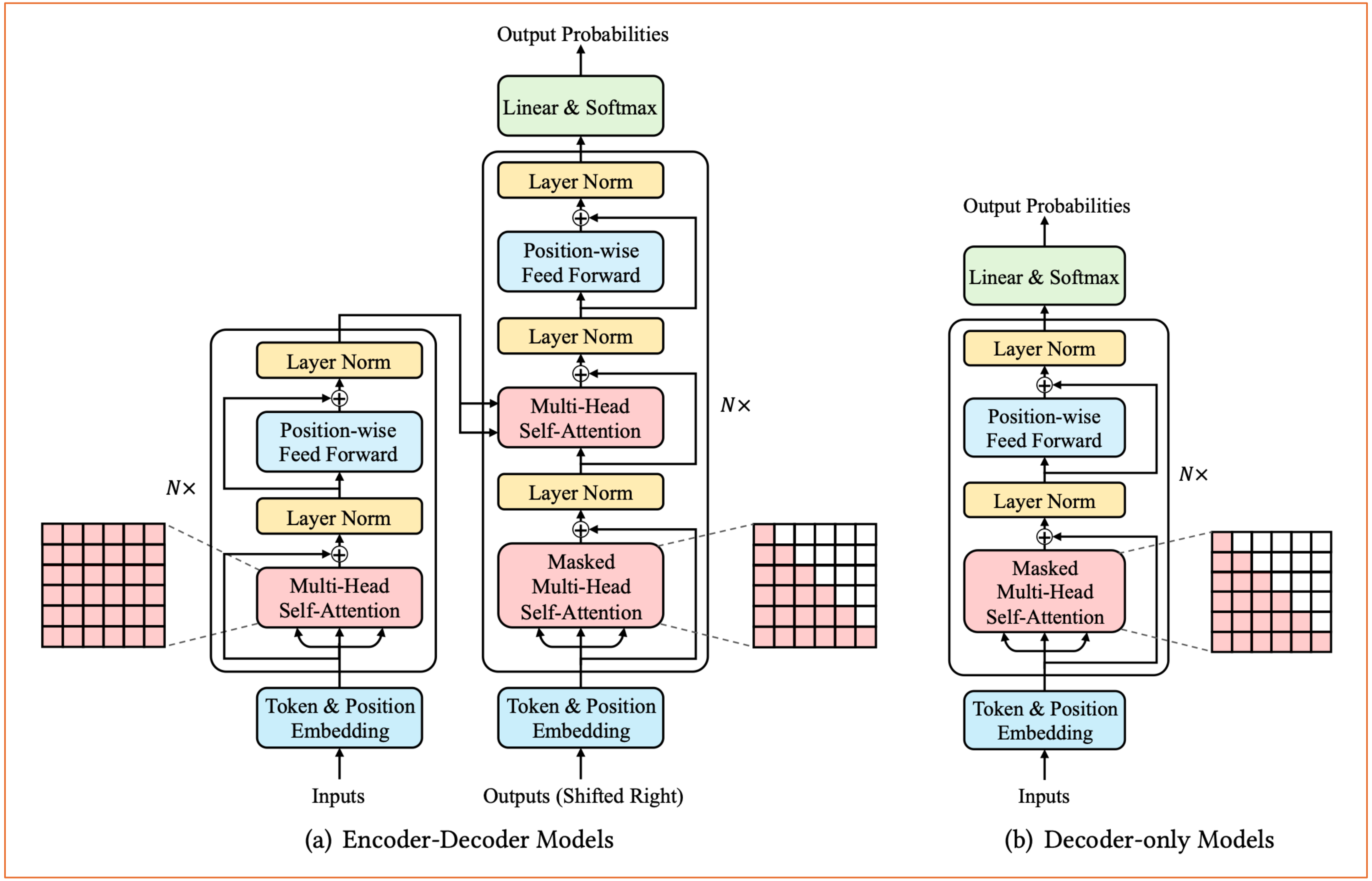


Figure 1: The Transformer - model architecture.

Attention is all you need, Vaswani et al., 2017

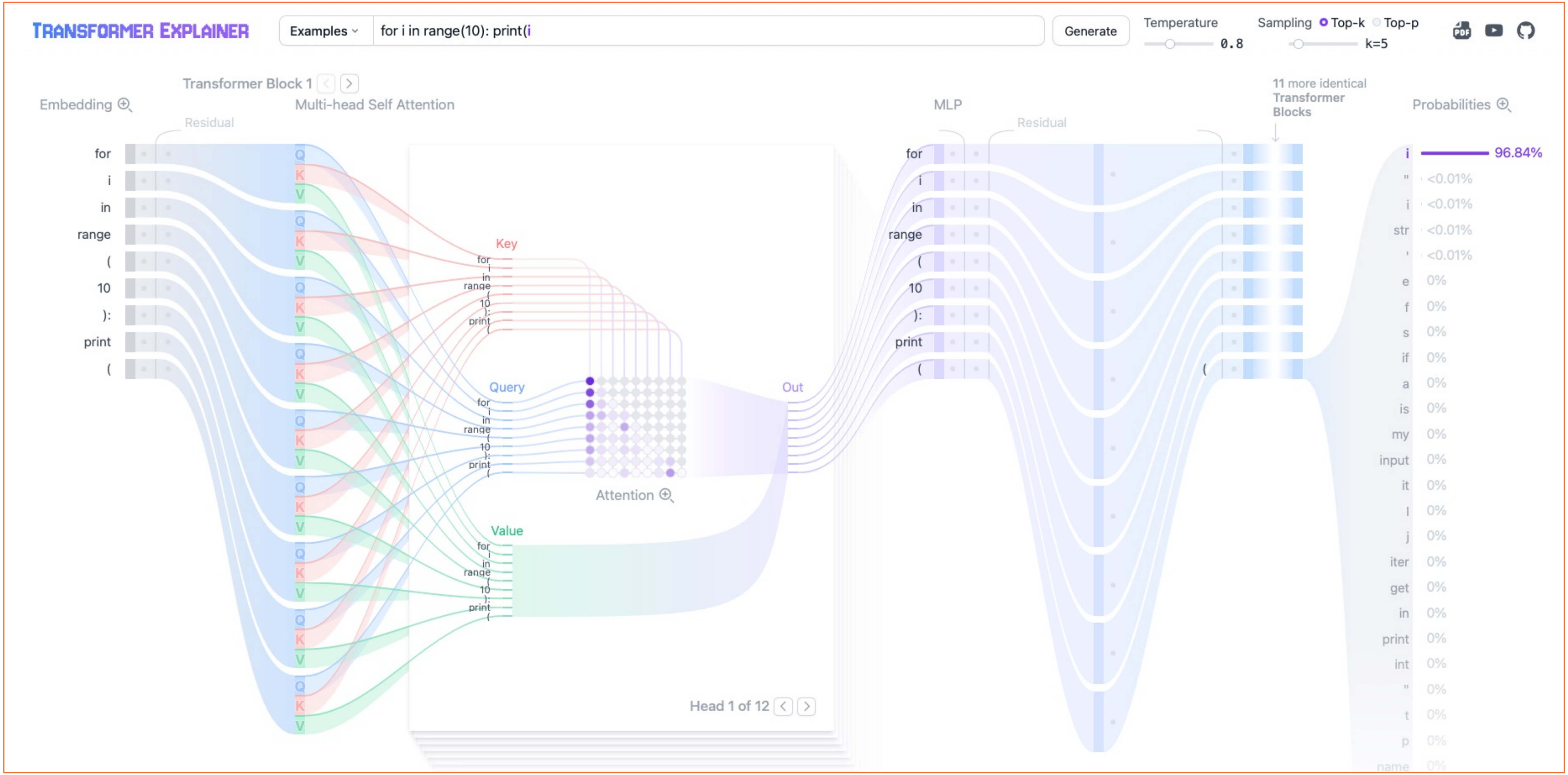
Pre-training: Model Architecture


Output Probabilities
Linear & Softmax
Layer Norm
Position-wise Feed Forward
Layer Norm
Multi-Head Self-Attention
Layer Norm
Masked Multi-Head Self-Attention
Token & Position Embedding
Outputs (Shifted Right)
N×
Layer Norm
Position-wise Feed Forward
Layer Norm
Multi-Head Self-Attention
Token & Position Embedding
Inputs
N×
Output Probabilities
Linear & Softmax
Layer Norm
Position-wise Feed Forward
Layer Norm
Masked Multi-Head Self-Attention
Token & Position Embedding
Inputs
N×

(a) Encoder-Decoder Models

(b) Decoder-only Models

A Survey on Large Language Models for Code Generation, Jiang et al., 2024



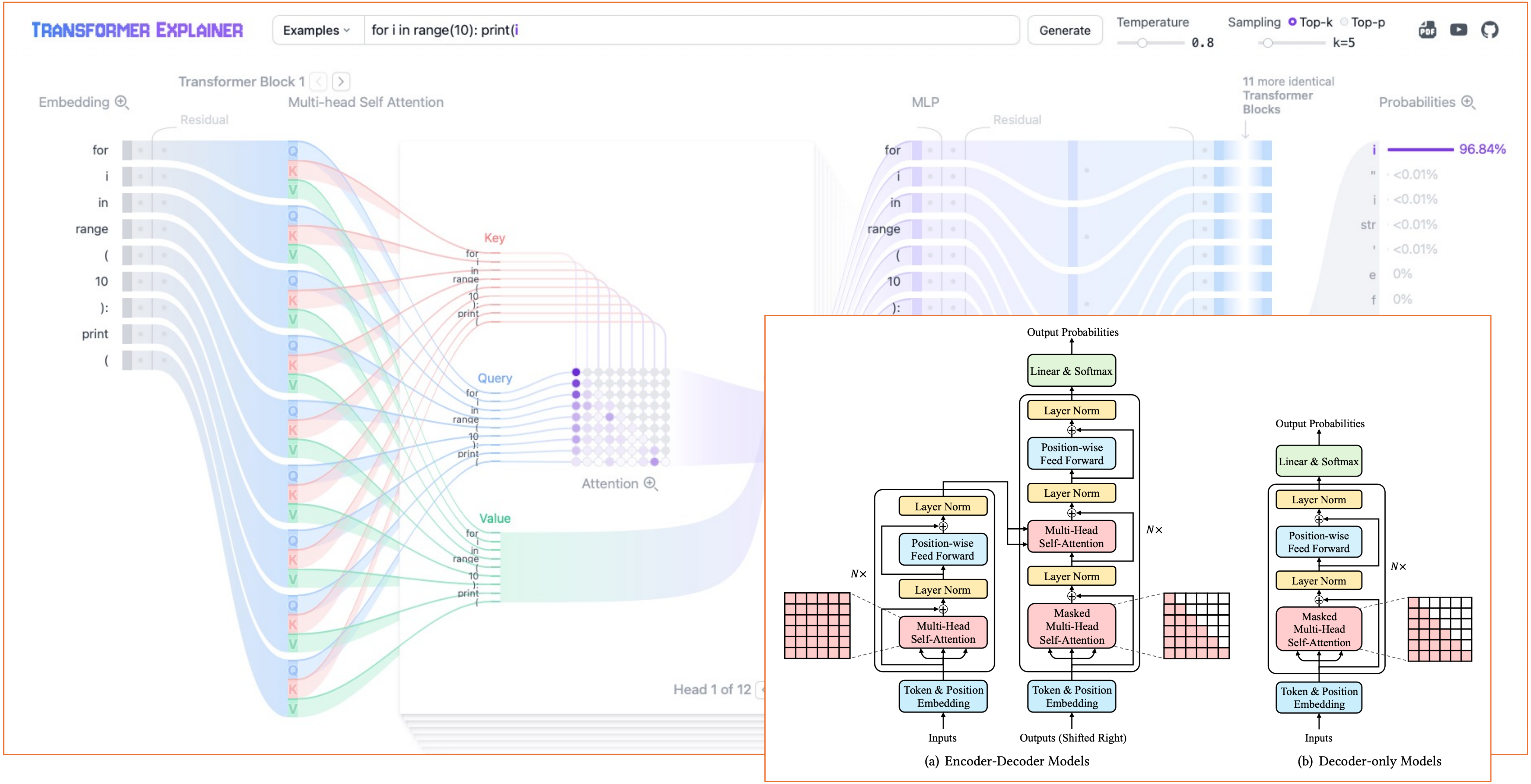
TRANSFORMER EXPLAINER

Examples ⌄ | for i in range(10): print(i | Generate

Temperature 0.8

Sampling ● Top-k ○ Top-p k=5

i	96.84%
"	<0.01%
i	<0.01%
str	<0.01%
'	<0.01%
e	0%
f	0%
s	0%
if	0%
a	0%
is	0%
my	0%
input	0%
it	0%
I	0%
j	0%
iter	0%
get	0%
in	0%
print	0%
int	0%
"	0%
t	0%
p	0%
name	0%



TRANSFORMER EXPLAINER

Examples: for i in range(10): print(i

Generate

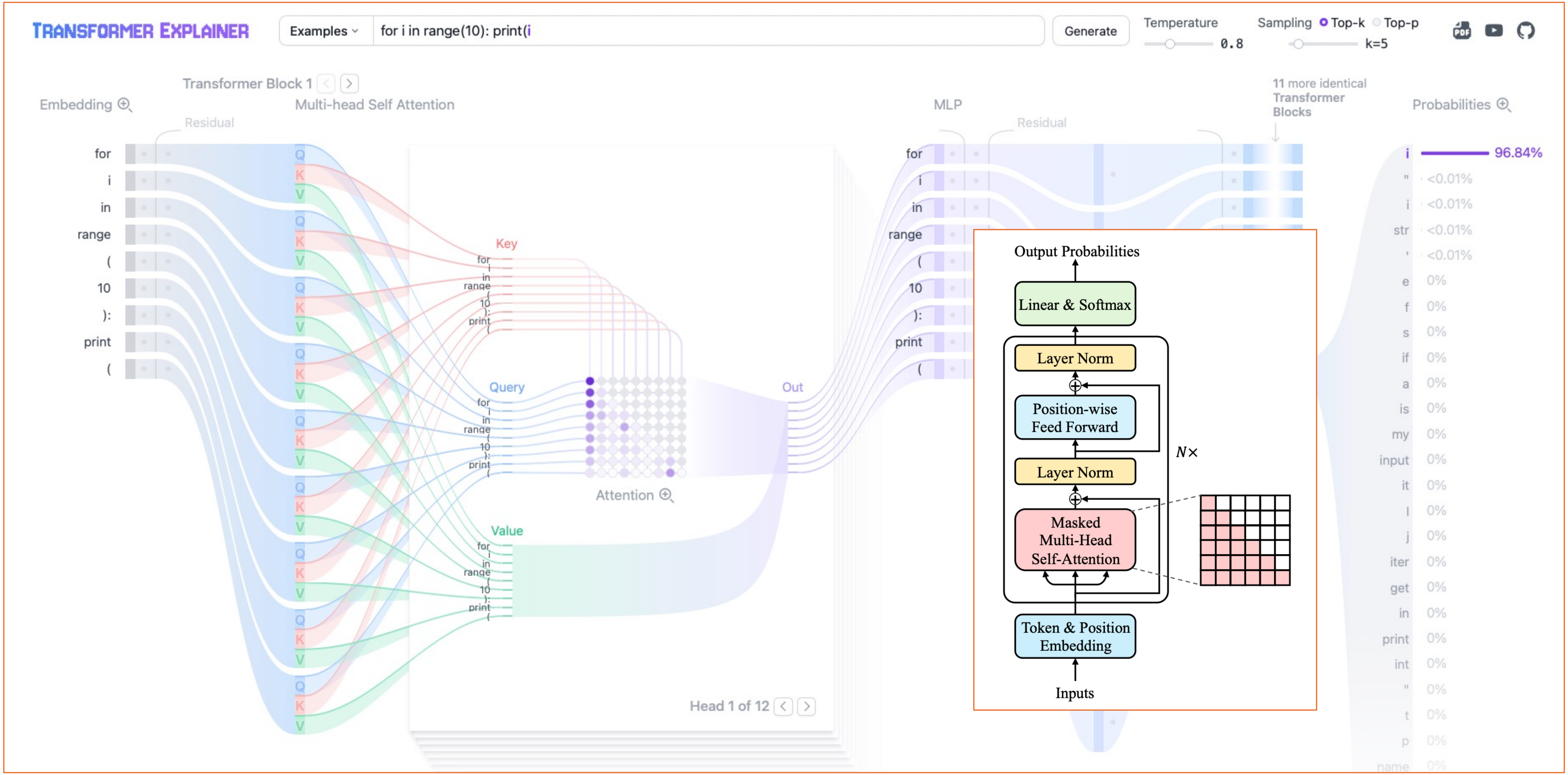
Temperature 0.8

Sampling ● Top-k ○ Top-p k=5

i	96.84%
"	<0.01%
i	<0.01%
str	<0.01%
'	<0.01%
e	0%
f	0%

(a) Encoder-Decoder Models

(b) Decoder-only Models



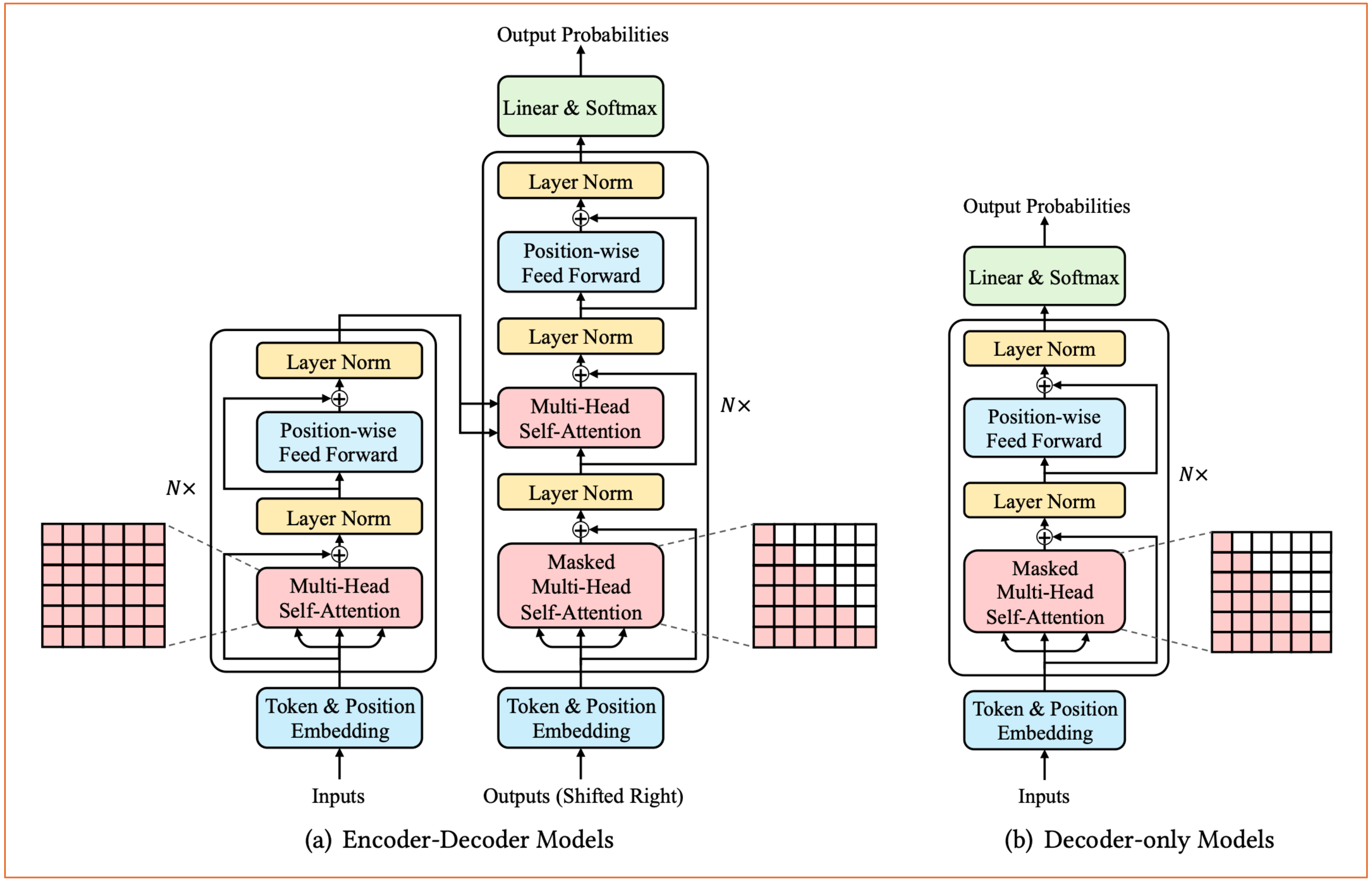
Examples ⌄ for i in range(10): print(i

Generate

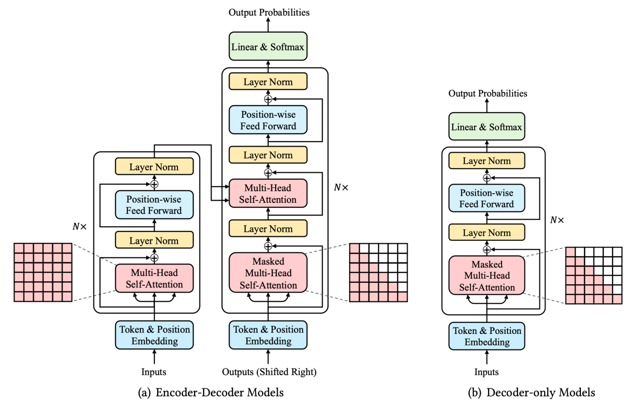
Temperature 0.8

Sampling ● Top-k ○ Top-p k=5

Probabilities	
i	96.84%
"	<0.01%
i	<0.01%
str	<0.01%
'	<0.01%
e	0%
f	0%
s	0%
if	0%
a	0%
is	0%
my	0%
input	0%
it	0%
l	0%
j	0%
iter	0%
get	0%
in	0%
print	0%
int	0%
"	0%
t	0%
p	0%
name	0%


Output Probabilities
Linear & Softmax
Layer Norm
Position-wise Feed Forward
Layer Norm
Multi-Head Self-Attention
Layer Norm
Masked Multi-Head Self-Attention
N×
Layer Norm
Position-wise Feed Forward
Layer Norm
Multi-Head Self-Attention
N×
Token & Position Embedding
Token & Position Embedding
Inputs
Outputs (Shifted Right)
(a) Encoder-Decoder Models
Output Probabilities
Linear & Softmax
Layer Norm
Position-wise Feed Forward
Layer Norm
Masked Multi-Head Self-Attention
N×
Token & Position Embedding
Inputs
(b) Decoder-only Models

A Survey on Large Language Models for Code Generation, Jiang et al., 2024


Output Probabilities
Linear & Softmax
Layer Norm
Position-wise Feed Forward
Layer Norm
Multi-Head Self-Attention
Layer Norm
Masked Multi-Head Self-Attention
Layer Norm
Position-wise Feed Forward
Layer Norm
Multi-Head Self-Attention
N×
N×
Token & Position Embedding
Token & Position Embedding
Inputs
Outputs (Shifted Right)
Output Probabilities
Linear & Softmax
Layer Norm
Position-wise Feed Forward
Layer Norm
Masked Multi-Head Self-Attention
N×
Token & Position Embedding
Inputs

(a) Encoder-Decoder Models

(b) Decoder-only Models

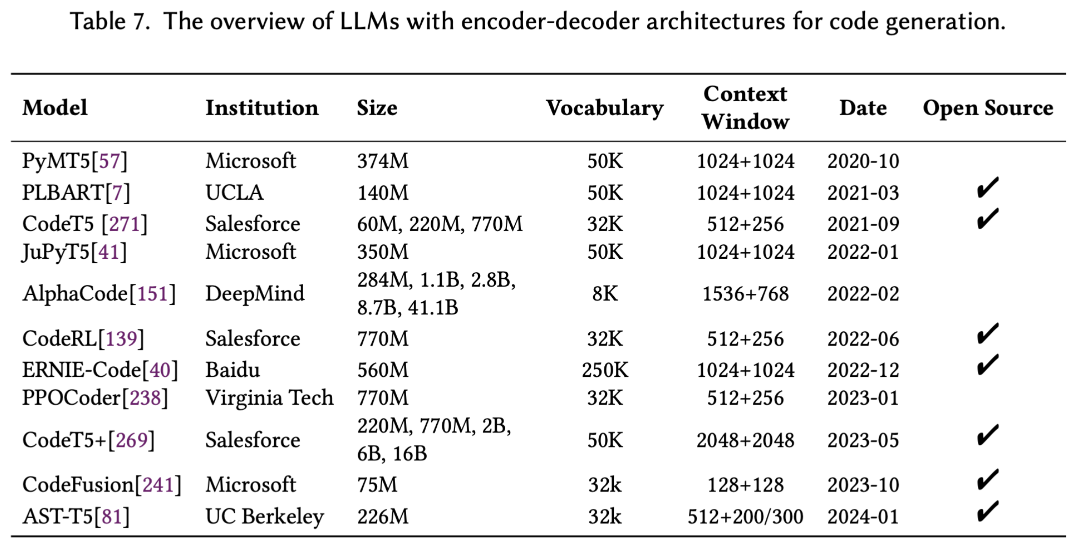


Table 7. The overview of LLMs with encoder-decoder architectures for code generation.

Model	Institution	Size	Vocabulary	Context Window	Date	Open Source
PyMT5[57]	Microsoft	374M	50K	1024+1024	2020-10	
PLBART[7]	UCLA	140M	50K	1024+1024	2021-03	✓
CodeT5 [271]	Salesforce	60M, 220M, 770M	32K	512+256	2021-09	✓
JuPyT5[41]	Microsoft	350M	50K	1024+1024	2022-01	
AlphaCode[151]	DeepMind	284M, 1.1B, 2.8B, 8.7B, 41.1B	8K	1536+768	2022-02	
CodeRL[139]	Salesforce	770M	32K	512+256	2022-06	✓
ERNIE-Code[40]	Baidu	560M	250K	1024+1024	2022-12	✓
PPOCoder[238]	Virginia Tech	770M	32K	512+256	2023-01	
CodeT5+[269]	Salesforce	220M, 770M, 2B, 6B, 16B	50K	2048+2048	2023-05	✓
CodeFusion[241]	Microsoft	75M	32k	128+128	2023-10	✓
AST-T5[81]	UC Berkeley	226M	32k	512+200/300	2024-01	✓

Encoder-decoder architectures

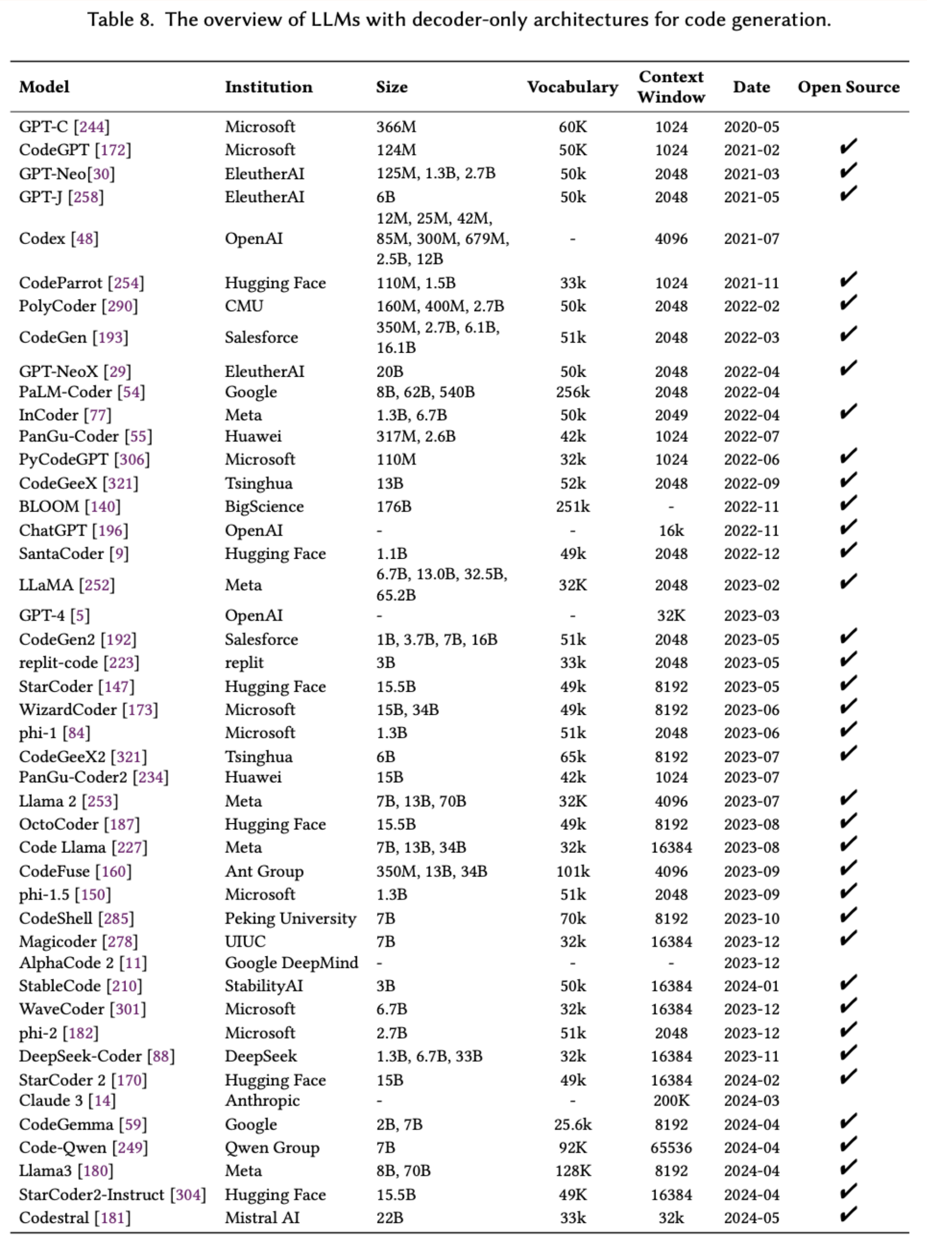


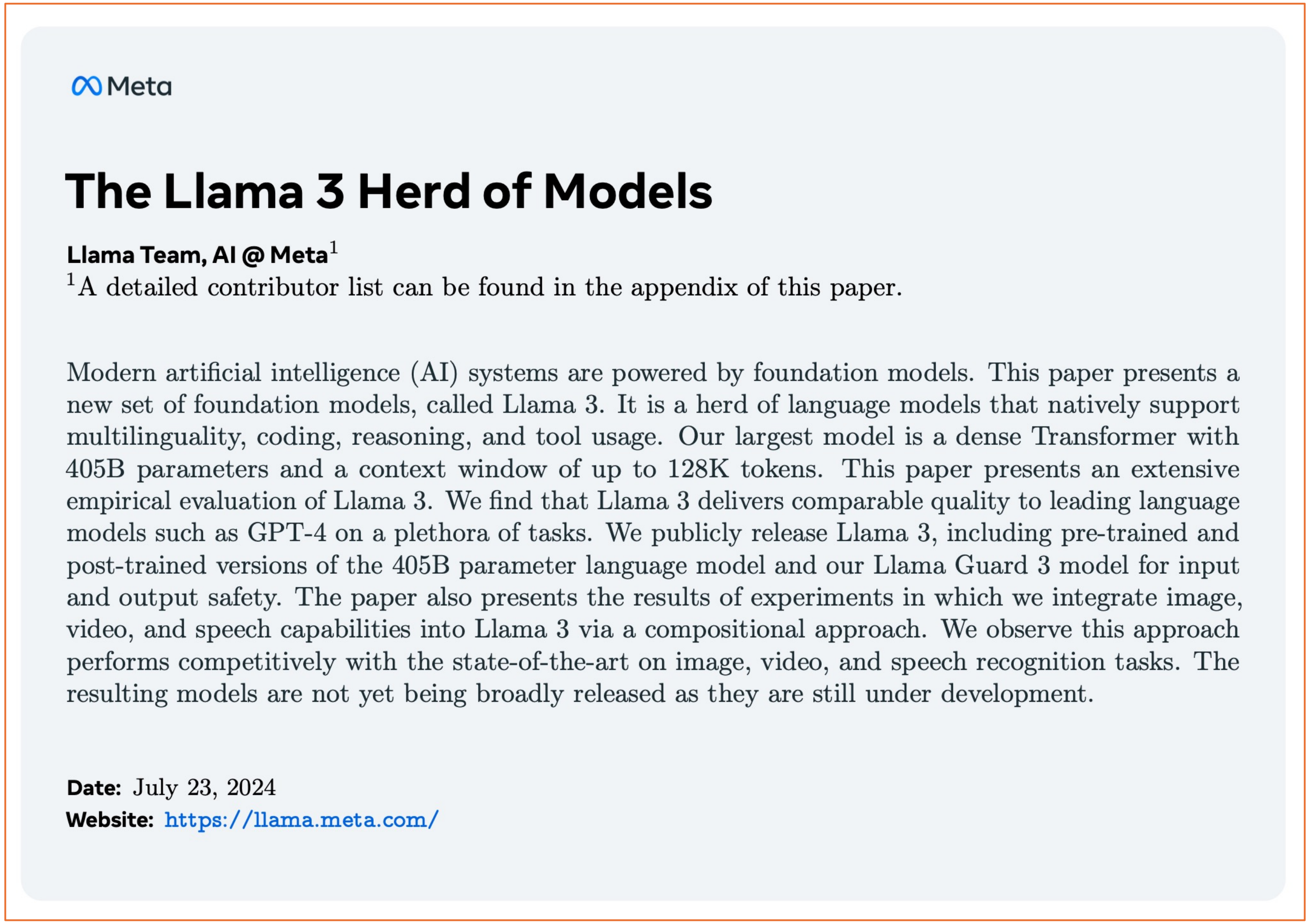
Table 8. The overview of LLMs with decoder-only architectures for code generation.

