

Rethinking Pretraining: Data and Architecture

Luke Zettlemoyer

University of Washington and Meta

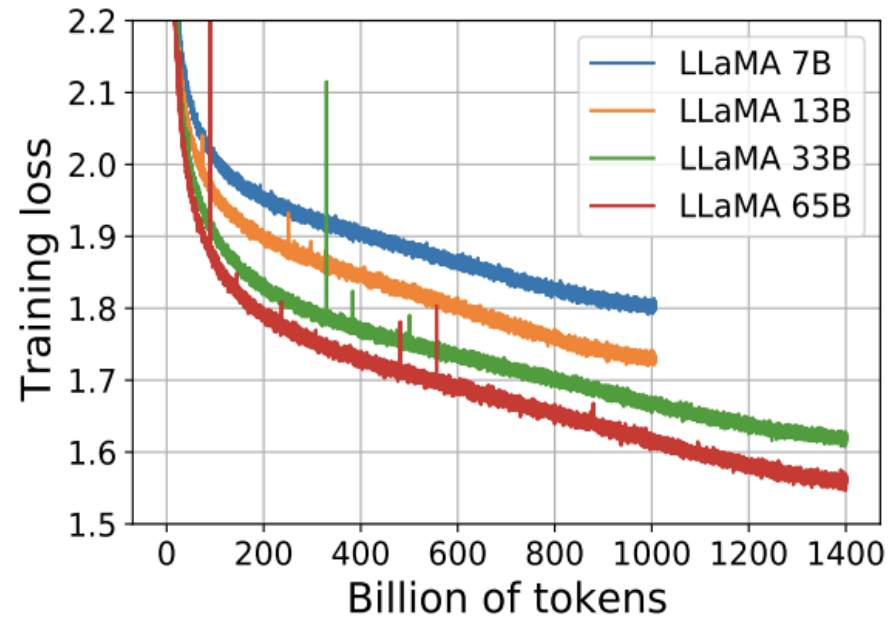
Based on projects led by Terra Blevins, Scott Geng, Marjan Ghazvininejad, Suchin Gururangan, Srinivasan Iyer, Omer Levy, Mike Lewis, Margaret Li, Shuyue Stella Li, Sewon Min, Artidoro Pagnoni, Rulin Shao, Weijia Shi, Rui Xin, Michihiro Yasunaga, Lili Yu, Chunting Zhou



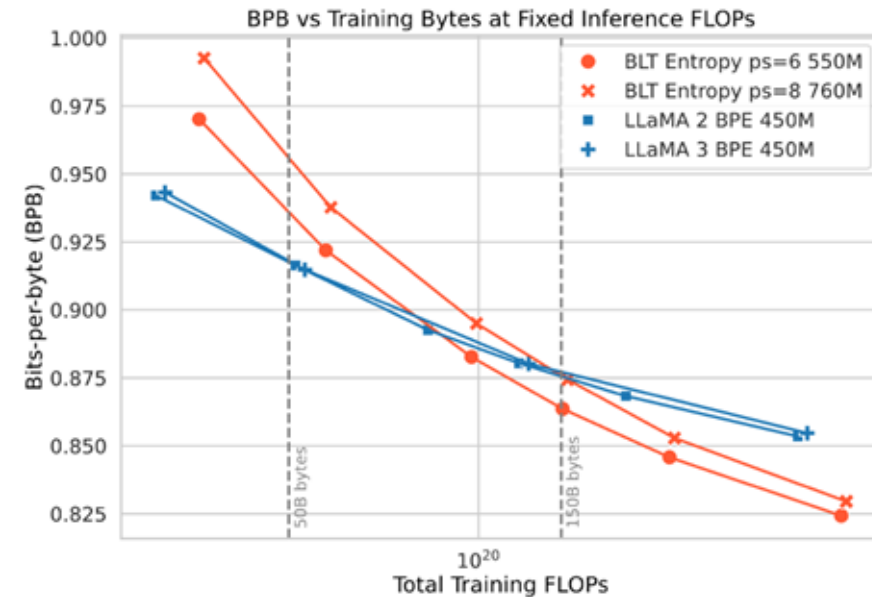
Claim: pretraining is all about the data!

(although I will talk mostly about architectures...)

New architectures can bend scaling curves...



Touvron et. al, 2023



Pagnoni et. al, 2025

... and also extract more from the data / unlock new capabilities

Data and Two Architectures

- Everything LLMs can do comes from pretraining (almost)
- Data long tail: tokenizer-free LLMs
- Data modularity: inference reconfigurable LLMs

1 Trillion is a VERY BIG NUMBER!!!



10,000 \$



1 million \$



1 billion \$



1 trillion \$

x 1500 if every training example (15T)
in modern models is \$1



Claim: Almost everything
comes from pretraining...

(What more could you really learn from another \lll 1T tokens...)

(lets try and see!!!)

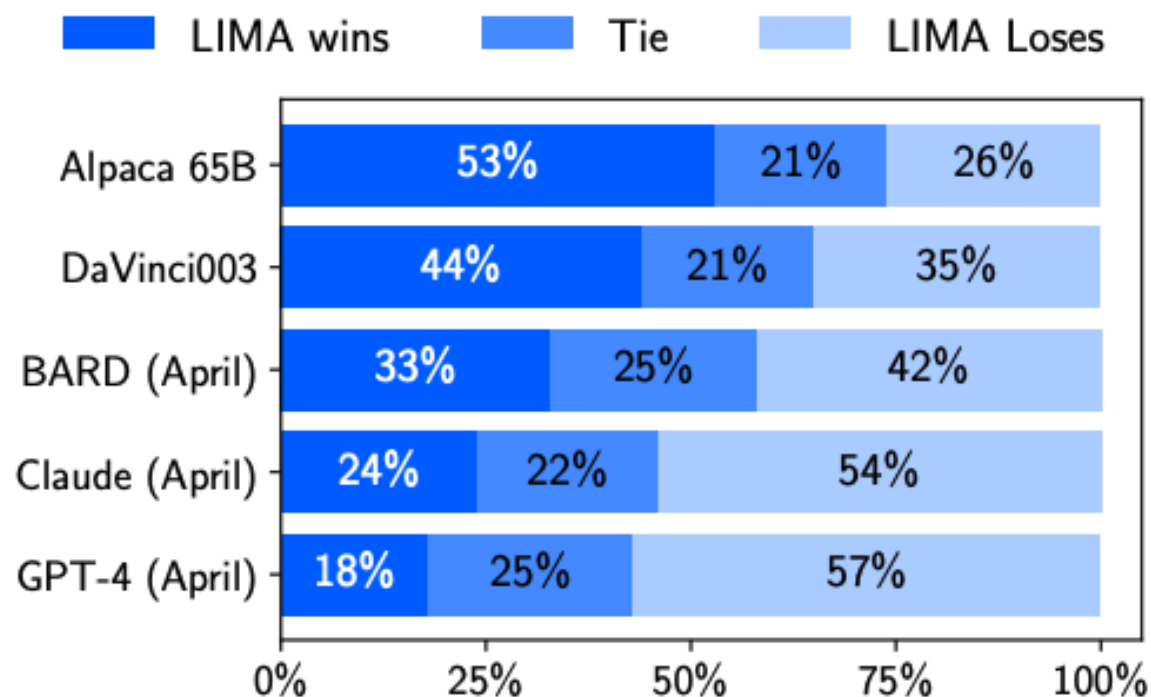
Simple Alignment Recipes

- SFT with very little data (LIMA, 2023)
- More data, self supervised, no annotations (ALMA, 2024)
- Works for inference scaling too (s1, 2025)
- Also RLVF with random rewards? (Spurious Rewards, 2025)

Less is More for Alignment (LIMA; Zhou et al, 2023)

Key Idea

1. Very carefully curate set of high-quality training examples
2. Do SFT, show you can radically change model behavior
3. Finding: 1,000 examples get ~80% of much more complex alignment scheme



LIMA: Diverse, High Quality Data is Crucial!

Source	#Examples	Avg Input Len.	Avg Output Len.
Training			
Stack Exchange (STEM)	200	117	523
Stack Exchange (Other)	200	119	530
wikiHow	200	12	1,811
Pushshift r/WritingPrompts	150	34	274
Natural Instructions	50	236	92
Paper Authors (Group A)	200	40	334

LIMA: Can Unlock Quite Complex Dialogues

LIMA with Dialogue Examples (1,030 Examples)

You are a scientist who just invented a time machine. Where do you travel first?

I would travel to Ancient Egypt.

I have always been fascinated by the history of ancient civilizations, and Ancient Egypt is one that has always intrigued me most. Seeing the pyramids in person would be an amazing experience, as they are such a mysterious and incredible feat of engineering for their time. Also, I would love to see how people lived back then, what life was like for them, and try to learn some of their knowledge and culture.

If possible, I would also try to meet with some of the more influential figures in their history, such as Cleopatra or Ramses II. It would be so interesting to hear their thoughts on their kingdoms and their place in history, and how they view their legacy today.

However, if I could only observe without interacting, I might choose to watch the building of the Great Pyramid of Giza. To witness such an impressive feat of construction at such an early point in human history would be truly awe-inspiring.

Could you turn it into a fictional essay?

Sure, here is a fictional story of my travels to Ancient Egypt:

Could you turn it into a fictional essay?

Sure, here is a fictional story of my travels to Ancient Egypt:

I step into the machine and set the controls for 2500 BC. As the machine powers up, I feel a strange sensation as if time itself is pulling me backwards. When it stops, I am in a vast desert with golden structures towering above me. I have arrived in ancient Egypt!

As I look around, I see people dressed in linen clothes [...]

[...] Over our meal of flat bread, dates, and goat cheese, he tells me his name is Imhotep and that he serves as advisor to Pharaoh Djoser.

Imhotep is fascinated by my stories of the future [...]

Before long, it was time for me to return home. I bid farewell to Imhotep and thanked him for his hospitality. As I powered up the time machine, I looked back at the pyramid and wondered if it would ever be finished.

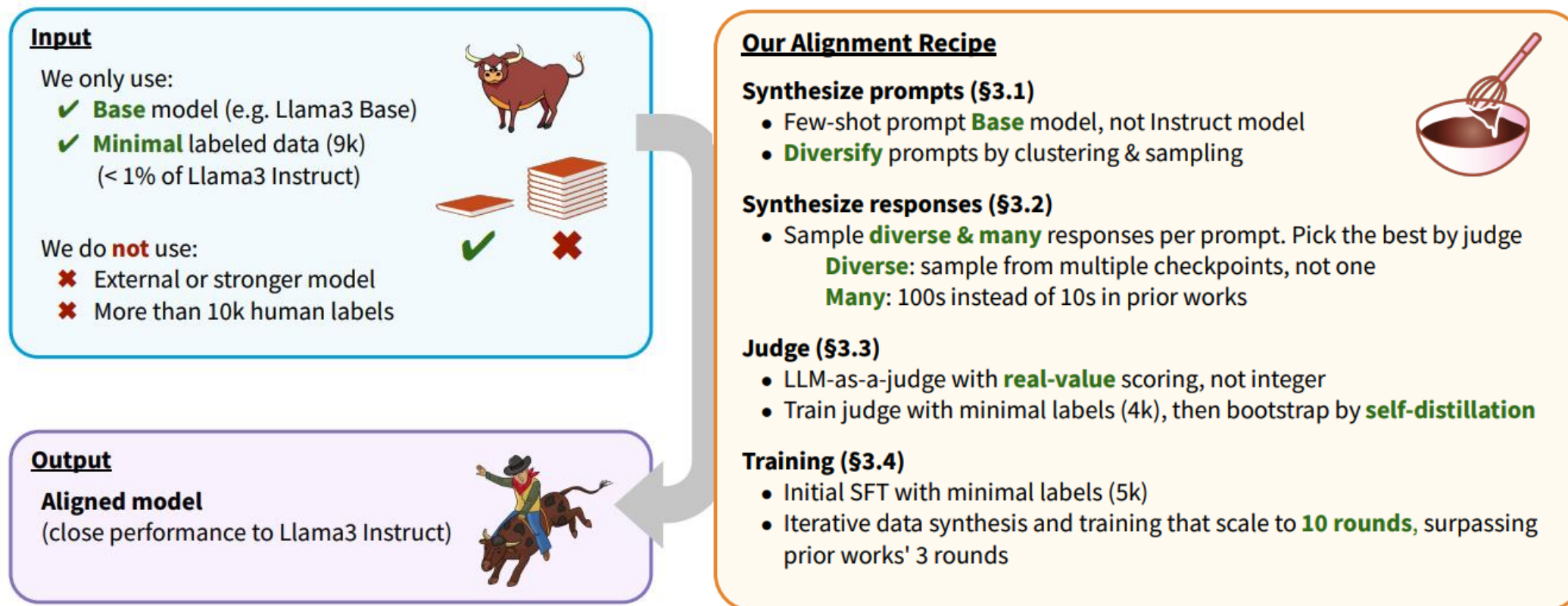
Can you create a title for the essay?

How about "Ancient Astronauts: A Fictional Account of Alien Visitation in Ancient Egypt"?

Why is the essay related to astronauts and aliens?

Alignment with Minimal Annotation (ALMA; Yasunaga et al, 2024)

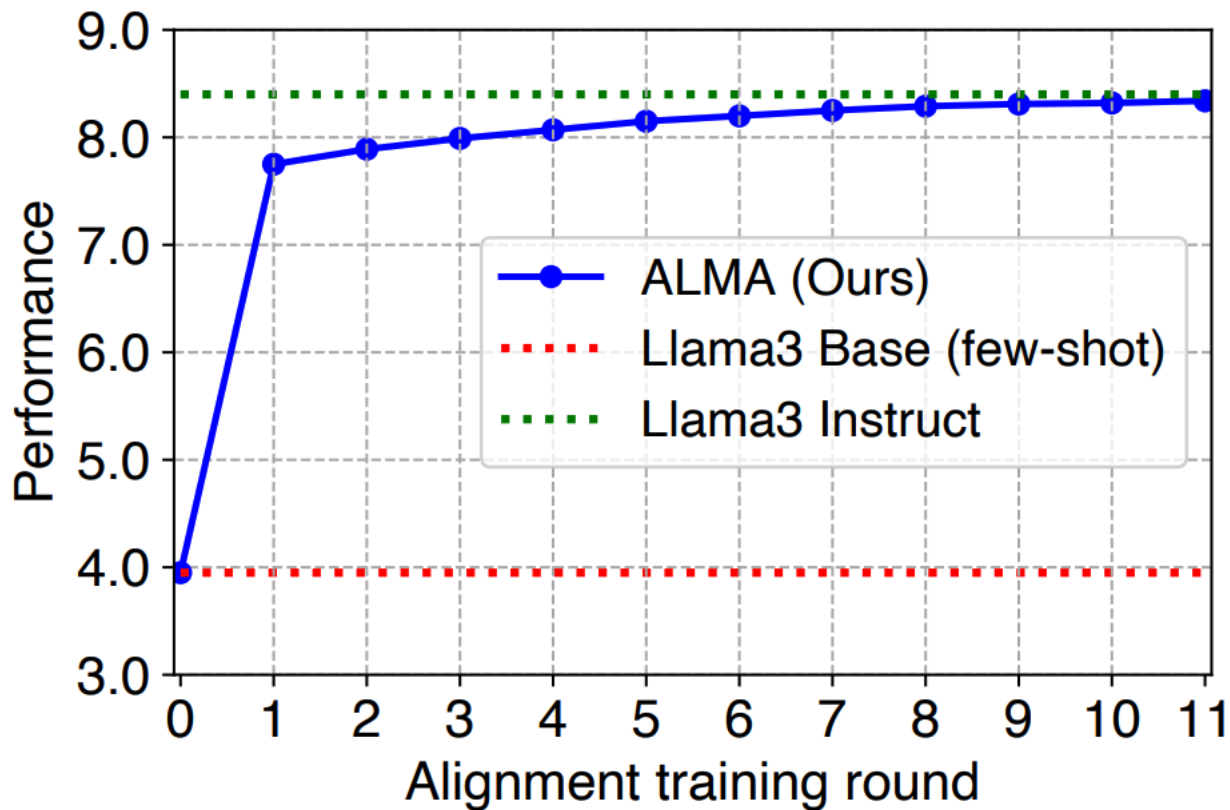
Key Idea: Again, start from small core, but now have model make its own data



Alignment with Minimal Annotation (ALMA; Yasunaga et al, 2024)

Key Finding

1. Data quality improves over multiple iterations
2. Matched strong alignment recipe (for the time) with no outside supervision
3. Final data has ~1M examples



Simple Test-time Scaling (s1; Muennighoff et al, 2025)

Key Idea

1. SFT on 1k traces of long inference reasoning
2. Balance data quality, difficulty, and diversity
3. Append “Wait” during inference, when needed

How many r in raspberry?

Question

Let's break down the process of counting the letter 'r' in the word "raspberry" ...

* First letter: 'r' - This is an 'r', count = 1.

* Second letter: 'a' - Not an 'r', count remains 1 ...

* Sixth letter: 'e' - Not an 'r', count remains 1.

* Seventh letter: 'r' - This is an 'r', count = 2.

* Eighth letter: 'y' - Not an 'r', count remains 2 ...

The number of 'r's in "raspberry" is 2.

