

CS11-711 Advanced NLP

Domain-specific Modeling: Code and Math

Xiang Yue



Carnegie Mellon University

Language Technologies Institute

<https://phontron.com/class/anlp-fall2024>

Slides are partially adapted from 11-891 [Neural Code Generation](#) (by Daniel Fried and Sean Welleck)

What will cover in this class

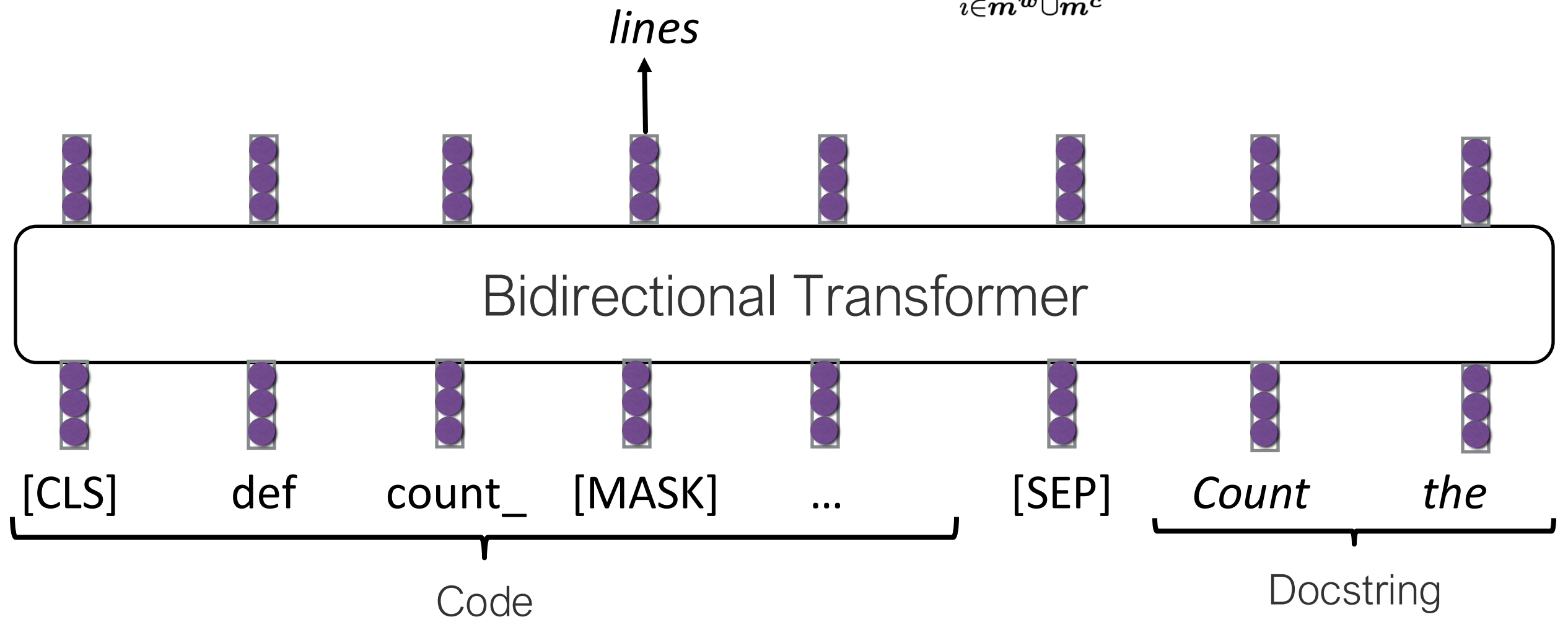
Domain-specific Modeling:

- Code Pre-training, Fine-tuning, Evaluation
- Math Pre-training, Fine-tuning, Evaluation

CodeBERT: Masked Language Modeling Objective

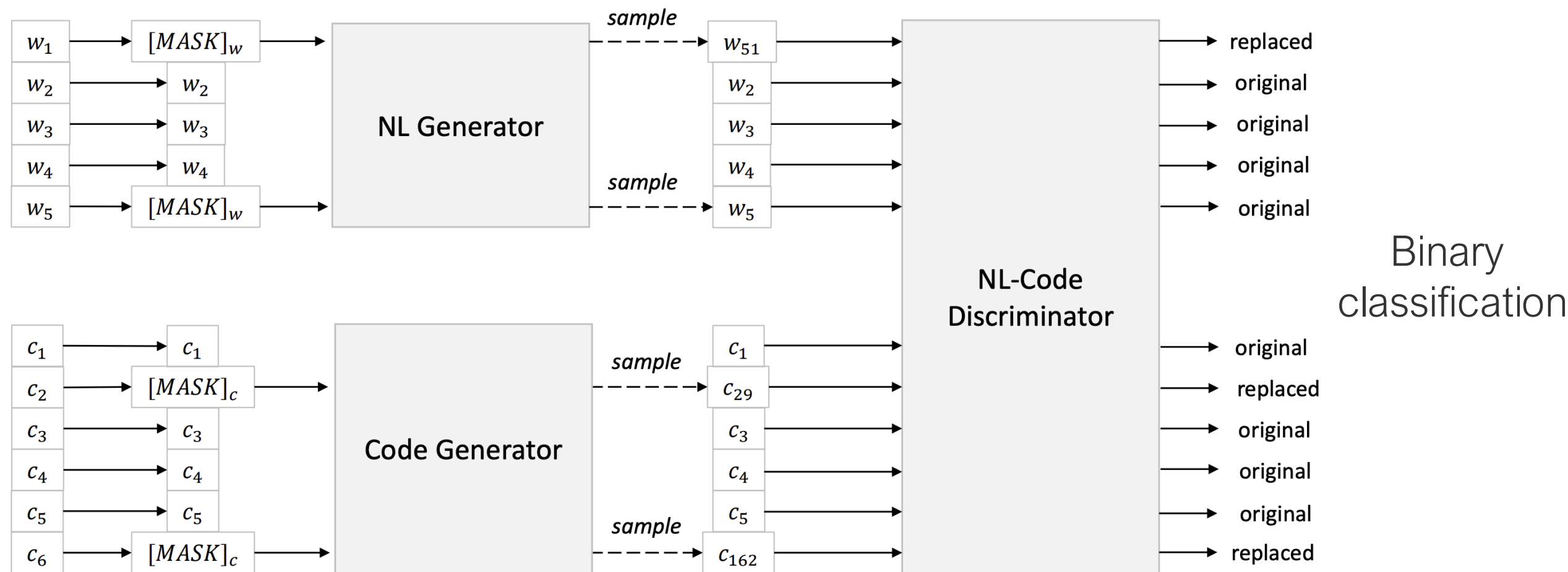
Mask 15% of the tokens, randomly, and try to predict these masked tokens.

$$\mathcal{L}_{\text{MLM}}(\theta) = \sum_{i \in m^w \cup m^c} -\log p^{D_1}(x_i | \mathbf{w}^{\text{masked}}, \mathbf{c}^{\text{masked}})$$



CodeBERT: Replaced Token Detection Objective

Rather than masked tokens, use tokens replaced by (weaker) LMs, and distinguish original tokens from replaced tokens.



CodeBERT: Pre-Training

125M parameter bidirectional encoder Transformer

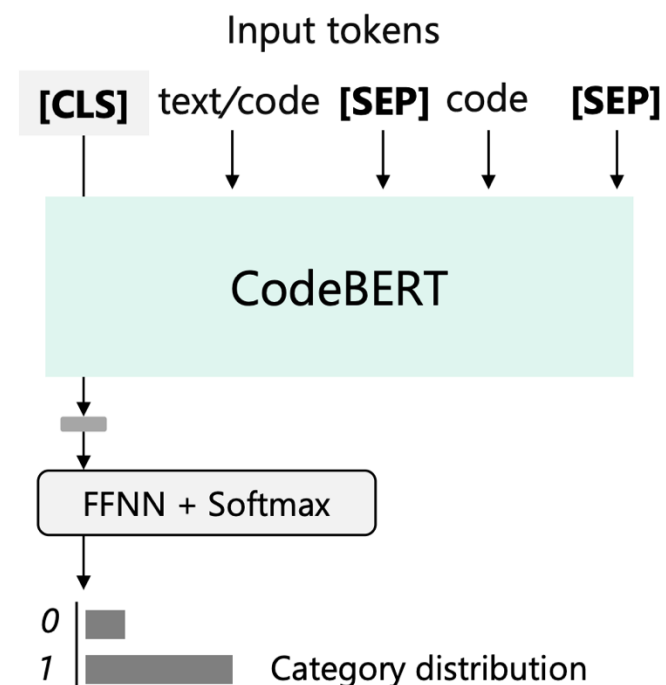
- Train on 2M documented functions (text & code) and 6M undocumented functions (code only) from GitHub

TRAINING DATA	<i>bimodal</i> DATA	<i>unimodal</i> CODES
GO	319,256	726,768
JAVA	500,754	1,569,889
JAVASCRIPT	143,252	1,857,835
PHP	662,907	977,821
PYTHON	458,219	1,156,085
RUBY	52,905	164,048
ALL	2,137,293	6,452,446

CodeBERT: Finetuning

Parts of the task network are initialized with CodeBERT parameters.

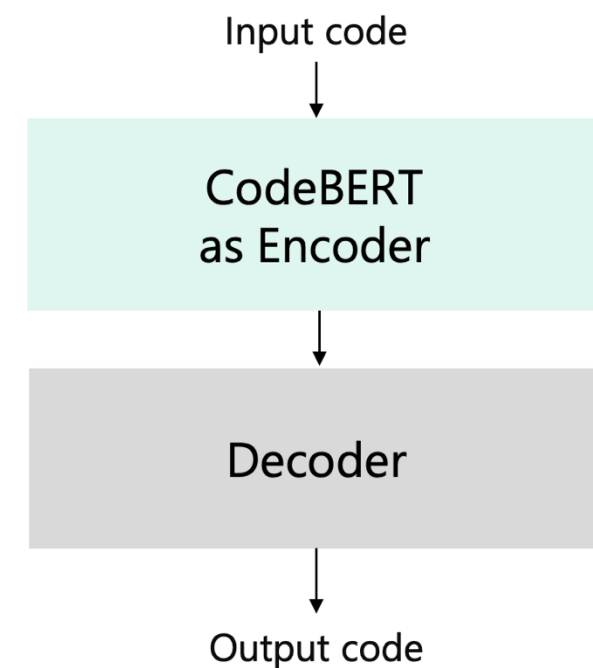
Classification Tasks



Supported tasks:

- code search
- code clone detection

Generation Tasks



Supported tasks:

- code repair
- code translation

CodeXGLUE Benchmark

Collection of tasks, largely with natural data mined from GitHub

Category	Task	Dataset Name	Language	Train/Dev/Test Size	Baselines	
Code-Code	Clone Detection	BigCloneBench [71]	Java	900K/416K/416K	CodeBERT	
		POJ-104 [52]	C/C++	32K/8K/12K		
	Defect Detection	Devign [99]	C	21K/2.7K/2.7K		
	Cloze Test	CT-all	Python,Java,PHP, JavaScript,Ruby,Go	-/-/176K		
		CT-max/min [18]	Python,Java,PHP, JavaScript,Ruby,Go	-/-/2.6K		
	Code Completion	PY150 [62]	Python	100K/5K/50K		CodeGPT
		Github Java Corpus[4]	Java	13K/7K/8K		
	Code Repair	Bugs2Fix [75]	Java	98K/12K/12K		Encoder-Decoder
Code Translation	CodeTrans	Java-C#	10K/0.5K/1K	Encoder-Decoder		
Text-Code	NL Code Search	CodeSearchNet [35], AdvTest	Python	251K/9.6K/19K	CodeBERT	
		CodeSearchNet [35], WebQueryTest	Python	251K/9.6K/1K		
	Text-to-Code Generation	CONCODE [38]	Java	100K/2K/2K	CodeGPT	
Code-Text	Code Summarization	CodeSearchNet [35]	Python,Java,PHP, JavaScript,Ruby,Go	908K/45K/53K	Encoder-Decoder	
Text-Text	Documentation Translation	Microsoft Docs	English-Latvian/Danish /Norwegian/Chinese	156K/4K/4K	Encoder-Decoder	

CodeBERT: Results

- Joint training on code and documentation > code alone
- Initializing with a text-only model (RoBERTa) helps

MODEL	RUBY	JAVASCRIPT	GO	PYTHON	JAVA	PHP	MA-AVG
ROBERTA	0.6245	0.6060	0.8204	0.8087	0.6659	0.6576	0.6972
PT W/ CODE ONLY (INIT=S)	0.5712	0.5557	0.7929	0.7855	0.6567	0.6172	0.6632
PT W/ CODE ONLY (INIT=R)	0.6612	0.6402	0.8191	0.8438	0.7213	0.6706	0.7260
CODEBERT (MLM, INIT=S)	0.5695	0.6029	0.8304	0.8261	0.7142	0.6556	0.6998
CODEBERT (MLM, INIT=R)	0.6898	0.6997	0.8383	0.8647	0.7476	0.6893	0.7549
CODEBERT (RTD, INIT=R)	0.6414	0.6512	0.8285	0.8263	0.7150	0.6774	0.7233
CODEBERT (MLM+RTD, INIT=R)	0.6926	0.7059	0.8400	0.8685	0.7484	0.7062	0.7603