Model	Institution	Size	Vocabulary	Context Window	Date	Open Source
GPT-C [244]	Microsoft	366M	60K	1024	2020-05	
CodeGPT [172]	Microsoft	124M	50K	1024	2021-02	✓
GPT-Neo[30]	EleutherAI	125M, 1.3B, 2.7B	50k	2048	2021-03	✓
GPT-J [258]	EleutherAI	6B	50k	2048	2021-05	✓
Codex [48]	OpenAI	12M, 25M, 42M, 85M, 300M, 679M, 2.5B, 12B	-	4096	2021-07	
CodeParrot [254]	Hugging Face	110M, 1.5B	33k	1024	2021-11	✓
PolyCoder [290]	CMU	160M, 400M, 2.7B	50k	2048	2022-02	✓
CodeGen [193]	Salesforce	350M, 2.7B, 6.1B, 16.1B	51k	2048	2022-03	✓
GPT-NeoX [29]	EleutherAI	20B	50k	2048	2022-04	✓
PaLM-Coder [54]	Google	8B, 62B, 540B	256k	2048	2022-04	
InCoder [77]	Meta	1.3B, 6.7B	50k	2049	2022-04	✓
PanGu-Coder [55]	Huawei	317M, 2.6B	42k	1024	2022-07	
PyCodeGPT [306]	Microsoft	110M	32k	1024	2022-06	✓
CodeGeeX [321]	Tsinghua	13B	52k	2048	2022-09	✓
BLOOM [140]	BigScience	176B	251k	-	2022-11	✓
ChatGPT [196]	OpenAI	-	-	16k	2022-11	✓
SantaCoder [9]	Hugging Face	1.1B	49k	2048	2022-12	✓
LLaMA [252]	Meta	6.7B, 13.0B, 32.5B, 65.2B	32K	2048	2023-02	✓
GPT-4 [5]	OpenAI	-	-	32K	2023-03	
CodeGen2 [192]	Salesforce	1B, 3.7B, 7B, 16B	51k	2048	2023-05	✓
replit-code [223]	replit	3B	33k	2048	2023-05	✓
StarCoder [147]	Hugging Face	15.5B	49k	8192	2023-05	✓
WizardCoder [173]	Microsoft	15B, 34B	49k	8192	2023-06	✓
phi-1 [84]	Microsoft	1.3B	51k	2048	2023-06	✓
CodeGeeX2 [321]	Tsinghua	6B	65k	8192	2023-07	✓
PanGu-Coder2 [234]	Huawei	15B	42k	1024	2023-07	
Llama 2 [253]	Meta	7B, 13B, 70B	32K	4096	2023-07	✓
OctoCoder [187]	Hugging Face	15.5B	49k	8192	2023-08	✓
Code Llama [227]	Meta	7B, 13B, 34B	32k	16384	2023-08	✓
CodeFuse [160]	Ant Group	350M, 13B, 34B	101k	4096	2023-09	✓
phi-1.5 [150]	Microsoft	1.3B	51k	2048	2023-09	✓
CodeShell [285]	Peking University	7B	70k	8192	2023-10	✓
Magicoder [278]	UIUC	7B	32k	16384	2023-12	✓
AlphaCode 2 [11]	Google DeepMind	-	-	-	2023-12	
StableCode [210]	StabilityAI	3B	50k	16384	2024-01	✓
WaveCoder [301]	Microsoft	6.7B	32k	16384	2023-12	✓
phi-2 [182]	Microsoft	2.7B	51k	2048	2023-12	✓
DeepSeek-Coder [88]	DeepSeek	1.3B, 6.7B, 33B	32k	16384	2023-11	✓
StarCoder 2 [170]	Hugging Face	15B	49k	16384	2024-02	✓
Claude 3 [14]	Anthropic	-	-	200K	2024-03	
CodeGemma [59]	Google	2B, 7B	25.6k	8192	2024-04	✓
Code-Qwen [249]	Qwen Group	7B	92K	65536	2024-04	✓
Llama3 [180]	Meta	8B, 70B	128K	8192	2024-04	✓
StarCoder2-Instruct [304]	Hugging Face	15.5B	49K	16384	2024-04	✓
Codestral [181]	Mistral AI	22B	33k	32k	2024-05	✓

Decoder-only architectures

A Survey on Large Language Models for Code Generation, Jiang et al., 2024

Model Architecture Case Study: Llama 3



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

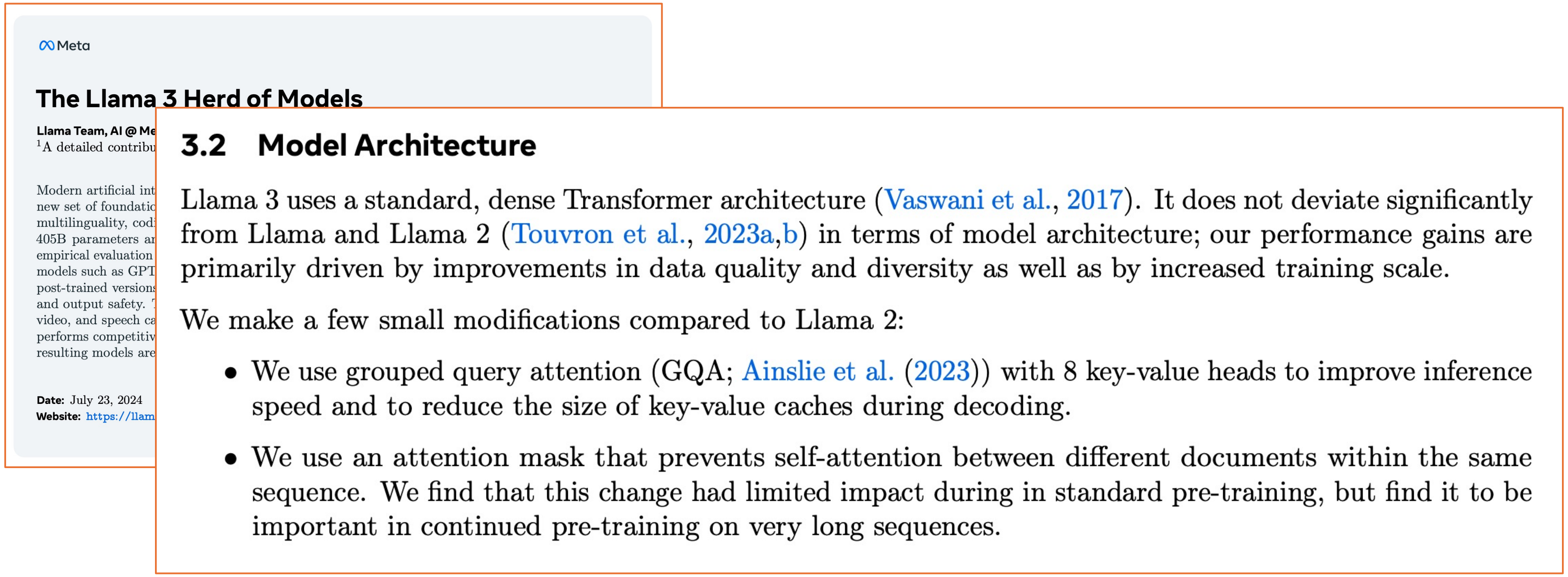
[1] A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Model Architecture Case Study: Llama 3



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Me

[1]A detailed contribu

Modern artificial int
new set of foundatio
multilinguality, codi
405B parameters ar
empirical evaluation
models such as GPT
post-trained versions
and output safety. T
video, and speech ca
performs competitiv
resulting models are

Date: July 23, 2024
Website: https://llam

3.2 Model Architecture

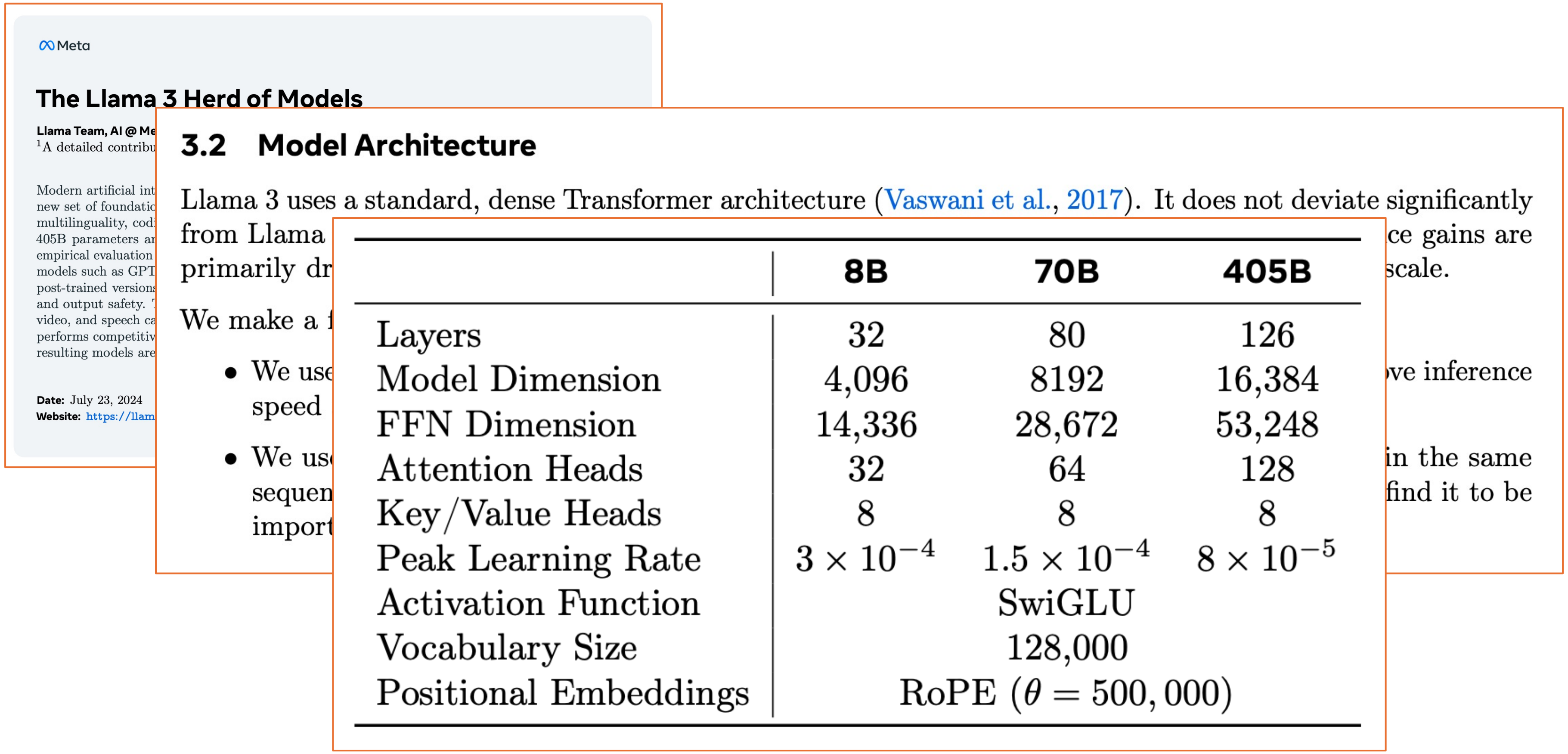
Llama 3 uses a standard, dense Transformer architecture (Vaswani et al., 2017). It does not deviate significantly from Llama and Llama 2 (Touvron et al., 2023a,b) in terms of model architecture; our performance gains are primarily driven by improvements in data quality and diversity as well as by increased training scale.

We make a few small modifications compared to Llama 2:

- We use grouped query attention (GQA; Ainslie et al. (2023)) with 8 key-value heads to improve inference speed and to reduce the size of key-value caches during decoding.
- We use an attention mask that prevents self-attention between different documents within the same sequence. We find that this change had limited impact during in standard pre-training, but find it to be important in continued pre-training on very long sequences.

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024

Model Architecture Case Study: Llama 3



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Me
[1]A detailed contribu

Modern artificial int
new set of foundatio
multilinguality, codi
405B parameters ar
empirical evaluation
models such as GPT
post-trained versions
and output safety. T
video, and speech ca
performs competitiv
resulting models are

Date: July 23, 2024
Website: https://llam

3.2 Model Architecture

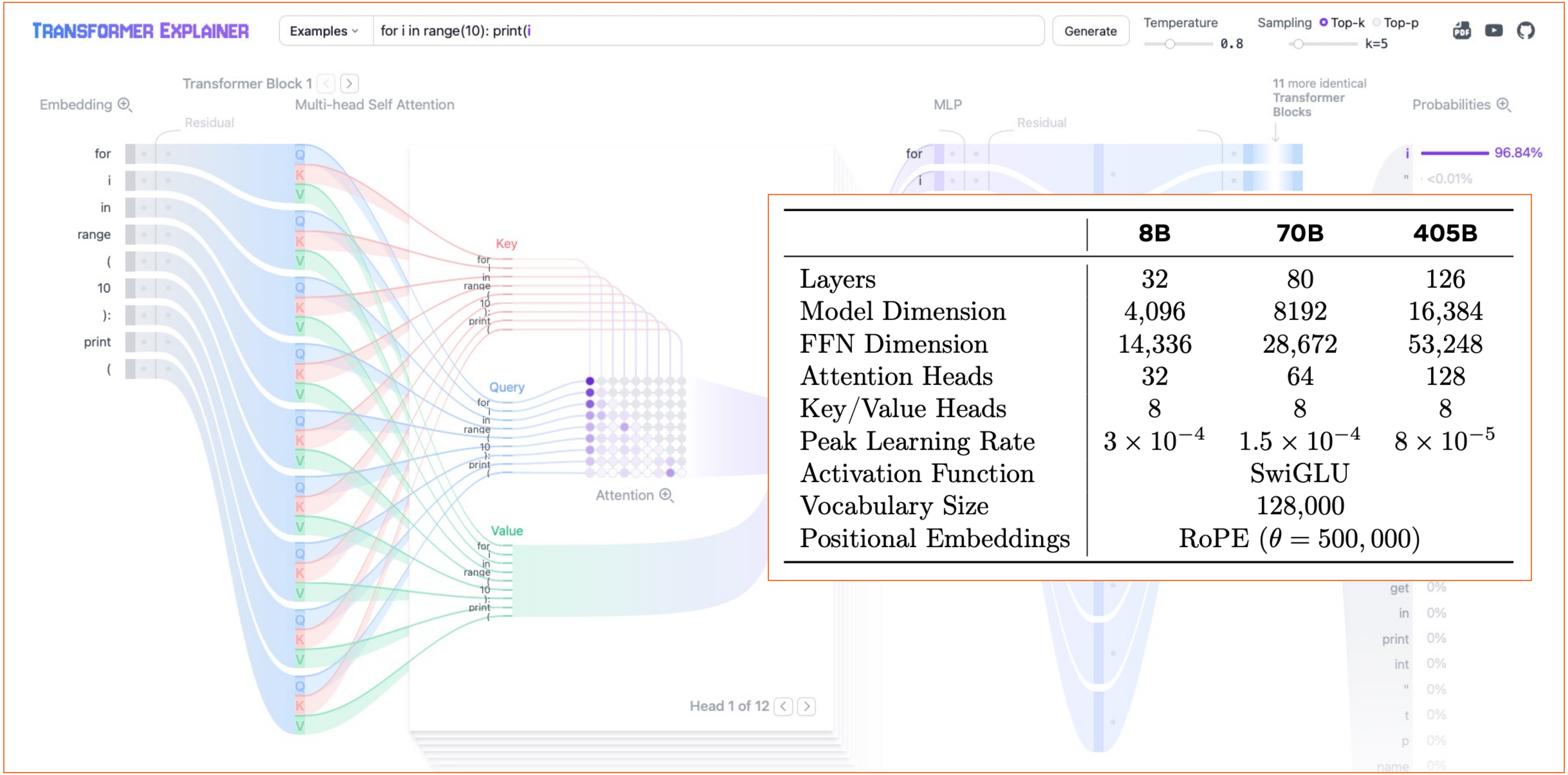
Llama 3 uses a standard, dense Transformer architecture (Vaswani et al., 2017). It does not deviate significantly
from Llama … ce gains are
primarily dr … scale.

We make a f …

- We use … ve inference
speed
- We us … in the same
sequen … find it to be
import

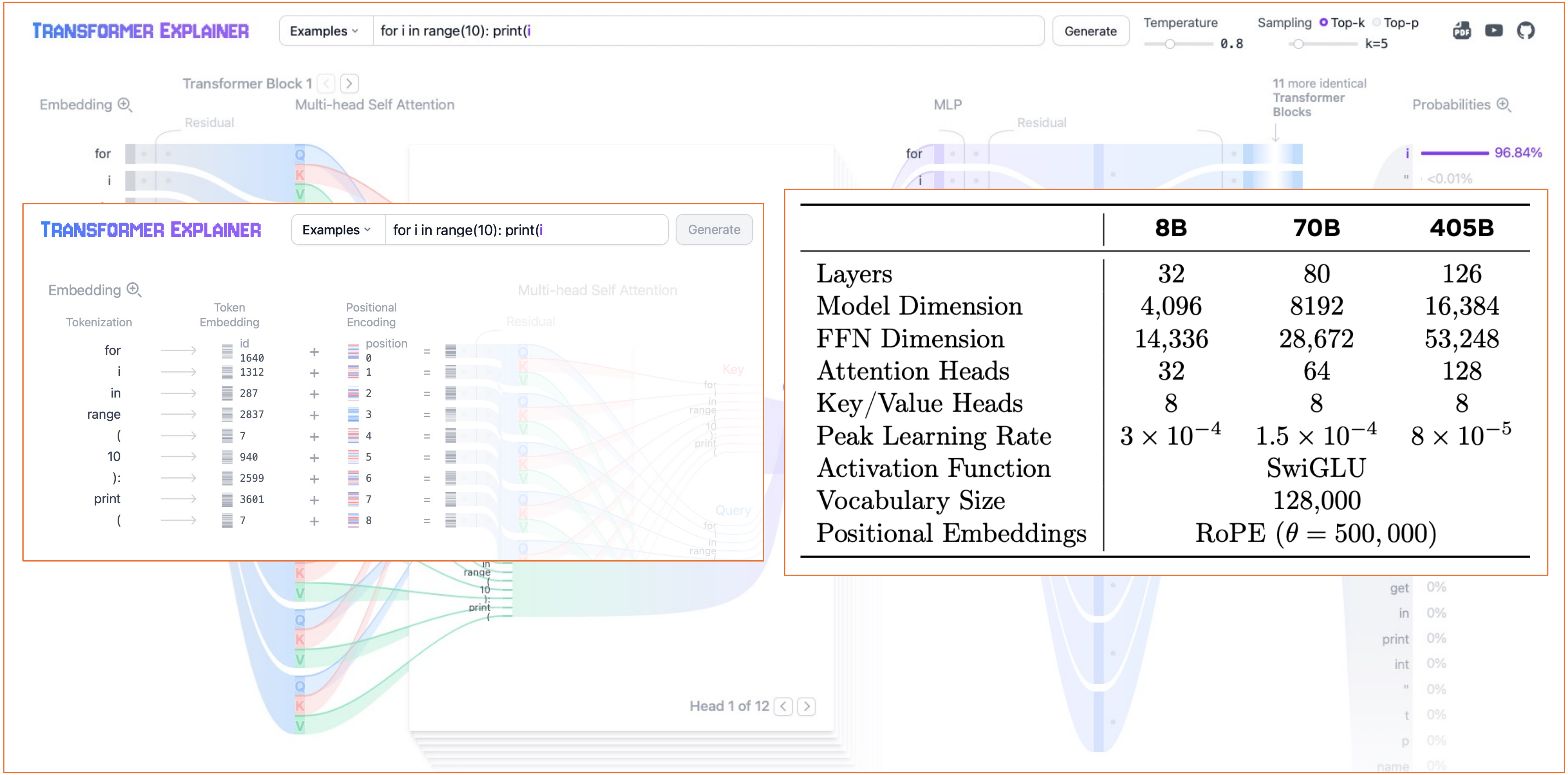
	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



for i in range(10): print(i

Temperature 0.8

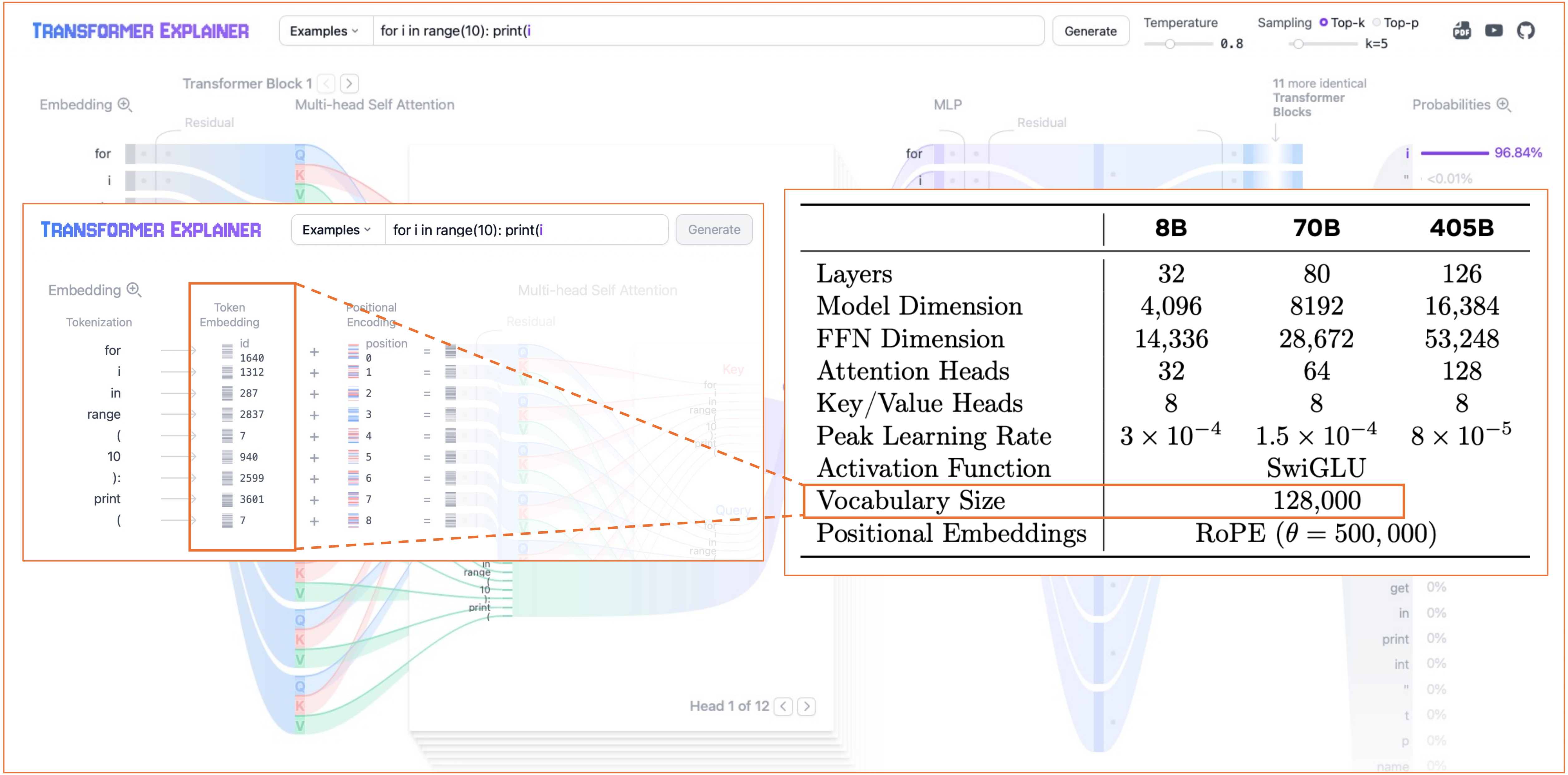
Sampling ● Top-k ○ Top-p k=5

i	96.84%
"	<0.01%
get	0%
in	0%
print	0%
int	0%
"	0%
t	0%
p	0%
name	0%

Tokenization	Token Embedding id	Positional Encoding position
for	1640	0
i	1312	1
in	287	2
range	2837	3
(	7	4
10	940	5
):	2599	6
print	3601	7
(	7	8

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ($\theta = 500,000$)	

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



for i in range(10): print(i

Temperature 0.8

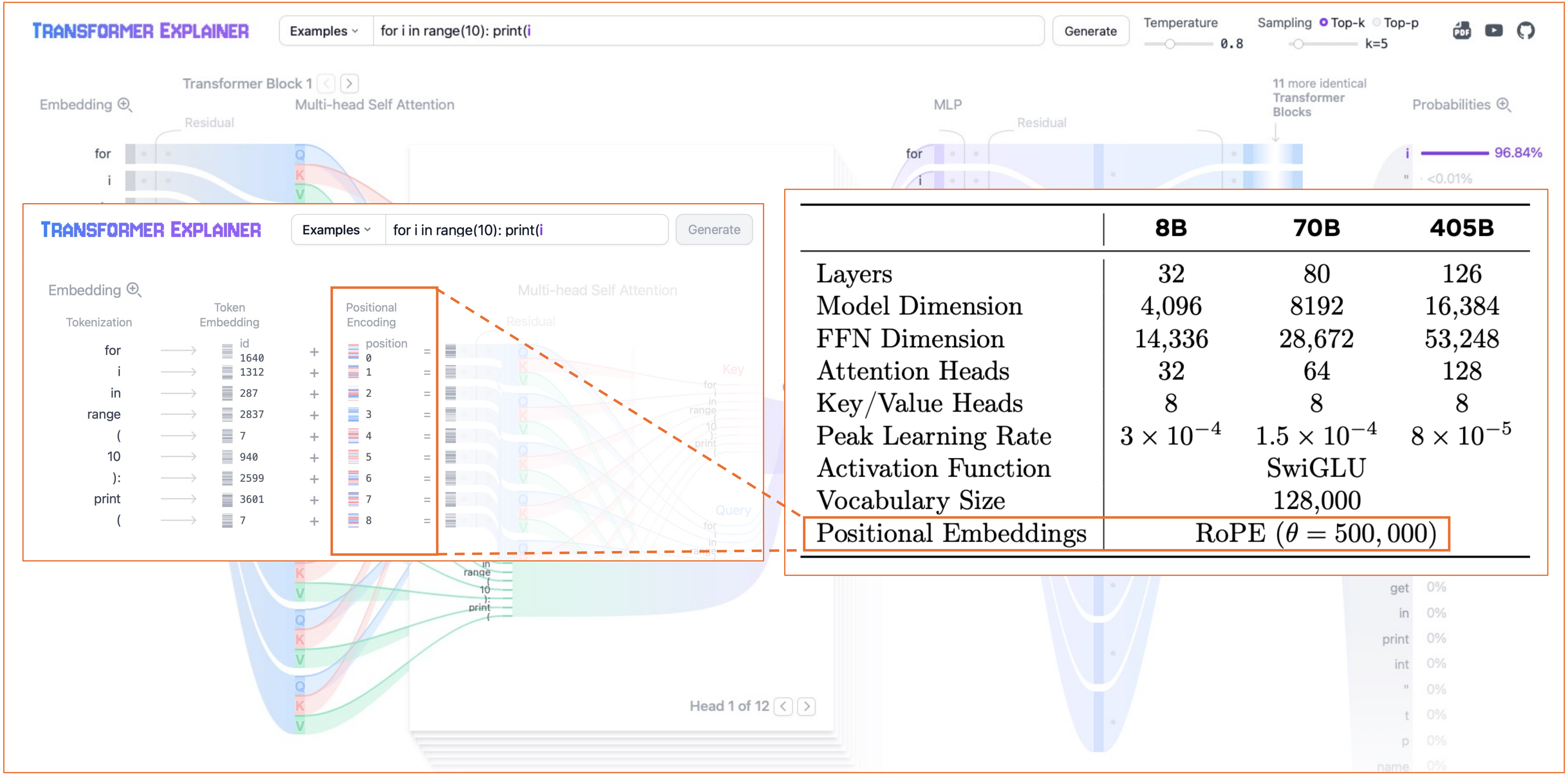
Sampling ● Top-k ○ Top-p

k=5

Tokenization	Token Embedding id	Positional Encoding position
for	1640	0
i	1312	1
in	287	2
range	2837	3
(	7	4
10	940	5
):	2599	6
print	3601	7
(	7	8

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



for i in range(10): print(i

Temperature 0.8

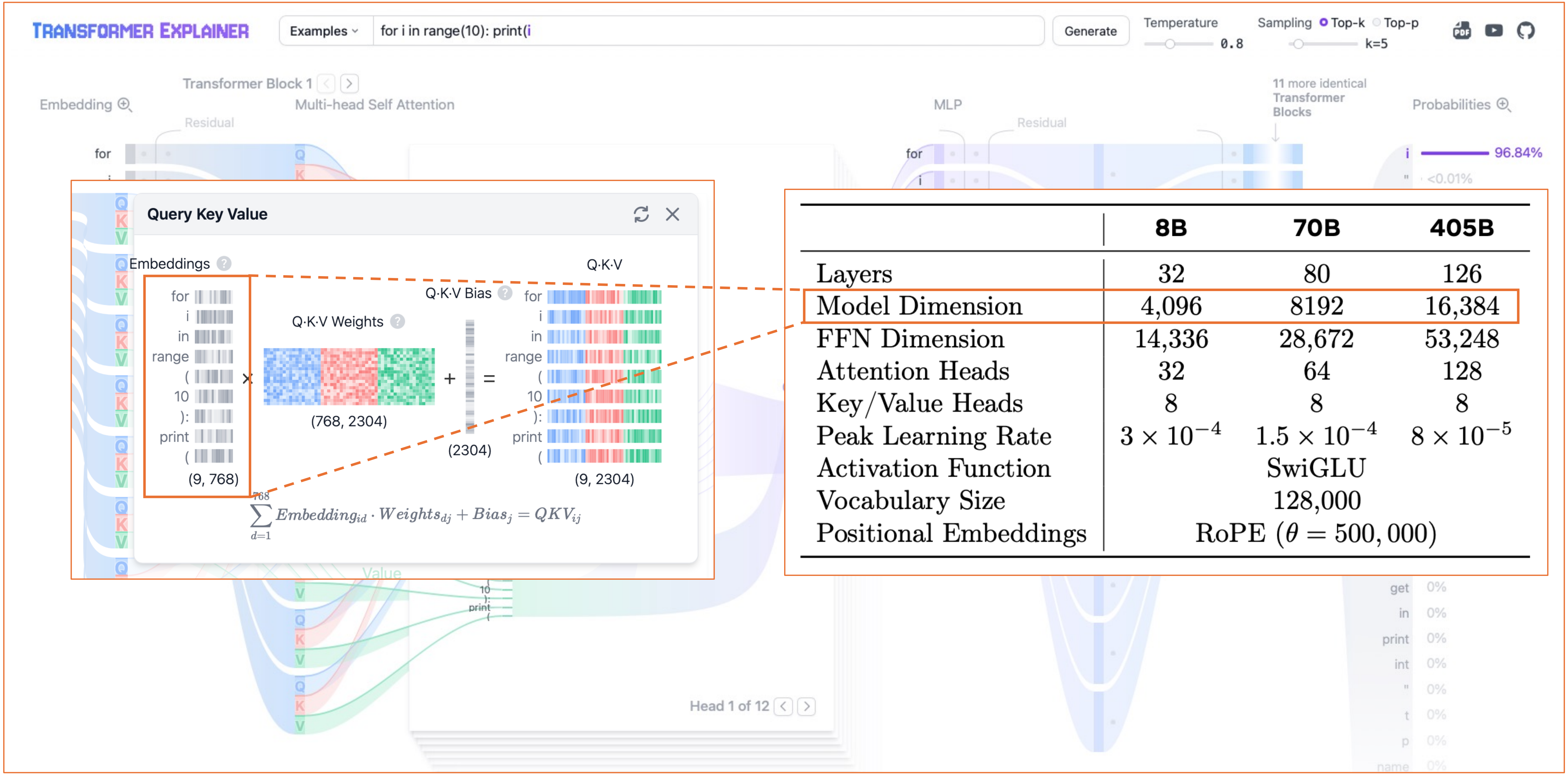
Sampling ● Top-k ○ Top-p k=5

Tokenization	Token Embedding id	Positional Encoding position
for	1640	0
i	1312	1
in	287	2
range	2837	3
(	7	4
10	940	5
):	2599	6
print	3601	7
(	7	8

Probabilities	
i	96.84%
"	<0.01%
get	0%
in	0%
print	0%
int	0%
"	0%
t	0%
p	0%
name	0%

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ($\theta = 500,000$)	

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



TRANSFORMER EXPLAINER

Examples ⌄ | for i in range(10): print(i | Generate

Temperature 0.8

Sampling ● Top-k ○ Top-p

k=5

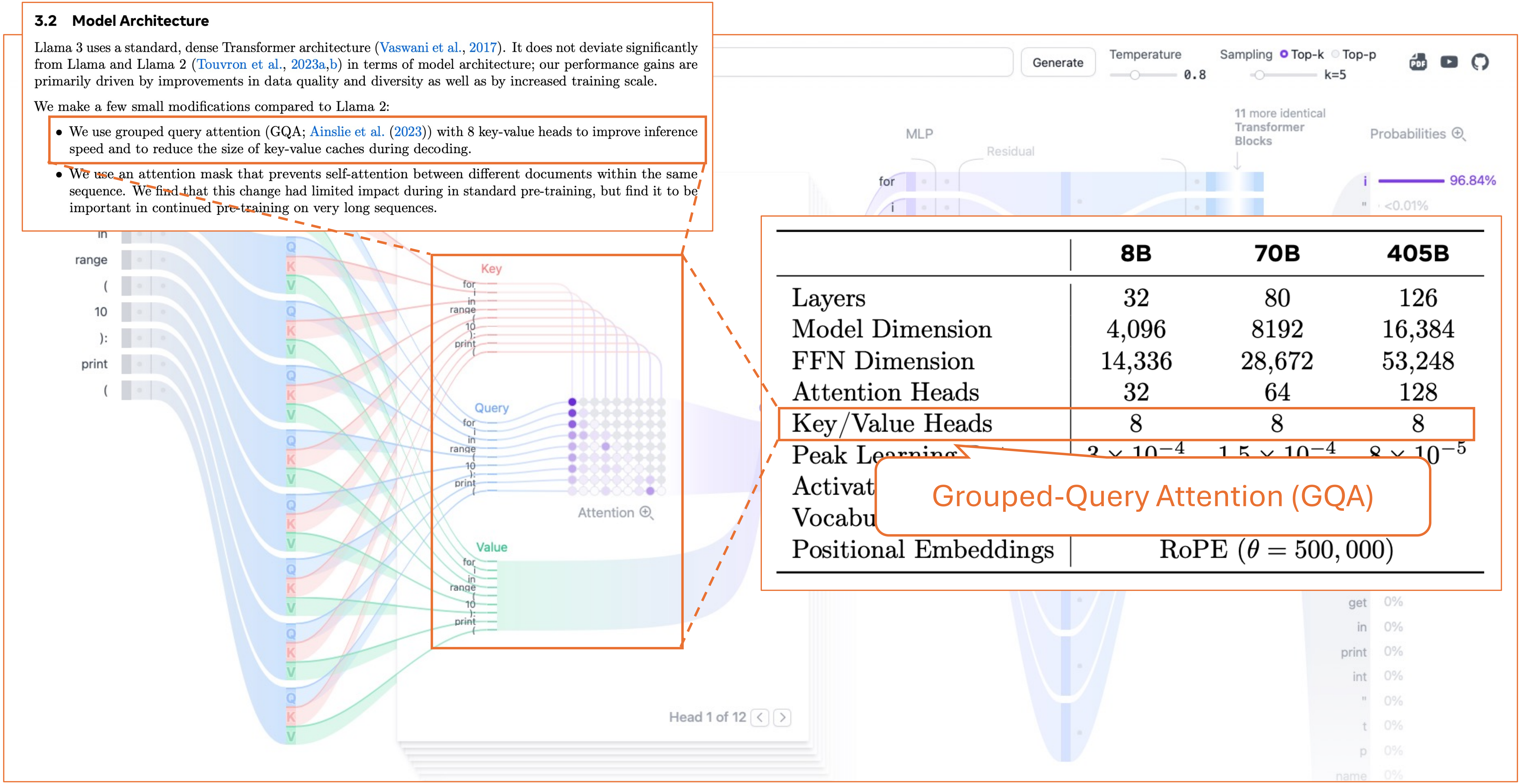
Query Key Value

Embeddings

$$\sum_{d=1}^{768} Embedding_{id} \cdot Weights_{dj} + Bias_j = QKV_{ij}$$

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



3.2 Model Architecture

Llama 3 uses a standard, dense Transformer architecture (Vaswani et al., 2017). It does not deviate significantly from Llama and Llama 2 (Touvron et al., 2023a,b) in terms of model architecture; our performance gains are primarily driven by improvements in data quality and diversity as well as by increased training scale.

We make a few small modifications compared to Llama 2:

- We use grouped query attention (GQA; Ainslie et al. (2023)) with 8 key-value heads to improve inference speed and to reduce the size of key-value caches during decoding.
- We use an attention mask that prevents self-attention between different documents within the same sequence. We find that this change had limited impact during in standard pre-training, but find it to be important in continued pre-training on very long sequences.

