2024-Spring Natural Language Processing (CSW6035)

Generative Representational Instruction Tuning

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang,
Furu Wei, Tao Yu, Amanpreet Singh, Douwe Kiela

Contextual AI, University of Hong Kong, Microsoft Corporation

Yejin Yoon

GRIT

Generative Representational Instruction Tuning

Contents

HYU 한양대학교 HANYANG UNIVERSITY

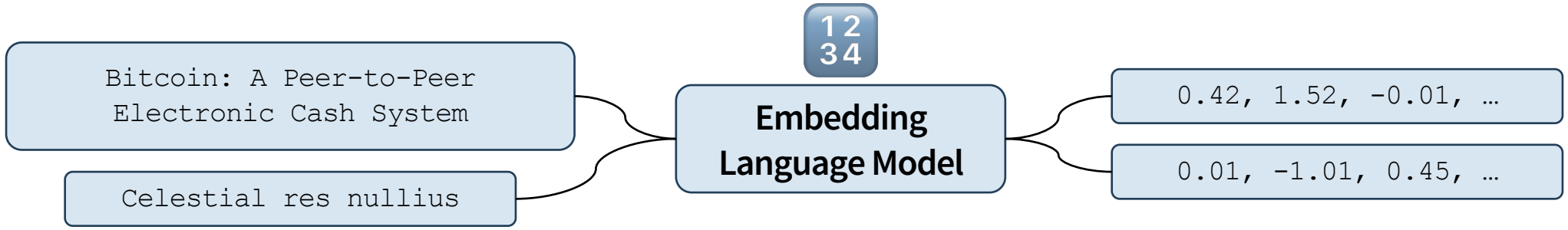
Introduction

Problem States

Background

2 types of Language Models (1/2)

- **Embedding Language Model**
  - a type of model that transforms high-dimensional categorical data, like words, into lower-dimensional, continuous vectors (=embeddings)
  - capture the semantic properties of the input data so that similar inputs are close to each other in the embedding space.



BAAI 智源研究院 BGE

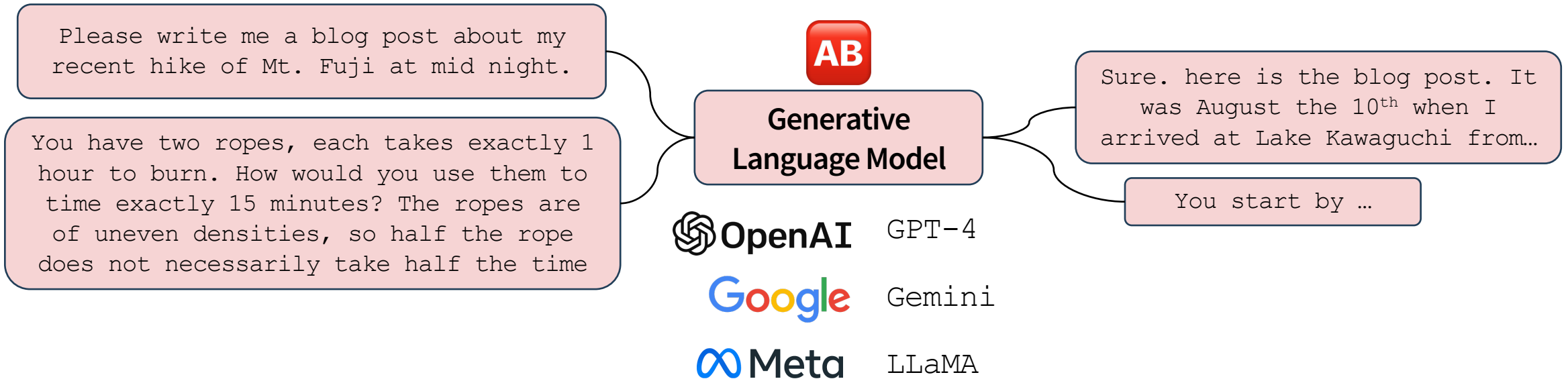
OpenAI text-embedding-3

Microsoft E5

Text Classification, Clustering, Semantic Search, …

2 types of Language Models (2/2)

- **Generative Language Model**
  - a type of model that is capable of generating text.
  - predict the probability of each word in a sequence, given the previous words, and can generate text by sampling from these probabilities.



Story Generation, Question Answering, Chat, …

Why Integrate Them into One Model?

- **Advantages of Combining Them**

Performance

Get better on both?

1 2
3 4

Embedding Benchmarks:
MTEB, …

AB

Generation Benchmarks:
AplacaEval, …

AlpacaFarm

Why Integrate Them into One Model?

- **Advantages of Combining Them**

Performance

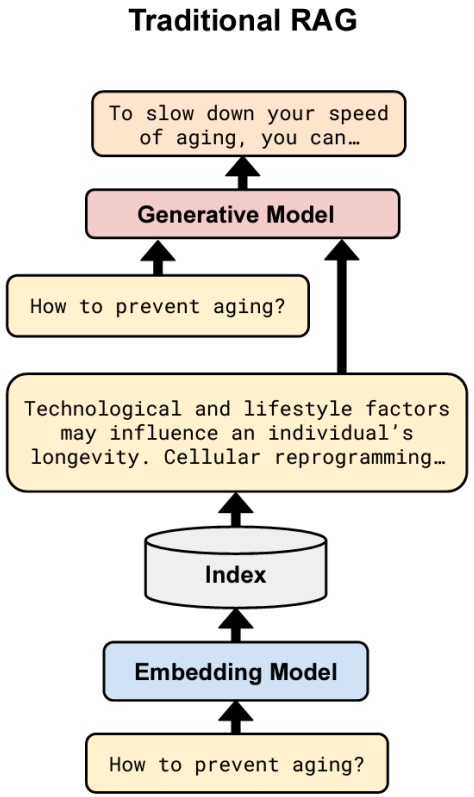
Get better on both?

Embedding Benchmarks:
MTEB, …

Generation Benchmarks:
AplacaEval, …

Efficiency

Speed-up joint use cases



Why Integrate Them into One Model?

- **Advantages of Combining Them**

Performance

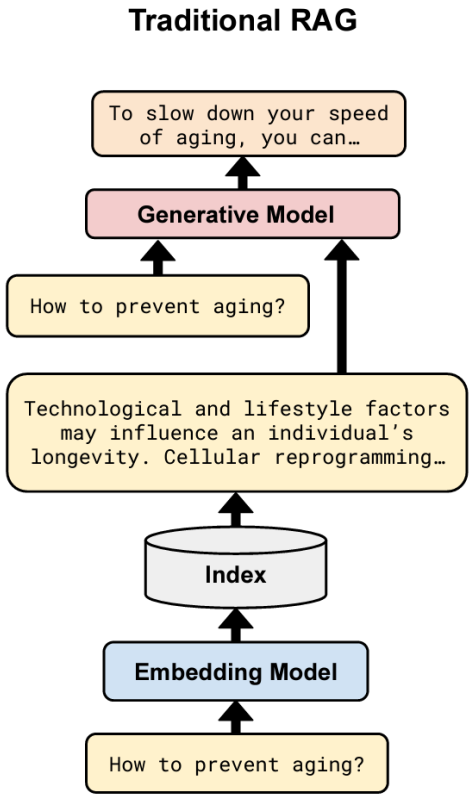
Get better on both?