Wait, let's re-read the question carefully. It asks "How many r in raspberry?" ... * r - a - s - p - b - e - r - r - y ... * First 'r' ... *

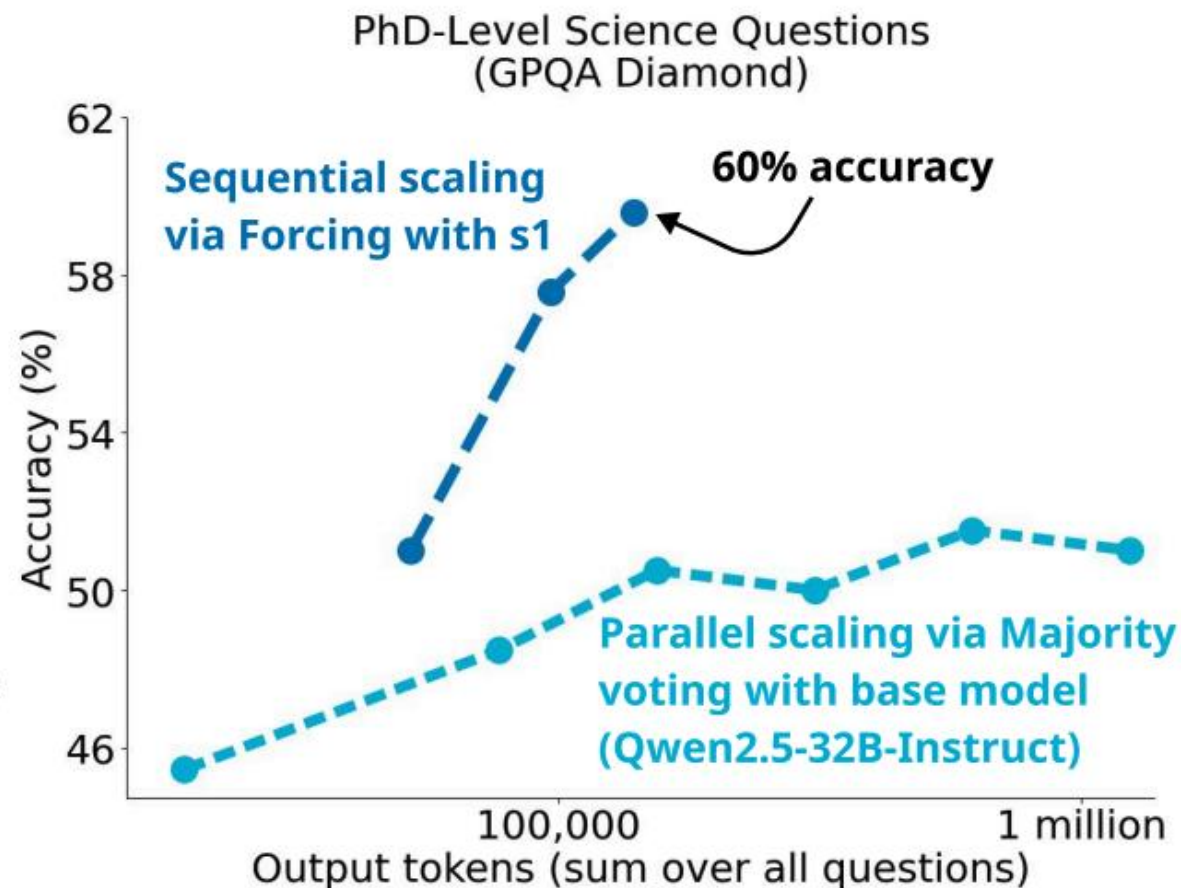
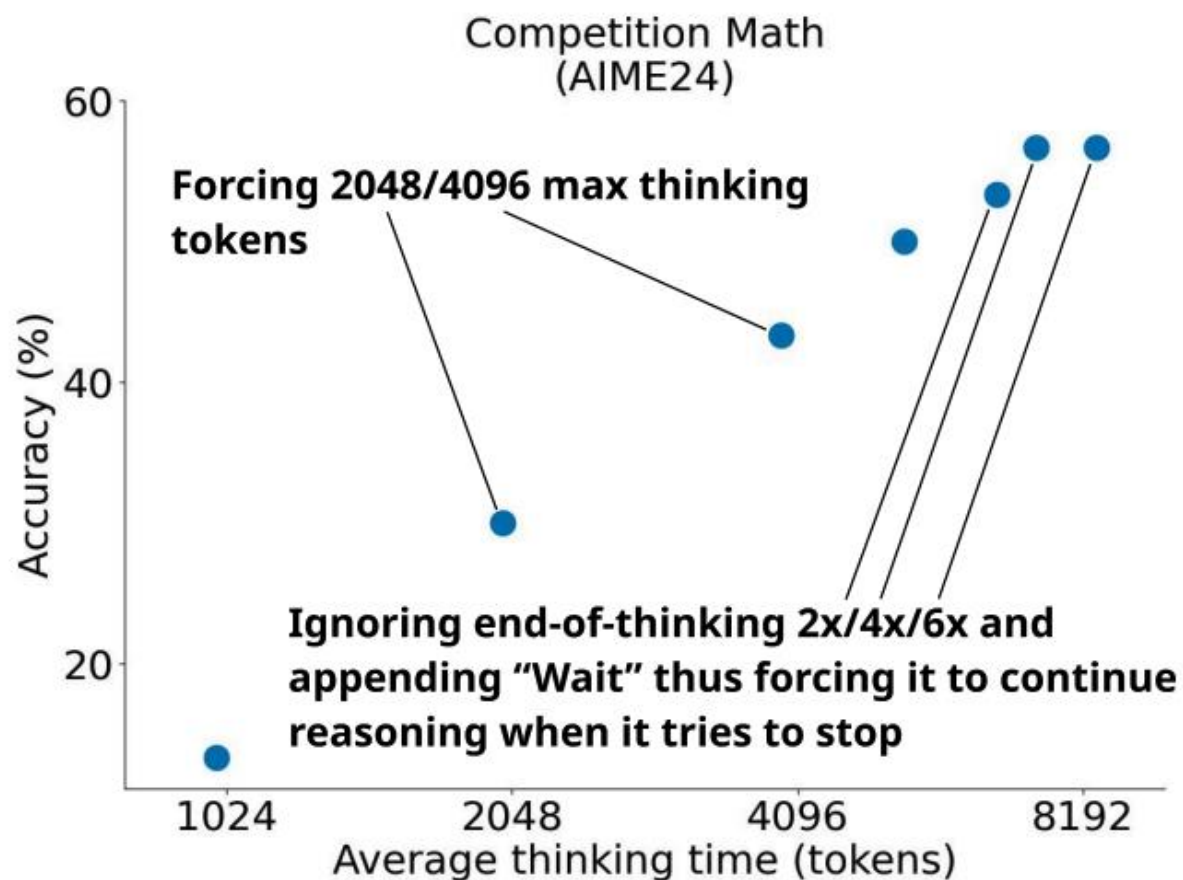
Second 'r' ... * Third 'r' ... Count = 3 ...

Reasoning trace

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is **3**

Response

Simple Test-time Scaling (s1; Muennighoff et al, 2025)

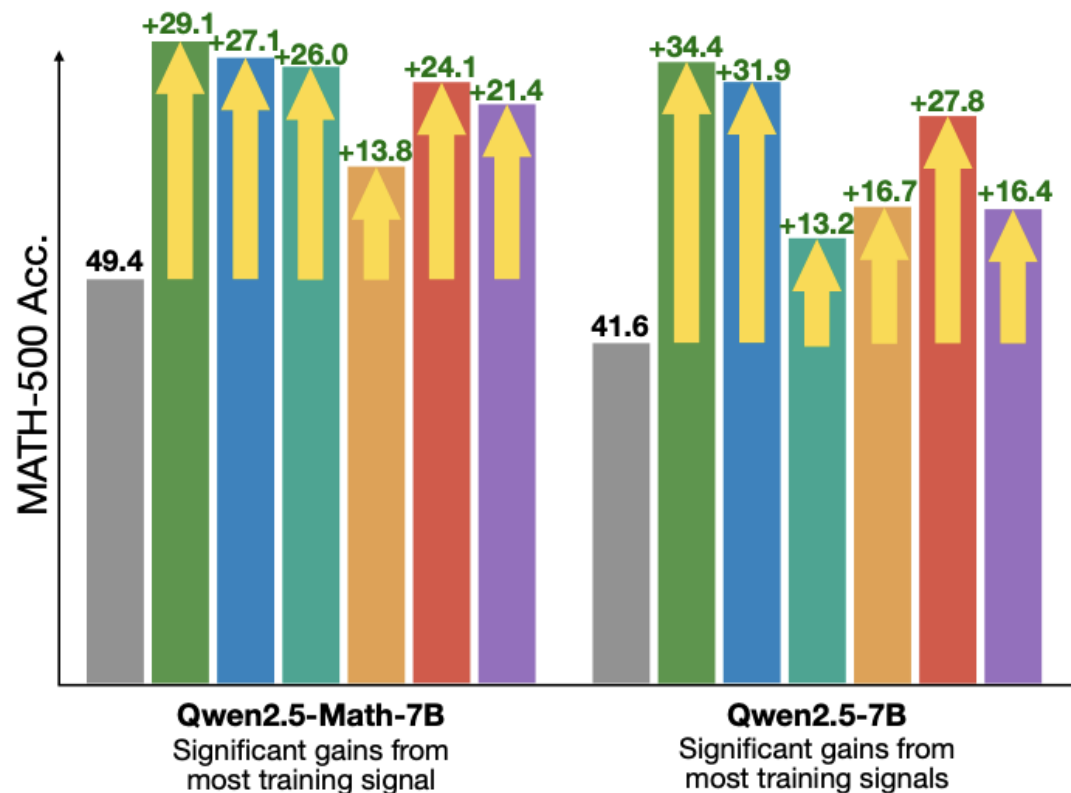
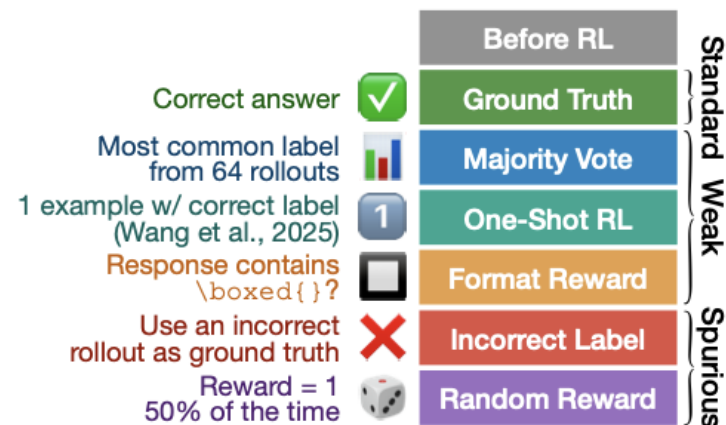


Spurious Rewards in RLVF

(Shao et al, 2025)

Key Finding

- RLVF improves models even with random rewards
- Not exactly clear why, paper has lots of analysis!
- Probably has something to do with the training data
- Not true for every model



THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.



Are you saying we don't need careful post training???

- Absolutely not!
- But, ***probably*** most of the capabilities are there already ... and what we are really doing is figuring out how to expose and better control them...
- Maybe recent very large scale RL post training suggests the opposite... but jury is still out in my opinion

Data and Two Architectures

Everything LLMs can do comes from pretraining (almost)

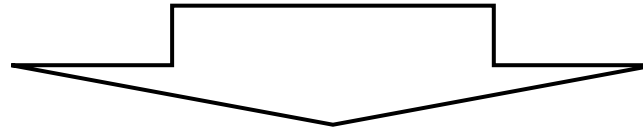
- Data long tail: tokenizer-free LLMs
- Data modular models: reconfigurable at test time

Challenge: How do we move
down the data long tail?

(it starts with the tokenizer....)

Tokenization: First Step for all LLMs

Daenerys Targaryen is in Game of Thrones, a fantasy epic by George R.R. Martin.



BPE Da enery s T arg ary en is in Game of Thrones , a fantasy epic by George R .R . Martin .

- Pros: limit lexicon size, no OOV, much more efficient!
- Cons: preprocessing step, hurts long tail, lots of other weird side effects, difficult to change after training

Points noted by Andrej Karpathy:

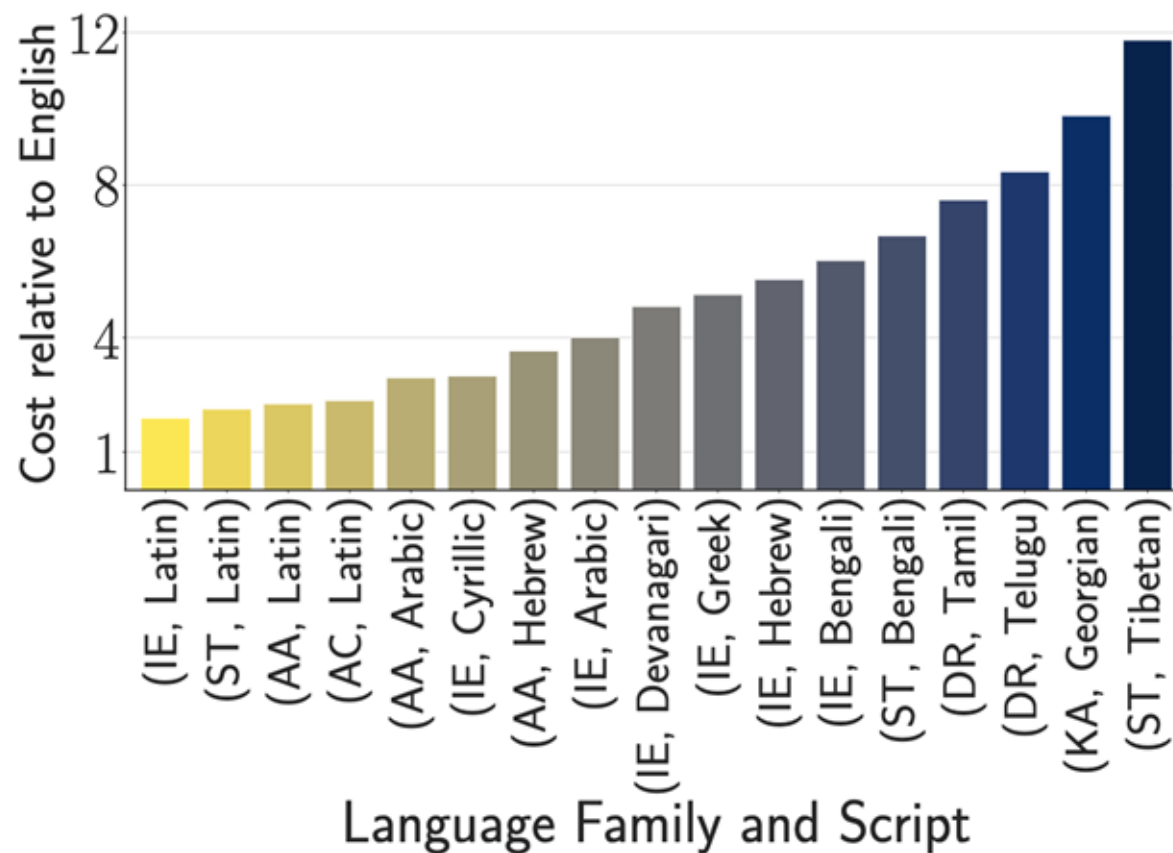
Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization.**
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization.**
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization.**
- Why is LLM bad at simple arithmetic? **Tokenization.**
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization.**
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization.**
- What is this weird warning I get about a "trailing whitespace"? **Tokenization.**
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization.**
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization.**
- Why is LLM not actually end-to-end language modeling? **Tokenization.**
- What is the real root of suffering? **Tokenization.**

BPE Tokenization Fairness

Also, fairness issues...

... less prominent languages are essentially byte level, have worse performance, higher costs in APIs, etc



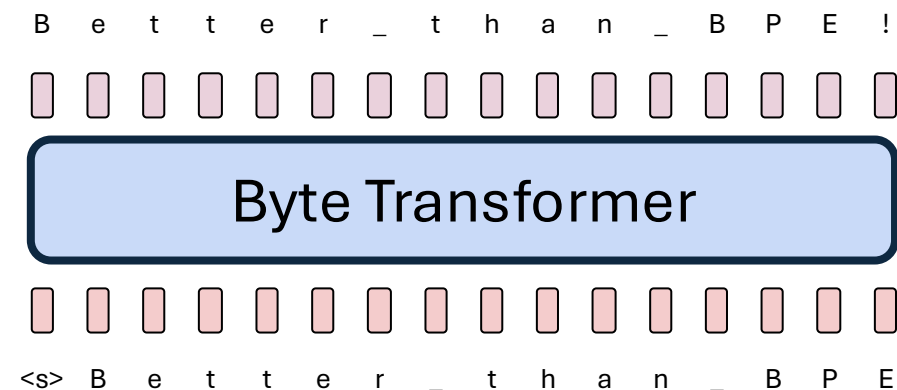
Ahia, Orevaoghene, et al. "Do all languages cost the same? tokenization in the era of commercial language models." arXiv preprint arXiv:2305.13707 (2023).

WE INTRODUCE A
NEW
ARCHITECTURE
THAT IS

- **Tokenizer-free:** directly model bytes
- **Hierarchical:** chunks bytes into larger units that get more compute
- **Dynamic:** chunks are variable size
- **Efficient:** if chunks are large (>4), can use less compute than BPE
- **Scales Better than BPE!** (up to 8B params / 1T tokens, at least...)

Architecture

- Byte Transformer Model
 - LM on every byte
- Actually works!
 - But roughly 20x more expensive to train...



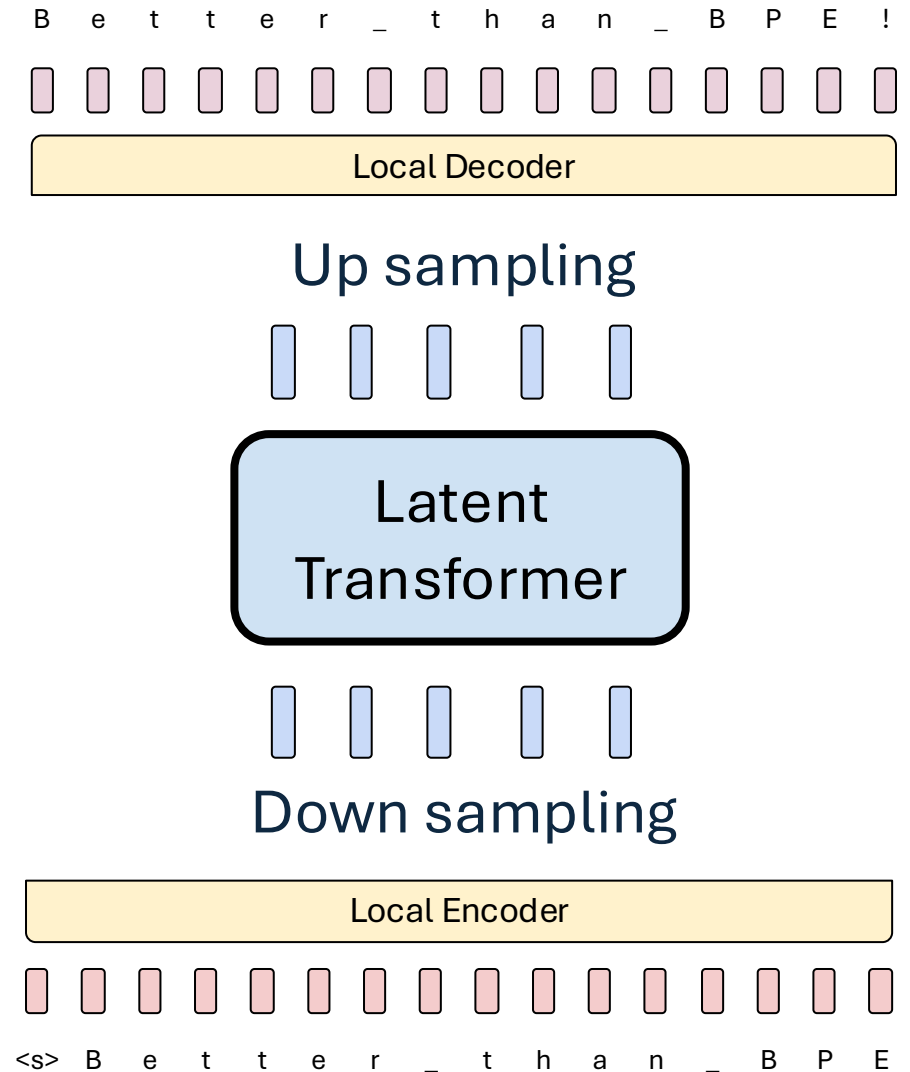
Key Idea: Add hierarchy

Latent Transformer operates on *patches*.

Encoder + Decoder << **Latent Transformer**

Open Question 1: how to patch?

Open Question 2: how does information flow from encoder->latent->decoder?