Results for function/documentation matching (code retrieval)

CodeBERT: Results

- Joint training on code and documentation > code alone
- Initializing with a text-only model (RoBERTa) helps

MODEL	RUBY	JAVASCRIPT	GO	PYTHON	JAVA	PHP	OVERALL
SEQ2SEQ	9.64	10.21	13.98	15.93	15.09	21.08	14.32
TRANSFORMER	11.18	11.59	16.38	15.81	16.26	22.12	15.56
ROBERTA	11.17	11.90	17.72	18.14	16.47	24.02	16.57
PRE-TRAIN W/ CODE ONLY	11.91	13.99	17.78	18.58	17.50	24.34	17.35
CODEBERT (RTD)	11.42	13.27	17.53	18.29	17.35	24.10	17.00
CODEBERT (MLM)	11.57	14.41	17.78	18.77	17.38	24.85	17.46
CODEBERT (RTD+MLM)	12.16	14.90	18.07	19.06	17.65	25.16	17.83

Results for function-to-docstring generation

CodeBERT: Masked Prediction

Probing

masked NL token

"Transforms a vector `np.arange(-N, M, dx)` to `np.arange(min(|vec|), max(N,M), dx)`"

```
def vec_to_halfvec(vec):

    d = vec[1:] - vec[:-1]
    if ((d/d.mean()).std() > 1e-14) or (d.mean() < 0):
        raise ValueError('vec must be np.arange() in increasing order')
    dx = d.mean()
    lowest = np.abs(vec).min()
    highest = np.abs(vec).max()
    return np.arange(lowest, highest + 0.1*dx, dx).astype(vec.dtype)
```

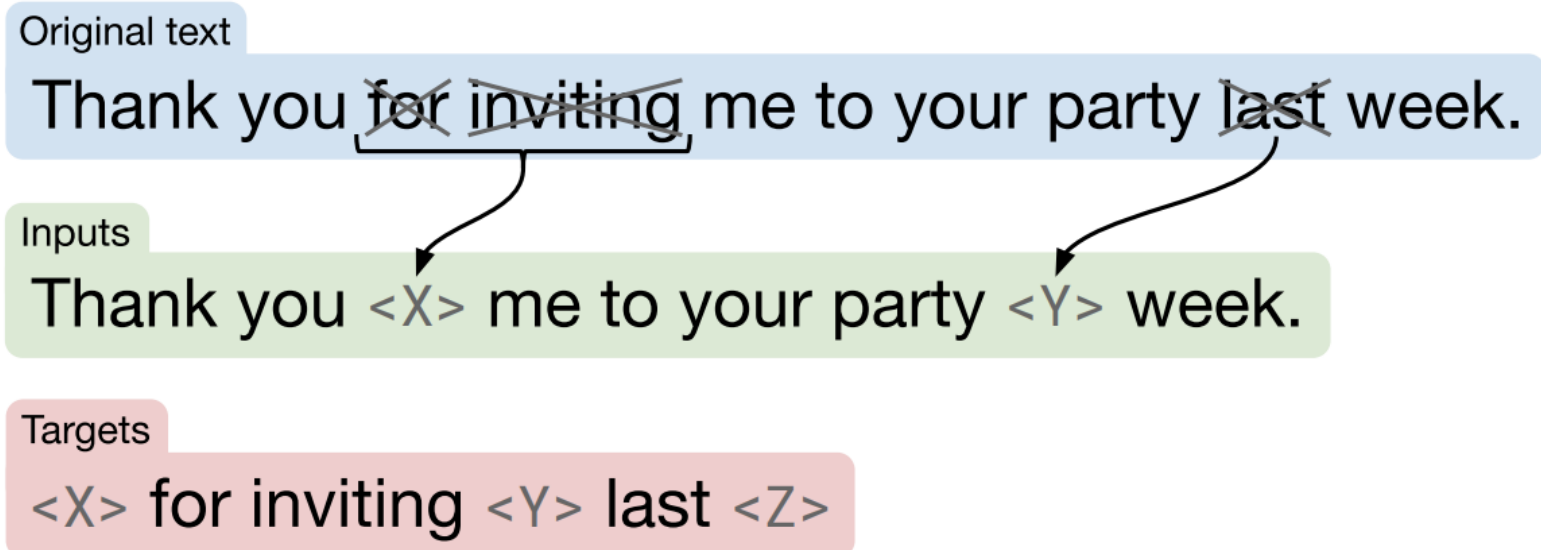
masked PL token

		<i>max</i>	<i>min</i>	<i>less</i>	<i>greater</i>
NL	Roberta	96.24%	3.73%	0.02%	0.01%
	CodeBERT (MLM)	39.38%	60.60%	0.02%	0.0003%
PL	Roberta	95.85%	4.15%	-	-
	CodeBERT (MLM)	0.001%	99.999%	-	-

Figure 3: Case study on python language. Masked tokens in NL (in blue) and PL (in yellow) are separately applied. Predicted probabilities of RoBERTa and CodeBERT are given.

T5: Text-to-Text Transfer Transformer

- ▶ **Objective:** similar denoising scheme to BART (they were released within a week of each other in fall 2019).
- ▶ Input: text with gaps. Output: a series of phrases to fill those gaps.
- ▶ Lower computational cost compared to BART: predicts fewer tokens.



CodeT5: Objectives

Wang et al. (2021)

Pre-train like T5 (seq-to-seq denoising/masked span prediction), but add identifier-specific objectives to learn code semantics.

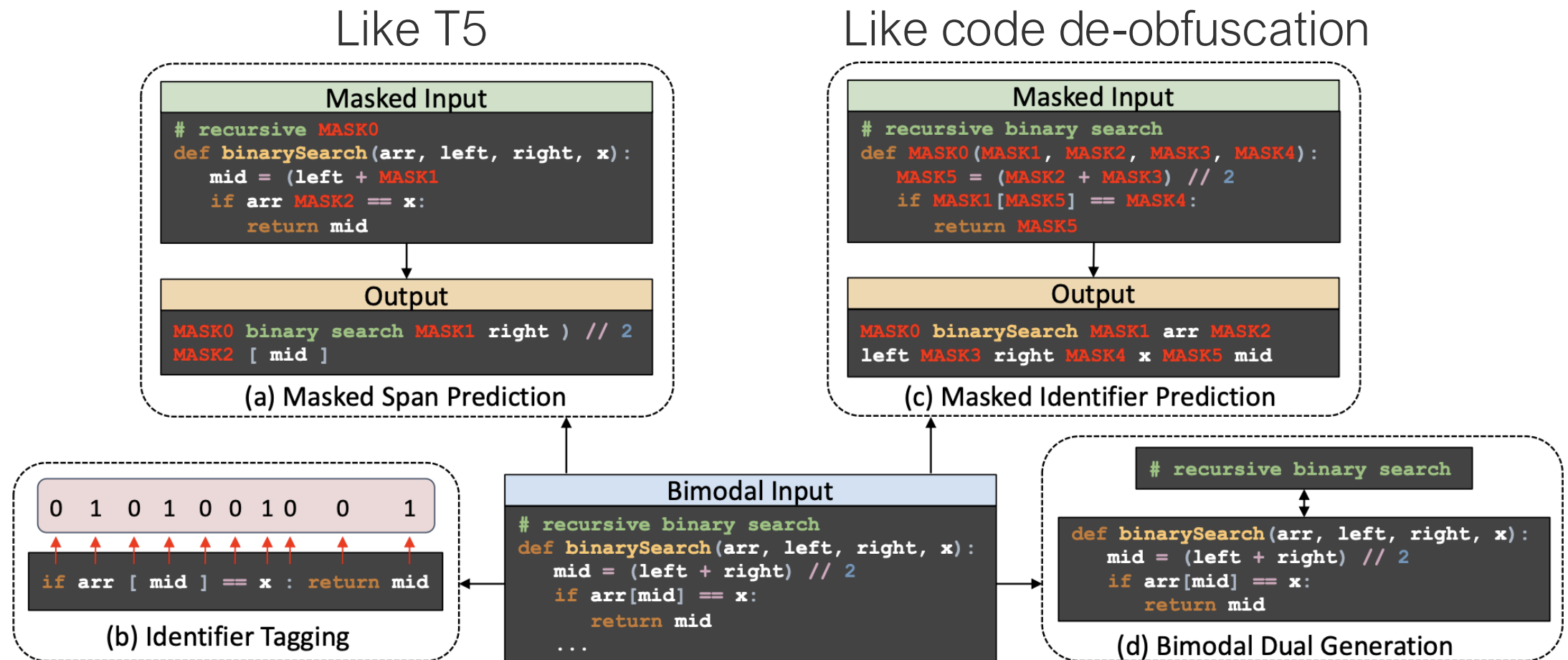


Figure 2: Pre-training tasks of CodeT5. We first alternately train span prediction, identifier prediction, and identifier tagging on both unimodal and bimodal data, and then leverage the bimodal data for dual generation training.

CodeT5: Training

Wang et al. (2021)

- Pre-train on CodeSearchNet (6 PLs) + BigQuery (C & C#); 8.4M instances
 - 60M and 220M parameter models, trained for 5 & 12 days on 16 GPUs.
 - Couldn't initialize with T5, because T5's tokenizer doesn't preserve code-specific symbols like { and }. Train own tokenizer
 - Then, optionally do **multi-task fine-tuning**: train on multiple seq-to-seq tasks from CodeXGLUE simultaneously (translation, refinement, summarization, ...).

CodeT5: Analysis

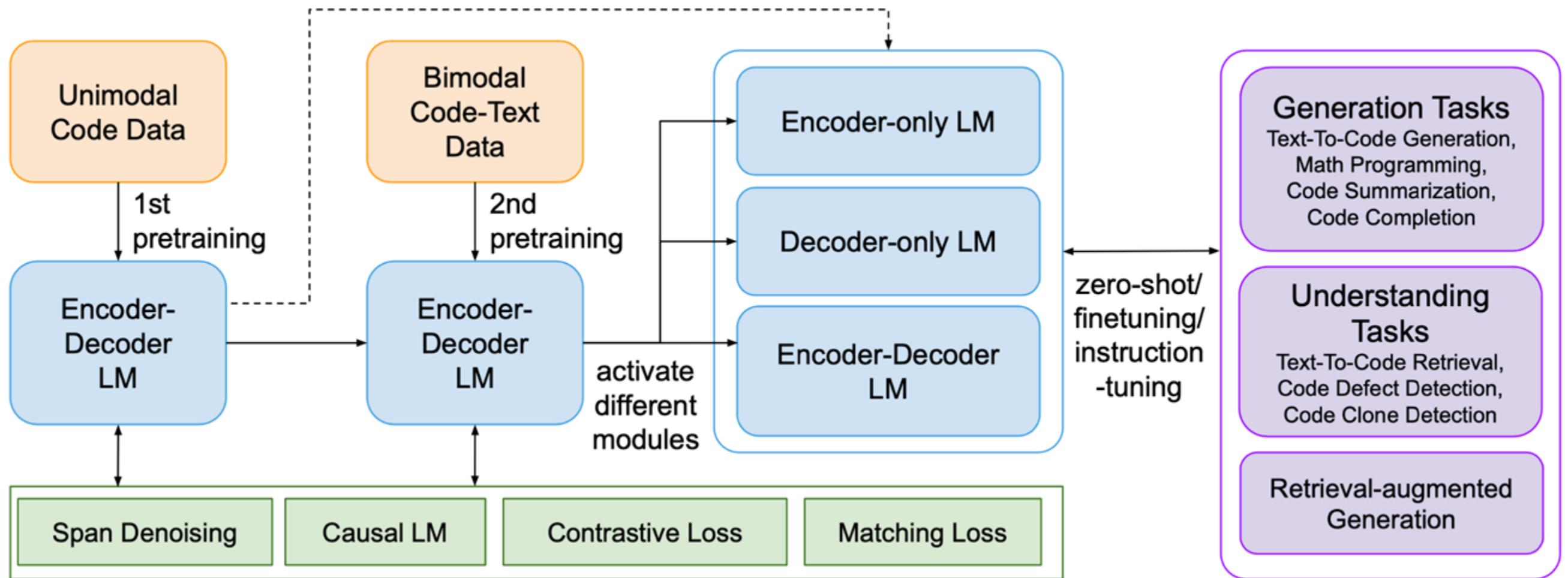
- All components of the objective help. MSP: masked span prediction. IT: identifier tagging. MIP: masked identifier prediction

Methods	Sum-PY (BLEU)	Code-Gen (CodeBLEU)	Refine Small (EM)	Defect (Acc)
CodeT5	20.04	41.39	19.06	63.40
-MSP	18.93	37.44	15.92	64.02
-IT	19.73	39.21	18.65	63.29
-MIP	19.81	38.25	18.32	62.92