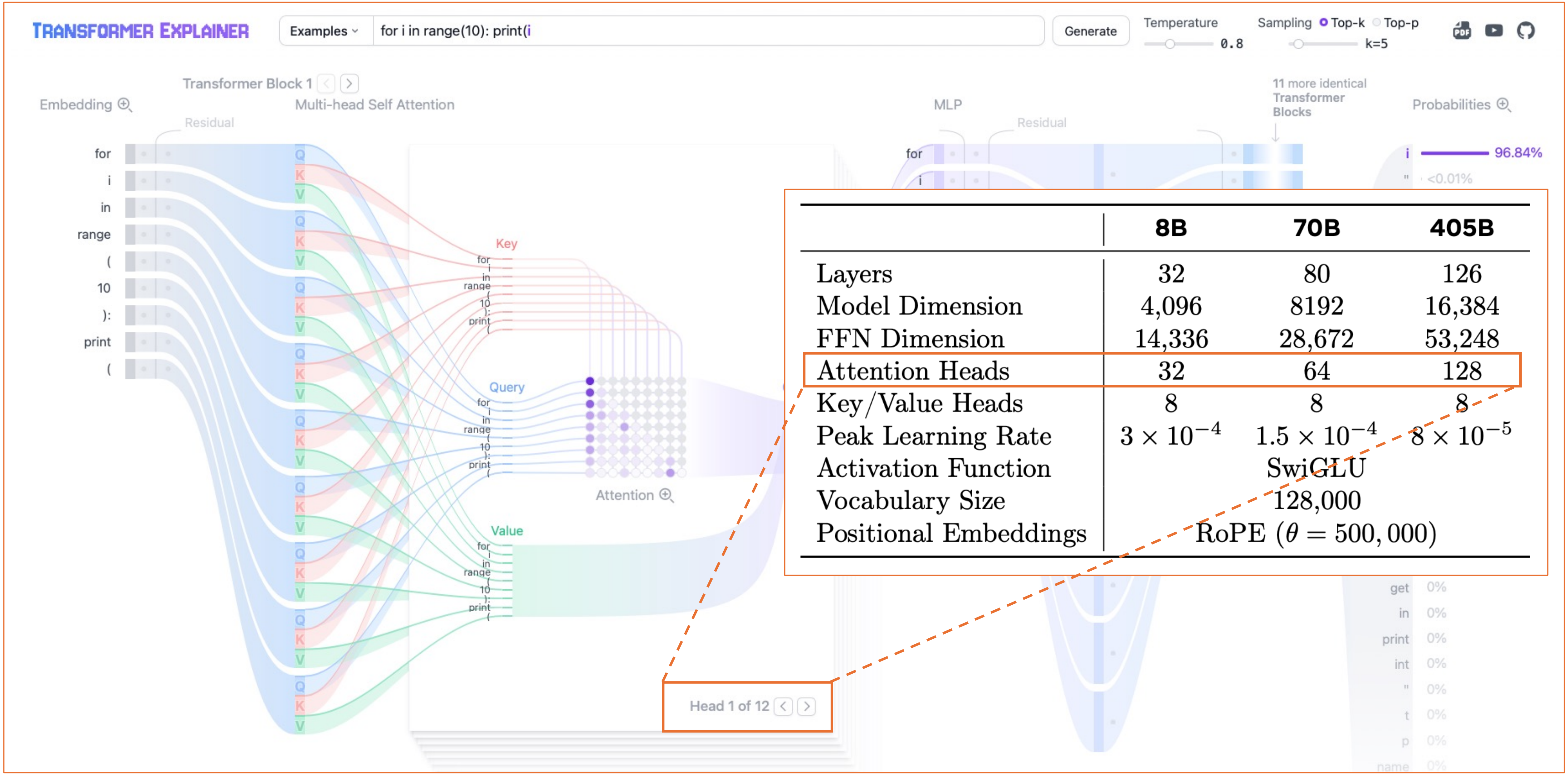
	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activat			
Vocabu			
Positional Embeddings	RoPE ($\theta = 500,000$)		

Grouped-Query Attention (GQA)

Temperature 0.8

Sampling ● Top-k ○ Top-p k=5

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



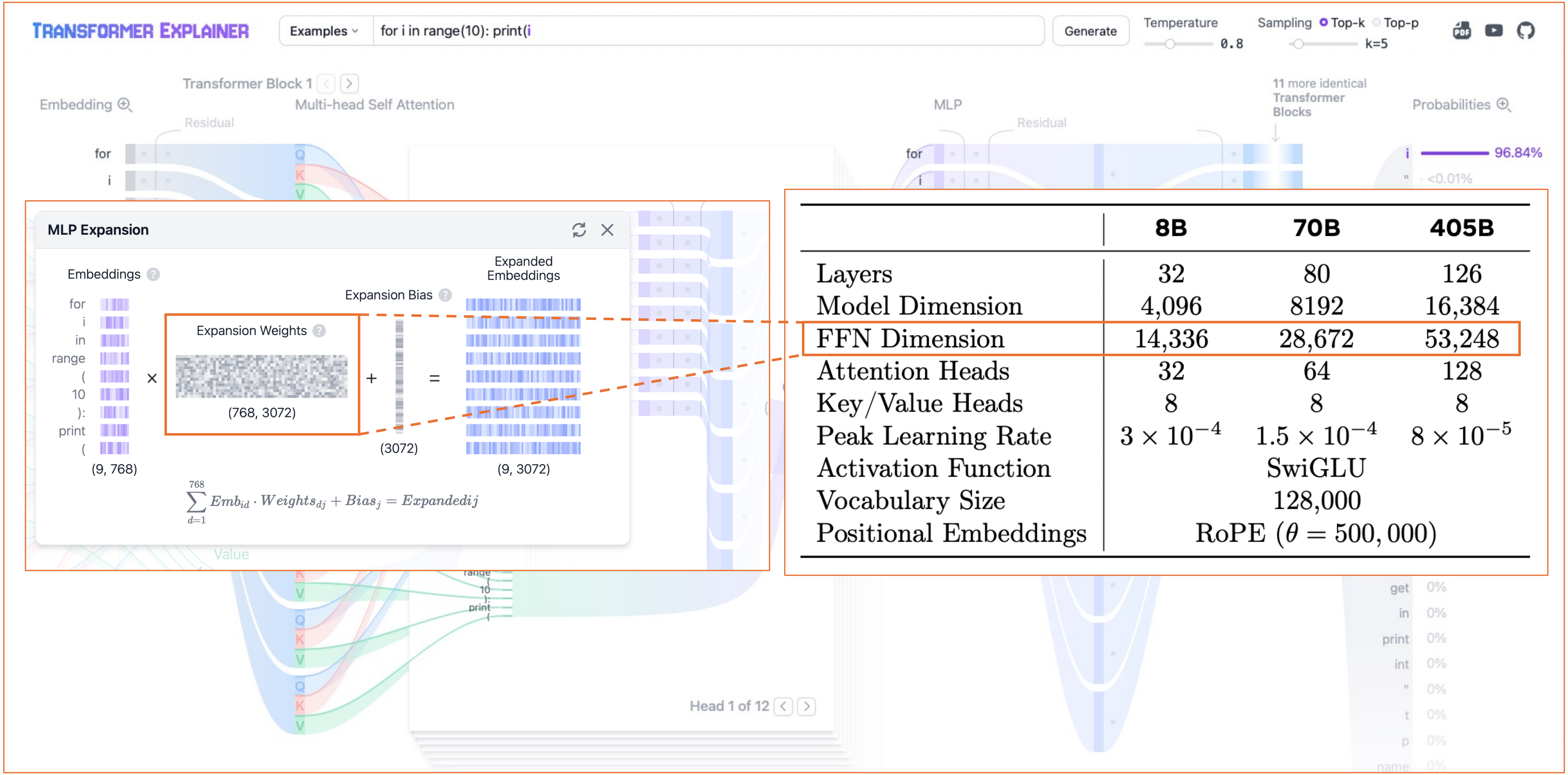
for i in range(10): print(i

Temperature 0.8

Sampling ● Top-k ○ Top-p k=5

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



TRANSFORMER EXPLAINER

Examples: for i in range(10): print(i

Generate

Temperature 0.8

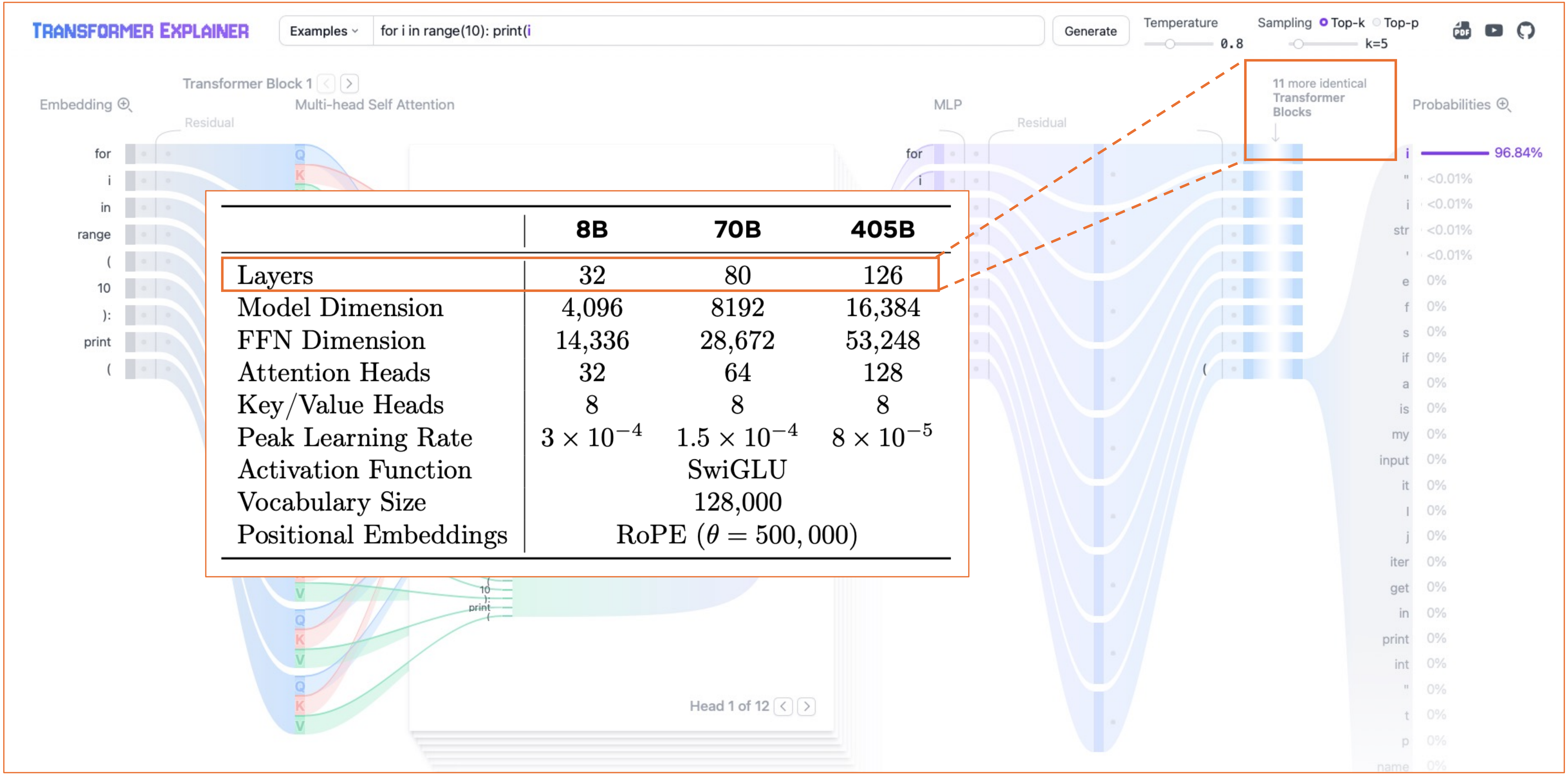
Sampling ● Top-k ○ Top-p k=5

MLP Expansion

$$\sum_{d=1}^{768} Emb_{id} \cdot Weights_{dj} + Bias_j = Expandedij$$

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function	SwiGLU		
Vocabulary Size	128,000		
Positional Embeddings	RoPE ($\theta = 500,000$)		

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



for i in range(10): print(i

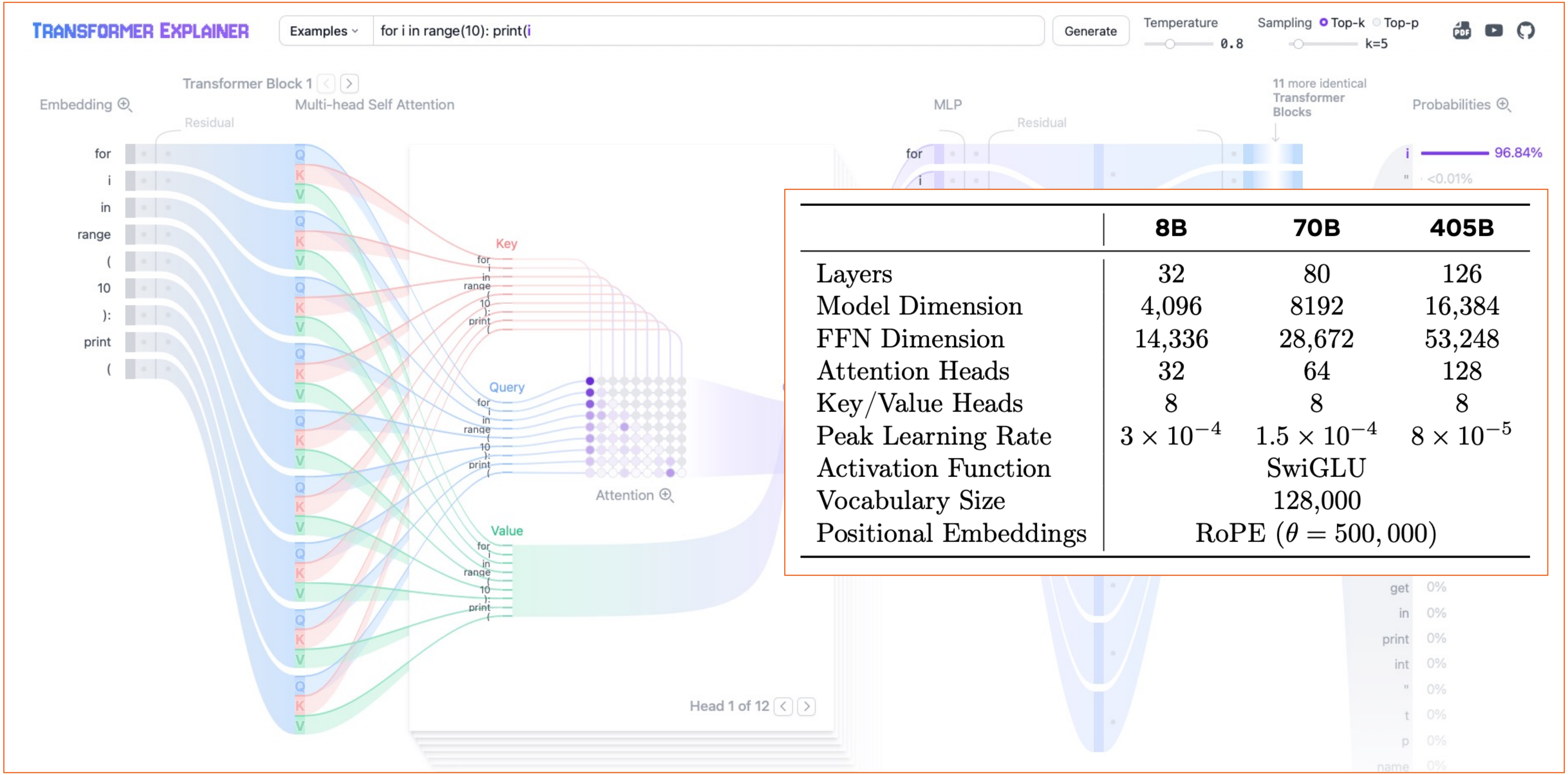
Temperature 0.8

Sampling ● Top-k ○ Top-p

k=5

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ($\theta = 500,000$)	

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



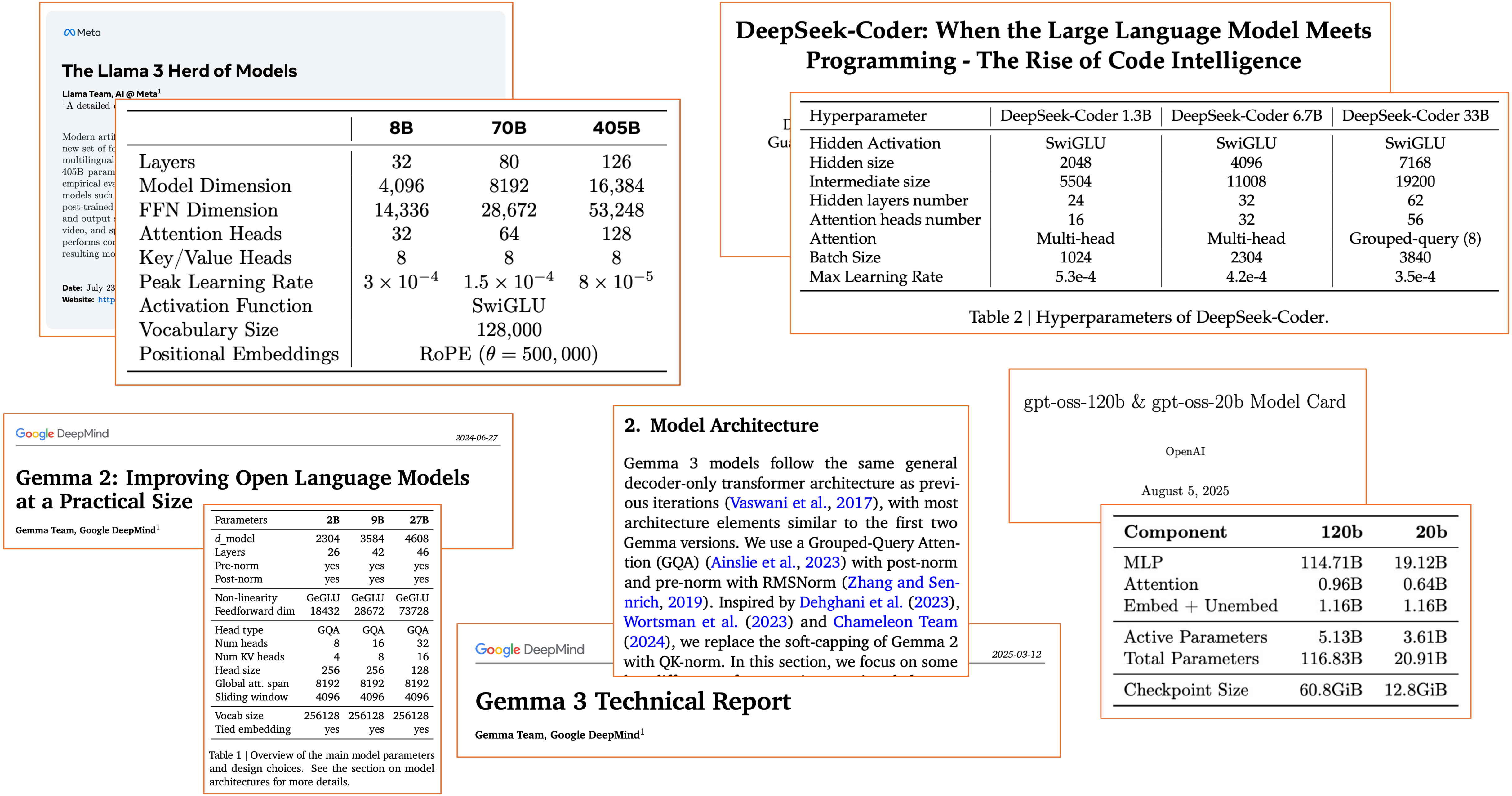
for i in range(10): print(i

Temperature 0.8

Sampling ● Top-k ○ Top-p k=5

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ($\theta = 500,000$)	

The Llama 3 Herd of Models, Llama Team, AI@Meta, 2024



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed

Modern arti
new set of fo
multilingual
405B param
empirical eva
models such
post-trained
and output
video, and sp
performs co
resulting mo

Date: July 23
Website: http

	8B	70B	405B
Layers	32	80	126
Model Dimension	4,096	8192	16,384
FFN Dimension	14,336	28,672	53,248
Attention Heads	32	64	128
Key/Value Heads	8	8	8
Peak Learning Rate	3×10^{-4}	1.5×10^{-4}	8×10^{-5}
Activation Function		SwiGLU	
Vocabulary Size		128,000	
Positional Embeddings		RoPE ($\theta = 500,000$)	

DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

Hyperparameter	DeepSeek-Coder 1.3B	DeepSeek-Coder 6.7B	DeepSeek-Coder 33B
Hidden Activation	SwiGLU	SwiGLU	SwiGLU
Hidden size	2048	4096	7168
Intermediate size	5504	11008	19200
Hidden layers number	24	32	62
Attention heads number	16	32	56
Attention	Multi-head	Multi-head	Grouped-query (8)
Batch Size	1024	2304	3840
Max Learning Rate	5.3e-4	4.2e-4	3.5e-4

Table 2 | Hyperparameters of DeepSeek-Coder.

Google DeepMind

2024-06-27

Gemma 2: Improving Open Language Models at a Practical Size

Gemma Team, Google DeepMind[1]

Parameters	2B	9B	27B
d_model	2304	3584	4608
Layers	26	42	46
Pre-norm	yes	yes	yes
Post-norm	yes	yes	yes
Non-linearity	GeGLU	GeGLU	GeGLU
Feedforward dim	18432	28672	73728
Head type	GQA	GQA	GQA
Num heads	8	16	32
Num KV heads	4	8	16
Head size	256	256	128
Global att. span	8192	8192	8192
Sliding window	4096	4096	4096
Vocab size	256128	256128	256128
Tied embedding	yes	yes	yes

Table 1 | Overview of the main model parameters and design choices. See the section on model architectures for more details.

2. Model Architecture

Gemma 3 models follow the same general decoder-only transformer architecture as previous iterations (Vaswani et al., 2017), with most architecture elements similar to the first two Gemma versions. We use a Grouped-Query Attention (GQA) (Ainslie et al., 2023) with post-norm and pre-norm with RMSNorm (Zhang and Sennrich, 2019). Inspired by Dehghani et al. (2023), Wortsman et al. (2023) and Chameleon Team (2024), we replace the soft-capping of Gemma 2 with QK-norm. In this section, we focus on some

gpt-oss-120b & gpt-oss-20b Model Card

OpenAI

August 5, 2025

Component	120b	20b
MLP	114.71B	19.12B
Attention	0.96B	0.64B
Embed + Unembed	1.16B	1.16B
Active Parameters	5.13B	3.61B
Total Parameters	116.83B	20.91B
Checkpoint Size	60.8GiB	12.8GiB

Google DeepMind

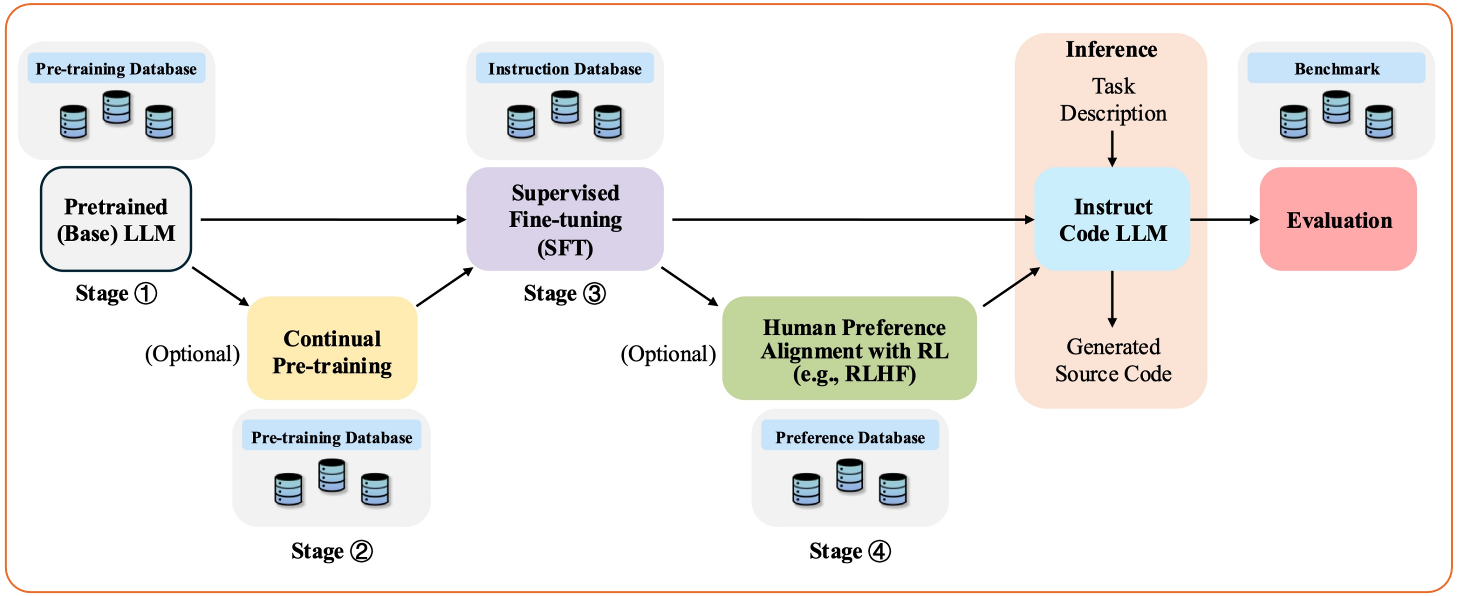
2025-03-12

Gemma 3 Technical Report

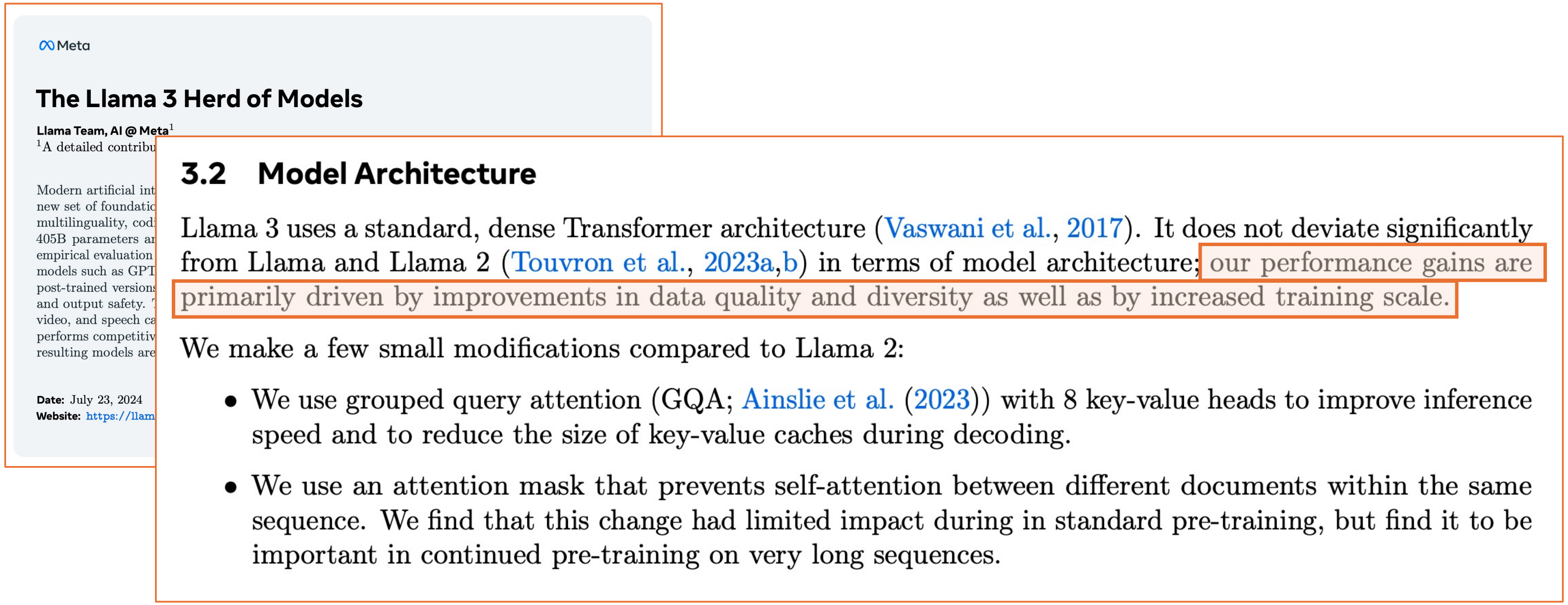
Gemma Team, Google DeepMind[1]

Today’s Agenda

- Pre-training stage
  - ~~Model architecture~~
  - Pre-training dataset
  - Learning objectives
  - Evaluation dataset
- Special topics
  - Post-training staging
  - Scaling law
  - Hallucination



Pre-training: Dataset



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed contribu

Modern artificial int
new set of foundatic
multilinguality, codi
405B parameters ar
empirical evaluation
models such as GPT
post-trained versions
and output safety. T
video, and speech ca
performs competitiv
resulting models are

Date: July 23, 2024
Website: https://llam

3.2 Model Architecture

Llama 3 uses a standard, dense Transformer architecture (Vaswani et al., 2017). It does not deviate significantly from Llama and Llama 2 (Touvron et al., 2023a,b) in terms of model architecture; our performance gains are primarily driven by improvements in data quality and diversity as well as by increased training scale.

We make a few small modifications compared to Llama 2:

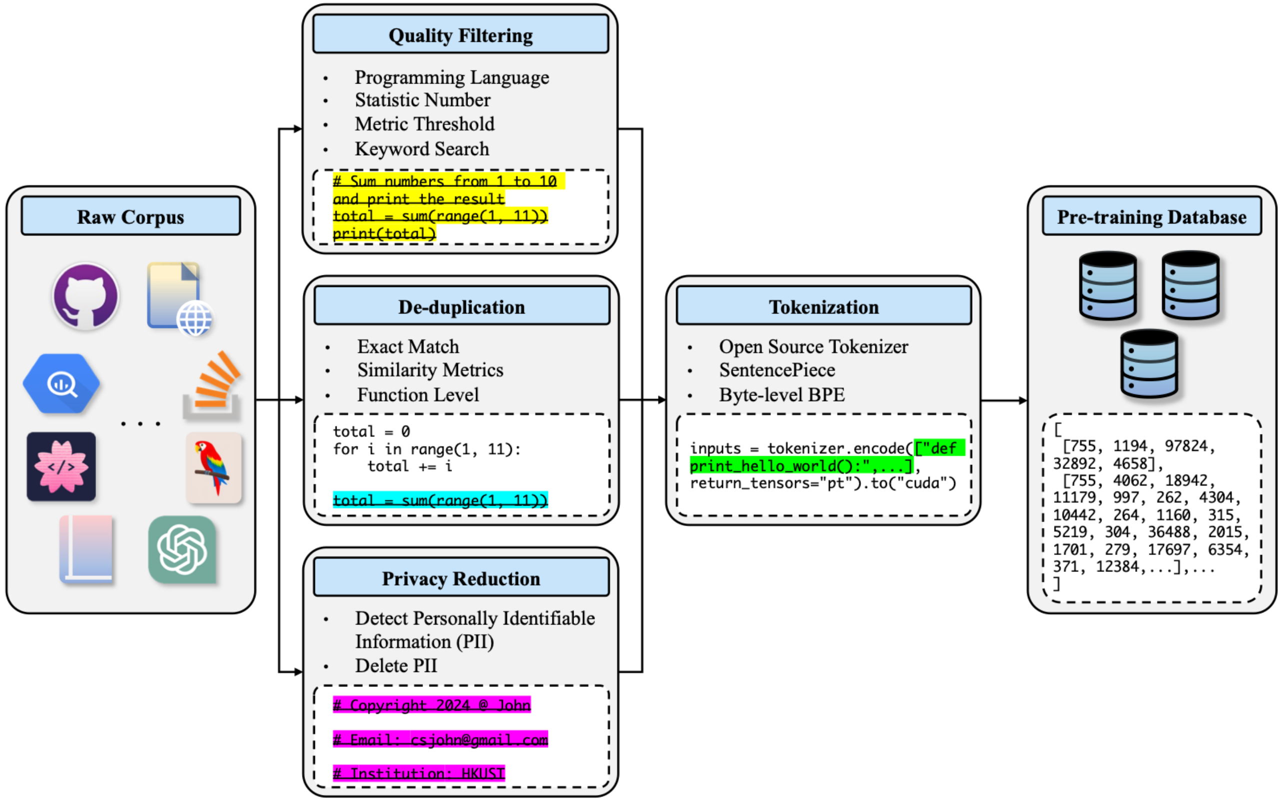
- We use grouped query attention (GQA; Ainslie et al. (2023)) with 8 key-value heads to improve inference speed and to reduce the size of key-value caches during decoding.
- We use an attention mask that prevents self-attention between different documents within the same sequence. We find that this change had limited impact during in standard pre-training, but find it to be important in continued pre-training on very long sequences.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Raw Corpus

Quality Filtering

- Programming Language
- Statistic Number
- Metric Threshold
- Keyword Search

~~# Sum numbers from 1 to 10~~
~~and print the result~~
~~total = sum(range(1, 11))~~
~~print(total)~~

De-duplication

- Exact Match
- Similarity Metrics
- Function Level

```
total = 0
for i in range(1, 11):
    total += i
```

~~total = sum(range(1, 11))~~

Privacy Reduction

- Detect Personally Identifiable Information (PII)
- Delete PII

~~# Copyright 2024 @ John~~

~~# Email: csjohn@gmail.com~~

~~# Institution: HKUST~~

Tokenization

- Open Source Tokenizer
- SentencePiece
- Byte-level BPE

```
inputs = tokenizer.encode(["def
print_hello_world():",...],
return_tensors="pt").to("cuda")
```

Pre-training Database

```
[
 [755, 1194, 97824,
32892, 4658],
 [755, 4062, 18942,
11179, 997, 262, 4304,
10442, 264, 1160, 315,
5219, 304, 36488, 2015,
1701, 279, 17697, 6354,
371, 12384,...],...
]
```

A Survey on Large Language Models for Code Generation, Jiang et al., 2024

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

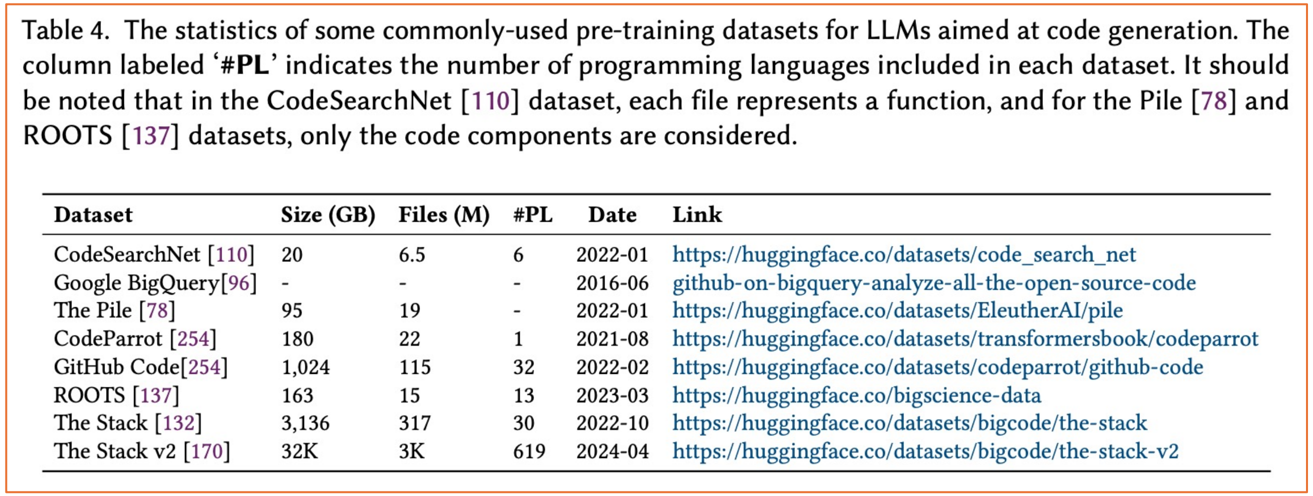


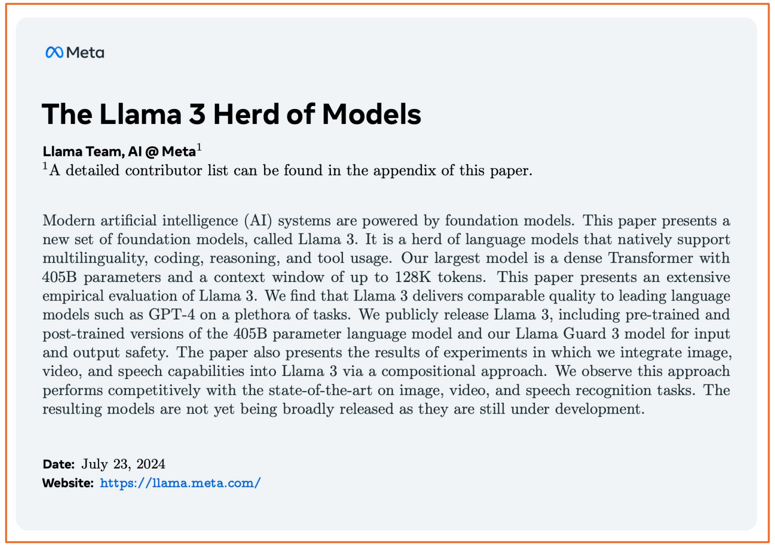
Table 4. The statistics of some commonly-used pre-training datasets for LLMs aimed at code generation. The column labeled '**#PL**' indicates the number of programming languages included in each dataset. It should be noted that in the CodeSearchNet [110] dataset, each file represents a function, and for the Pile [78] and ROOTS [137] datasets, only the code components are considered.

Dataset	Size (GB)	Files (M)	#PL	Date	Link
CodeSearchNet [110]	20	6.5	6	2022-01	https://huggingface.co/datasets/code_search_net
Google BigQuery[96]	-	-	-	2016-06	github-on-bigquery-analyze-all-the-open-source-code
The Pile [78]	95	19	-	2022-01	https://huggingface.co/datasets/EleutherAI/pile
CodeParrot [254]	180	22	1	2021-08	https://huggingface.co/datasets/transformersbook/codeparrot
GitHub Code[254]	1,024	115	32	2022-02	https://huggingface.co/datasets/codeparrot/github-code
ROOTS [137]	163	15	13	2023-03	https://huggingface.co/bigscience-data
The Stack [132]	3,136	317	30	2022-10	https://huggingface.co/datasets/bigcode/the-stack
The Stack v2 [170]	32K	3K	619	2024-04	https://huggingface.co/datasets/bigcode/the-stack-v2

A Survey on Large Language Models for Code Generation, Jiang et al., 2024

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

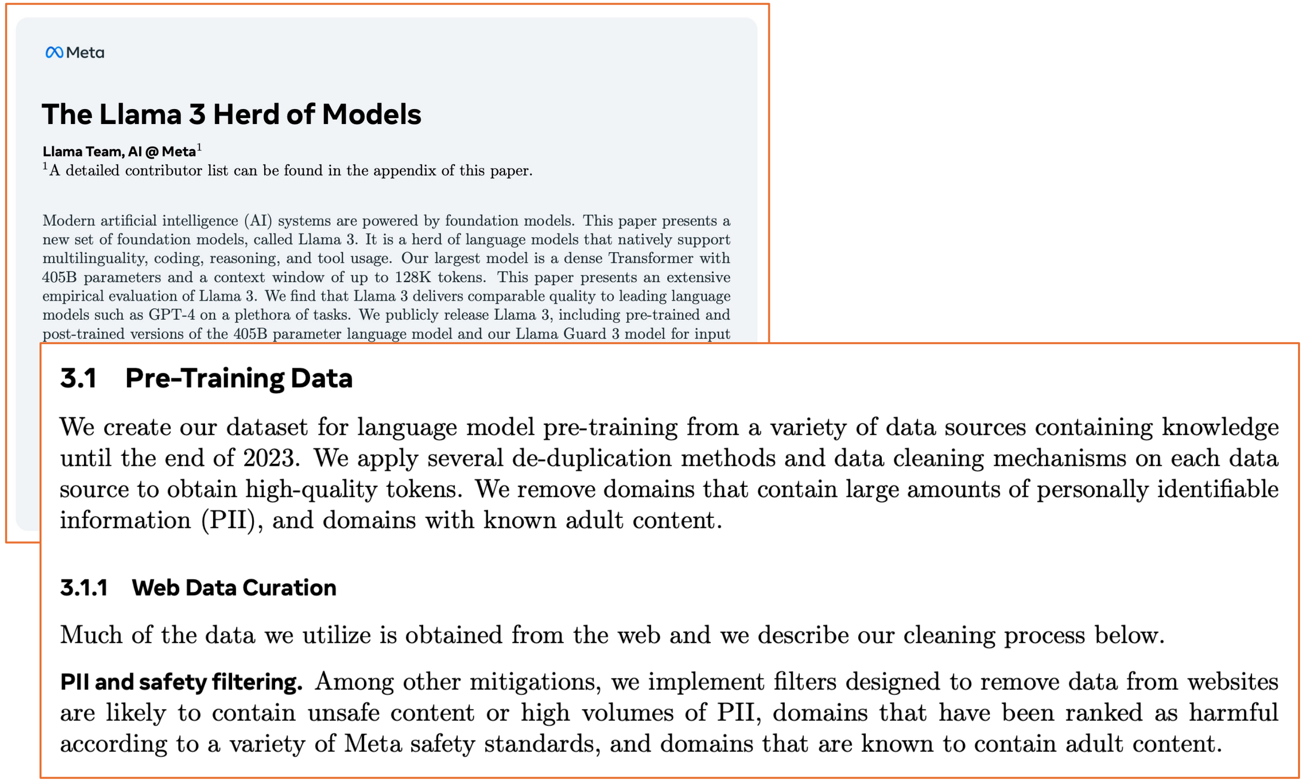
[1] A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1] A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input

3.1 Pre-Training Data

We create our dataset for language model pre-training from a variety of data sources containing knowledge until the end of 2023. We apply several de-duplication methods and data cleaning mechanisms on each data source to obtain high-quality tokens. We remove domains that contain large amounts of personally identifiable information (PII), and domains with known adult content.