Baseline: Fixed Strides

Patching:

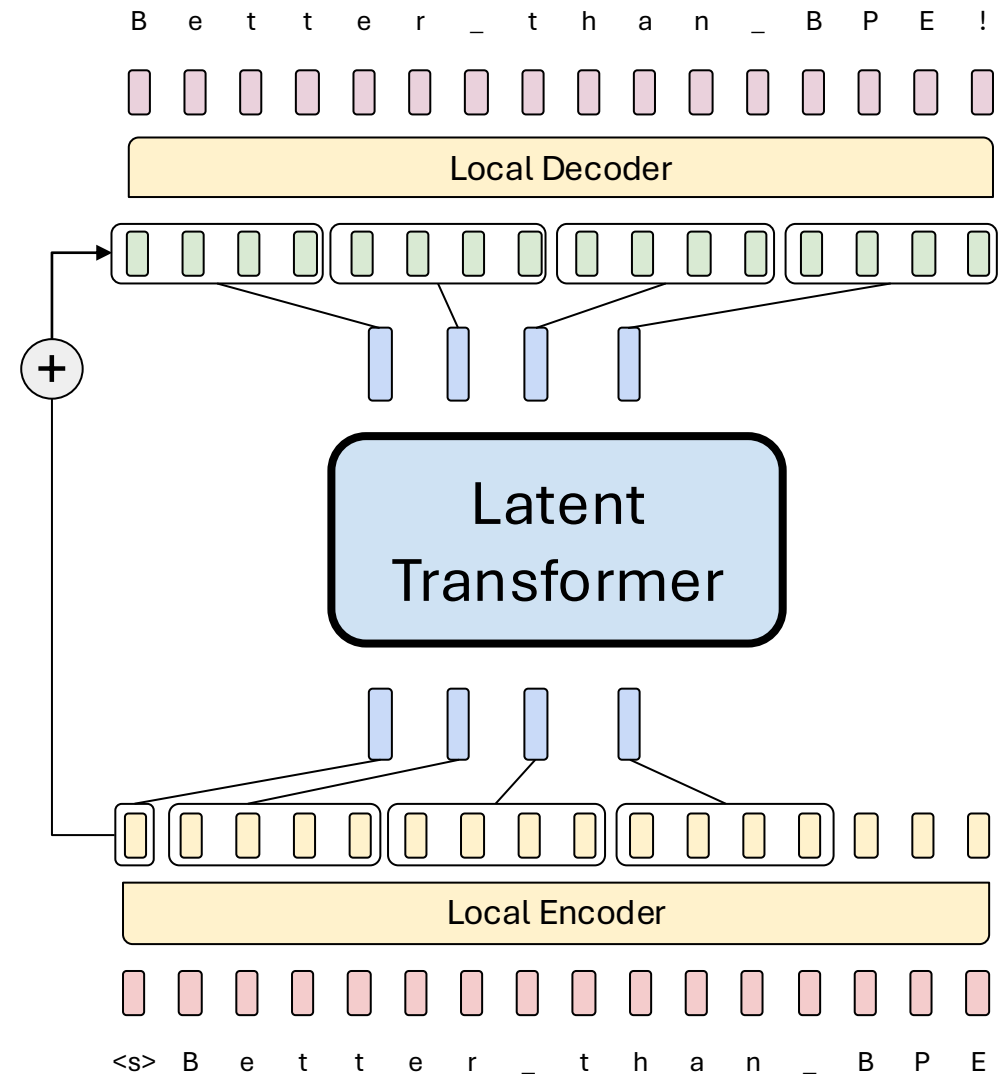
→ **Fixed strides / size**

Information Flow:

→ **Average Pooling** + projection

Does not match BPE tokenization

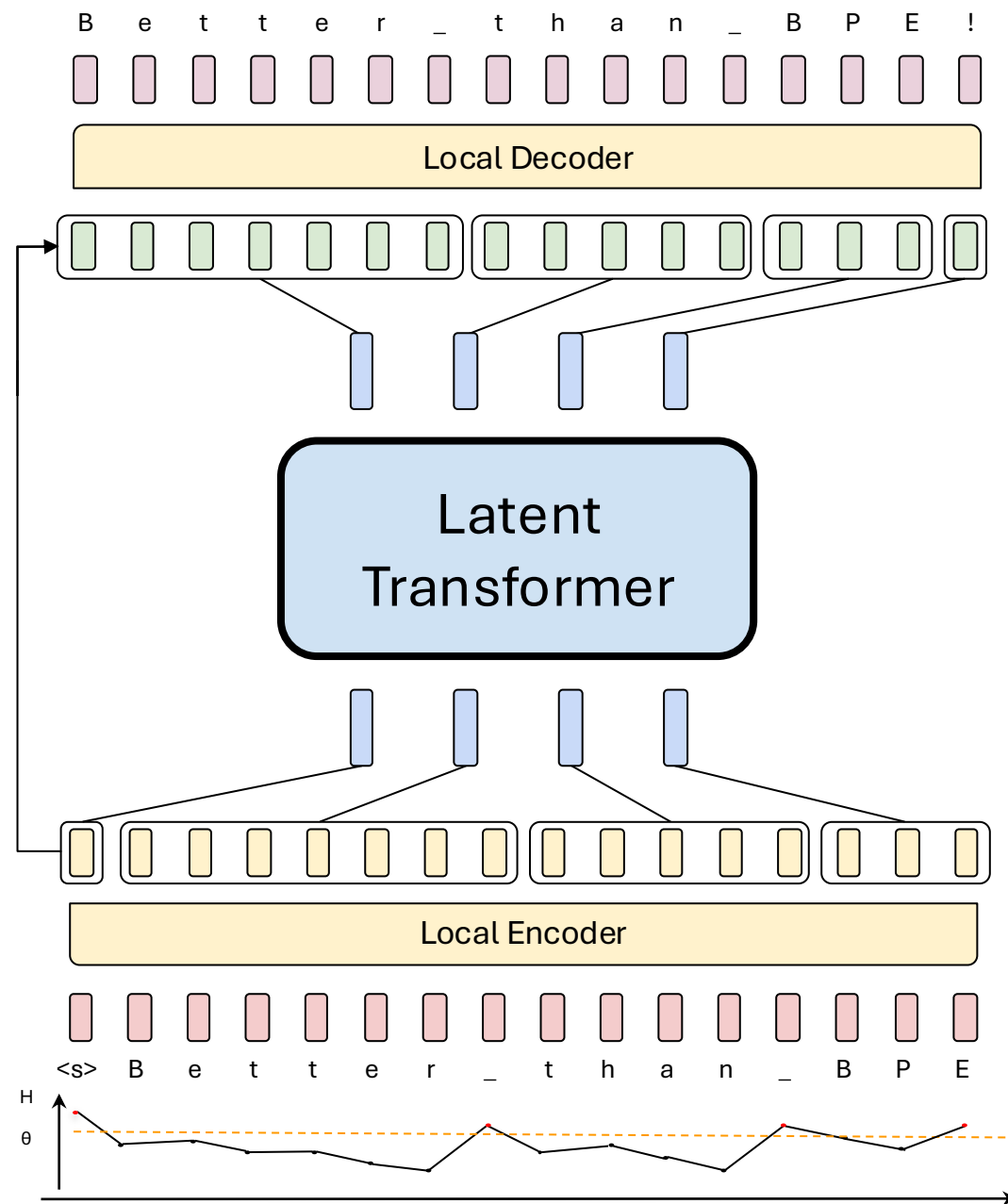
Yu, Lili, et al. "Megabyte: Predicting million-byte sequences with multiscale transformers." *Advances in Neural Information Processing Systems* 36 (2023)



Our Approach (Pagnoni et al, 2024)

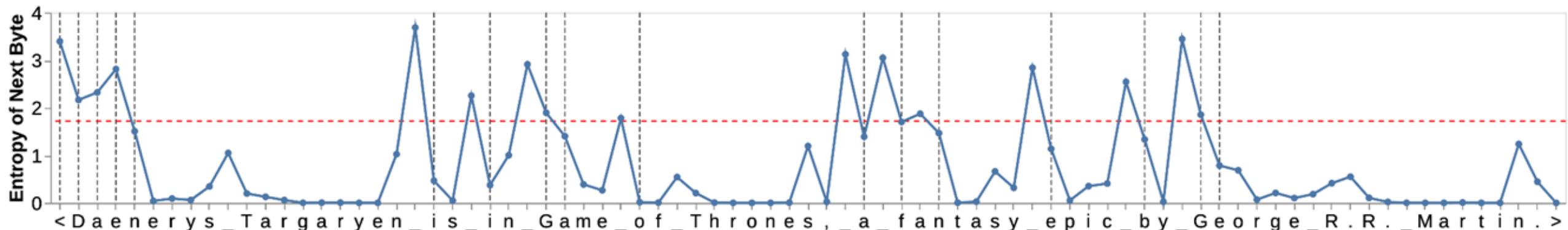
BLT: byte-latent transformer

- Dynamic patching
- Pick boundaries based on byte-level entropy
- New attention mechanisms for information flow



Segmentation: Entropy Patching

Idea: Group predictable bytes following every hard choice

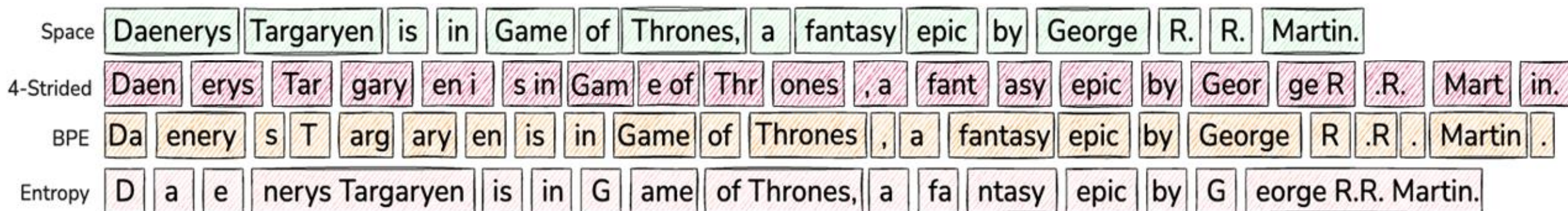


Entropy for the next byte:

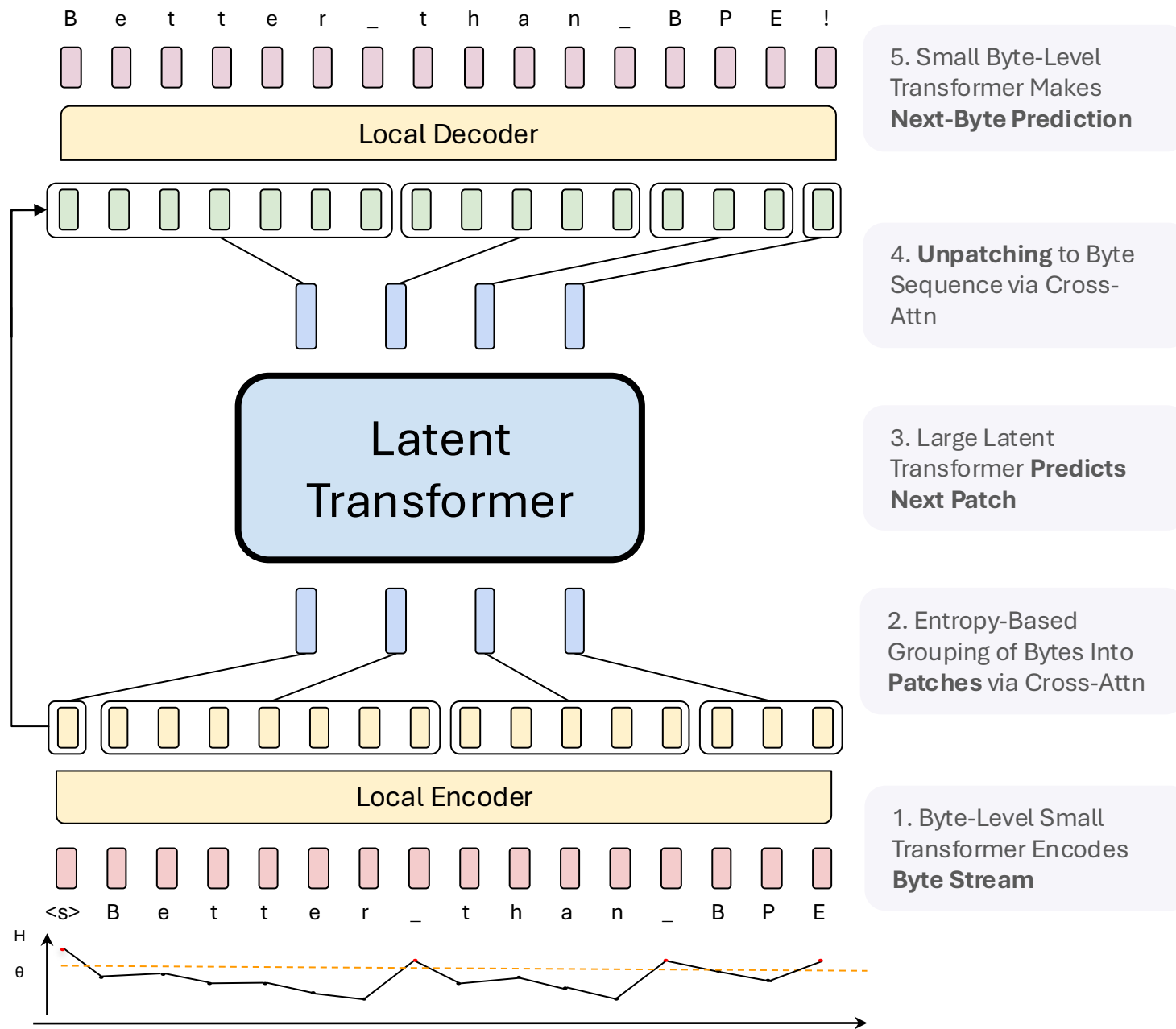
$$H(x_i) = - \sum_{v \in \mathcal{V}} p_e(x_i = v | \mathbf{x}_{<i}) \log p_e(x_i = v | \mathbf{x}_{<i})$$

Threshold to achieve average **Patch Size**

Patching Examples

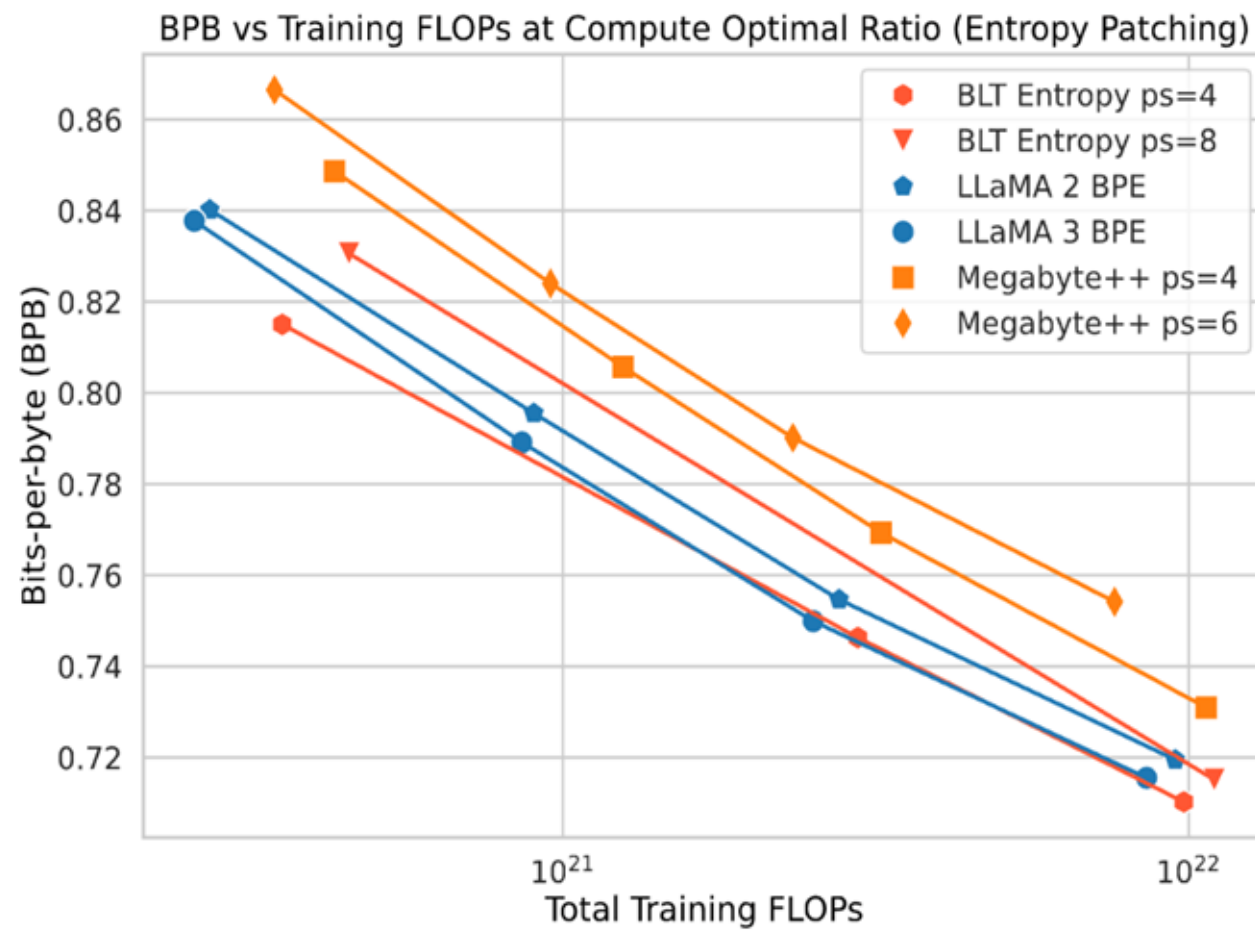


Byte Latent Transformer: Information Flow



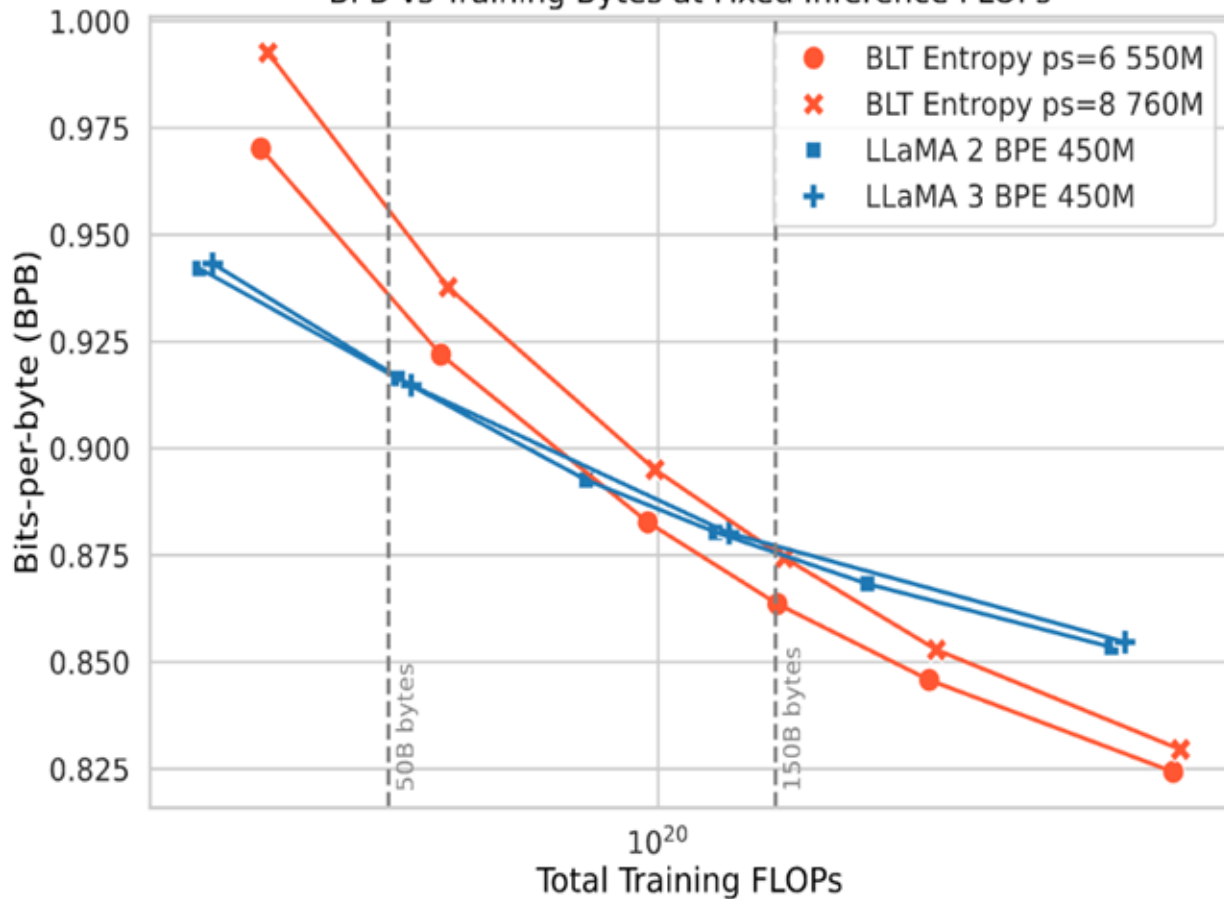
Compute Optimal Scaling Trends

1. BLT architecture improves over previous byte LMs
2. BLT ps=4 **matches BPE** compute optimal scaling trends
3. BLT ps=8 ~ Llama 2 but **2x inference speedup**

