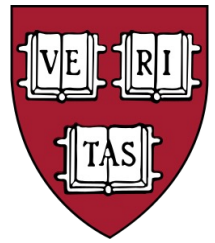
Online Adaptation of Language Models with a Memory of Amortized Contexts

Jihoon Tack, Jaehyung Kim, Eric Mitchell, Jinwoo Shin, Yee Whye Teh, Jonathan Richard Schwarz



Problem of interest: Online learning of Language Models (LMs)

Making Language Models (LMs) up to date is highly important… but quite hard…

- Not arguing that we don't need "training from scratch"
- **Current question**: How to update the LLaMA-2 until the release of LLaMA-3…?

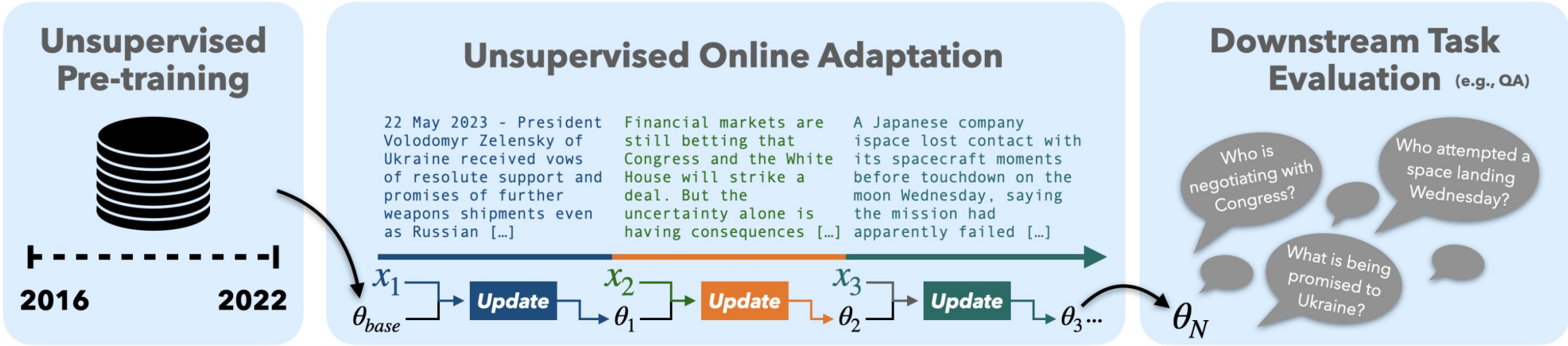
Problem of interest: Online learning of Language Models (LMs)

Making Language Models (LMs) up to date is highly important… but quite hard…

- Not arguing that we don't need "training from scratch"
- **Current question**: How to update the LLaMA-2 until the release of LLaMA-3…?

Problem of interest: Online adaptation of LMs

- Adapting LM on a stream of documents



[1] Hu et al., Meta-Learning Online Adaptation of Language Models, arXiv 2023

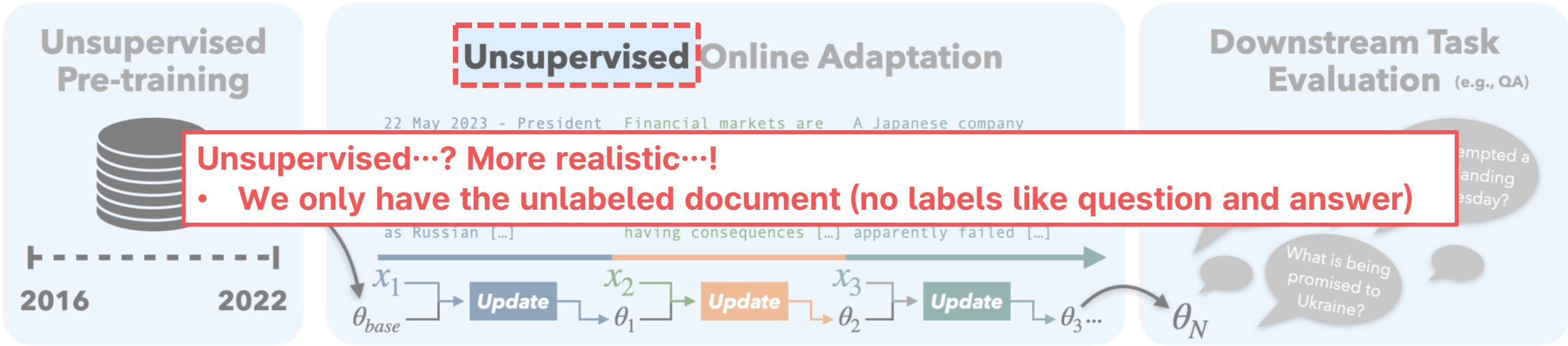
Problem of interest: Online learning of Language Models (LMs)

Making Language Models (LMs) up to date is highly important… but quite hard…

- Not arguing that we don't need "training from scratch"
- **Current question**: How to update the LLaMA-2 until the release of LLaMA-3…?

Problem of interest: Online adaptation of LMs

- Adapting LM on a stream of documents



Unsupervised…? More realistic…!

- **We only have the unlabeled document (no labels like question and answer)**

[1] Hu et al., Meta-Learning Online Adaptation of Language Models, arXiv 2023

Problem of interest: Online learning of Language Models (LMs)

Making Language Models (LMs) up to date is highly important… but quite hard…

- Not arguing that we don't need "training from scratch"
- **Current question**: How to update the LLaMA-2 until the release of LLaMA-3…?

