

Data as a Binding Resource: How Pre-training Should Change

Sewon Min

sewonmin.com

Berkeley **BAIR**  **Ai2**
UNIVERSITY OF CALIFORNIA

Pre-training

Pre-training

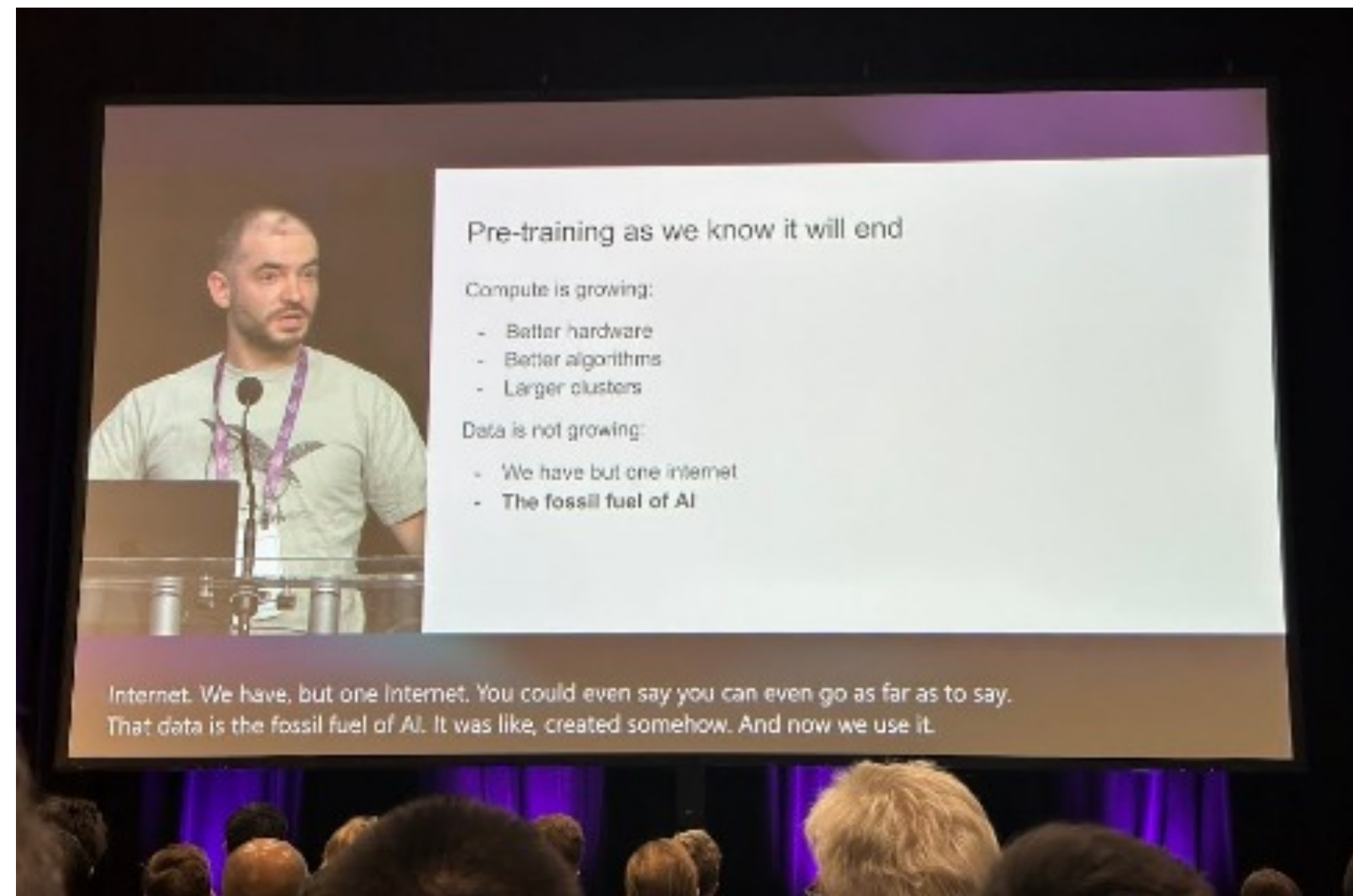
- Previously: Compute is a binding resource; data is nearly infinite

Pre-training

- Previously: Compute is a binding resource; data is nearly infinite
- Now: Compute is getting cheaper, but data is running out!

Pre-training

- Previously: Compute is a binding resource; data is nearly infinite
- Now: Compute is getting cheaper, but data is running out!

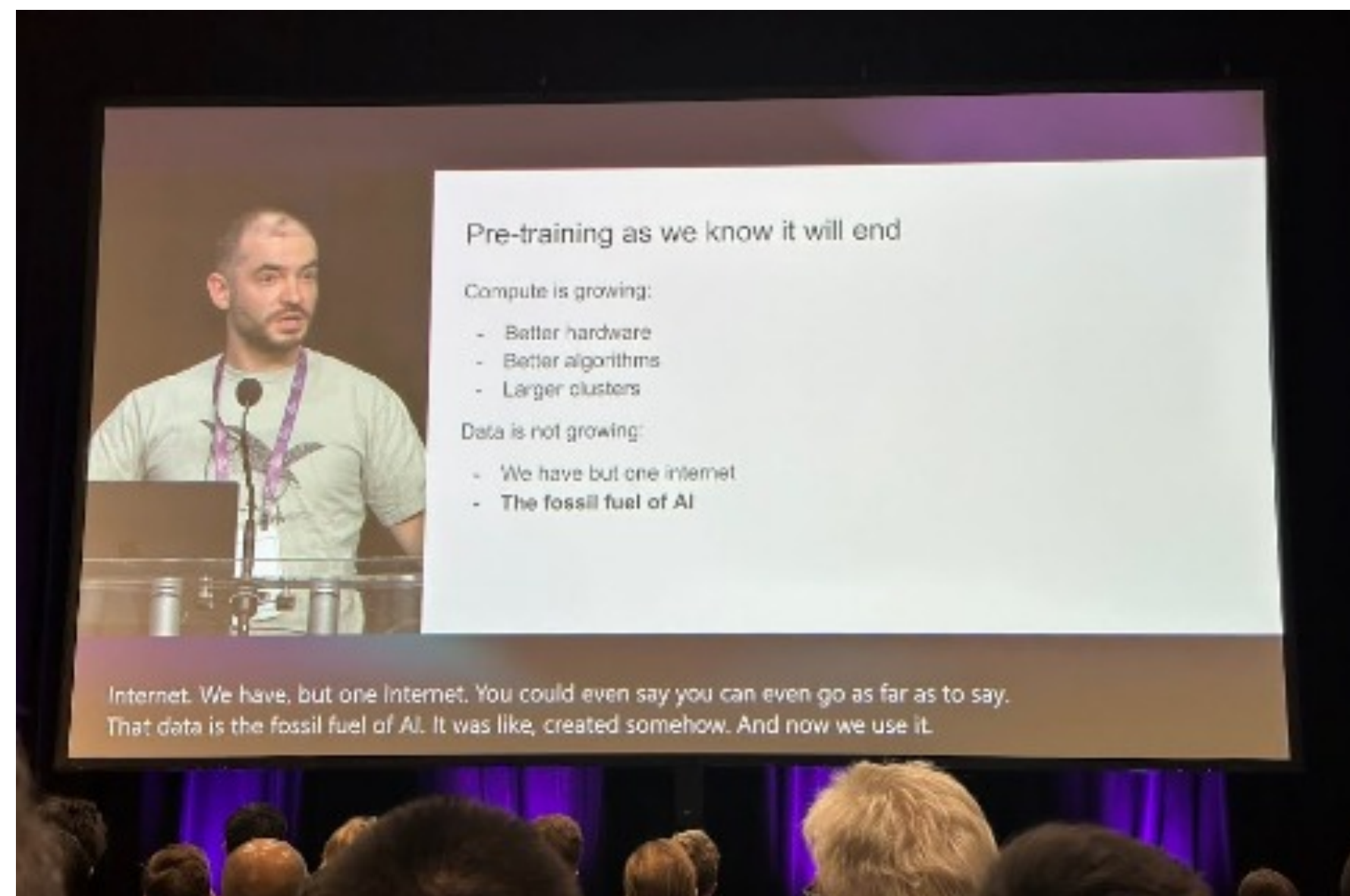


“We have but one Internet.”

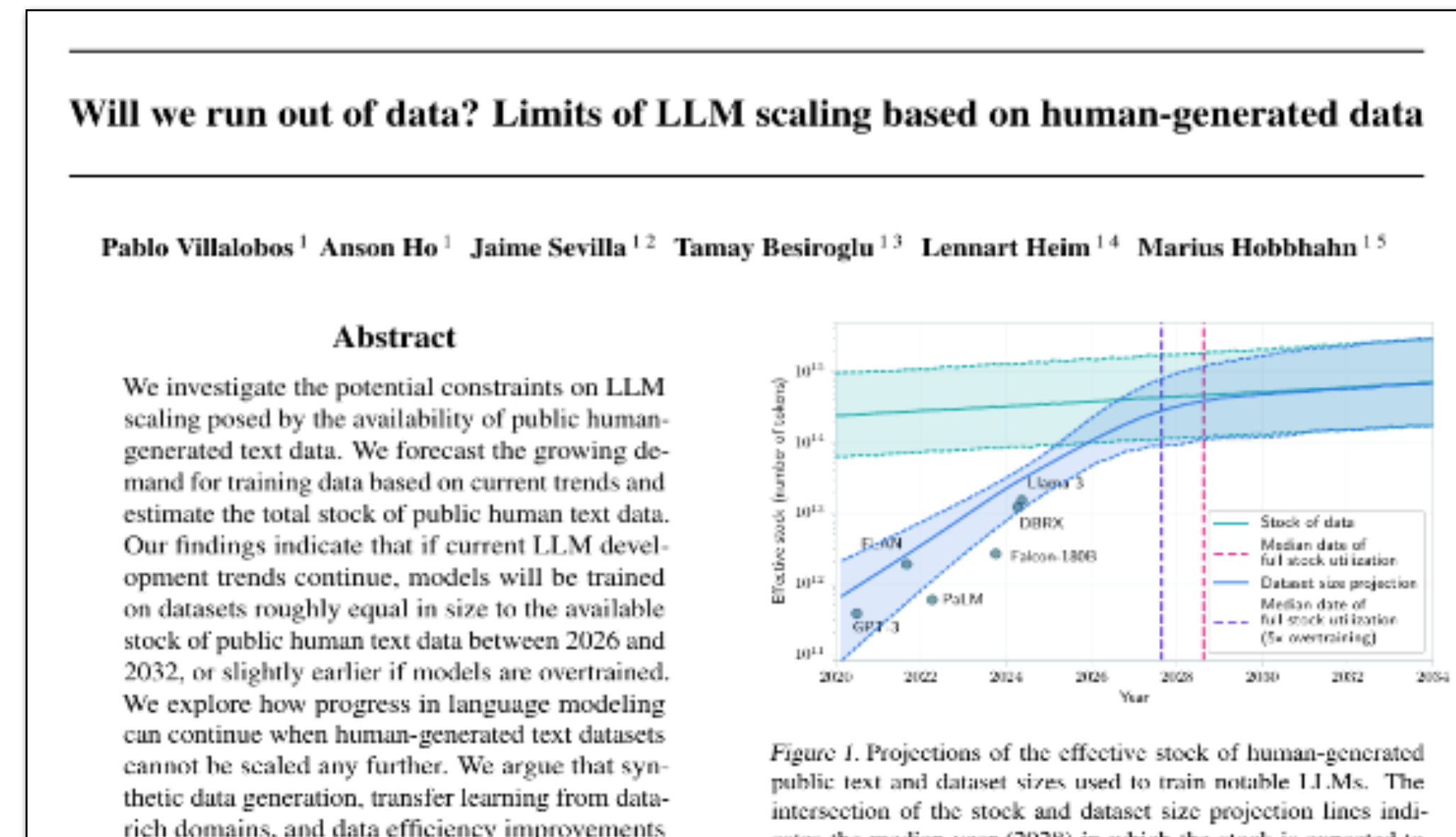
Ilya Sutskever @ NeurIPS 2024

Pre-training

- Previously: Compute is a binding resource; data is nearly infinite
- Now: Compute is getting cheaper, but data is running out!



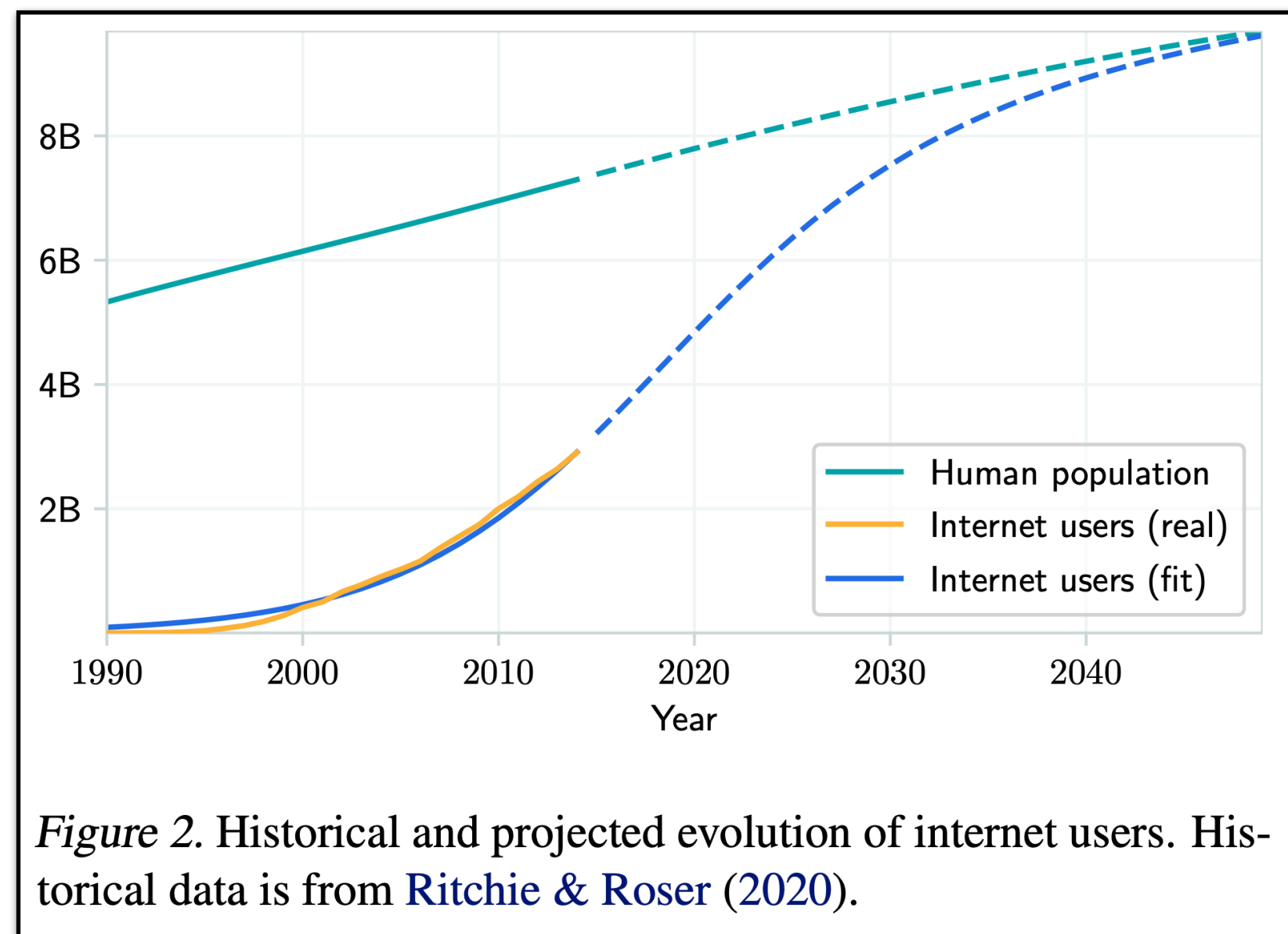
“We have but one Internet.”
Ilya Sutskever @ NeurIPS 2024



Forecast: Human text may be exhausted in 2028.

Why is Data becoming a Binding Resource?

1. The absolute quantity is limited physically.



Finite Internet users, finite token generation rate per user

Why is Data becoming a Binding Resource?

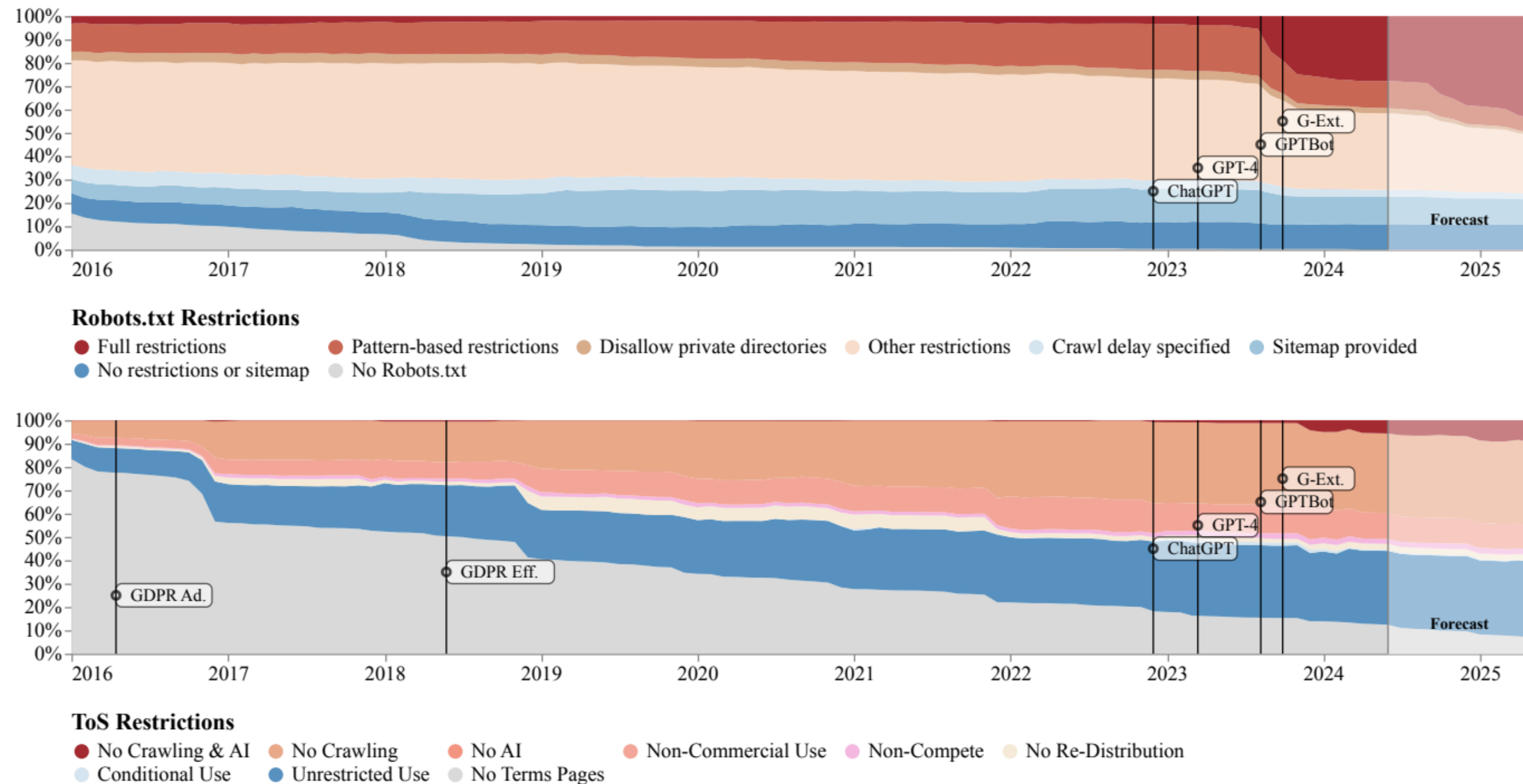
2. Data is fragmented



Valuable data is often restricted or proprietary

Why is Data becoming a Binding Resource?

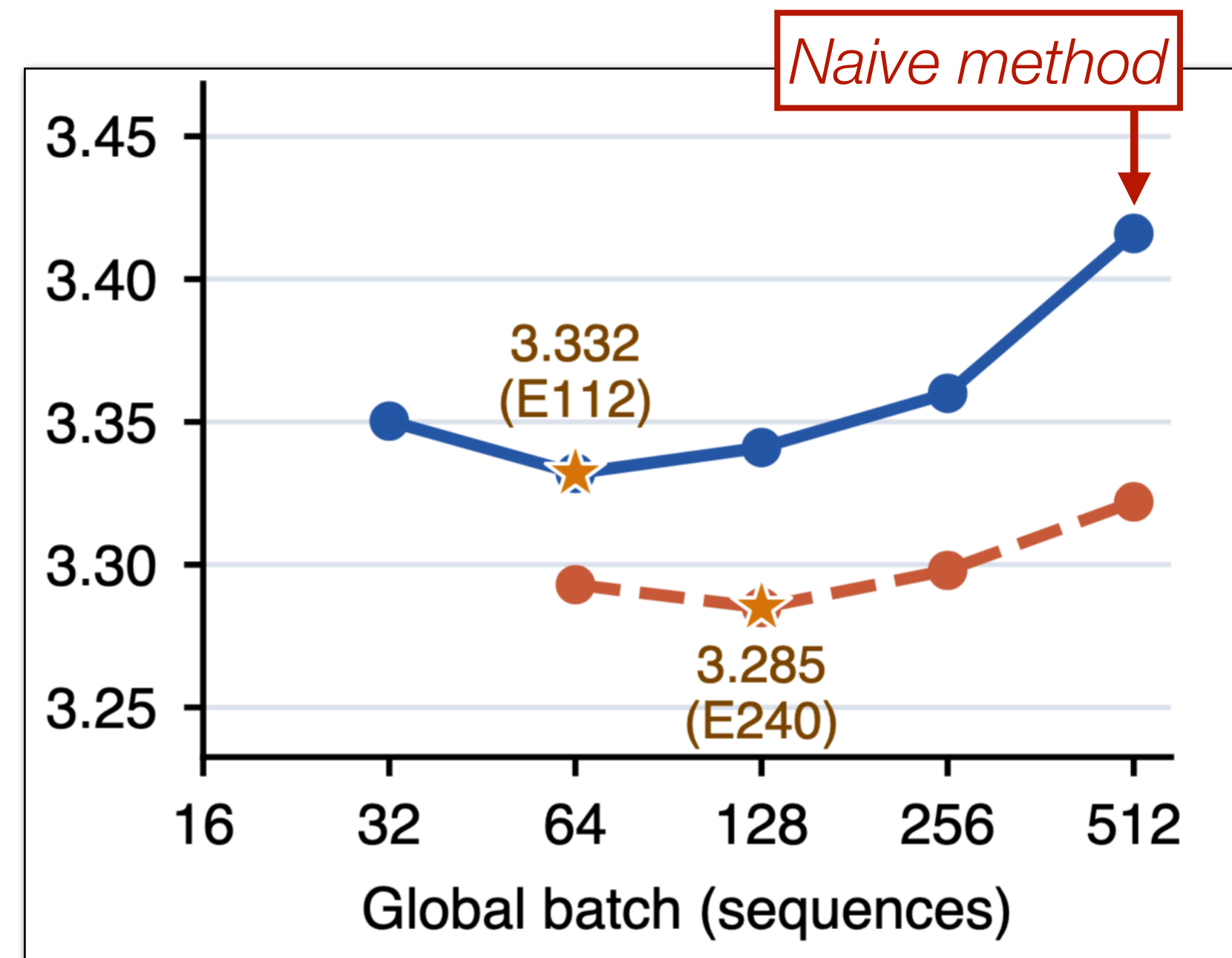
2. Data is fragmented



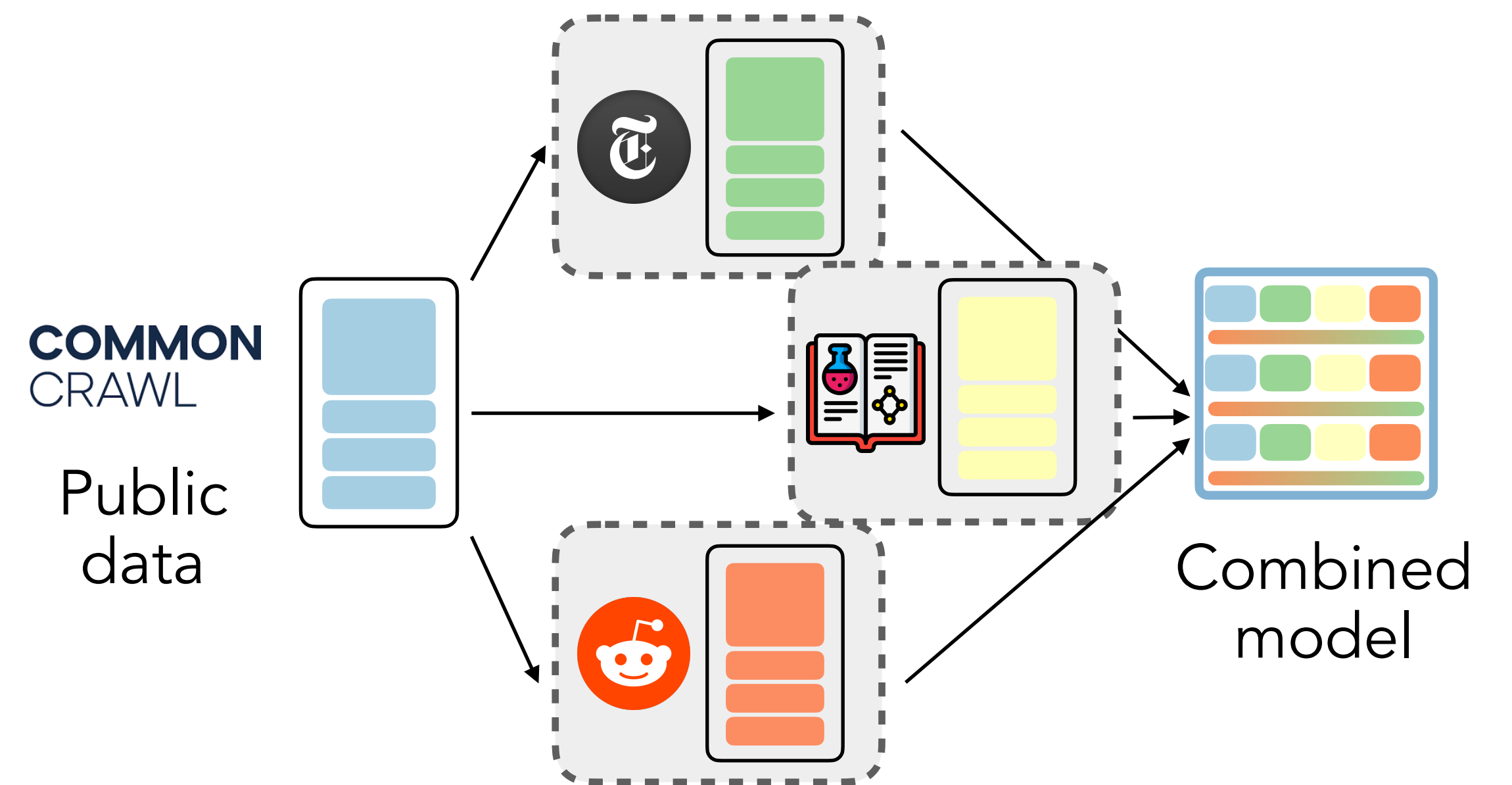
Previously public data is becoming proprietary