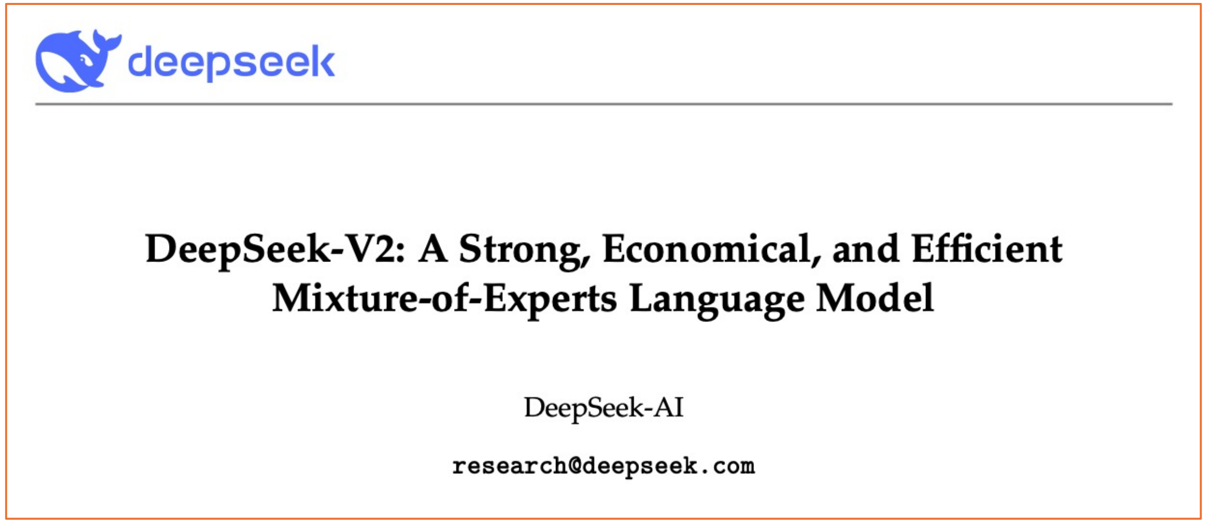
3.1.1 Web Data Curation

Much of the data we utilize is obtained from the web and we describe our cleaning process below.

PII and safety filtering. Among other mitigations, we implement filters designed to remove data from websites are likely to contain unsafe content or high volumes of PII, domains that have been ranked as harmful according to a variety of Meta safety standards, and domains that are known to contain adult content.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



deepseek

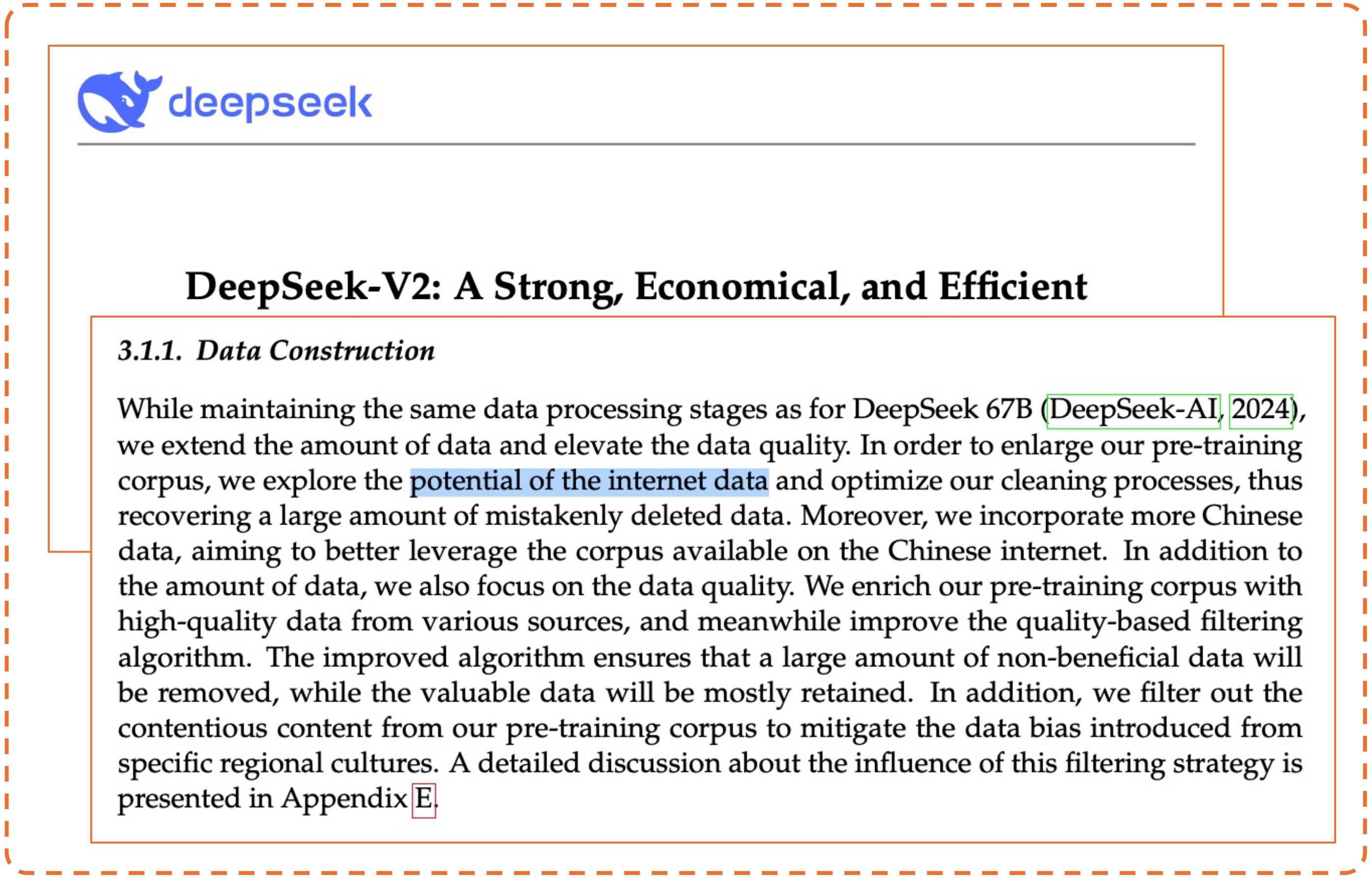
DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model

DeepSeek-AI

research@deepseek.com

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



deepseek

DeepSeek-V2: A Strong, Economical, and Efficient

3.1.1. Data Construction

While maintaining the same data processing stages as for DeepSeek 67B (DeepSeek-AI, 2024), we extend the amount of data and elevate the data quality. In order to enlarge our pre-training corpus, we explore the potential of the internet data and optimize our cleaning processes, thus recovering a large amount of mistakenly deleted data. Moreover, we incorporate more Chinese data, aiming to better leverage the corpus available on the Chinese internet. In addition to the amount of data, we also focus on the data quality. We enrich our pre-training corpus with high-quality data from various sources, and meanwhile improve the quality-based filtering algorithm. The improved algorithm ensures that a large amount of non-beneficial data will be removed, while the valuable data will be mostly retained. In addition, we filter out the contentious content from our pre-training corpus to mitigate the data bias introduced from specific regional cultures. A detailed discussion about the influence of this filtering strategy is presented in Appendix E.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

deepseek

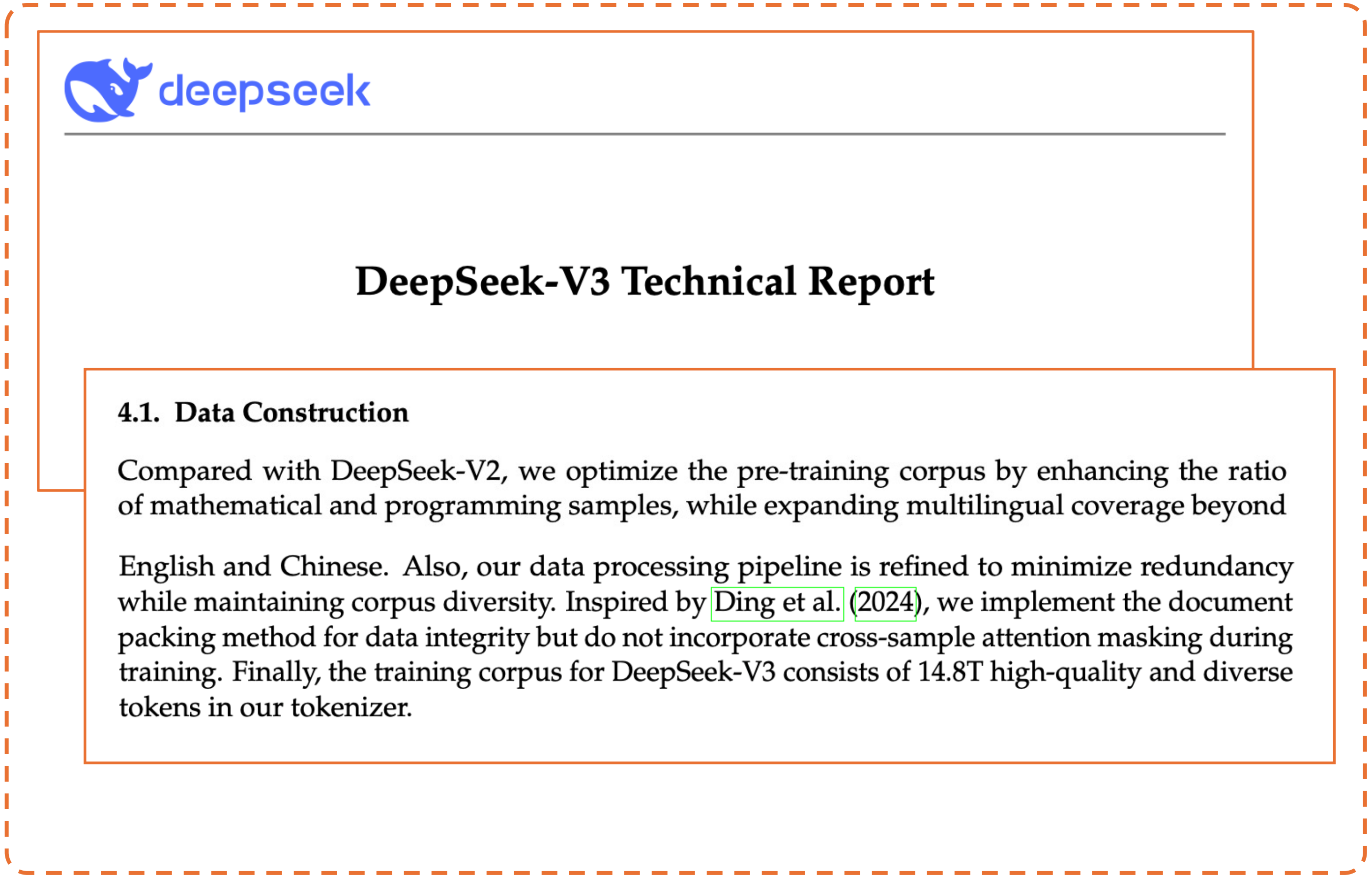
DeepSeek-V3 Technical Report

DeepSeek-AI

research@deepseek.com

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



deepseek

DeepSeek-V3 Technical Report

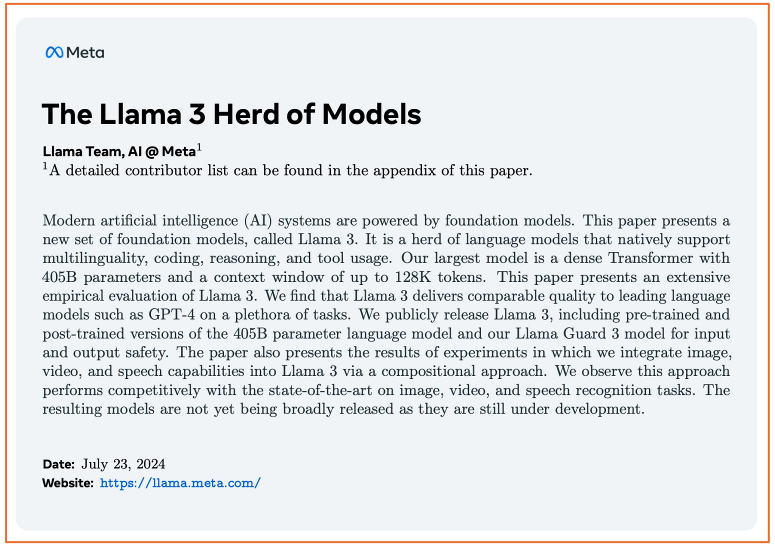
4.1. Data Construction

Compared with DeepSeek-V2, we optimize the pre-training corpus by enhancing the ratio of mathematical and programming samples, while expanding multilingual coverage beyond

English and Chinese. Also, our data processing pipeline is refined to minimize redundancy while maintaining corpus diversity. Inspired by Ding et al. (2024), we implement the document packing method for data integrity but do not incorporate cross-sample attention masking during training. Finally, the training corpus for DeepSeek-V3 consists of 14.8T high-quality and diverse tokens in our tokenizer.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

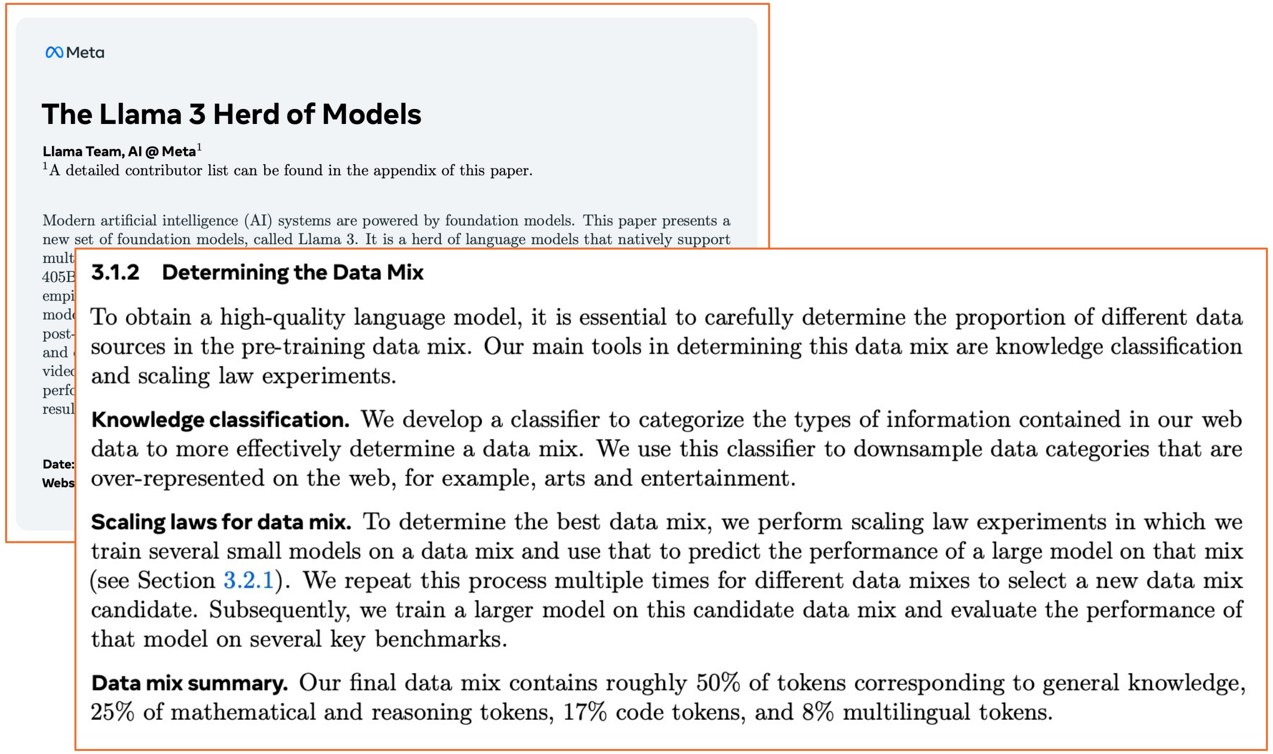
[1] A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Date: July 23, 2024
Website: https://llama.meta.com/

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support
mult
405B
empi
mod
post-
and
vide
perf
resul

Date:
Webs

3.1.2 Determining the Data Mix

To obtain a high-quality language model, it is essential to carefully determine the proportion of different data sources in the pre-training data mix. Our main tools in determining this data mix are knowledge classification and scaling law experiments.

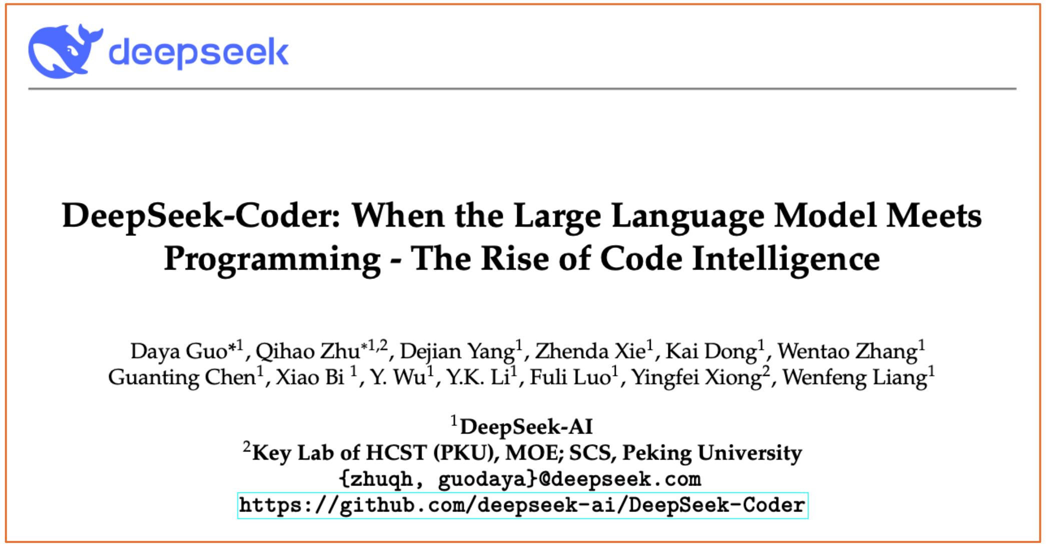
Knowledge classification. We develop a classifier to categorize the types of information contained in our web data to more effectively determine a data mix. We use this classifier to downsample data categories that are over-represented on the web, for example, arts and entertainment.

Scaling laws for data mix. To determine the best data mix, we perform scaling law experiments in which we train several small models on a data mix and use that to predict the performance of a large model on that mix (see Section 3.2.1). We repeat this process multiple times for different data mixes to select a new data mix candidate. Subsequently, we train a larger model on this candidate data mix and evaluate the performance of that model on several key benchmarks.

Data mix summary. Our final data mix contains roughly 50% of tokens corresponding to general knowledge, 25% of mathematical and reasoning tokens, 17% code tokens, and 8% multilingual tokens.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



deepseek

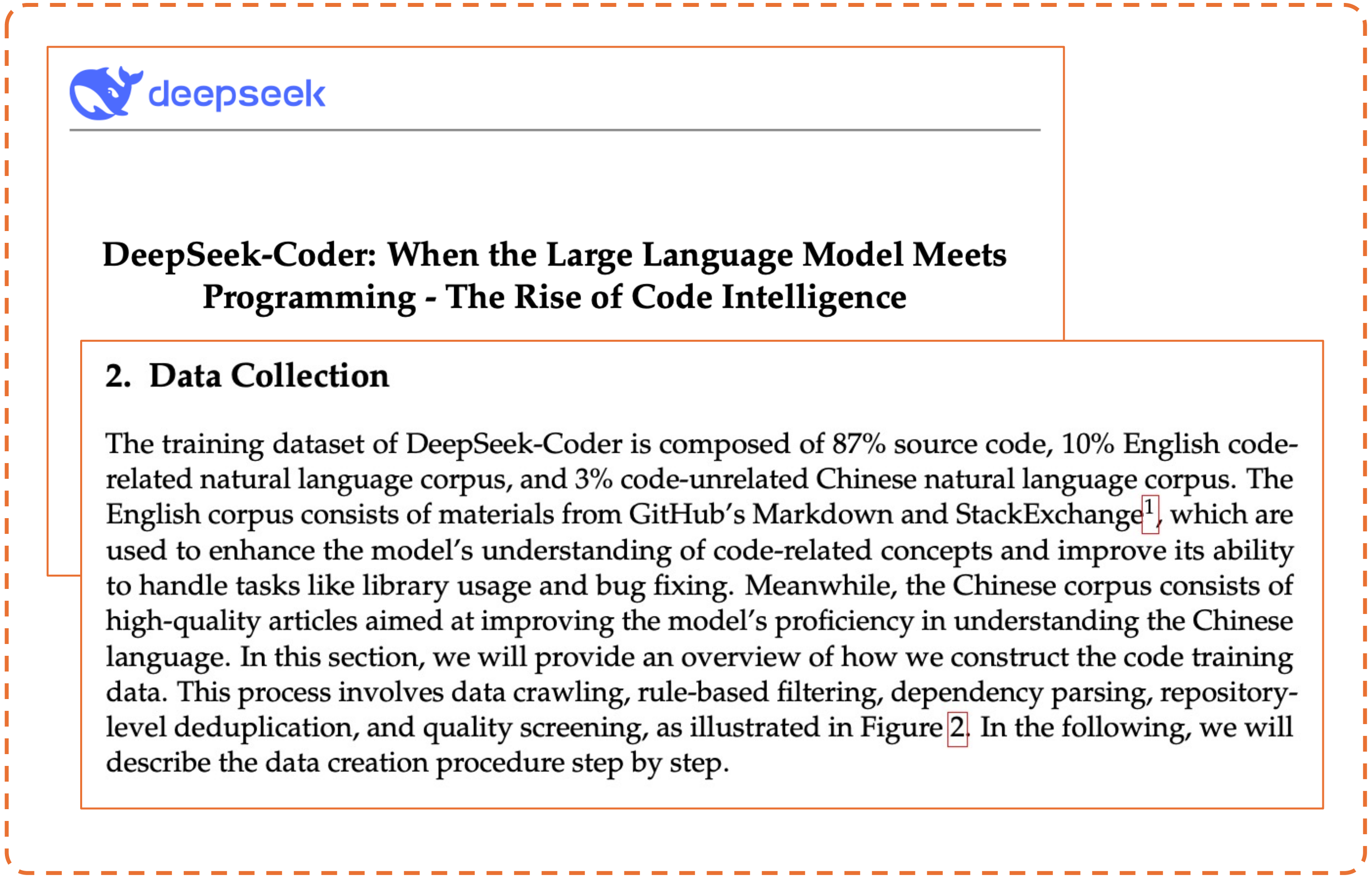
DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

Daya Guo[*1], Qihao Zhu[*1,2], Dejian Yang[1], Zhenda Xie[1], Kai Dong[1], Wentao Zhang[1]
Guanting Chen[1], Xiao Bi [1], Y. Wu[1], Y.K. Li[1], Fuli Luo[1], Yingfei Xiong[2], Wenfeng Liang[1]

[1]DeepSeek-AI
[2]Key Lab of HCST (PKU), MOE; SCS, Peking University
{zhuqh, guodaya}@deepseek.com
https://github.com/deepseek-ai/DeepSeek-Coder

Pre-training: Dataset

- Dimensions:
    - Data curation
    - Size and data mix
- Key considerations:
    - Specializing for coding
    - Data pollution
    - Scaling law



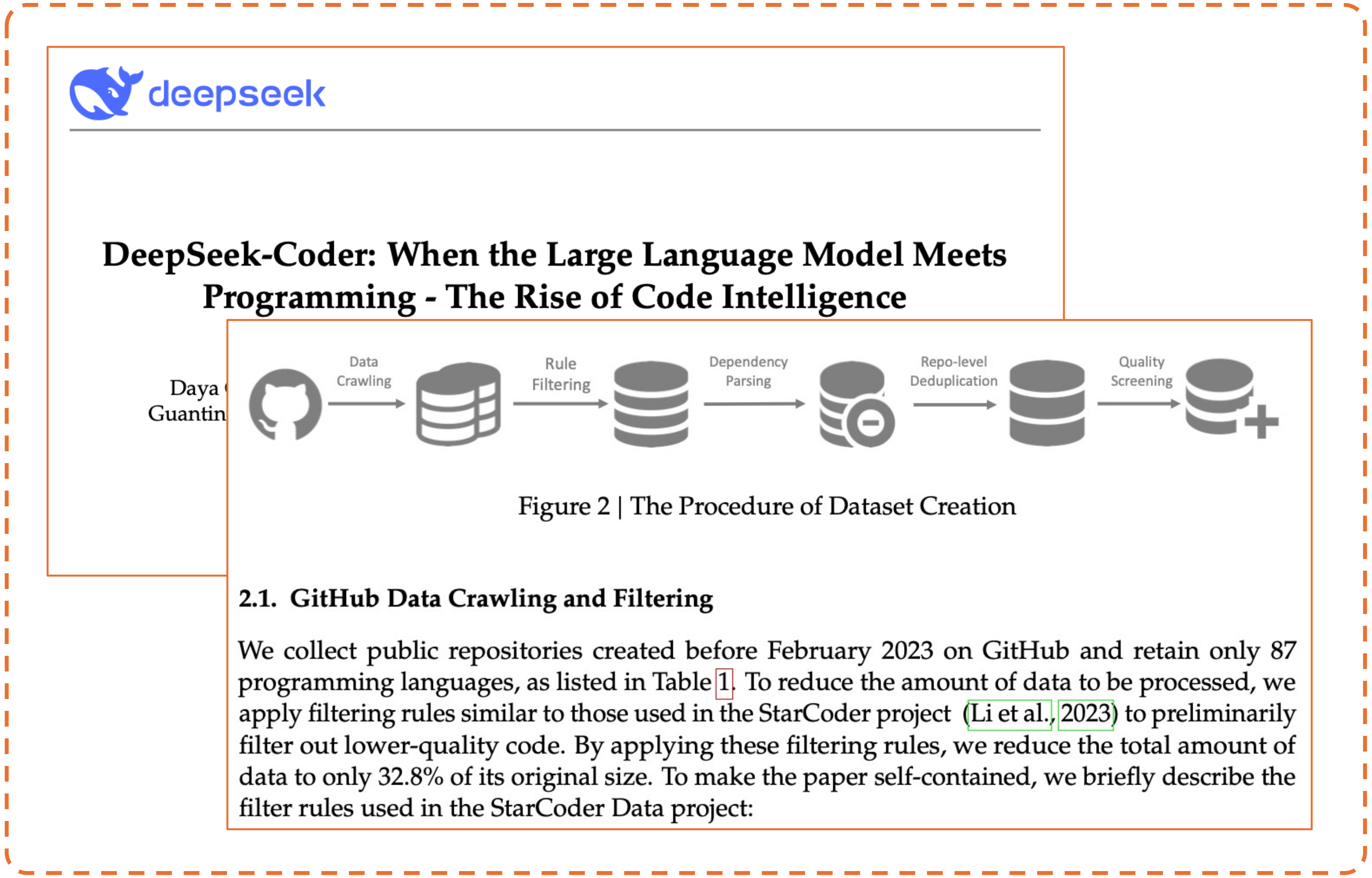
DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

2. Data Collection

The training dataset of DeepSeek-Coder is composed of 87% source code, 10% English code-related natural language corpus, and 3% code-unrelated Chinese natural language corpus. The English corpus consists of materials from GitHub's Markdown and StackExchange[1], which are used to enhance the model's understanding of code-related concepts and improve its ability to handle tasks like library usage and bug fixing. Meanwhile, the Chinese corpus consists of high-quality articles aimed at improving the model's proficiency in understanding the Chinese language. In this section, we will provide an overview of how we construct the code training data. This process involves data crawling, rule-based filtering, dependency parsing, repository-level deduplication, and quality screening, as illustrated in Figure 2. In the following, we will describe the data creation procedure step by step.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

Daya
Guantin

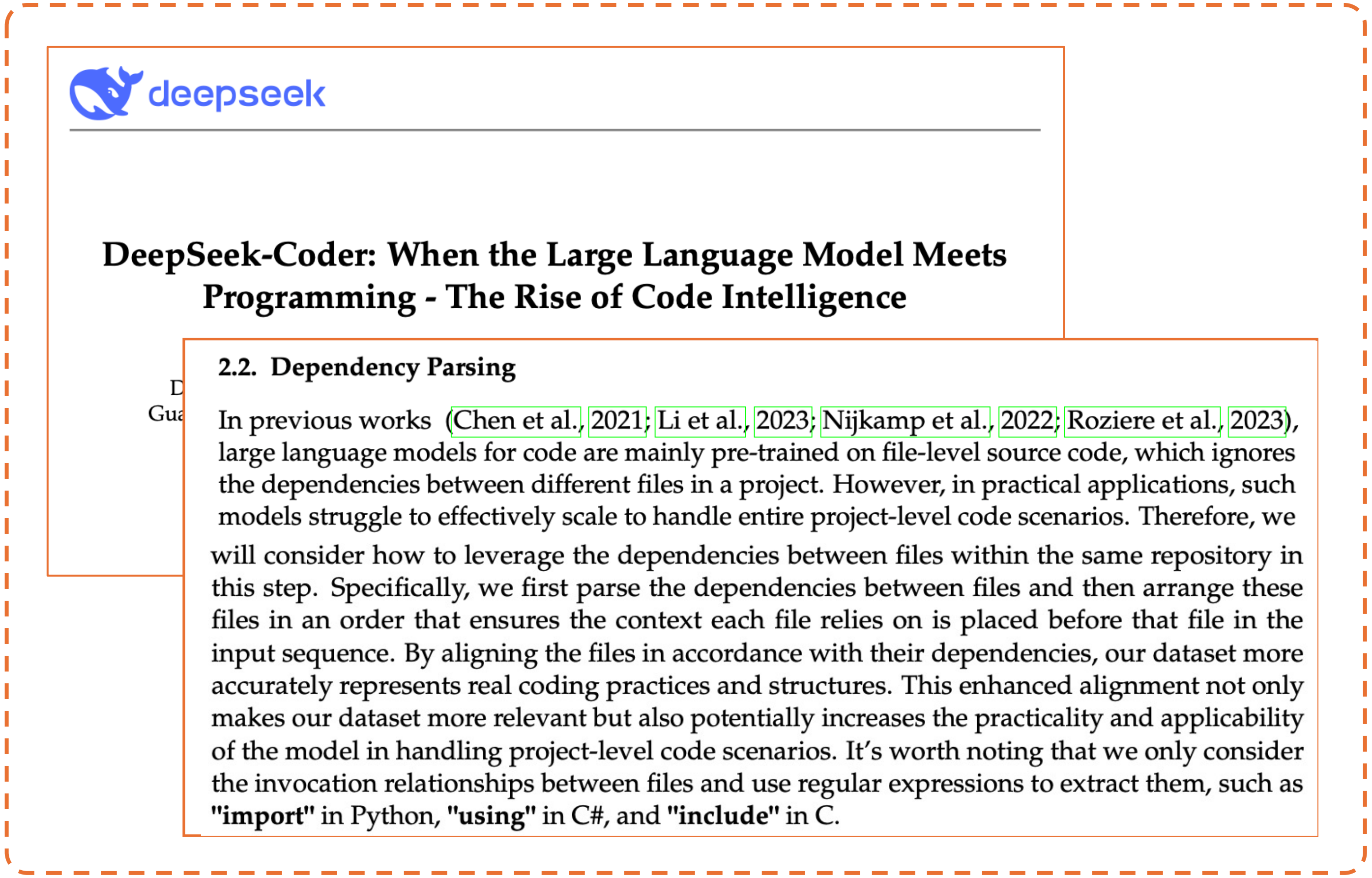
Figure 2 | The Procedure of Dataset Creation

2.1. GitHub Data Crawling and Filtering

We collect public repositories created before February 2023 on GitHub and retain only 87 programming languages, as listed in Table 1. To reduce the amount of data to be processed, we apply filtering rules similar to those used in the StarCoder project (Li et al., 2023) to preliminarily filter out lower-quality code. By applying these filtering rules, we reduce the total amount of data to only 32.8% of its original size. To make the paper self-contained, we briefly describe the filter rules used in the StarCoder Data project:

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



deepseek

DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

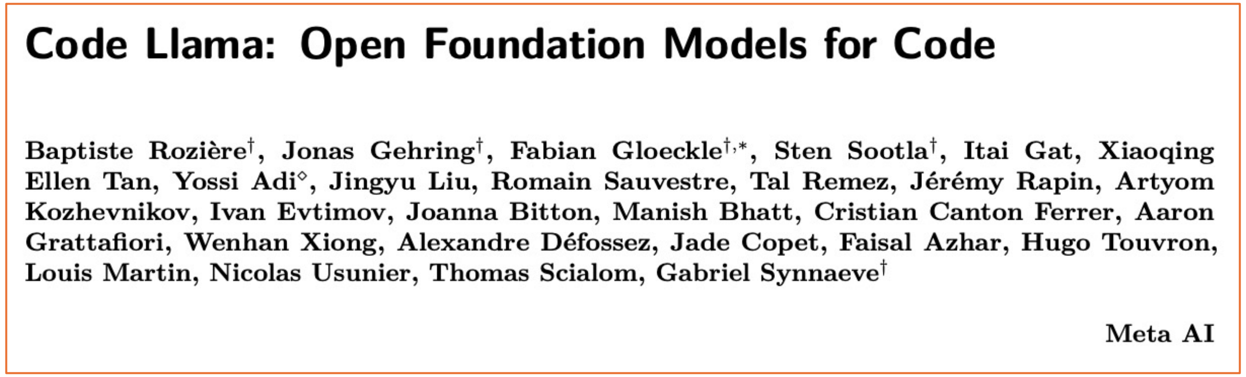
D
Gua

2.2. Dependency Parsing

In previous works (Chen et al., 2021; Li et al., 2023; Nijkamp et al., 2022; Roziere et al., 2023), large language models for code are mainly pre-trained on file-level source code, which ignores the dependencies between different files in a project. However, in practical applications, such models struggle to effectively scale to handle entire project-level code scenarios. Therefore, we will consider how to leverage the dependencies between files within the same repository in this step. Specifically, we first parse the dependencies between files and then arrange these files in an order that ensures the context each file relies on is placed before that file in the input sequence. By aligning the files in accordance with their dependencies, our dataset more accurately represents real coding practices and structures. This enhanced alignment not only makes our dataset more relevant but also potentially increases the practicality and applicability of the model in handling project-level code scenarios. It's worth noting that we only consider the invocation relationships between files and use regular expressions to extract them, such as **"import"** in Python, **"using"** in C#, and **"include"** in C.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



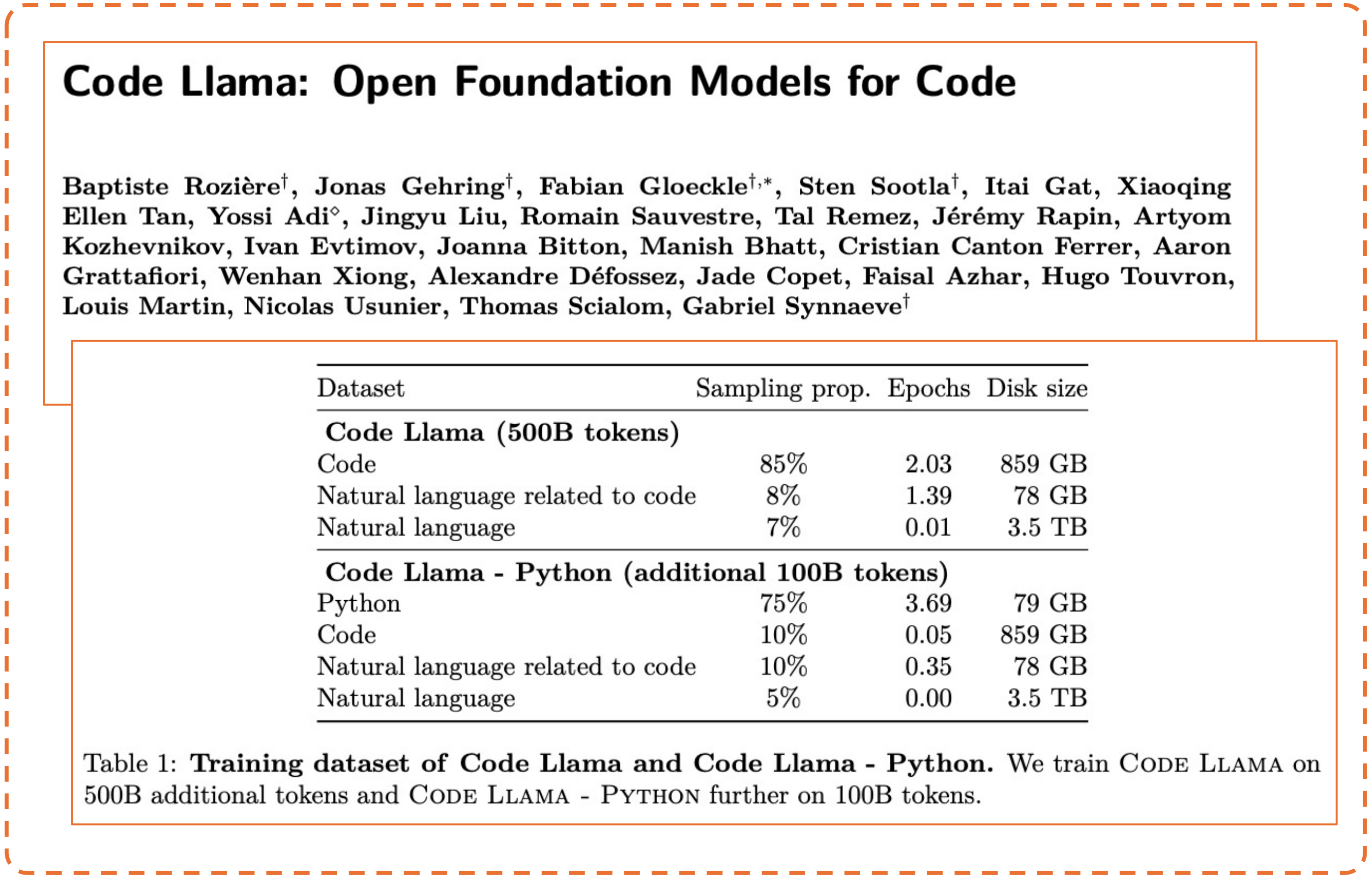
Code Llama: Open Foundation Models for Code