Problem of interest: Online adaptation of LMs

- Adapting LM on a stream of documents

Just a bit more mathematical explanation for easier understanding

Adapt the update on unlabeled documents $\theta_{\texttt{base}} \longrightarrow \mathcal{C}_{\texttt{test}} := (\mathbf{d}_1, \cdots, \mathbf{d}_N) \longrightarrow \theta_{\texttt{update}}$

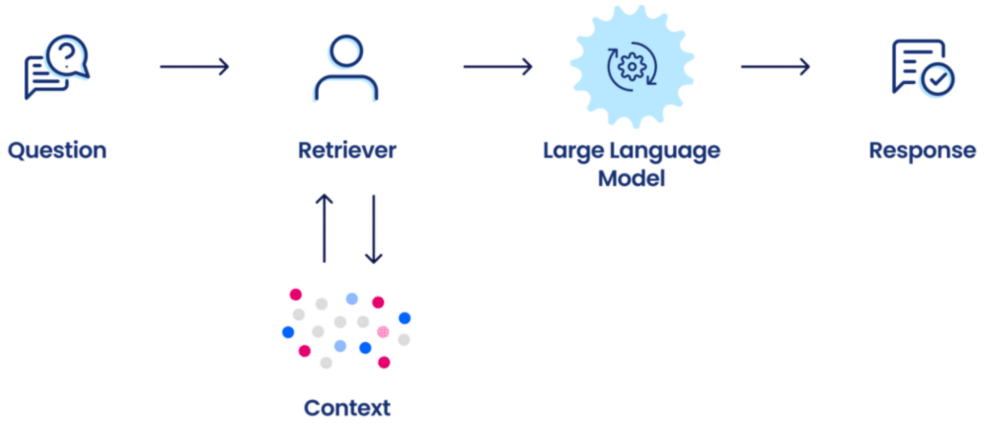
Evaluation on the query input and labels $(\mathbf{x}_i, \mathbf{y}_i) \sim p(\mathbf{x}, \mathbf{y} | \mathbf{d}_i)$ $\quad \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}\big(\mathrm{LM}_{\theta_{\texttt{base}}}(\mathbf{x}_i), \mathbf{y}_i\big)$

[1] Hu et al., Meta-Learning Online Adaptation of Language Models, arXiv 2023

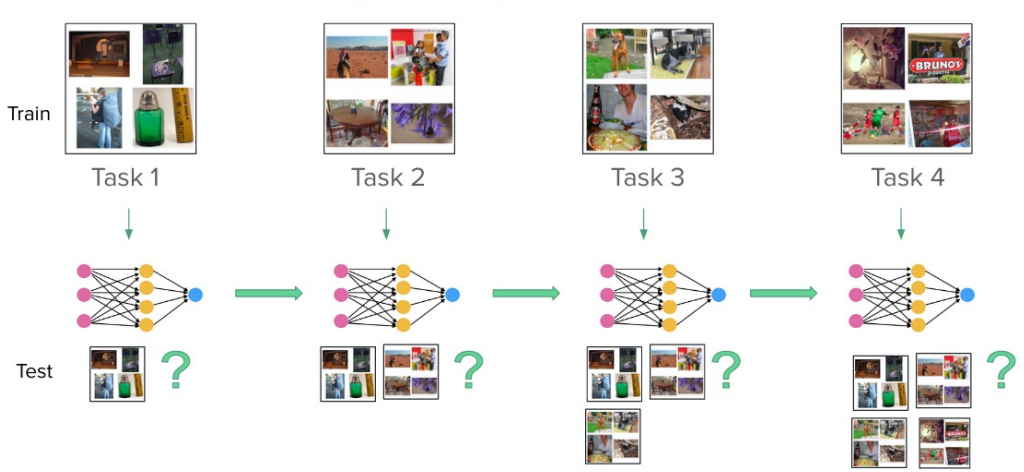
Previous works?

Retrieval augmentation: Save documents into a memory bank and later retrieve and prepend it

Online finetuning: Update the model's parameter with the stream of documents



Retrieval augmentation

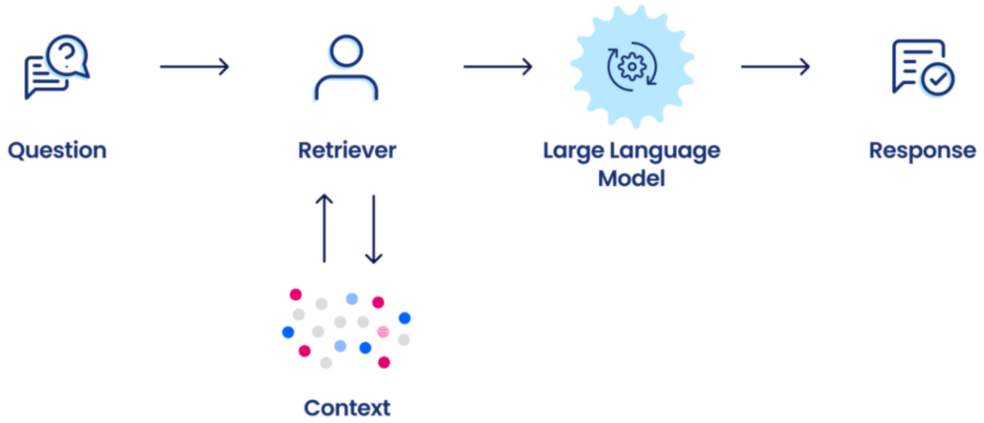


Online finetuning
(use autoregressive loss)

Previous works?