Longpre et al. 2024. "Consent in Crisis: The Rapid Decline of the AI Data Commons"

This Talk

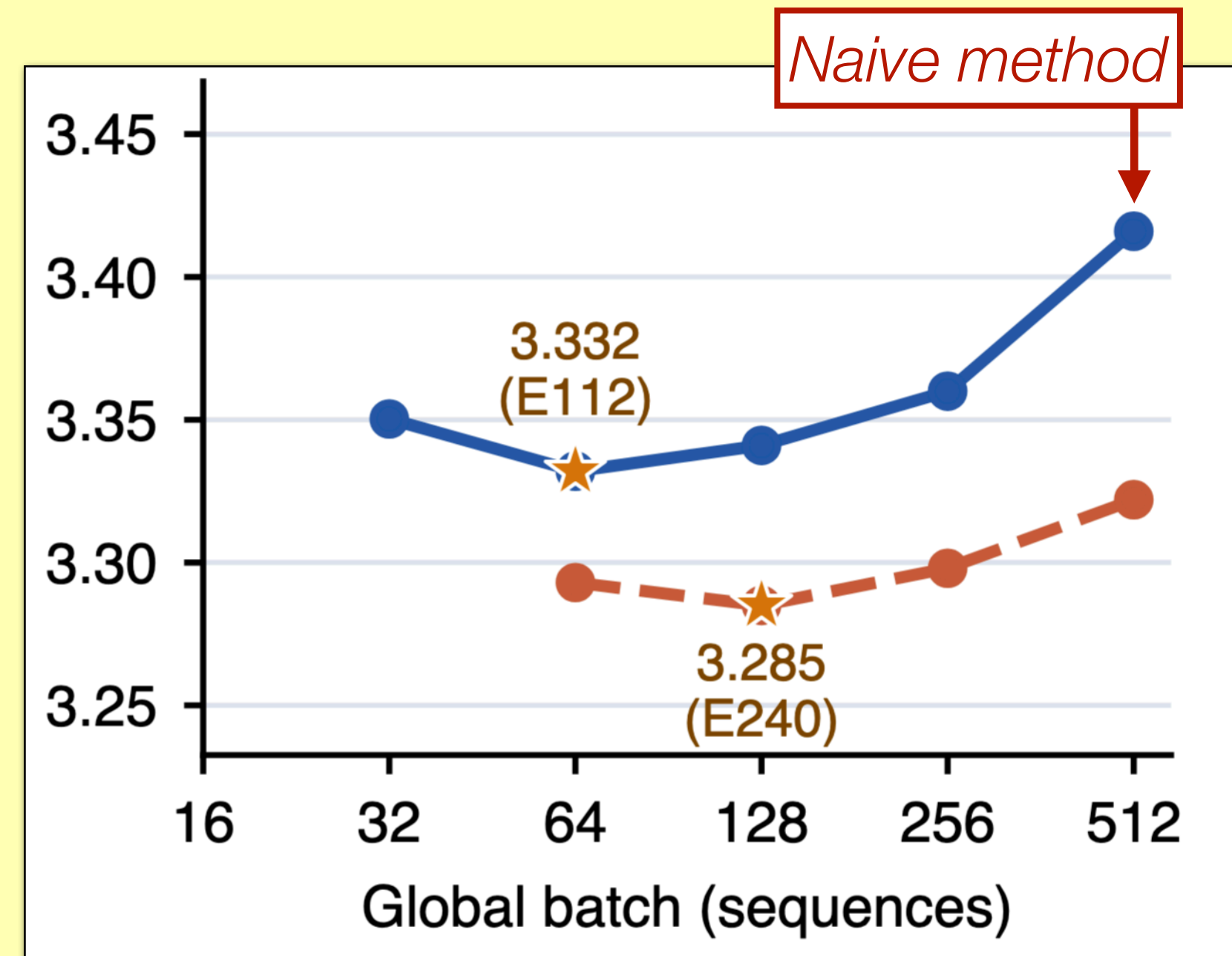


1. How to do repeated-data training *correctly*

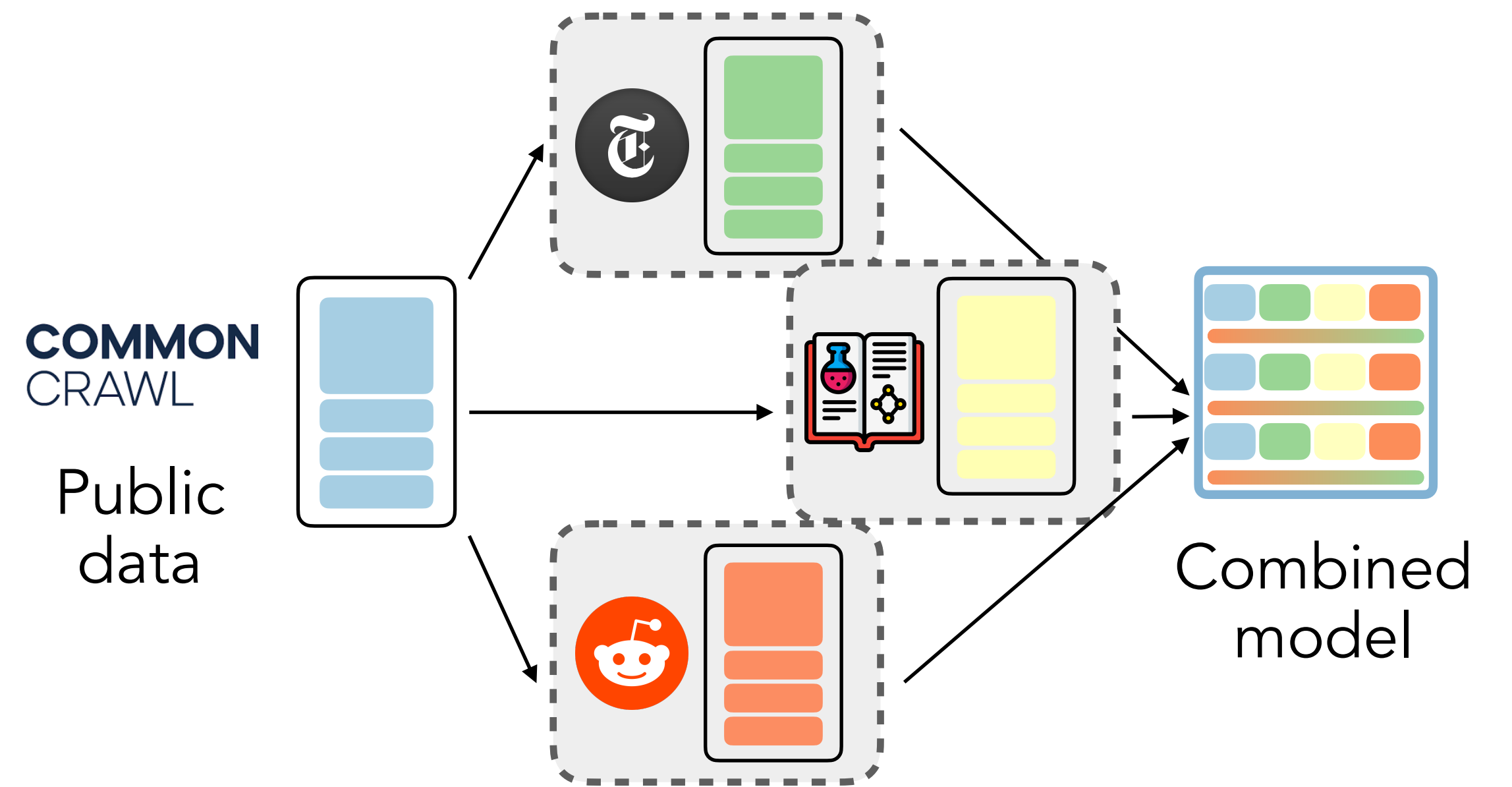


2. How training algorithms can change to leverage *fragmented* data

This Talk



1. How to do repeated-data training *correctly*



2. How training algorithms can change to leverage *fragmented* data

Prior Repeated-Data Training

Prior Repeated-Data Training

Scaling Data-Constrained Language Models

Niklas Muennighoff ¹ Alexander M. Rush ¹ Boaz Barak ² Teven Le Scao ¹
Aleksandra Piktus ¹ Nouamane Tazi ¹ Sampo Pyysalo ³ Thomas Wolf ¹ Colin Raffel ¹
¹ Hugging Face ² Harvard University ³ University of Turku

“Up to 4 epochs yields similar performance
to fresh data”

Training recipe remains unchanged

Prior Repeated-Data Training

Scaling Data-Constrained Language Models

Niklas Muennighoff¹ Alexander M. Rush¹ Boaz Barak² Teven Le Scao¹
Aleksandra Piktus¹ Nouamane Tazi¹ Sampo Pyysalo³ Thomas Wolf¹ Colin Raffel¹
¹ Hugging Face ² Harvard University ³ University of Turku

Pre-training under infinite compute

Konwoo Kim[∞], Suhas Kotha[∞], Percy Liang, Tatsunori Hashimoto
Stanford University

“Up to 4 epochs yields similar performance
to fresh data”

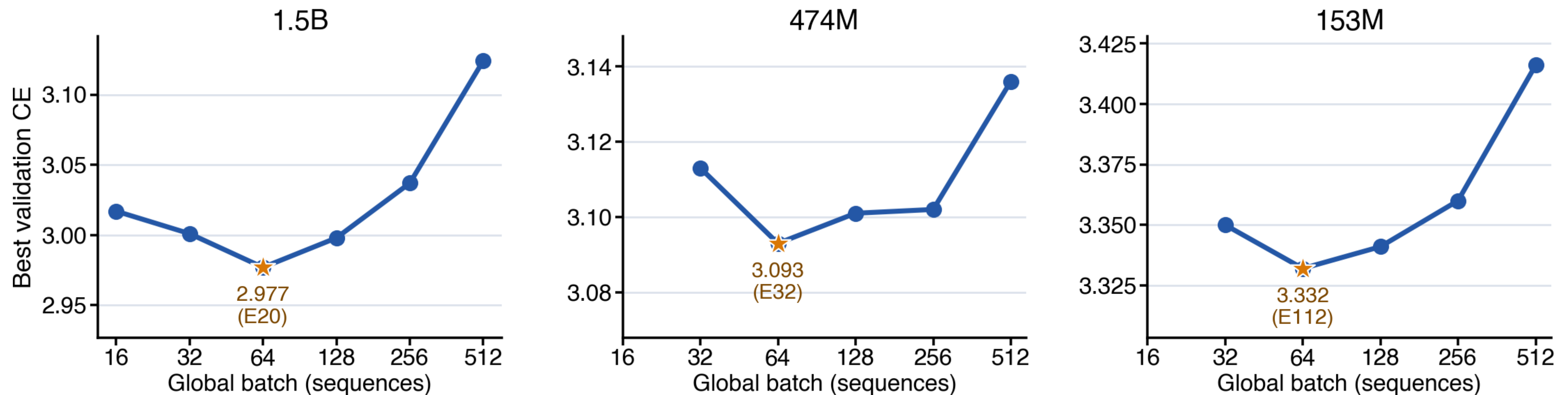
Training recipe remains unchanged

“Tuning weight decay improves
repeated-data frontier”

Also parameter scaling & ensembling, which
increase inference cost

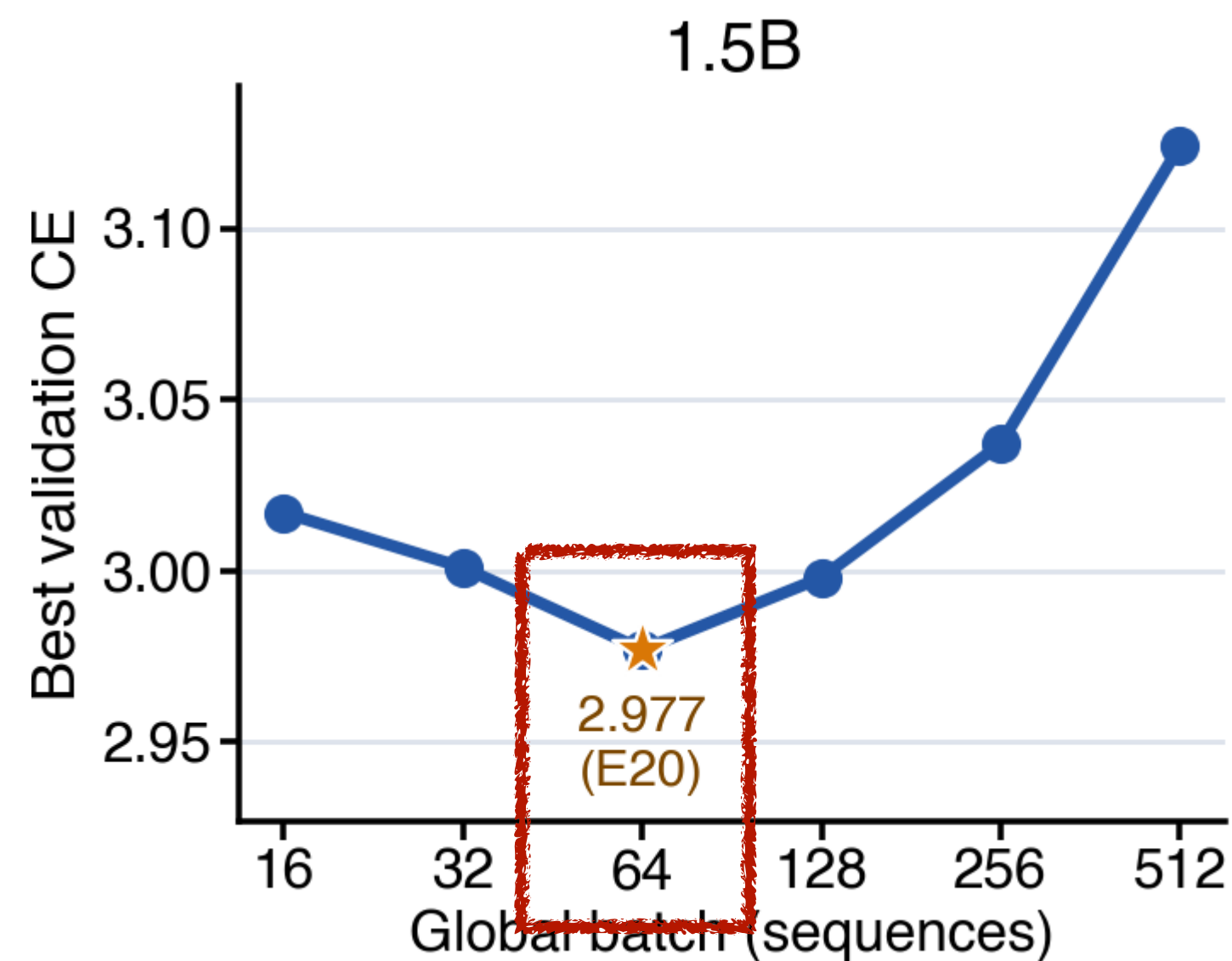
Finding 1: Infinite-compute optimal batch size is much smaller than what's normally used

Finding 1: Infinite-compute optimal batch size is much smaller than what's normally used



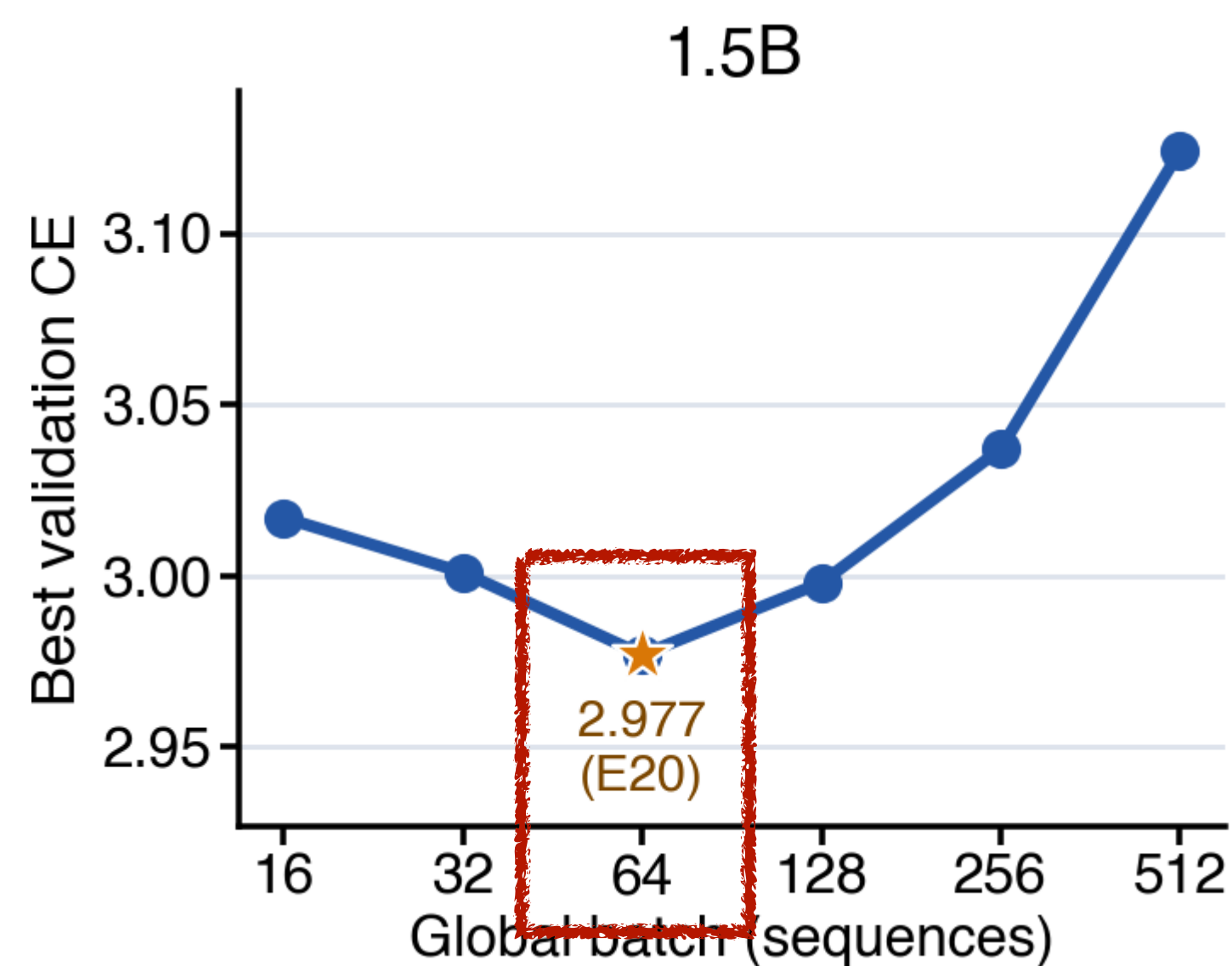
(batch size, learning rate, weight decay, number of repetitions) jointly tuned; 1B-token unique-data pool

Finding 1: Infinite-compute optimal batch size is much smaller than what's normally used



- Much smaller than what's used for public artifacts — GPT-2, Cerebras-GPT, Pythia, and Olmo all used global batches that are 8x to 32x larger.

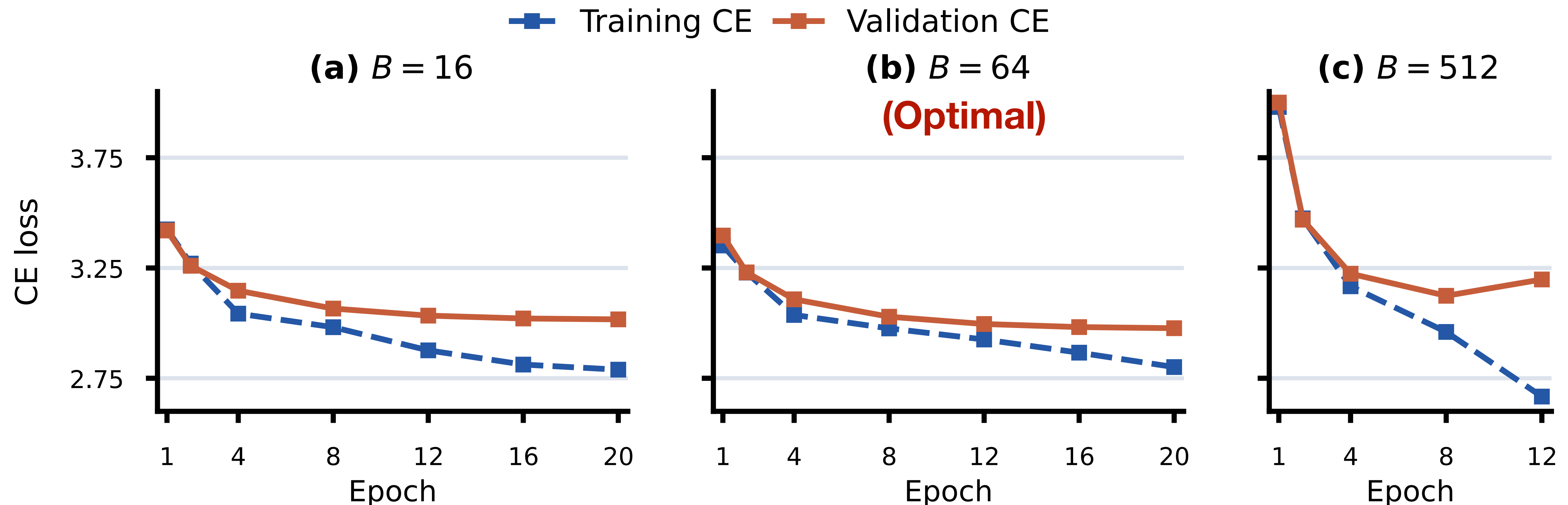
Finding 1: Infinite-compute optimal batch size is much smaller than what's normally used



- Much smaller than what's used for public artifacts — GPT-2, Cerebras-GPT, Pythia, and Olmo all used global batches that are 8x to 32x larger.
- Much smaller than critical batch size identified by prior work

	C_F (equiv. E)	Batch size (million tokens)
Bi et al.	0.90×10^{19} ($E = 1$)	0.463
Bi et al.	7.20×10^{19} ($E = 8$)	0.914
Bi et al.	18.0×10^{19} ($E = 20$)	1.233
Shuai et al.	0.90×10^{19} ($E = 1$)	0.551
Shuai et al.	7.20×10^{19} ($E = 8$)	0.681
Shuai et al.	18.0×10^{19} ($E = 20$)	0.747
Ours	∞ (C_F : $E = 20$)	0.262

Finding 1: Infinite-compute optimal batch size is much smaller than what's normally used



Noisy gradients from small-batch training provide useful regularization!