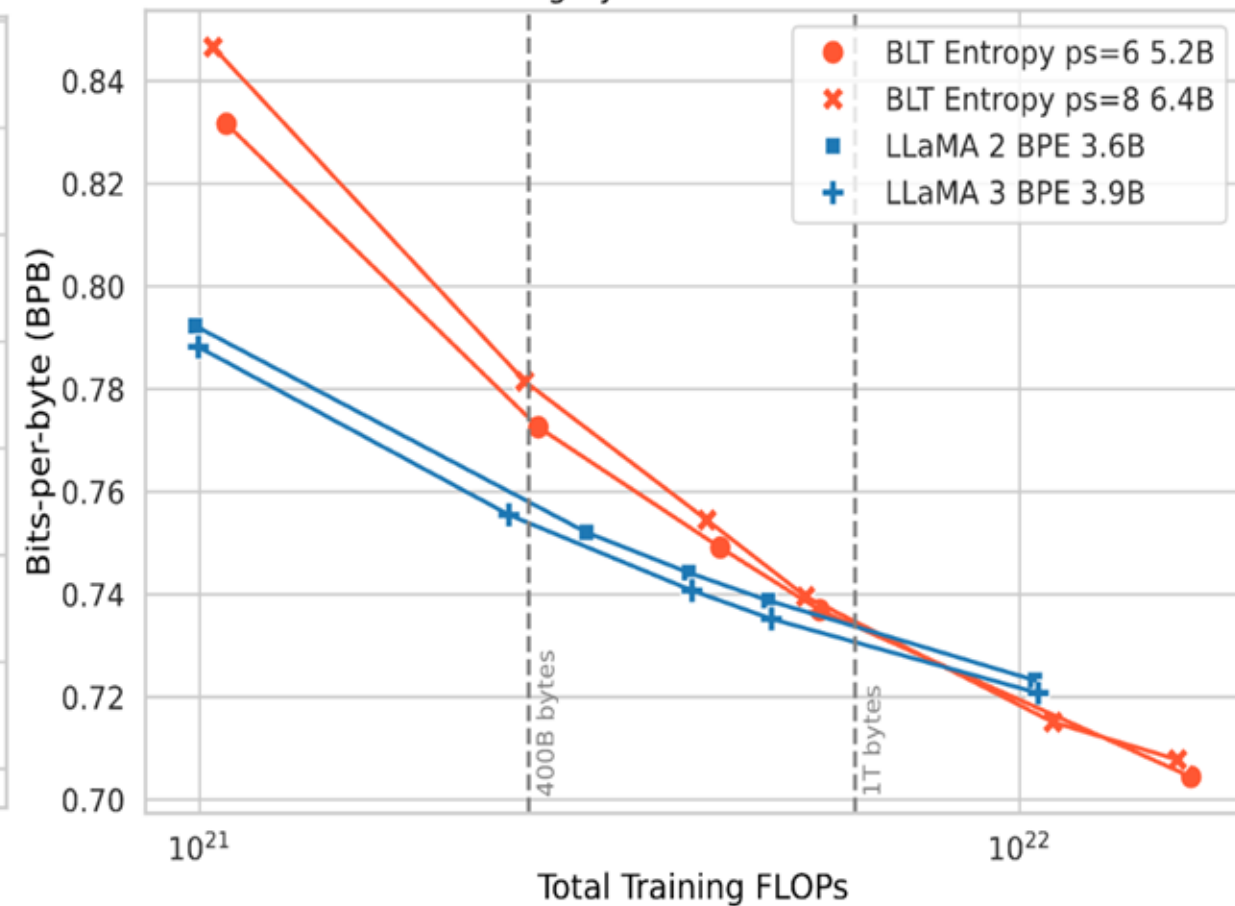


Fixed Inference Scaling

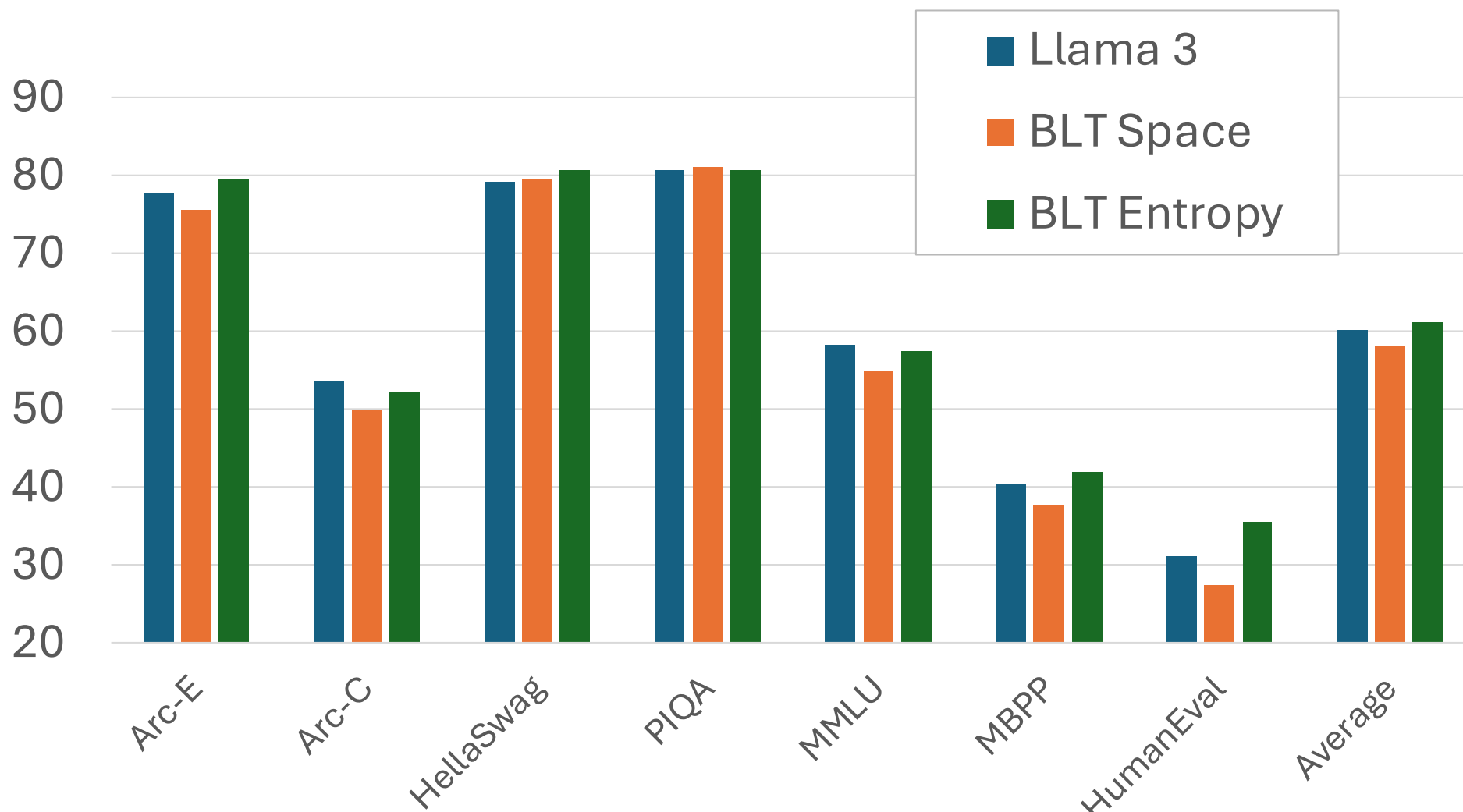
BPB vs Training Bytes at Fixed Inference FLOPs



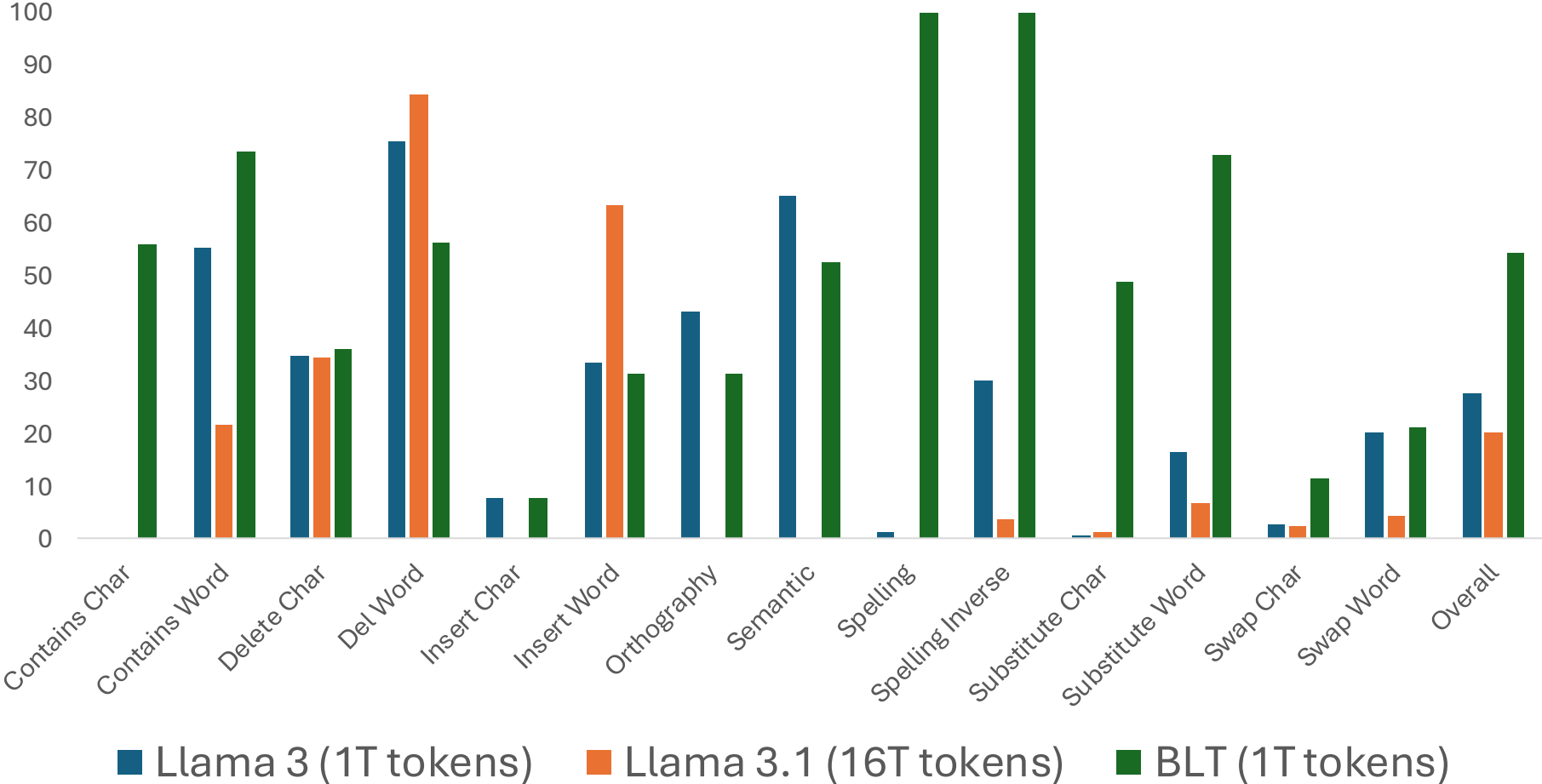
BPB vs Training Bytes at Fixed Inference FLOPs



Beyond Compute Optimal (8B params; 1T tokens)



CUTE: Robustness to Perturbations



Tokenization as Compression

How to pick your LM vocabulary?

1. Language is Zipfian, need fewer words
2. Tradeoff: smaller vocab = more Transformer steps

BPE tokenization is standard (but vocab sizes vary)

Open Questions: How would you do compute optimal tokenization?

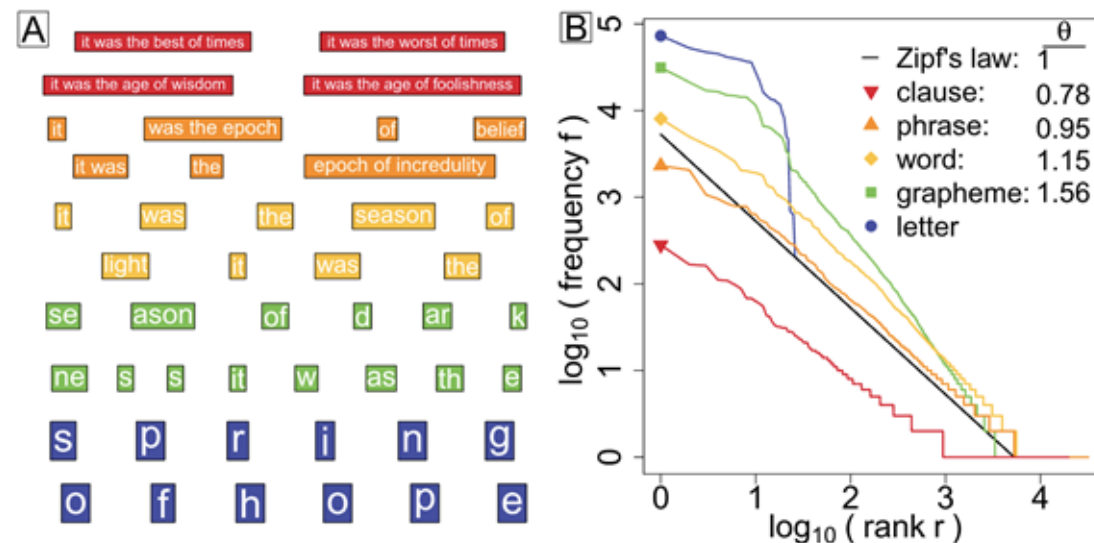


Figure 1. A. Partition examples for the start of Charles Dickens's "Tale of Two Cities" at five distinct levels: clauses (red), pure random partitioning phrases ($q = \frac{1}{2}$, orange), words (yellow), pure random partitioning graphemes ($q = \frac{1}{2}$, green), and letters (blue). The specific phrases and graphemes shown are for one realization of pure random partitioning. B. Zipf distributions for the five kinds of partitions along with estimates of the Zipf exponent θ when scaling is observed. No robust scaling is observed at the letter scale. The colors match those used in panel A, and the symbols at the start of each distribution are intended to strengthen the connection to the legend. See Ref. 28 and the Supplementary Information for measurement details.

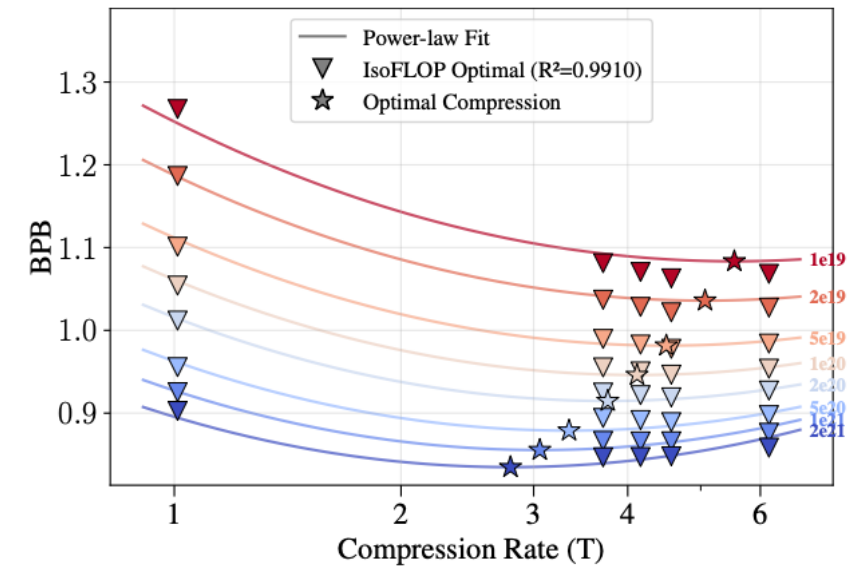
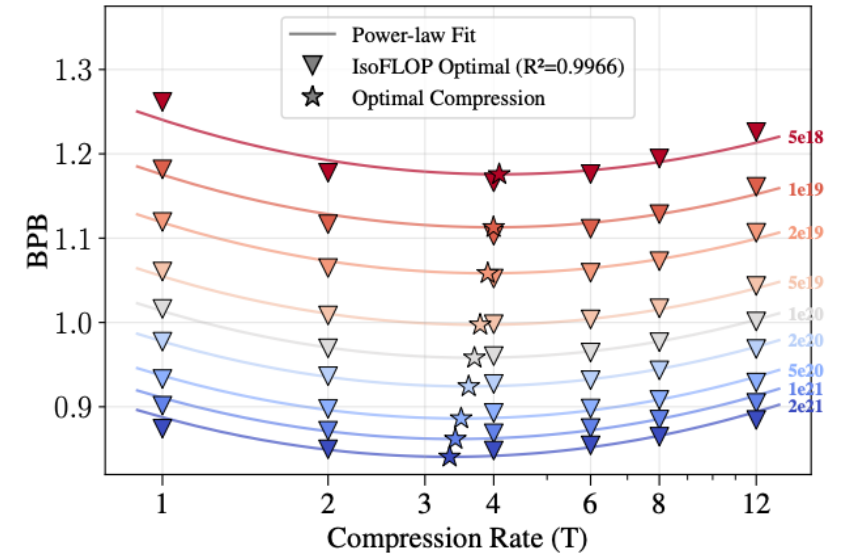
Ryland Williams, Jake, et al. "Zipf's law holds for phrases, not words." *Scientific reports* 5.1 (2015)

Compute Optimal Tokenization

Scaling law study varying:

- Vary compression rate and compute budget
- Both for BLT (*top*) and BPE (*bottom*)
- Fit scaling laws, find optimal compression