1 2 3 4 **Embedding Benchmarks:** MTEB, …

AB **Generation Benchmarks:** AplacaEval, …

Efficiency

Speed-up joint use cases



Simplicity

Unify endpoints

1 2 3 4 **Embedding LM endpoints**

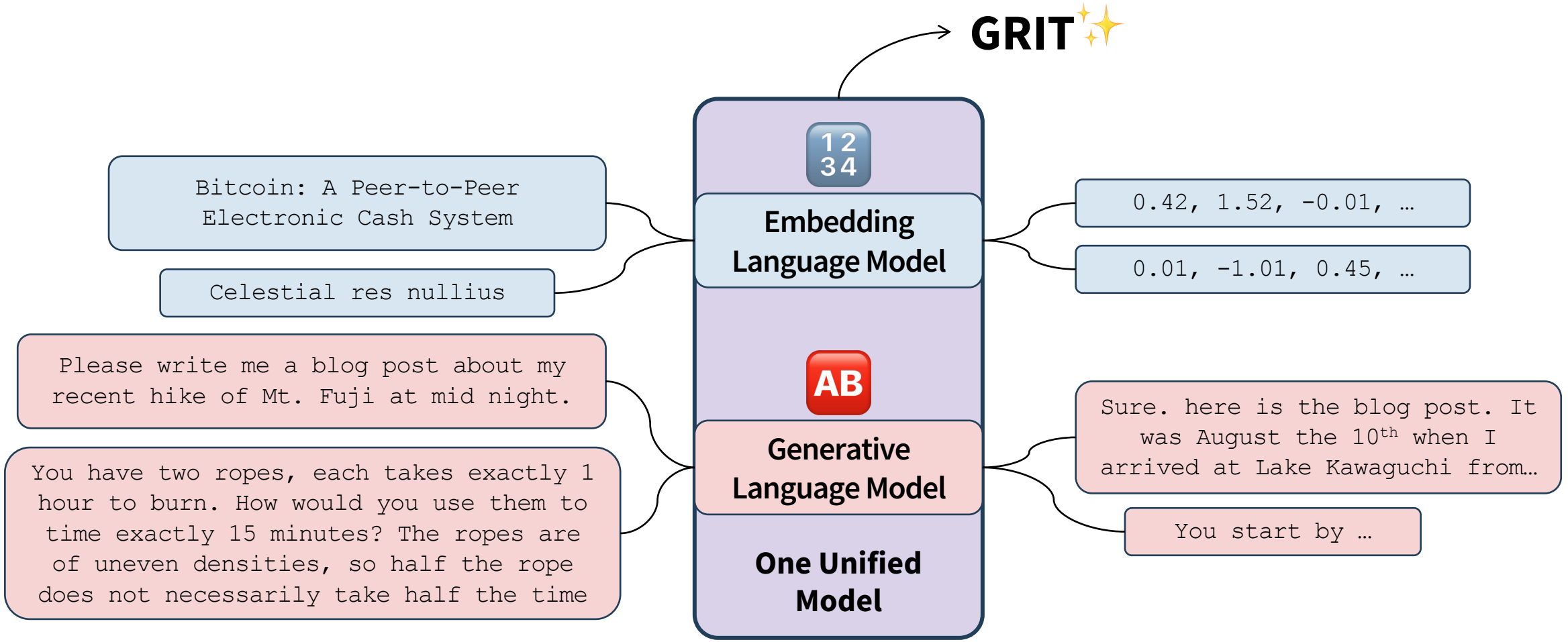
```
Example: Getting embeddings                    curl
1 curl https://api.openai.com/v1/embeddings \
2   -H "Content-Type: application/json" \
3   -H "Authorization: Bearer $OPENAI_API_KEY" \
4   -d '{
5     "input": "Your text string goes here",
6     "model": "text-embedding-3-small"
7   }'
```

AB **Generative LM endpoints**

```
                                               curl
1 curl https://api.openai.com/v1/chat/completions \
2   -H "Content-Type: application/json" \
3   -H "Authorization: Bearer $OPENAI_API_KEY" \
4   -d '{
5     "model": "gpt-3.5-turbo",
6     "messages": [
7       {
```

Motivation for GRIT

- **What GRIT Aims to Achieve**



GRIT

Instruction Tuning

Combining Losses

GRIT: Generative Representational Insturction Tuning

- **Unifying representation & Generation**

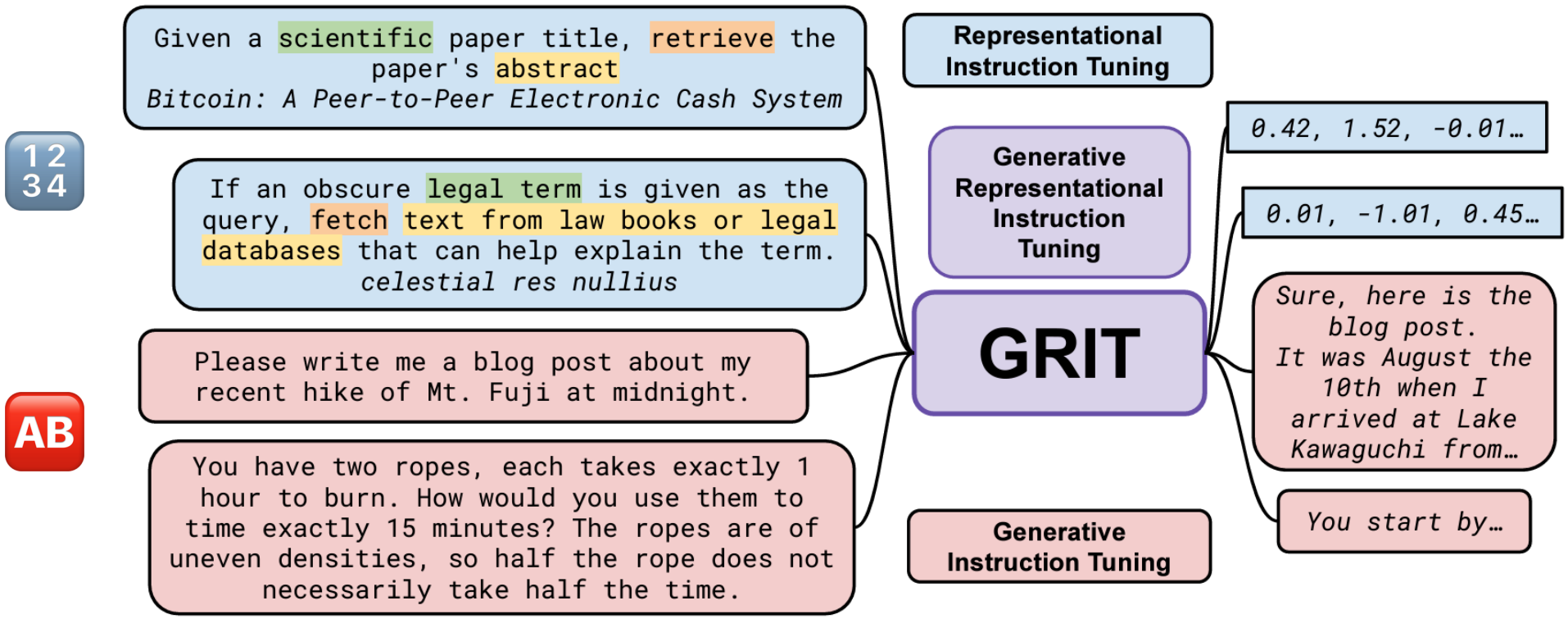


Figure 2: **GRIT.** The same model handles both text representation and generation tasks based on the given instruction. For representation tasks, instructions ideally contain the target domain, intent, and unit [5]. The representation is a tensor of numbers, while the generative output is text.