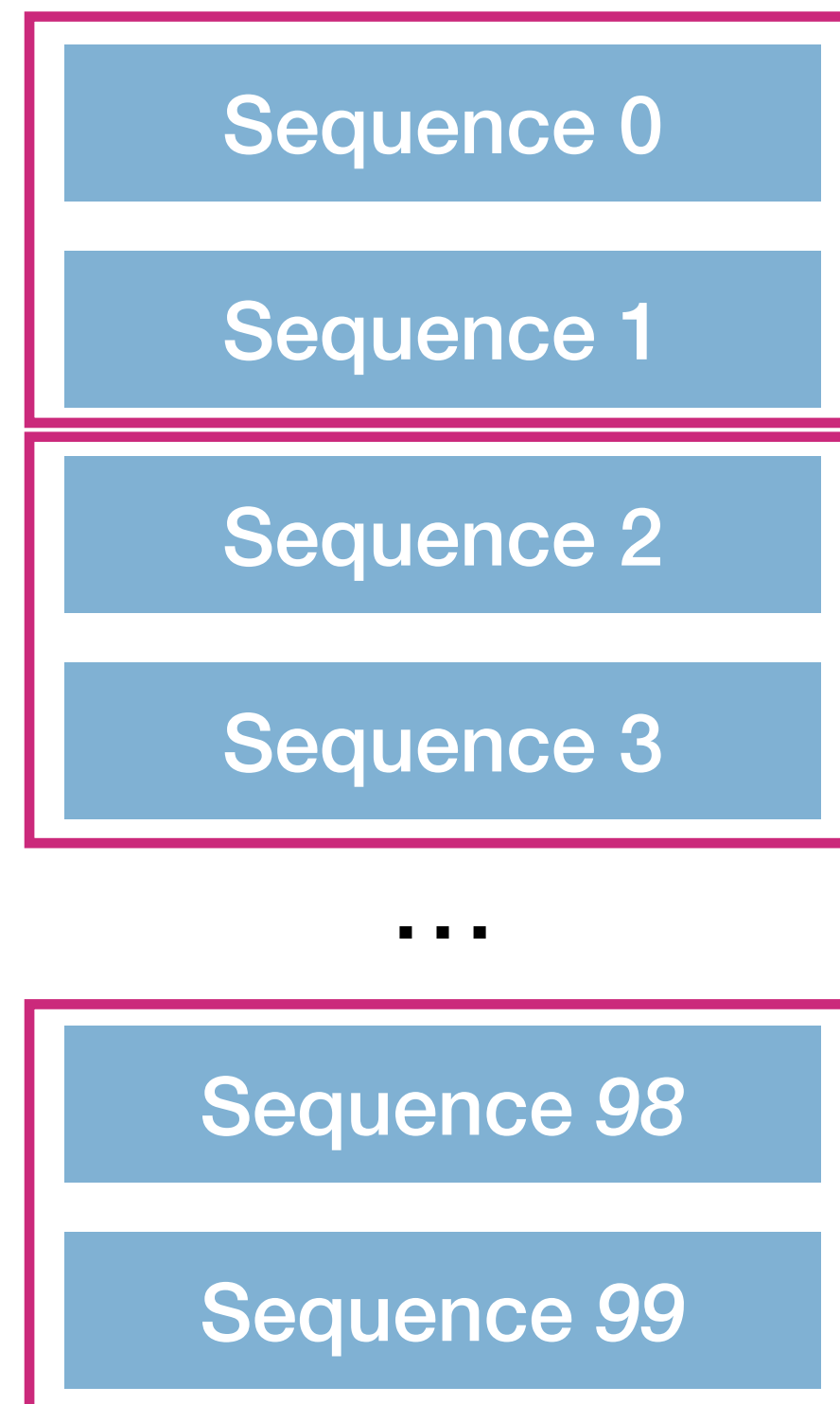
Finding 2: Recipe itself can be tuned

1. Data loader

Finding 2: Recipe itself can be tuned

1. Data loader

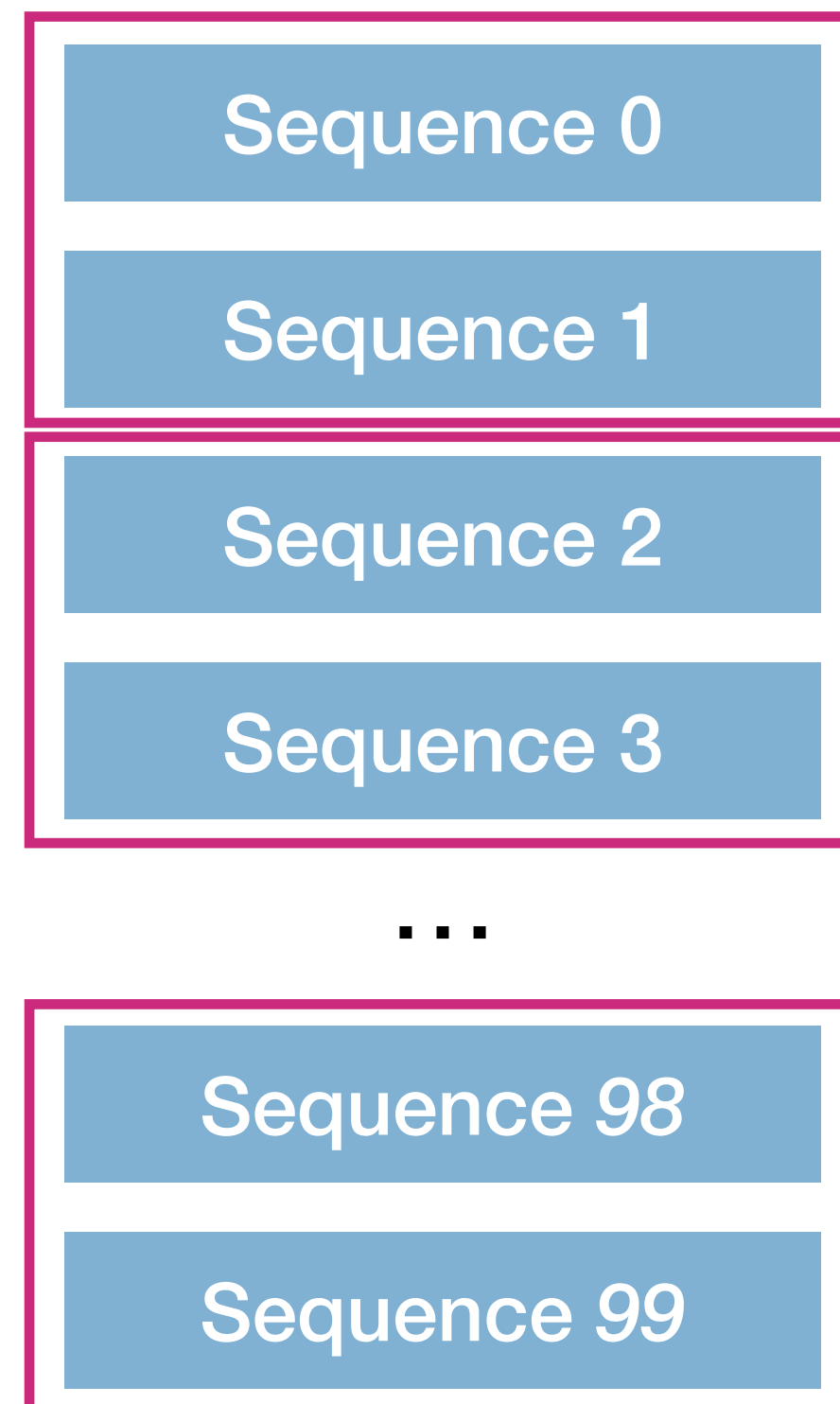
Epoch 1



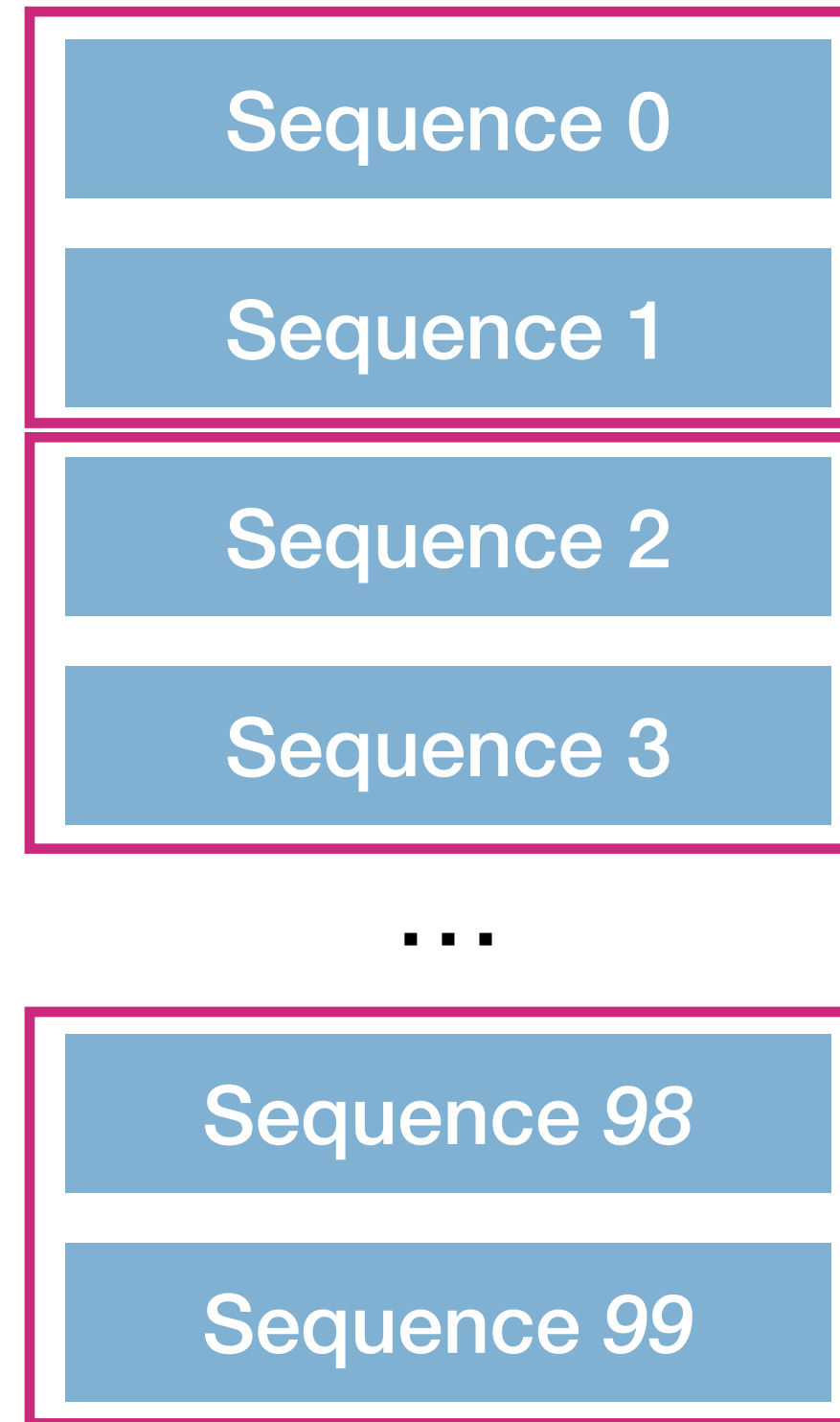
Finding 2: Recipe itself can be tuned

1. Data loader

Epoch 1



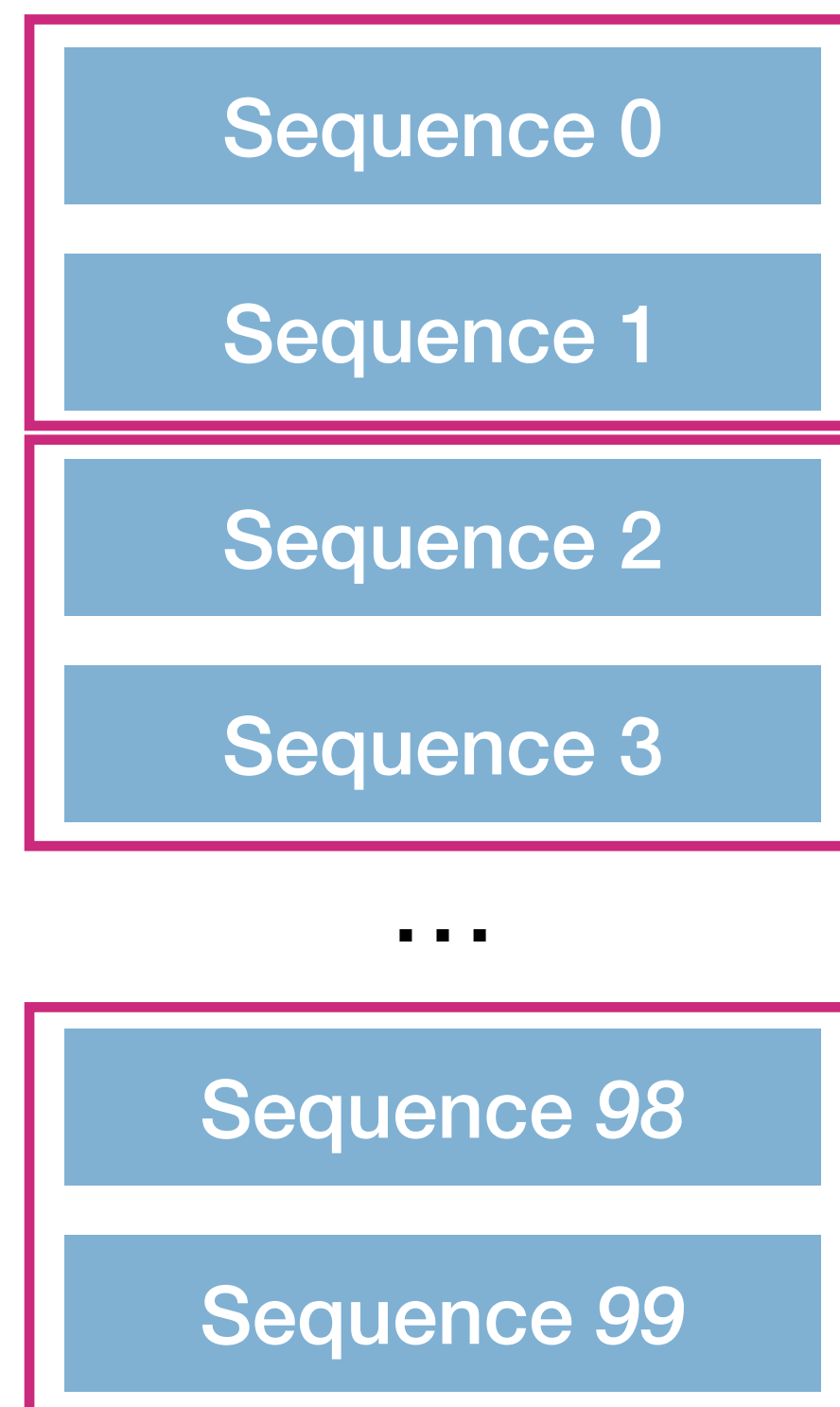
*Epoch 2
w. FIXED dataloader*



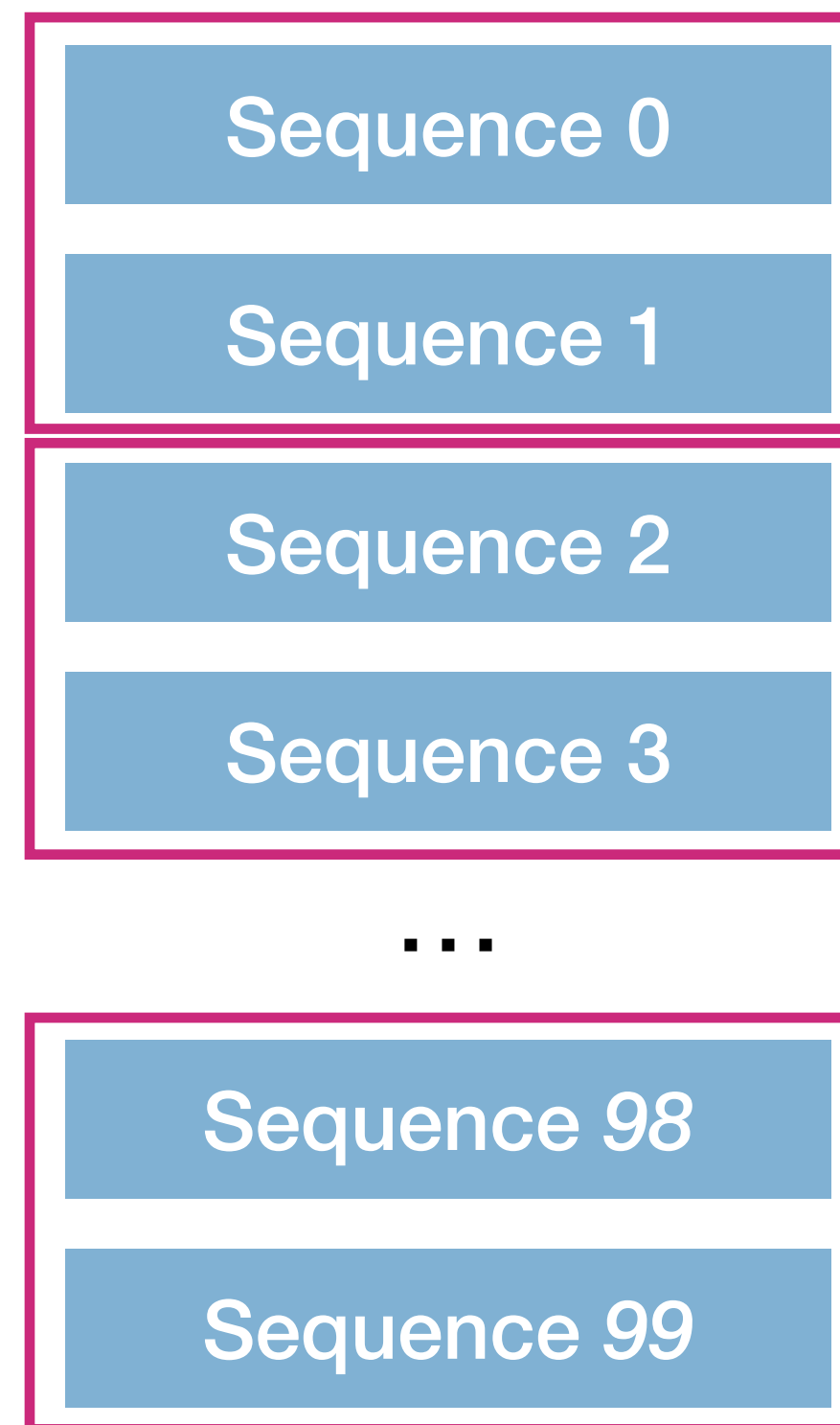
Finding 2: Recipe itself can be tuned

1. Data loader

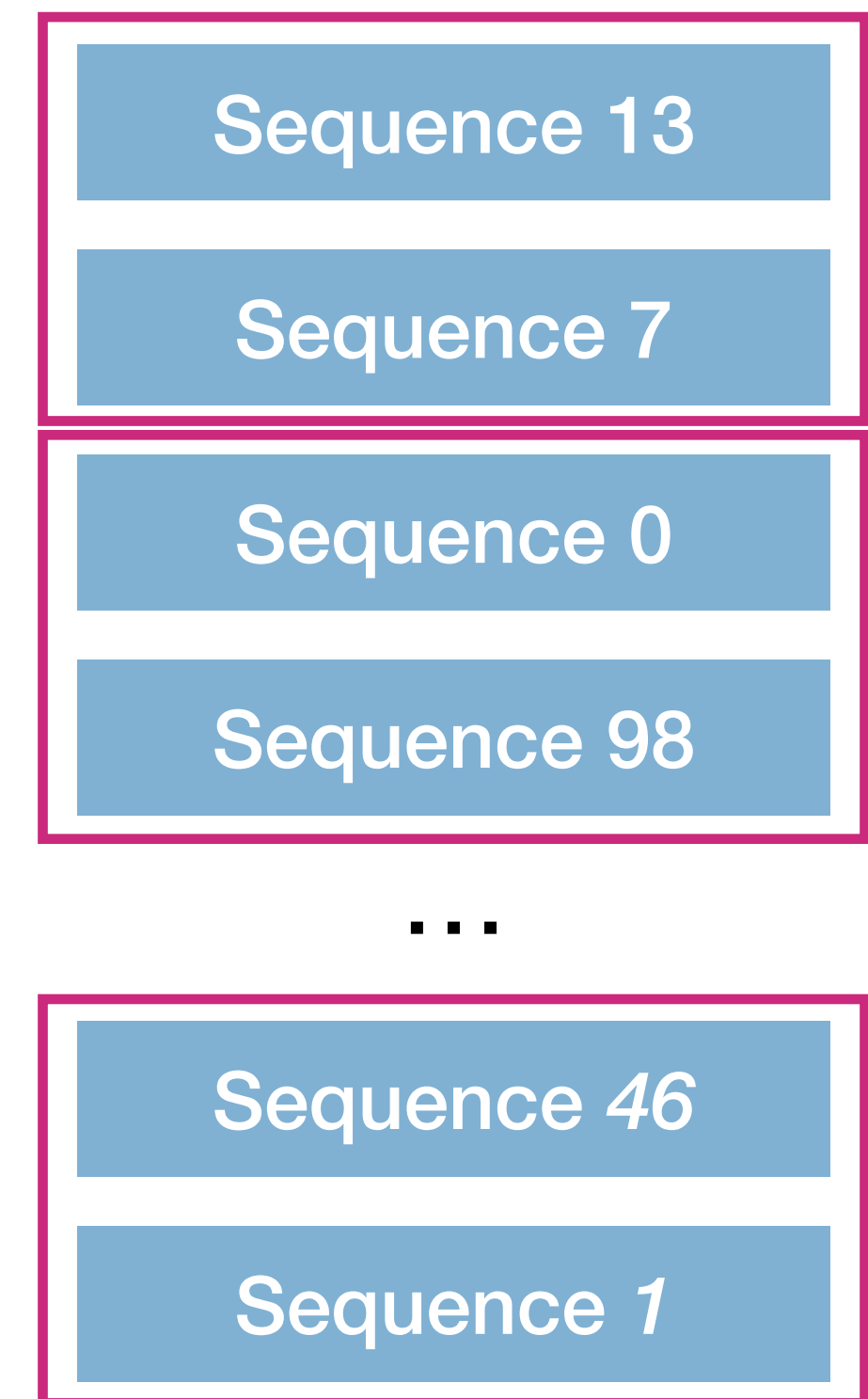
Epoch 1



*Epoch 2
w. FIXED dataloader*



*Epoch 2
w. SHUFFLE dataloader*



Finding 2: Recipe itself can be tuned

1. Data loader — REPACK

Training data

Doc 1 (82 tokens)

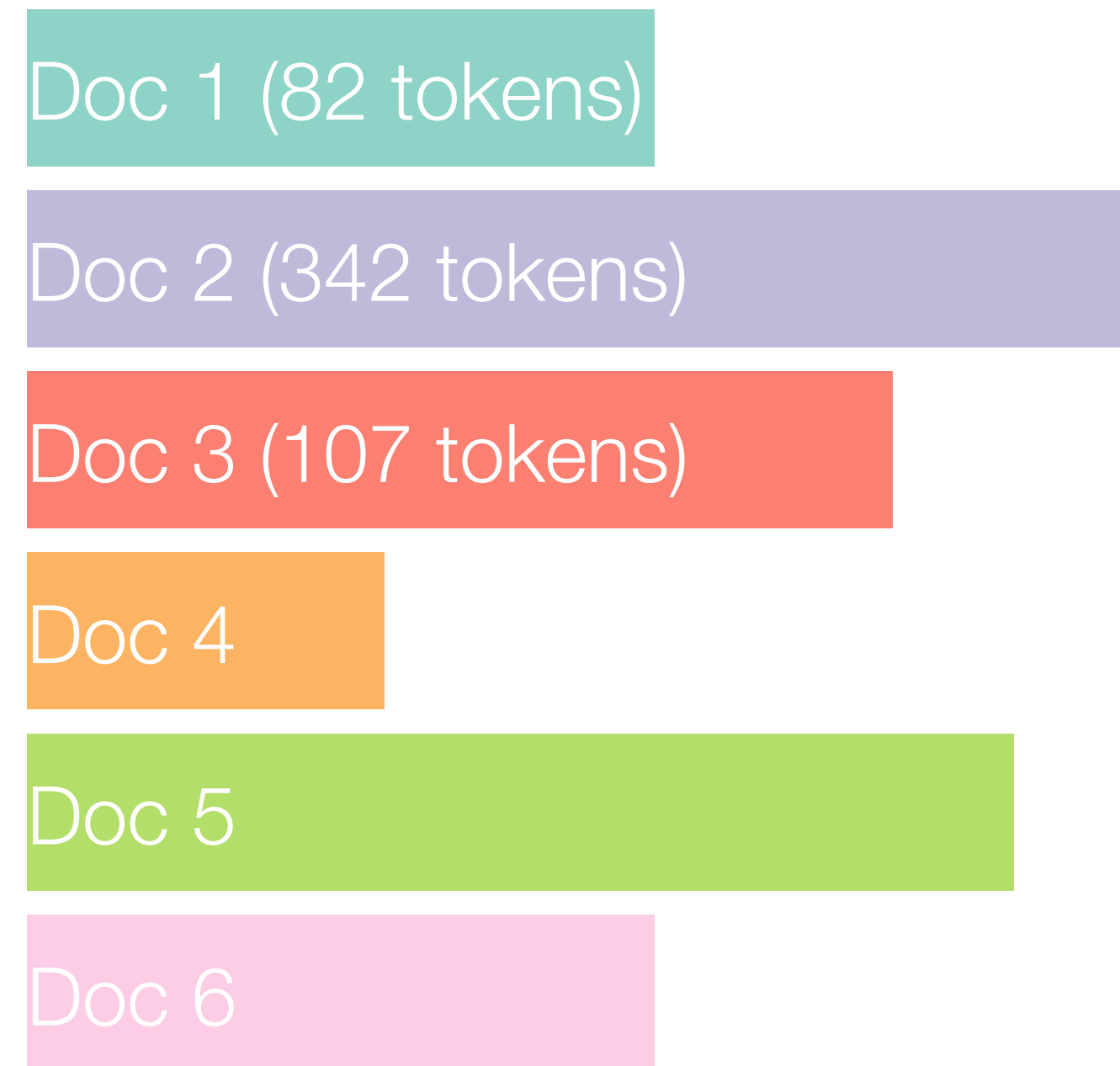
Doc 2 (342 tokens)

Doc 3 (107 tokens)

Doc 4

Doc 5

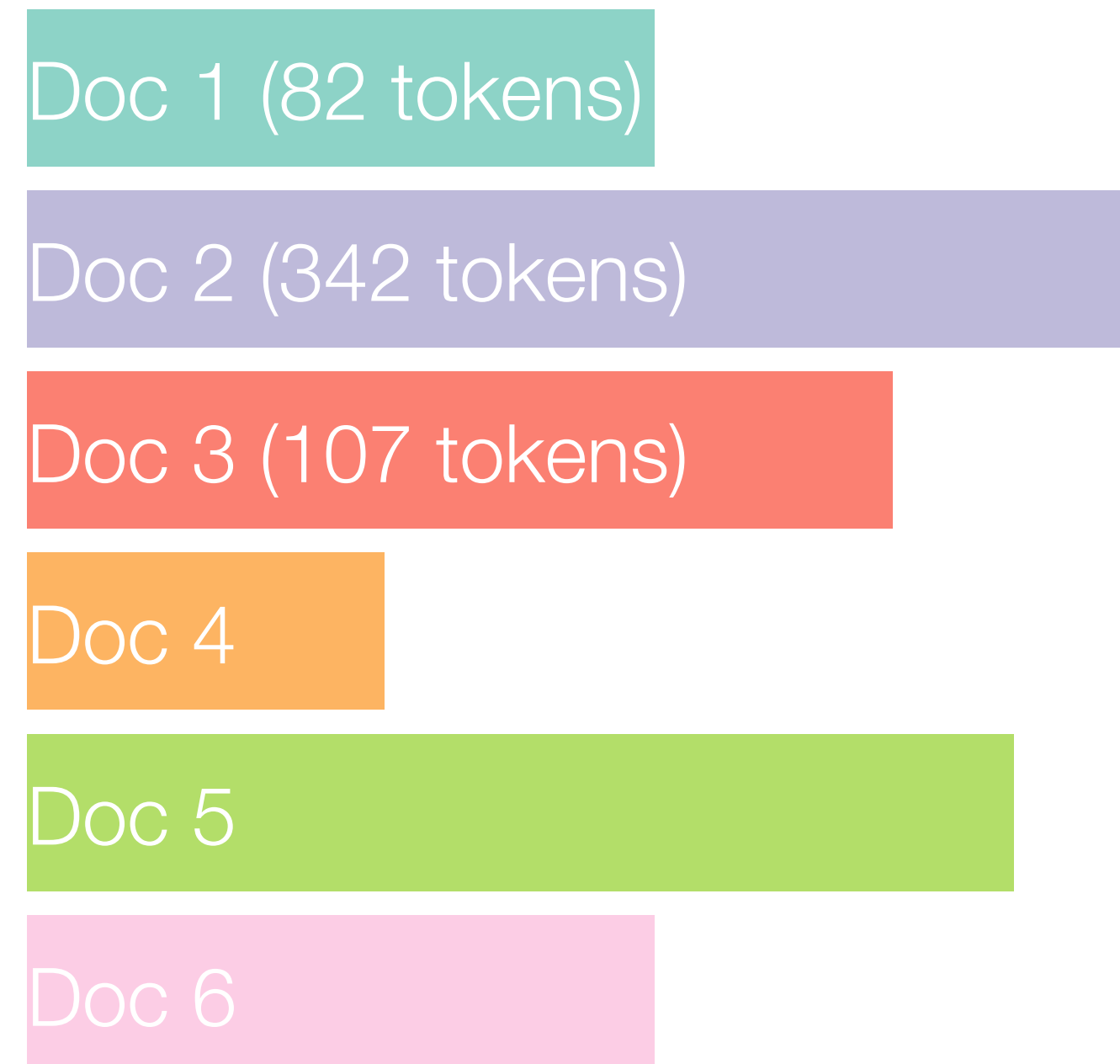
Doc 6



Finding 2: Recipe itself can be tuned

1. Data loader — REPACK

Training data



Packing

Sequence 0:



Sequence 1:



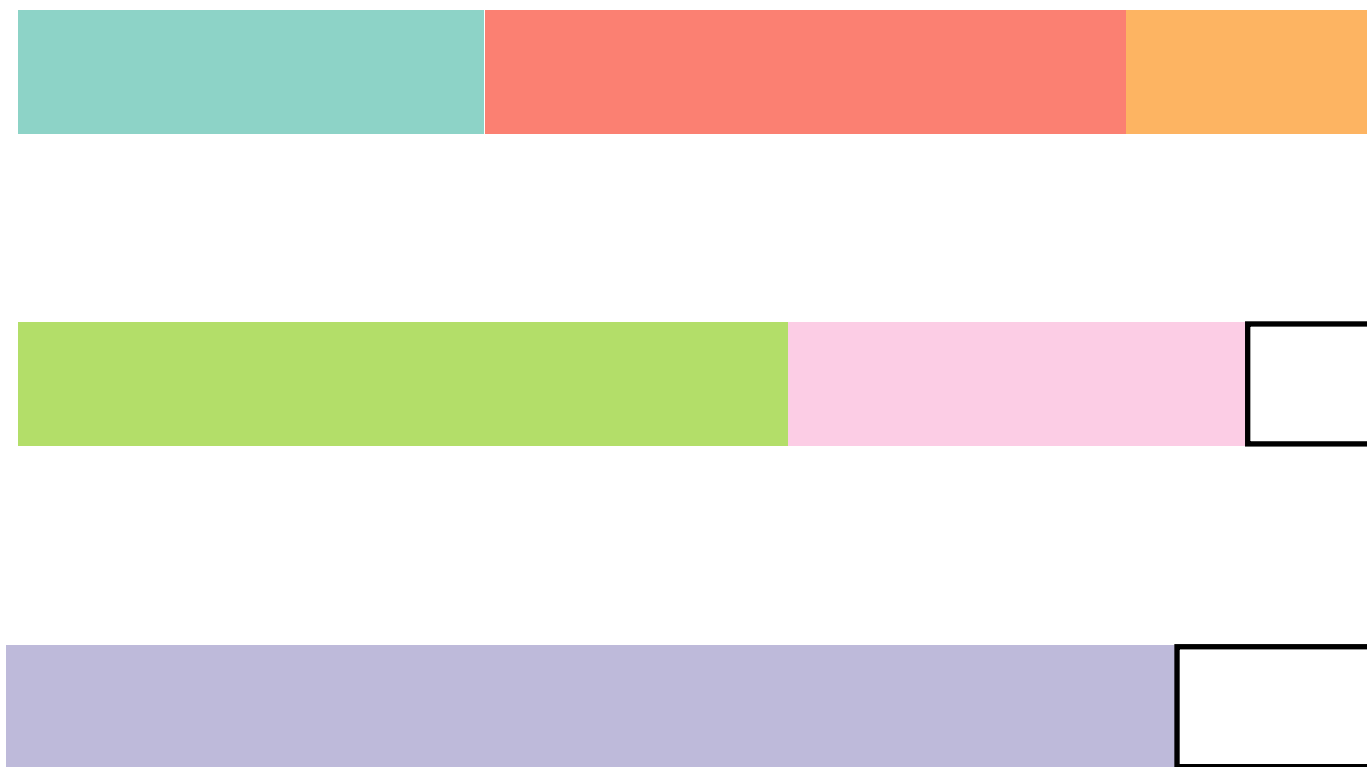
Sequence 2:



Finding 2: Recipe itself can be tuned

1. Data loader — REPACK

Epoch 1



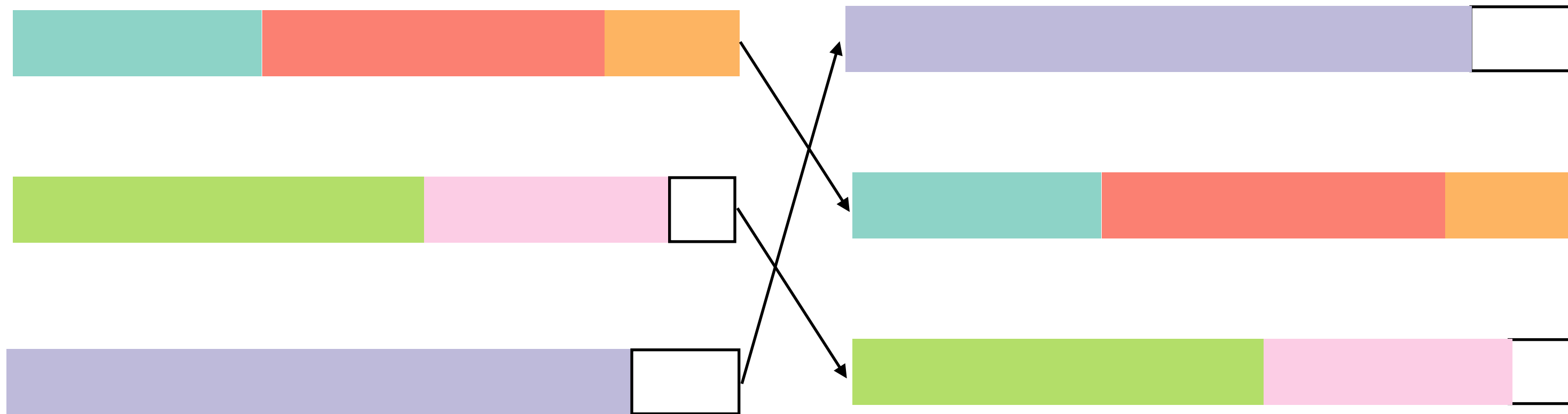
Finding 2: Recipe itself can be tuned

1. Data loader — REPACK

Epoch 1

Epoch 2

w. SHUFFLE dataloader

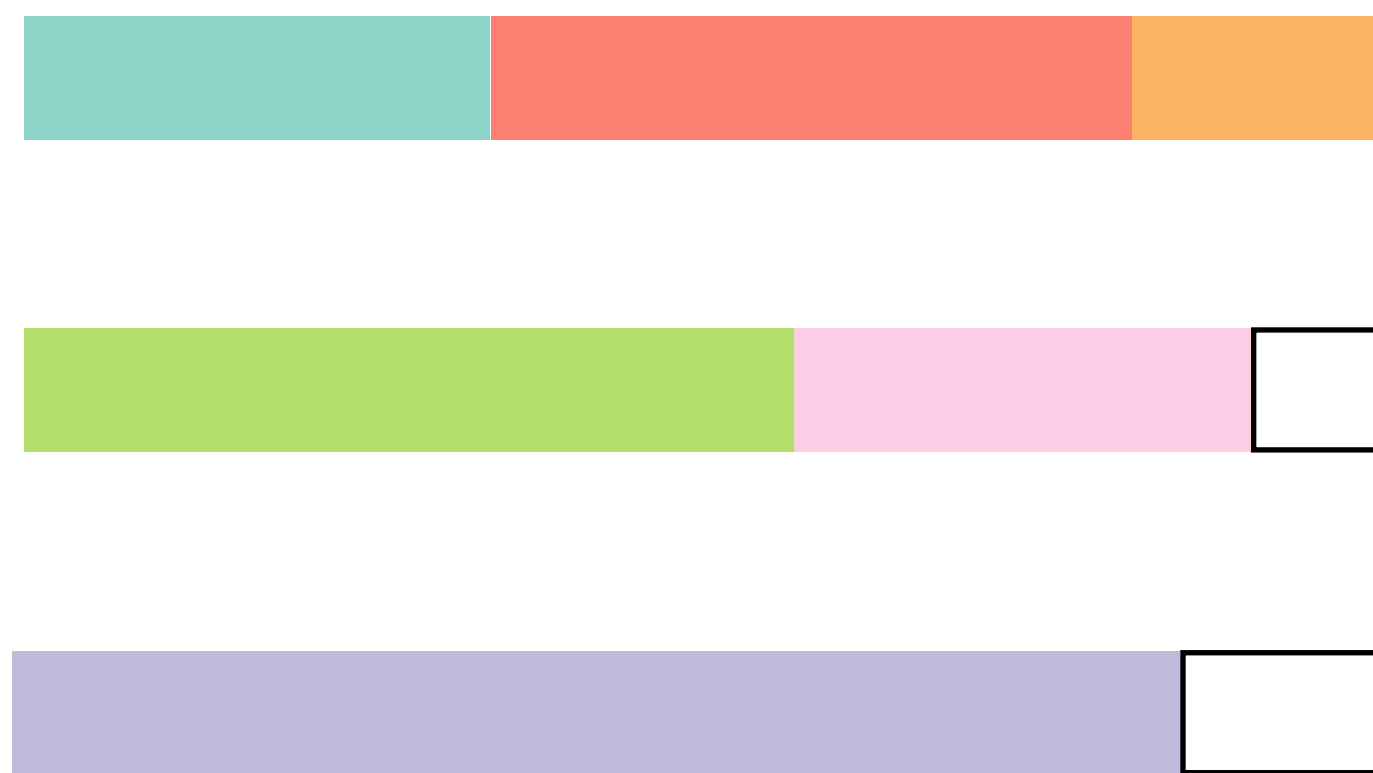


*Order is shuffled, but the
same sequences are seen*

Finding 2: Recipe itself can be tuned

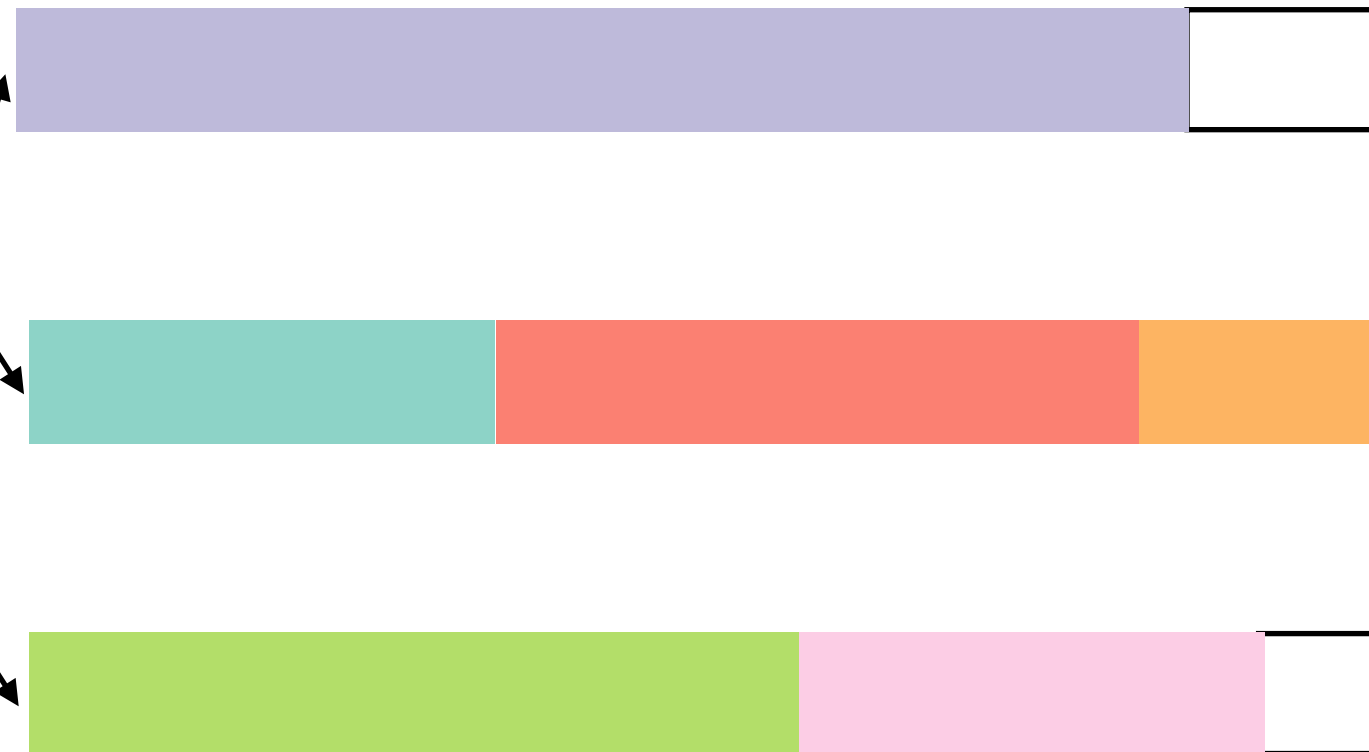
1. Data loader — REPACK

Epoch 1



Epoch 2

w. SHUFFLE dataloader



*Order is shuffled, but the
same sequences are seen*

Epoch 2

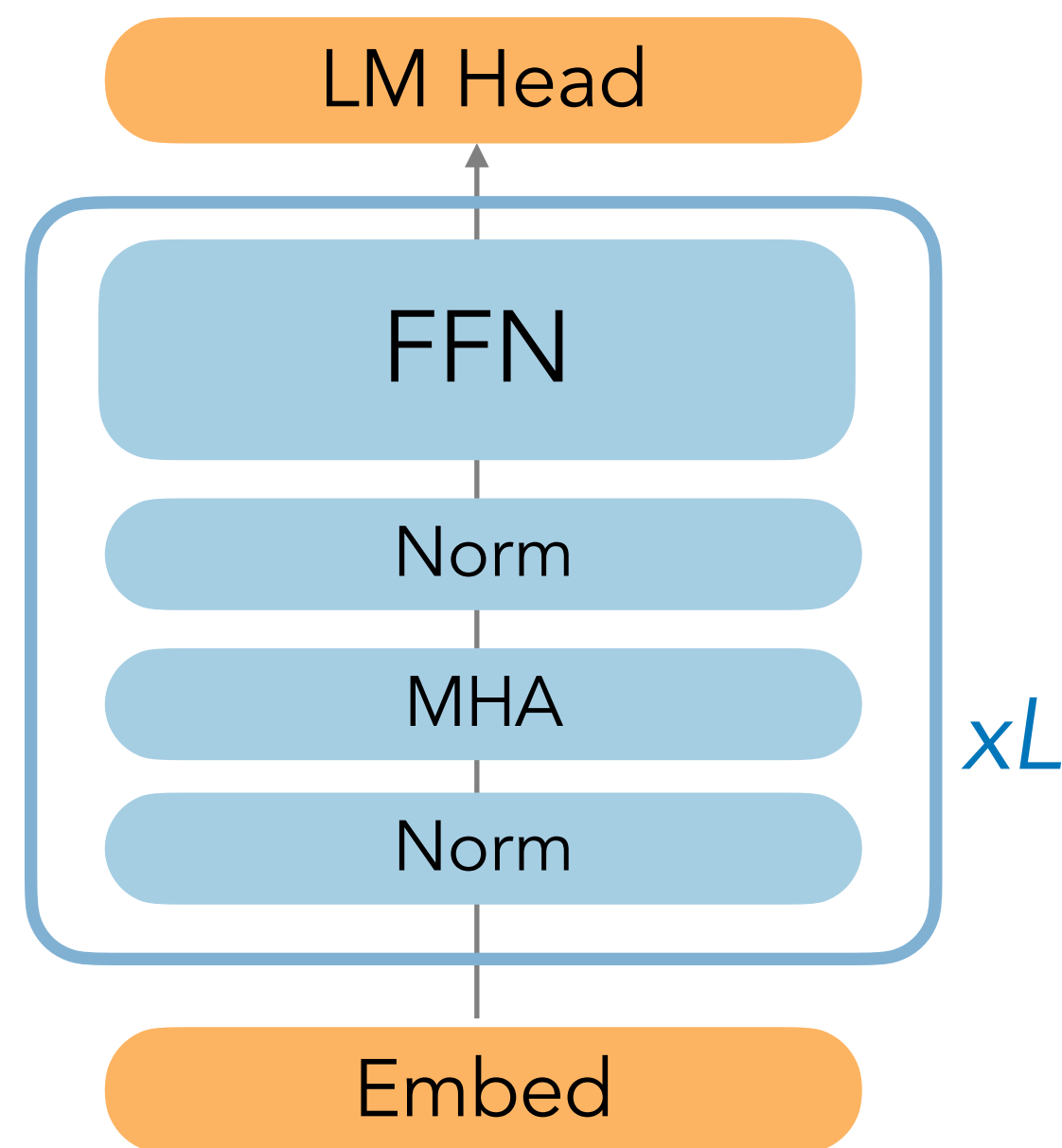
w. REPACK dataloader



*Every sequence is literally
fresh*

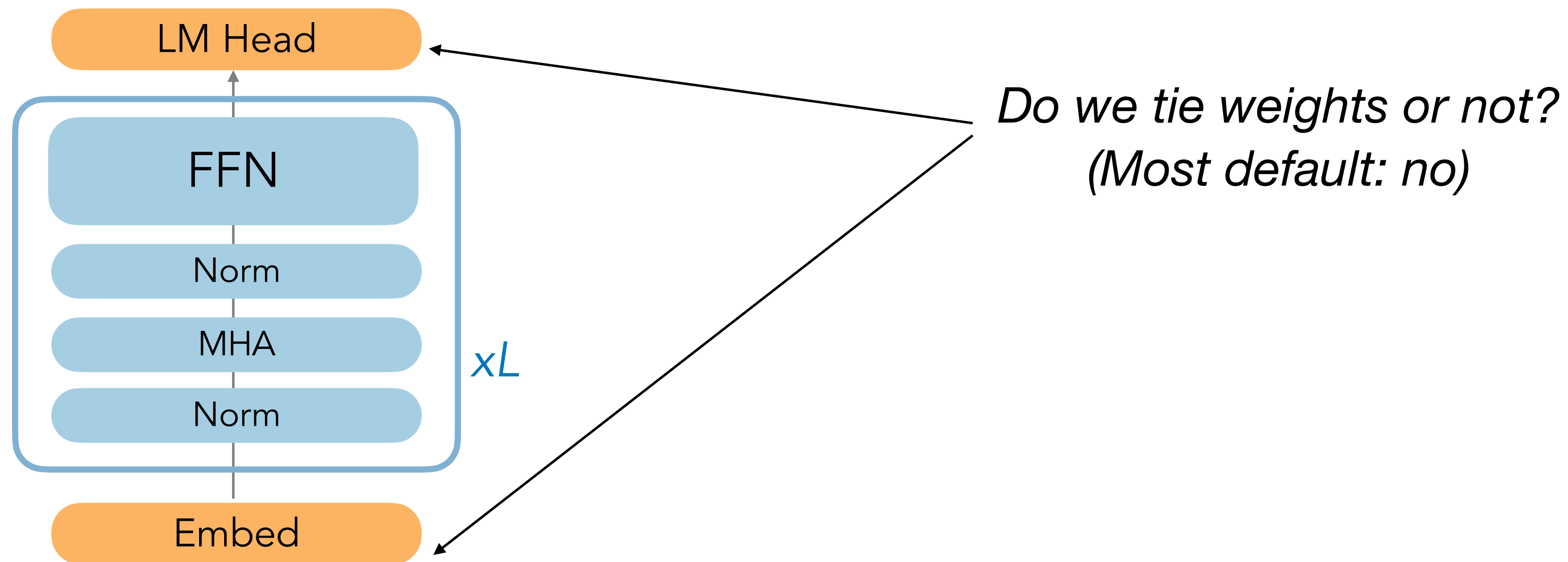
Finding 2: Recipe itself can be tuned

2. Embeddings (tying, decay)



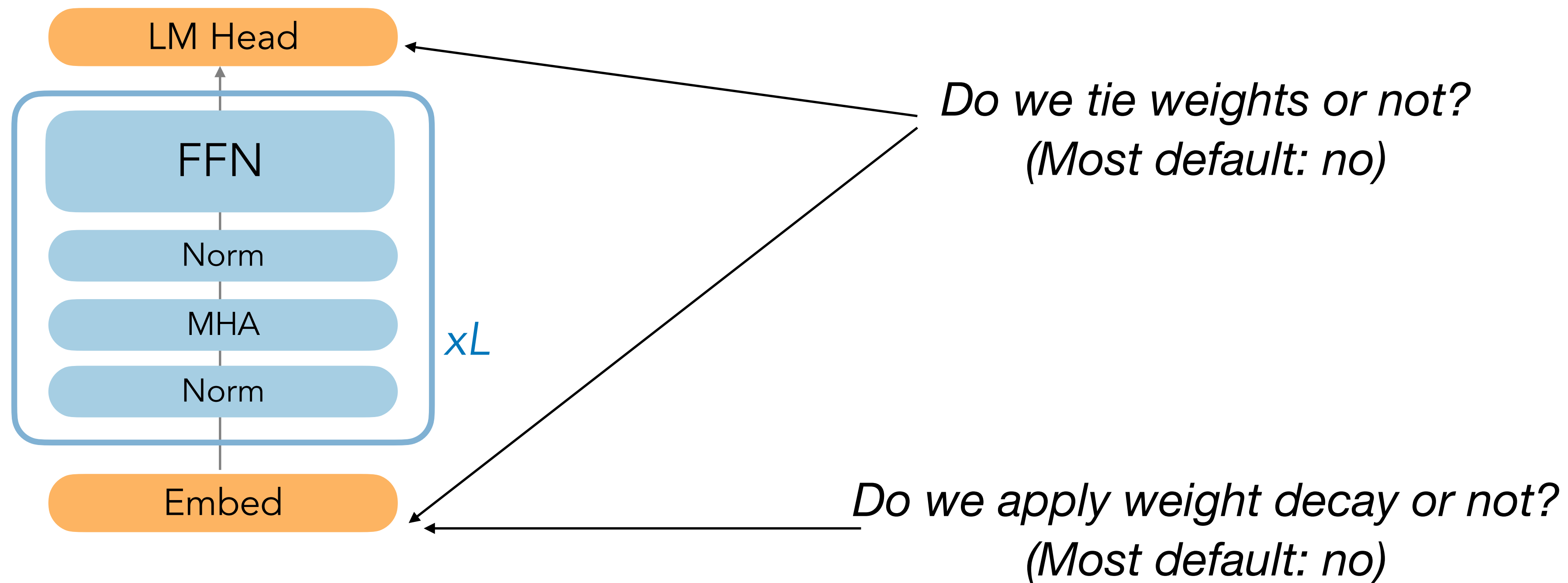
Finding 2: Recipe itself can be tuned

2. Embeddings (tying, decay)



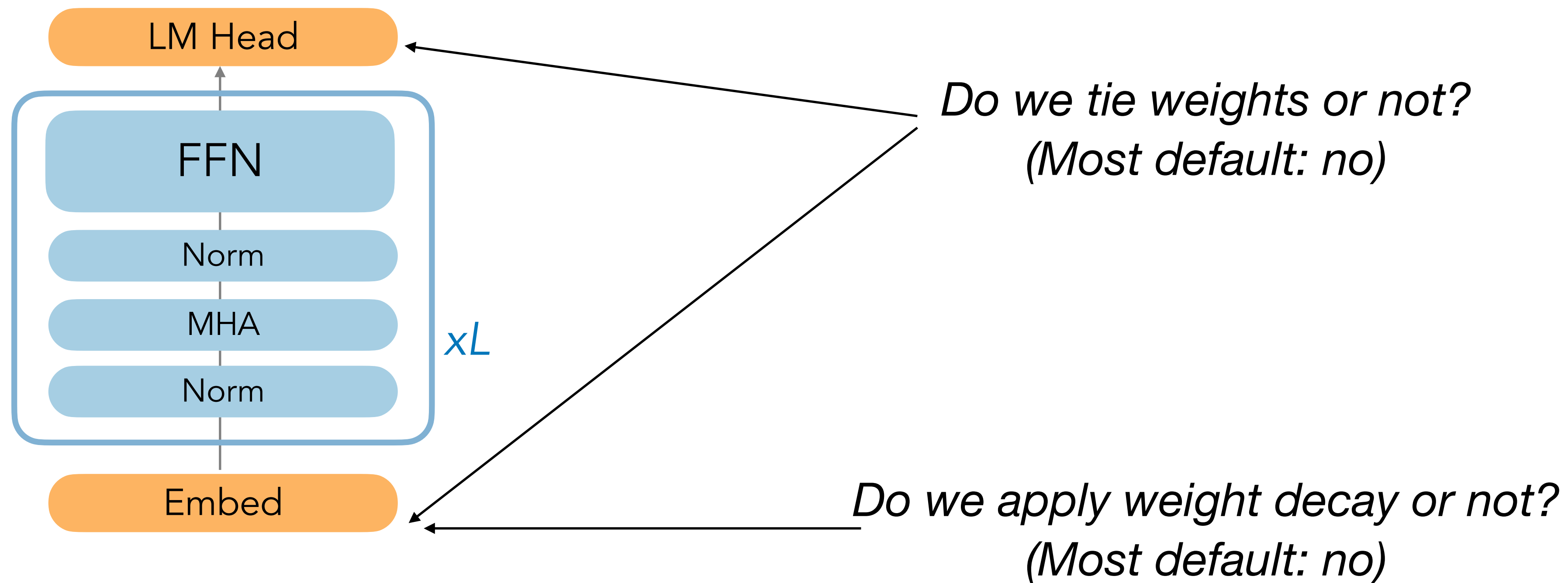
Finding 2: Recipe itself can be tuned

2. Embeddings (tying, decay)



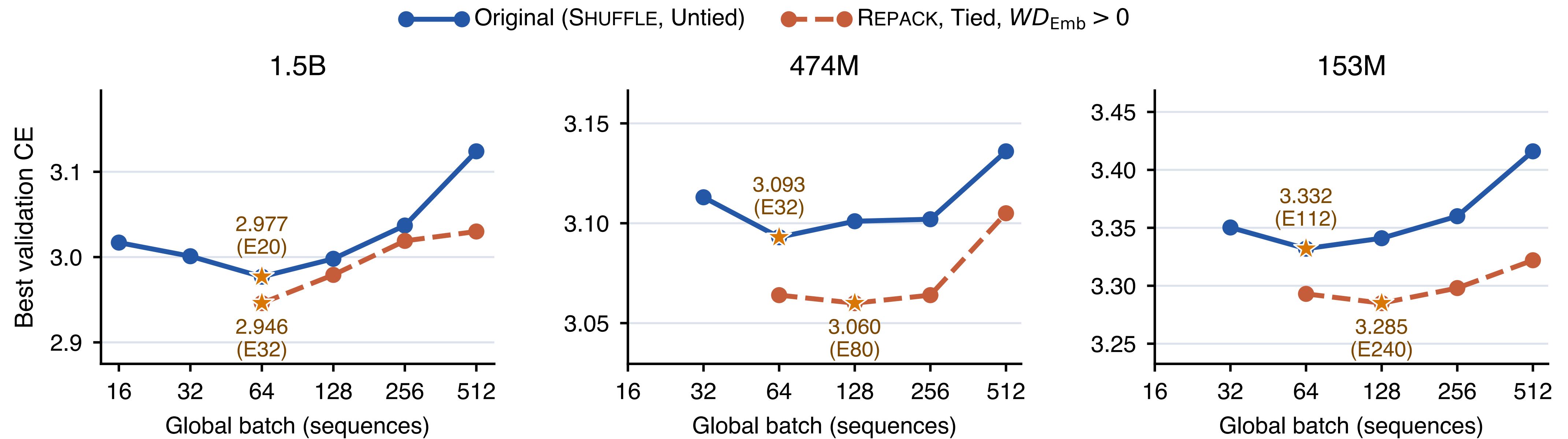
Finding 2: Recipe itself can be tuned

2. Embeddings (tying, decay)

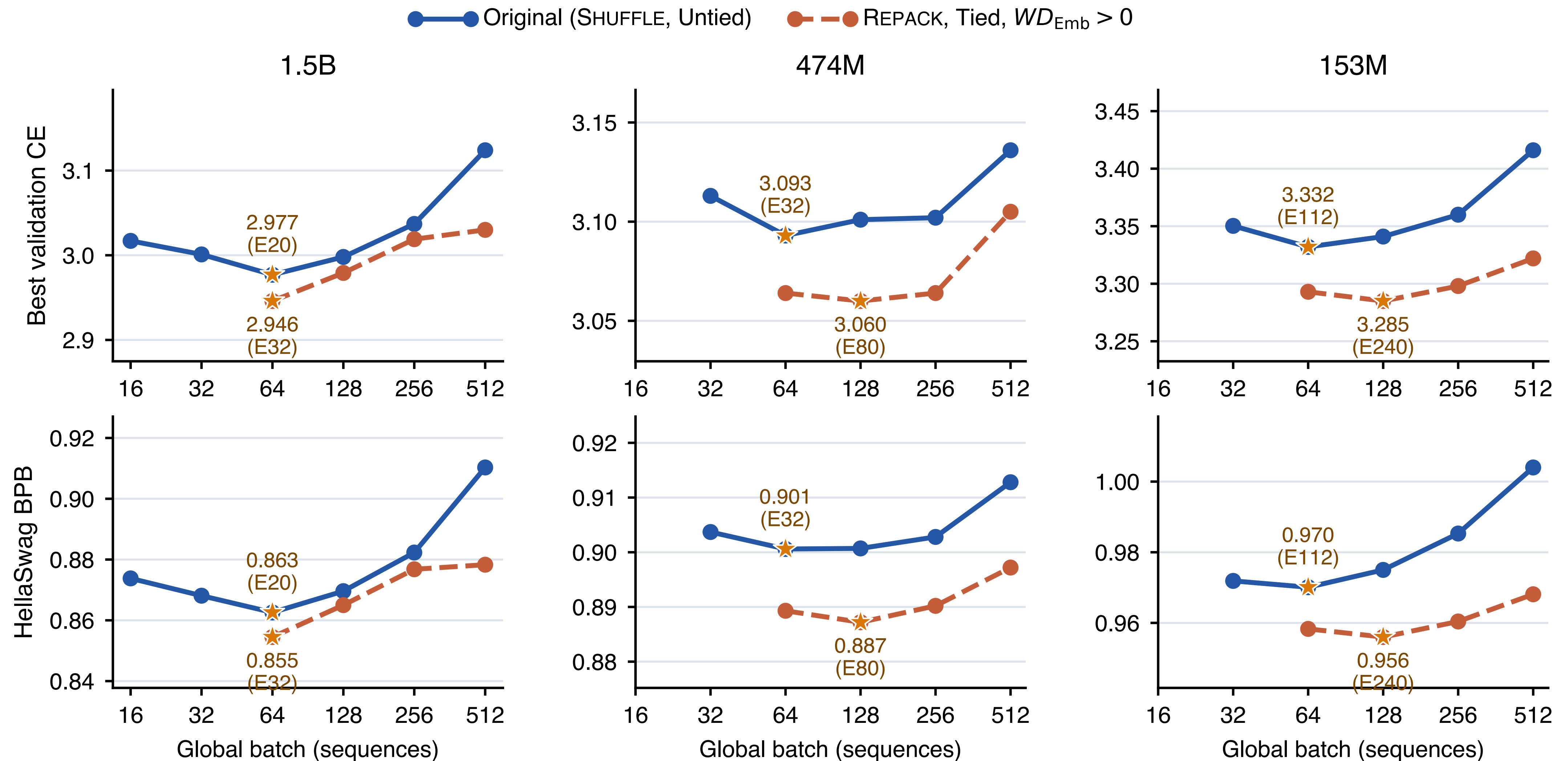


Turns out: weight tying & weight decay to Embed helps

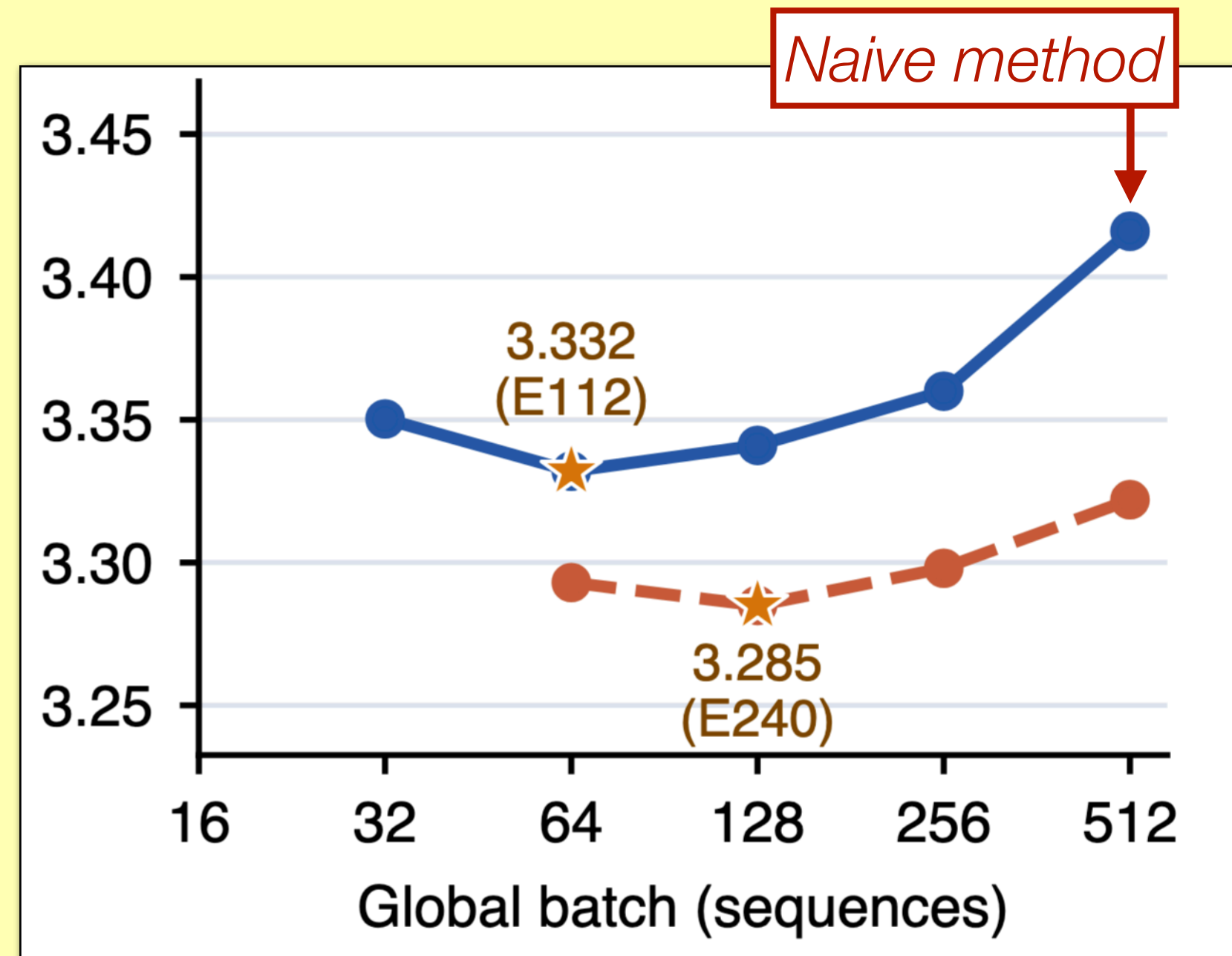
Finding 2: Recipe itself can be tuned



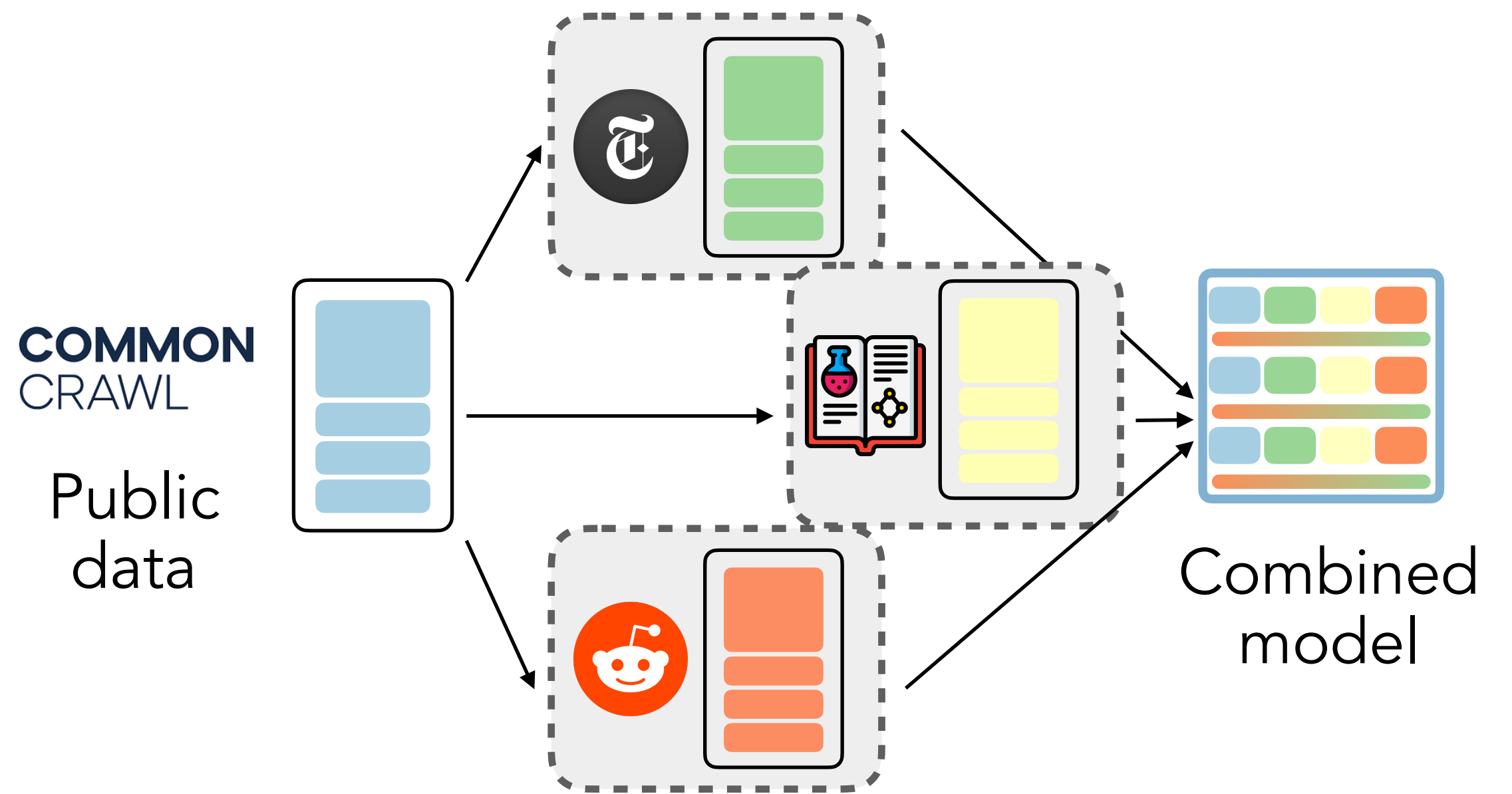
Finding 2: Recipe itself can be tuned



This Talk

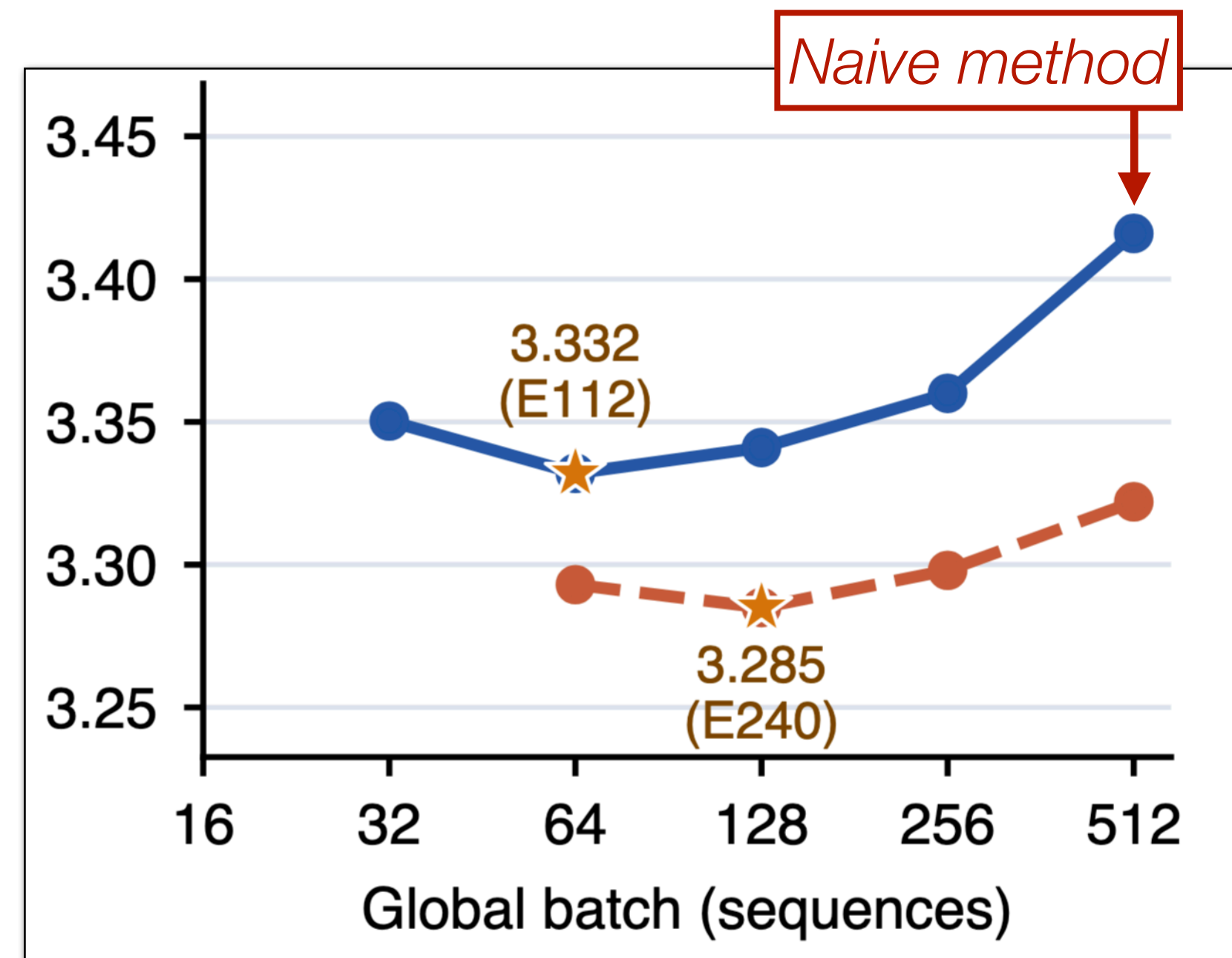


1. How to do repeated-data training *correctly*

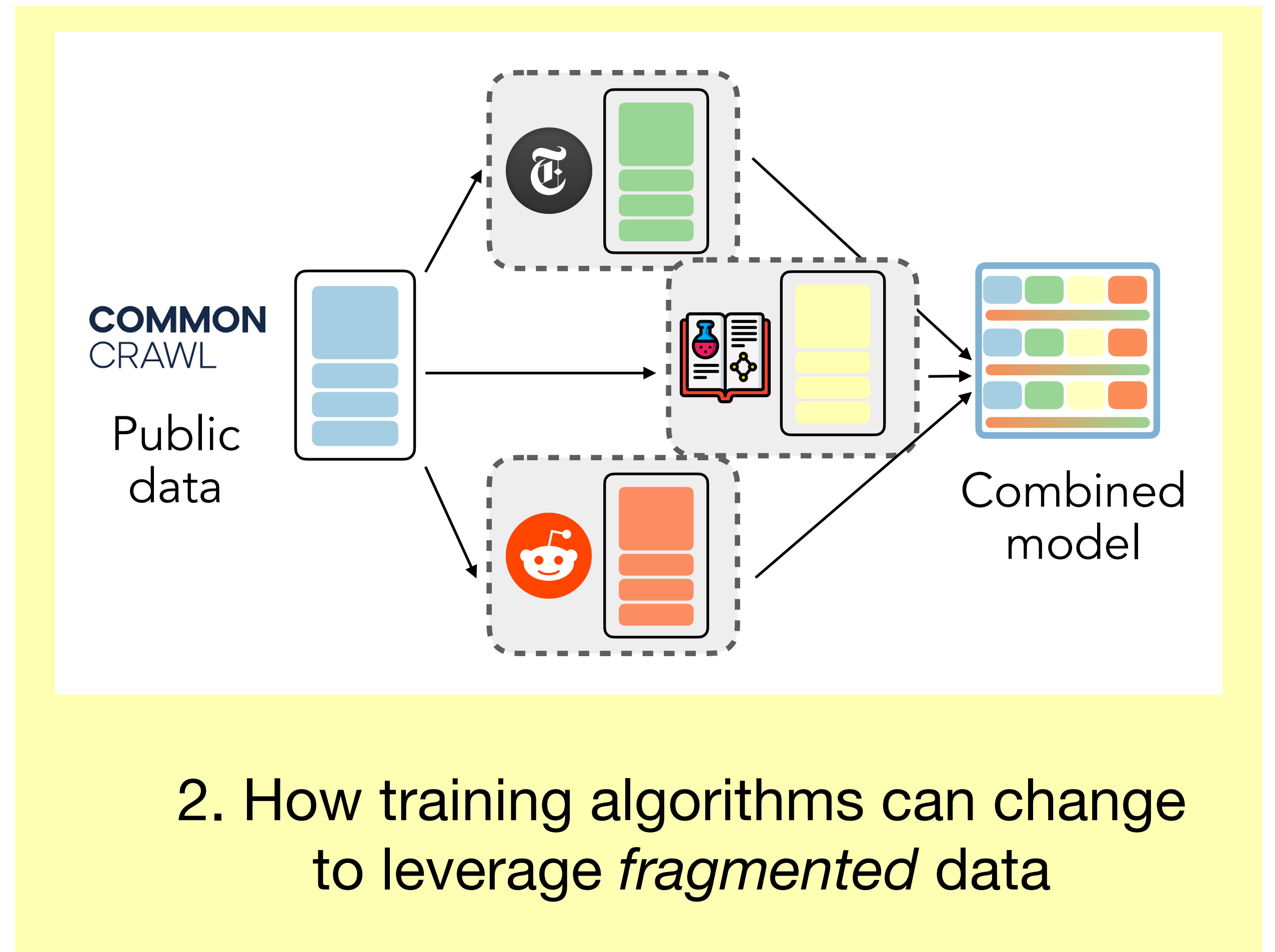


2. How training algorithms can change to leverage *fragmented* data

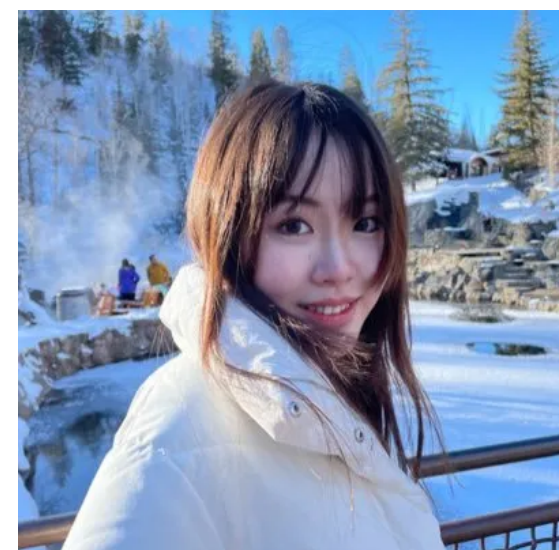
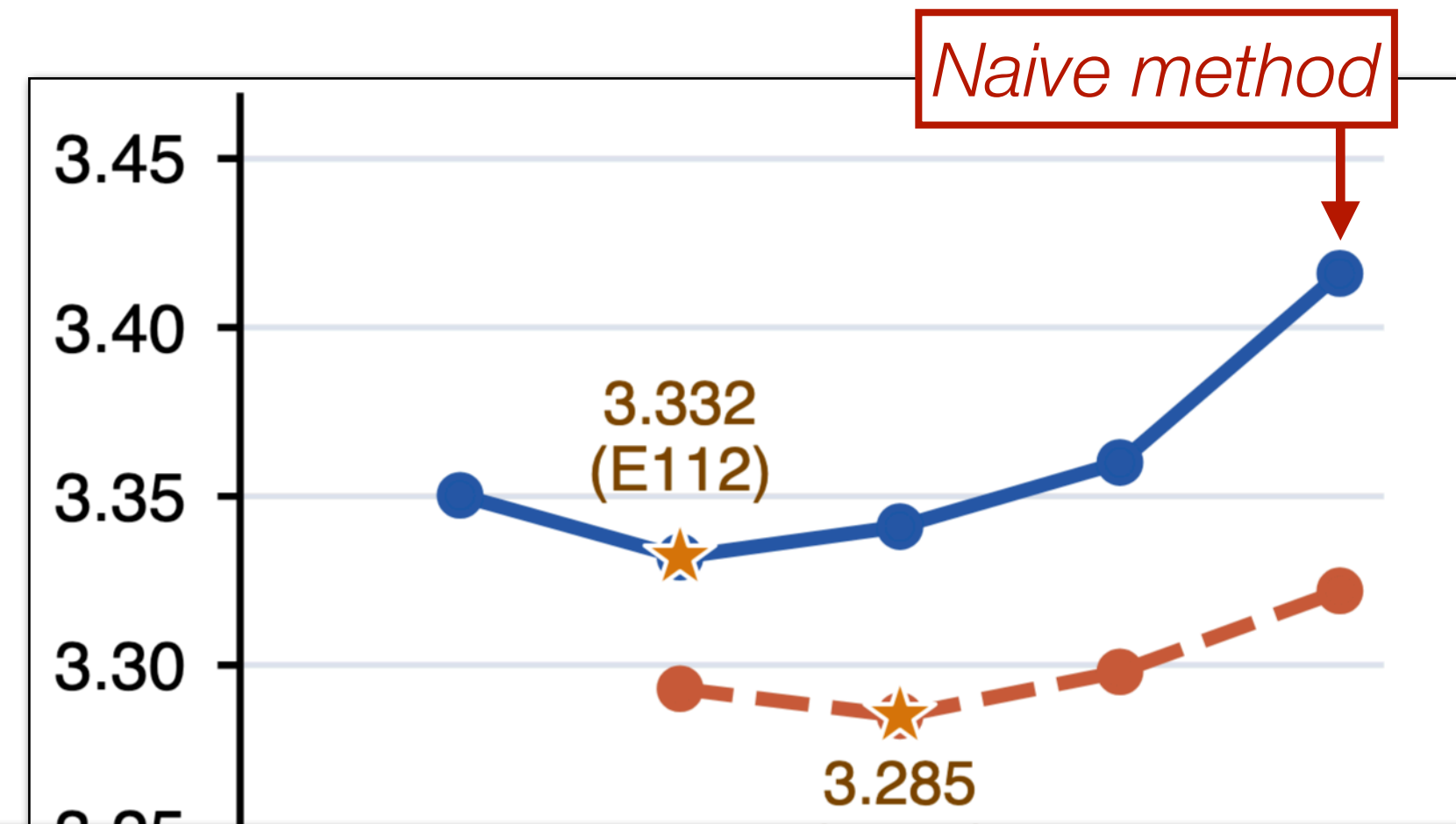
This Talk



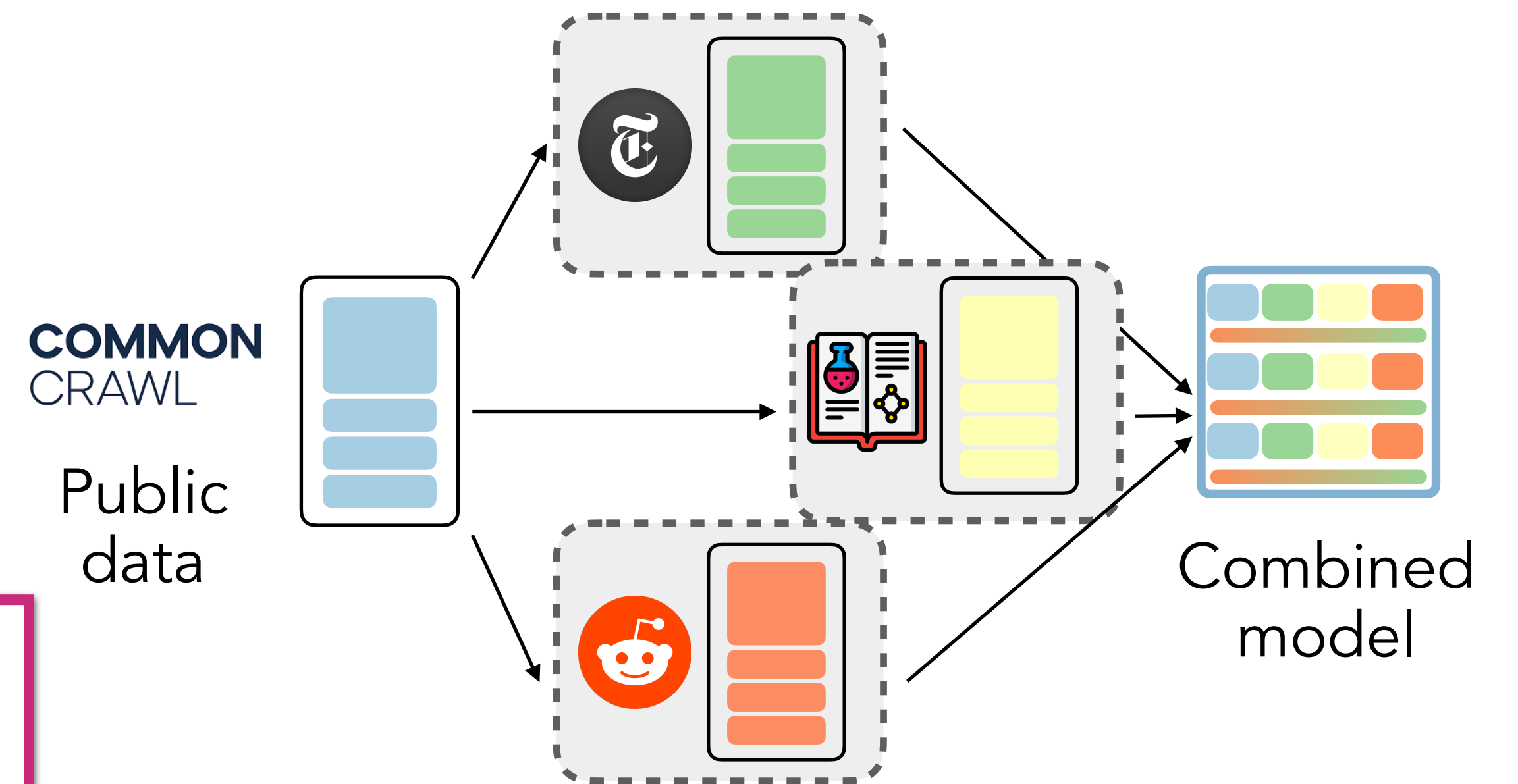
1. How to do repeated-data training *correctly*