See decreasing optimal compression rate for higher training budgets

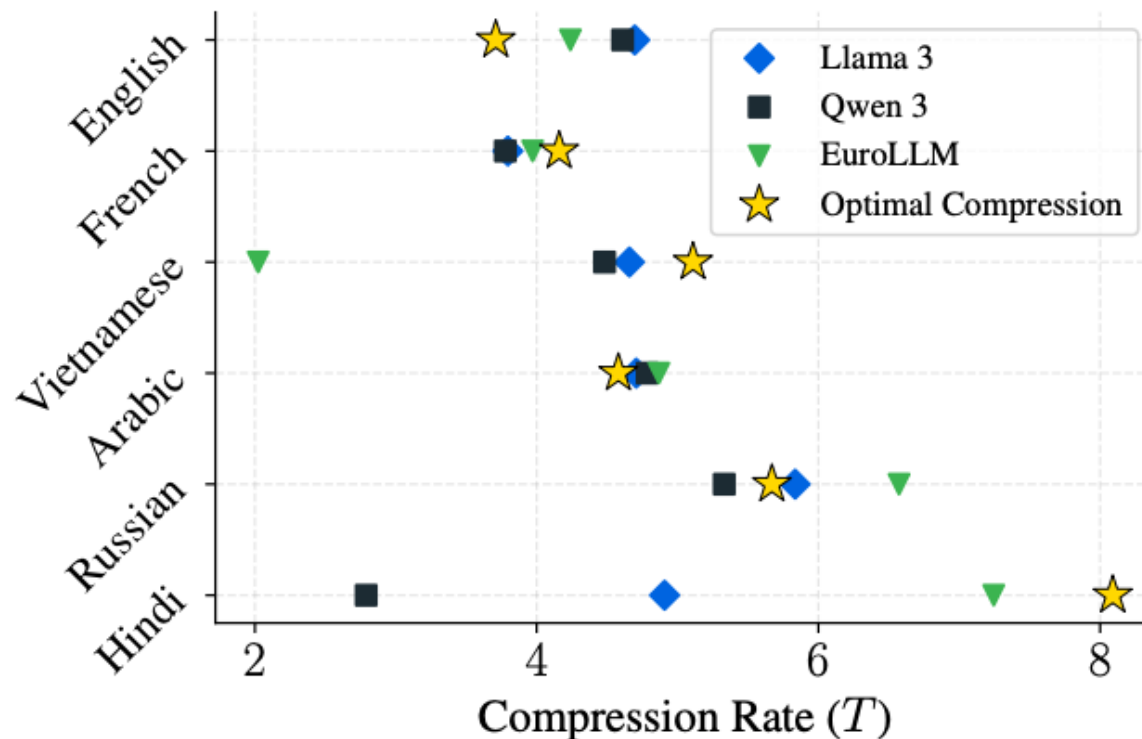


(Limisiewicz, et al 2026)

Compute Optimal Tokenization

Multilingual case:

- Fit separately for different languages
- **Optimal compression rate is language specific**
- **Most existing models are not optimal**



(Limisiewicz, et al 2026)

BLT Summary

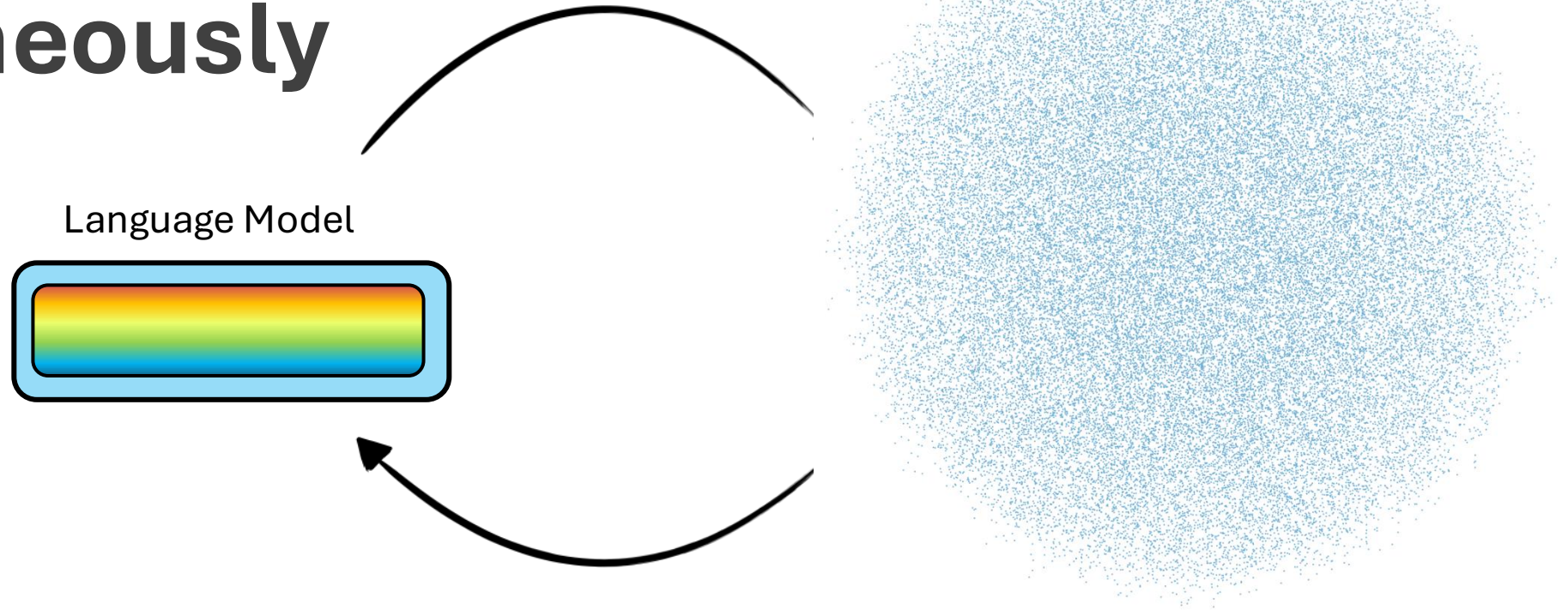
- Hierarchical byte level model with no tokenization
- Scales better, more compute efficient
- Better models long tail of the data
- Almost, but not quite fully, end-to-end training
- Can do more science, e.g. compute optimal tokenization study

Code and Models: <https://github.com/facebookresearch/blt>

Data and Two Architectures

- ☑ Everything LLMs can do comes from pretraining (almost)
- ☑ Data long tail: tokenizer-free LLMs
 - Data modular models: reconfigurable at test time

Language models
treat data
homogeneously



Dense training: update all parameters on all data

Critical issues with dense training



It harms **generalization** by emphasizing domains based on their **prevalence**.
Whose Language Counts as High Quality?
Gururangan et. al, 2022



It reduces **efficiency** by requiring **synchronous** computation.
Lo-fi
Wortsman, Gururangan et. al, 2022



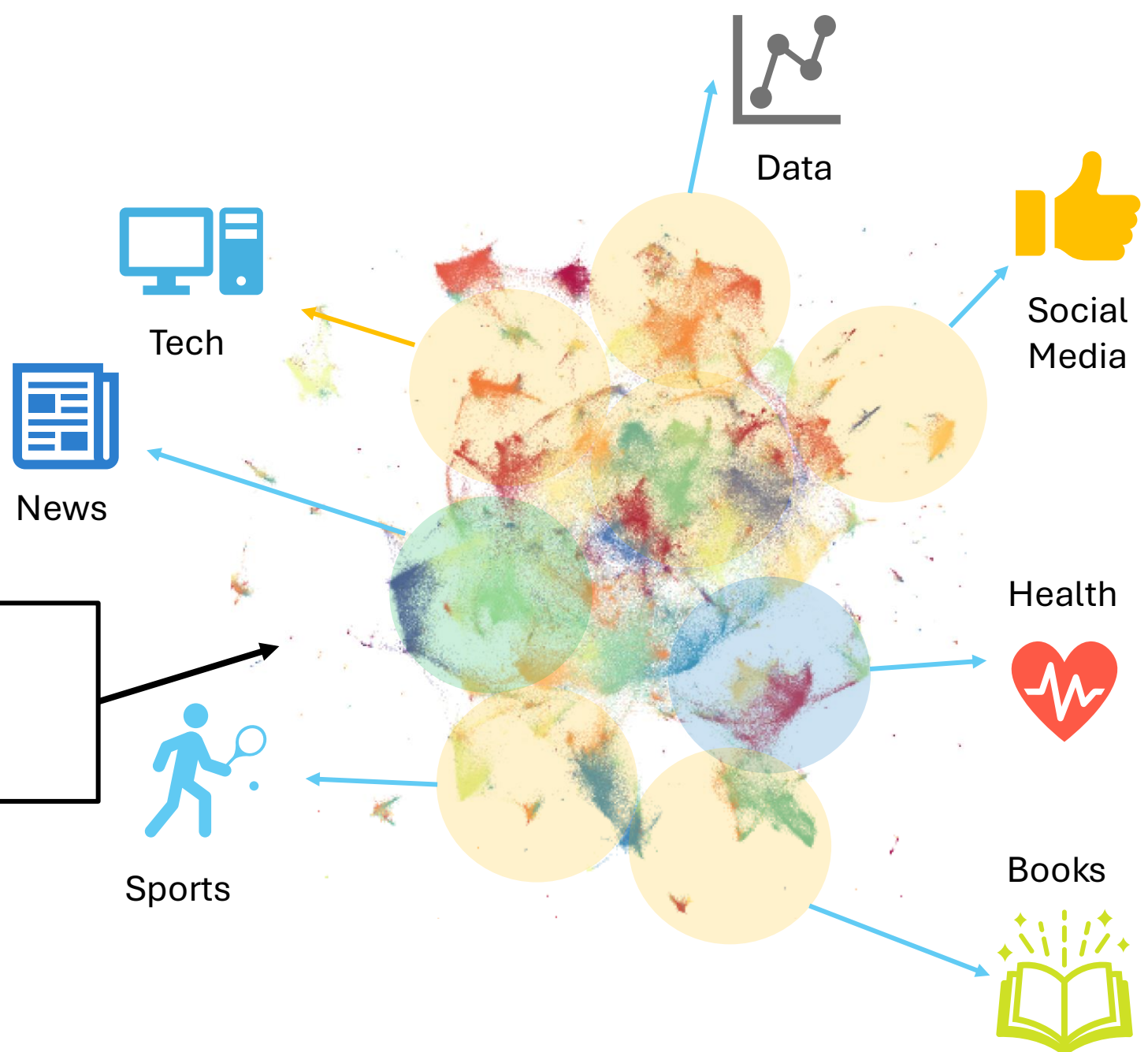
It reduces **flexibility** by making LMs susceptible to **catastrophic forgetting**.
Don't Stop Pretraining
Gururangan et. al, 2020



It increases **risk** because we can't "remove" unwanted domains at test time.
RealToxicityPrompts
Gehman, Gururangan et. al, 2020

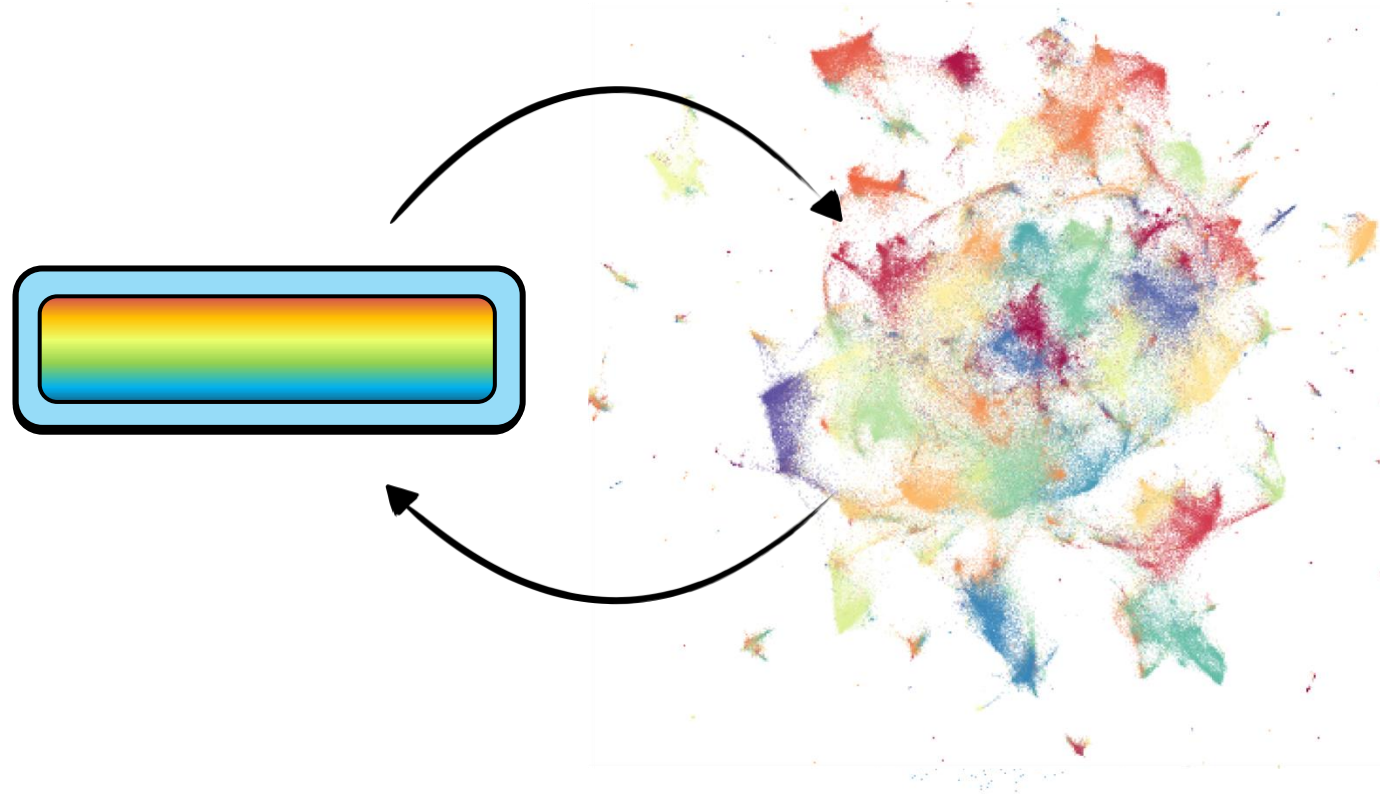
Web Crawl

- 400K documents of Common Crawl
- Embedded with tf-idf
- Clustered with k-means
- Visualized with UMAP

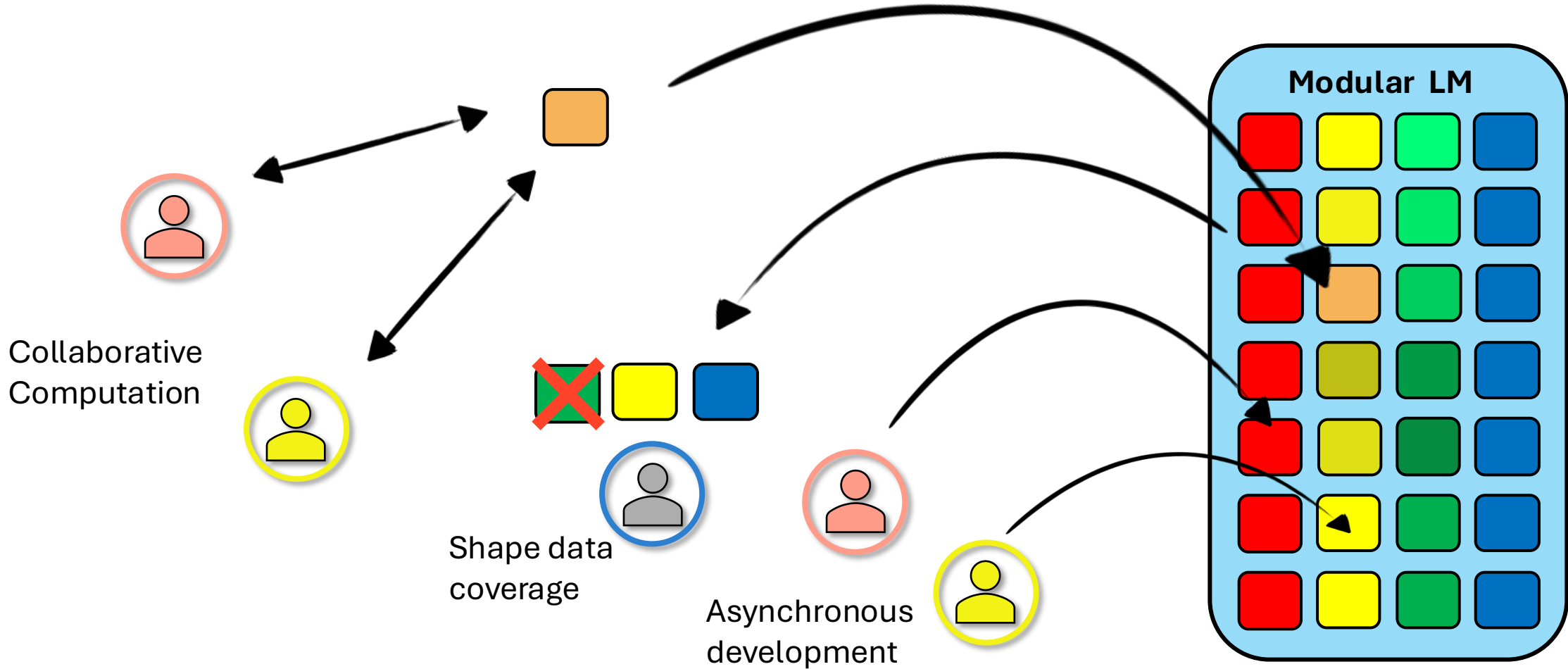


Modular Language Models

- **Specialize** different parts of the model (**experts**) to different parts of the data (**domains**)
- **Mix, add, or remove** experts as necessary
- Emphasizes **flexibility** and **customization** at test time