Baptiste Rozière[†], Jonas Gehring[†], Fabian Gloeckle[†,*], Sten Sootla[†], Itai Gat, Xiaoqing Ellen Tan, Yossi Adi[◇], Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, Gabriel Synnaeve[†]

Meta AI

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Code Llama: Open Foundation Models for Code

Baptiste Rozière[†], Jonas Gehring[†], Fabian Gloeckle[†,*], Sten Sootla[†], Itai Gat, Xiaoqing Ellen Tan, Yossi Adi[◇], Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, Gabriel Synnaeve[†]

Dataset	Sampling prop.	Epochs	Disk size
Code Llama (500B tokens)			
Code	85%	2.03	859 GB
Natural language related to code	8%	1.39	78 GB
Natural language	7%	0.01	3.5 TB
Code Llama - Python (additional 100B tokens)			
Python	75%	3.69	79 GB
Code	10%	0.05	859 GB
Natural language related to code	10%	0.35	78 GB
Natural language	5%	0.00	3.5 TB

Table 1: **Training dataset of Code Llama and Code Llama - Python.** We train Code Llama on 500B additional tokens and Code Llama - Python further on 100B tokens.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

Pre-training: Dataset

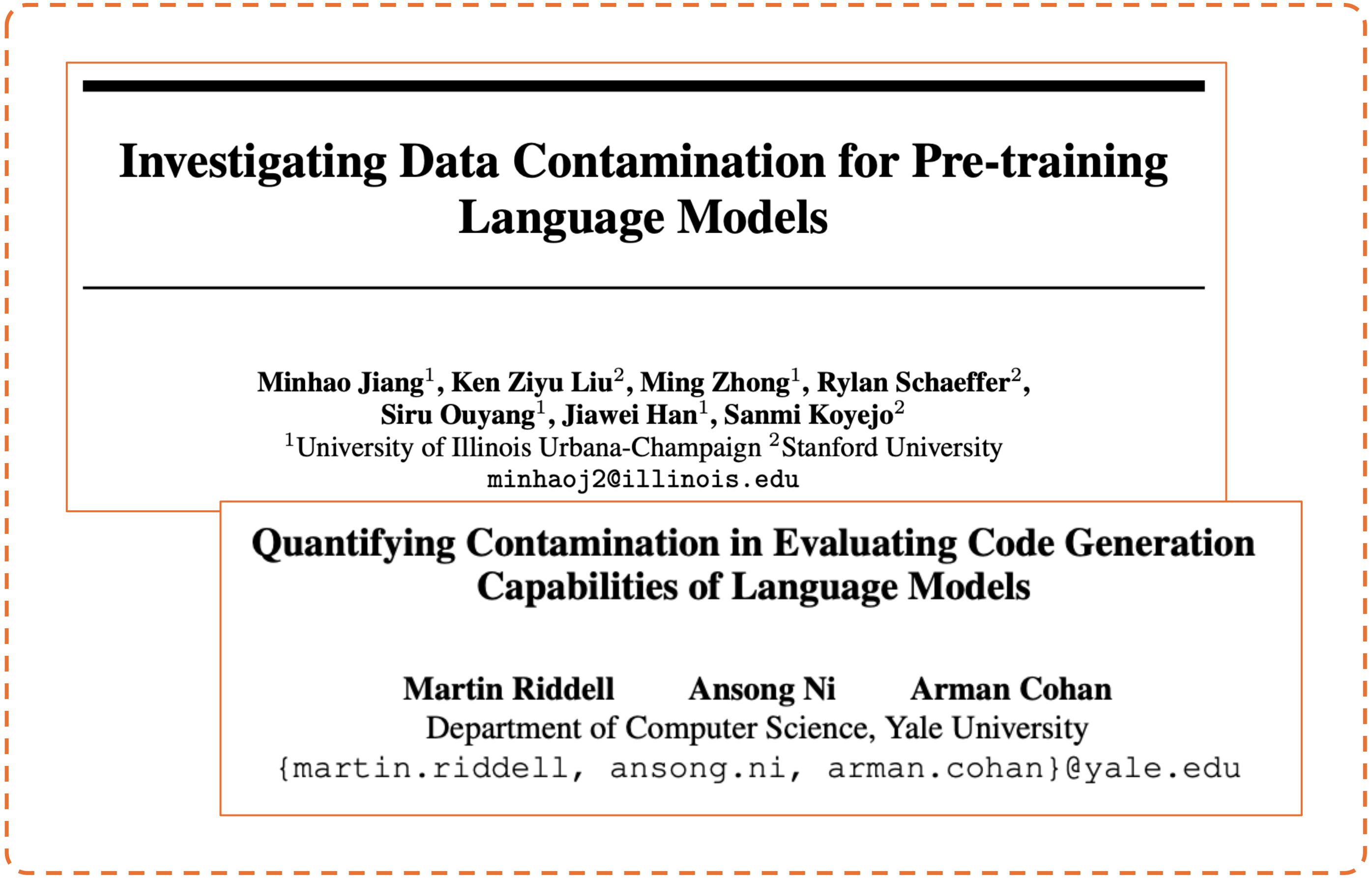
- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

Datasets are classified as:
- Training dataset
- Validation dataset
- Testing dataset (evaluation benchmarks)

Training dataset should not be polluted by validation and testing data samples.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Investigating Data Contamination for Pre-training Language Models

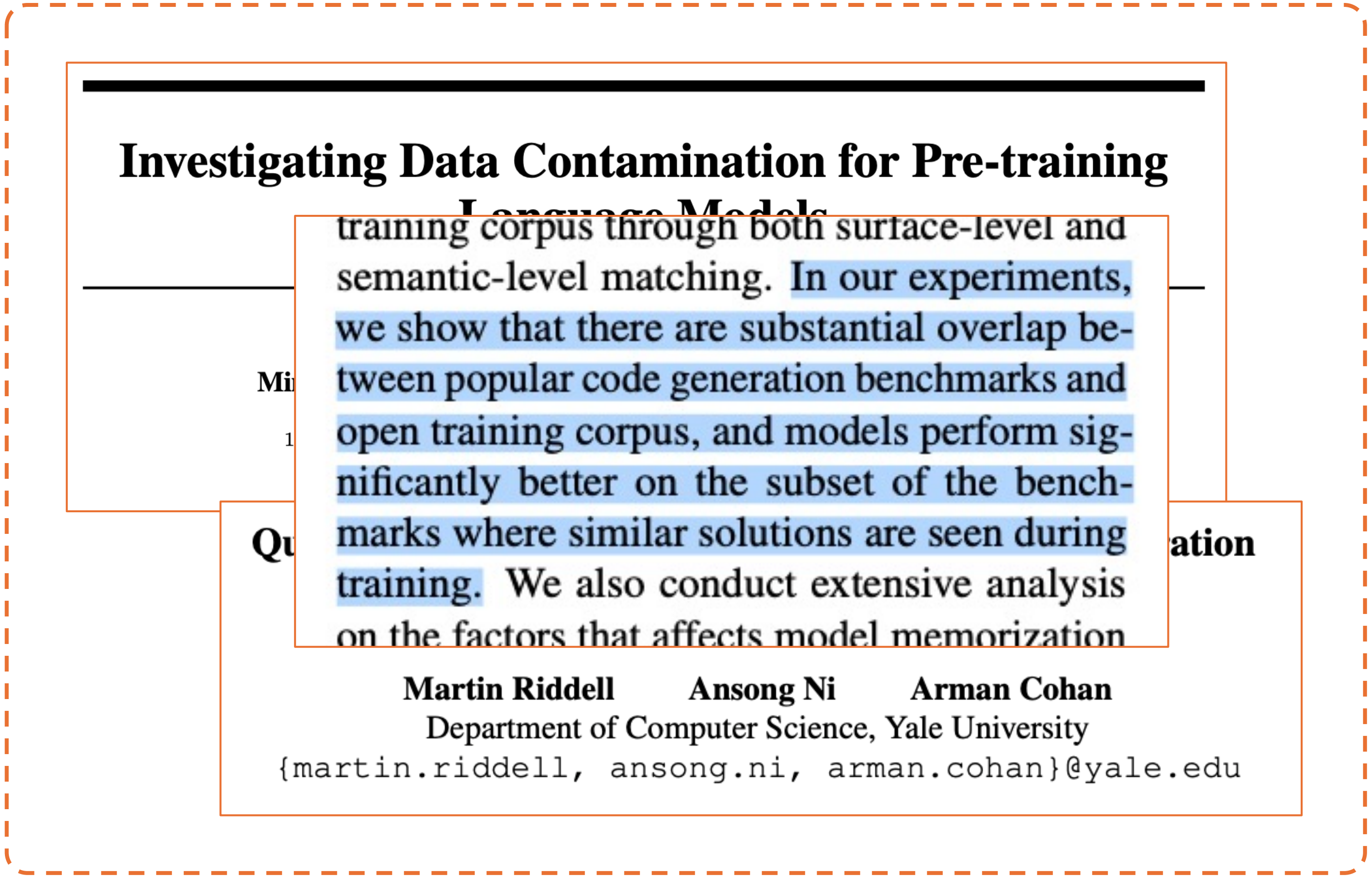
Minhao Jiang[1], Ken Ziyu Liu[2], Ming Zhong[1], Rylan Schaeffer[2], Siru Ouyang[1], Jiawei Han[1], Sanmi Koyejo[2]
[1]University of Illinois Urbana-Champaign [2]Stanford University
minhaoj2@illinois.edu

Quantifying Contamination in Evaluating Code Generation Capabilities of Language Models

Martin Riddell Ansong Ni Arman Cohan
Department of Computer Science, Yale University
{martin.riddell, ansong.ni, arman.cohan}@yale.edu

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Investigating Data Contamination for Pre-training Language Models

Mi

1

training corpus through both surface-level and semantic-level matching. In our experiments, we show that there are substantial overlap between popular code generation benchmarks and open training corpus, and models perform significantly better on the subset of the benchmarks where similar solutions are seen during training. We also conduct extensive analysis on the factors that affects model memorization

Qu ation

Martin Riddell Ansong Ni Arman Cohan
Department of Computer Science, Yale University
{martin.riddell, ansong.ni, arman.cohan}@yale.edu

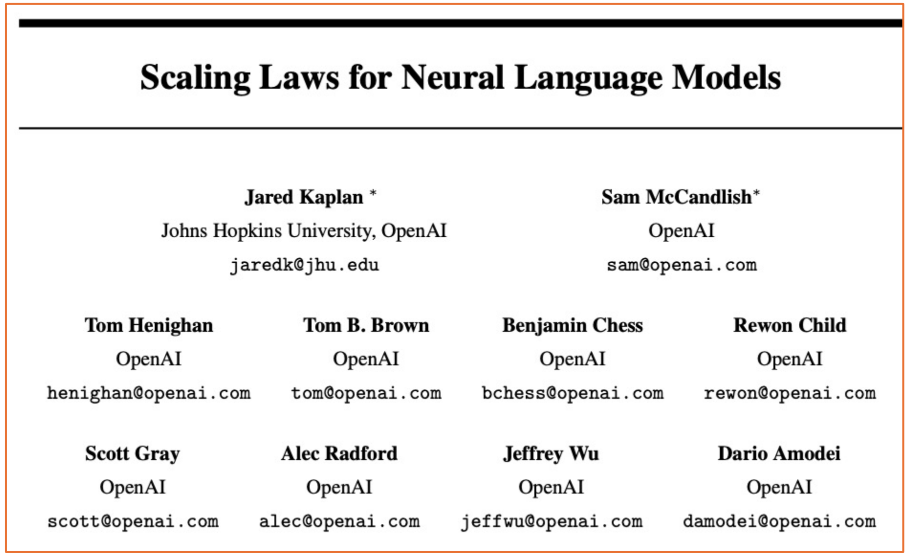
Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

The **scaling law** concept is about finding predictable, quantitative relationships between model performance and scale factors, without having to train every possible configuration.



Scaling Laws for Neural Language Models

Jared Kaplan *
Johns Hopkins University, OpenAI
jaredk@jhu.edu

Sam McCandlish*
OpenAI
sam@openai.com

Tom Henighan
OpenAI
henighan@openai.com

Tom B. Brown
OpenAI
tom@openai.com

Benjamin Chess
OpenAI
bchess@openai.com

Rewon Child
OpenAI
rewon@openai.com

Scott Gray
OpenAI
scott@openai.com

Alec Radford
OpenAI
alec@openai.com

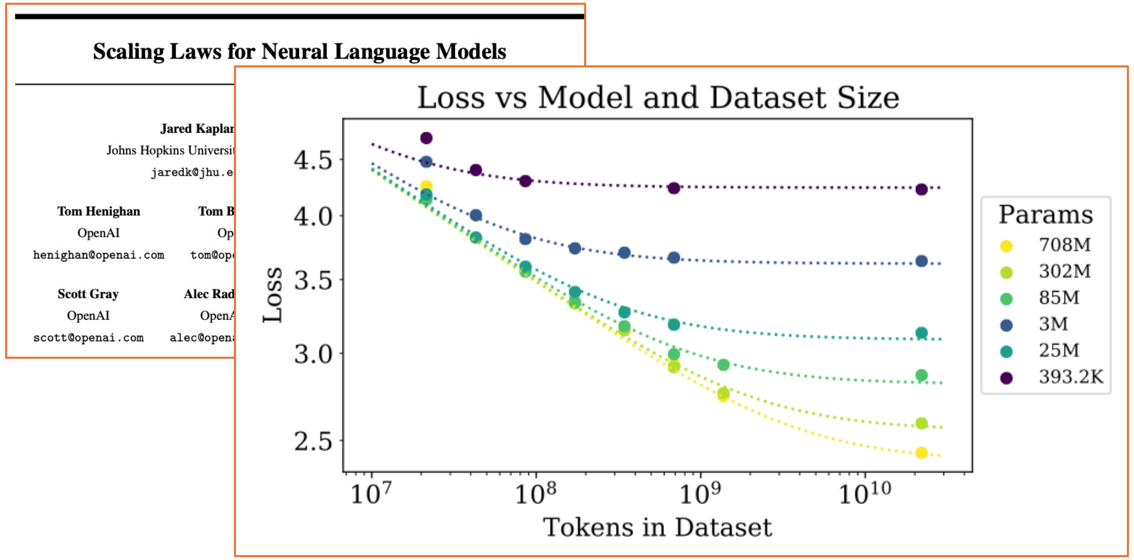
Jeffrey Wu
OpenAI
jeffwu@openai.com

Dario Amodei
OpenAI
damodei@openai.com

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law

The **scaling law** concept is about finding predictable, quantitative relationships between model performance and scale factors, without having to train every possible configuration.



Scaling Laws for Neural Language Models

Jared Kaplan
Johns Hopkins Universit
jaredk@jhu.e

Tom Henighan
OpenAI
henighan@openai.com

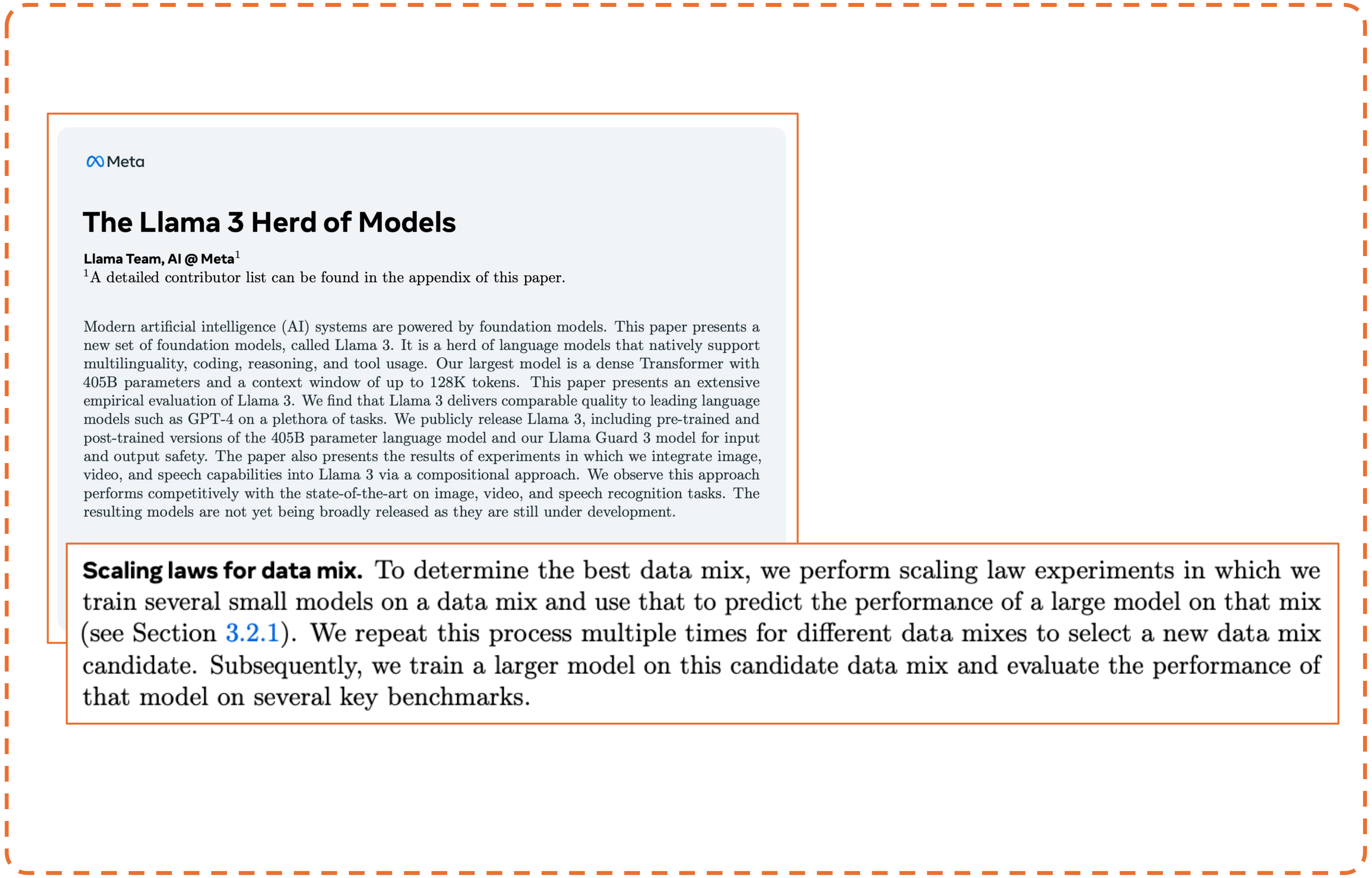
Tom B
Op
tom@ope

Scott Gray
OpenAI
scott@openai.com

Alec Rad
Open
alec@open

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

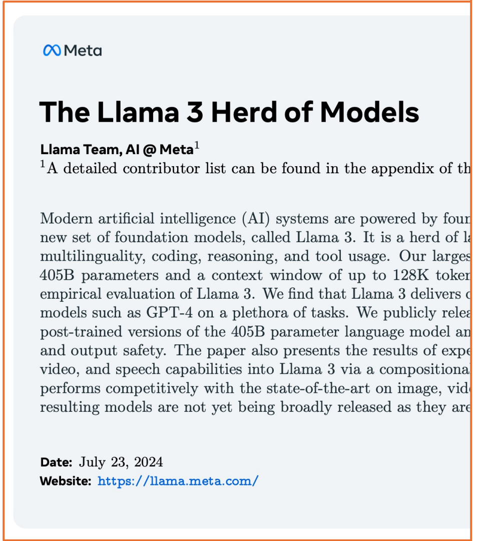
[1] A detailed contributor list can be found in the appendix of this paper.

Modern artificial intelligence (AI) systems are powered by foundation models. This paper presents a new set of foundation models, called Llama 3. It is a herd of language models that natively support multilinguality, coding, reasoning, and tool usage. Our largest model is a dense Transformer with 405B parameters and a context window of up to 128K tokens. This paper presents an extensive empirical evaluation of Llama 3. We find that Llama 3 delivers comparable quality to leading language models such as GPT-4 on a plethora of tasks. We publicly release Llama 3, including pre-trained and post-trained versions of the 405B parameter language model and our Llama Guard 3 model for input and output safety. The paper also presents the results of experiments in which we integrate image, video, and speech capabilities into Llama 3 via a compositional approach. We observe this approach performs competitively with the state-of-the-art on image, video, and speech recognition tasks. The resulting models are not yet being broadly released as they are still under development.

Scaling laws for data mix. To determine the best data mix, we perform scaling law experiments in which we train several small models on a data mix and use that to predict the performance of a large model on that mix (see Section 3.2.1). We repeat this process multiple times for different data mixes to select a new data mix candidate. Subsequently, we train a larger model on this candidate data mix and evaluate the performance of that model on several key benchmarks.

Pre-training: Dataset

- Dimensions:
  - Data curation
  - Size and data mix
- Key considerations:
  - Specializing for coding
  - Data pollution
  - Scaling law



Meta

The Llama 3 Herd of Models

Llama Team, AI @ Meta[1]

[1]A detailed contributor list can be found in the appendix of th

Modern artificial intelligence (AI) systems are powered by four
new set of foundation models, called Llama 3. It is a herd of la
multilinguality, coding, reasoning, and tool usage. Our larges
405B parameters and a context window of up to 128K toker
empirical evaluation of Llama 3. We find that Llama 3 delivers c
models such as GPT-4 on a plethora of tasks. We publicly relea
post-trained versions of the 405B parameter language model an
and output safety. The paper also presents the results of expe
video, and speech capabilities into Llama 3 via a compositiona
performs competitively with the state-of-the-art on image, vid
resulting models are not yet being broadly released as they are

Date: July 23, 2024
Website: https://llama.meta.com/

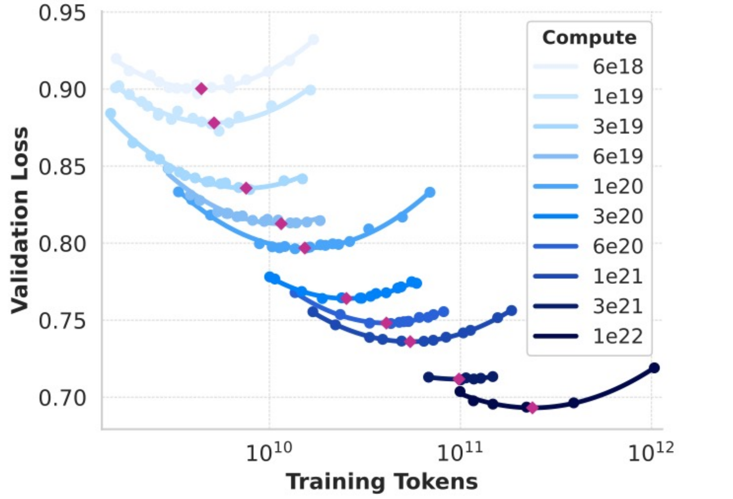
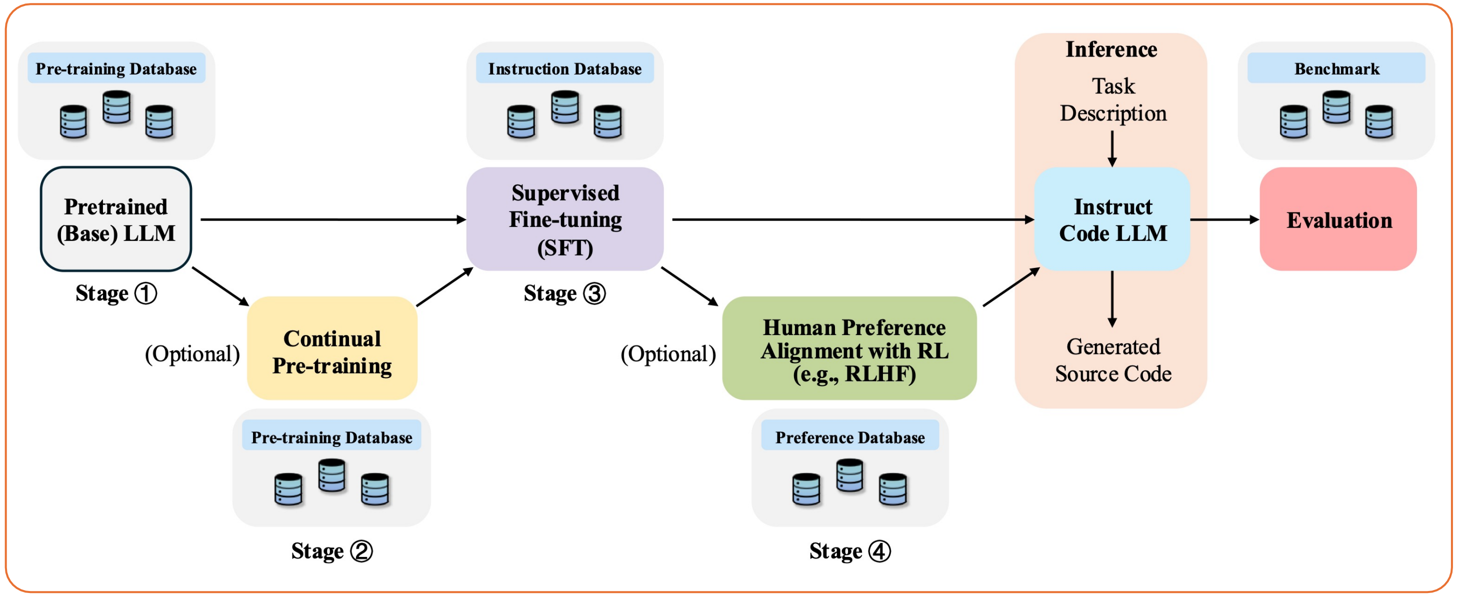


Figure 2 Scaling law IsoFLOPs curves between 6×10^{18} and 10^{22} FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

Today’s Agenda

- Pre-training stage
  - ~~Model architecture~~
  - ~~Pre-training dataset~~
  - Learning objectives
  - Evaluation dataset
- Special topics
  - Post-training staging
  - Scaling law
  - Hallucination



Pre-training: Learning Objectives

- Causal Language Modeling
  - Next token prediction
  - Infilling
- Auxiliary pre-training tasks
  - Masked token prediction
  - (Coding) Masked identifier prediction
  - (Coding) Identifier tagging
  - (Coding) Text-code matching
  - (Coding) Text-code contrastive learning

Pre-training: Learning Objectives

- Learning Objective (Machine Learning 101)
  - Loss function $\mathcal{L}(\mathbf{x}; \theta)$ where θ is the model parameter

$$\theta = \mathrm{argmin}_{\theta} \sum_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}; \theta)$$

Pre-training: Learning Objectives

- Learning Objective (Machine Learning 101)
  - Loss function $\mathcal{L}(\mathbf{x};\theta)$ where θ is the model parameter

$$\theta = \mathrm{argmin}_{\theta} \sum_{\mathbf{x}\in\mathcal{D}} \mathcal{L}(\mathbf{x};\theta)$$

- Next-token prediction

$$\mathcal{L}(\mathbf{x};\theta) = \sum_{i=1}^{n} -\log P_{\theta}(x_i \mid \mathbf{x}_{<i})$$

Pre-training: Learning Objectives

- Learning Objective (Machine Learning 101)
    - Loss function $\mathcal{L}(\mathbf{x};\theta)$ where θ is the model parameter

$$\theta = \text{argmin}_{\theta} \sum_{\mathbf{x}\in\mathcal{D}} \mathcal{L}(\mathbf{x};\theta)$$

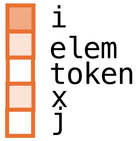
- Next-token prediction

$$\mathcal{L}(\mathbf{x};\theta) = \sum_{i=1}^{n} -\log P_{\theta}(x_i \mid \mathbf{x}_{<i})$$

- Example

```
[for, i, in, range, (, 10, ), :, print, (]      i
```

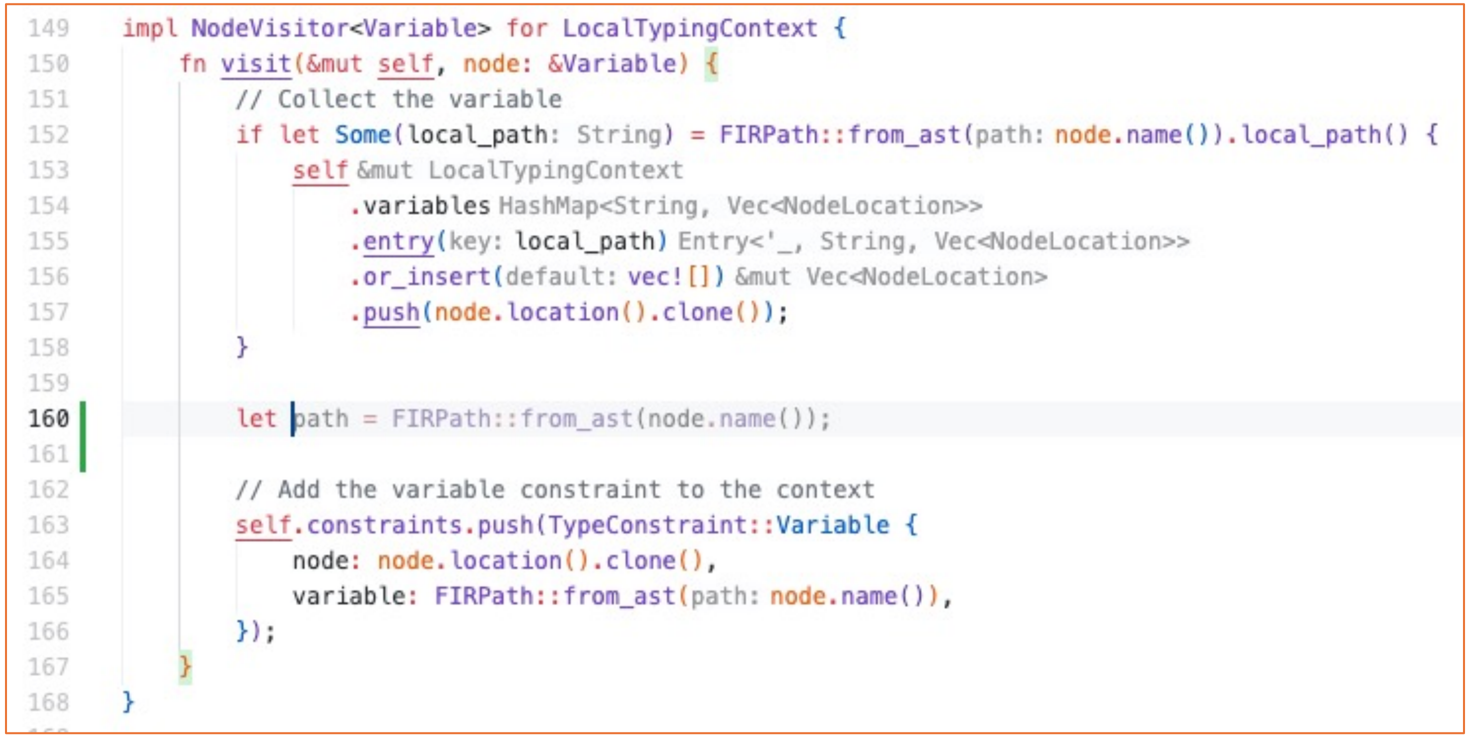
$\mathbf{x}_{<11}$ x_i



$P_{\theta}(x_i \mid \mathbf{x}_{<i})$

Pre-training: Learning Objectives

- Next-token prediction
  - Taking prefix $\mathbf{x}_{<i}$ and predict the next token x_i
  - But what about code editing happening in the middle?



```
impl NodeVisitor<Variable> for LocalTypingContext {
    fn visit(&mut self, node: &Variable) {
        // Collect the variable
        if let Some(local_path: String) = FIRPath::from_ast(path: node.name()).local_path() {
            self &mut LocalTypingContext
                .variables HashMap<String, Vec<NodeLocation>>
                .entry(key: local_path) Entry<'_, String, Vec<NodeLocation>>
                .or_insert(default: vec![]) &mut Vec<NodeLocation>
                .push(node.location().clone());
        }

        let path = FIRPath::from_ast(node.name());

        // Add the variable constraint to the context
        self.constraints.push(TypeConstraint::Variable {
            node: node.location().clone(),
            variable: FIRPath::from_ast(path: node.name()),
        });
    }
}
```

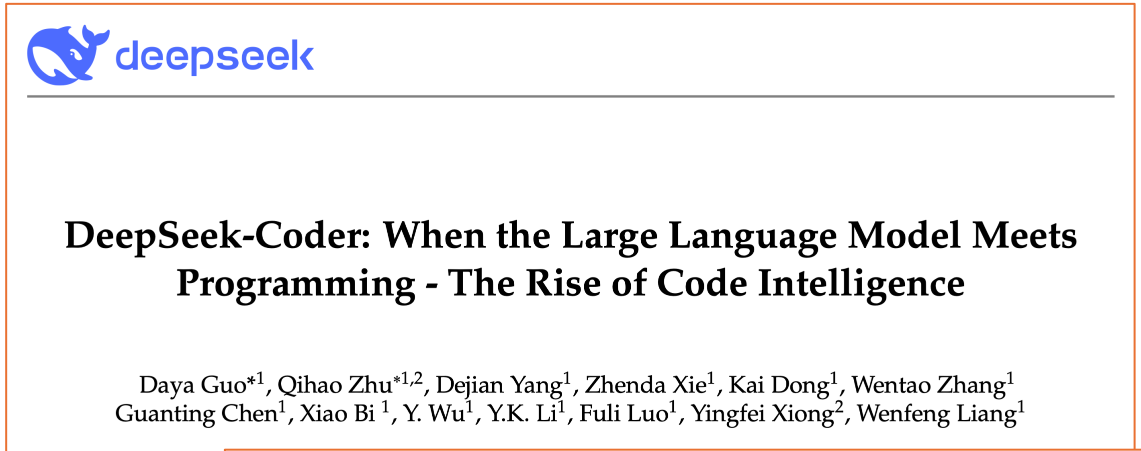
Pre-training: Learning Objectives

- Next-token prediction
  - Taking prefix $\mathbf{x}_{<i}$ and predict the next token x_i
  - But what about code editing happening in the middle?
- Infilling
  - Assume prefix $\mathbf{x}_{<i}$ and suffix $\mathbf{x}_{>j}$, predict the middle infill $\mathbf{x}_{i:j}$
  - Idea: reduce the problem of infilling to next-token prediction

DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

Daya Guo*[1], Qihao Zhu*[1,2], Dejian Yang[1], Zhenda Xie[1], Kai Dong[1], Wentao Zhang[1]
Guanting Chen[1], Xiao Bi [1], Y. Wu[1], Y.K. Li[1], Fuli Luo[1], Yingfei Xiong[2], Wenfeng Liang[1]

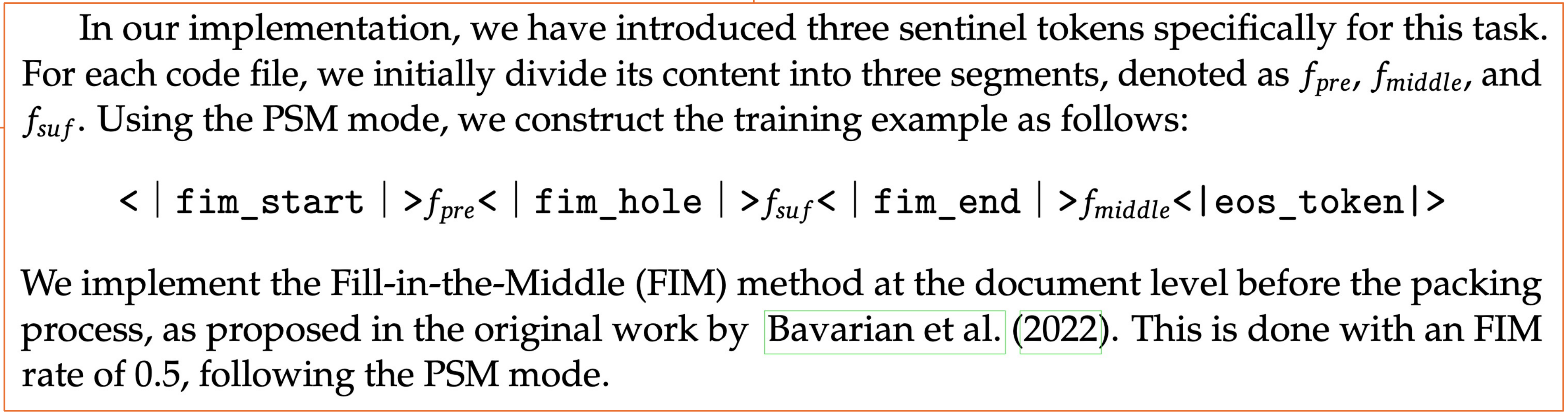
[1]DeepSeek-AI
[2]Key Lab of HCST (PKU), MOE; SCS, Peking University
`{zhuqh, guodaya}@deepseek.com`
`https://github.com/deepseek-ai/DeepSeek-Coder`



deepseek

DeepSeek-Coder: When the Large Language Model Meets Programming - The Rise of Code Intelligence