This Talk



w/ Weijia Shi, Akshita Bhagia, Kevin Farhat + Other collaborators



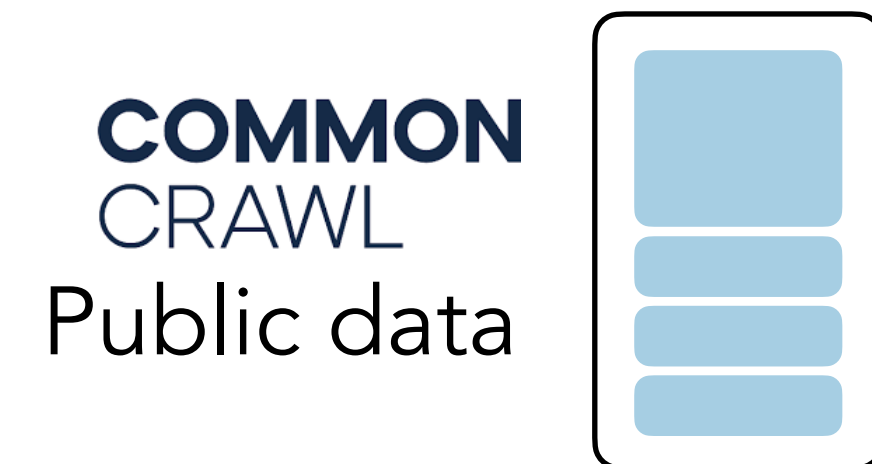
2. How training algorithms can change to leverage *fragmented* data

Data is Fragmented

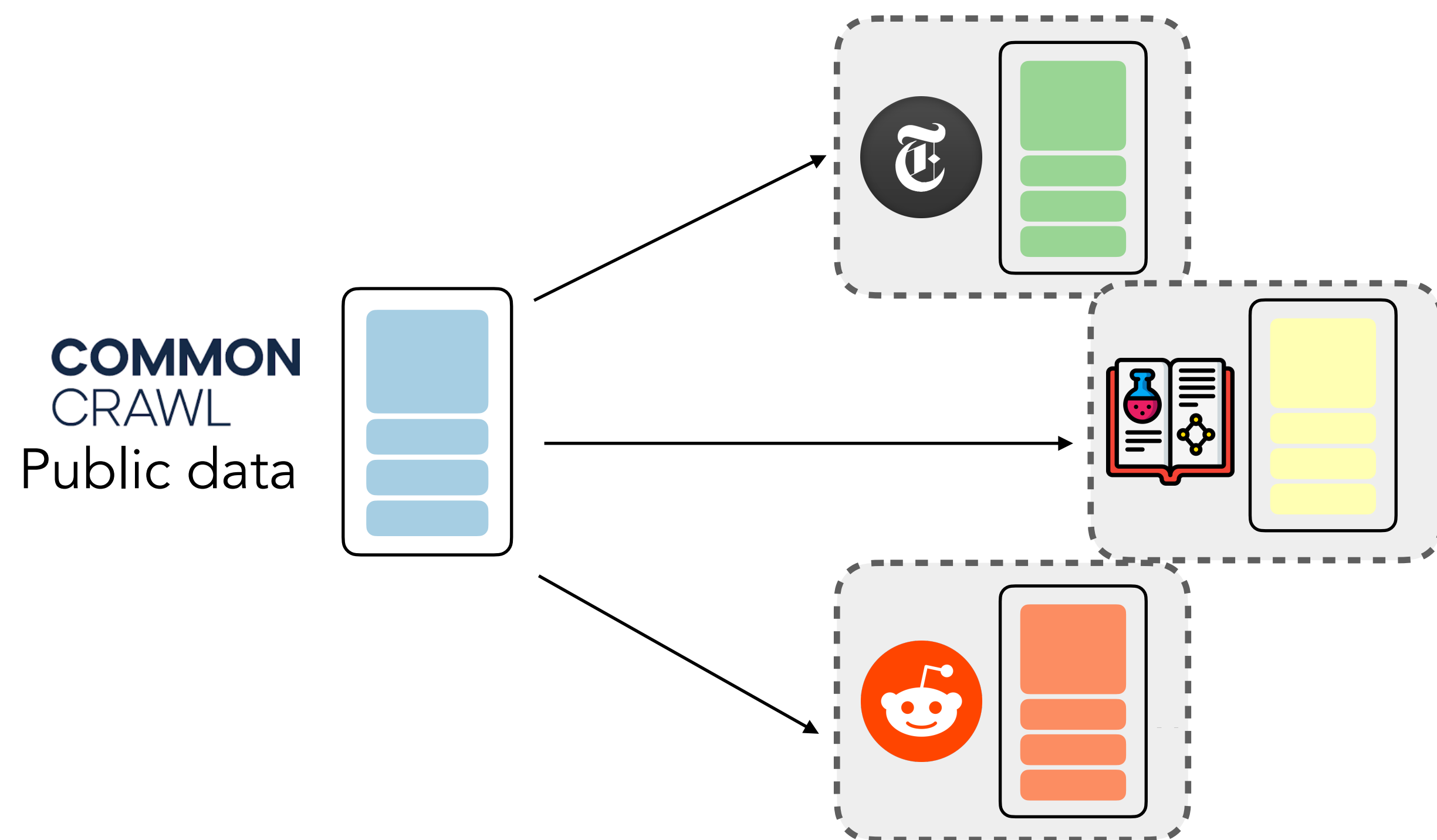


Valuable data is often restricted or proprietary (especially for highly specialized domains)