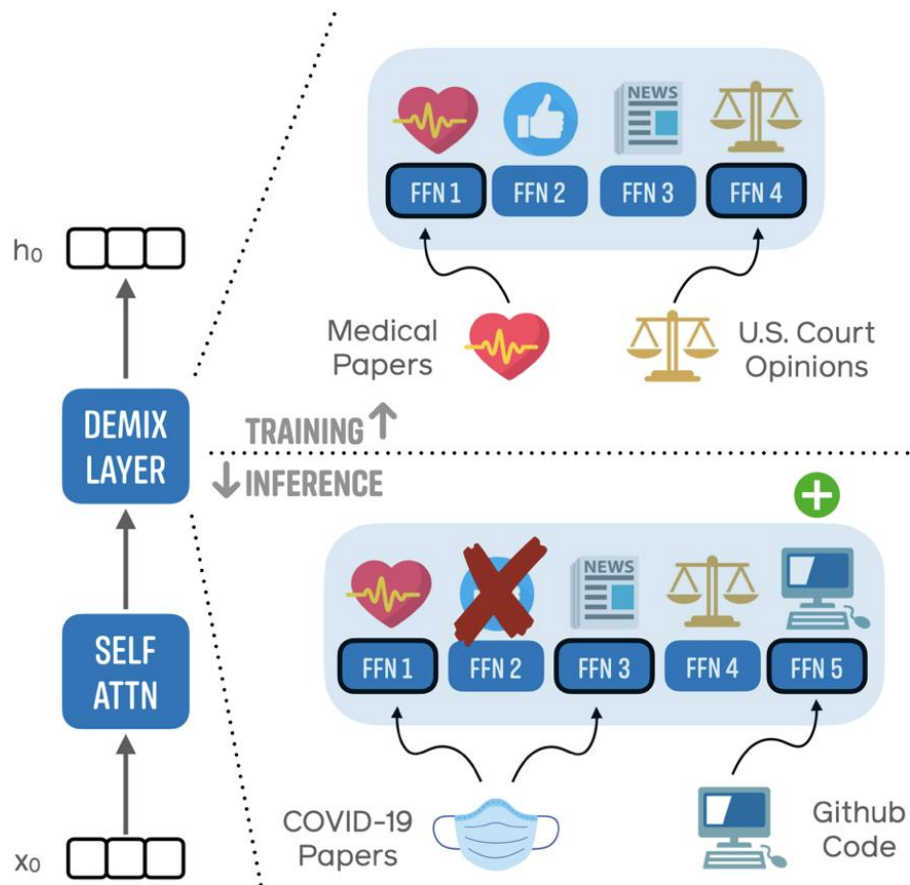
A Decentralized Future



STEPS WE TAKE
TOWARDS THIS
GOAL

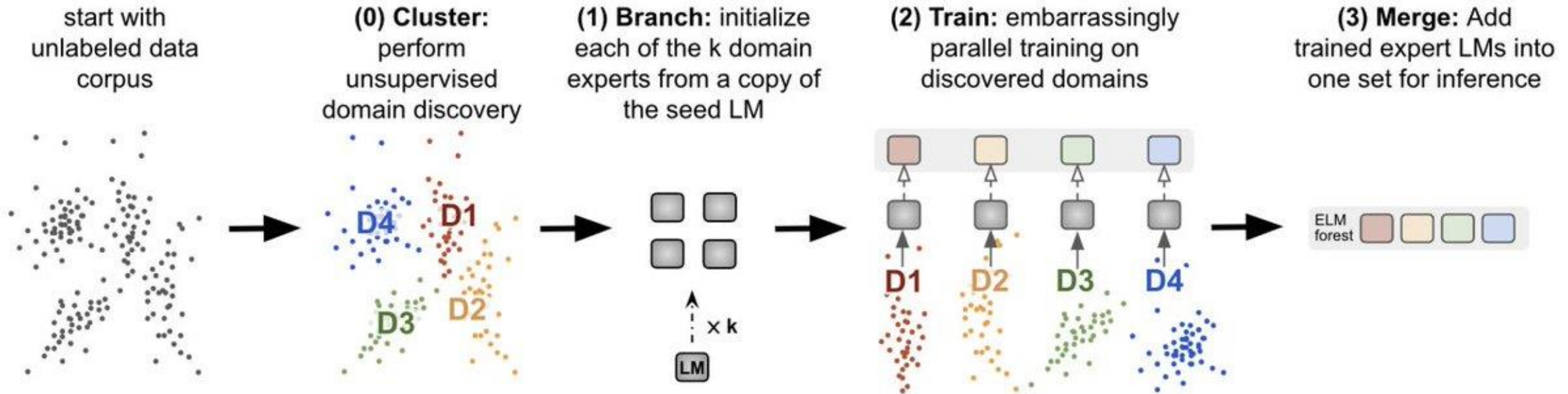
- **Modular:** Incorporate experts that can be cheaply customized **after training**.
- **Asynchronous:** Train and update different experts **independently** at arbitrary **scale**.
- **Sparse:** Despite training many experts, use only a **few experts** at inference.

DEMix Layers: Conditional Compute for Modular Models (Gururangan et. al, 2021)



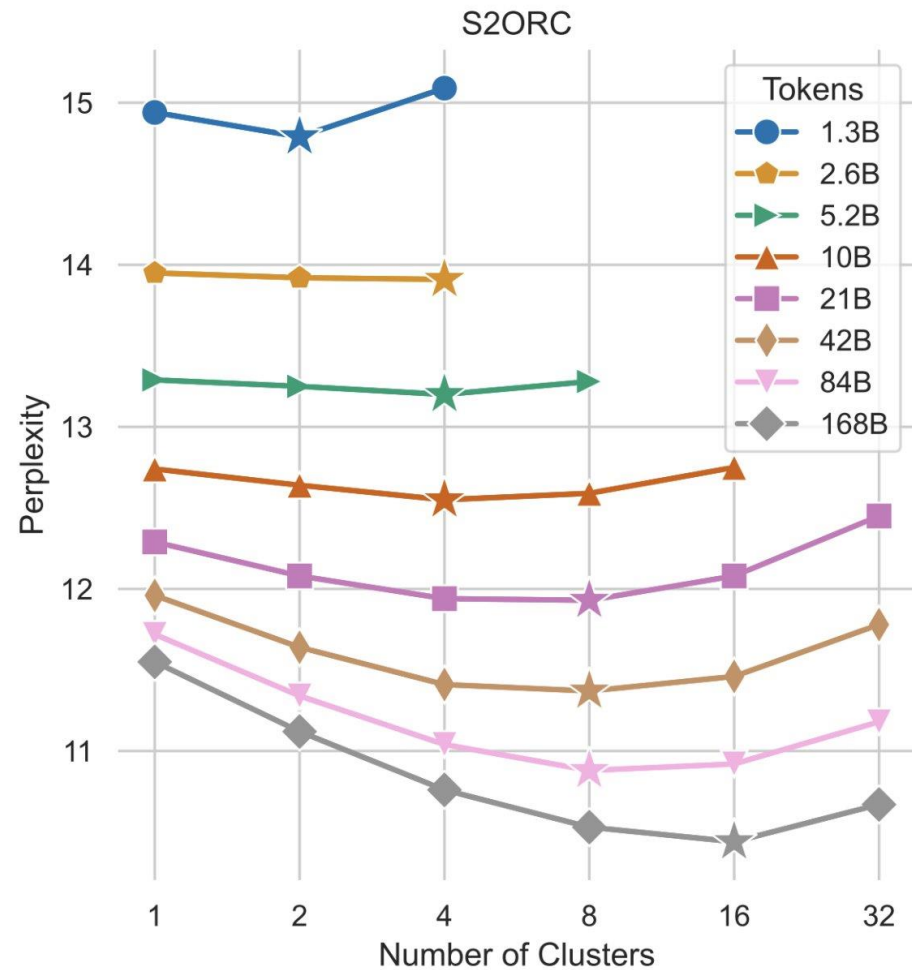
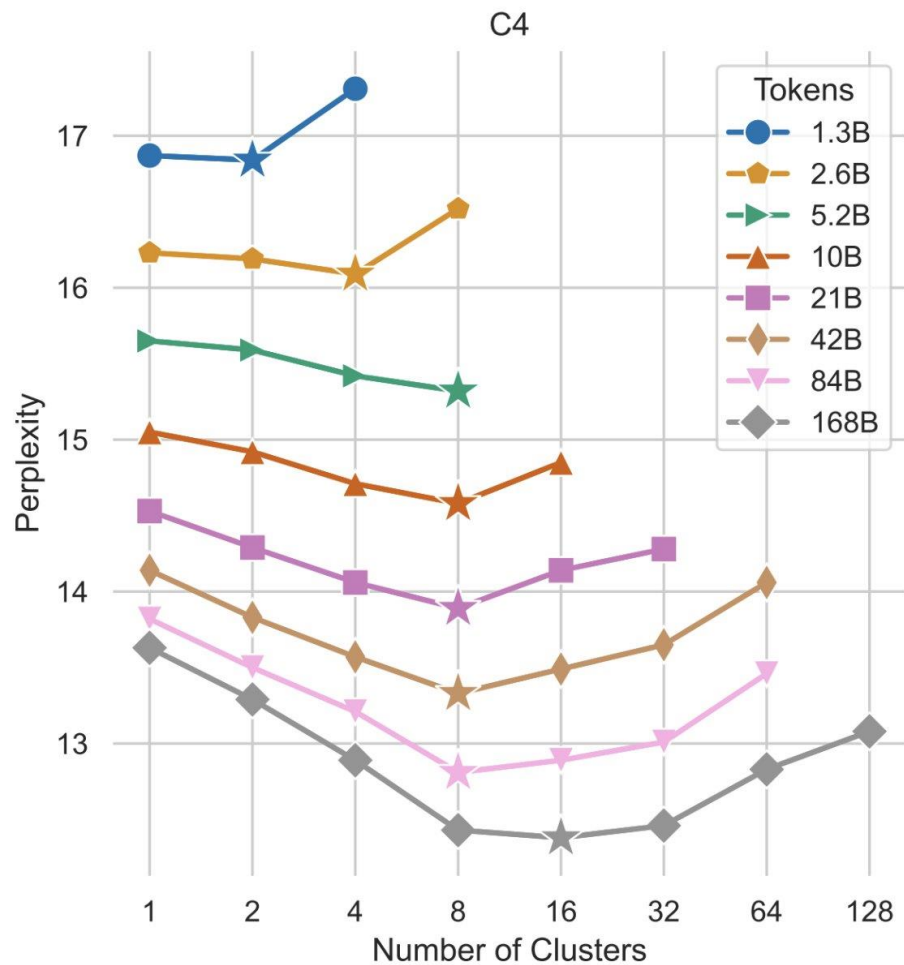
- Different FFNs for each domain
- Deterministic routing
- Modular: mix, add or remove experts as you want, due to domain specialization

Branch Train Merge (Li et al 2022; Gururangan et al 2023)



1. Cluster data as preprocessing step
2. Train LMs on different subsets of data with common initialization
3. Ensemble / average the models to form new, general models

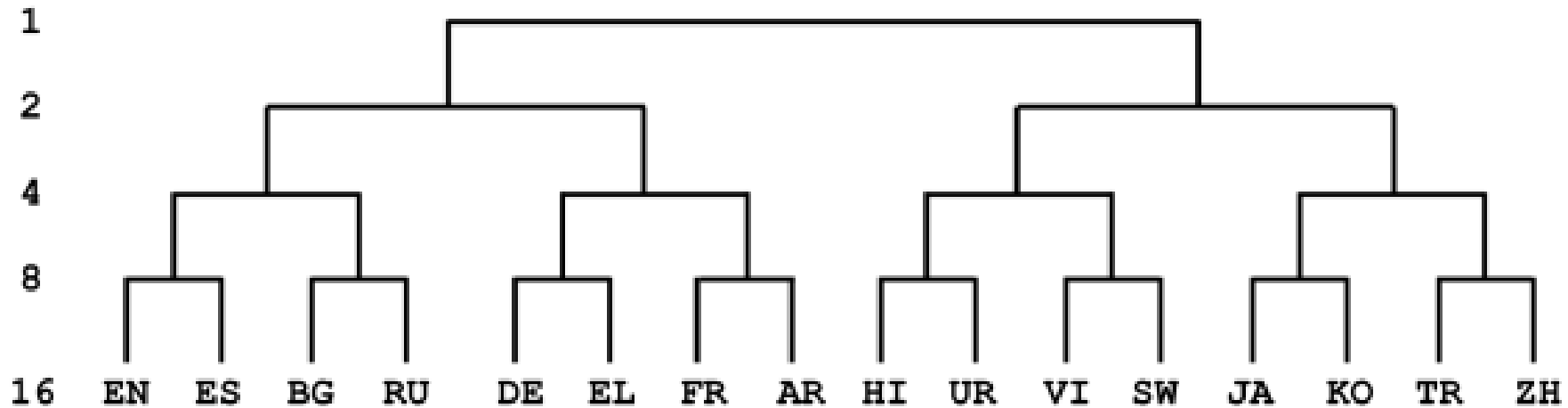
Branch Train Merge (Li et al 2022; Gururangan et al 2023)



Also works for multilingual (Blevins et al 2024)

- Can group by language instead of domain
- Can organize data hierarchically and pool by similarly, e.g. related language families

k =



Challenge: how do we do this for
modern MoE architectures

MoE is the dominant (modular) architecture (Shazeer et al, 2017)

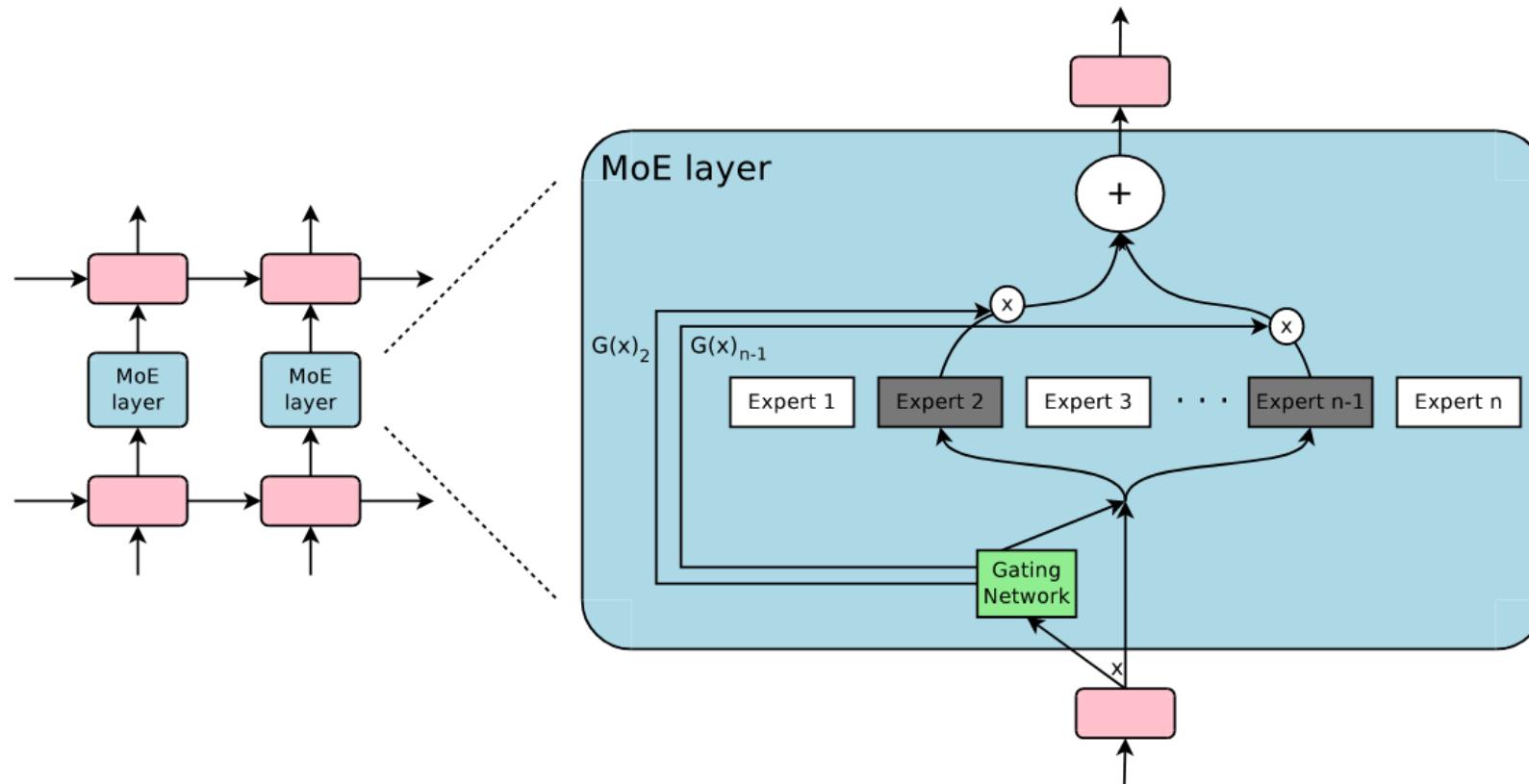


Figure 1: A Mixture of Experts (MoE) layer embedded within a recurrent language model. In this case, the sparse gating function selects two experts to perform computations. Their outputs are modulated by the outputs of the gating network.

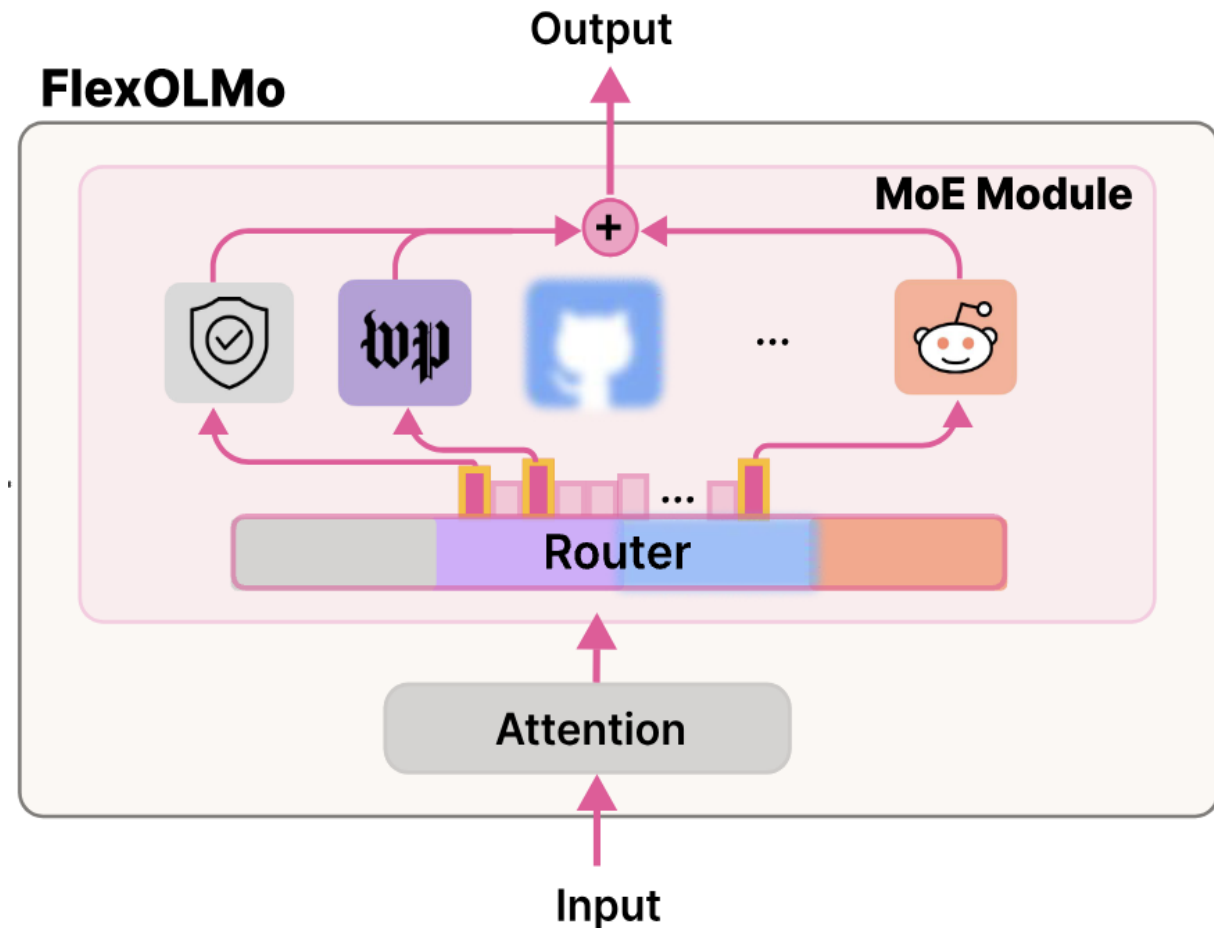
FlexOLMo (Shi et al, 2025)



Experts independently trained on different sources

Key Challenges:

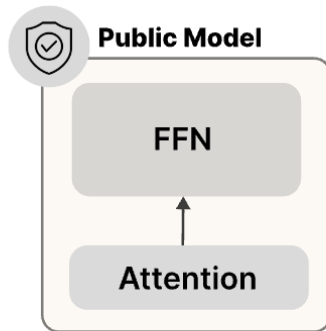
- How to train independently?
- How to efficiently recombine?
- Don't expose any private expert data.