GRIT: Generative Representational Insturction Tuning

- **Differing instructions, format & attention**

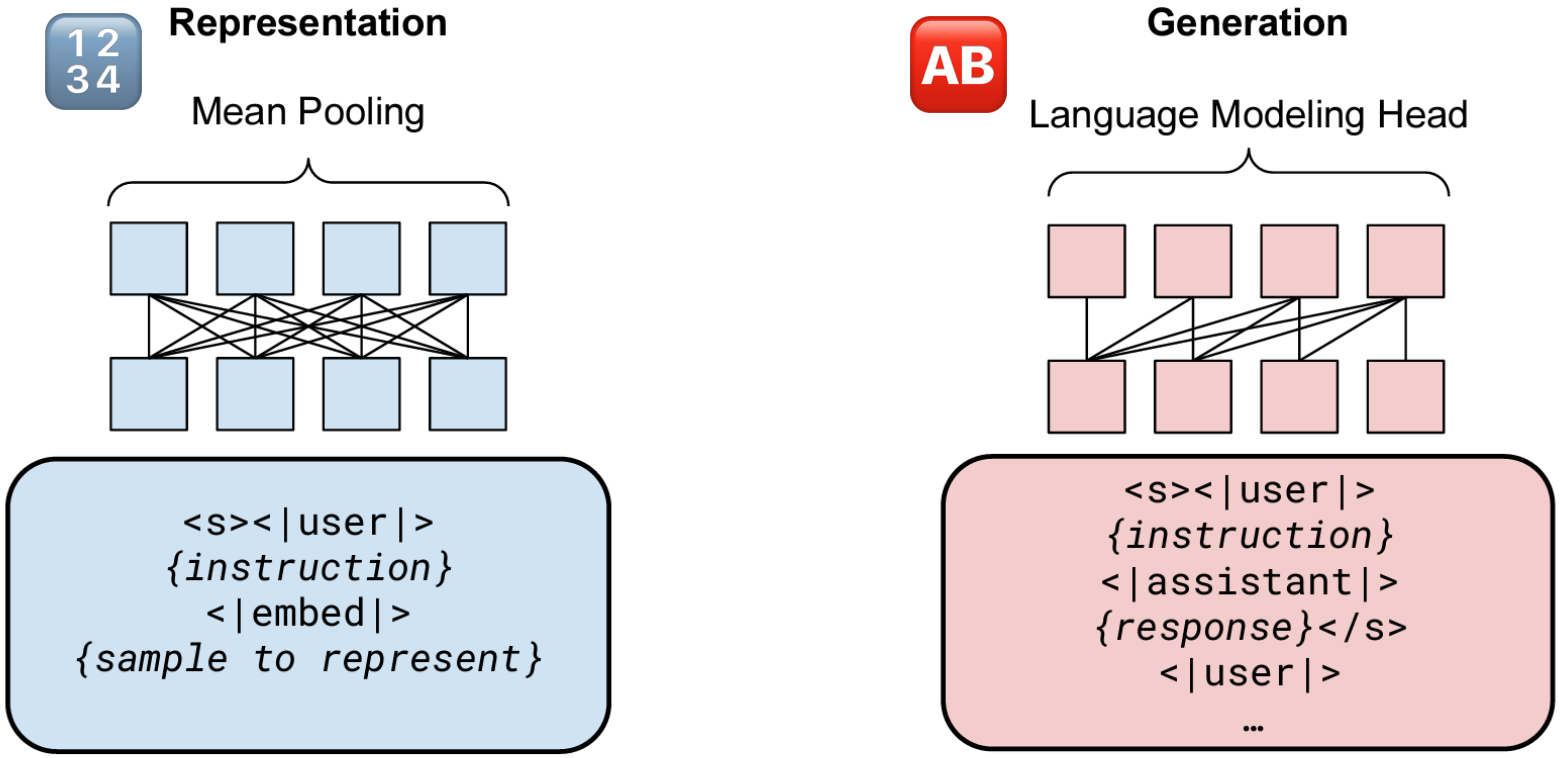


Figure 3: **GRITLM architecture and format.** *Left:* GRITLM uses bidirectional attention over the input for embedding tasks. Mean pooling is applied over the final hidden state to yield the final representation. *Right:* GRITLM uses causal attention over the input for generative tasks. A language modeling head on top of the hidden states predicts the next tokens. The format supports conversations with multiple turns (indicated with "...").

GRIT: Generative Representational Insturction Tuning

- **Differing instructions, format & attention**

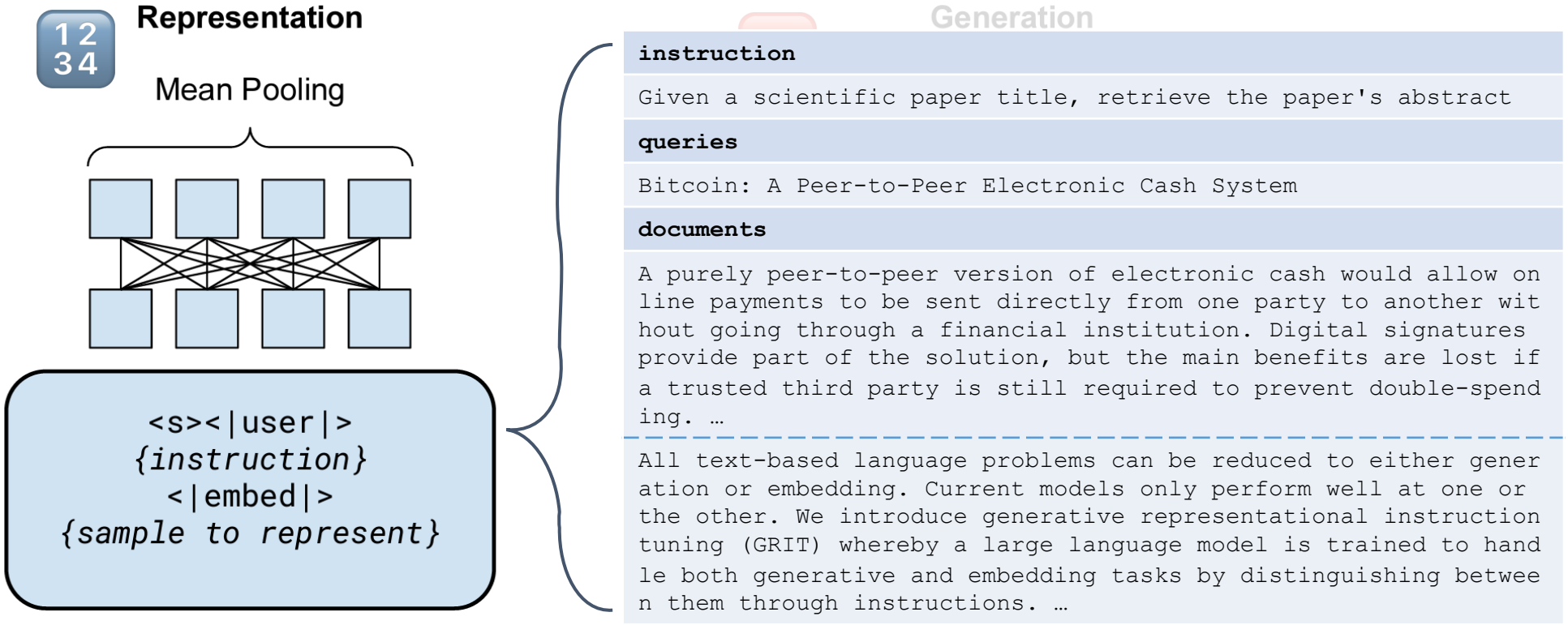


Figure 3: **GRITLM architecture and format.** *Left:* GRITLM uses bidirectional attention over the input for embedding tasks. Mean pooling is applied over the final hidden state to yield the final representation. *Right:* GRITLM uses causal attention over the input for generative tasks. A language modeling head on top of the hidden states predicts the next tokens. The format supports conversations with multiple turns (indicated with "...").