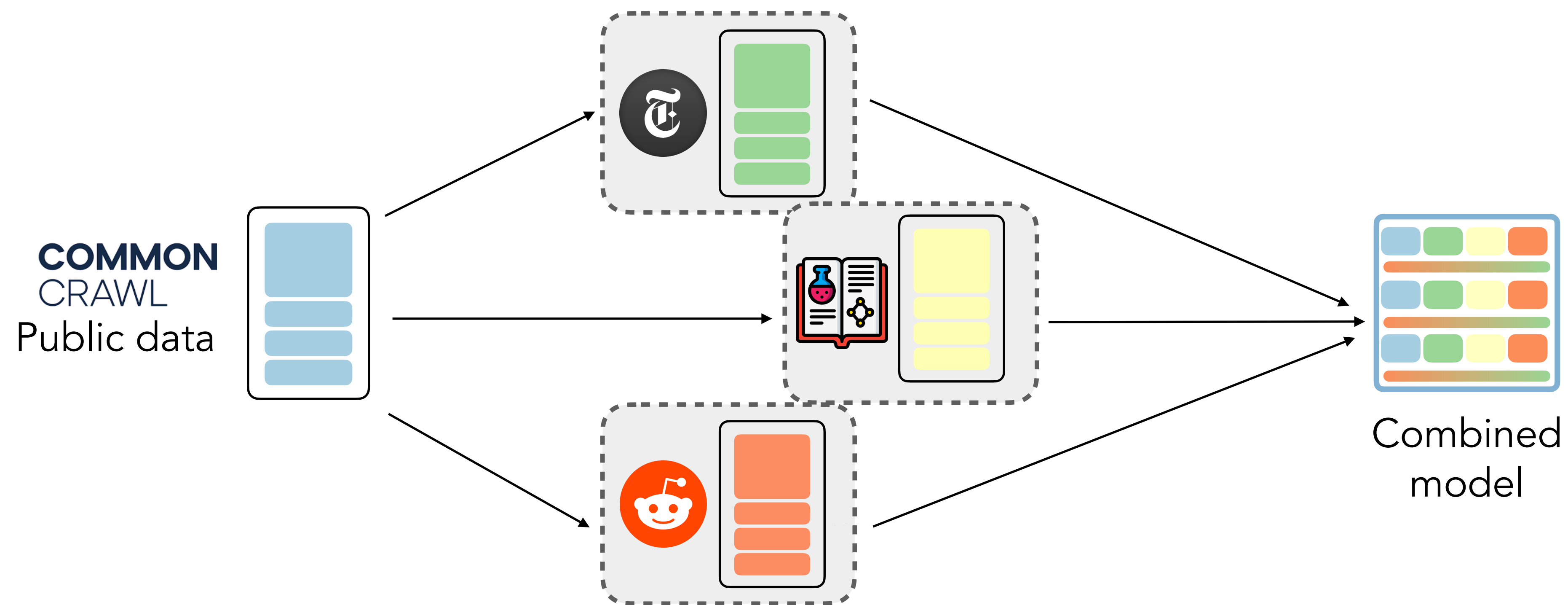
Proposal: Modular, Distributed Training



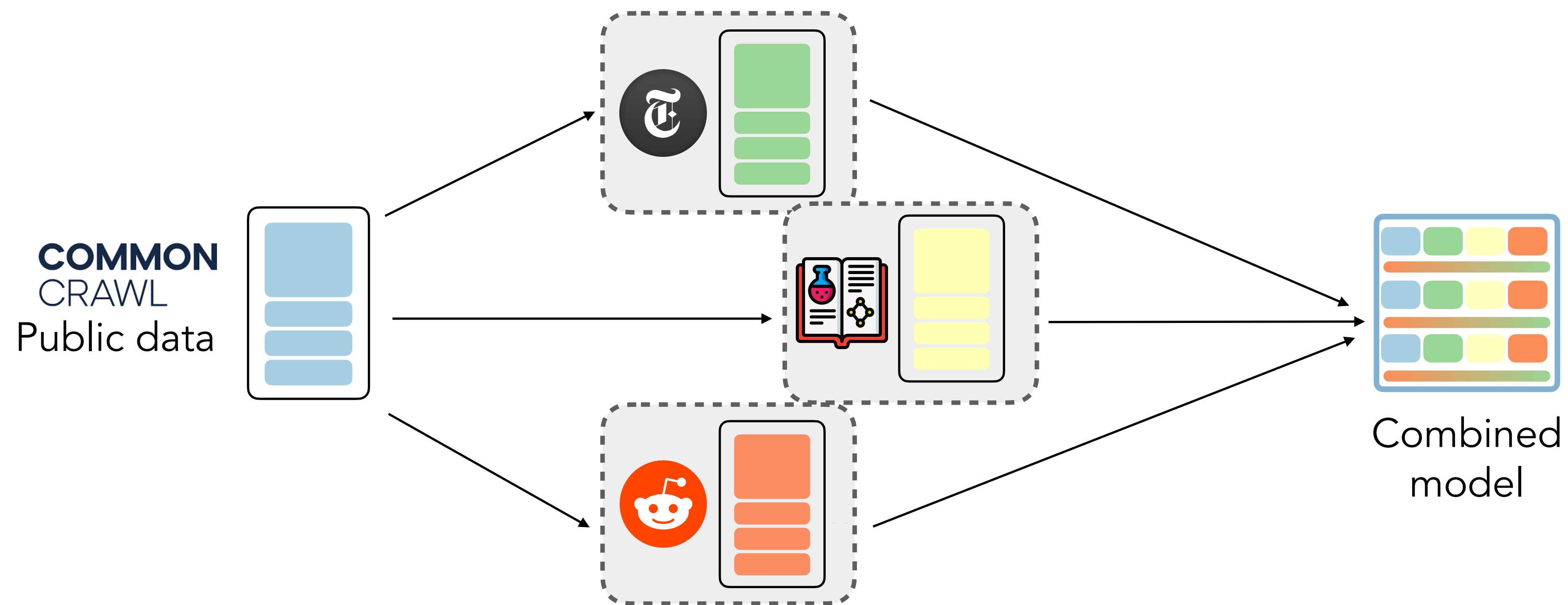
Proposal: Modular, Distributed Training



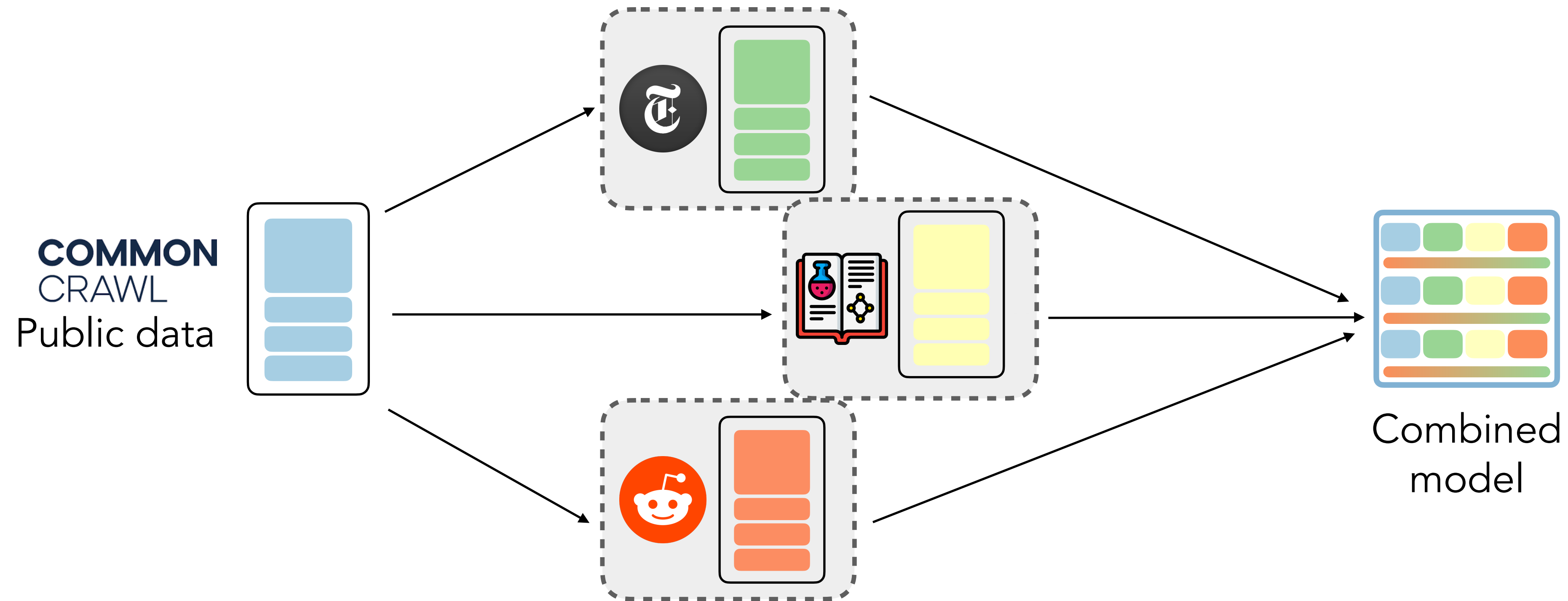
Proposal: Modular, Distributed Training



Proposal: Modular, Distributed Training

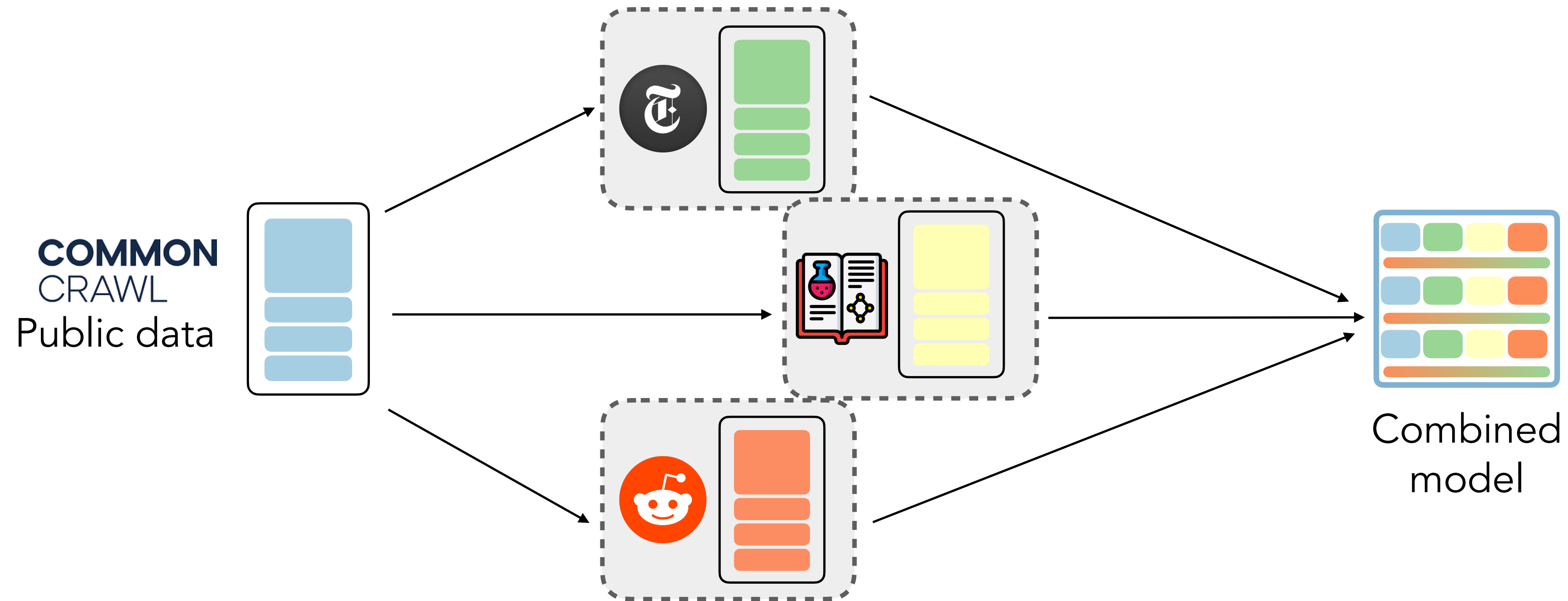


Proposal: Modular, Distributed Training



Collaborative training
without data sharing

Proposal: Modular, Distributed Training

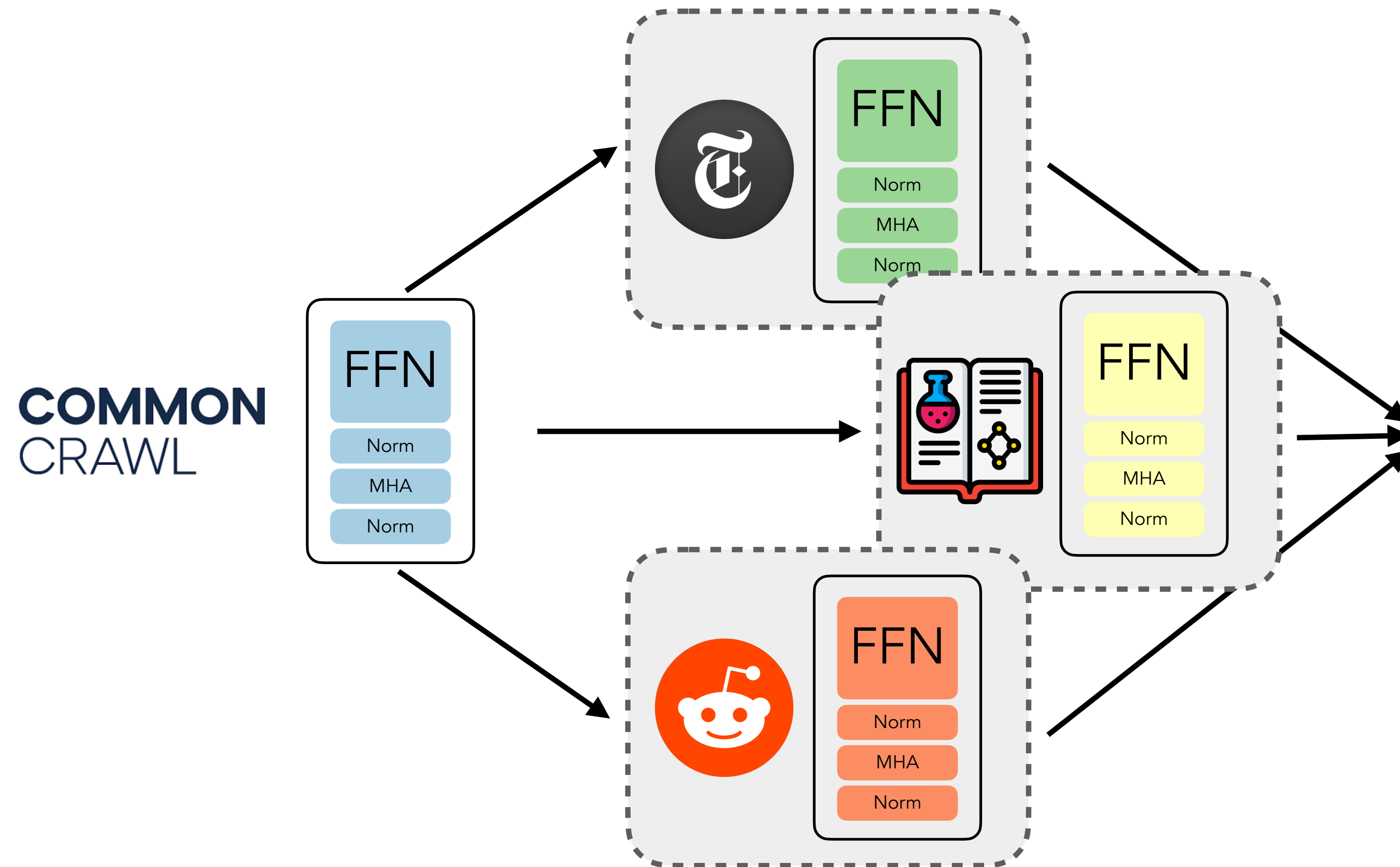


Collaborative training
without data sharing

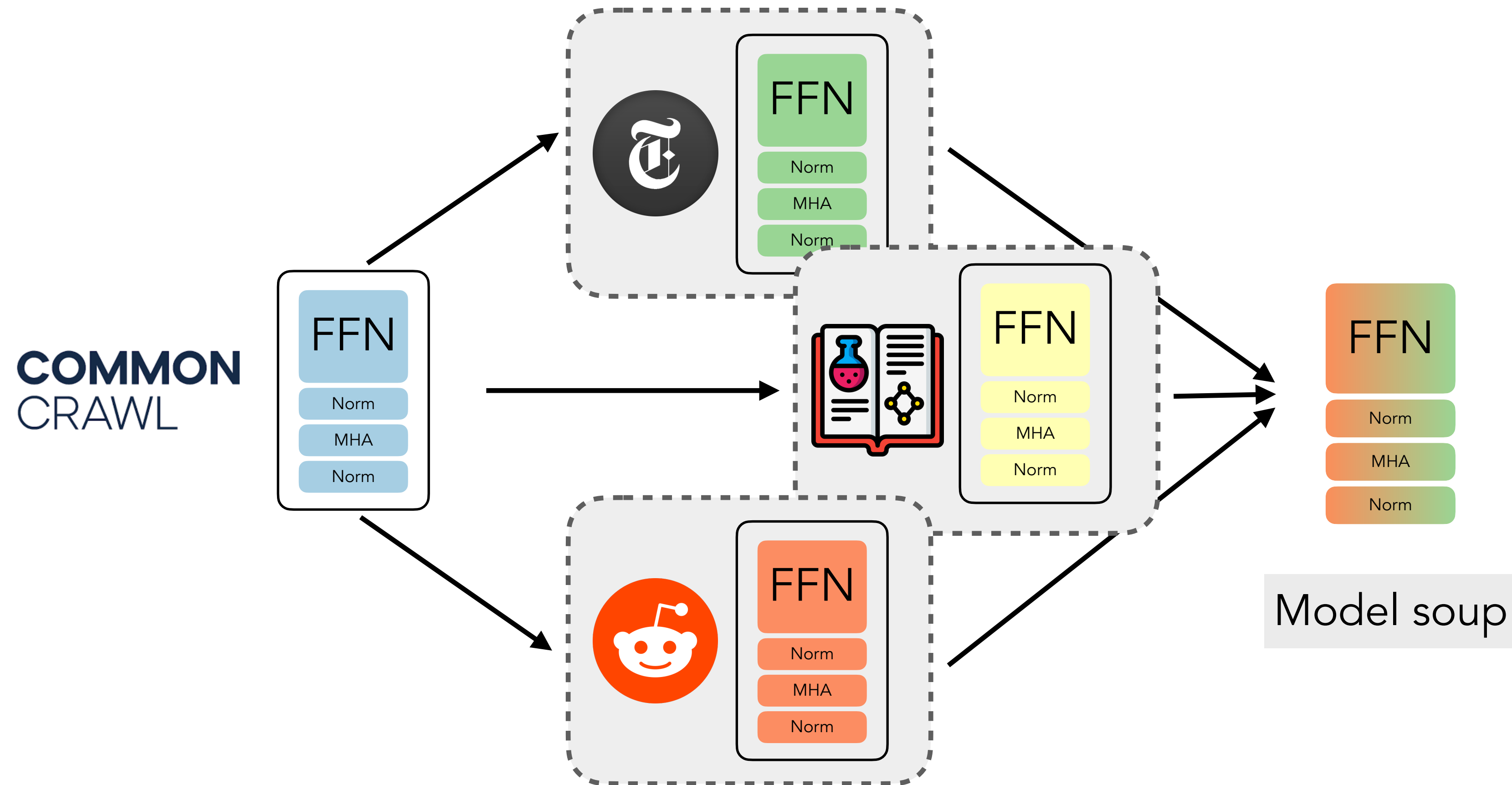
Can add/remove data
at any time

Prior work: Model merging

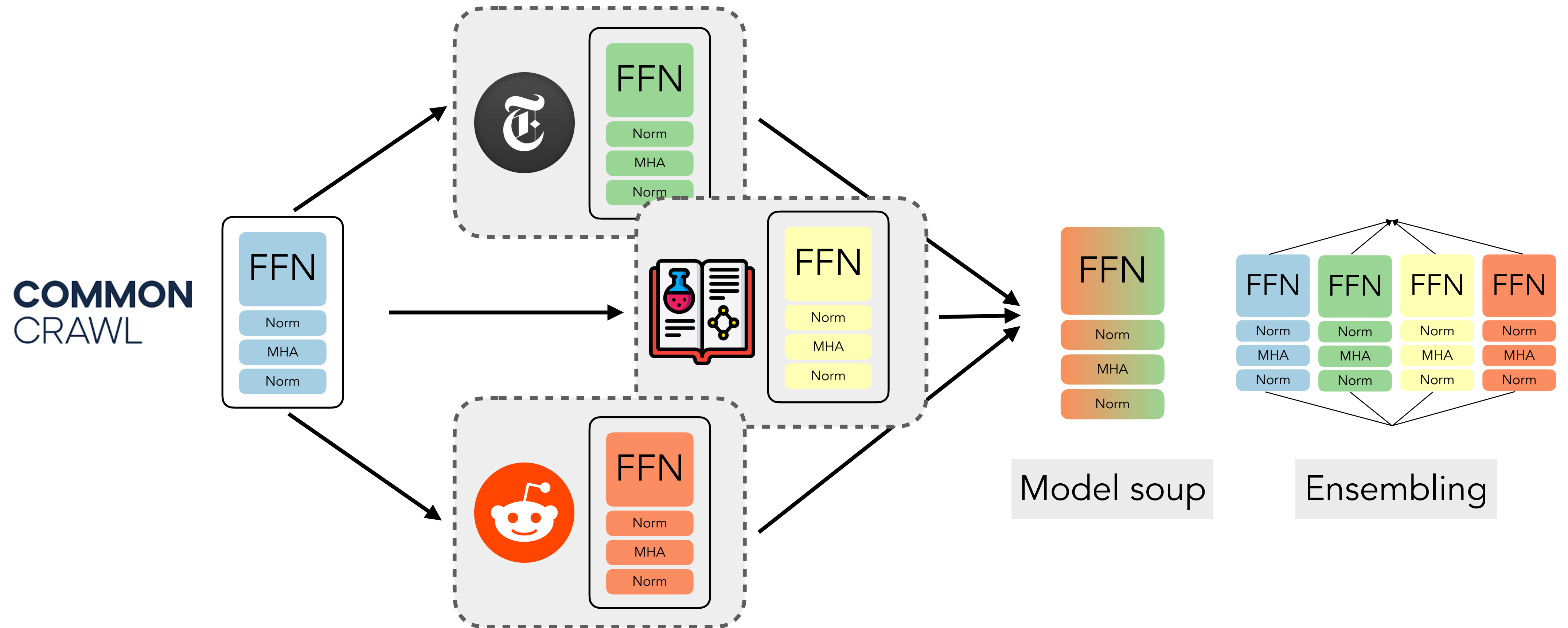
Prior work: Model merging



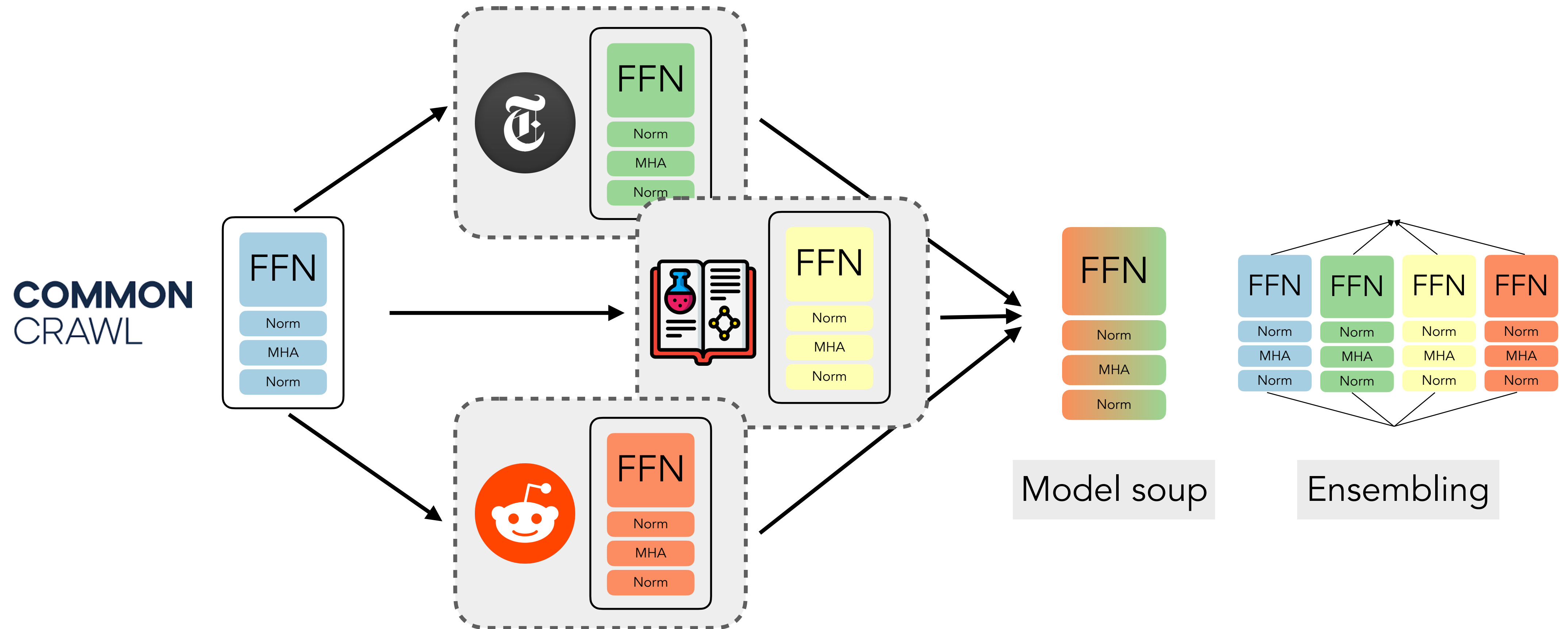
Prior work: Model merging



Prior work: Model merging

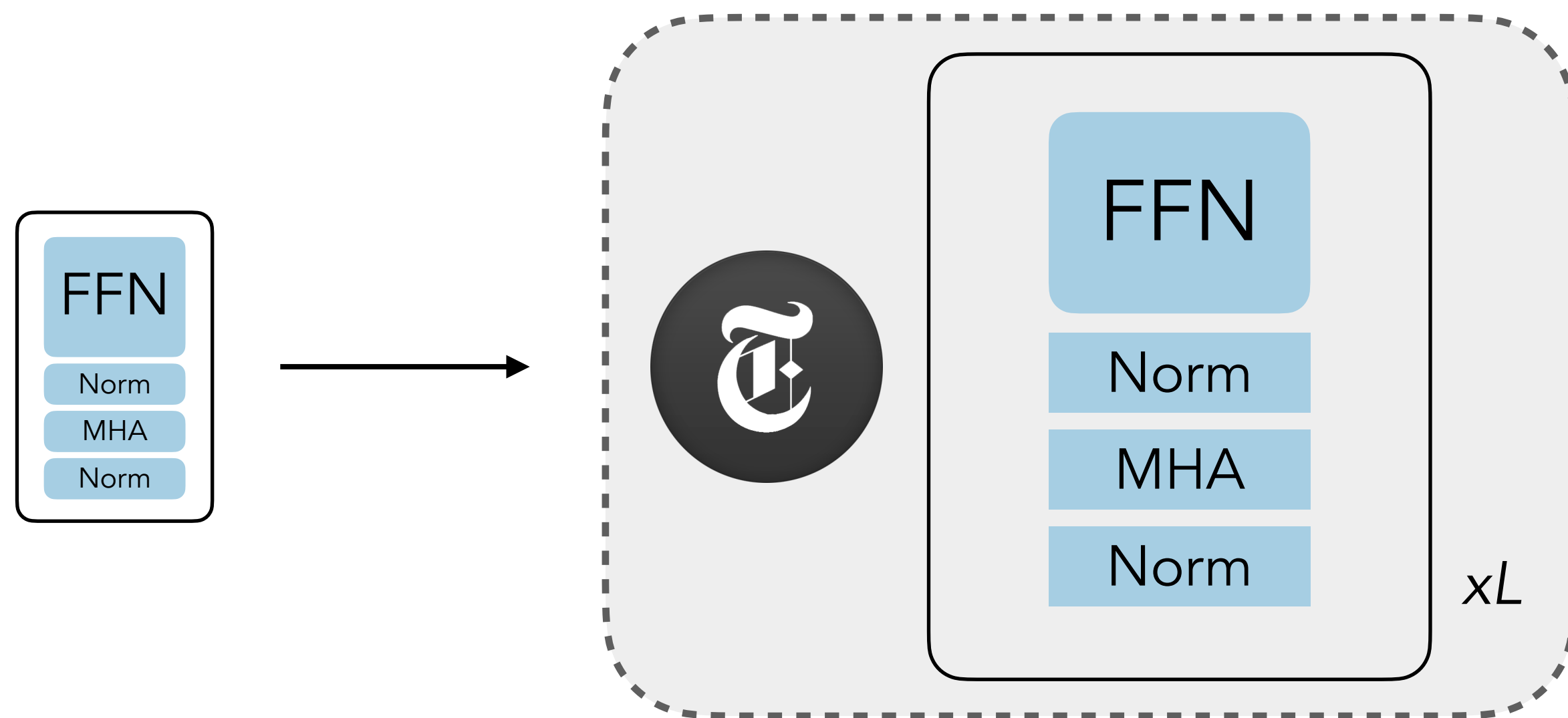


Prior work: Model merging

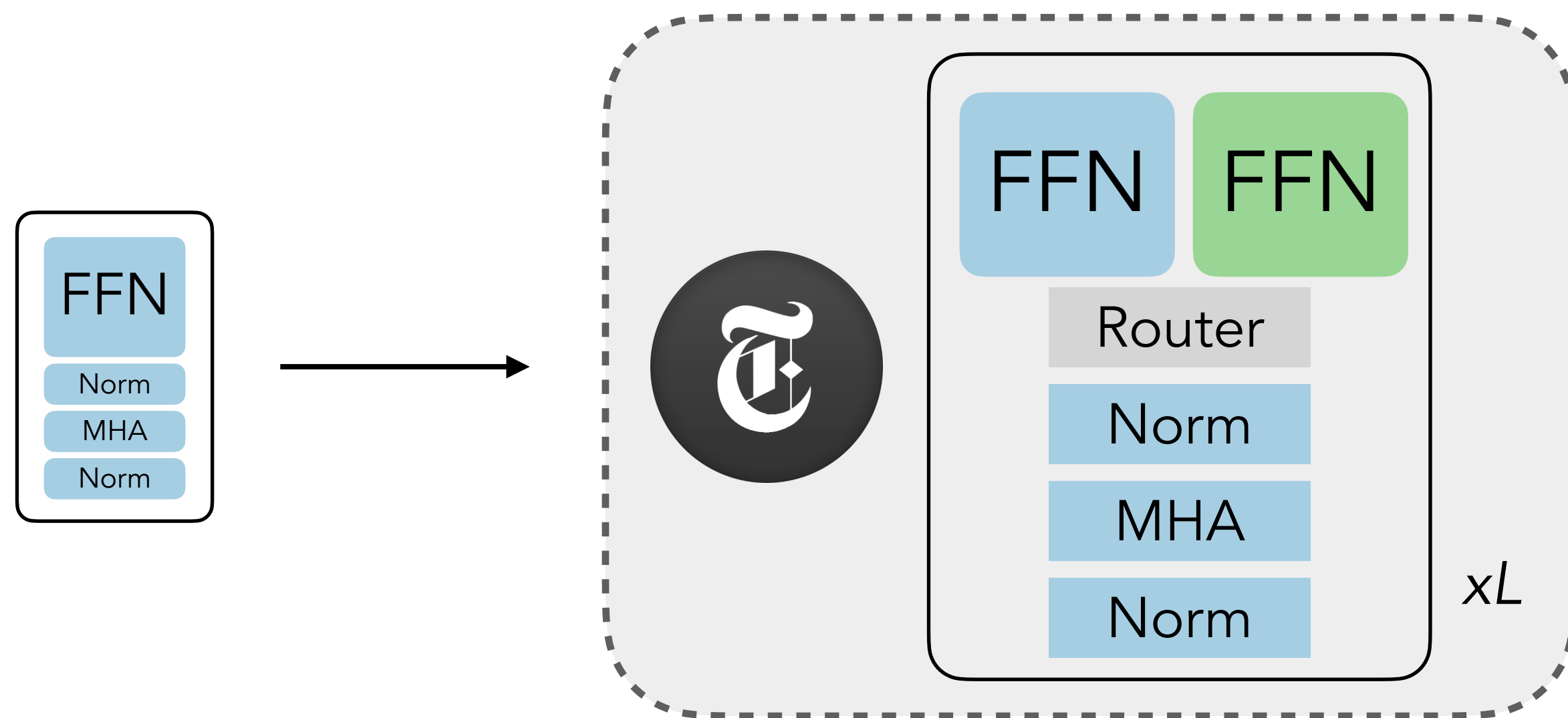


Merging models trained on disjoint datasets is harder

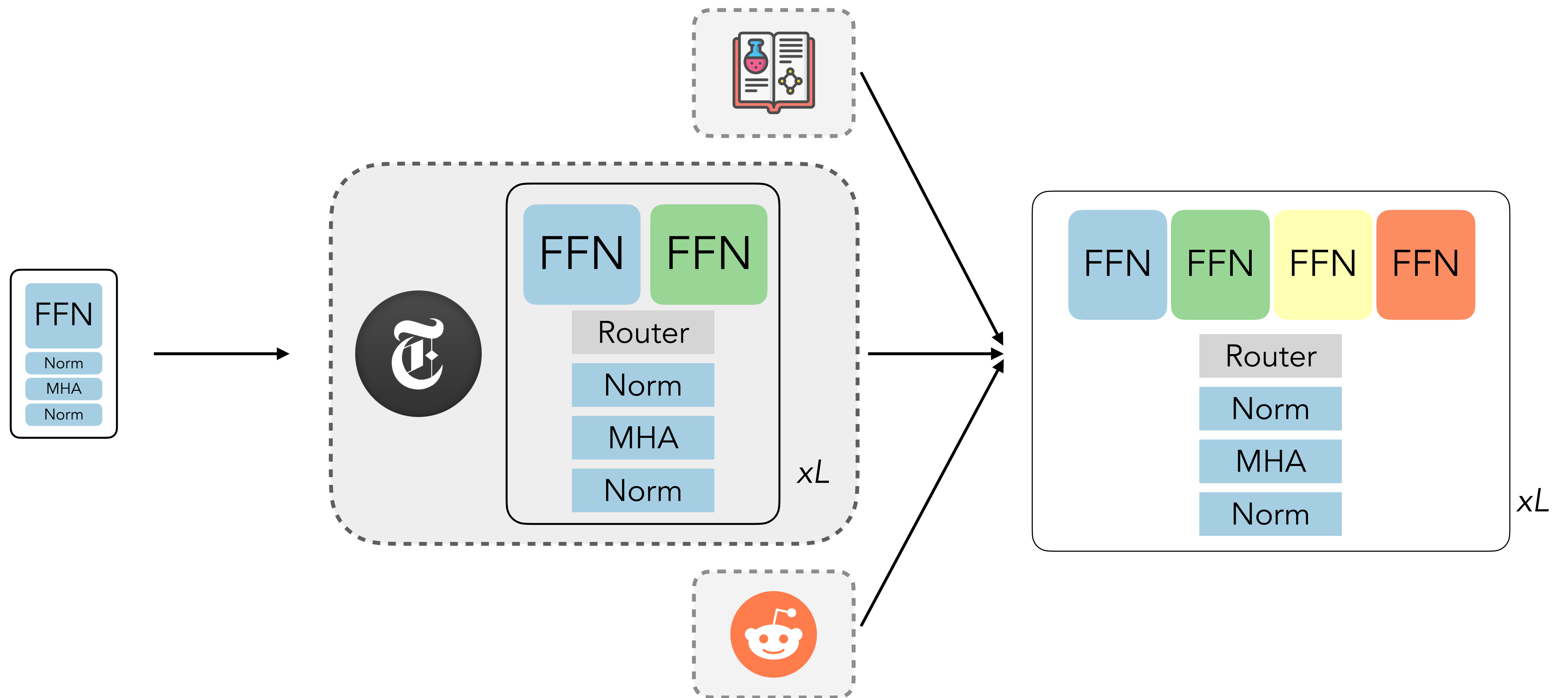
Our proposal: Leveraging MoE



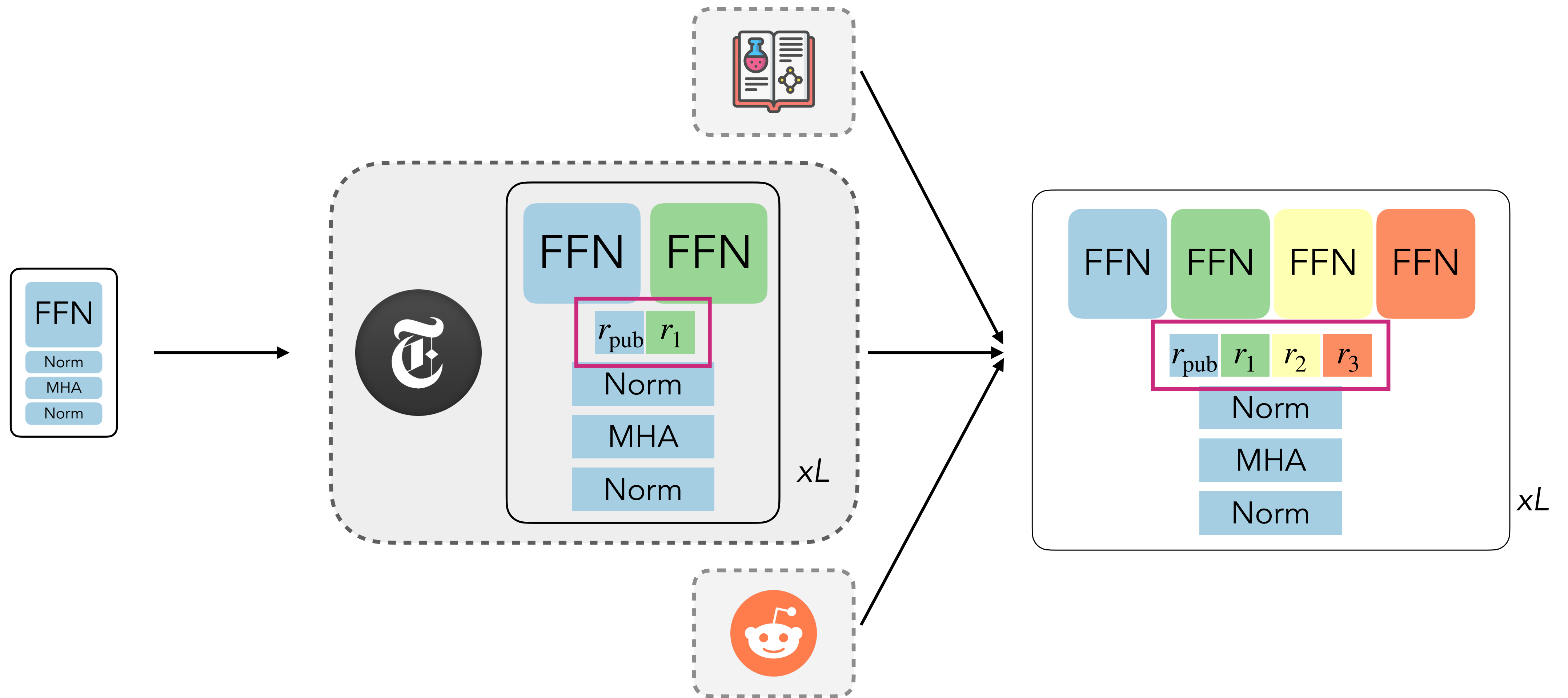
Our proposal: Leveraging MoE



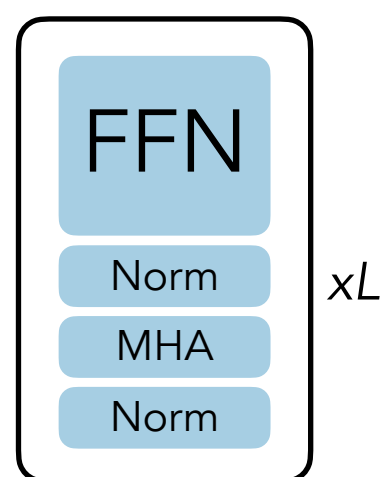
Our proposal: Leveraging MoE



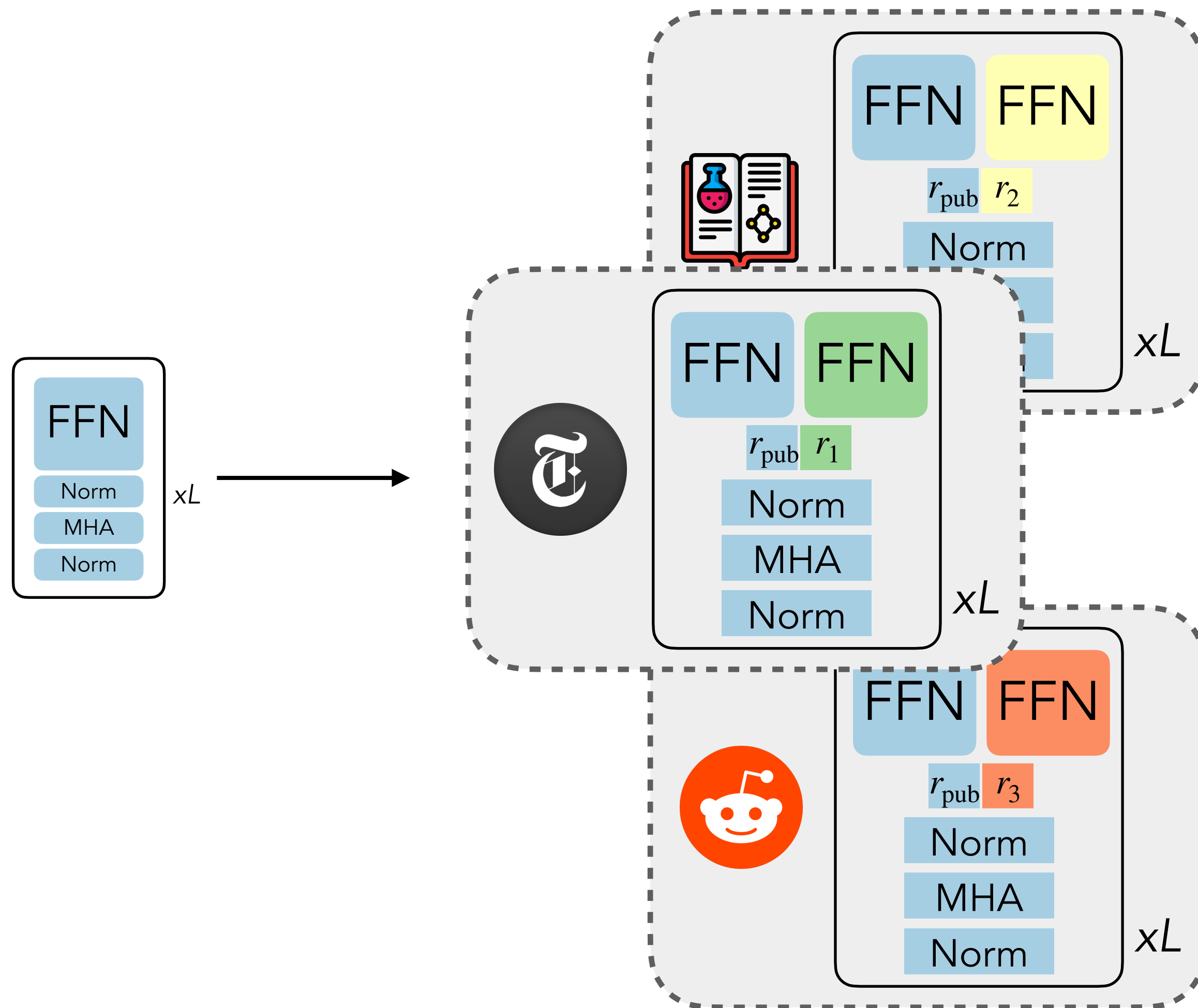
Our proposal: Router decomposition



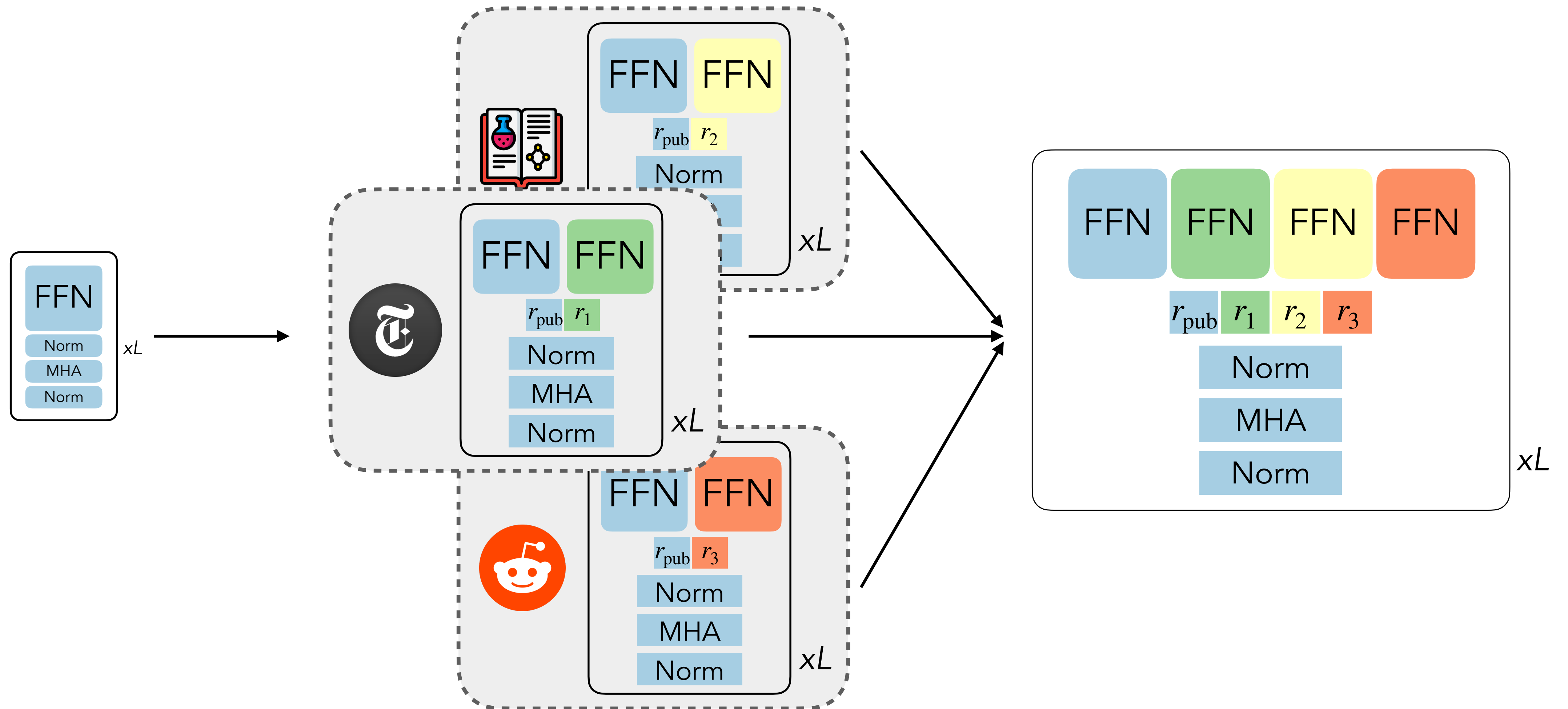