<https://allenai.org/blog/flexolmo>

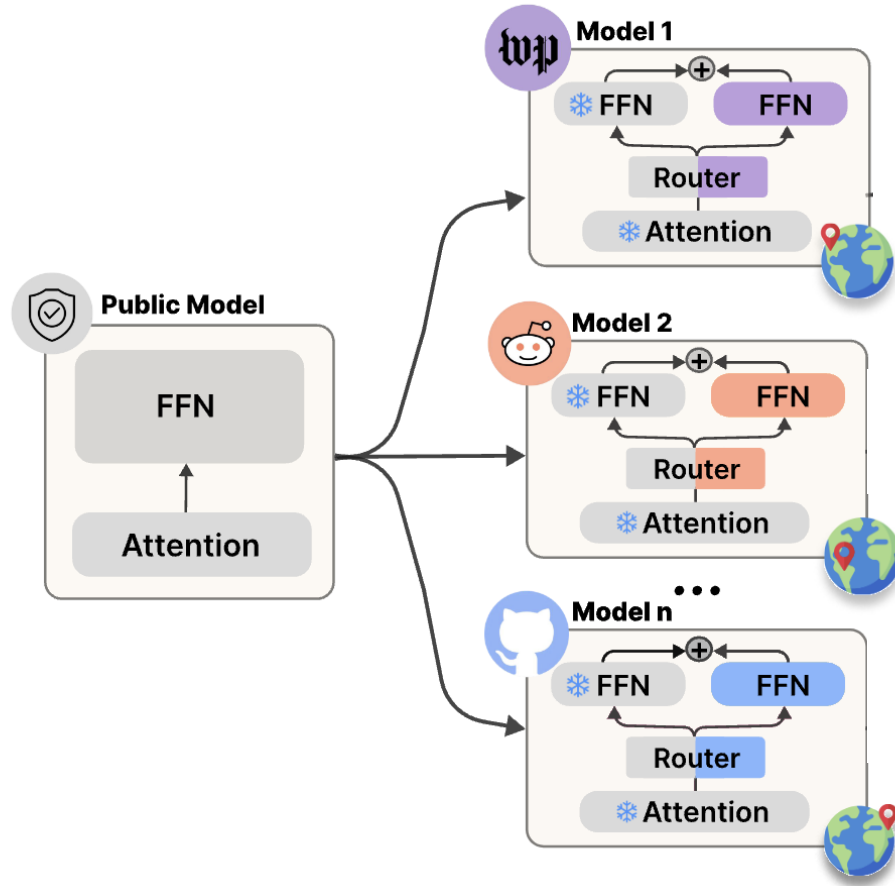
Training Experts to Coordinate

Anchor on a single expert that they all share



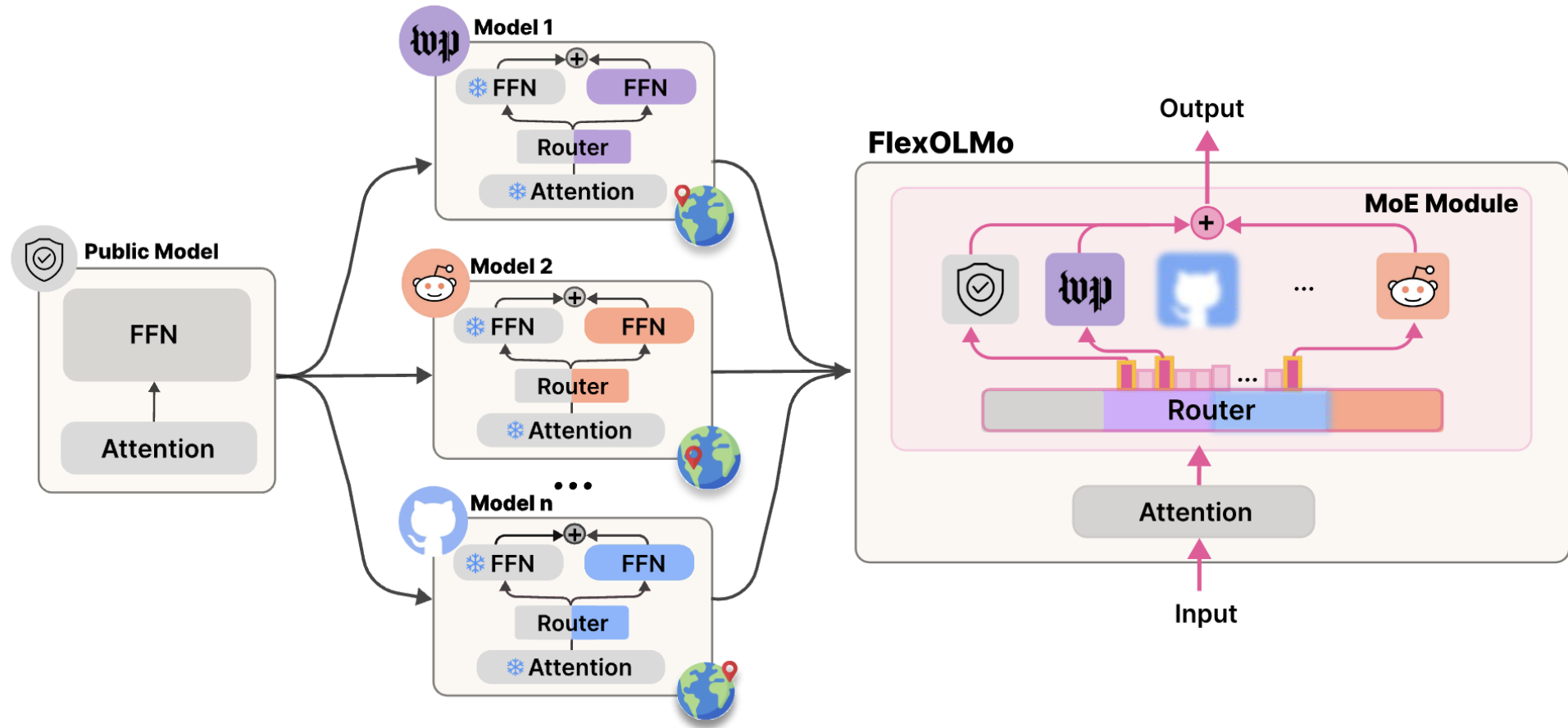
Training Experts to Coordinate

Anchor on a single expert that they all share



Training Experts to Coordinate

Anchor on a single expert that they all share



How to recombine?

Goal 1: independently **train** experts on disjoint (potentially private) data

Goal 2: add and remove experts during **inference** with no further training

Approach:

- Introduce a new non-parametric router
- Add expert biases to loss to improve generalization

Domain-informed nonparametric router

Router

$$r(\mathbf{x}) = \mathbf{W}_r \mathbf{x}, \quad \mathbf{W}_r \in \mathbb{R}^{(n+1) \times h}$$

In standard MoEs, router W is trained end-to-end alongside all expert modules

But we decompose W into expert-specific router embeddings:

$$\mathbf{W}_r = \begin{bmatrix} \text{---} & \mathbf{r}_{\text{pub}} & \text{---} \\ \text{---} & \mathbf{r}_1 & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{r}_n & \text{---} \end{bmatrix}, \text{ where } \mathbf{r}_i = \frac{1}{|S_i|} \sum_{d_k \in S_i} \mathbf{E}(d_k) \in \mathbb{R}^h$$

↑
Embedder

Routing as a classification problem

Training: n pairwise binary classifiers

$$C_{pub} \text{ vs. } C_1$$

$$C_{pub} \text{ vs. } C_2$$

...

$$C_{pub} \text{ vs. } C_n$$

Inference: a multi-class classifier

$$C_{pub}, C_1, C_2, \dots, C_n$$

Routing as a classification problem

Training: n pairwise binary classifiers

Inference: a multi-class classifier

~~C_{pub}~~ vs. ~~C_1~~ C_1 vs. not C_1

$C_{pub}, C_1, C_2, \dots, C_n$

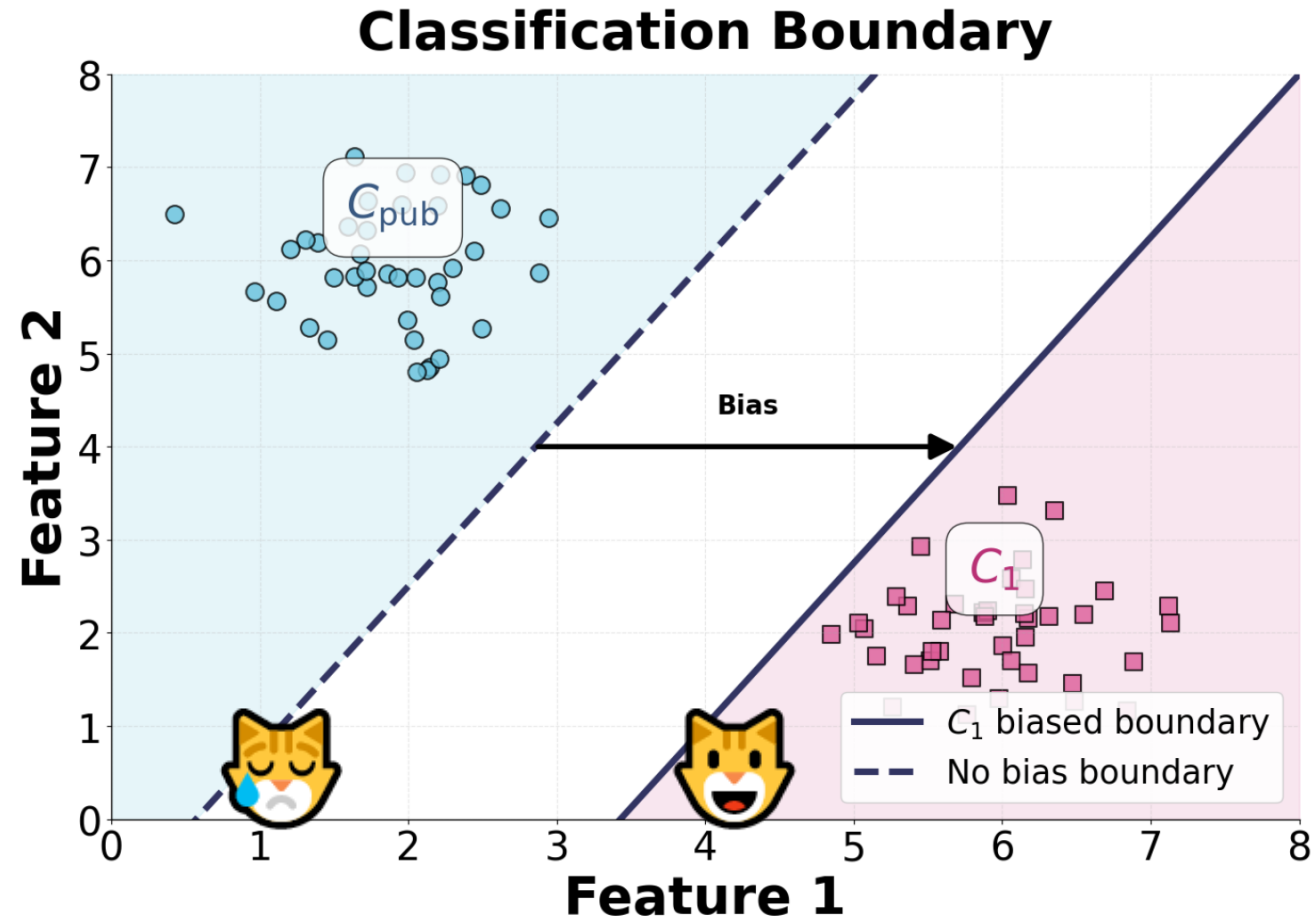
~~C_{pub}~~ vs. ~~C_2~~ C_2 vs. not C_2

...

~~C_{pub}~~ vs. ~~C_n~~ C_n vs. not C_n

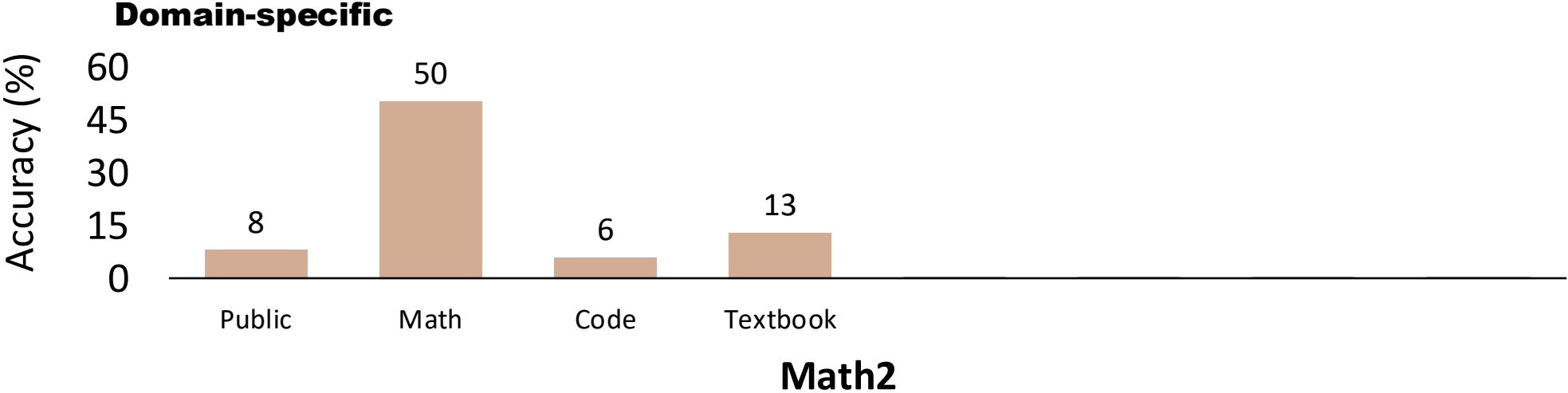
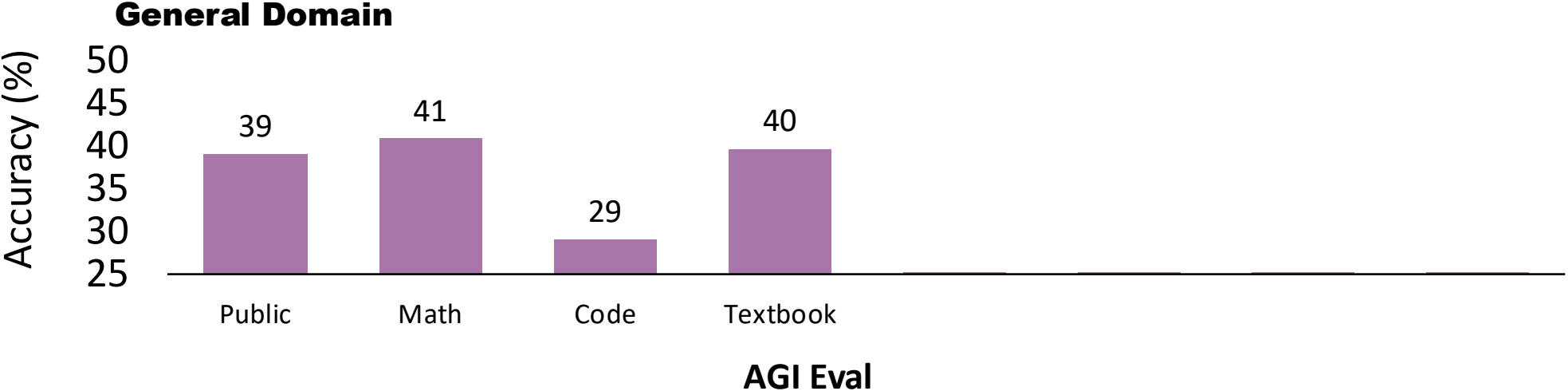
Key Idea: push decision boundary towards the classes C_i as much as possible

Expert bias

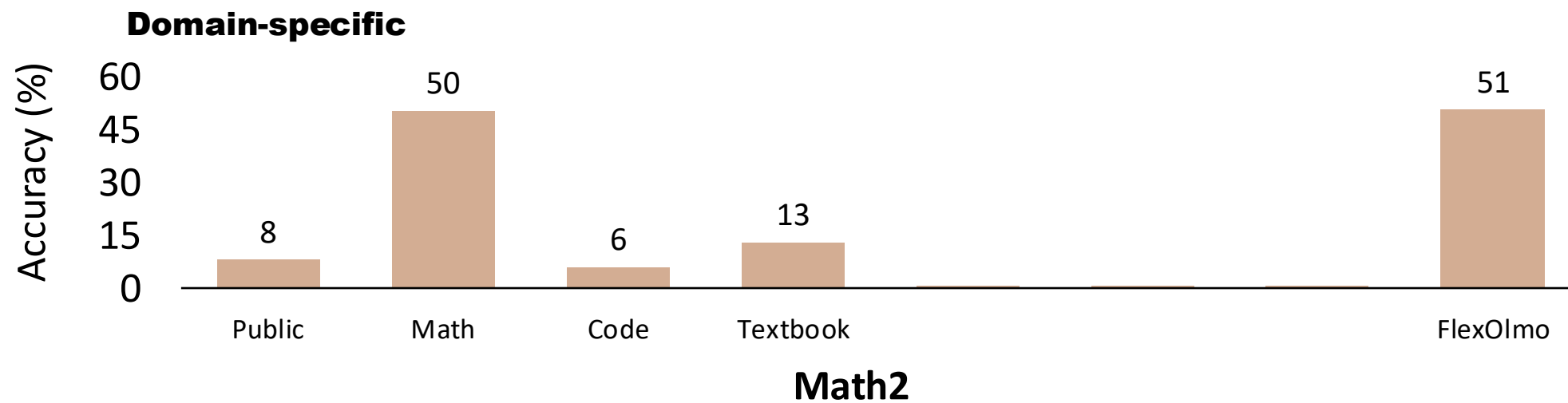
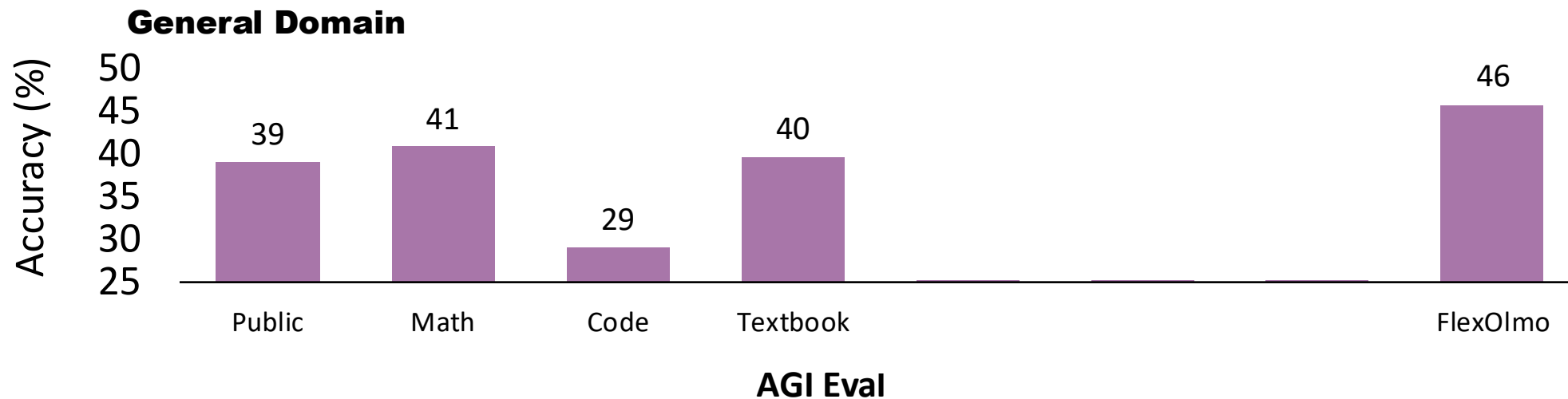