Retrieval augmentation: Save documents into a memory bank and later retrieve and prepend it

Online finetuning: Update the model's parameter with the stream of documents



Retrieval augmentation

Pros

- Strong knowledge retention
- Effective performance
- Efficient adaptation (no need to calculate the gradients)

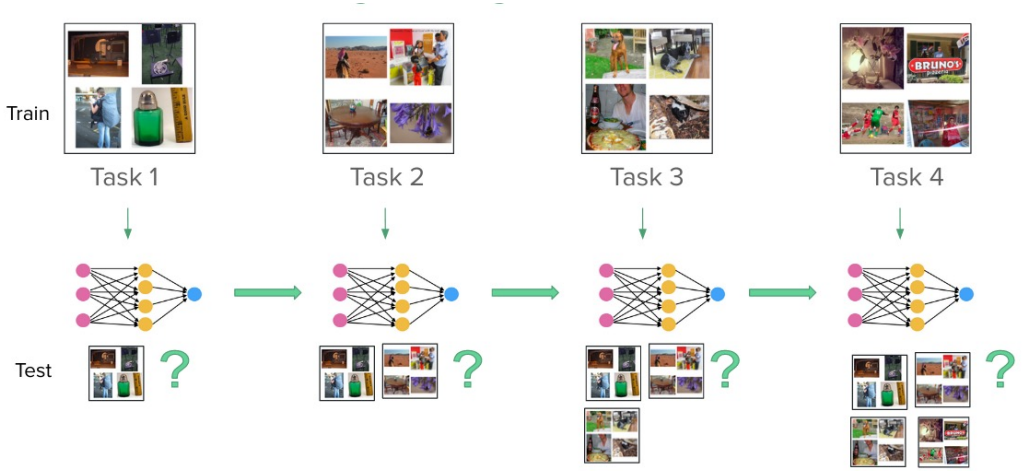
Cons

- Inefficient inference
- Possibility of choosing the wrong document
- + even large models struggle when the information is counterfactual with the learned knowledge

Previous works?

Retrieval augmentation: Save documents into a memory bank and later retrieve and prepend it

Online finetuning: Update the model's parameter with the stream of documents



Online finetuning
(use autoregressive loss)

Pros

- Efficient inference
- Can edit the knowledge itself from the parameter

Cons

- Inevitable forgetting of the learned knowledge
- Highly sensitive to the online optimization hyper-parameter
- Inefficient adaptation (need to calculate the gradients)

How about ours…?

Retrieval augmentation: Save documents into a memory bank and later retrieve and prepend it

Online finetuning: Update the model's parameter with the stream of documents

Idea: Let's have a middle point

- 1) Efficient adaptation
- 2) Efficient inference
- 3) Strong retention of the learned knowledge
- 4) No hyper-parameter during the adaptation
- 5) Possibility of using the knowledge from similar documents

How about ours…?

Retrieval augmentation: Save documents into a memory bank and later retrieve and prepend it

Online finetuning: Update the model's parameter with the stream of documents

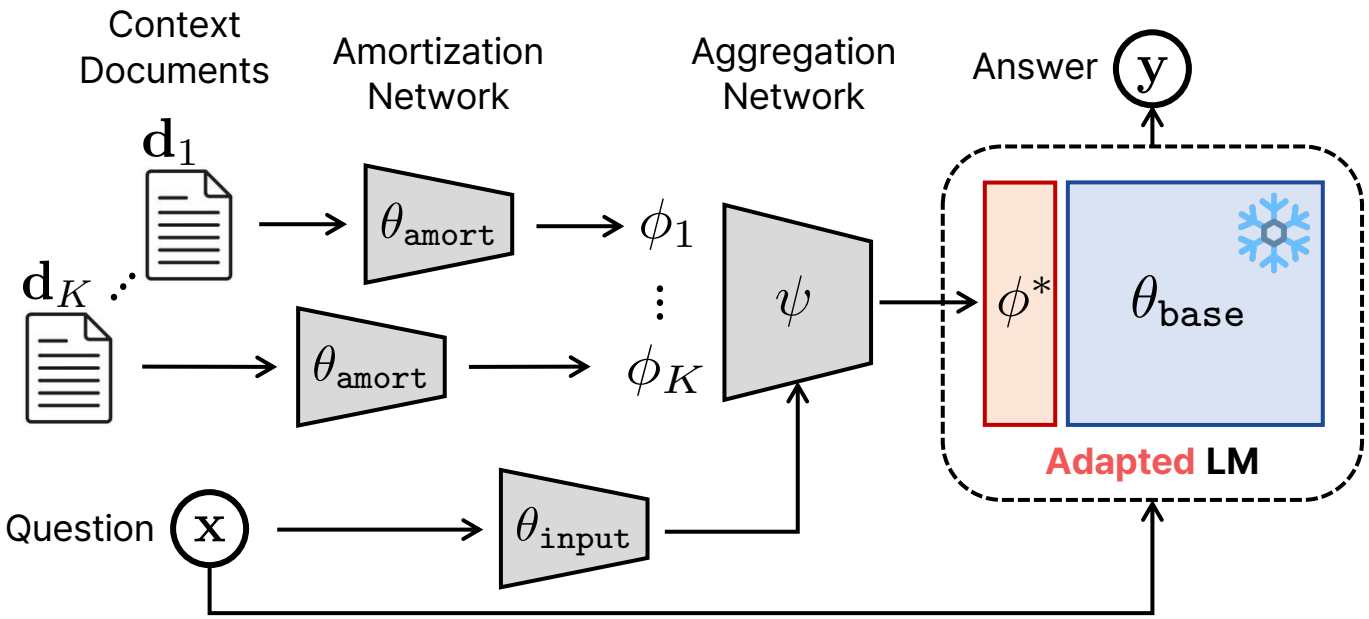
Idea: Let's have a middle point

- 1) Efficient adaptation
- 2) Efficient inference
- 3) Strong retention of the learned knowledge
- 4) No hyper-parameter during the adaptation
- 5) Possibility of using the knowledge from similar documents