FlexOImo: Summary



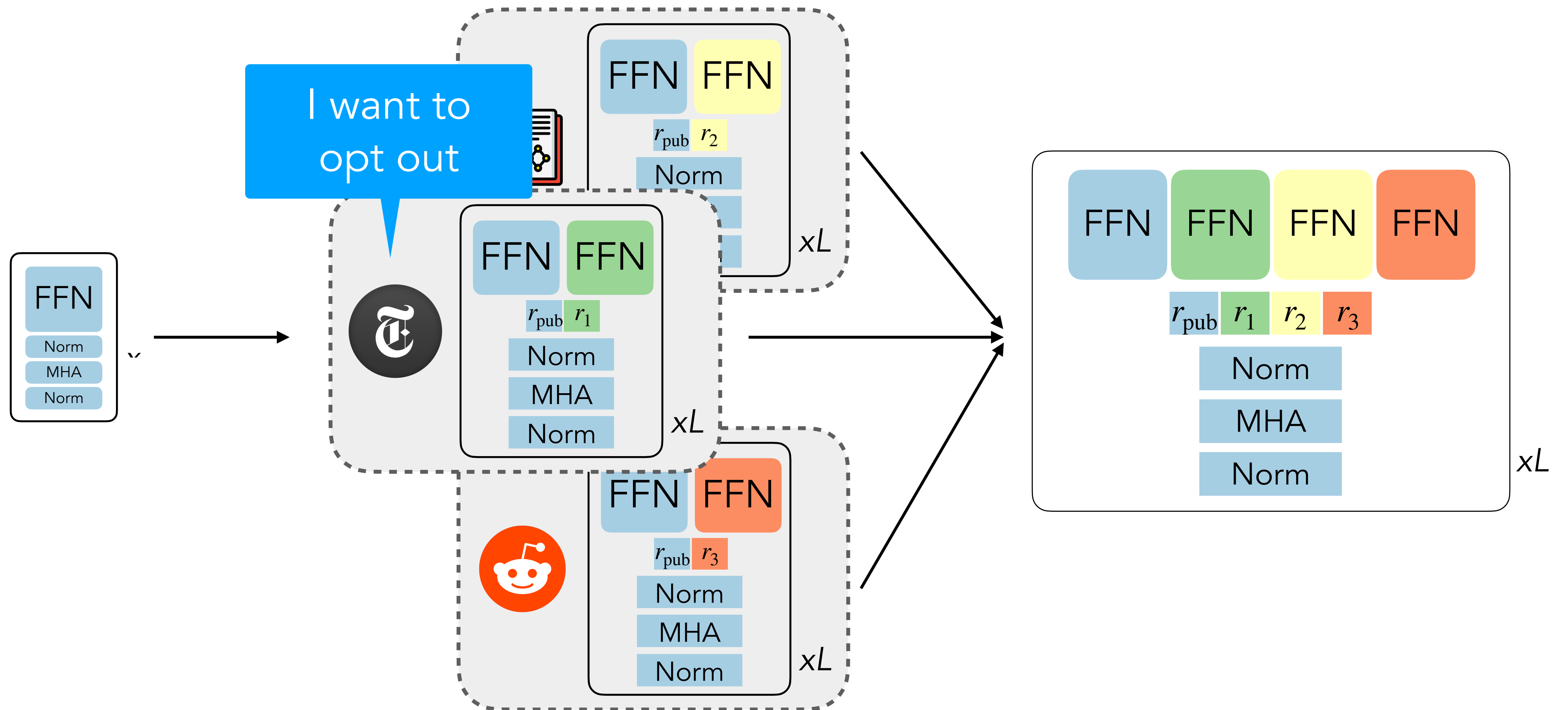
FlexOlmO: Summary



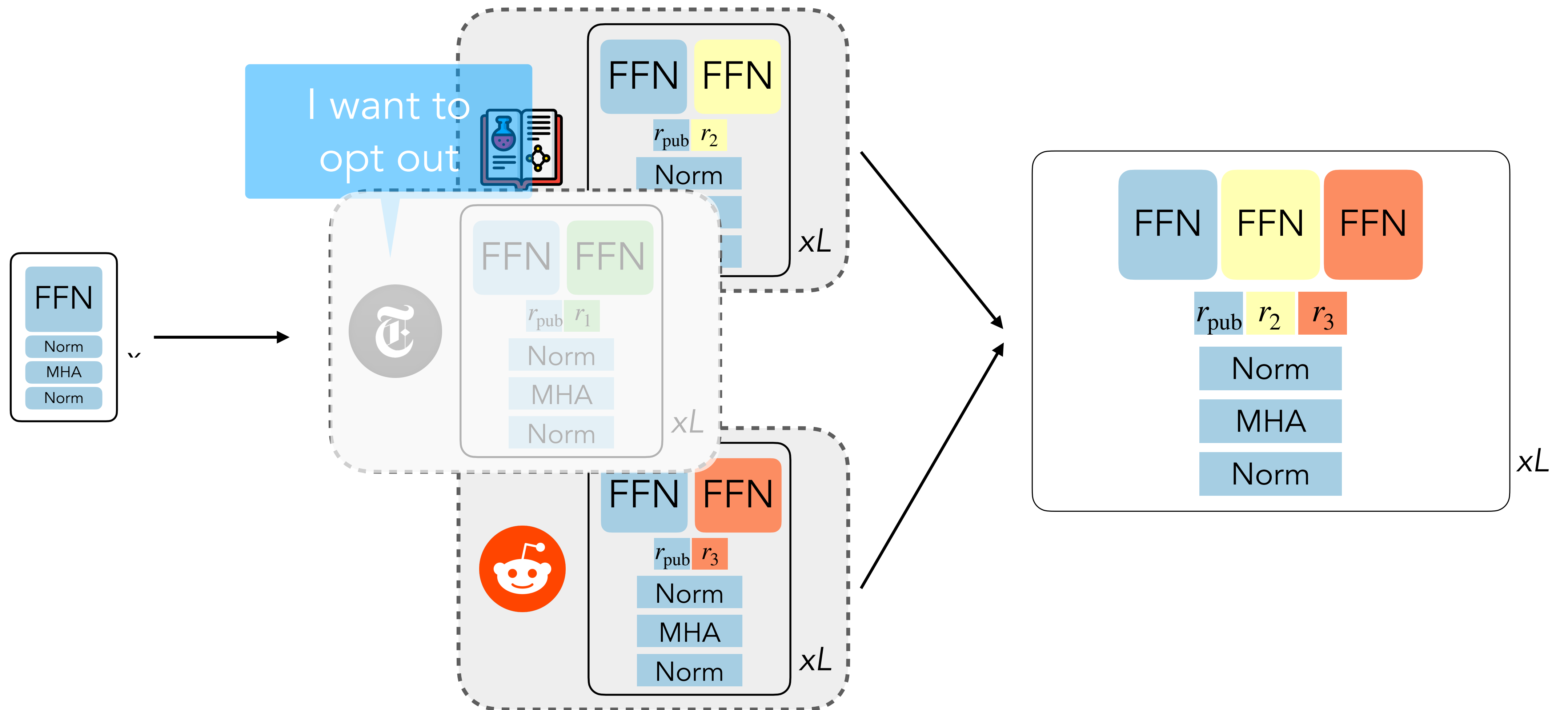
FlexOlmO: Summary

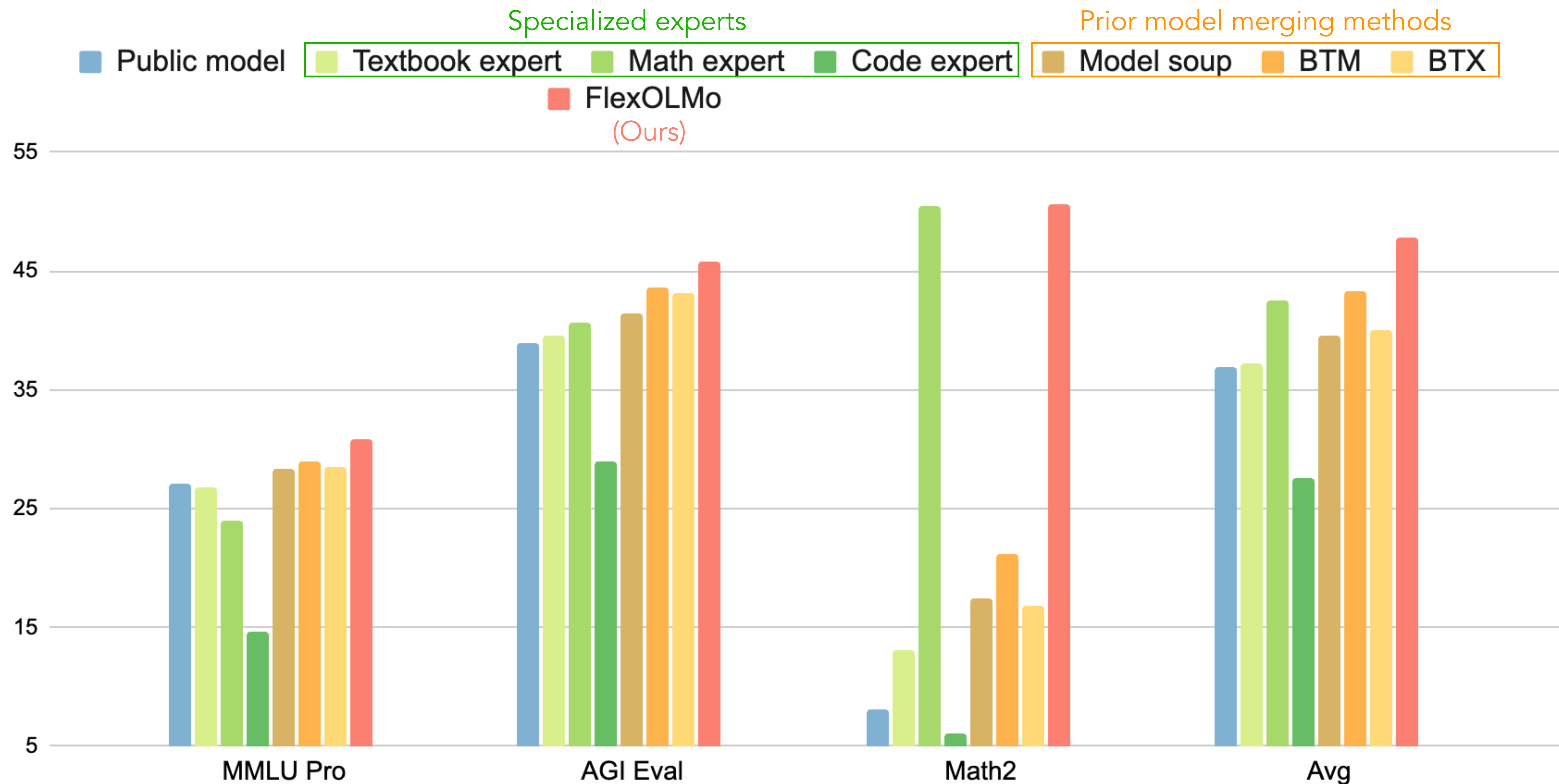


FlexOlm: Summary

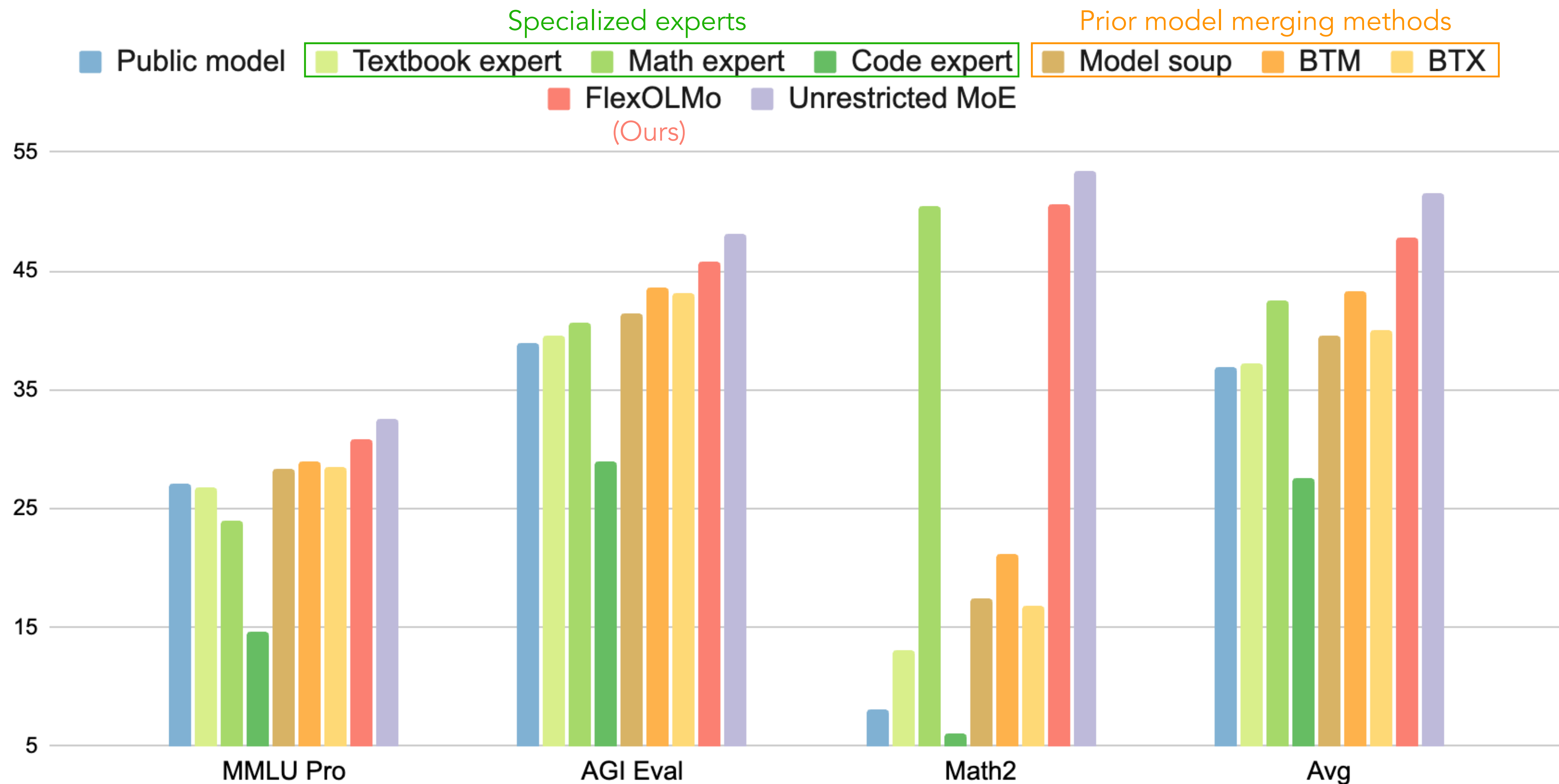


FlexOlm: Summary





FlexOLMo achieves 41% relative gain over prev public model



FlexOLMo retains 90% of the benefits of fully unrestricted training

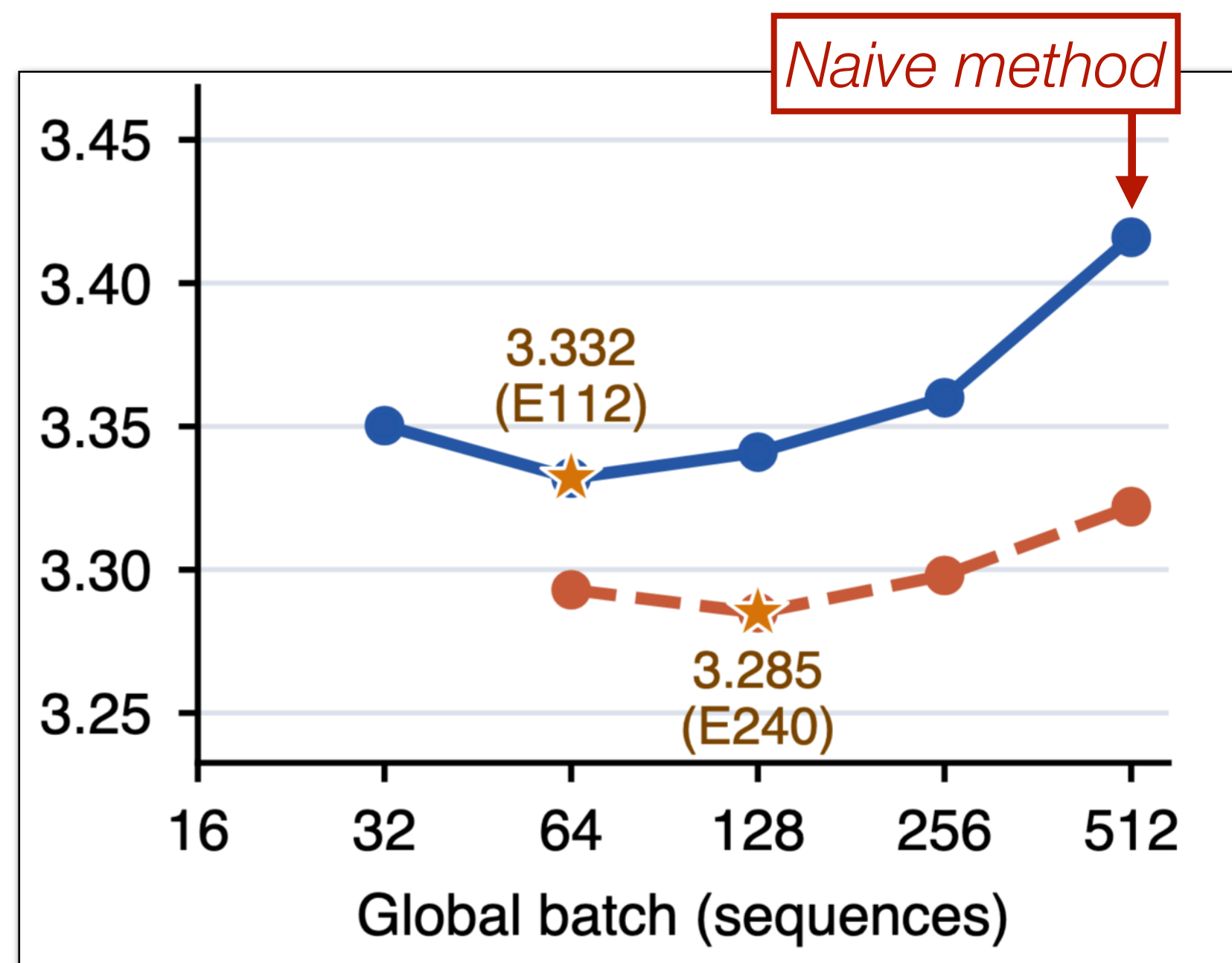
Wrapping Up

Wrapping Up

- Previously: Compute is a binding resource; data is nearly infinite
- Now: Compute is getting cheaper, but data is running out!

Wrapping Up

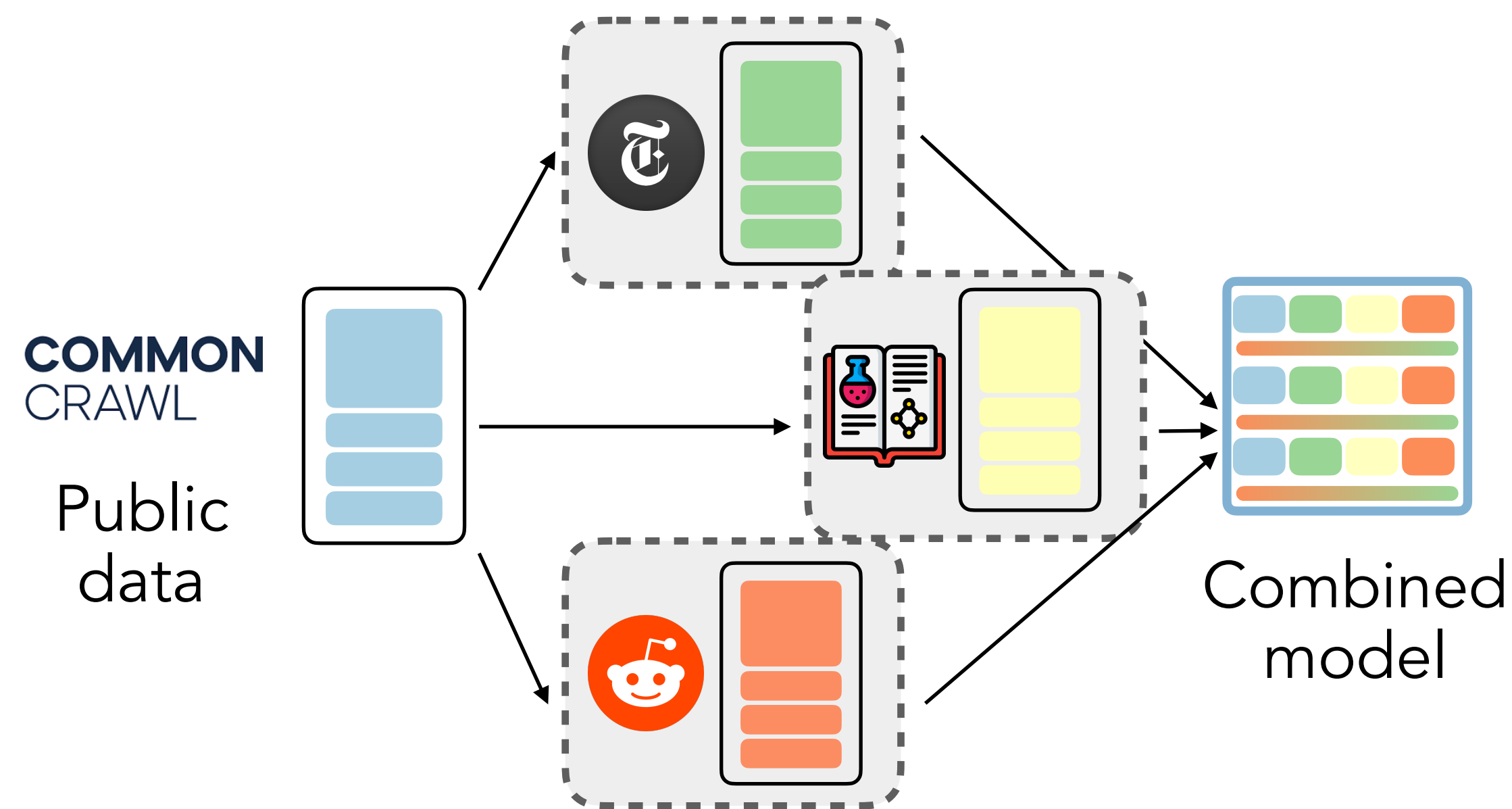
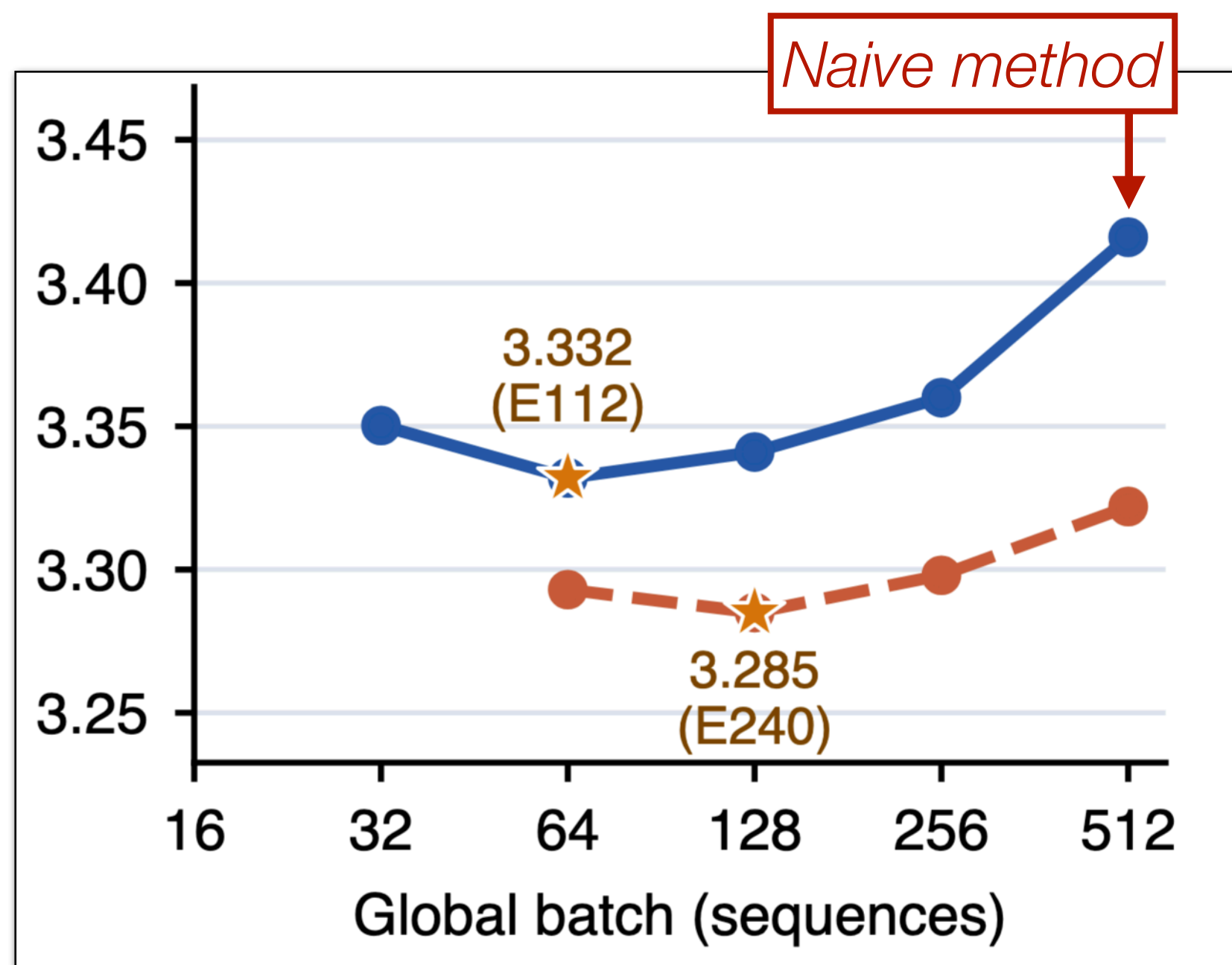
- Previously: Compute is a binding resource; data is nearly infinite
- Now: Compute is getting cheaper, but data is running out!



1. How to do repeated-data training
correctly

Wrapping Up

- Previously: Compute is a binding resource; data is nearly infinite
- Now: Compute is getting cheaper, but data is running out!



1. How to do repeated-data training *correctly*

2. How training algorithms can change to leverage *fragmented* data

Thank you for listening!

Repeated-data training: arxiv.org/abs/2610.15965

FlexOlmo: allenai.org/papers/FlexOlmo



sewonmin.com



sewonm@berkeley.edu

Please leave feedback at tinyurl.com/sewonm-talk