CodeT5+

- Specializations of past approaches:
 - For **translation**: T5-like (seq-to-seq denoising) generally best
 - For **generating new content**: GPT-like (unidirectional decoder-only) generally best
 - For **doc-level embeddings**: BERT-like (MLM bidirectional encoder) generally best
- CodeT5+: use a seq-to-seq model but train it with a progression of objectives, and pre-trained initializations

CodeT5+: Overview



CodeT5+, <https://arxiv.org/abs/2305.07922>

Stage 1: Code-only pre-training

Goal: Train model to recover code contexts at different scales

Data: Code from GitHub

Tasks:

- Span Denoising (15% masked tokens)
- Causal LM
 - Partial programs
 - Complete programs

Stage 2: Code and text pre-training

Goal: Train model for cross-modal understanding and generation

Data: CodeSearchNet (Docstring & Code)

Tasks:

- Contrastive Learning (align feature space of code and text representation)
- Text-Code Matching (predict if semantics match)
- Text-Code Causal LM (text-to-code and code-to-text generation)

CodeT5+: Results

HumanEval code generation: slightly outperforms the CodeGen models it is initialized with

Model	Model size	pass@1	pass@10	pass@100
Closed-source models				
Codex	2.5B	21.4	35.4	59.5
Codex	12B	28.8	46.8	72.3
code-cushman-001	-	33.5	54.3	77.4
code-davinci-002	-	47.0	74.9	92.1
GPT-3.5	-	48.1	-	-
Open-source models				
CodeGen-mono	2B	23.7	36.6	57.0
CodeGen-mono	6B	26.1	42.3	65.8
CodeGen-mono	16B	29.3	49.9	75.0
CodeT5+	220M	12.0	20.7	31.6
CodeT5+	770M	15.5	27.2	42.7
CodeT5+	2B	24.2	38.2	57.8
CodeT5+	6B	28.0	47.2	69.8
CodeT5+	16B	30.9	51.6	76.7

CodeT5+: Results

Code retrieval: outperforms CodeT5 and CodeBERT

Table 6: Text-to-Code Retrieval results (MRR) on CodeXGLUE: CodeT5+ achieves consistent performance gains over the original CodeT5 models across all 3 retrieval benchmarks in 7 programming languages. Overall, our models demonstrate remarkable performance, outperforming many strong encoder-based models pretrained with contrastive loss such as SYNCOBERT and UniXcoder.

Model	CodeSearchNet							CosQA	AdvTest
	Ruby	JS	Go	Python	Java	PHP	Overall		
CodeBERT 125M	67.9	62.0	88.2	67.2	67.6	62.8	69.3	65.7	27.2
GraphCodeBERT 125M	70.3	64.4	89.7	69.2	69.1	64.9	71.3	68.4	35.2
SYNCOBERT 125M	72.2	67.7	91.3	72.4	72.3	67.8	74.0	-	38.3
UniXcoder 125M	74.0	68.4	91.5	72.0	72.6	67.6	74.4	70.1	41.3
CodeGen-multi 350M	66.0	62.2	90.0	68.6	70.1	63.9	70.1	64.8	34.8
PLBART 140M	67.5	61.6	88.7	66.3	66.3	61.1	68.6	65.0	34.7
CodeT5 220M	71.9	65.5	88.8	69.8	68.6	64.5	71.5	67.8	39.3
CodeT5+ 220M	77.7	70.8	92.4	75.6	76.1	69.8	77.1	72.7	43.3
CodeT5+ 770M	78.0	71.3	92.7	75.8	76.2	70.1	77.4	74.0	44.7

Filling-in-the-Middle

LLM Training Objectives

```
def minimize_in_graph(build_loss_fn, num_steps=200, optimizer=None):
```

```
    """ Minimize a loss function using gradient.
```

```
    Args:
```

```
        build_loss_fn: a function that returns a loss tensor for a mini-batch of examples.
```

```
        num_steps: number of gradient descent steps to perform.
```

```
        optimizer: an optimizer to use when minimizing the loss function. If None, will use Adam
```

```
    """
```

```
    optimizer = tf.compat.v1.train.AdamOptimizer(0.1) if optimizer is None else optimizer
```

```
    minimize_op = tf.compat.v1.train_while_loop(
```

```
        cond=lambda step: step < num_steps,
```

```
        body=train_loop_body,
```

```
        loop_vars=[tf.constant(0)], return_same_structure=True)[0]
```

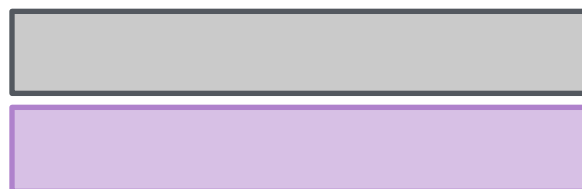
```
    return minimize_op
```

Prefix

Target

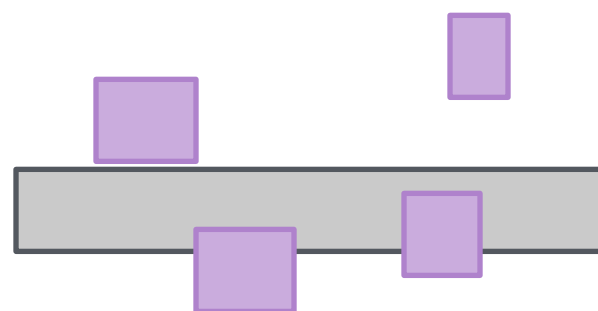
Suffix

“Causal” (L-to-R)



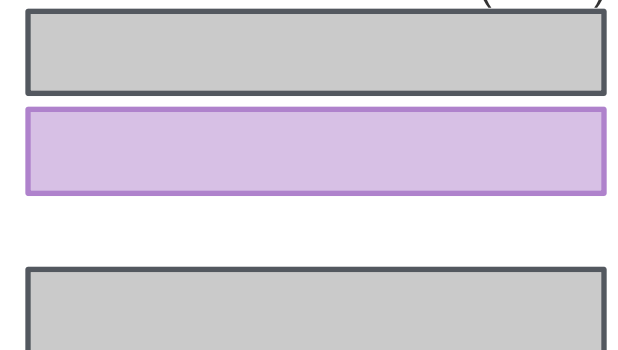
[e.g. GPT-*, Codex]

Masked Infilling



[e.g. BERT, CodeBERT]

“Causal Masking” /
Fill-in-the-Middle (FIM)



[Donahue+ 2020, Aghajanyan+
2022, ours, Bavarian+ 2022]

Causal Masking / FIM Objective

Training

Original Document

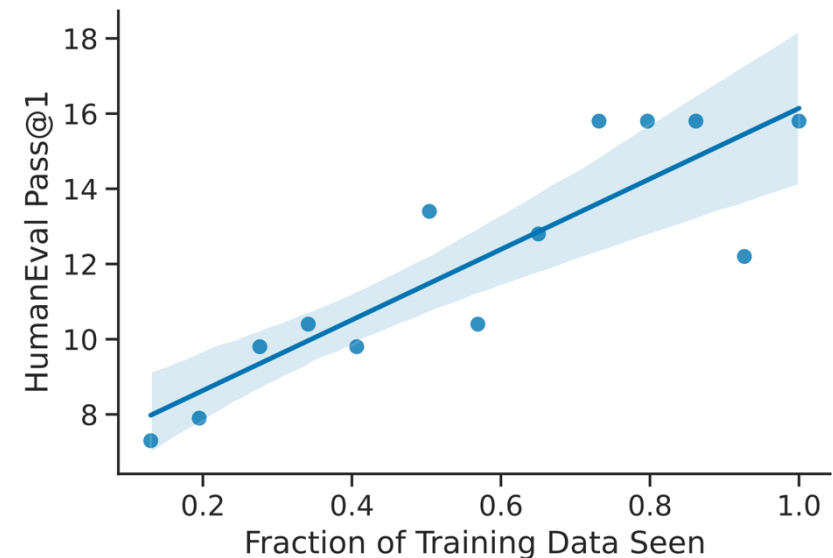
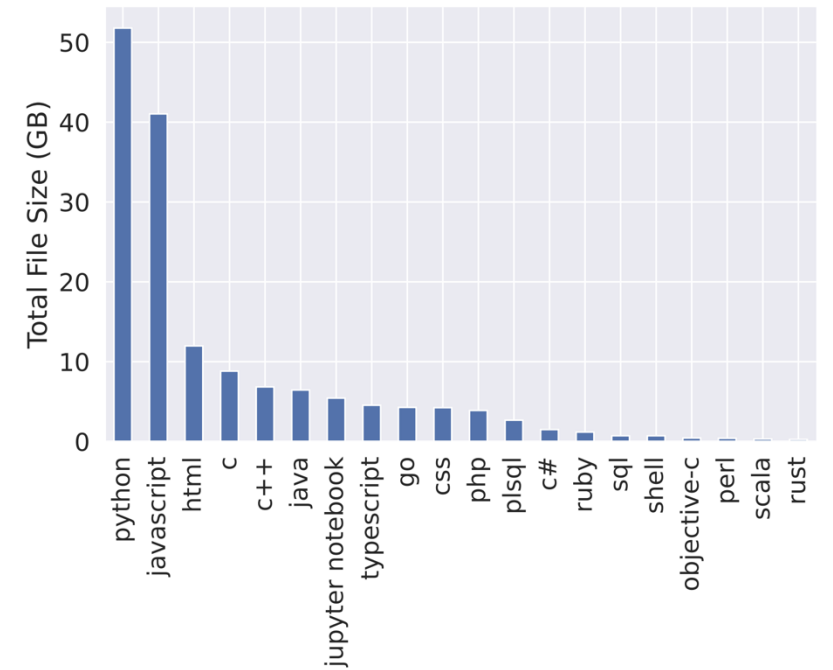
```
def count_words(filename: str) -> Dict[str, int]:  
    """Count the number of occurrences of each word in the file."""  
    with open(filename, 'r') as f:  
        word_counts = {}  
        for line in f:  
            for word in line.split():  
                if word in word_counts:  
                    word_counts[word] += 1  
                else:  
                    word_counts[word] = 1  
    return word_counts
```

Masked Document

```
def count_words(filename: str) -> Dict[str, int]:  
    """Count the number of occurrences of each word in the file."""  
    with open(filename, 'r') as f:  
        <MASK:0> in word_counts:  
  
        word_counts[word] = 1  
    return word_counts  
<MASK:0> word_counts = {}  
    for line in f:  
        for word in line.split():  
            if word <EOM>
```

InCoder: Model Training

- Training Data
 - 600K permissively-licensed repositories from GitHub & GitLab. ~150GB total
 - StackOverflow: questions, answers, comments. ~50GB
- Models
 - Unidirectional, decoder-only Transformer
 - 1B model: ~1 week on 128 V100s
 - 6B model: ~3 weeks on 240 V100s



Other Infilling Code Models

Efficient Training of Language Models to Fill in the Middle

Mohammad Bavarian*

Heewoo Jun*

Nikolas Tezak

John Schulman

Christine McLeavey

Jerry Tworek

Mark Chen

OpenAI

SANTACODER: DON'T REACH FOR THE STARS!

Loubna Ben Allal*
Hugging Face

Raymond Li*
ServiceNow Research

Denis Kocetkov*
ServiceNow Research

StarCoder: may the source be with you!