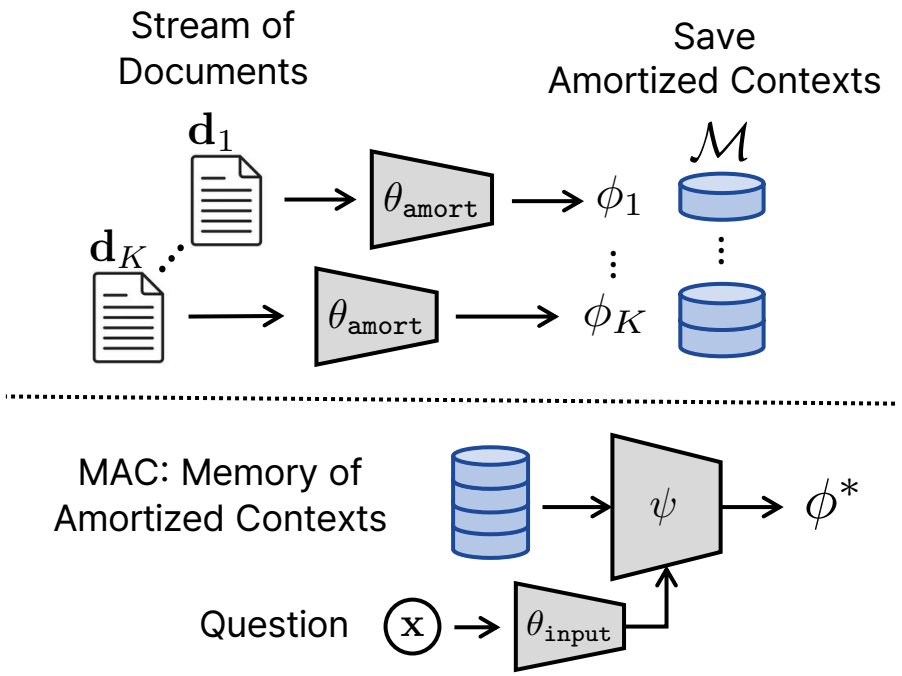
Key idea? Very simple, but will explain why this resolves the issues

1. Amortize each document into parameter-efficient finetuning (PEFT) parameters, e.g., LoRA
2. Save the amortized documents into a memory bank
3. Learn to aggregate the memory bank to output the possible best modulation

Overall summary

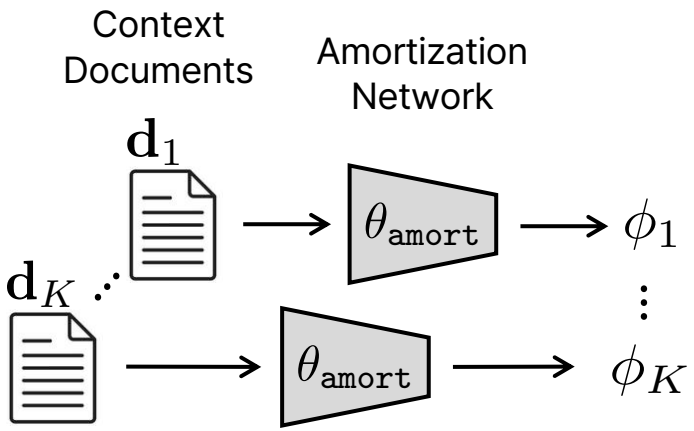


Training: **Learning to Amortize and Aggregate**



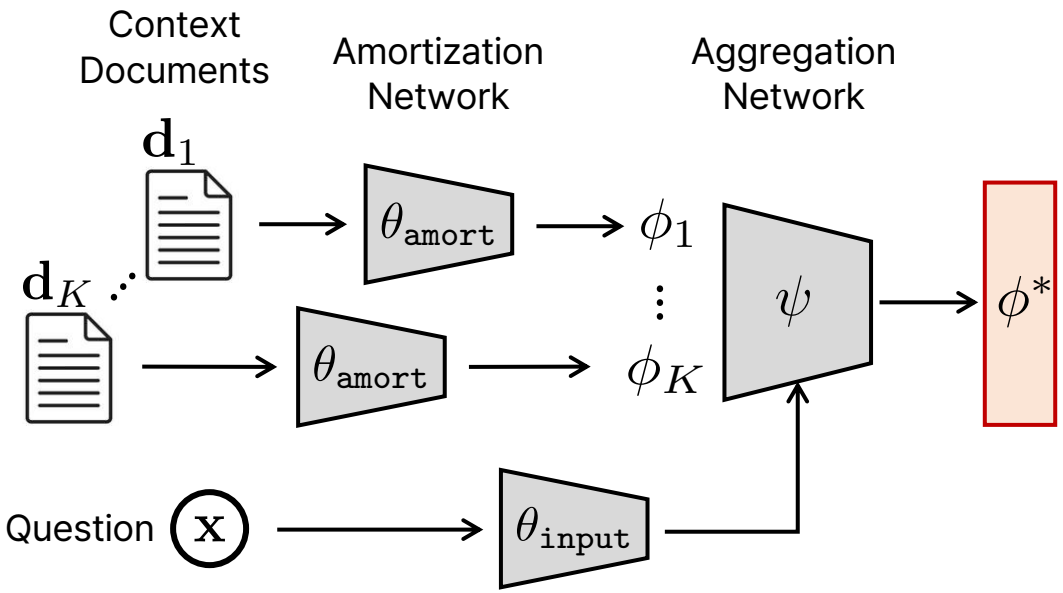
Inference: **Online Adaptation**

Training: Learning to amortize and aggregate



Amortize each document into PEFT parameters (e.g., LoRA)