GRIT: Generative Representational Insturction Tuning

- **Differing instructions, format & attention**

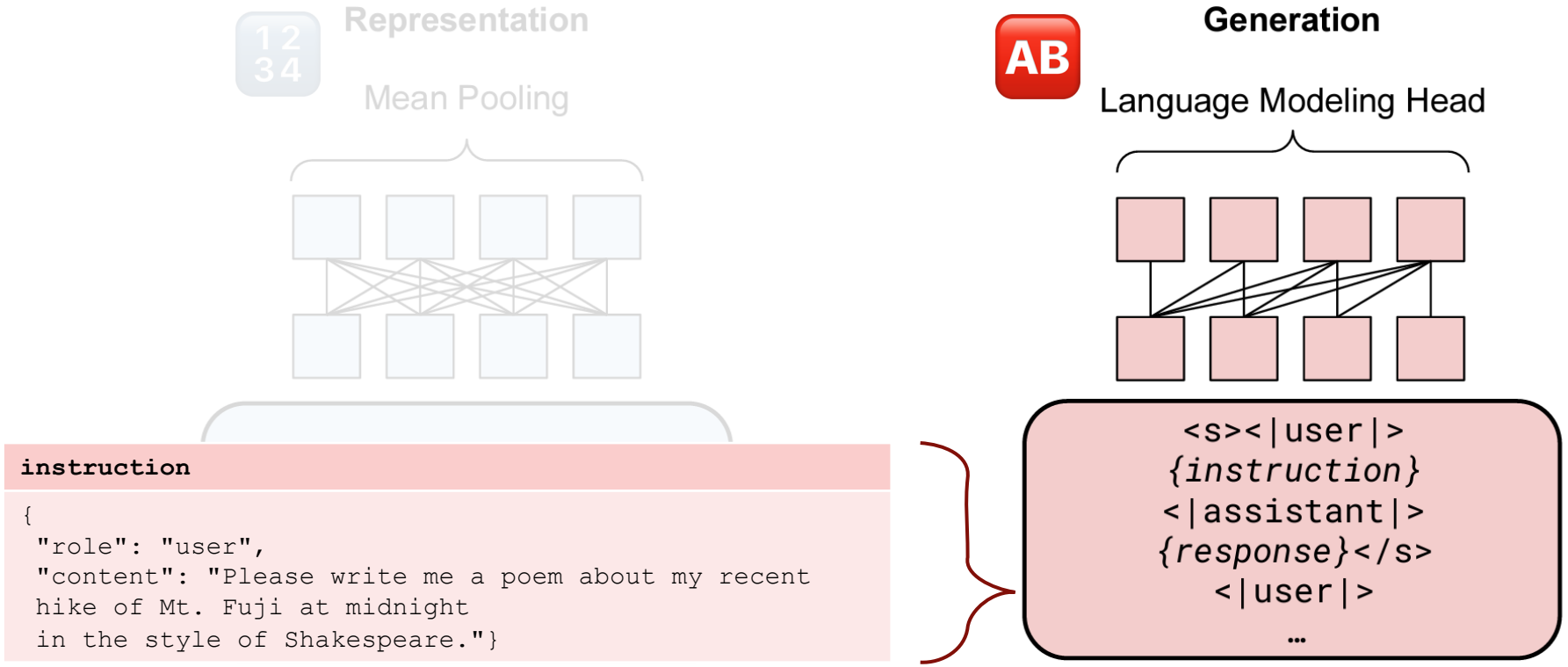


Figure 3: **GRITLM architecture and format.** *Left:* GRITLM uses bidirectional attention over the input for embedding tasks. Mean pooling is applied over the final hidden state to yield the final representation. *Right:* GRITLM uses causal attention over the input for generative tasks. A language modeling head on top of the hidden states predicts the next tokens. The format supports conversations with multiple turns (indicated with "...").

GRIT: Generative Representational Insturction Tuning

- **Combining Losses**
  - Finetune a pretrained language model
    with embedding and generative instruction data in consistent format as depict in 12-14 slides
  - For Embedding – Constrastive Loss

1 2
3 4

Embedding Language Model

$$\mathcal{L}_{\text{Rep}} = -\frac{1}{M}\sum_{i=1}^{M}\log\frac{\exp(\tau\cdot\sigma(f_\theta(q^{(i)}), f_\theta(d^{(i)})))}{\sum_{j=1}^{M}\exp(\tau\cdot\sigma(f_\theta(q^{(i)}), f_\theta(d^{(j)})))}$$

: Contrastive embedding loss with hard negatives

GRIT: Generative Representational Insturction Tuning

- **Combining Losses**
  - Finetune a pretrained language model
    with embedding and generative instruction data in consistent format as depict in 12-14 slides
  - For Embedding – Constrastive Loss
  - For Generation – Negative Log-Likelihood Loss

1 2
3 4

Embedding Language Model

$$\mathcal{L}_{\text{Rep}} = -\frac{1}{M}\sum_{i=1}^{M}\log\frac{\exp(\tau\cdot\sigma(f_\theta(q^{(i)}), f_\theta(d^{(i)})))}{\sum_{j=1}^{M}\exp(\tau\cdot\sigma(f_\theta(q^{(i)}), f_\theta(d^{(j)})))}$$

: Contrastive embedding loss with hard negatives

AB

Generative Language Model

$$\mathcal{L}_{\text{Gen}} = -\frac{1}{N}\sum_{i=1}^{N}\log P(f_{\theta,\eta}(x^{(i)})|f_{\theta,\eta}(x^{(<i)}))$$

: Next token prediction loss for generation

GRIT: Generative Representational Insturction Tuning

- **Combining Losses**

$$\mathcal{L}_{GRIT} = \lambda_{Rep}\mathcal{L}_{Rep} + \lambda_{Gen}\mathcal{L}_{Gen}$$

1 2 3 4 AB

1 2 3 4

$$\mathcal{L}_{\text{Rep}} = -\frac{1}{M}\sum_{i=1}^{M}\log\frac{\exp(\tau\cdot\sigma(f_\theta(q^{(i)}), f_\theta(d^{(i)})))}{\sum_{j=1}^{M}\exp(\tau\cdot\sigma(f_\theta(q^{(i)}), f_\theta(d^{(j)})))}$$

+

AB

$$\mathcal{L}_{\text{Gen}} = -\frac{1}{N}\sum_{i=1}^{N}\log P(f_{\theta,\eta}(x^{(i)})|f_{\theta,\eta}(x^{(<i)}))$$

Effects

Experiment Setup

Performance

Set up

- **Backbone**
  - Mistral 7B and Mistral 8x7B

Variant	Emb	Gen
Mistral 7B	**54.6**	**22.4**
Llama 2 7B	48.2	20.8
GPT-J 6B	51.9	14.0

(b) Base model

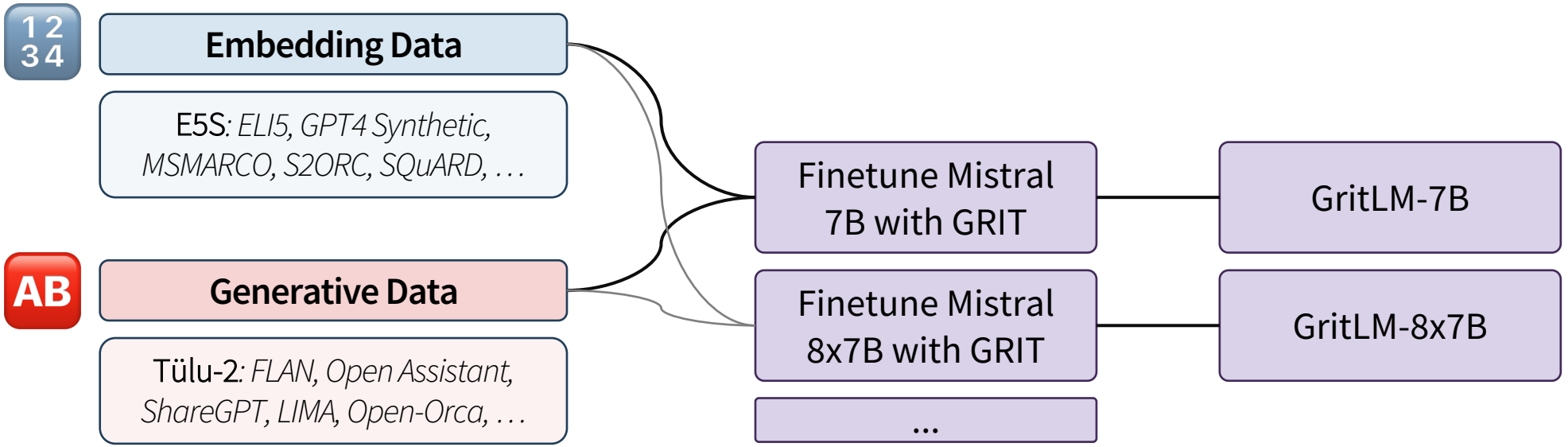
Dataset	Emb
MEDI	64.0
MEDI2	64.7
E5	**66.0**

(c) Embedding dataset

Dataset	Gen
Tülu 2	**55.2**
OASST	37.7
UltraChat	47.4

(d) Generative dataset

- **Adaptation Datasets**
  - E5 (only dataset not publicly available)
  - Tülu 2 data (filter out their custom prompts that contain answers related to the origin of their model)



Experiments – Embedding Performance

- **Embedding Performance**
  - MTEB (total 56 datasets, across 8 tasks)

Task (→) Metric (→) Dataset # (→)	CLF Acc. 12	Clust. V-Meas. 11	PairCLF AP 3	Rerank MAP 4	Retrieval nDCG 15	STS Spear. 10	Summ. Spear. 1	Avg. 56
Proprietary models♥								
OpenAI v3	75.5	49.0	85.7	59.2	55.4	81.7	29.9	64.6
Other Open Models♥								
Llama 2 70B	60.4	29.0	47.1	38.5	9.0	49.1	26.1	35.6
Mistral 7B	63.5	34.6	53.5	43.2	13.2	57.4	19.7	40.5
Mistral 7B Instruct	67.1	34.6	59.6	44.8	16.3	63.4	25.9	43.7
GPT-J 6B	66.2	39.0	60.6	48.9	19.8	60.9	26.3	45.2
SGPT BE 5.8B	68.1	40.3	82.0	56.6	50.3	78.1	31.5	58.9
Instructor XL 1.5B	73.1	44.7	86.6	57.3	49.3	83.1	**32.3**	61.8
BGE Large 0.34B	76.0	46.1	87.1	60.0	54.3	83.1	31.6	64.2
E5 Mistral 7B	78.5	50.3	**88.3**	60.2	56.9	**84.6**	31.4	66.6
GRITLM								
Gen.-only 7B	65.4	32.7	54.2	43.0	13.7	60.2	21.1	41.2
Emb.-only 7B	78.8	**51.1**	87.1	**60.7**	**57.5**	83.8	30.2	**66.8**
GRITLM 7B	**79.5**	50.6	87.2	60.5	57.4	83.4	30.4	**66.8**
GRITLM 8X7B	78.5	50.1	85.0	59.8	55.1	83.3	29.8	65.7