Raymond Li² Loubna Ben Allal¹ Yangtian Zi⁴ Niklas Muennighoff¹ Denis Kocetkov²
Chenghao Mou⁵ Marc Marone⁸ Christopher Akiki^{9,10} Jia Li⁵ Jenny Chim¹¹ Qian Liu¹³

Code Llama: Open Foundation Models for Code

Baptiste Rozière[†], Jonas Gehring[†], Fabian Gloeckle^{†,*}, Sten Sootla[†], Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, 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[†]

CODEGEN2: LESSONS FOR TRAINING LLMs ON PROGRAMMING AND NATURAL LANGUAGES

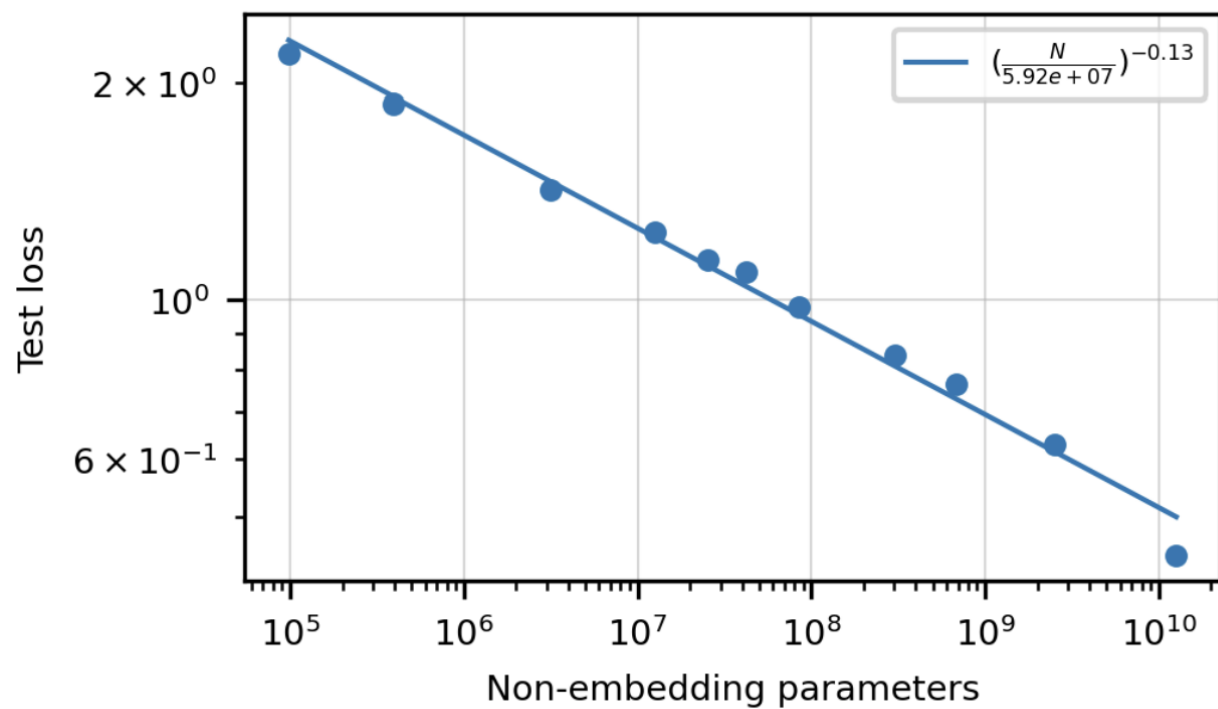
Erik Nijkamp*, Hiroaki Hayashi*, Caiming Xiong, Silvio Savarese, Yingbo Zhou

Codex

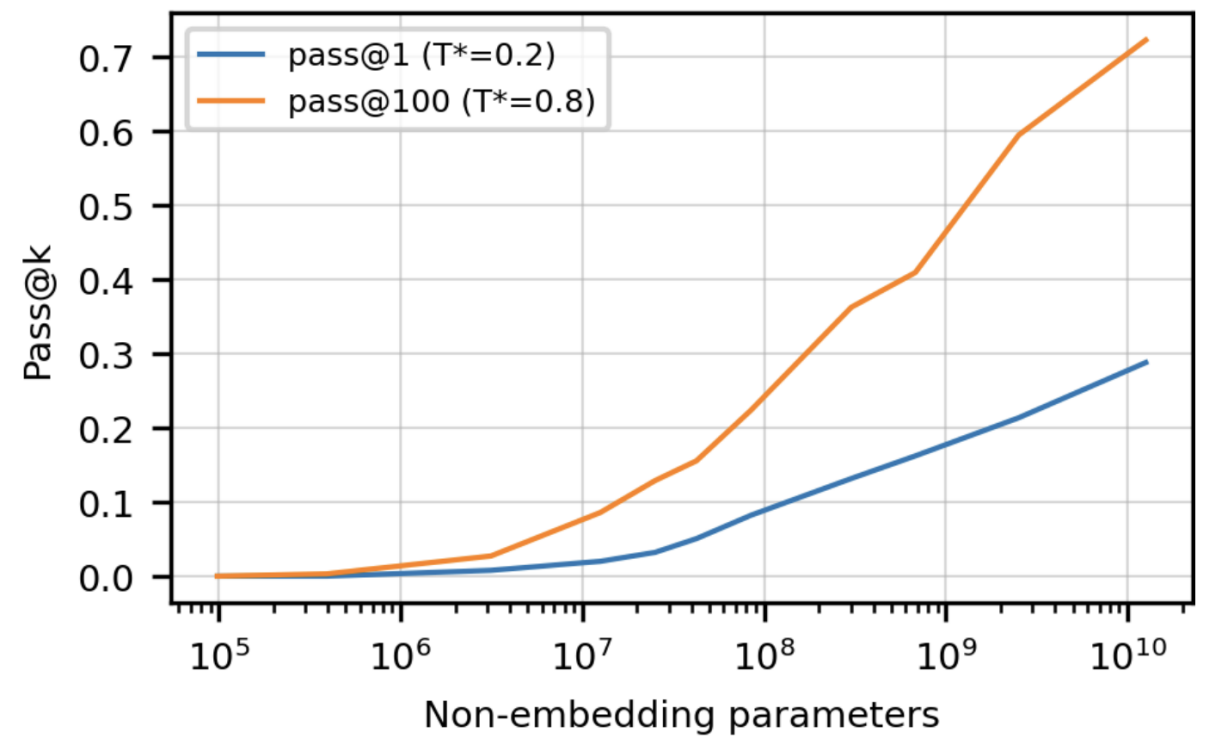
- Typically trained on lots of code from GitHub, often mixed with text
- Codex (Chen et al. 2021): OpenAI continues to train GPT-3 12B on 160GB of Python data from GitHub
- All GPT 3.5 models are trained on mixtures of code and text.
<https://platform.openai.com/docs/model-index-for-researchers>
- Many open-source models since then follow this recipe (PolyCoder, CodeGen, StarCoder)

Codex: Scaling Laws

Codex Loss Scaling



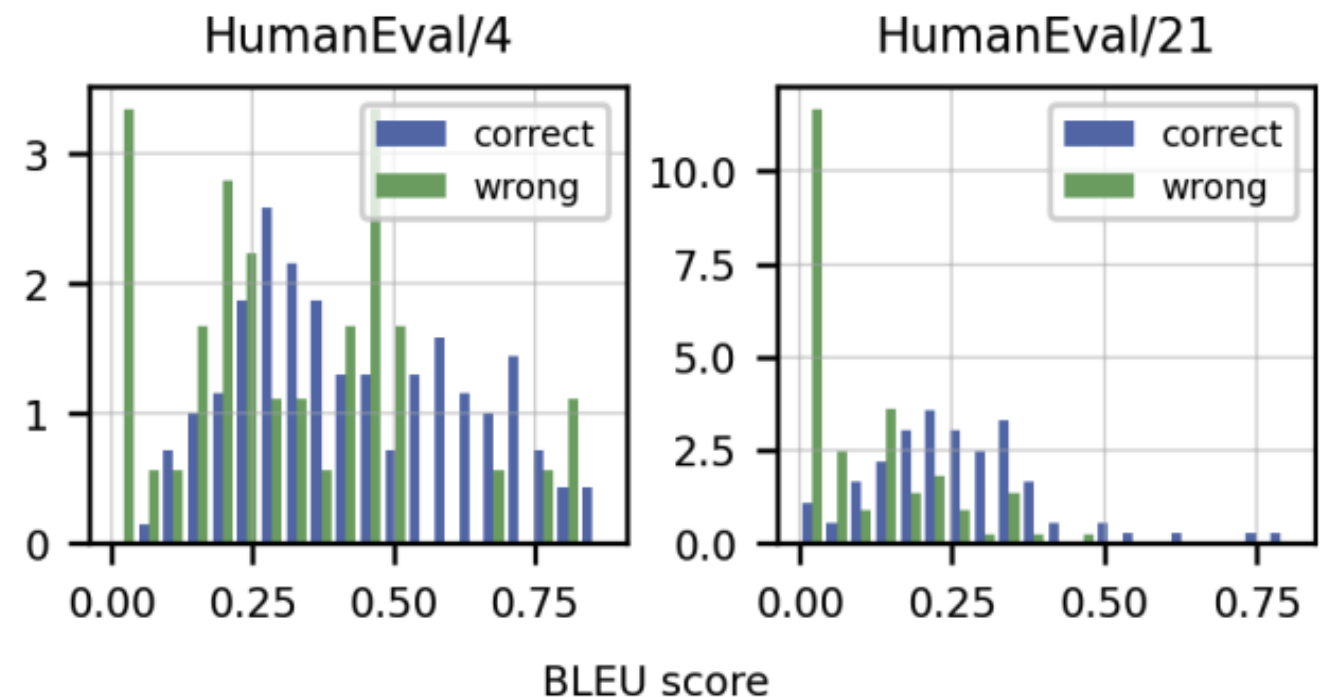
Pass Rate vs Model Size



Codex: “HumanEval” Benchmark

- Evaluation: test case execution
- 164 hand-written examples
- Why human-written?
 - “It is important for these tasks to be hand-written, since our models are trained on a large fraction of GitHub, which already contains solutions to problems from a variety of sources.”
- Optimizing BLEU \neq Improving Functional Correctness

```
def solution(lst):  
    """Given a non-empty list of integers, return the sum of all of the odd elements  
    that are in even positions.  
  
    Examples  
    solution([5, 8, 7, 1]) ==>12  
    solution([3, 3, 3, 3, 3]) ==>9  
    solution([30, 13, 24, 321]) ==>0  
    """  
    return sum(lst[i] for i in range(0, len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```



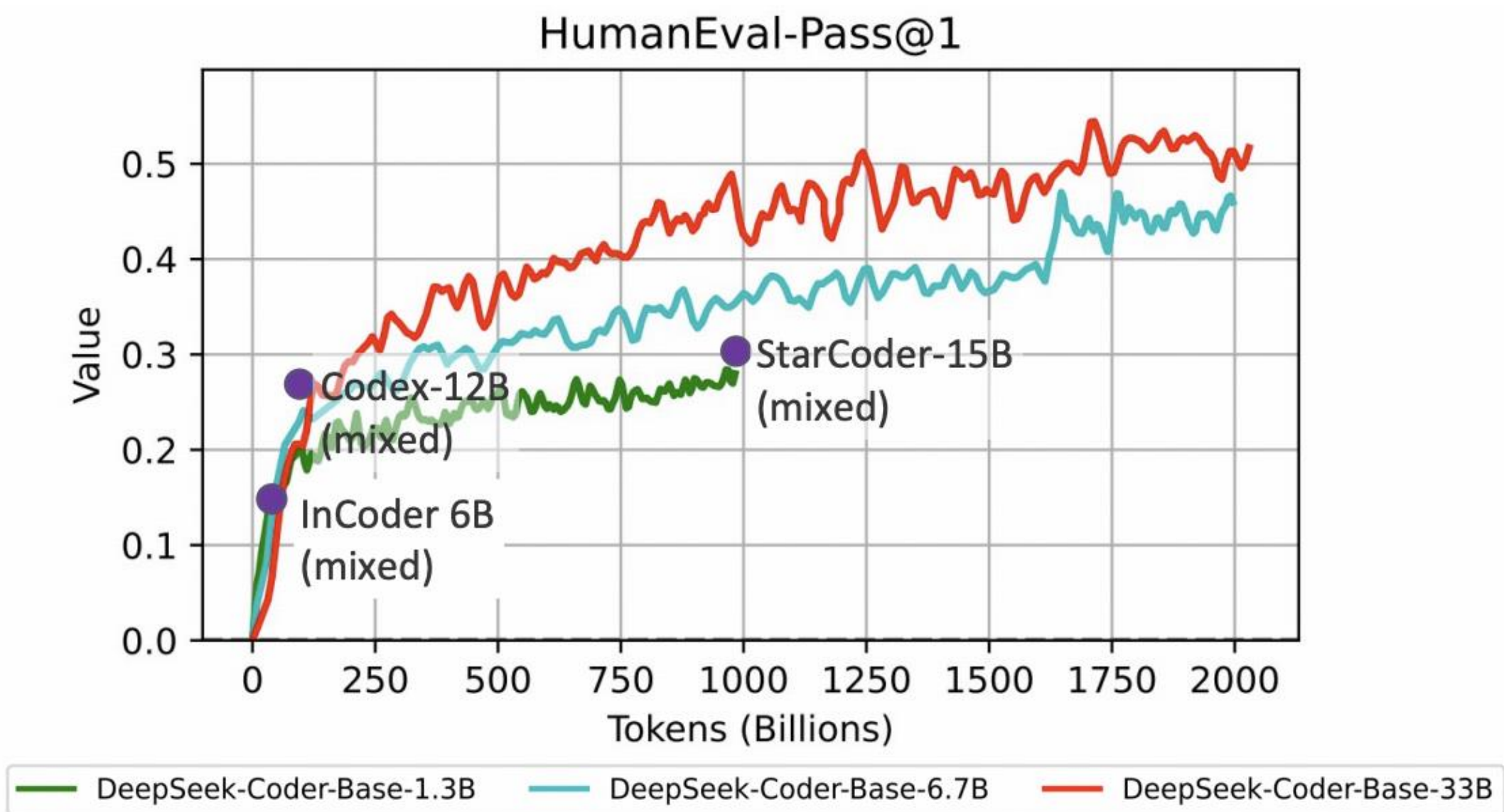
DeepSeek Coder

- ▶ 1.3B, 6.7B, and 33B parameter models
- ▶ Trained from scratch on 2 Trillion tokens of code from 87 languages
- ▶ FIM loss, and 16K context length

DeepSeek Coder: Repo-Level Context

- Parse file dependencies and arrange repo files in the context window using a topological ordering.
- Theoretically can handle 64K tokens, but “empirical observations suggest that the model delivers its most reliable outputs within a 16K token range”

Deepseek Coder: High-quality data matters



MBPP: Mostly Basic Python Programs

- Similar to HumanEval, but a bit easier
- 974 short Python problems, written by crowdworkers
 - 58% mathematical, 43% list processing, 19% string processing

HumanEval Looks Like Toy Examples?

- HumanEval Examples

```
def incr_list(l: list):  
    """Return list with elements incremented by 1.  
    >>> incr_list([1, 2, 3])  
    [2, 3, 4]  
    >>> incr_list([5, 3, 5, 2, 3, 3, 9, 0, 123])  
    [6, 4, 6, 3, 4, 4, 10, 1, 124]  
    """  
    return [i + 1 for i in l]
```

```
def solution(lst):  
    """Given a non-empty list of integers, return the sum of all of the odd elements  
    that are in even positions.  
  