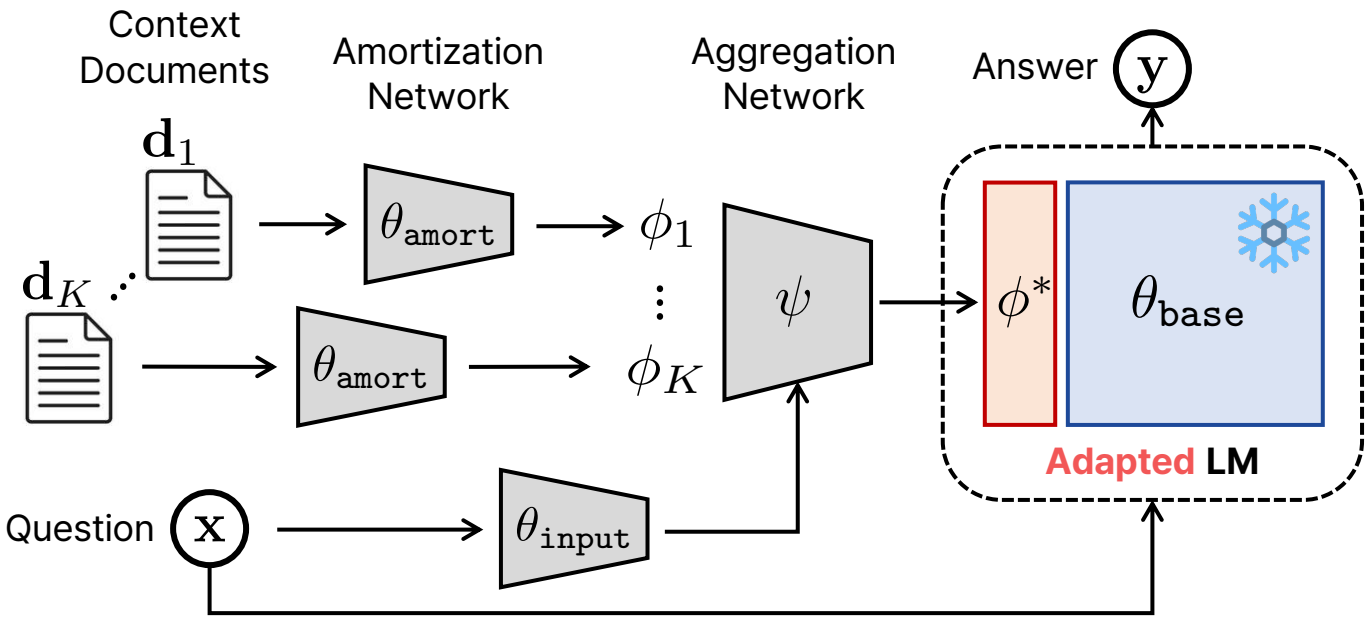
Training: Learning to amortize and aggregate



Amortize each document into PEFT parameters (e.g., LoRA)

Aggregate the PEFT parameters into a single parameter

Training: Learning to amortize and aggregate

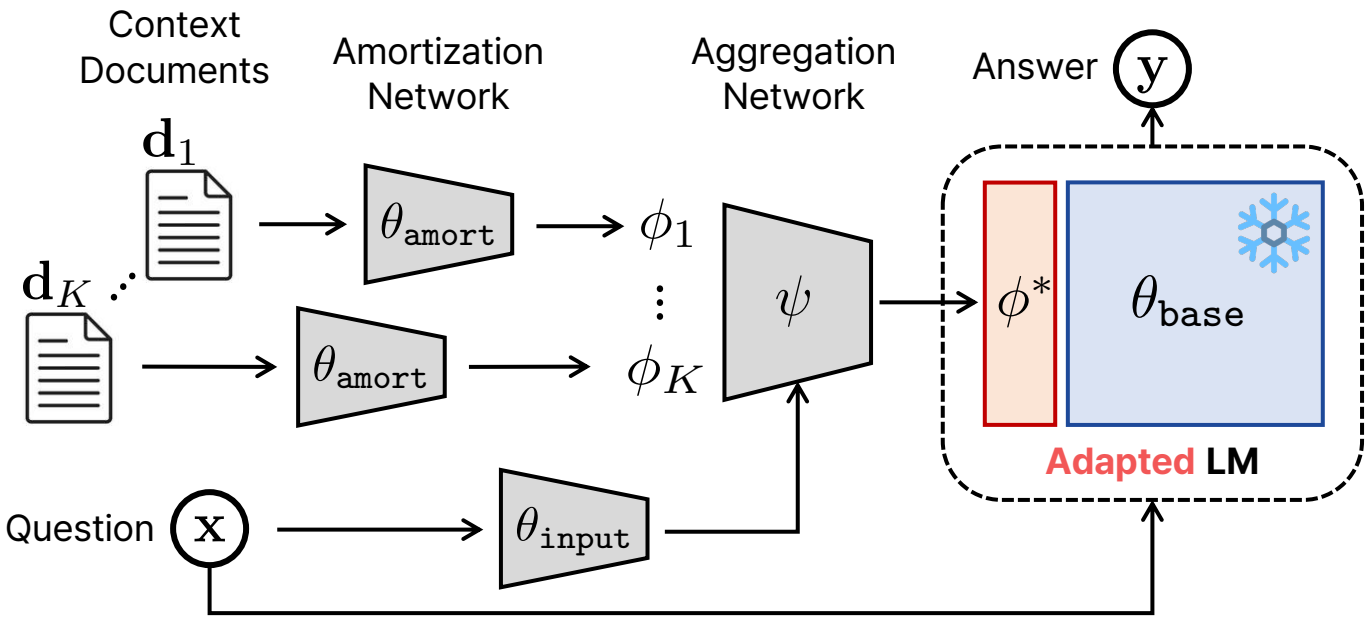


Amortize each document into PEFT parameters (e.g., LoRA)