Experiments – Generation Performance

- **Generation Performance**
  - Aplaca Eval

Dataset (→) Setup (→) Metric (→)	MMLU 0 FS EM	GSM8K 8 FS, CoT EM	BBH 3 FS, CoT EM	TyDi QA 1 FS, GP F1	HumanEval 0 FS pass@1	Alpaca 0 FS, 1.0 % Win	Avg.
Proprietary models♥							
GPT-4-0613	81.4	95.0	89.1	65.2	86.6†	91.2	84.8
Other Open Models♥							
GPT-J 6B	27.7	2.5	30.2	9.4	9.8	0.0	13.3
SGPT BE 5.8B	24.4	1.0	0.0	22.8	0.0	0.0	8.0
Zephyr 7B β	58.6	28.0	44.9	23.7	28.5	85.8	44.9
Llama 2 7B	41.8	12.0	39.3	51.2	12.8◆	0.0	26.2
Llama 2 13B	52.0	25.0	48.9	56.5	18.3◆	0.0	33.5
Llama 2 70B	64.5	55.5	66.0	**62.6**	29.9◆	0.0	46.4
Llama 2 Chat 13B	53.2	9.0	40.3	32.1	19.6†	91.4	40.9
Llama 2 Chat 70B	60.9	59.0	49.0	44.4	34.3†	94.5	57.0
Tülu 2 7B	50.4	34.0	48.5	46.4	24.5†	73.9	46.3
Tülu 2 13B	55.4	46.0	49.5	53.2	31.4	78.9	52.4
Tülu 2 70B	67.3	**73.0**	68.4	53.6	41.6	86.6	65.1
Mistral 7B	60.1	44.5	55.6	55.8	30.5	0.0	41.1
Mistral 7B Instruct	53.0	36.0	38.5	27.8	34.0	75.3	44.1
Mixtral 8x7B Instruct	**68.4**	65.0	55.9	24.3	**53.5**	**94.8**	60.3
GRITLM							
Emb.-only 7B	23.5	1.0	0.0	21.0	0.0	0.0	7.6
Gen.-only 7B	57.5	52.0	55.4	56.6	34.5	75.4	55.2
GRITLM 7B	57.6	57.5	54.8	55.4	32.8	74.8	55.5
GRITLM 8X7B	66.7	61.5	**70.2**	58.2	53.4	84.0	**65.7**

Experiments – Embedding Performance

- **Embedding Performance**
  - MTEB (total 56 datasets, across 8 tasks)

Task (→) Metric (→) Dataset # (→)	CLF Acc. 12	Clust. V-Meas. 11	PairCLF AP 3	Rerank MAP 4	Retrieval nDCG 15	STS Spear. 10	Summ. Spear. 1	Avg. 56
Proprietary models♥								
OpenAI v3	75.5	49.0	85.7	59.2	55.4	81.7	29.9	64.6
Other Open Models♥								
Llama 2 70B	60.4	29.0	47.1	38.5	9.0	49.1	26.1	35.6
Mistral 7B	63.5	34.6	53.5	43.2	13.2	57.4	19.7	40.5
Mistral 7B Instruct	67.1	34.6	59.6	44.8	16.3	63.4	25.9	43.7
GPT-J 6B	66.2	39.0	60.6	48.9	19.8	60.9	26.3	45.2
SGPT BE 5.8B	68.1	40.3	82.0	56.6	50.3	78.1	31.5	58.9
Instructor XL 1.5B	73.1	44.7	86.6	57.3	49.3	83.1	**32.3**	61.8
BGE Large 0.34B	76.0	46.1	87.1	60.0	54.3	83.1	31.6	64.2
E5 Mistral 7B	78.5	50.3	**88.3**	60.2	56.9	**84.6**	31.4	66.6
GRITLM								
Gen.-only 7B	65.4	32.7	54.2	43.0	13.7	60.2	21.1	41.2
Emb.-only 7B	78.8	**51.1**	87.1	**60.7**	**57.5**	83.8	30.2	**66.8**
GRITLM 7B	**79.5**	50.6	87.2	60.5	57.4	83.4	30.4	**66.8**
GRITLM 8X7B	78.5	50.1	85.0	59.8	55.1	83.3	29.8	65.7

Experiments – Generation Performance

- **Generation Performance**
  - Aplaca Eval

Dataset (→) Setup (→) Metric (→)	MMLU 0 FS EM	GSM8K 8 FS, CoT EM	BBH 3 FS, CoT EM	TyDi QA 1 FS, GP F1	HumanEval 0 FS pass@1	Alpaca 0 FS, 1.0 % Win	Avg.
Proprietary models♥							
GPT-4-0613	81.4	95.0	89.1	65.2	86.6†	91.2	84.8
Other Open Models♥							
GPT-J 6B	27.7	2.5	30.2	9.4	9.8	0.0	13.3
SGPT BE 5.8B	24.4	1.0	0.0	22.8	0.0	0.0	8.0
Zephyr 7B β	58.6	28.0	44.9	23.7	28.5	85.8	44.9
Llama 2 7B	41.8	12.0	39.3	51.2	12.8◆	0.0	26.2
Llama 2 13B	52.0	25.0	48.9	56.5	18.3◆	0.0	33.5
Llama 2 70B	64.5	55.5	66.0	**62.6**	29.9◆	0.0	46.4
Llama 2 Chat 13B	53.2	9.0	40.3	32.1	19.6†	91.4	40.9
Llama 2 Chat 70B	60.9	59.0	49.0	44.4	34.3†	94.5	57.0
Tülu 2 7B	50.4	34.0	48.5	46.4	24.5†	73.9	46.3
Tülu 2 13B	55.4	46.0	49.5	53.2	31.4	78.9	52.4
Tülu 2 70B	67.3	**73.0**	68.4	53.6	41.6	86.6	65.1
Mistral 7B	60.1	44.5	55.6	55.8	30.5	0.0	41.1
Mistral 7B Instruct	53.0	36.0	38.5	27.8	34.0	75.3	44.1
Mixtral 8x7B Instruct	**68.4**	65.0	55.9	24.3	**53.5**	**94.8**	60.3
GRITLM							
Emb.-only 7B	23.5	1.0	0.0	21.0	0.0	0.0	7.6
Gen.-only 7B	57.5	52.0	55.4	56.6	34.5	75.4	55.2
GRITLM 7B	57.6	57.5	54.8	55.4	32.8	74.8	55.5
GRITLM 8X7B	66.7	61.5	**70.2**	58.2	53.4	84.0	**65.7**

GRITLM is the Only Model Handling Both with Good Performance !

- **GRIT models are the only ones that can handle both embedding and generation at best-in-class performance.**