    Examples  
    solution([5, 8, 7, 1]) ==>12  
    solution([3, 3, 3, 3, 3]) ==>9  
    solution([30, 13, 24, 321]) ==>0  
    """  
    return sum(lst[i] for i in range(0, len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

- Real-World Development Code

Asking the user for input until they give a valid response

Asked 9 years, 6 months ago Modified 1 year, 5 months ago Viewed 1.0m times

I am writing a program that accepts user input.

```
750 #note: Python 2.7 users should use `raw_input`, the equivalent of 3.X's `input`  
age = int(input("Please enter your age: "))  
if age >= 18:  
    print("You are able to vote in the United States!")  
else:  
    print("You are not able to vote in the United States.")
```

transformers Public

Watch 1.1k Fork 22.9k Starred 114k

main 262 branches 139 tags

Go to file Add file Code

About

Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX.

python nlp machine-learning natural-language-processing deep-learning tensorflow pytorch transformer speech-recognition seq2seq flax pretrained-models language-models nlp-library language-model hacktoberfest bart jax pytorch-transformers model-hub

SWE-Bench: Solving GitHub Issues

Model Input

▼ Instructions • 1 line
You will be provided with a partial code base and an issue statement explaining a problem to resolve.

▼ Issue • 67 lines
napoleon_use_param should also affect "other parameters" section Subject: napoleon_use_param should also affect "other parameters" section
Problem
Currently, napoleon always renders the Other parameters section as if napoleon_use_param was False, see source

```
def _parse_other_parameters_section(self, section: str) -> List[str]:  
    # type: (unicode) -> List[unicode]  
    return self._format_fields(_('Other Parameters'), self._consume_fields())  
  
def _parse_parameters_section(self, section):  
    # type: (unicode) -> List[unicode]  
    fields = self._consume_fields()  
    if self._config.napoleon_use_param: ...
```

▼ Code • 1431 lines
▶ **README.rst** • 132 lines
▶ **sphinx/ext/napoleon/docstring.py** • 1295 lines
▶ **Additional Instructions** • 57 lines

Gold Patch

```
sphinx/ext/napoleon/docstring.py  
    def _parse_other_parameters_section(self, section: str) -> List[str]:  
-     return self._format_fields(_('Other Parameters'), self._consume_fields())  
+     if self._config.napoleon_use_param:  
+         # Allow to declare multiple parameters at once (ex: x, y: int)  
+         fields = self._consume_fields(multiple=True)  
+         return self._format_docutils_params(fields)  
+     else:  
+         fields = self._consume_fields()  
+         return self._format_fields(_('Other Parameters'), fields)
```

Generated Patch

```
sphinx/ext/napoleon/docstring.py  
    def _parse_other_parameters_section(self, section: str) -> List[str]:  
-     return self._format_fields(_('Other Parameters'), self._consume_fields())  
+     return self._format_docutils_params(self._consume_fields())
```

Generated Patch Test Results

```
PASSED NumpyDocstringTest (test_yield_types)  
PASSED TestNumpyDocstring (test_escape_args_and_kwargs 1)  
PASSED TestNumpyDocstring (test_escape_args_and_kwargs 2)  
PASSED TestNumpyDocstring (test_escape_args_and_kwargs 3)  
PASSED TestNumpyDocstring (test_pep526_annotations)  
FAILED NumpyDocstringTest (test_parameters_with_class_reference)  
FAILED TestNumpyDocstring (test_token_type_invalid)  
==== 2 failed, 45 passed, 8 warnings in 5.16s ====
```

Figure 6: We show an example of an formatted task instance, a model prediction, and the testing framework logs. Results and inputs are stylized for readability. In the gold and generated patch file, red-highlighted lines represent deletions and green-highlighted lines represent additions.

SWE-Bench Leaderboard

Leaderboard

Lite	Verified	Full			
Model	% Resolved	Date	Logs	Trajs	Site
🏆 Gru(2024-08-24)	45.20	2024-08-24	🔗	🔗	🔗
🏆 Honeycomb	40.60	2024-08-20	🔗	🔗	🔗
🏆 Amazon Q Developer Agent (v20240719-dev)	38.80	2024-07-21	🔗	🔗	🔗
AutoCodeRover (v20240620) + GPT 4o (2024-05-13)	38.40	2024-06-28	🔗	-	🔗
Factory Code Droid	37.00	2024-06-17	🔗	-	🔗
🏆 ✅ SWE-agent + Claude 3.5 Sonnet	33.60	2024-06-20	🔗	🔗	-
🏆 ✅ AppMap Navie + GPT 4o (2024-05-13)	26.20	2024-06-15	🔗	-	🔗
Amazon Q Developer Agent (v20240430-dev)	25.60	2024-05-09	🔗	-	🔗
EPAM AI/Run Developer Agent + GPT4o	24.00	2024-08-20	🔗	🔗	🔗
🏆 ✅ SWE-agent + GPT 4o (2024-05-13)	23.20	2024-07-28	🔗	🔗	🔗
🏆 ✅ SWE-agent + GPT 4 (1106)	22.40	2024-04-02	🔗	🔗	🔗
🏆 ✅ SWE-agent + Claude 3 Opus	18.20	2024-04-02	🔗	🔗	-
🏆 ✅ RAG + Claude 3 Opus	7.00	2024-04-02	🔗	-	🔗
🏆 ✅ RAG + Claude 2	4.40	2023-10-10	🔗	-	-
🏆 ✅ RAG + GPT 4 (1106)	2.80	2024-04-02	🔗	-	-
🏆 ✅ RAG + SWE-Llama 7B	1.40	2023-10-10	🔗	-	-
🏆 ✅ RAG + SWE-Llama 13B	1.20	2023-10-10	🔗	-	-
🏆 ✅ RAG + ChatGPT 3.5	0.40	2023-10-10	🔗	-	-

We will cover more in Language Agents class!

Math Language Models

Chain-of-Thought (CoT)

Standard Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The answer is 27. ❌

Chain of Thought Prompting

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✅

GSM8K (Cobbe et al., 2021)

- Middle school math word problems

Problem: Beth bakes 4, 2 dozen batches of cookies in a week. If these cookies are shared amongst 16 people equally, how many cookies does each person consume?

Solution: Beth bakes 4 2 dozen batches of cookies for a total of $4 \times 2 = \langle\langle 4 \times 2 = 8 \rangle\rangle 8$ dozen cookies

There are 12 cookies in a dozen and she makes 8 dozen cookies for a total of $12 \times 8 = \langle\langle 12 \times 8 = 96 \rangle\rangle 96$ cookies

She splits the 96 cookies equally amongst 16 people so they each eat $96/16 = \langle\langle 96/16 = 6 \rangle\rangle 6$ cookies

Final Answer: 6

Problem: Mrs. Lim milks her cows twice a day. Yesterday morning, she got 68 gallons of milk and in the evening, she got 82 gallons. This morning, she got 18 gallons fewer than she had yesterday morning. After selling some gallons of milk in the afternoon, Mrs. Lim has only 24 gallons left. How much was her revenue for the milk if each gallon costs \$3.50?

Mrs. Lim got 68 gallons - 18 gallons = $\langle\langle 68 - 18 = 50 \rangle\rangle 50$ gallons this morning.

So she was able to get a total of 68 gallons + 82 gallons + 50 gallons = $\langle\langle 68 + 82 + 50 = 200 \rangle\rangle 200$ gallons.

She was able to sell 200 gallons - 24 gallons = $\langle\langle 200 - 24 = 176 \rangle\rangle 176$ gallons.

Thus, her total revenue for the milk is $\$3.50/\text{gallon} \times 176 \text{ gallons} = \langle\langle 3.50 \times 176 = 616 \rangle\rangle 616$.

Final Answer: 616

Problem: Tina buys 3 12-packs of soda for a party. Including Tina, 6 people are at the party. Half of the people at the party have 3 sodas each, 2 of the people have 4, and 1 person has 5. How many sodas are left over when the party is over?

Solution: Tina buys 3 12-packs of soda, for $3 \times 12 = \langle\langle 3 \times 12 = 36 \rangle\rangle 36$ sodas

6 people attend the party, so half of them is $6/2 = \langle\langle 6/2 = 3 \rangle\rangle 3$ people

Each of those people drinks 3 sodas, so they drink $3 \times 3 = \langle\langle 3 \times 3 = 9 \rangle\rangle 9$ sodas

Two people drink 4 sodas, which means they drink $2 \times 4 = \langle\langle 4 \times 2 = 8 \rangle\rangle 8$ sodas

With one person drinking 5, that brings the total drank to $5 + 9 + 8 + 3 = \langle\langle 5 + 9 + 8 + 3 = 25 \rangle\rangle 25$ sodas

As Tina started off with 36 sodas, that means there are $36 - 25 = \langle\langle 36 - 25 = 11 \rangle\rangle 11$ sodas left

Final Answer: 11

MATH (Hendricks et al., 2021)

- Competition mathematics problems

MATH Dataset (Ours)

Problem: Tom has a red marble, a green marble, a blue marble, and three identical yellow marbles. How many different groups of two marbles can Tom choose?

Solution: There are two cases here: either Tom chooses two yellow marbles (1 result), or he chooses two marbles of different colors ($\binom{4}{2} = 6$ results). The total number of distinct pairs of marbles Tom can choose is $1 + 6 = \boxed{7}$.

Problem: The equation $x^2 + 2x = i$ has two complex solutions. Determine the product of their real parts.

Solution: Complete the square by adding 1 to each side. Then $(x + 1)^2 = 1 + i = e^{\frac{i\pi}{4}} \sqrt{2}$, so $x + 1 = \pm e^{\frac{i\pi}{8}} \sqrt[4]{2}$. The desired product is then $(-1 + \cos(\frac{\pi}{8}) \sqrt[4]{2})(-1 - \cos(\frac{\pi}{8}) \sqrt[4]{2}) = 1 - \cos^2(\frac{\pi}{8}) \sqrt{2} = 1 - \frac{(1 + \cos(\frac{\pi}{4}))}{2} \sqrt{2} = \boxed{\frac{1 - \sqrt{2}}{2}}$.

- Step-by-step solutions written in LATEX and natural language.
- Models are tasked with generating tokens to construct the final (boxed)

Math Pre-training: Minerva

Data source	Proportion of data	Tokens	Present during pretraining
Math Web Pages	47.5%	17.5B	No
arXiv	47.5%	21.0B	No
General Natural Language Data	5%	>100B	Yes

- Models were trained on a dataset of 38.5B tokens from webpages filtered for mathematical content and papers from the arXiv preprint server.
- The dataset includes general natural language data, which is the same as the one used for pretraining PaLM.
- Mathematical webpages were processed to remove most HTML tags while preserving MathJax expressions, LATEX symbols, and formatting.