Aggregate the PEFT parameters into a single parameter

Adapt the frozen base LM

Training: Learning to amortize and aggregate



Amortize each document into PEFT parameters (e.g., LoRA)

Aggregate the PEFT parameters into a single parameter

Adapt the frozen base LM

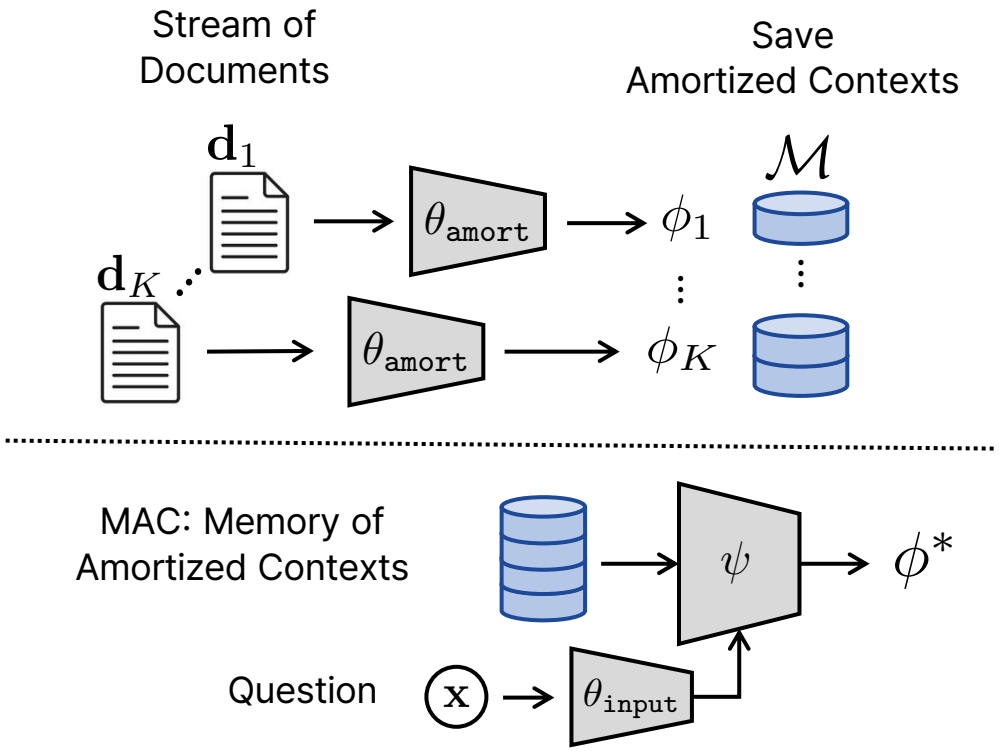
End-to-end training objective

$$\min_{\theta_{\text{amort}},\theta_{\text{input}},\psi} \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}\left(\text{LM}_{\theta_{\text{base}}}(\mathbf{x}_i;\phi_i^*),\mathbf{y}_i\right) \quad \text{where} \quad \phi_i^* := h_\psi\left(g_{\theta_{\text{input}}}(\mathbf{x}_i),\{\phi_k\}_{k=1}^K\right)$$

$$\phi_k := g_{\theta_{\text{amort}}}(\mathbf{d}_k)$$

Amortized optimization: No gradient computation rather, use an encoder to predict the parameter