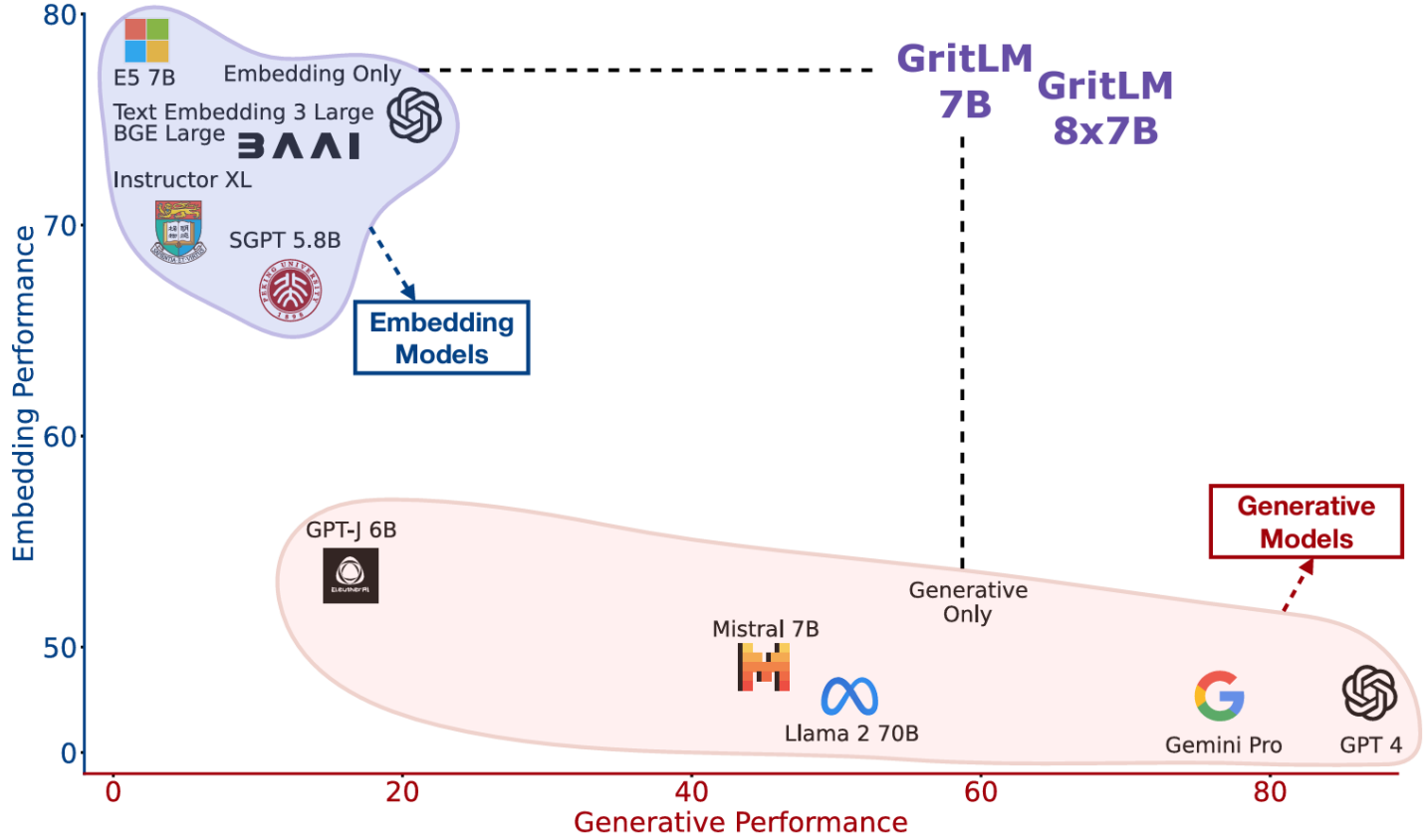


Figure 1: **Performance of various models on text representation (embedding) and generation tasks.** GRITLM is the first model to perform best-in-class at both types of tasks simultaneously.

Ablation Study

- **More than 10 ablation studies**
  - Bidirectional for Embeddings + Causal for generative

Attention Emb Instruction	Attention Emb Sample	Attention Gen Instruction	Attention Gen Sample	Pooling	Emb	Gen
Embedding Only						
Causal				Wmean	60.0	-
Causal	Bidirectional			Mean	61.0	-
Bidirectional				Mean	61.8	-
Generative Only						
		Causal			-	**55.2**
		Bidirectional	Causal		-	50.7
Unified						
Causal		Causal		Last token	61.2	53.0
Causal		Causal		Wmean	62.8	52.8
Bidirectional		**Causal**		**Mean**	**64.0**	52.9

(a) Attention and pooling ablations. Wmean is position-weighted mean pooling [104].

Variant	Emb	Gen
Mistral 7B	**54.6**	**22.4**
Llama 2 7B	48.2	20.8
GPT-J 6B	51.9	14.0

(b) Base model

Dataset	Emb
MEDI	64.0
MEDI2	64.7
E5	**66.0**

(c) Embedding dataset

Variant	Emb	Gen
No head	**62.7**	**49.2**
-> 1024	62.1	48.0

(e) Embedding head

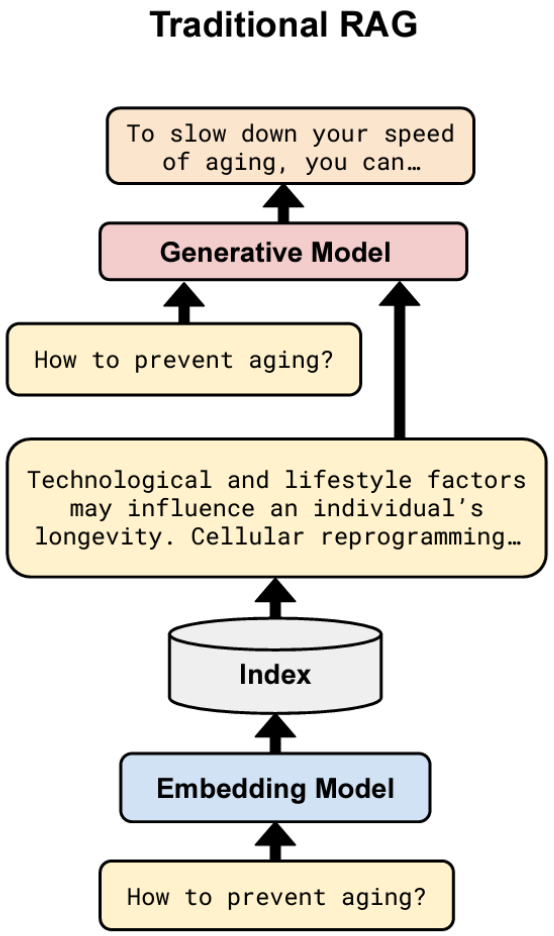
Dataset	Gen
Tülu 2	**55.2**
OASST	37.7
UltraChat	47.4