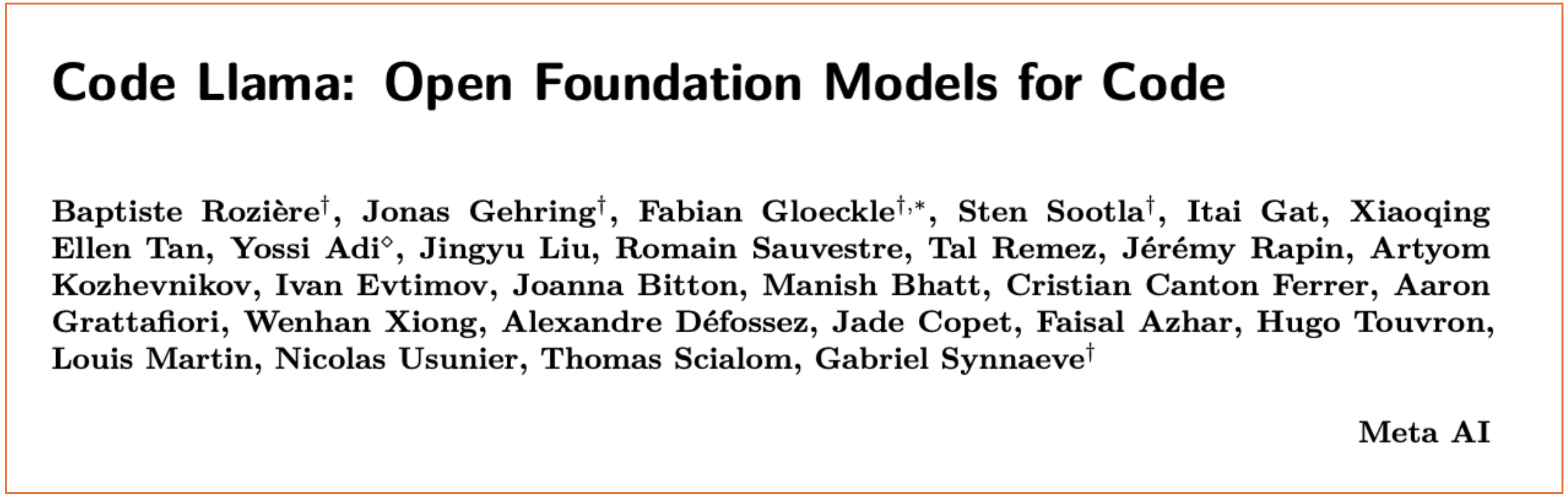
Daya Guo*[1], Qihao Zhu*[1,2], Dejian Yang[1], Zhenda Xie[1], Kai Dong[1], Wentao Zhang[1]
Guanting Chen[1], Xiao Bi [1], Y. Wu[1], Y.K. Li[1], Fuli Luo[1], Yingfei Xiong[2], Wenfeng Liang[1]



In our implementation, we have introduced three sentinel tokens specifically for this task. For each code file, we initially divide its content into three segments, denoted as f_{pre}, f_{middle}, and f_{suf}. Using the PSM mode, we construct the training example as follows:

<｜fim_start｜>f_{pre}<｜fim_hole｜>f_{suf}<｜fim_end｜>f_{middle}<|eos_token|>

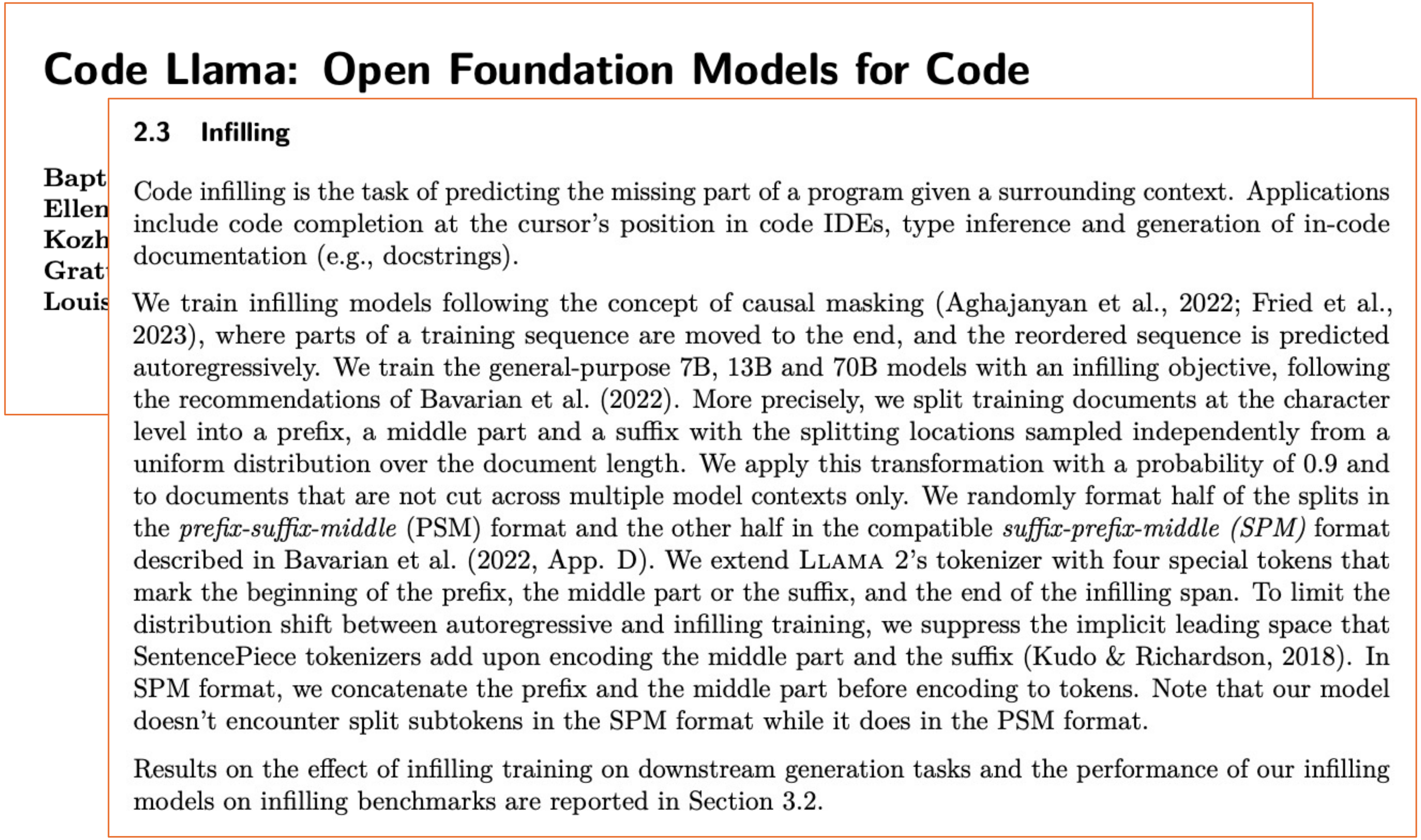
We implement the Fill-in-the-Middle (FIM) method at the document level before the packing process, as proposed in the original work by Bavarian et al. (2022). This is done with an FIM rate of 0.5, following the PSM mode.



Code Llama: Open Foundation Models for Code

Baptiste Rozière[†], Jonas Gehring[†], Fabian Gloeckle[†,*], Sten Sootla[†], Itai Gat, Xiaoqing Ellen Tan, Yossi Adi[◇], Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, Gabriel Synnaeve[†]

Meta AI



Code Llama: Open Foundation Models for Code

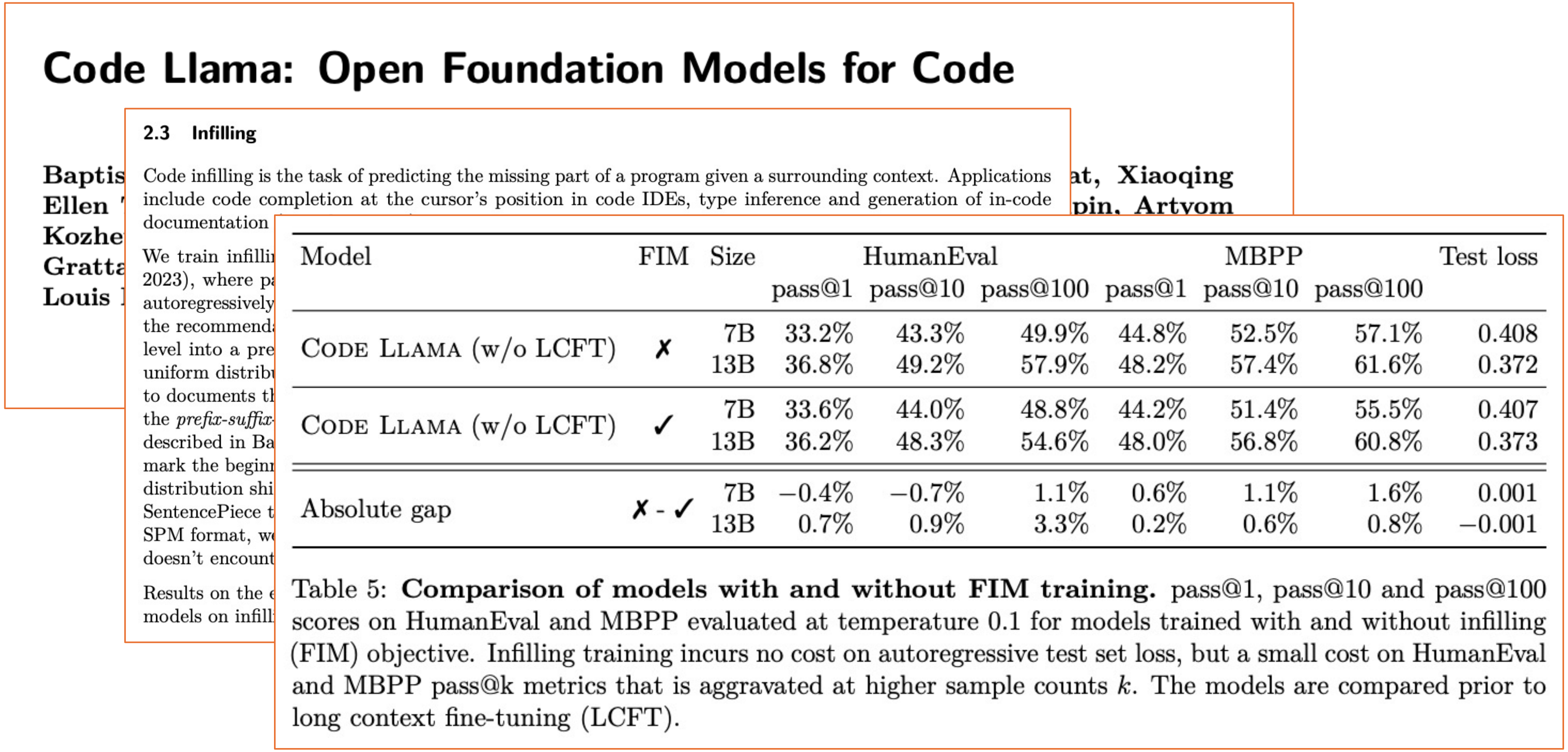
Bapt
Ellen
Kozh
Grat
Louis

2.3 Infilling

Code infilling is the task of predicting the missing part of a program given a surrounding context. Applications include code completion at the cursor's position in code IDEs, type inference and generation of in-code documentation (e.g., docstrings).

We train infilling models following the concept of causal masking (Aghajanyan et al., 2022; Fried et al., 2023), where parts of a training sequence are moved to the end, and the reordered sequence is predicted autoregressively. We train the general-purpose 7B, 13B and 70B models with an infilling objective, following the recommendations of Bavarian et al. (2022). More precisely, we split training documents at the character level into a prefix, a middle part and a suffix with the splitting locations sampled independently from a uniform distribution over the document length. We apply this transformation with a probability of 0.9 and to documents that are not cut across multiple model contexts only. We randomly format half of the splits in the *prefix-suffix-middle* (PSM) format and the other half in the compatible *suffix-prefix-middle (SPM)* format described in Bavarian et al. (2022, App. D). We extend Llama 2's tokenizer with four special tokens that mark the beginning of the prefix, the middle part or the suffix, and the end of the infilling span. To limit the distribution shift between autoregressive and infilling training, we suppress the implicit leading space that SentencePiece tokenizers add upon encoding the middle part and the suffix (Kudo & Richardson, 2018). In SPM format, we concatenate the prefix and the middle part before encoding to tokens. Note that our model doesn't encounter split subtokens in the SPM format while it does in the PSM format.

Results on the effect of infilling training on downstream generation tasks and the performance of our infilling models on infilling benchmarks are reported in Section 3.2.



Code Llama: Open Foundation Models for Code

Baptis … at, Xiaoqing
Ellen ' … pin, Artyom
Kozhe
Gratta
Louis I

2.3 Infilling

Code infilling is the task of predicting the missing part of a program given a surrounding context. Applications include code completion at the cursor's position in code IDEs, type inference and generation of in-code documentation

We train infilli
2023), where pa
autoregressively
the recommend
level into a pre
uniform distribu
to documents th
the *prefix-suffix*
described in Ba
mark the begin
distribution shi
SentencePiece t
SPM format, w
doesn't encount

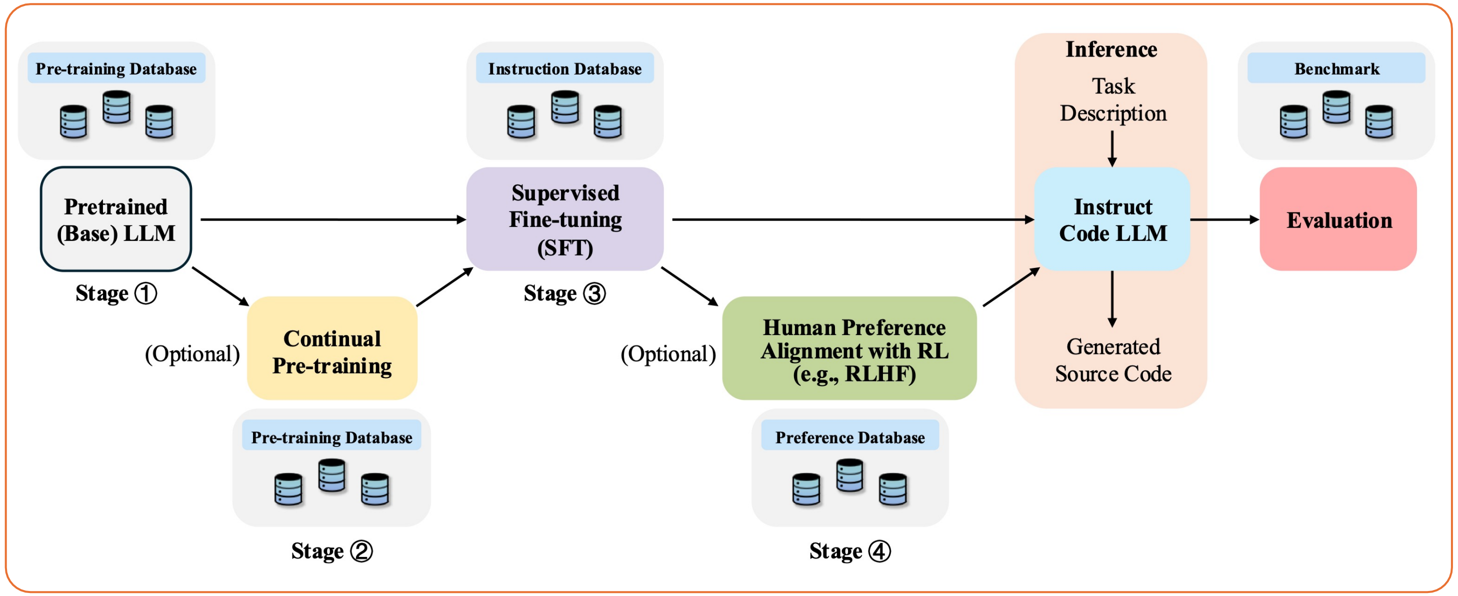
Results on the
models on infill

Model	FIM	Size	HumanEval			MBPP			Test loss
			pass@1	pass@10	pass@100	pass@1	pass@10	pass@100	
Code Llama (w/o LCFT)	✗	7B	33.2%	43.3%	49.9%	44.8%	52.5%	57.1%	0.408
		13B	36.8%	49.2%	57.9%	48.2%	57.4%	61.6%	0.372
Code Llama (w/o LCFT)	✓	7B	33.6%	44.0%	48.8%	44.2%	51.4%	55.5%	0.407
		13B	36.2%	48.3%	54.6%	48.0%	56.8%	60.8%	0.373
Absolute gap	✗ - ✓	7B	−0.4%	−0.7%	1.1%	0.6%	1.1%	1.6%	0.001
		13B	0.7%	0.9%	3.3%	0.2%	0.6%	0.8%	−0.001

Table 5: **Comparison of models with and without FIM training.** pass@1, pass@10 and pass@100 scores on HumanEval and MBPP evaluated at temperature 0.1 for models trained with and without infilling (FIM) objective. Infilling training incurs no cost on autoregressive test set loss, but a small cost on HumanEval and MBPP pass@k metrics that is aggravated at higher sample counts *k*. The models are compared prior to long context fine-tuning (LCFT).

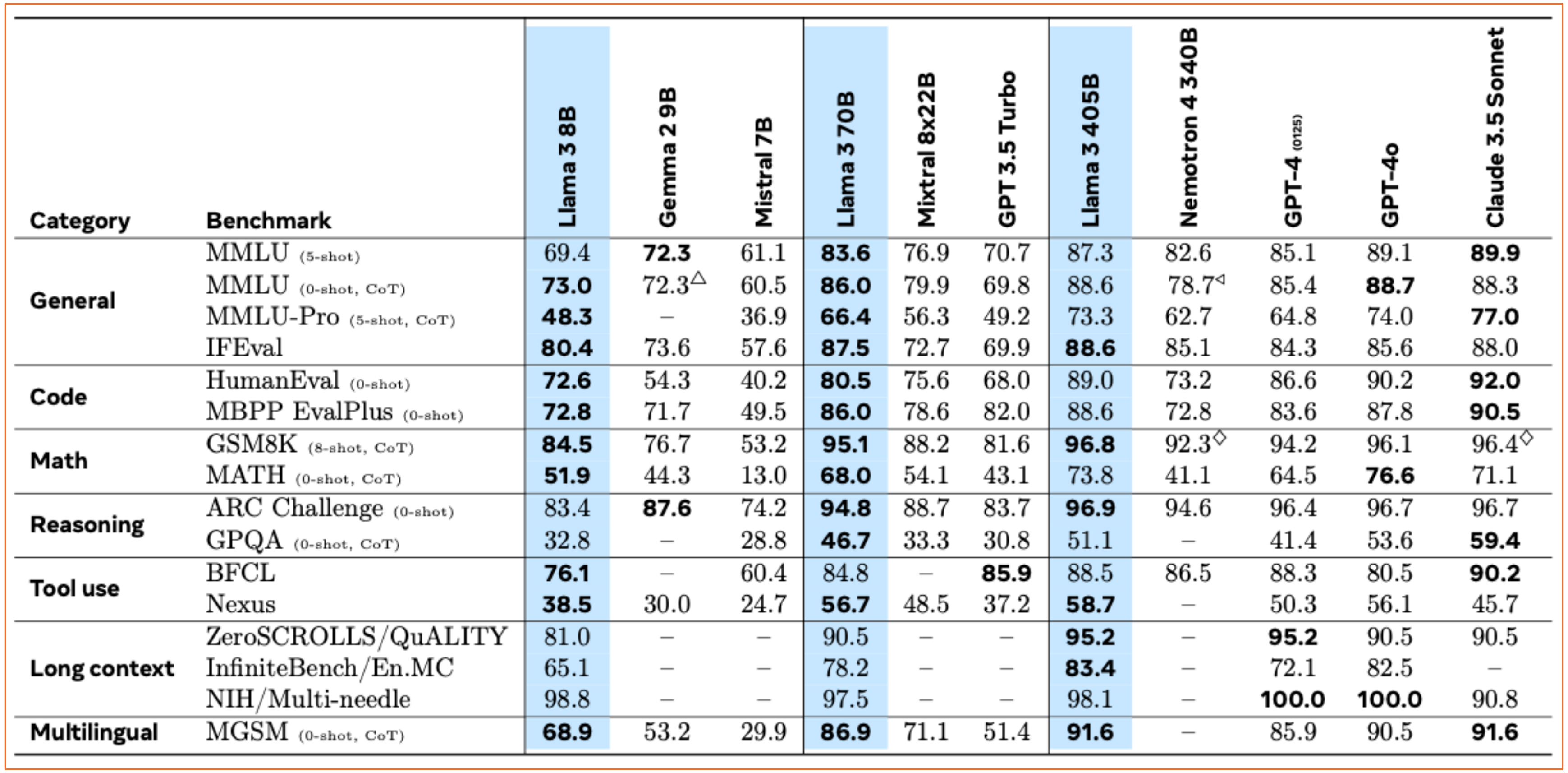
Today's Agenda

- Pre-training stage
  - ~~Model architecture~~
  - ~~Pre-training dataset~~
  - ~~Learning objectives~~
  - Evaluation dataset
- Special topics
  - Post-training staging
  - Scaling law
  - Hallucination



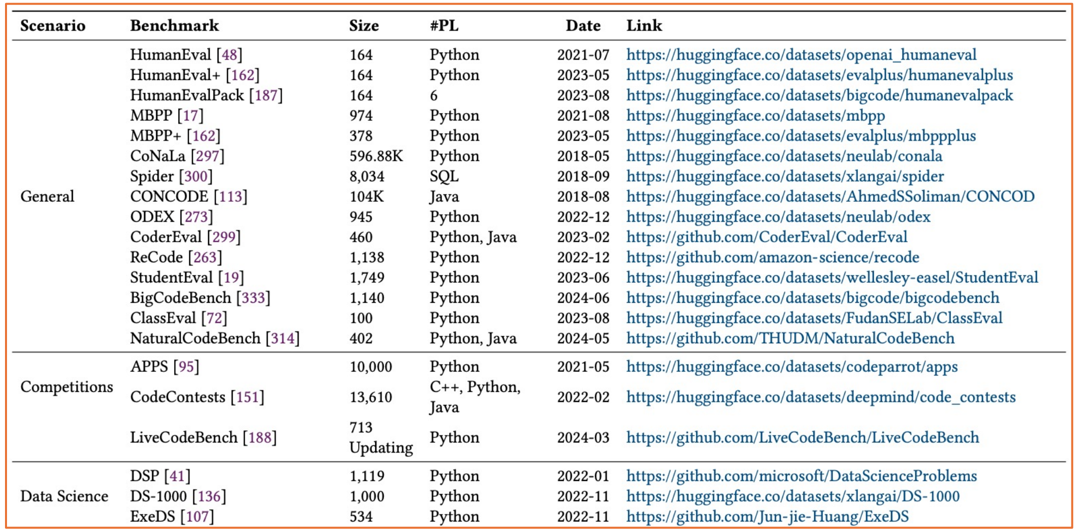
Evaluation Benchmark

- Coding benchmarks can be used to evaluate LLMs' abilities

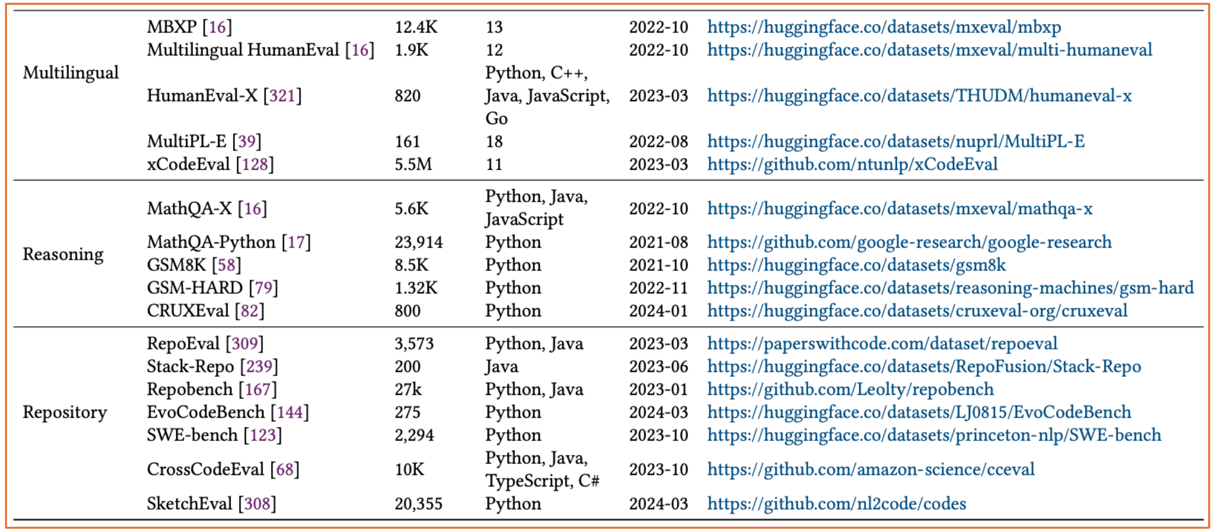


Category	Benchmark	Llama 3 8B	Gemma 2 9B	Mistral 7B	Llama 3 70B	Mixtral 8x22B	GPT 3.5 Turbo	Llama 3 405B	Nemotron 4 340B	GPT-4 (0125)	GPT-4o	Claude 3.5 Sonnet
General	MMLU (5-shot)	69.4	**72.3**	61.1	**83.6**	76.9	70.7	87.3	82.6	85.1	89.1	**89.9**
	MMLU (0-shot, CoT)	**73.0**	72.3△	60.5	**86.0**	79.9	69.8	88.6	78.7◁	85.4	**88.7**	88.3
	MMLU-Pro (5-shot, CoT)	**48.3**	–	36.9	**66.4**	56.3	49.2	73.3	62.7	64.8	74.0	**77.0**
	IFEval	**80.4**	73.6	57.6	**87.5**	72.7	69.9	**88.6**	85.1	84.3	85.6	88.0
Code	HumanEval (0-shot)	**72.6**	54.3	40.2	**80.5**	75.6	68.0	89.0	73.2	86.6	90.2	**92.0**
	MBPP EvalPlus (0-shot)	**72.8**	71.7	49.5	**86.0**	78.6	82.0	88.6	72.8	83.6	87.8	**90.5**
Math	GSM8K (8-shot, CoT)	**84.5**	76.7	53.2	**95.1**	88.2	81.6	**96.8**	92.3◇	94.2	96.1	96.4◇
	MATH (0-shot, CoT)	**51.9**	44.3	13.0	**68.0**	54.1	43.1	73.8	41.1	64.5	**76.6**	71.1
Reasoning	ARC Challenge (0-shot)	83.4	**87.6**	74.2	**94.8**	88.7	83.7	**96.9**	94.6	96.4	96.7	96.7
	GPQA (0-shot, CoT)	32.8	–	28.8	**46.7**	33.3	30.8	51.1	–	41.4	53.6	**59.4**
Tool use	BFCL	**76.1**	–	60.4	84.8	–	**85.9**	88.5	86.5	88.3	80.5	**90.2**
	Nexus	**38.5**	30.0	24.7	**56.7**	48.5	37.2	**58.7**	–	50.3	56.1	45.7
Long context	ZeroSCROLLS/QuALITY	81.0	–	–	90.5	–	–	**95.2**	–	**95.2**	90.5	90.5
	InfiniteBench/En.MC	65.1	–	–	78.2	–	–	**83.4**	–	72.1	82.5	–
	NIH/Multi-needle	98.8	–	–	97.5	–	–	98.1	–	**100.0**	**100.0**	90.8
Multilingual	MGSM (0-shot, CoT)	**68.9**	53.2	29.9	**86.9**	71.1	51.4	**91.6**	–	85.9	90.5	**91.6**

Evaluation Benchmark



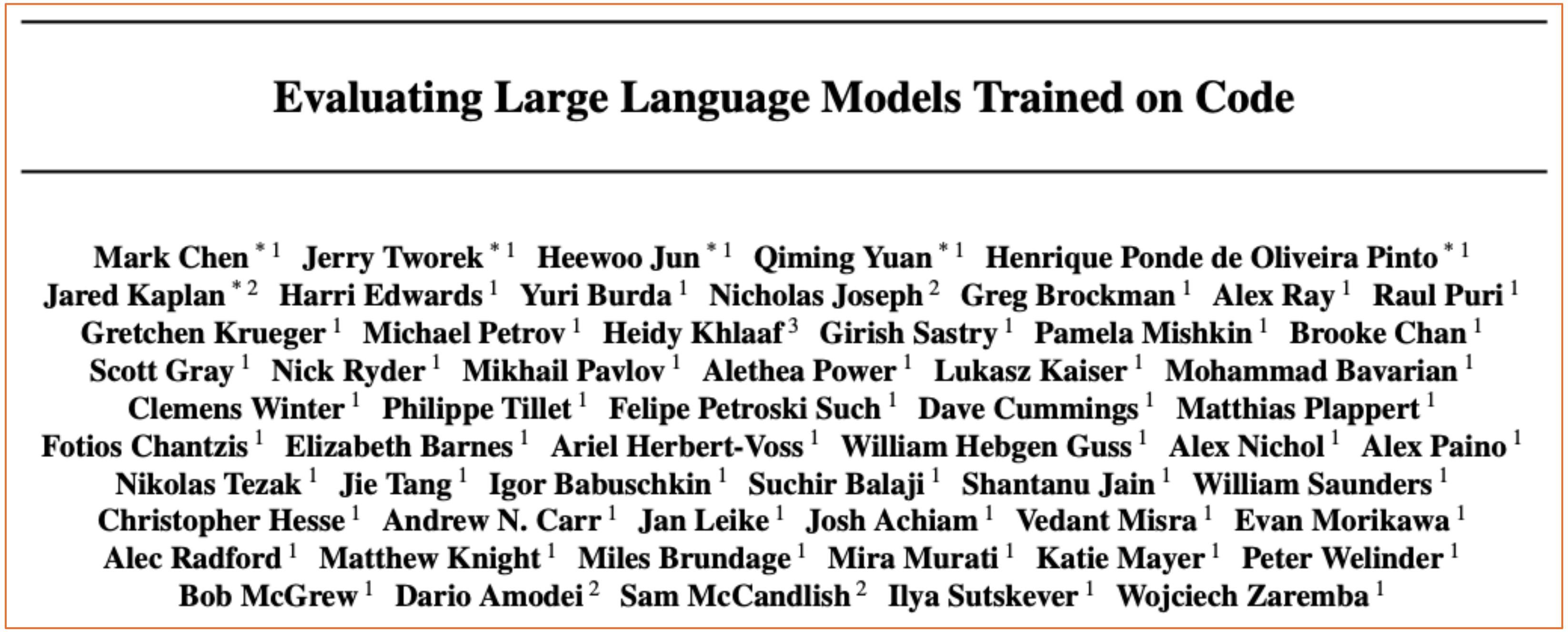
Scenario	Benchmark	Size	#PL	Date	Link
General	HumanEval [48]	164	Python	2021-07	https://huggingface.co/datasets/openai_humaneval
	HumanEval+ [162]	164	Python	2023-05	https://huggingface.co/datasets/evalplus/humanevalplus
	HumanEvalPack [187]	164	6	2023-08	https://huggingface.co/datasets/bigcode/humanevalpack
	MBPP [17]	974	Python	2021-08	https://huggingface.co/datasets/mbpp
	MBPP+ [162]	378	Python	2023-05	https://huggingface.co/datasets/evalplus/mbppplus
	CoNaLa [297]	596.88K	Python	2018-05	https://huggingface.co/datasets/neulab/conala
	Spider [300]	8,034	SQL	2018-09	https://huggingface.co/datasets/xlangai/spider
	CONCODE [113]	104K	Java	2018-08	https://huggingface.co/datasets/AhmedSSoliman/CONCOD
	ODEX [273]	945	Python	2022-12	https://huggingface.co/datasets/neulab/odex
	CoderEval [299]	460	Python, Java	2023-02	https://github.com/CoderEval/CoderEval
	ReCode [263]	1,138	Python	2022-12	https://github.com/amazon-science/recode
	StudentEval [19]	1,749	Python	2023-06	https://huggingface.co/datasets/wellesley-easel/StudentEval
	BigCodeBench [333]	1,140	Python	2024-06	https://huggingface.co/datasets/bigcode/bigcodebench
	ClassEval [72]	100	Python	2023-08	https://huggingface.co/datasets/FudanSELab/ClassEval
	NaturalCodeBench [314]	402	Python, Java	2024-05	https://github.com/THUDM/NaturalCodeBench
Competitions	APPS [95]	10,000	Python	2021-05	https://huggingface.co/datasets/codeparrot/apps
	CodeContests [151]	13,610	C++, Python, Java	2022-02	https://huggingface.co/datasets/deepmind/code_contests
	LiveCodeBench [188]	713 Updating	Python	2024-03	https://github.com/LiveCodeBench/LiveCodeBench
Data Science	DSP [41]	1,119	Python	2022-01	https://github.com/microsoft/DataScienceProblems
	DS-1000 [136]	1,000	Python	2022-11	https://huggingface.co/datasets/xlangai/DS-1000
	ExeDS [107]	534	Python	2022-11	https://github.com/Jun-jie-Huang/ExeDS



Multilingual	MBXP [16]	12.4K	13	2022-10	https://huggingface.co/datasets/mxeval/mbxp
	Multilingual HumanEval [16]	1.9K	12	2022-10	https://huggingface.co/datasets/mxeval/multi-humaneval
	HumanEval-X [321]	820	Python, C++, Java, JavaScript, Go	2023-03	https://huggingface.co/datasets/THUDM/humaneval-x
	MultiPL-E [39]	161	18	2022-08	https://huggingface.co/datasets/nuprl/MultiPL-E
	xCodeEval [128]	5.5M	11	2023-03	https://github.com/ntunlp/xCodeEval
Reasoning	MathQA-X [16]	5.6K	Python, Java, JavaScript	2022-10	https://huggingface.co/datasets/mxeval/mathqa-x
	MathQA-Python [17]	23,914	Python	2021-08	https://github.com/google-research/google-research
	GSM8K [58]	8.5K	Python	2021-10	https://huggingface.co/datasets/gsm8k
	GSM-HARD [79]	1.32K	Python	2022-11	https://huggingface.co/datasets/reasoning-machines/gsm-hard
	CRUXEval [82]	800	Python	2024-01	https://huggingface.co/datasets/cruxeval-org/cruxeval
Repository	RepoEval [309]	3,573	Python, Java	2023-03	https://paperswithcode.com/dataset/repoeval
	Stack-Repo [239]	200	Java	2023-06	https://huggingface.co/datasets/RepoFusion/Stack-Repo
	Repobench [167]	27k	Python, Java	2023-01	https://github.com/Leolty/repobench
	EvoCodeBench [144]	275	Python	2024-03	https://huggingface.co/datasets/LJ0815/EvoCodeBench
	SWE-bench [123]	2,294	Python	2023-10	https://huggingface.co/datasets/princeton-nlp/SWE-bench
	CrossCodeEval [68]	10K	Python, Java, TypeScript, C#	2023-10	https://github.com/amazon-science/cceval
	SketchEval [308]	20,355	Python	2024-03	https://github.com/nl2code/codes

A Survey on Large Language Models for Code Generation, Jiang et al., 2024

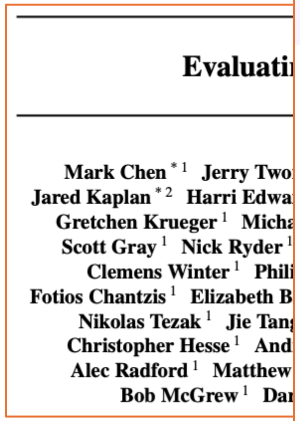
Evaluation Benchmark: HumanEval



Evaluating Large Language Models Trained on Code

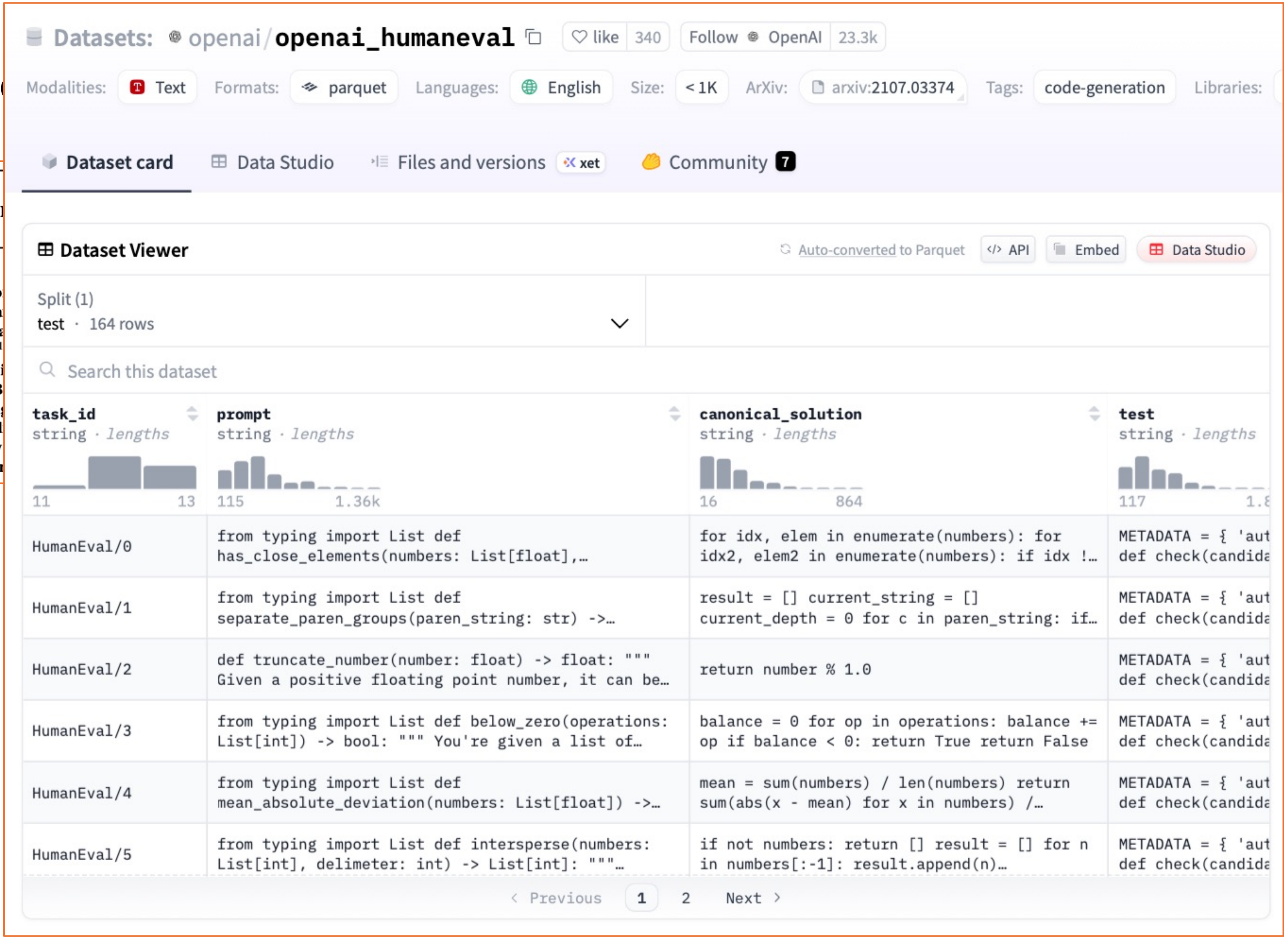
Mark Chen[*1] Jerry Tworek[*1] Heewoo Jun[*1] Qiming Yuan[*1] Henrique Ponde de Oliveira Pinto[*1]
Jared Kaplan[*2] Harri Edwards[1] Yuri Burda[1] Nicholas Joseph[2] Greg Brockman[1] Alex Ray[1] Raul Puri[1]
Gretchen Krueger[1] Michael Petrov[1] Heidy Khlaaf[3] Girish Sastry[1] Pamela Mishkin[1] Brooke Chan[1]
Scott Gray[1] Nick Ryder[1] Mikhail Pavlov[1] Alethea Power[1] Lukasz Kaiser[1] Mohammad Bavarian[1]
Clemens Winter[1] Philippe Tillet[1] Felipe Petroski Such[1] Dave Cummings[1] Matthias Plappert[1]
Fotios Chantzis[1] Elizabeth Barnes[1] Ariel Herbert-Voss[1] William Hebgen Guss[1] Alex Nichol[1] Alex Paino[1]
Nikolas Tezak[1] Jie Tang[1] Igor Babuschkin[1] Suchir Balaji[1] Shantanu Jain[1] William Saunders[1]
Christopher Hesse[1] Andrew N. Carr[1] Jan Leike[1] Josh Achiam[1] Vedant Misra[1] Evan Morikawa[1]
Alec Radford[1] Matthew Knight[1] Miles Brundage[1] Mira Murati[1] Katie Mayer[1] Peter Welinder[1]
Bob McGrew[1] Dario Amodei[2] Sam McCandlish[2] Ilya Sutskever[1] Wojciech Zaremba[1]