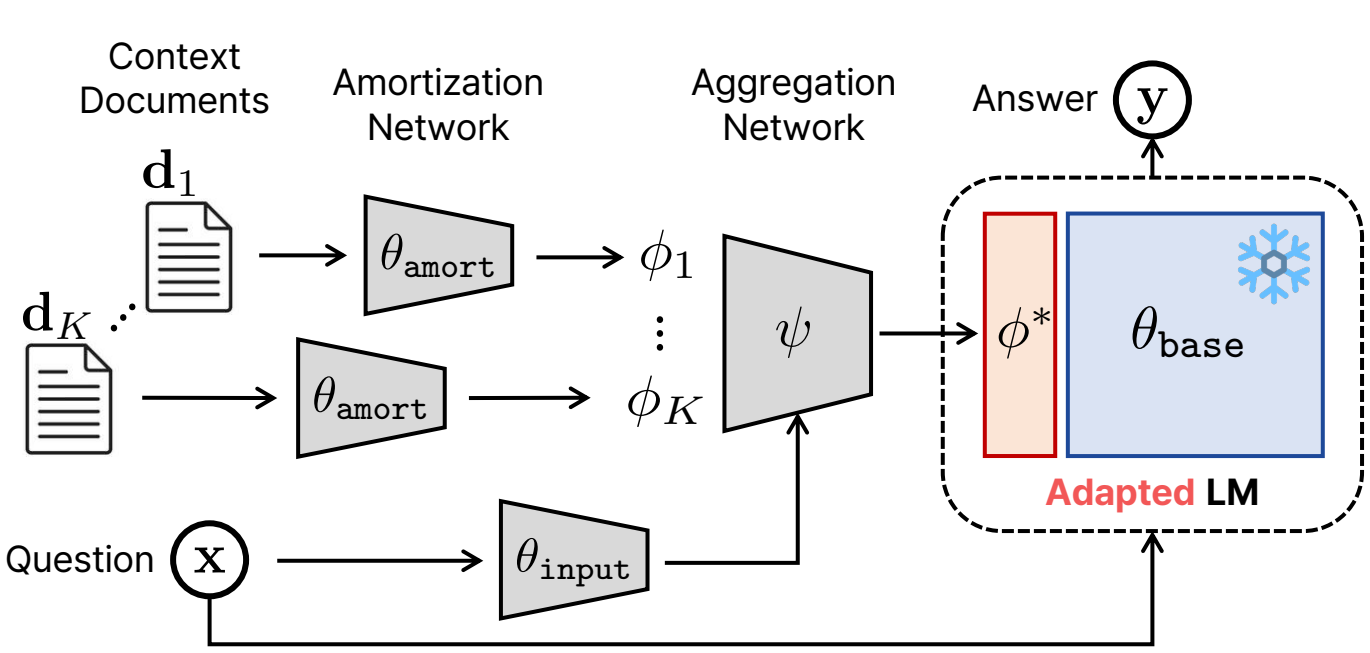
Inference: Online adaptation

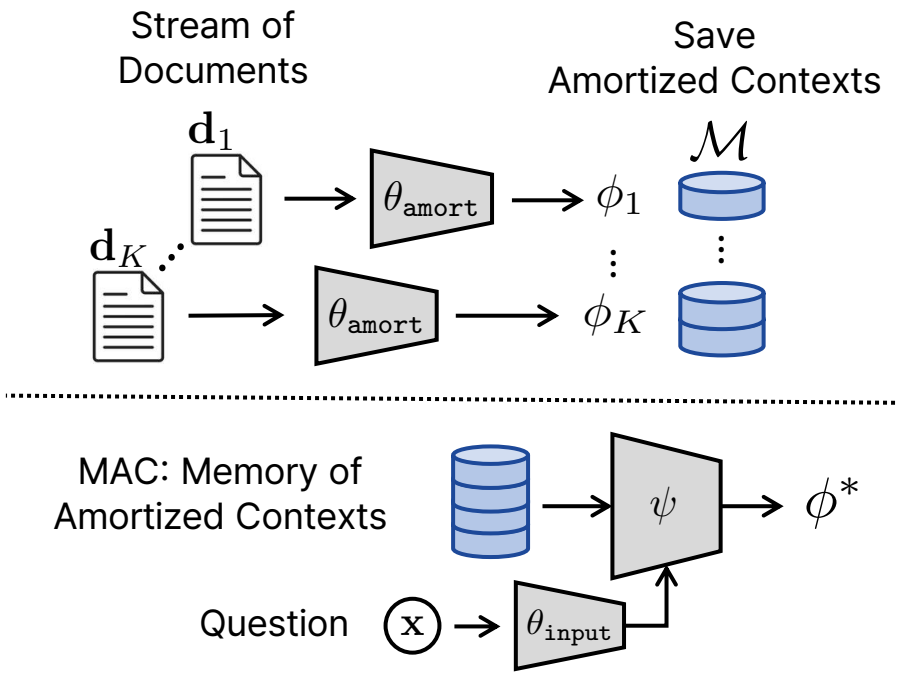


When the stream of documents are keep emerging

When the user question is raised

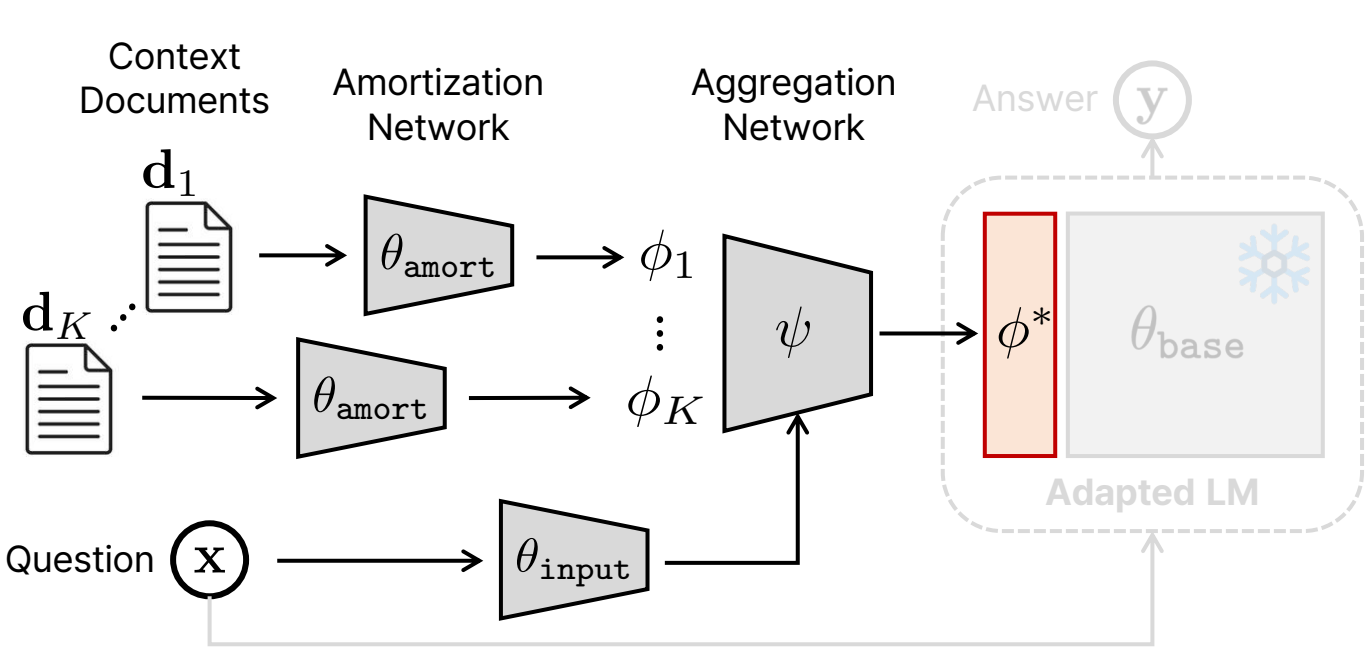
Back to the original question

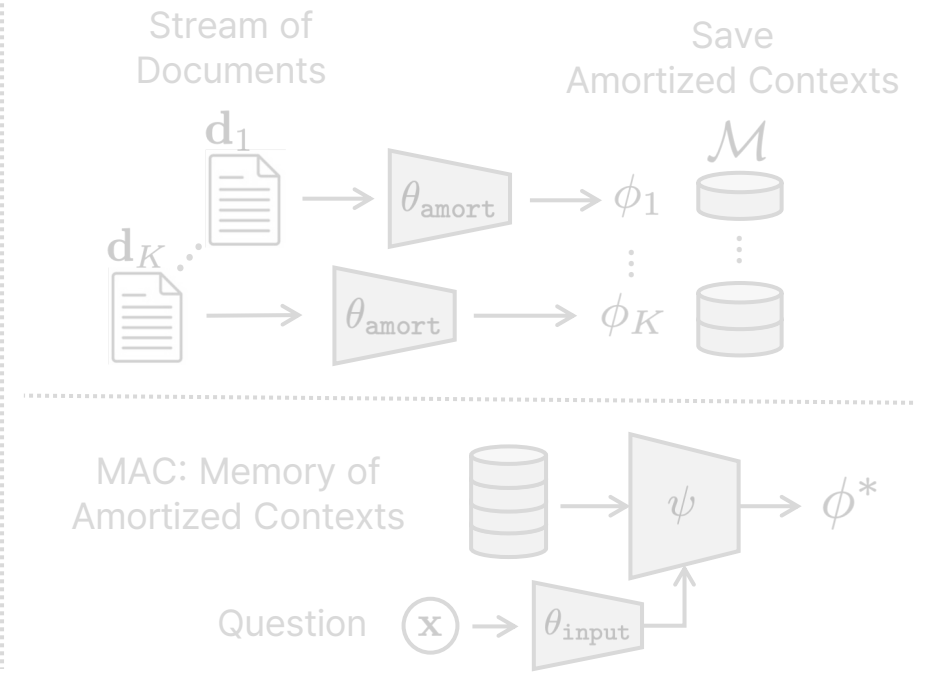