(d) Generative dataset

GRIT in RAG

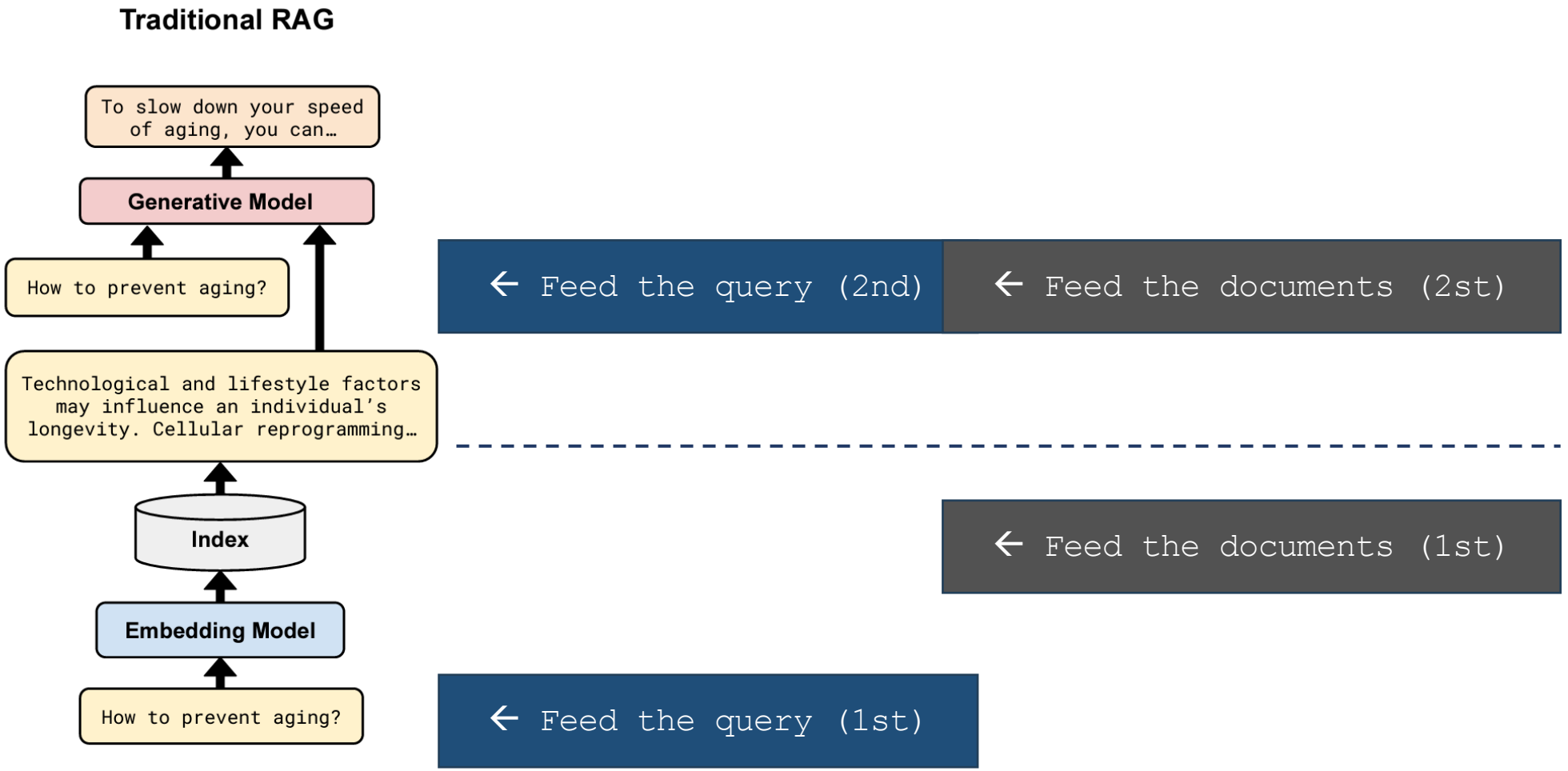
Efficiency of GRITLM in RAG

- **Traditional RAG**



Efficiency of GRITLM in RAG

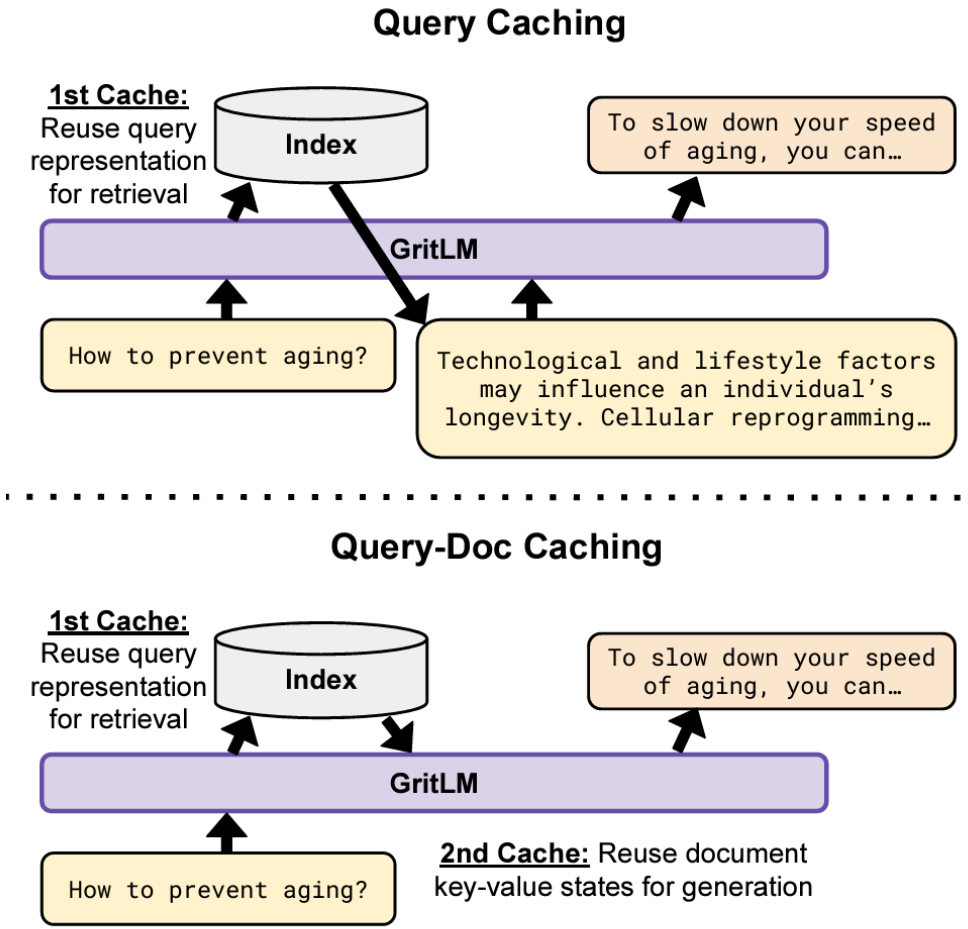
- **Traditional RAG ← Inefficient**



Efficiency of GRITLM in RAG

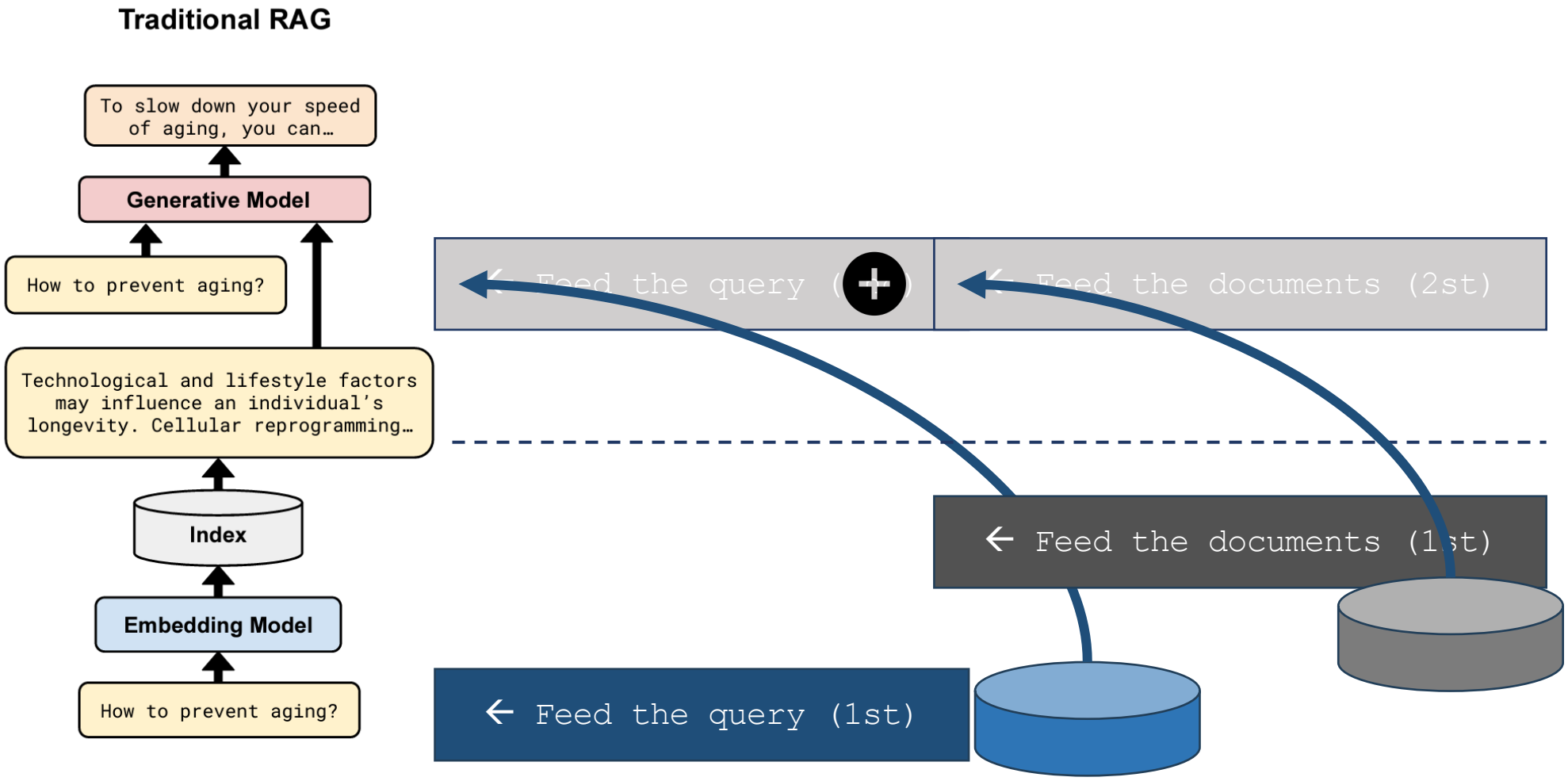
- **Intuitive: We just don't need to store the text anmore!**
  - Just store the key-value caches
    - For attention mechanism
  - Then reuse them during RAG process
  - Query Caching + Query-Doc Caching
  - Doc Caching + Doc-Query Caching