Evalu



Evaluati

Mark Chen [*1] Jerry Two
Jared Kaplan [*2] Harri Edwa
Gretchen Krueger [1] Micha
Scott Gray [1] Nick Ryder [1]
Clemens Winter [1] Phili
Fotios Chantzis [1] Elizabeth B
Nikolas Tezak [1] Jie Tang
Christopher Hesse [1] And
Alec Radford [1] Matthew
Bob McGrew [1] Dar



Datasets: openai/**openai_humaneval** like 340 Follow OpenAI 23.3k

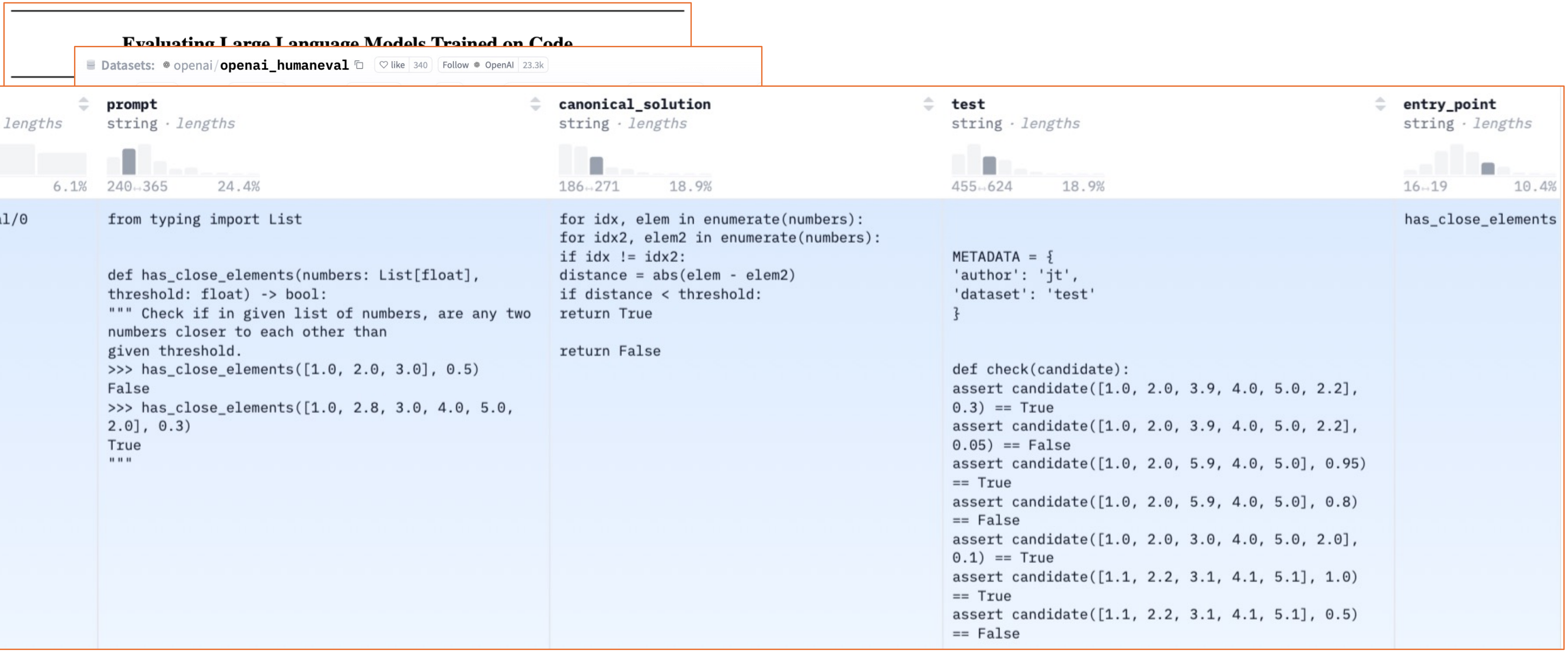
Modalities: Text Formats: parquet Languages: English Size: < 1K ArXiv: arxiv:2107.03374 Tags: code-generation Libraries:

Dataset Viewer

Split (1)
test · 164 rows

task_id string · lengths 11 13	prompt string · lengths 115 1.36k	canonical_solution string · lengths 16 864	test string · lengths 117 1.8
HumanEval/0	from typing import List def has_close_elements(numbers: List[float],…	for idx, elem in enumerate(numbers): for idx2, elem2 in enumerate(numbers): if idx !…	METADATA = { 'aut def check(candida
HumanEval/1	from typing import List def separate_paren_groups(paren_string: str) ->…	result = [] current_string = [] current_depth = 0 for c in paren_string: if…	METADATA = { 'aut def check(candida
HumanEval/2	def truncate_number(number: float) -> float: """ Given a positive floating point number, it can be…	return number % 1.0	METADATA = { 'aut def check(candida
HumanEval/3	from typing import List def below_zero(operations: List[int]) -> bool: """ You're given a list of…	balance = 0 for op in operations: balance += op if balance < 0: return True return False	METADATA = { 'aut def check(candida
HumanEval/4	from typing import List def mean_absolute_deviation(numbers: List[float]) ->…	mean = sum(numbers) / len(numbers) return sum(abs(x - mean) for x in numbers) /…	METADATA = { 'aut def check(candida
HumanEval/5	from typing import List def intersperse(numbers: List[int], delimeter: int) -> List[int]: """…	if not numbers: return [] result = [] for n in numbers[:-1]: result.append(n)…	METADATA = { 'aut def check(candida

Evaluation Benchmark: HumanEval



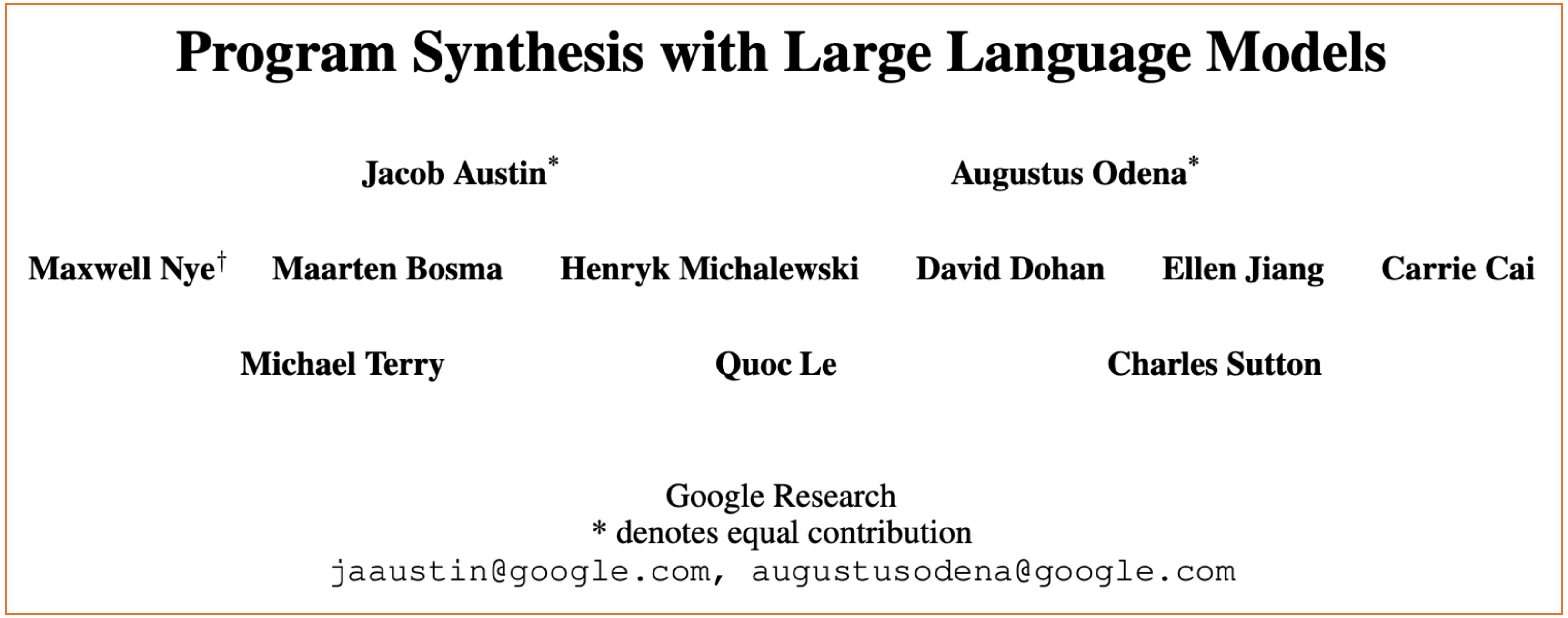
Evaluating Large Language Models Trained on Code

Datasets: openai/**openai_humaneval** like 340 Follow OpenAI 23.3k

task_id	prompt	canonical_solution	test	entry_point
string · *lengths*	string · *lengths*	string · *lengths*	string · *lengths*	string · *lengths*
11↔12 6.1%	240↔365 24.4%	186↔271 18.9%	455↔624 18.9%	16↔19 10.4%

HumanEval/0	from typing import List
	
	def has_close_elements(numbers: List[float], threshold: float) -> bool:
	""" Check if in given list of numbers, are any two numbers closer to each other than
	given threshold.
	>>> has_close_elements([1.0, 2.0, 3.0], 0.5)
	False
	>>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
	True
	"""
		for idx, elem in enumerate(numbers):
		for idx2, elem2 in enumerate(numbers):
		if idx != idx2:
		distance = abs(elem - elem2)
		if distance < threshold:
		return True
		
		return False
			
			METADATA = {
			'author': 'jt',
			'dataset': 'test'
			}
			
			def check(candidate):
			assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.3) == True
			assert candidate([1.0, 2.0, 3.9, 4.0, 5.0, 2.2], 0.05) == False
			assert candidate([1.0, 2.0, 5.9, 4.0, 5.0], 0.95) == True
			assert candidate([1.0, 2.0, 5.9, 4.0, 5.0], 0.8) == False
			assert candidate([1.0, 2.0, 3.0, 4.0, 5.0, 2.0], 0.1) == True
			assert candidate([1.1, 2.2, 3.1, 4.1, 5.1], 1.0) == True
			assert candidate([1.1, 2.2, 3.1, 4.1, 5.1], 0.5) == False
				has_close_elements

Evaluation Benchmark: MBPP



Program Synthesis with Large Language Models

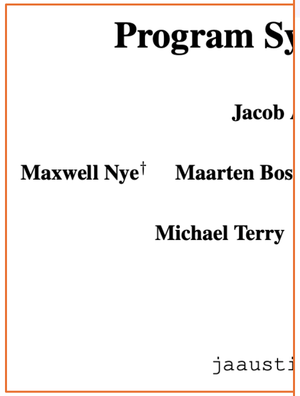
Jacob Austin* **Augustus Odena***

Maxwell Nye† **Maarten Bosma** **Henryk Michalewski** **David Dohan** **Ellen Jiang** **Carrie Cai**

Michael Terry **Quoc Le** **Charles Sutton**

Google Research
* denotes equal contribution
jaaustin@google.com, augustusodena@google.com

Evalu



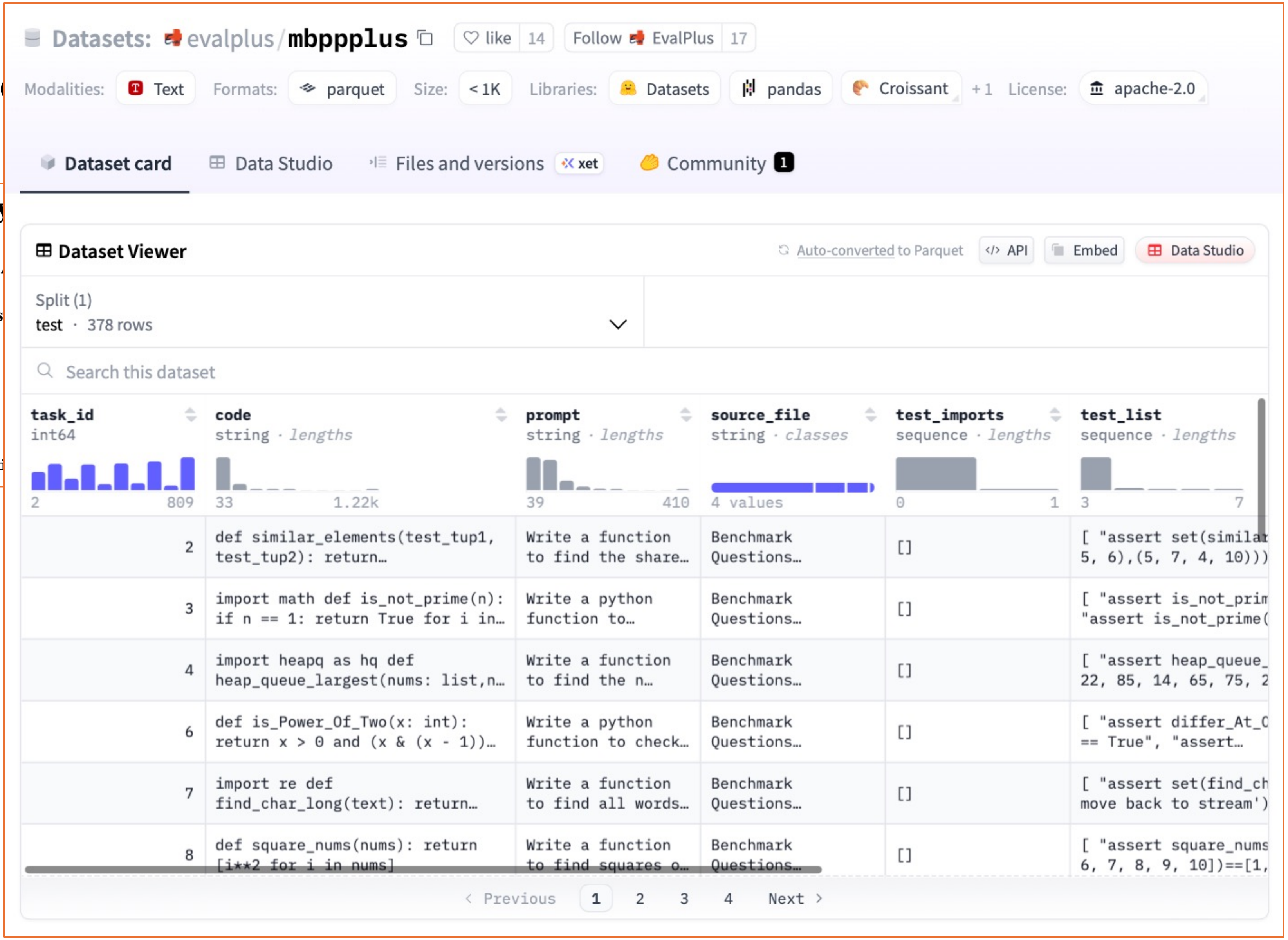
Program Sy

Jacob

Maxwell Nye[†] Maarten Bos

Michael Terry

jaausti



Datasets: evalplus/**mbppplus** ♡ like 14 Follow EvalPlus 17

Modalities: Text Formats: parquet Size: < 1K Libraries: Datasets pandas Croissant + 1 License: apache-2.0

Dataset card | Data Studio | Files and versions xet | Community 1

Dataset Viewer Auto-converted to Parquet | API | Embed | Data Studio

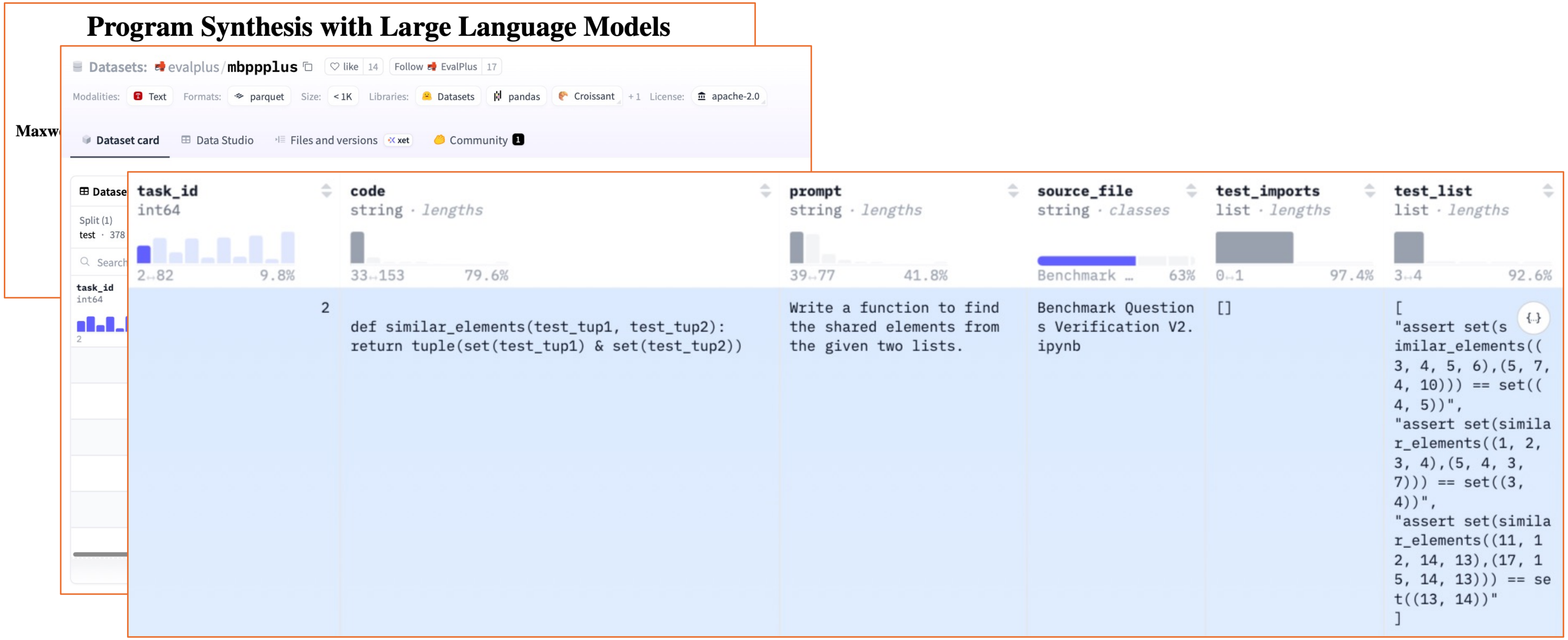
Split (1)
test · 378 rows

Search this dataset

task_id int64 2 – 809	code string · lengths 33 – 1.22k	prompt string · lengths 39 – 410	source_file string · classes 4 values	test_imports sequence · lengths 0 – 1	test_list sequence · lengths 3 – 7
2	def similar_elements(test_tup1, test_tup2): return…	Write a function to find the share…	Benchmark Questions…	[]	["assert set(similar 5, 6),(5, 7, 4, 10)))
3	import math def is_not_prime(n): if n == 1: return True for i in…	Write a python function to…	Benchmark Questions…	[]	["assert is_not_prim "assert is_not_prime(
4	import heapq as hq def heap_queue_largest(nums: list,n…	Write a function to find the n…	Benchmark Questions…	[]	["assert heap_queue_ 22, 85, 14, 65, 75, 2
6	def is_Power_Of_Two(x: int): return x > 0 and (x & (x - 1))…	Write a python function to check…	Benchmark Questions…	[]	["assert differ_At_O == True", "assert…
7	import re def find_char_long(text): return…	Write a function to find all words…	Benchmark Questions…	[]	["assert set(find_ch move back to stream')
8	def square_nums(nums): return [i**2 for i in nums]	Write a function to find squares o…	Benchmark Questions…	[]	["assert square_nums 6, 7, 8, 9, 10])==[1,

‹ Previous 1 2 3 4 Next ›

Evaluation Benchmark: MBPP



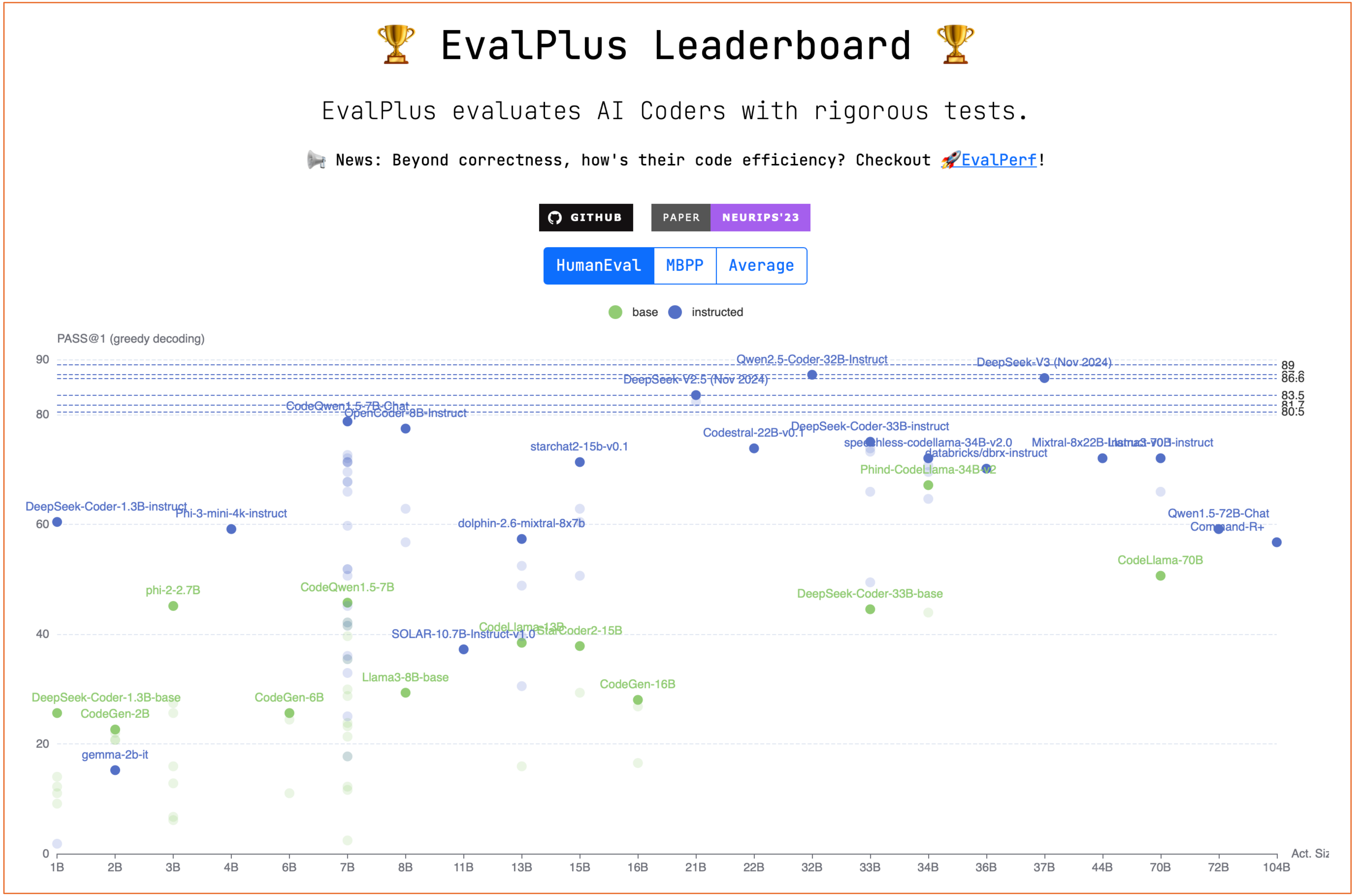
Program Synthesis with Large Language Models

Maxw

Datasets: evalplus/mbppplus

task_id int64 2↔82 9.8%	code string · *lengths* 33↔153 79.6%	prompt string · *lengths* 39↔77 41.8%	source_file string · *classes* Benchmark … 63%	test_imports list · *lengths* 0↔1 97.4%	test_list list · *lengths* 3↔4 92.6%

2	def similar_elements(test_tup1, test_tup2):
	return tuple(set(test_tup1) & set(test_tup2))
		Write a function to find the shared elements from the given two lists.	Benchmark Questions Verification V2.ipynb	[]	["assert set(similar_elements((3, 4, 5, 6),(5, 7, 4, 10))) == set((4, 5))", "assert set(similar_elements((1, 2, 3, 4),(5, 4, 3, 7))) == set((3, 4))", "assert set(similar_elements((11, 12, 14, 13),(17, 15, 14, 13))) == set((13, 14))"]



🏆 EvalPlus Leaderboard 🏆

EvalPlus evaluates AI Coders with rigorous tests.

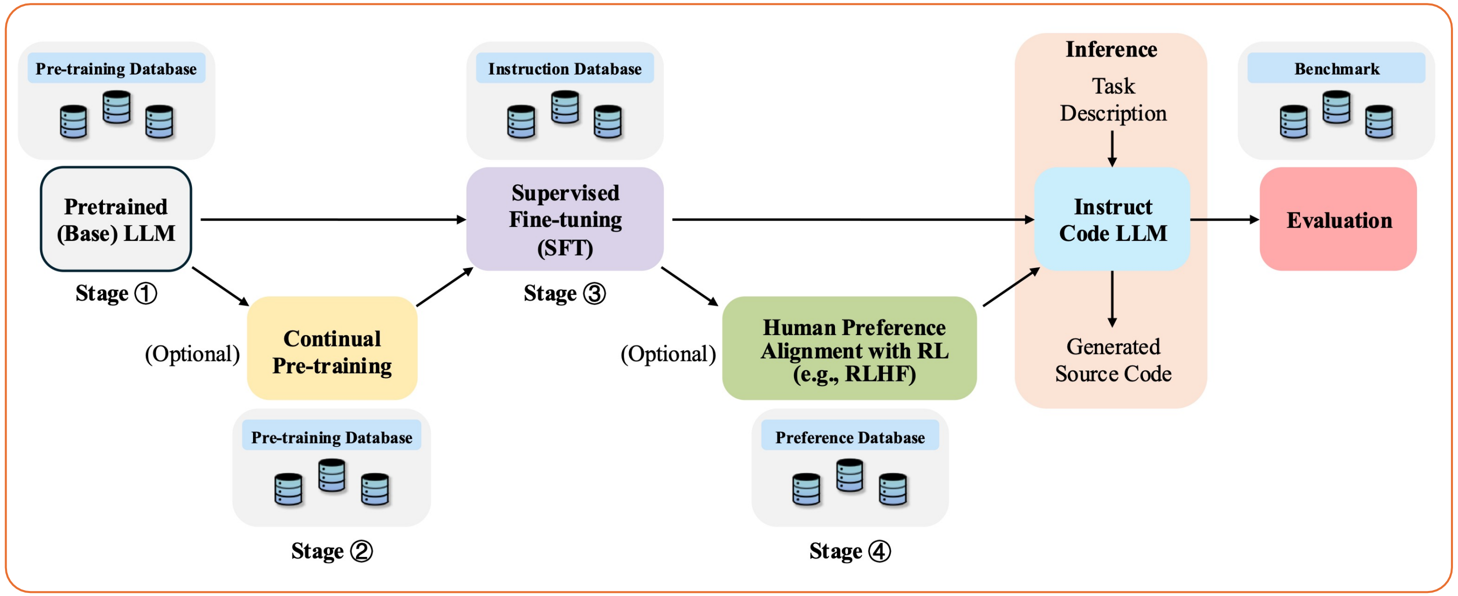
📢 News: Beyond correctness, how's their code efficiency? Checkout 🚀EvalPerf!

GITHUB | PAPER NEURIPS'23

HumanEval | MBPP | Average

Today’s Agenda

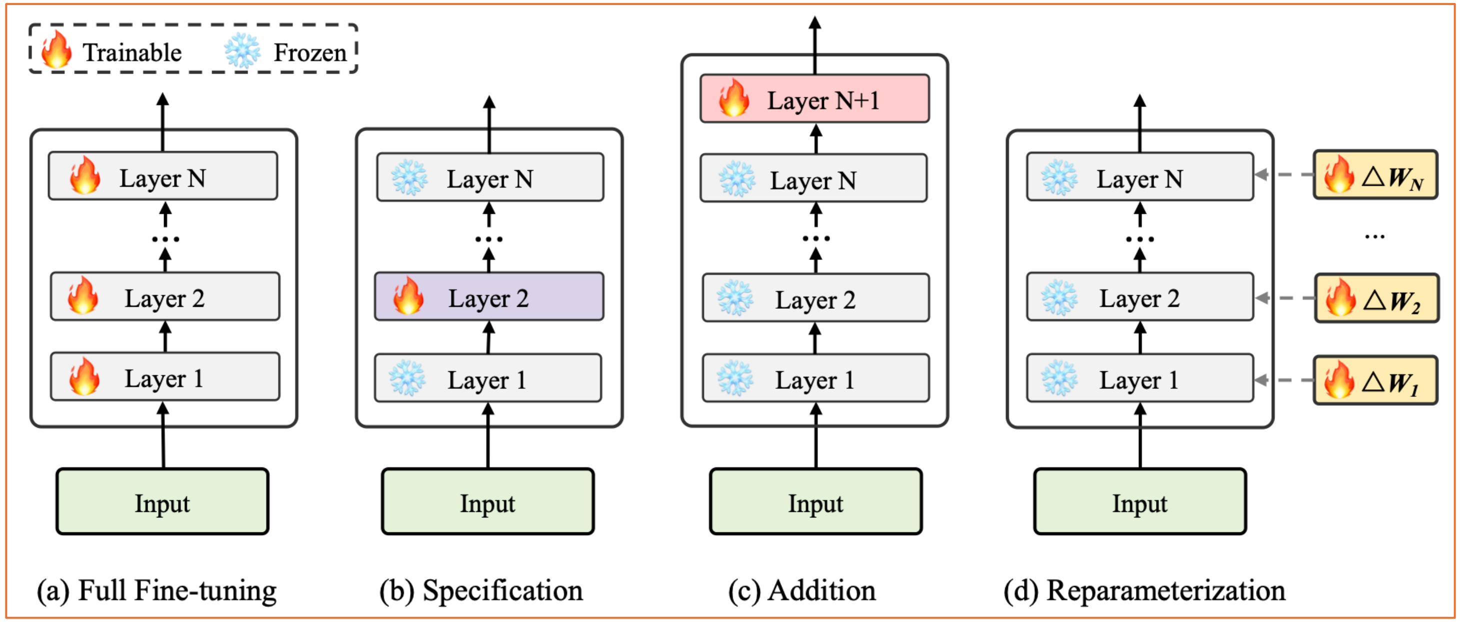
- Pre-training stage
  - ~~Model architecture~~
  - ~~Pre-training dataset~~
  - ~~Learning objectives~~
  - ~~Evaluation dataset~~
- Special topics
  - Post-training staging
  - Scaling law
  - Hallucination



Post-Training Staging

- Instruction tuning
- Full parameter fine-tuning (FFT)
- Supervised fine-tuning (SFT)
- Parameter-efficient fine-tuning (PEFT)

- Reinforcement learning (RL)
  - Human Feedback
  - Logical Feedback
  - Compiler Feedback
  - LLM-as-judge Feedback



Scaling Law

- “Compute”: FLOP
  - FLOP: Floating point operations
  - Total training compute that aggregates over model size, dataset size and training duration
  - Approximation:
    - FLOPs $\approx 6 \times N \times D$
    - N: number of model parameters
    - D: number of dataset tokens
    - Factor 6: forward + backward passes + architecture constants

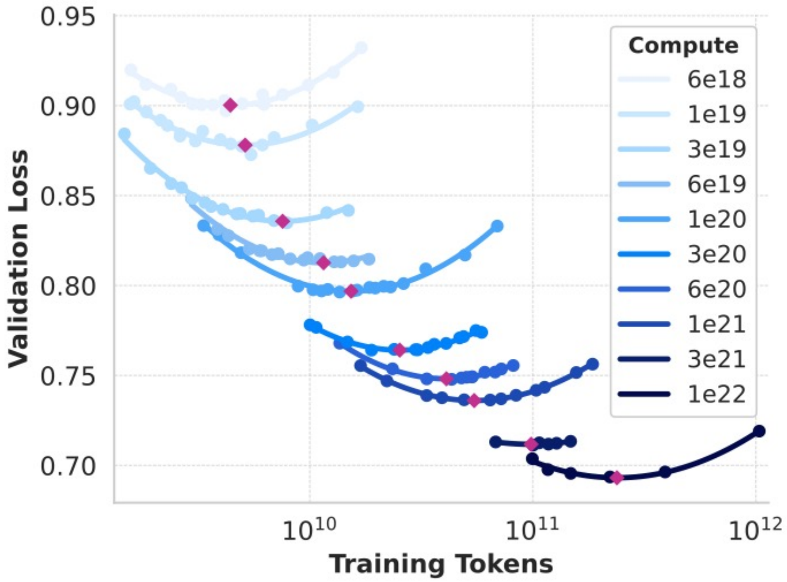
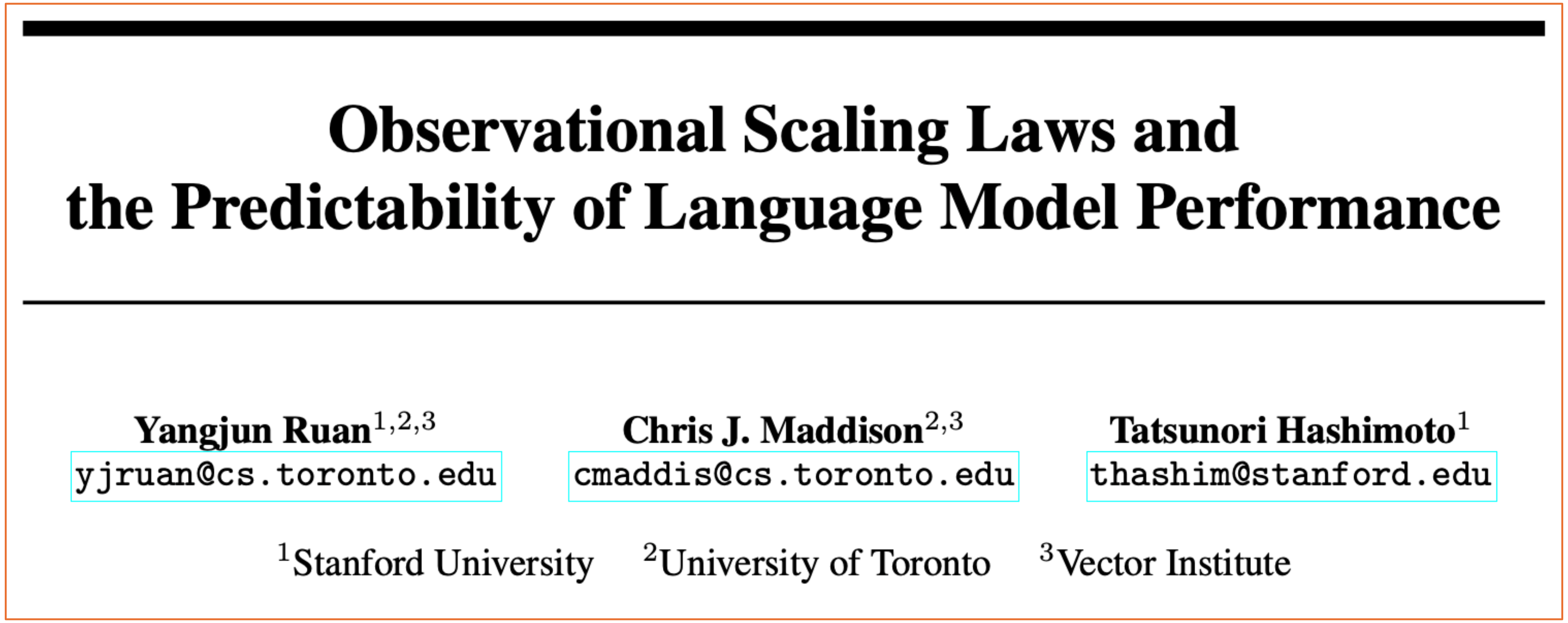


Figure 2 **Scaling law IsoFLOPs curves** between 6×10^{18} and 10^{22} FLOPs. The loss is the negative log-likelihood on a held-out validation set. We approximate measurements at each compute scale using a second degree polynomial.