Minerva Performance

	MATH	OCWCourses	GSM8k	MMLU-STEM
PaLM 8B	1.5%	1.5%	4.1%	22.0%
Minerva 8B	14.1%	7.7%	16.2%	35.6%
Minerva 8B, maj1@k	25.4%	12.5%	28.4%	43.4%
PaLM 62B	4.4%	5.9%	33.0%	39.1%
Minerva 62B	27.6%	12.9%	52.4%	53.9%
Minerva 62B, maj1@k	43.4%	23.5%	68.5%	63.5%
PaLM 540B	8.8%	7.1%	56.5%	58.7%
Minerva 540B	33.6%	17.6%	58.8%	63.9%
Minerva 540B, maj1@k	50.3%	30.8%	78.5%	75.0%
OpenAI davinci-002	19.1%	14.8%	-	-
Published SOTA	6.9% ^a	-	74.4% ^b	54.9% ^c

Inference-Time Techniques

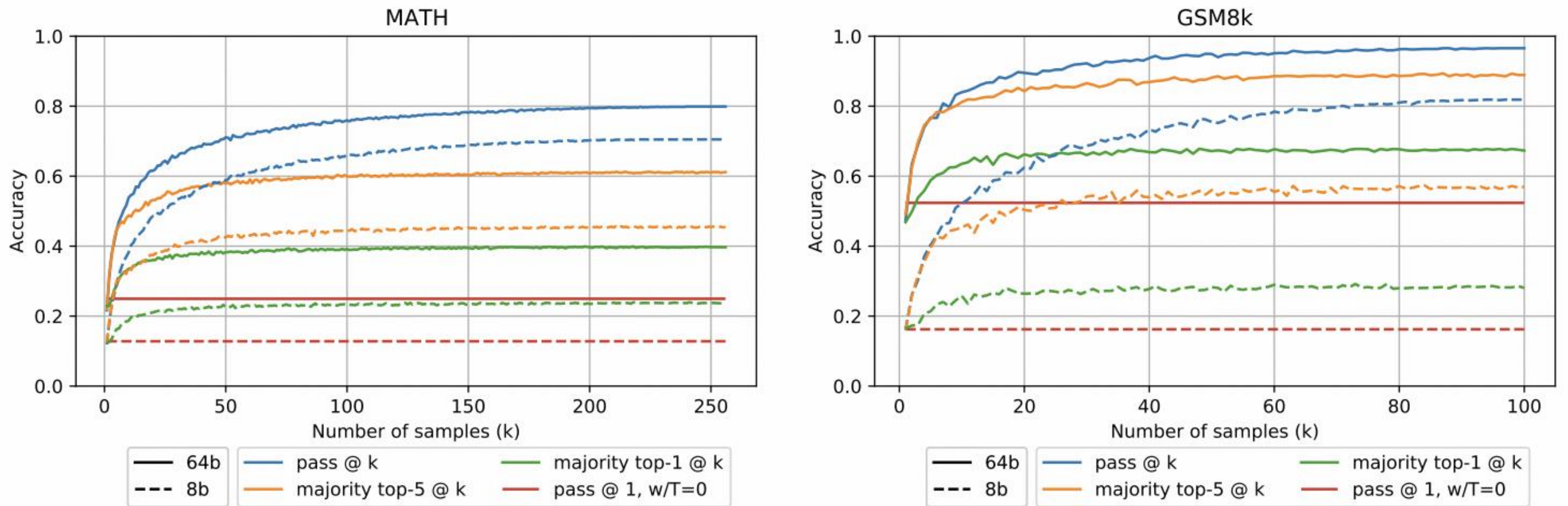
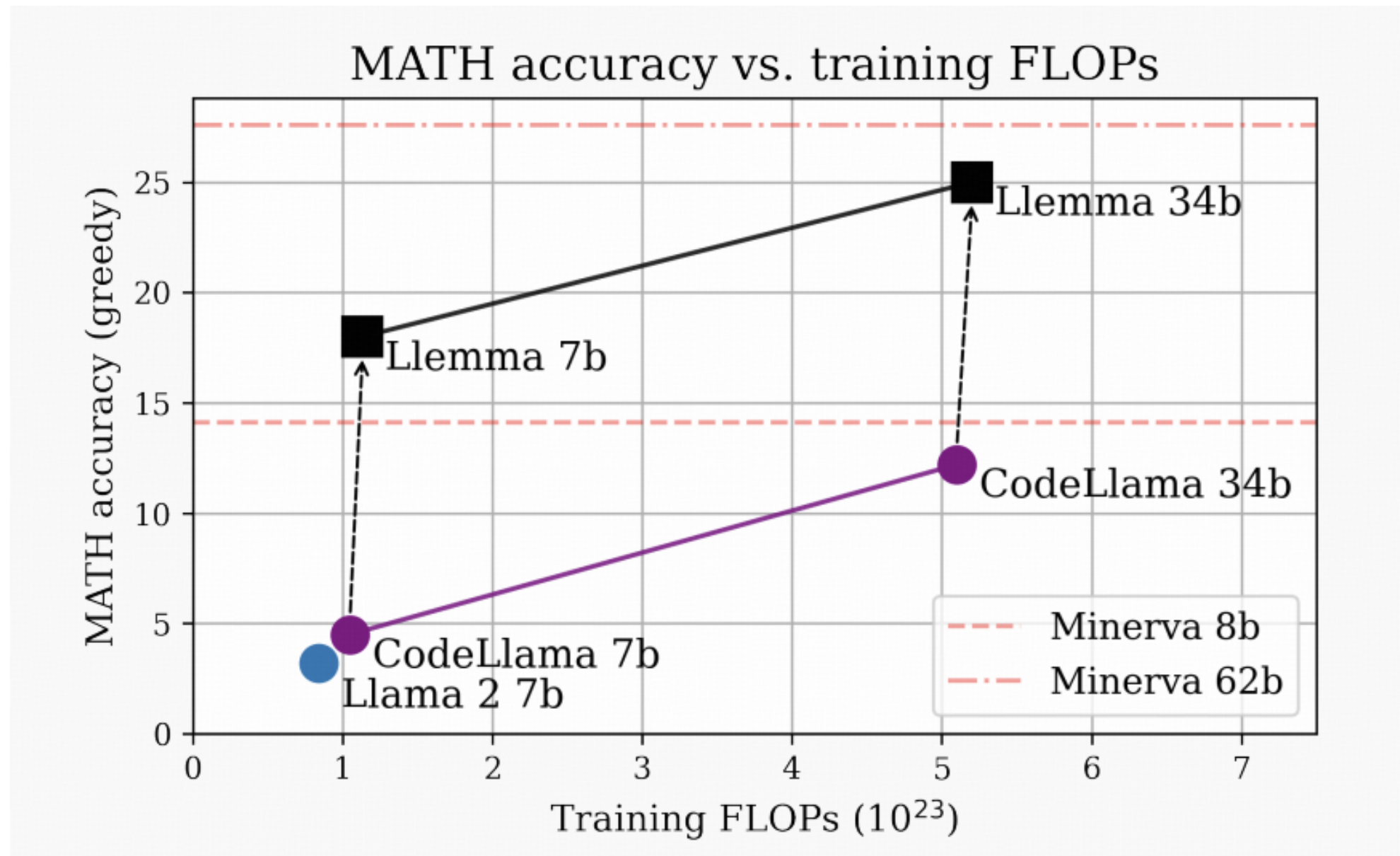


Figure 6: Accuracy as a function of k , the number of samples per task. Majority voting performance saturates quickly while $\text{pass}@k$ seems to continue improving slowly. Accuracies were computed using exact string match (without SymPy processing).

We will cover more in Inference Algorithm class!

LLEMMA: An open LM for Math



LLEMMA improves with a modest amount of math-specific compute

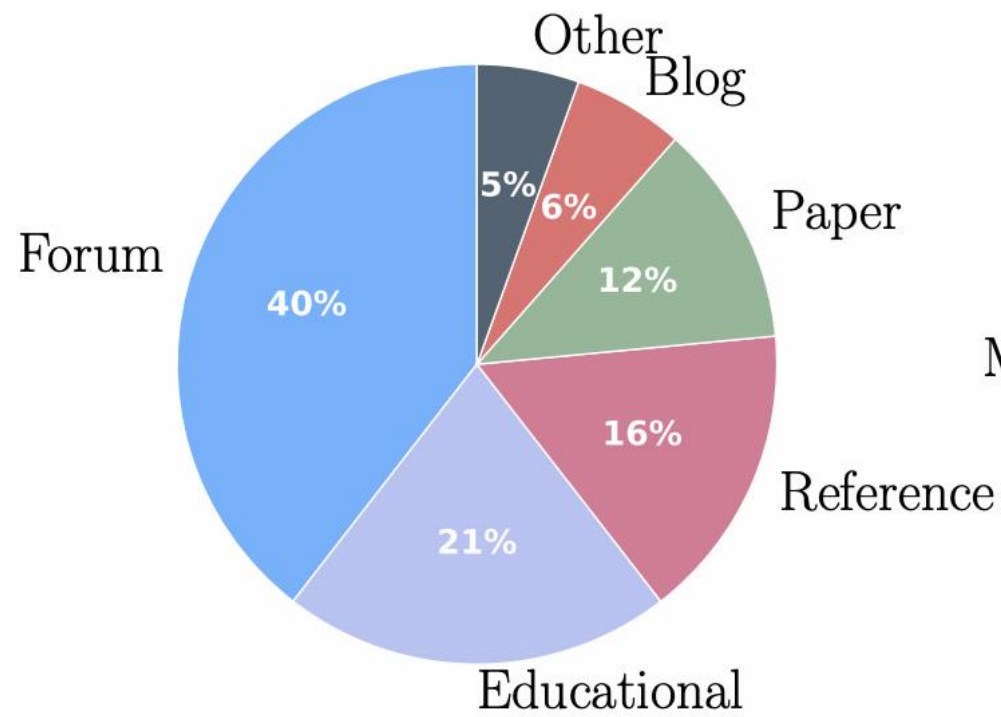
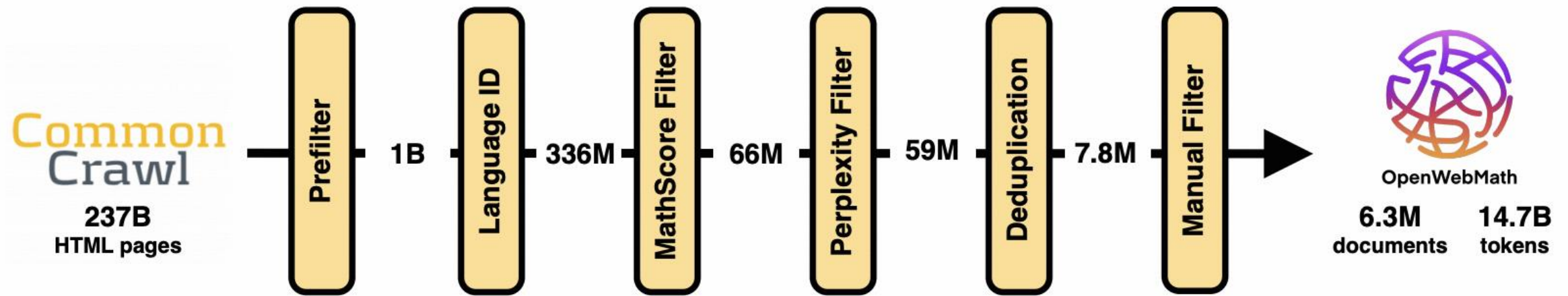
LLEMMA Data: Proof-Pile-2

Model	Adaptation tokens	Open
Minerva-8b	164B	✗
Minerva-62b	109B	✗
LLEMMA-7b (ours)	200B	✓
LLEMMA-34b (ours)	50B	✓

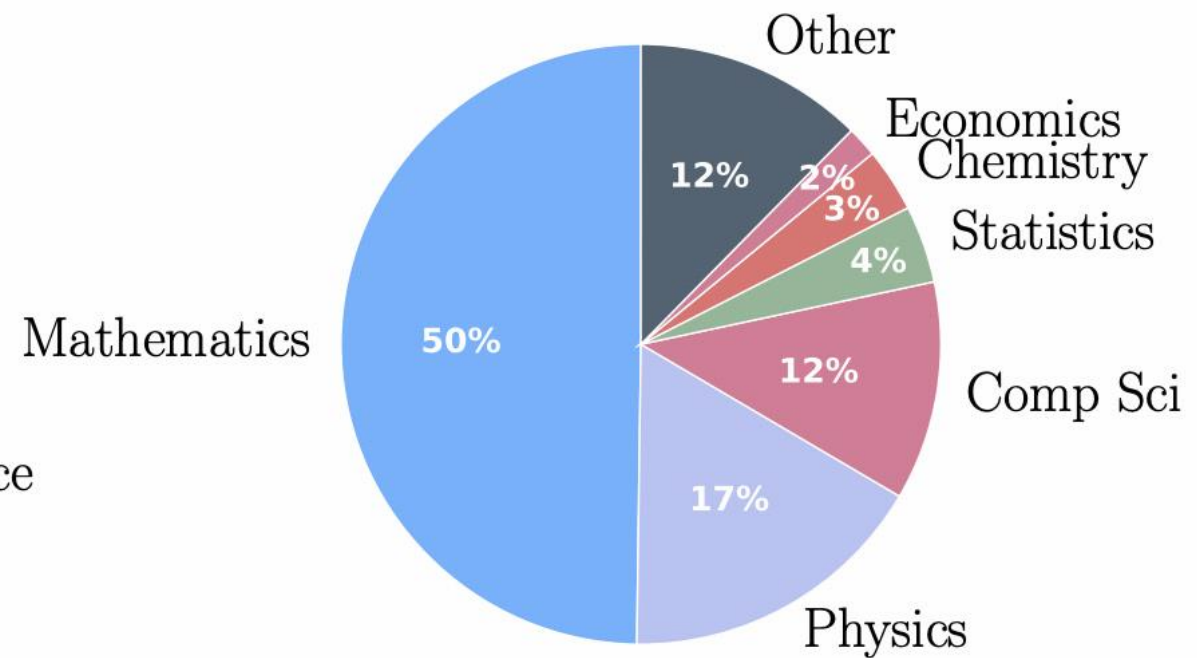
Dataset	Tokens	Open
Minerva Dataset	38.5B	✗
Proof-Pile-2 (ours)	55B	✓
Code (AlgebraicStack)	11B	✓
OpenWebMath (Paster et al., 2023))	15B	✓
ArXiv (Computer, 2023))	29B	✓