Learned bias pushes decision boundary closer to C_1

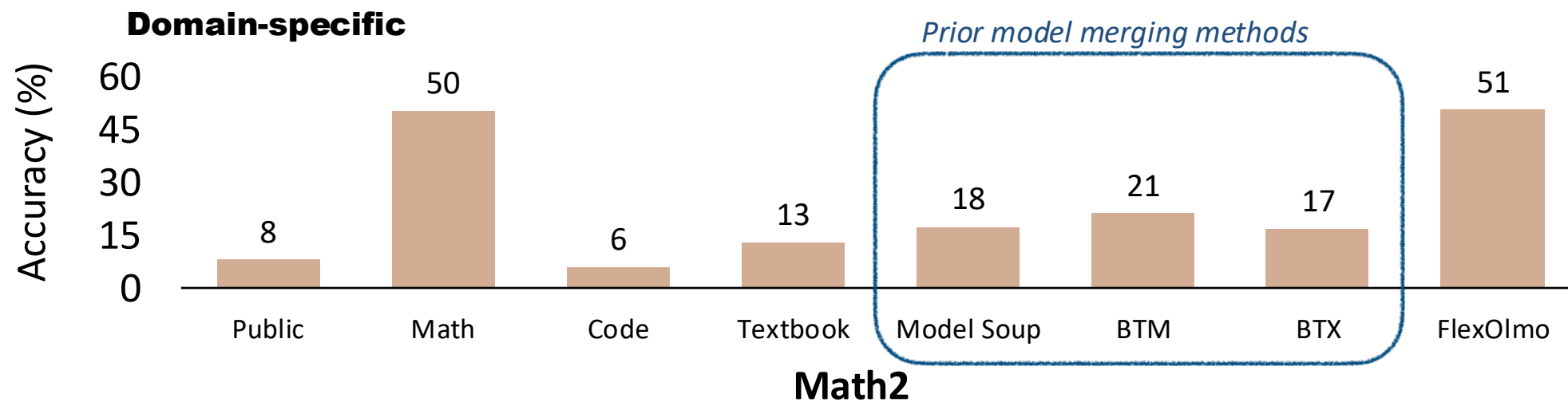
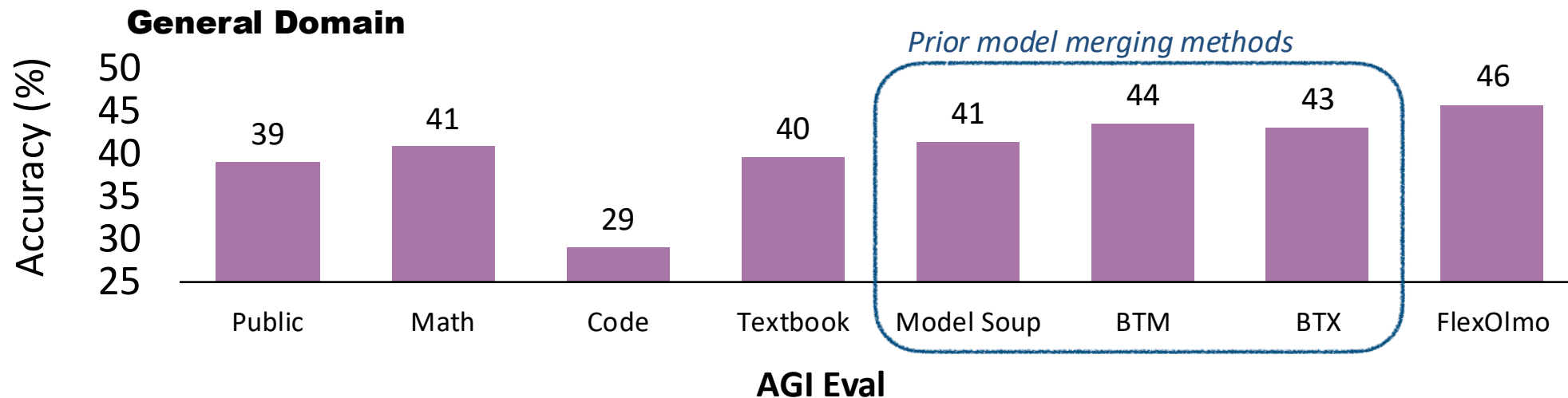
Individual Experts Excel at their own tasks



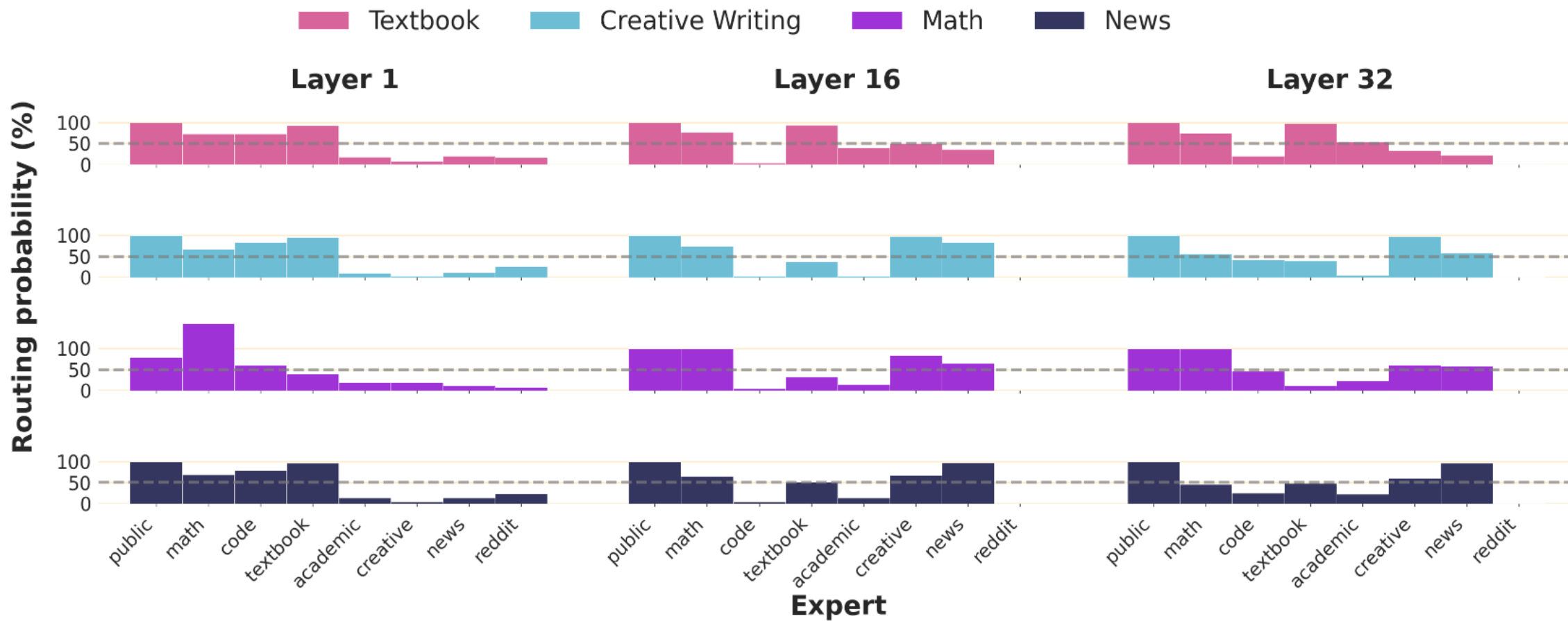
FlexOLMo outperforms individual experts



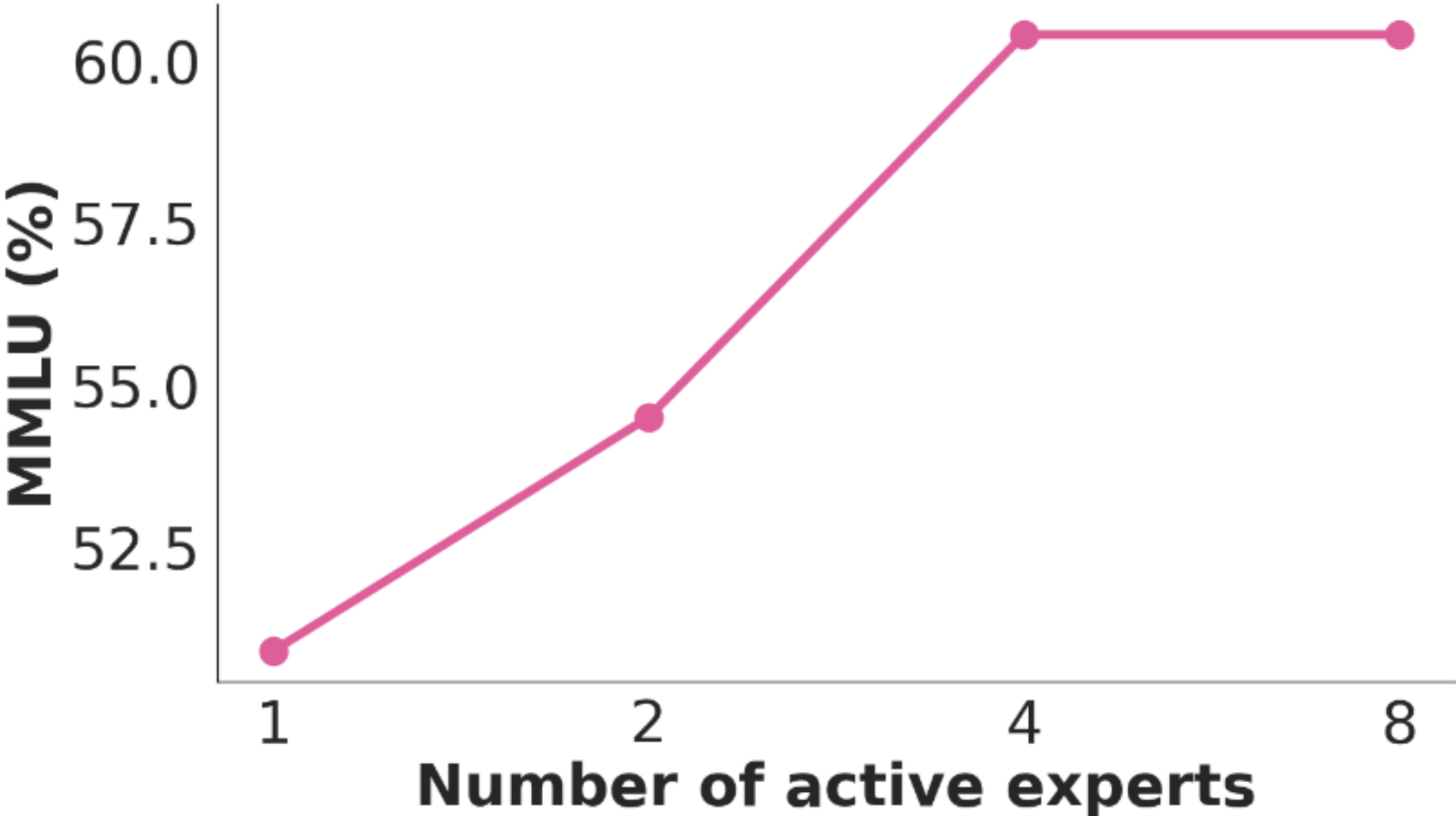
FlexOLMo achieves better merging than baselines



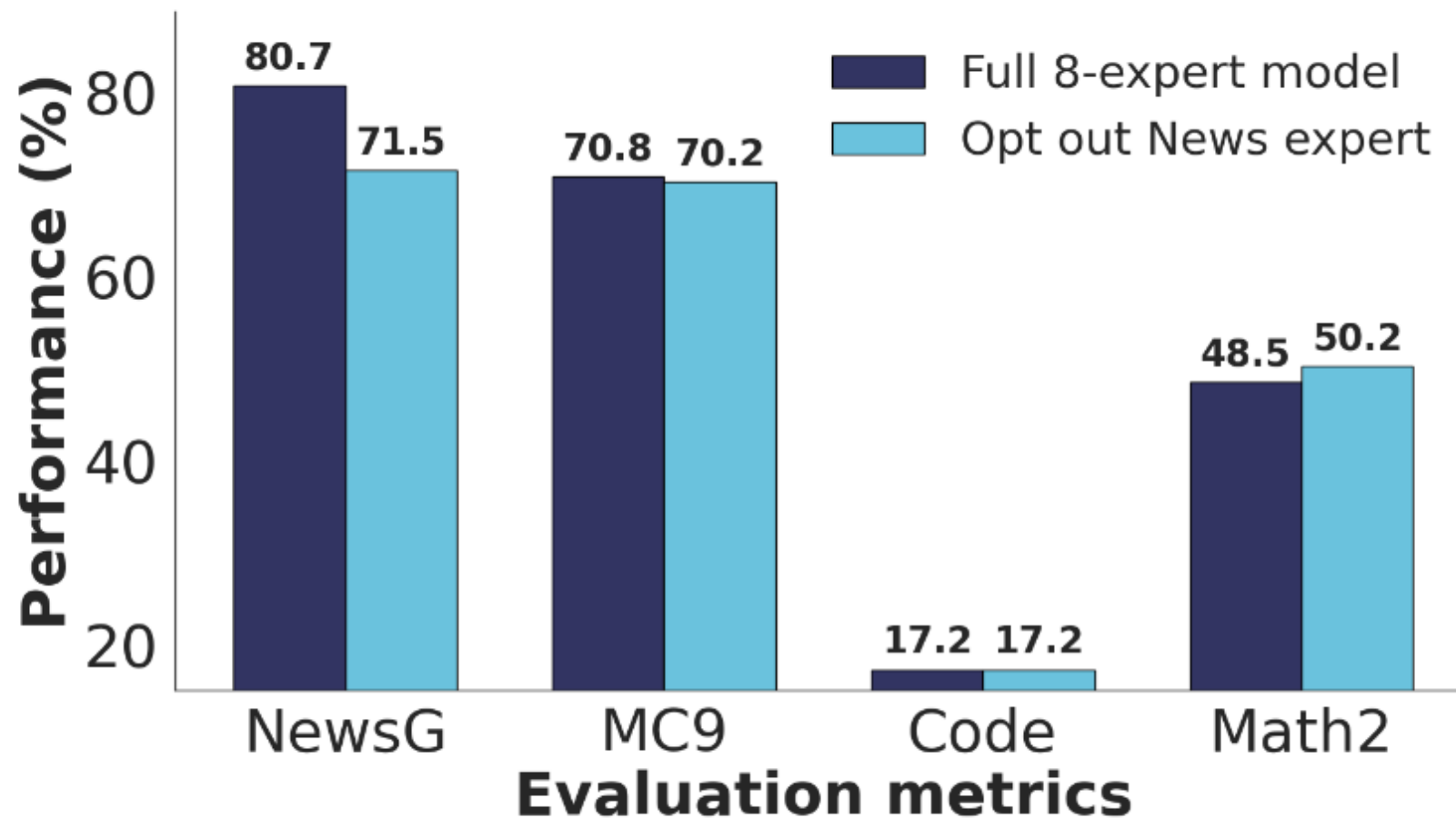
Routing analysis



Routing
analysis:
more
experts
helps



Removing
News
Expert
hurts News
eval, but
not others



FlexOLMo Summary

- Modular, distributed model training with independent (potentially private) data
- New routing scheme that allows freely adding and removing experts during inference
- Strong performance up to 37B parameters (20B active)



Model

hf.co/allenai/FlexOlm0-7x7B-1T



Code

github.com/allenai/FlexOlm0

Blog

allenai.org/blog/flexolmo

Data and Two Architectures

- ☑ Everything LLMs can do comes from pretraining
- ☑ Lets do end-to-end pretraining: no tokenizer
- ☑ Data modular models: reconfigurable at test time

Open Questions: Pretraining vs Post training

- Does everything really come from pretraining?
- What about large-scale RL*F methods?
 - Amount of new supervision is still $\lll 10T$
 - In the limit of adding more judges/reward models and scaling RL, will eventually it rival/replace pretraining entirely?
- Can we develop new science and tools to study these issues?

Open Questions: New Architectures

- Continue to improve architectures, as mentioned earlier...
- Need to take the next scaling step, see where things break!
- Need to figure out post training for new architecture too! E.g.
 - BLT: can you add segmentation to RLAIIF, learn task specific segmentation?
 - FlexOLMo: how to do distributed SFT and RLAIIF? Can we present strong privacy guaranteed without sacrificing performance?
 - Code and models are open, anyone can study these right now!

Papers

LIMA: Less Is More for Alignment. Chunting Zhou*, Pengfei Liu*, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, Omer Levy (2023)

ALMA: Alignment with Minimal Annotation. Michihiro Yasunaga, Leonid Shamis, Chunting Zhou, Andrew Cohen, Jason Weston, Luke Zettlemoyer, Marjan Ghazvininejad. (2024)

s1: Simple test-time scaling. Niklas Muennighoff*, ZitongYang*, Weijia Shi*, Xiang Lisa Li*, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, Tatsunori Hashimoto. (2025)

Spurious Rewards: Rethinking Training Signals in RLVR. Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, Luke Zettlemoyer. (2025)

MEGABYTE: Predicting Million-byte Sequences with Multiscale Transformers. Lili Yu, Daniel Simig*, Colin Flaherty*, Armen Aghajanyan, Luke Zettlemoyer, Mike Lewis (2023)

Byte Latent Transformer: Patches Scale Better Than Tokens. Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, Srinivasan Iyer. (2024)

Branch-Train-Merge: Embarrassingly Parallel Training of Expert Language Models. Margaret Li*, Suchin Gururangan*, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, Luke Zettlemoyer. (2022)

Scaling Expert Language Models with Unsupervised Domain Discovery. Suchin Gururangan*, Margaret Li*, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, Luke Zettlemoyer (2023)

Breaking the curse of Multilinguality with Unsupervised Domain Discovery. Terra Blevins, Tomasz Limisiewicz, Suchin Gururangan, Margaret Li, Hila Gonen, Noah A Smith, Luke Zettlemoyer (2024)

FlexOLMo: Open Language Models for Flexible Data Use. Weijia Shi*, Akshita Bhagia*, Kevin Farhat*, Niklas Muennighoffas, Pete Walsha, Jacob Morrison, Dustin Schwenka, Shayne Longprem, Jake Poznanskia, Allyson Ettingera, Daogao Li, Margaret Li, Dirk Groeneveld, Mike Lewis, Wen-tau Yih, Luca Soldainia, Kyle Lo, Noah A. Smith, Luke Zettlemoyer, Pang Wei Koh, Hannaneh Hajishirzi, Ali Farhadi, Sewon Min. (2025)

Compute Optimal Tokenization. Tomasz Limisiewicz, Artidoro Pagnoni, Srini Iyer, Mike Lewis, Sachin Mehta, Alisa Liu, Margaret Li, Gargi Ghosh, Luke Zettlemoyer. (2026)

Questions?