Scaling Law



Observational Scaling Laws and the Predictability of Language Model Performance

Yangjun Ruan[1,2,3]
yjruan@cs.toronto.edu

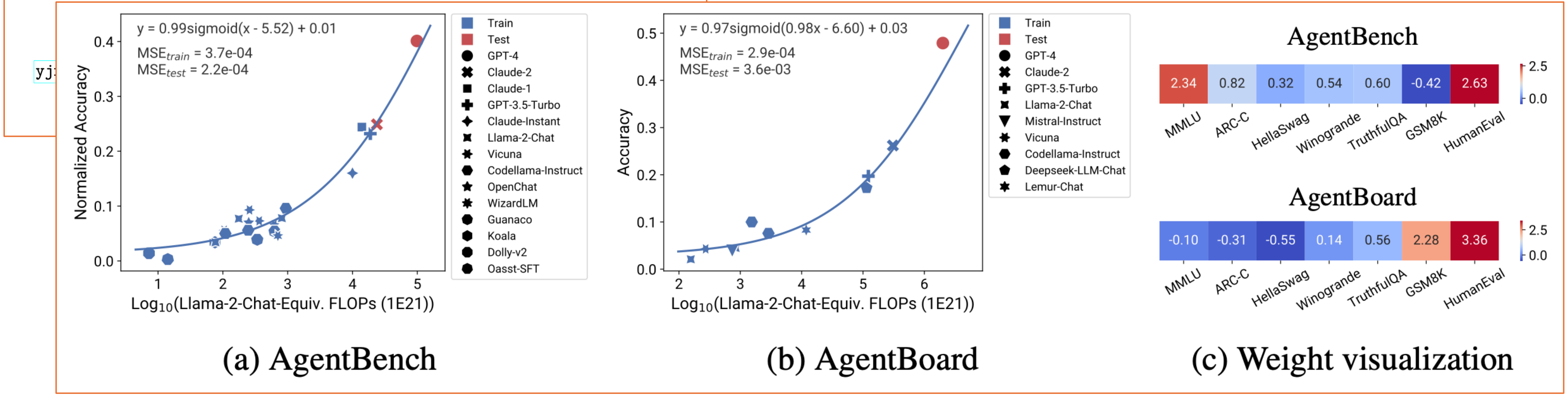
Chris J. Maddison[2,3]
cmaddis@cs.toronto.edu

Tatsunori Hashimoto[1]
thashim@stanford.edu

[1]Stanford University [2]University of Toronto [3]Vector Institute

Scaling Law

Observational Scaling Laws and the Predictability of Language Model Performance

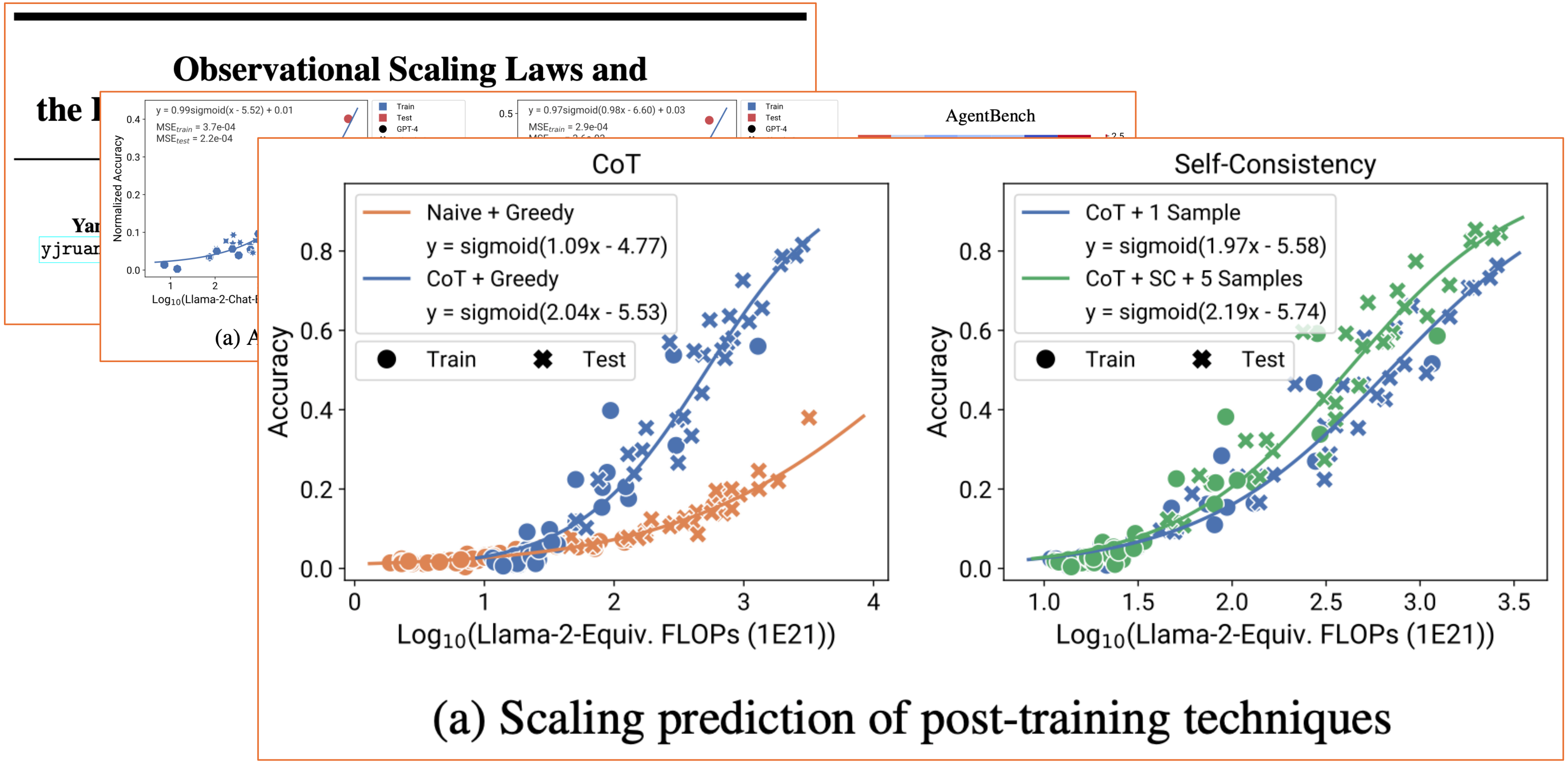


(a) AgentBench

(b) AgentBoard

(c) Weight visualization

Scaling Law



Observational Scaling Laws and

the I

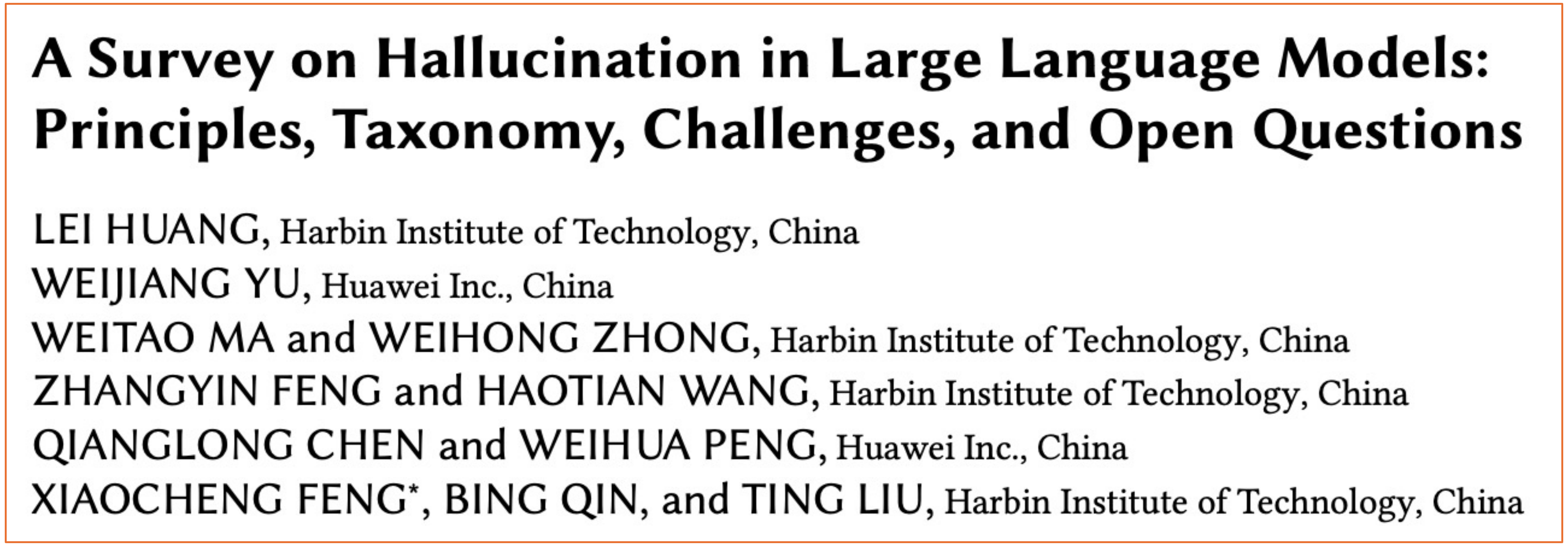
Yan

yjruan

(a) A

(a) Scaling prediction of post-training techniques

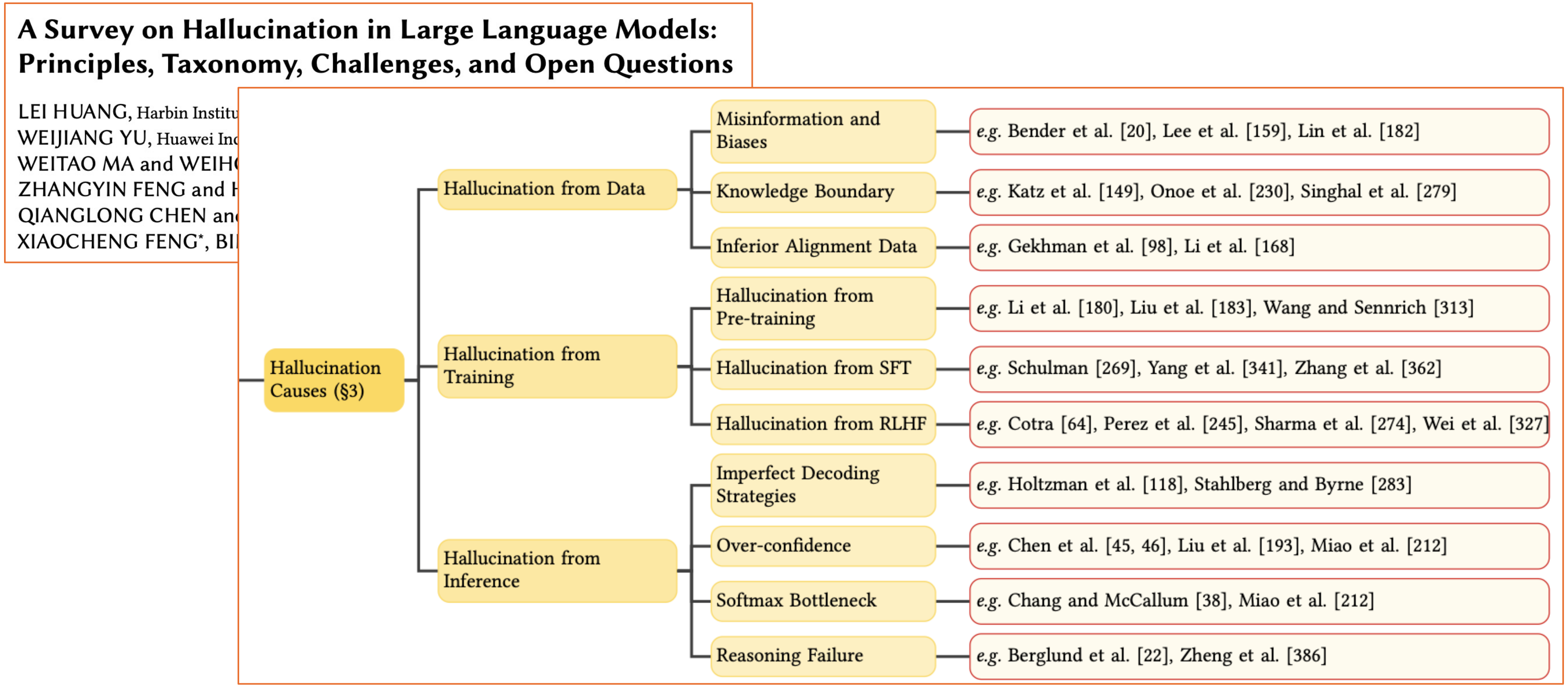
Hallucination



A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions

LEI HUANG, Harbin Institute of Technology, China
WEIJIANG YU, Huawei Inc., China
WEITAO MA and WEIHONG ZHONG, Harbin Institute of Technology, China
ZHANGYIN FENG and HAOTIAN WANG, Harbin Institute of Technology, China
QIANGLONG CHEN and WEIHUA PENG, Huawei Inc., China
XIAOCHENG FENG*, BING QIN, and TING LIU, Harbin Institute of Technology, China

Hallucination

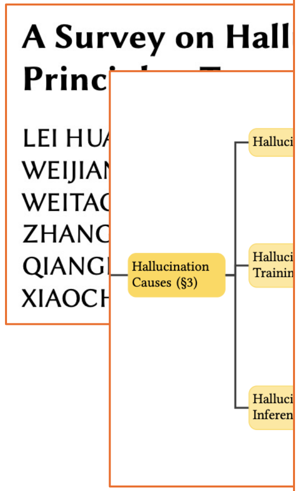


A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions

LEI HUANG, Harbin Institu
WEIJIANG YU, Huawei Inc
WEITAO MA and WEIH
ZHANGYIN FENG and H
QIANGLONG CHEN and
XIAOCHENG FENG*, BI

- Hallucination Causes (§3)
  - Hallucination from Data
    - Misinformation and Biases — *e.g.* Bender et al. [20], Lee et al. [159], Lin et al. [182]
    - Knowledge Boundary — *e.g.* Katz et al. [149], Onoe et al. [230], Singhal et al. [279]
    - Inferior Alignment Data — *e.g.* Gekhman et al. [98], Li et al. [168]
  - Hallucination from Training
    - Hallucination from Pre-training — *e.g.* Li et al. [180], Liu et al. [183], Wang and Sennrich [313]
    - Hallucination from SFT — *e.g.* Schulman [269], Yang et al. [341], Zhang et al. [362]
    - Hallucination from RLHF — *e.g.* Cotra [64], Perez et al. [245], Sharma et al. [274], Wei et al. [327]
  - Hallucination from Inference
    - Imperfect Decoding Strategies — *e.g.* Holtzman et al. [118], Stahlberg and Byrne [283]
    - Over-confidence — *e.g.* Chen et al. [45, 46], Liu et al. [193], Miao et al. [212]
    - Softmax Bottleneck — *e.g.* Chang and McCallum [38], Miao et al. [212]
    - Reasoning Failure — *e.g.* Berglund et al. [22], Zheng et al. [386]

Halluc



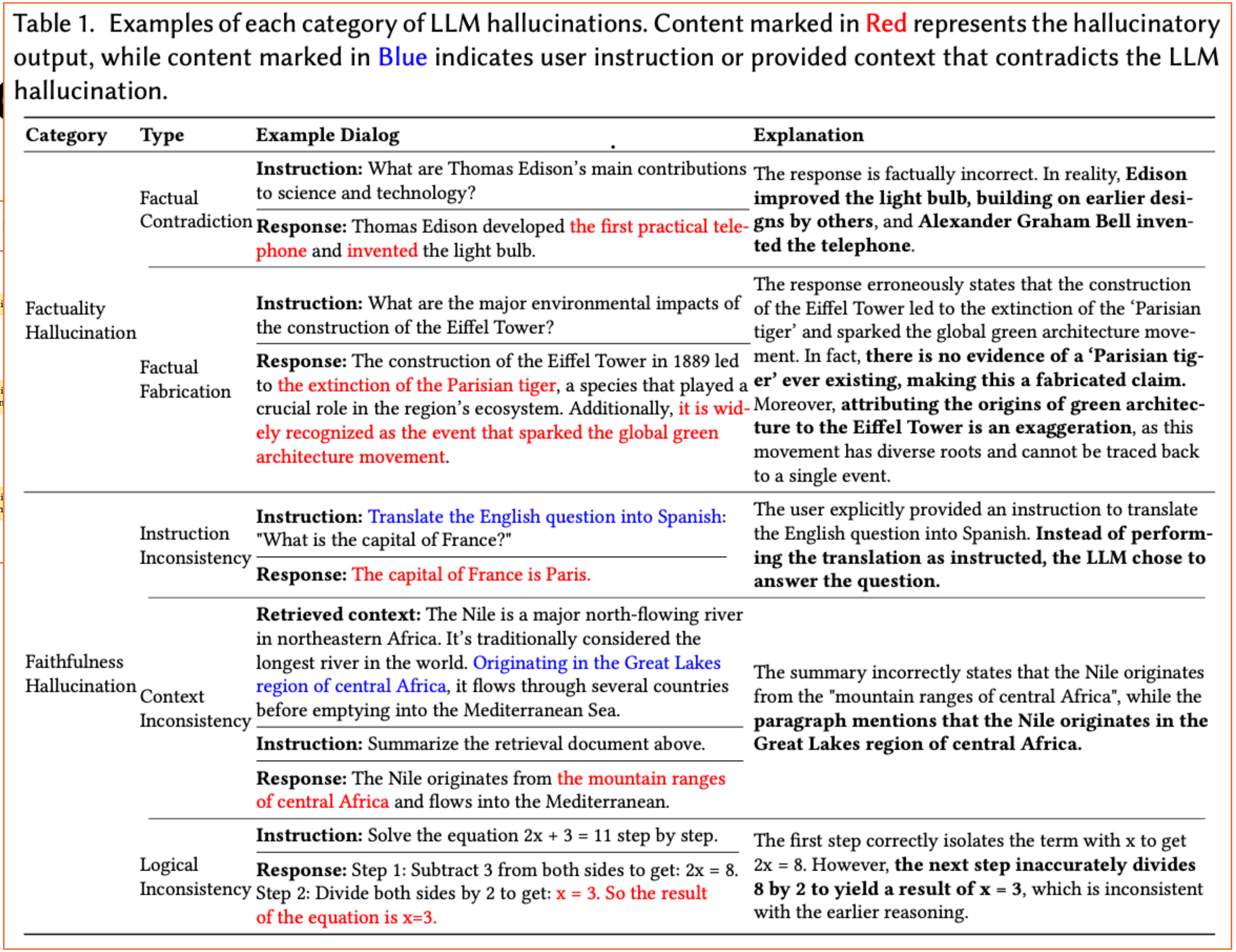
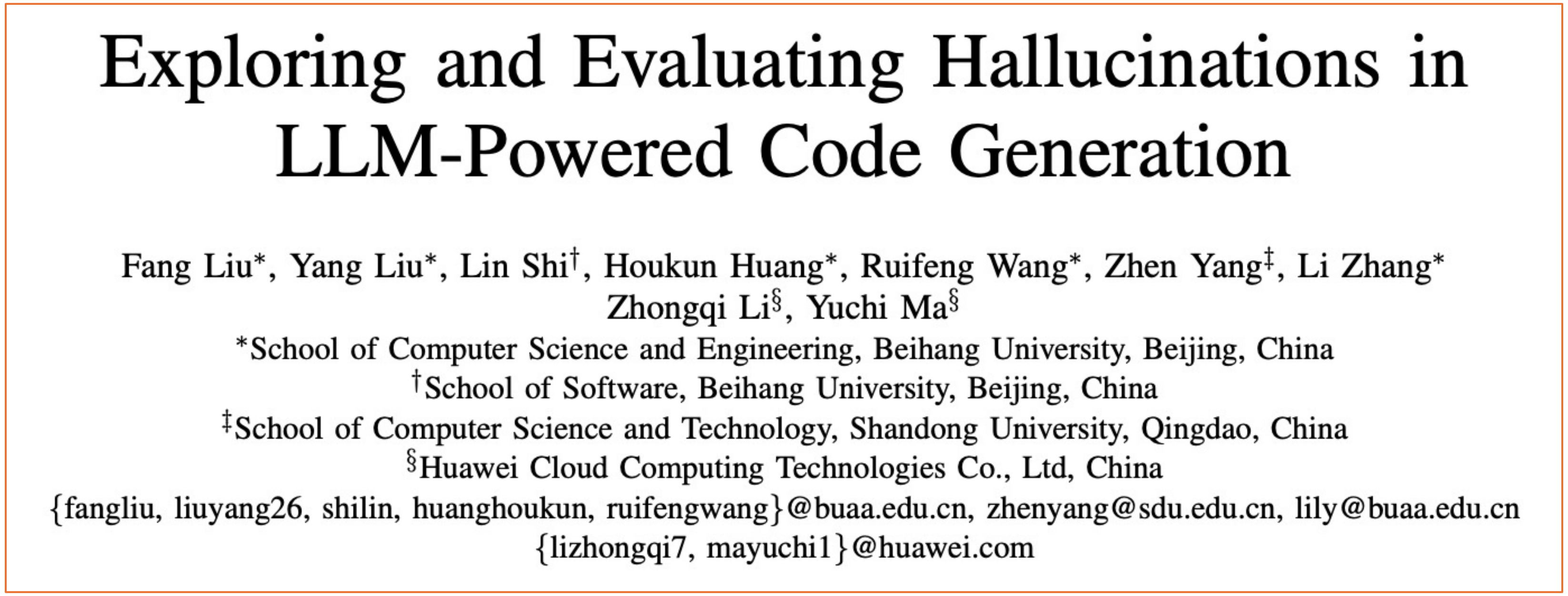


Table 1. Examples of each category of LLM hallucinations. Content marked in Red represents the hallucinatory output, while content marked in Blue indicates user instruction or provided context that contradicts the LLM hallucination.

Category	Type	Example Dialog	Explanation
Factuality Hallucination	Factual Contradiction	**Instruction:** What are Thomas Edison's main contributions to science and technology? **Response:** Thomas Edison developed the first practical telephone and invented the light bulb.	The response is factually incorrect. In reality, **Edison improved the light bulb, building on earlier designs by others**, and **Alexander Graham Bell invented the telephone**.
	Factual Fabrication	**Instruction:** What are the major environmental impacts of the construction of the Eiffel Tower? **Response:** The construction of the Eiffel Tower in 1889 led to the extinction of the Parisian tiger, a species that played a crucial role in the region's ecosystem. Additionally, it is widely recognized as the event that sparked the global green architecture movement.	The response erroneously states that the construction of the Eiffel Tower led to the extinction of the 'Parisian tiger' and sparked the global green architecture movement. In fact, **there is no evidence of a 'Parisian tiger' ever existing, making this a fabricated claim.** Moreover, **attributing the origins of green architecture to the Eiffel Tower is an exaggeration**, as this movement has diverse roots and cannot be traced back to a single event.
Faithfulness Hallucination	Instruction Inconsistency	**Instruction:** Translate the English question into Spanish: "What is the capital of France?" **Response:** The capital of France is Paris.	The user explicitly provided an instruction to translate the English question into Spanish. **Instead of performing the translation as instructed, the LLM chose to answer the question.**
	Context Inconsistency	**Retrieved context:** The Nile is a major north-flowing river in northeastern Africa. It's traditionally considered the longest river in the world. Originating in the Great Lakes region of central Africa, it flows through several countries before emptying into the Mediterranean Sea. **Instruction:** Summarize the retrieval document above. **Response:** The Nile originates from the mountain ranges of central Africa and flows into the Mediterranean.	The summary incorrectly states that the Nile originates from the "mountain ranges of central Africa", while the **paragraph mentions that the Nile originates in the Great Lakes region of central Africa.**
	Logical Inconsistency	**Instruction:** Solve the equation 2x + 3 = 11 step by step. **Response:** Step 1: Subtract 3 from both sides to get: 2x = 8. Step 2: Divide both sides by 2 to get: x = 3. So the result of the equation is x=3.	The first step correctly isolates the term with x to get 2x = 8. However, **the next step inaccurately divides 8 by 2 to yield a result of x = 3**, which is inconsistent with the earlier reasoning.

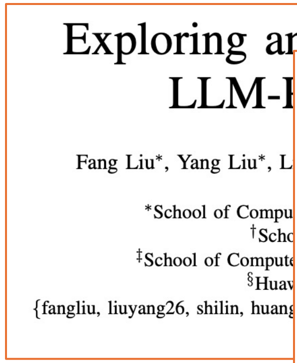
Hallucination



Exploring and Evaluating Hallucinations in LLM-Powered Code Generation

Fang Liu[*], Yang Liu[*], Lin Shi[†], Houkun Huang[*], Ruifeng Wang[*], Zhen Yang[‡], Li Zhang[*]
Zhongqi Li[§], Yuchi Ma[§]
[*]School of Computer Science and Engineering, Beihang University, Beijing, China
[†]School of Software, Beihang University, Beijing, China
[‡]School of Computer Science and Technology, Shandong University, Qingdao, China
[§]Huawei Cloud Computing Technologies Co., Ltd, China
{fangliu, liuyang26, shilin, huanghoukun, ruifengwang}@buaa.edu.cn, zhenyang@sdu.edu.cn, lily@buaa.edu.cn
{lizhongqi7, mayuchi1}@huawei.com

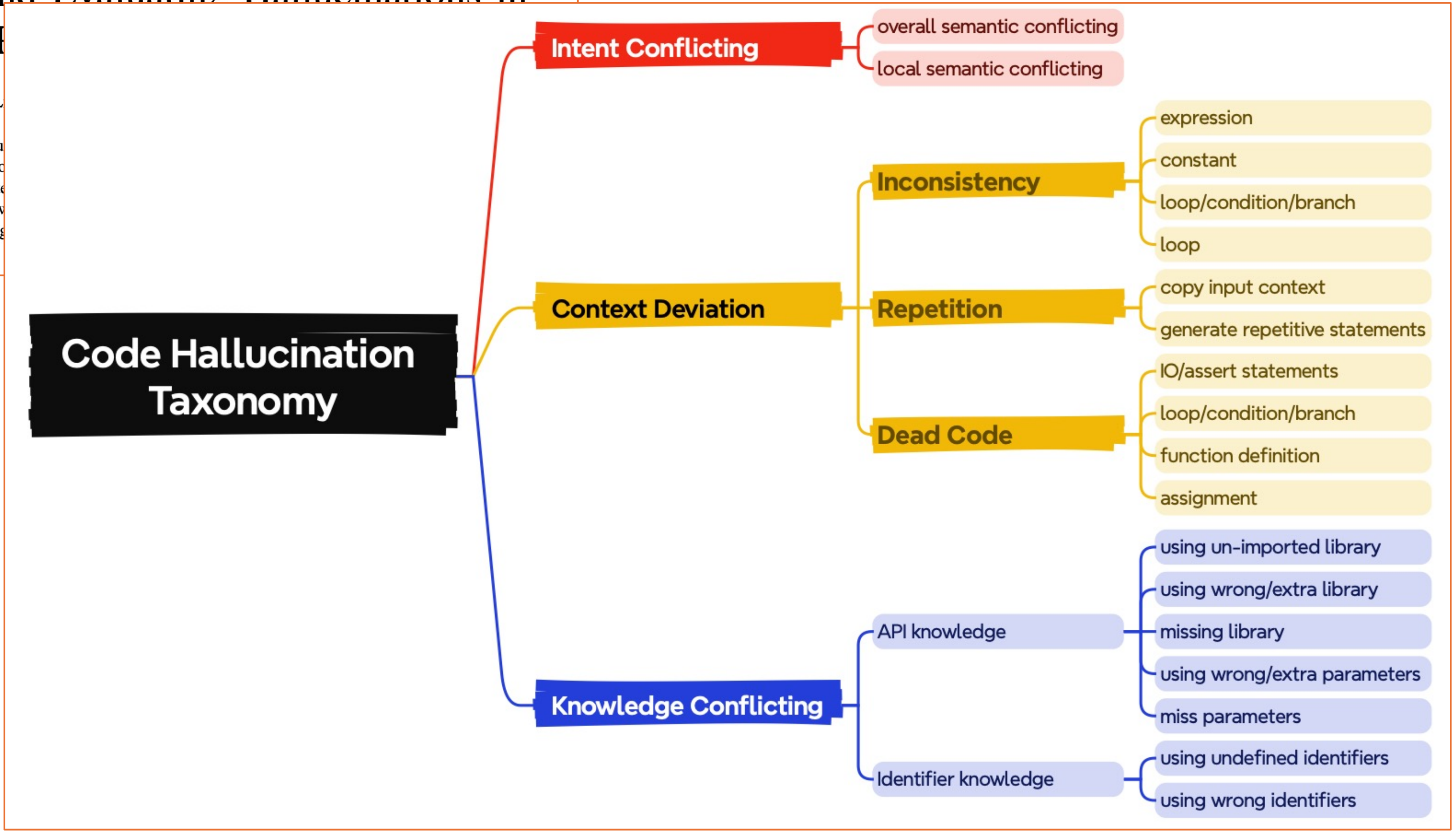
Hallucination



Exploring and Evaluating Hallucinations in
LLM-P

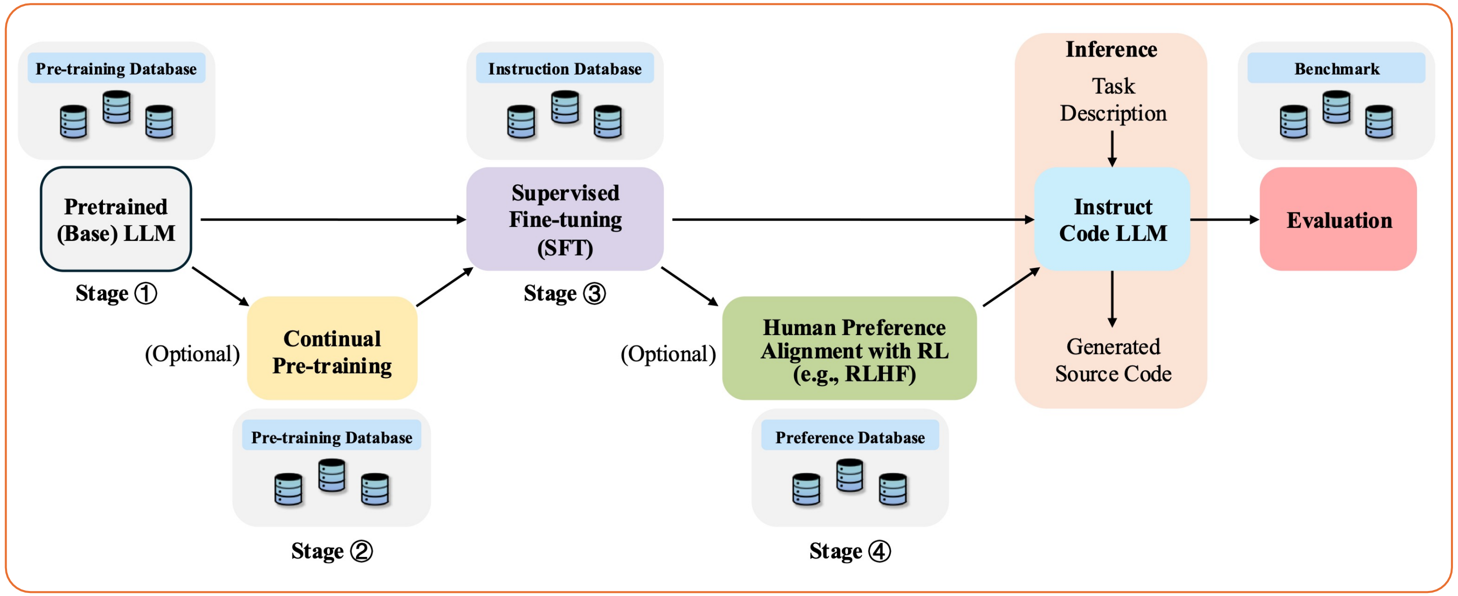
Fang Liu*, Yang Liu*, L

*School of Compu
†Scho
‡School of Compute
§Huaw
{fangliu, liuyang26, shilin, huang

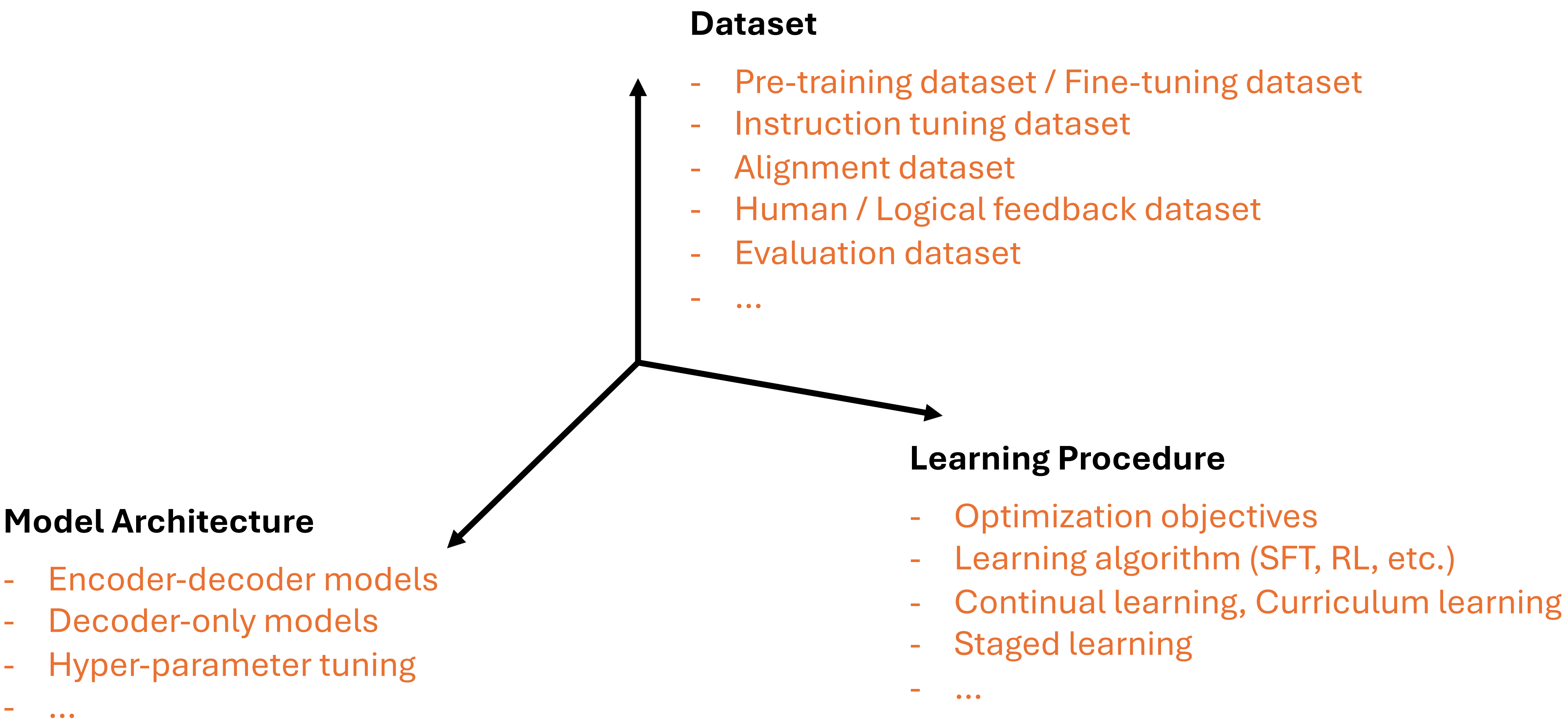


Today’s Agenda

- Pre-training stage
  - ~~Model architecture~~
  - ~~Pre-training dataset~~
  - ~~Learning objectives~~
  - ~~Evaluation dataset~~
- Special topics
  - ~~Post-training staging~~
  - ~~Scaling law~~
  - ~~Hallucination~~



How to obtain a “good enough” LLM



Dataset

- Pre-training dataset / Fine-tuning dataset
- Instruction tuning dataset
- Alignment dataset
- Human / Logical feedback dataset
- Evaluation dataset
- …

Learning Procedure

- Optimization objectives
- Learning algorithm (SFT, RL, etc.)
- Continual learning, Curriculum learning
- Staged learning
- …

Model Architecture

- Encoder-decoder models
- Decoder-only models
- Hyper-parameter tuning
- …

Logistics – Week 7

- Assignment 3
  - https://github.com/machine-programming/assignment-3
  - Releasing tomorrow; due two weeks from now (Oct 23)
- Oral presentation sign up sheet
  - Sent out during the weekend
  - Oral presentation starting on Week 9
- Forming groups for your final projects!
  - Sign up form will be sent out on Thursday
  - Form a group of 2-3 before Next Thursday (Oct 16)