Figure 2: Comparison of LLEMMA and Minerva training

OpenWebMath

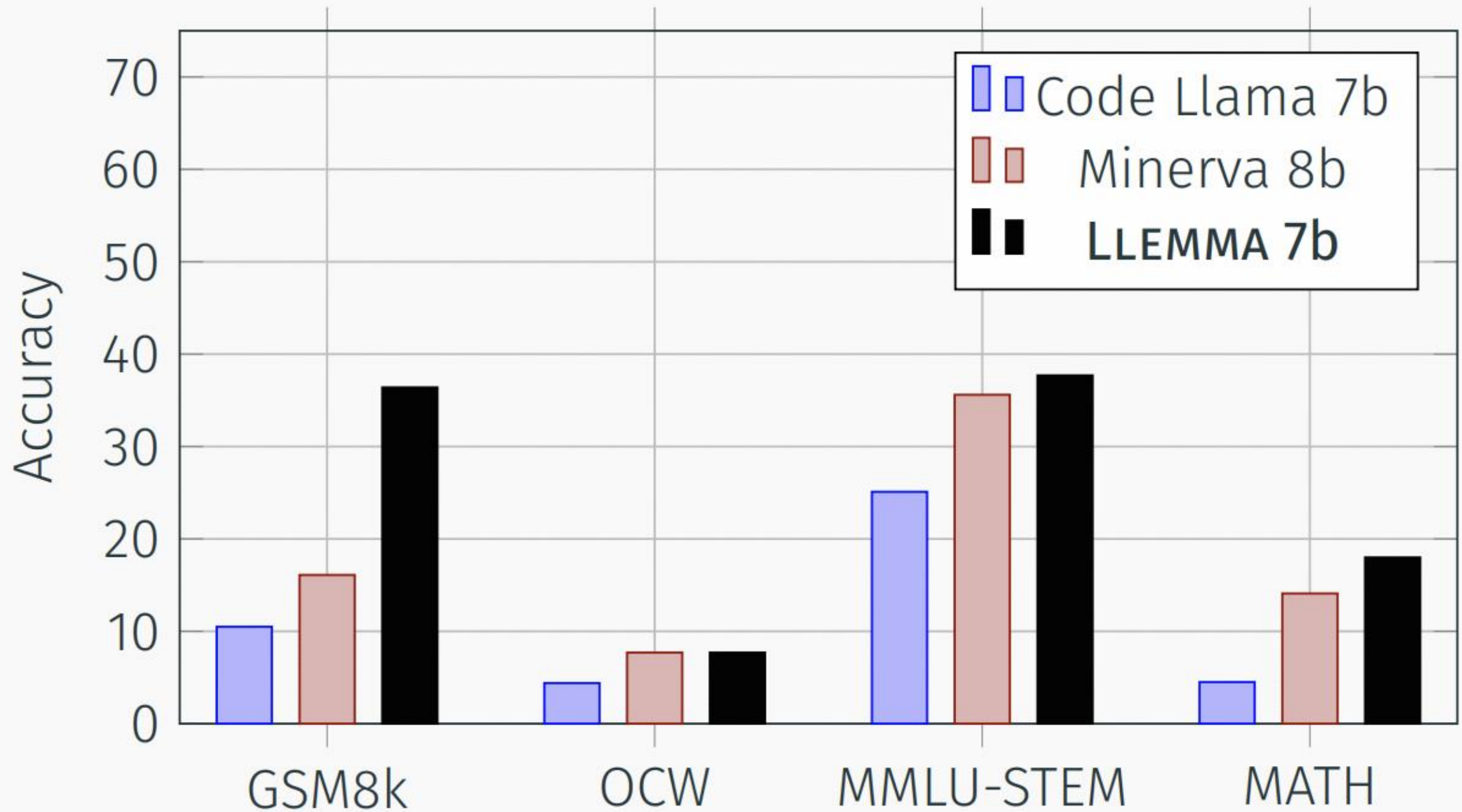


Types of OpenWebMath documents

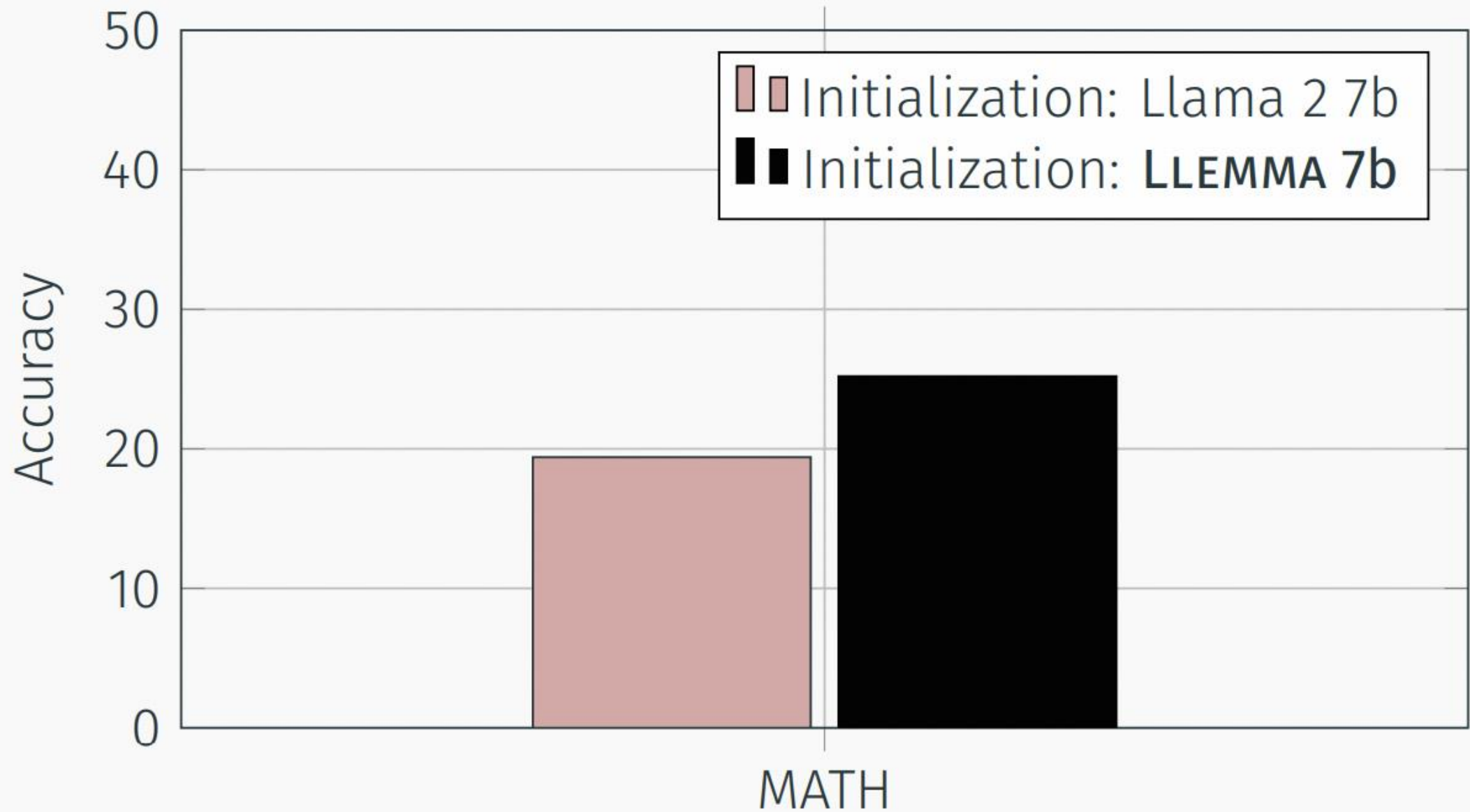


Subjects of OpenWebMath documents

LLEMMA Performance

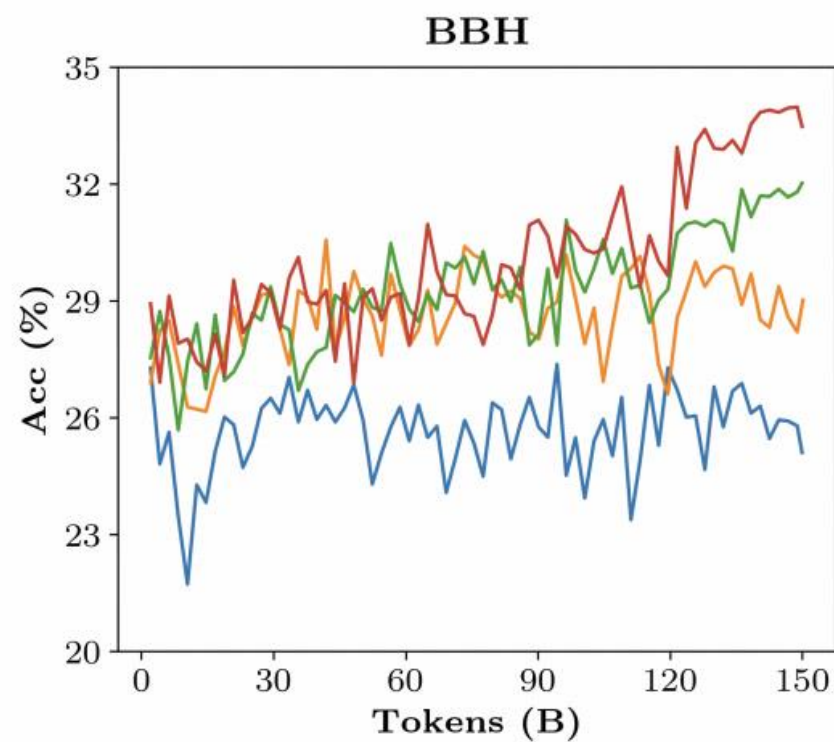
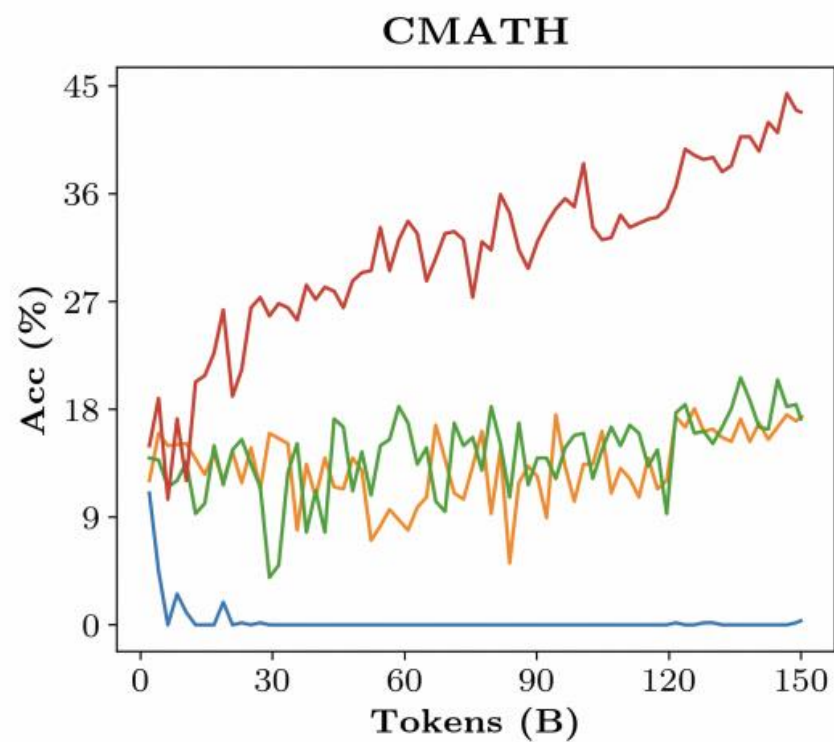
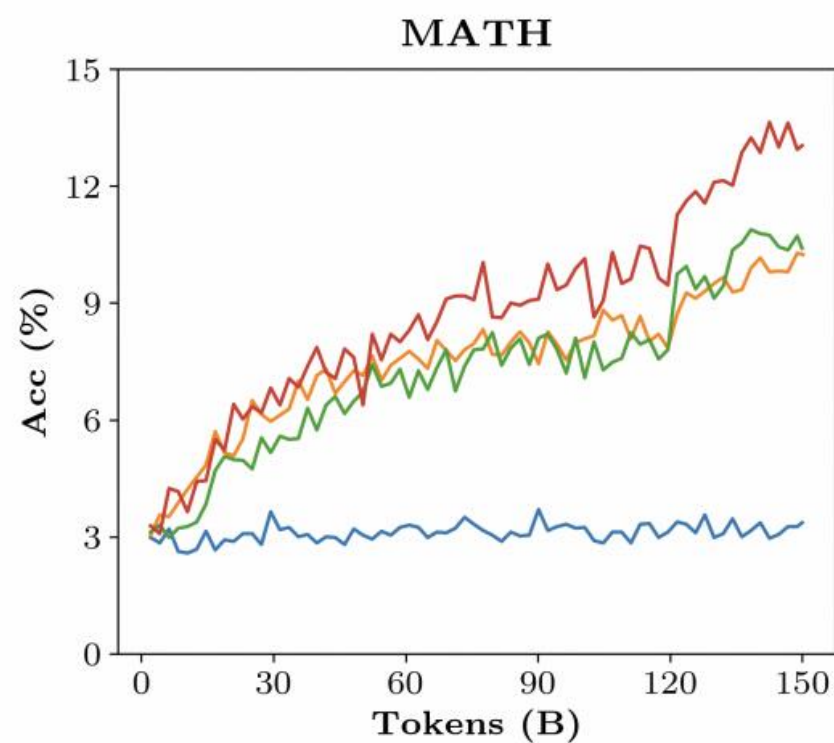
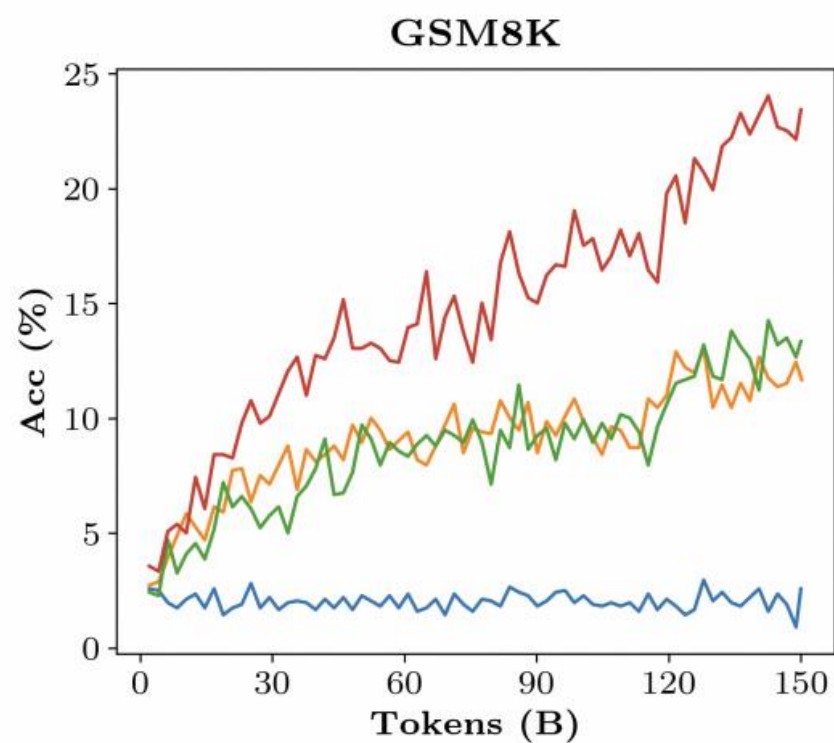