→ simple order difference



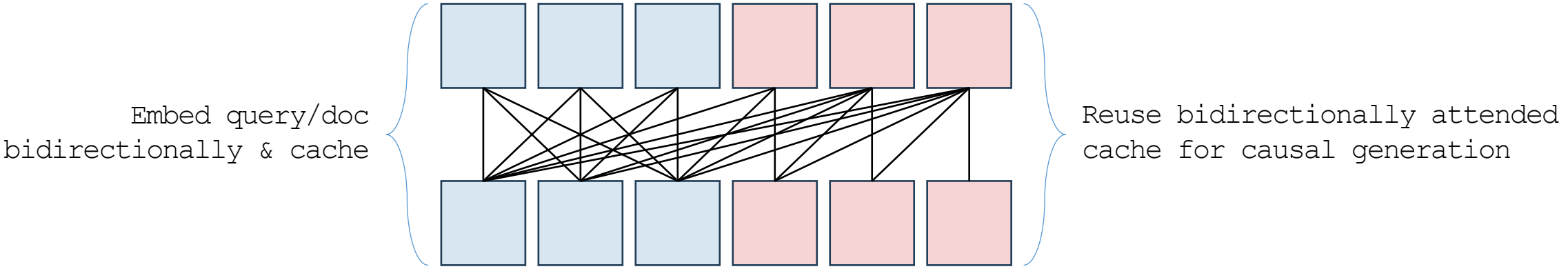
Efficiency of GRITLM in RAG

- **Traditional RAG ← Inefficient**

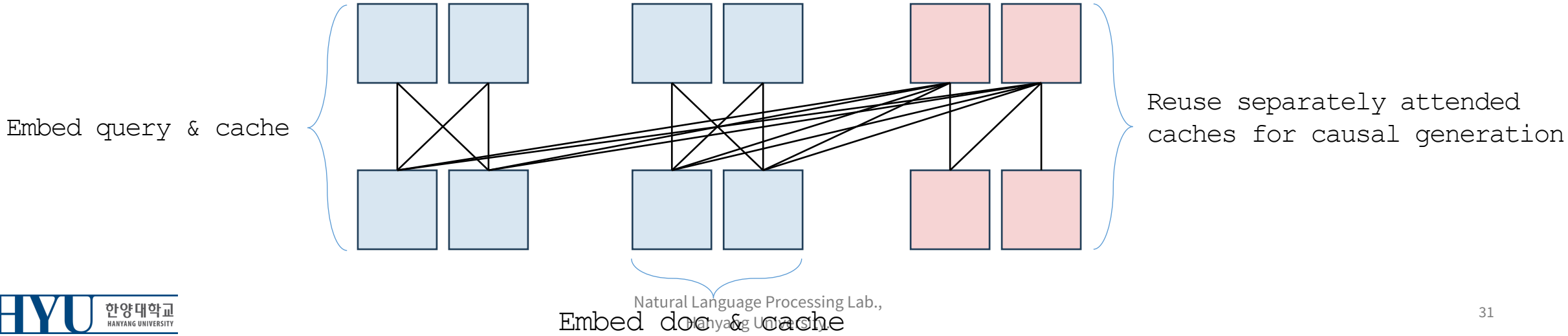


Limitation of RAG with GRIT

- **Attention Mismatch Problem**
  - Combining bidirectional & causal attention



  - Combining separtely attended texts (only if caching both, query-doc/doc-query)



Results of GRITLM's RAG

- **Caching Performance**

	Match (0-shot, ↑)	CPU Latency (s, ↓) Sample A	CPU Latency (s, ↓) Sample B	GPU Latency (s, ↓) Sample A	GPU Latency (s, ↓) Sample B	Storage (↓)
No RAG	21.00	4.3 ± 0.36	13.69 ± 1.0	0.24 ± 0.04	0.38 ± 0.04	**0GB**
Query then document prompt						
RAG	30.50	11.64 ± 0.74	14.88 ± 0.87	0.39 ± 0.02	0.40 ± 0.02	43GB
Query Caching	25.46	18.30 ± 0.76	6.87 ± 0.89	0.44 ± 0.03	**0.27 ± 0.02**	43GB
Query-Doc Caching	21.63	**5.12 ± 0.23**	6.62 ± 0.97	0.27 ± 0.03	0.29 ± 0.01	30TB
Document then query prompt						
RAG	30.47	14.18 ± 1.01	15.33 ± 0.87	0.39 ± 0.01	0.4 ± 0.01	43GB
Doc Caching	**33.38**	5.25 ± 0.34	23.23 ± 1.05	0.27 ± 0.03	0.45 ± 0.02	30TB
Doc-Query Caching	18.39	5.23 ± 0.37	**6.41 ± 0.96**	**0.26 ± 0.03**	**0.27 ± 0.02**	30TB

- Task: NQ(Natural Questions)
- Model: GRITLM 7B

1. The RAG model outperforms the baseline(=no RAG).
2. Caching Queries and Documents' performances are mismatched
3. Trade-off between speed and performance using cache.

Results of GRITLM's RAG

- **Caching Performance**

	Match (0-shot, ↑)	CPU Latency (s, ↓) Sample A	CPU Latency (s, ↓) Sample B	GPU Latency (s, ↓) Sample A	GPU Latency (s, ↓) Sample B	Storage (↓)
No RAG	21.00	4.3 ± 0.36	13.69 ± 1.0	0.24 ± 0.04	0.38 ± 0.04	**0GB**
Query then document prompt						
RAG	30.50	11.64 ± 0.74	14.88 ± 0.87	0.39 ± 0.02	0.40 ± 0.02	43GB
Query Caching	25.46	18.30 ± 0.76	6.87 ± 0.89	0.44 ± 0.03	**0.27 ± 0.02**	43GB
Query-Doc Caching	21.63	**5.12 ± 0.23**	6.62 ± 0.97	0.27 ± 0.03	0.29 ± 0.01	30TB
Document then query prompt						
RAG	30.47	14.18 ± 1.01	15.33 ± 0.87	0.39 ± 0.01	0.4 ± 0.01	43GB
Doc Caching	**33.38**	5.25 ± 0.34	23.23 ± 1.05	0.27 ± 0.03	0.45 ± 0.02	30TB
Doc-Query Caching	18.39	5.23 ± 0.37	**6.41 ± 0.96**	**0.26 ± 0.03**	**0.27 ± 0.02**	30TB

- Task: NQ(Natural Questions)
- Model: GRITLM 7B

1. The RAG model outperforms the baseline(=no RAG).
2. Caching Queries and Documents' performances are mismatched.
3. Trade-off between speed and performance using cache.

Results of GRITLM's RAG

- **Caching Performance**

	Match (0-shot, ↑)	CPU Latency (s, ↓) Sample A	CPU Latency (s, ↓) Sample B	GPU Latency (s, ↓) Sample A	GPU Latency (s, ↓) Sample B	Storage (↓)
No RAG	21.00	4.3 ± 0.36	13.69 ± 1.0	0.24 ± 0.04	0.38 ± 0.04	**0GB**
Query then document prompt						
RAG	30.50	11.64 ± 0.74	14.88 ± 0.87	0.39 ± 0.02	0.40 ± 0.02	43GB
Query Caching	25.46	18.30 ± 0.76	6.87 ± 0.89	0.44 ± 0.03	**0.27 ± 0.02**	43GB
Query-Doc Caching	21.63	**5.12 ± 0.23**	6.62 ± 0.97	0.27 ± 0.03	0.29 ± 0.01	30TB
Document then query prompt						
RAG	30.47	14.18 ± 1.01	15.33 ± 0.87	0.39 ± 0.01	0.4 ± 0.01	43GB
Doc Caching	**33.38**	5.25 ± 0.34	23.23 ± 1.05	0.27 ± 0.03	0.45 ± 0.02	30TB
Doc-Query Caching	18.39	5.23 ± 0.37	**6.41 ± 0.96**	**0.26 ± 0.03**	**0.27 ± 0.02**	30TB

- Task: NQ(Natural Questions)
- Model: GRITLM 7B

1. The RAG model outperforms the baseline(=no RAG).
2. Caching Queries and Documents' performances are mismatched.
3. Trade-off between speed and performance using cache.

Conclusion

Findings

Limitations

Conclusion

- **GRIT model unifies the tasks of embedding and generative.**

- **From this approach, the model provides us with 3 advantages**
  - : performance, efficiency and simplicity
  - It leads the state-of-the-art models both at embedding and generative.

- **There are some limitations**
  - Requires more compute to pre-train with 2 different objectives
  - On RAG wit GRITLM, Naïve caching approach can make compute easier but it's a performance tradeoff.

Thank You

Yejin Yoon

HYU NLP Lab.
Hanyang University, South Korea

stillwithyou@hanyang.ac.kr