LLEMMMA Performance

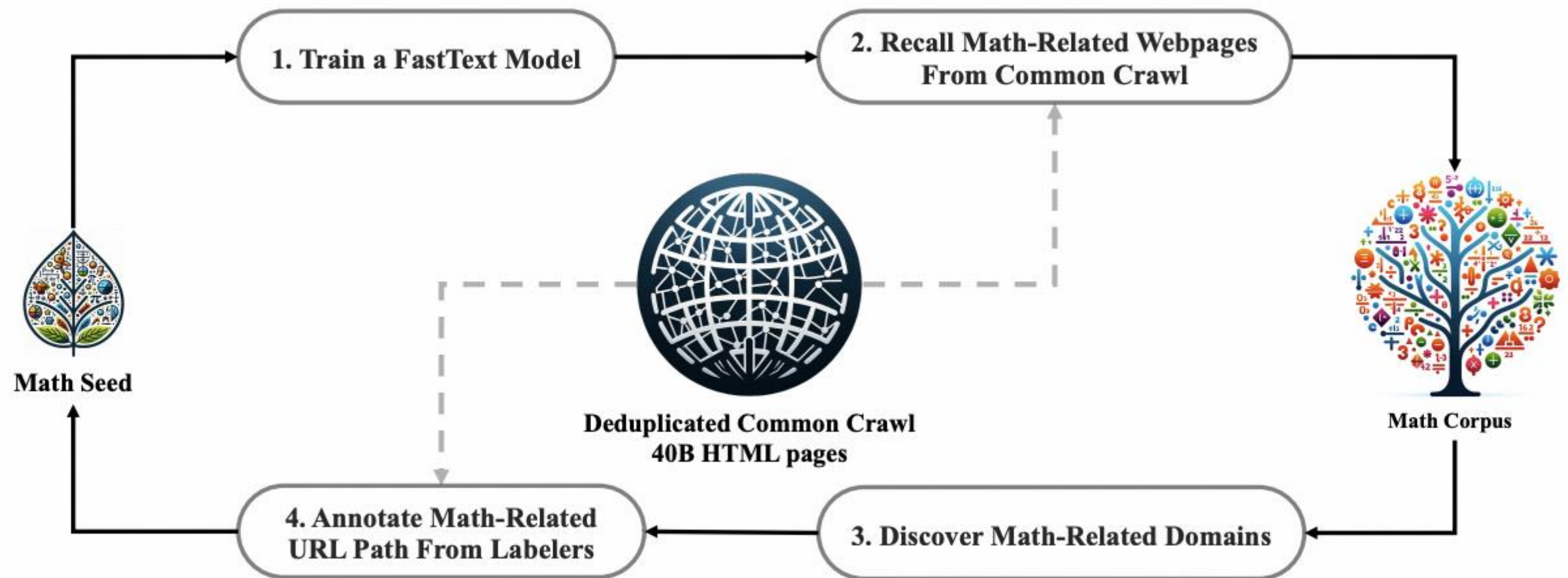


LLEMMMA vs. Llama 2 as initialization for finetuning on MetaMathQA

DeepSeek Math



DeepSeekMath Corpus



DeepSeekMath Performance

Model	Size	English Benchmarks				
		GSM8K	MATH	OCW	SAT	MMLU STEM
Closed-Source Base Model						
Minerva	7B	16.2%	14.1%	7.7%	-	35.6%
Minerva	62B	52.4%	27.6%	12.0%	-	53.9%
Minerva	540B	58.8%	33.6%	17.6%	-	63.9%
Open-Source Base Model						
Mistral	7B	40.3%	14.3%	9.2%	71.9%	51.1%
Llemma	7B	37.4%	18.1%	6.3%	59.4%	43.1%
Llemma	34B	54.0%	25.3%	10.3%	71.9%	52.9%
DeepSeekMath-Base	7B	64.2%	36.2%	15.4%	84.4%	56.5%

Training on Code Improves Math

Training Setting	Training Tokens			w/o Tool Use		
	General	Code	Math	GSM8K	MATH	CMATH
No Continual Training	–	–	–	2.9%	3.0%	12.3%
Two-Stage Training						
Stage 1: General Training	400B	–	–	2.9%	3.2%	14.8%
Stage 2: Math Training	–	–	150B	19.1%	14.4%	37.2%
Stage 1: Code Training	–	400B	–	5.9%	3.6%	19.9%
Stage 2: Math Training	–	–	150B	21.9%	15.3%	39.7%
One-Stage Training						
Math Training	–	–	150B	20.5%	13.1%	37.6%
Code & Math Mixed Training	–	400B	150B	17.6%	12.1%	36.3%

Write Code to Solve Math Problems

Chain-of-Thought (Wei et al., 2022)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves. They sold 93 in the morning and 39 in the afternoon. So they sold $93 + 39 = 132$ loaves. The grocery store returned 6 loaves. So they had $200 - 132 - 6 = 62$ loaves left.

The answer is 62.



Program-aided Language models (this work)

Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 tennis balls.

```
tennis_balls = 5
```

```
2 cans of 3 tennis balls each is
```

```
bought_balls = 2 * 3
```

```
tennis balls. The answer is
```

```
answer = tennis_balls + bought_balls
```

Q: The bakers at the Beverly Hills Bakery baked 200 loaves of bread on Monday morning. They sold 93 loaves in the morning and 39 loaves in the afternoon. A grocery store returned 6 unsold loaves. How many loaves of bread did they have left?

Model Output

A: The bakers started with 200 loaves

```
loaves_baked = 200
```

```
They sold 93 in the morning and 39 in the afternoon
```

```
loaves_sold_morning = 93
```

```
loaves_sold_afternoon = 39
```

```
The grocery store returned 6 loaves.
```

```
loaves_returned = 6
```

```
The answer is
```

```
answer = loaves_baked - loaves_sold_morning
```

```
- loaves_sold_afternoon + loaves_returned
```

```
>>> print(answer)
```

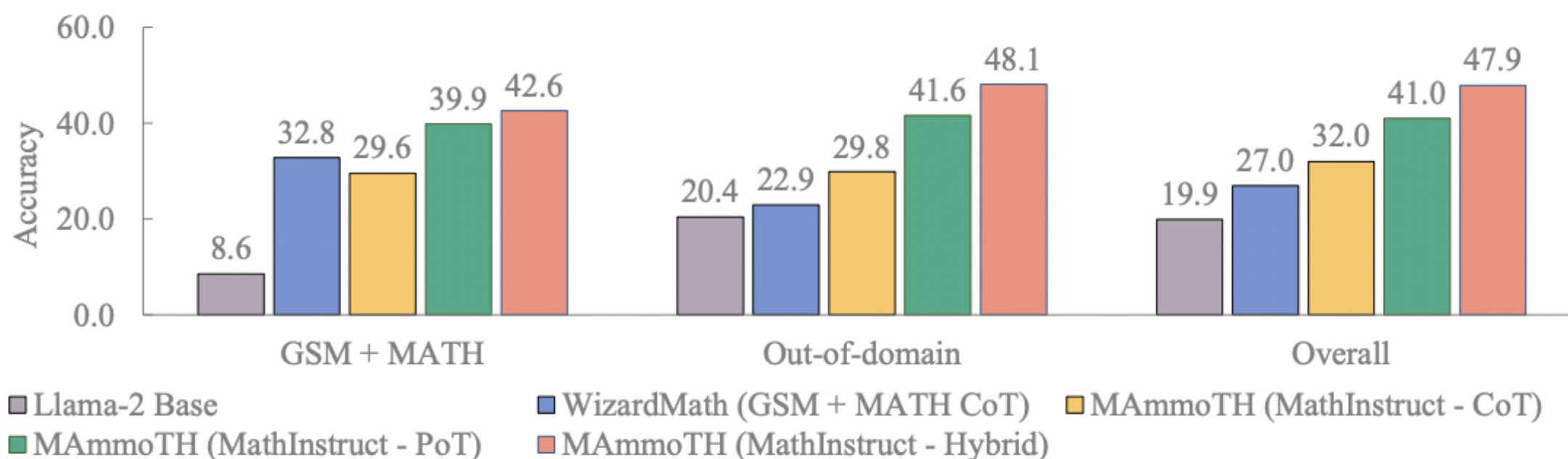
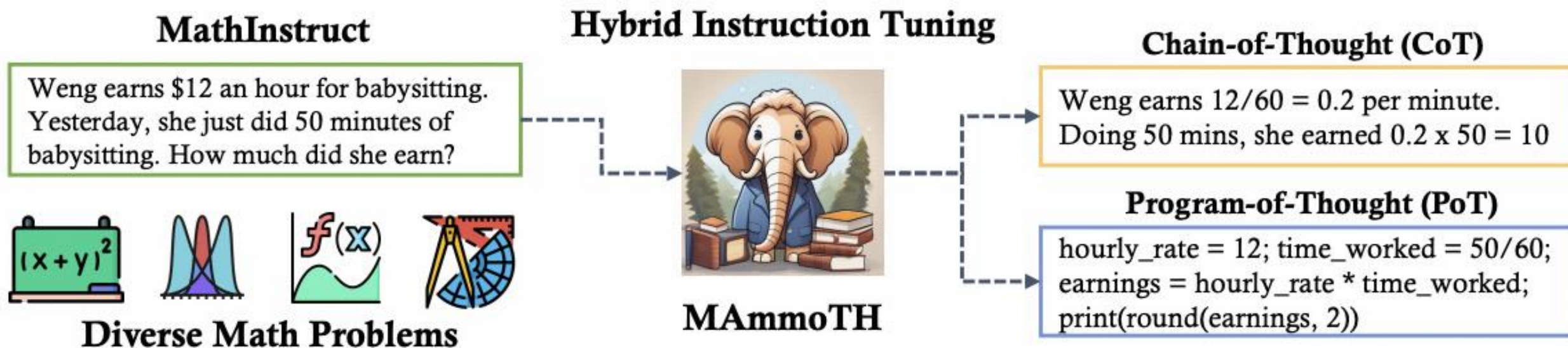
```
74
```



<https://arxiv.org/pdf/2211.10435>

<https://arxiv.org/abs/2211.12588>

MAMmoTH: Hybrid Thoughts Instruction Tuning



Questions?