1) Efficient adaptation
2) Efficient inference
3) Strong retention of the learned knowledge
4) No hyper-parameter during the adaptation
5) Possibility of using the knowledge from similar documents

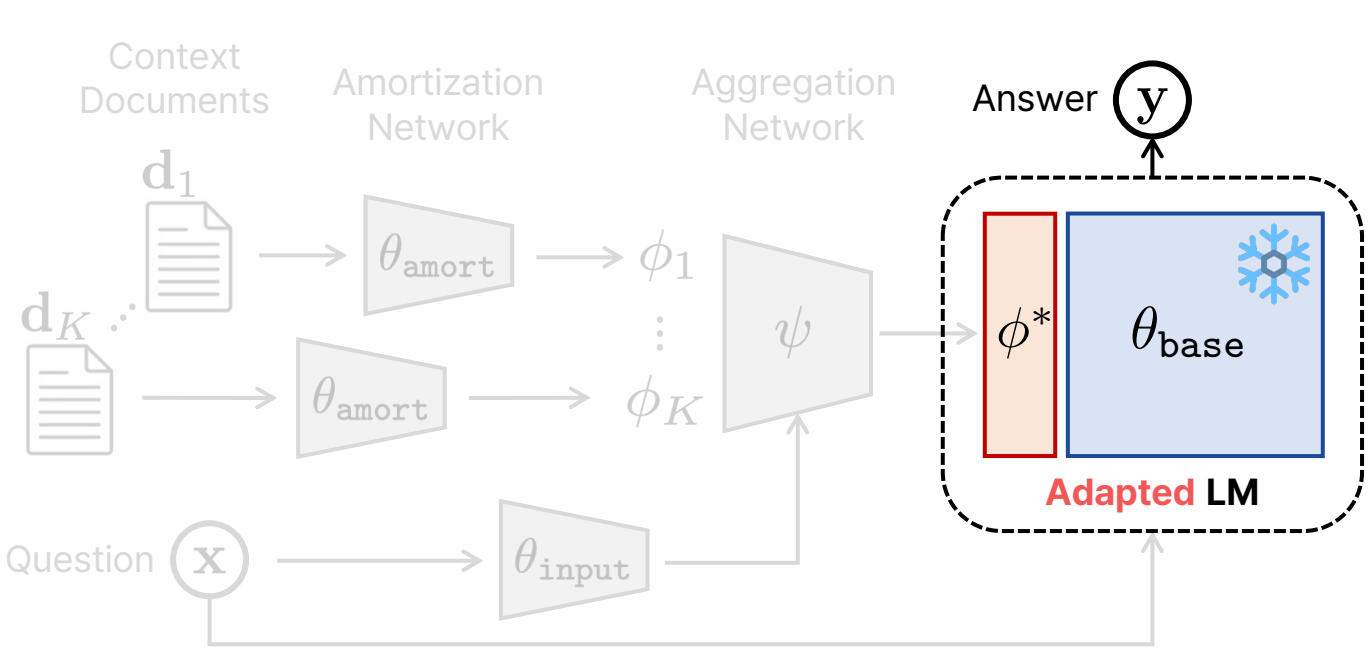
Back to the original question

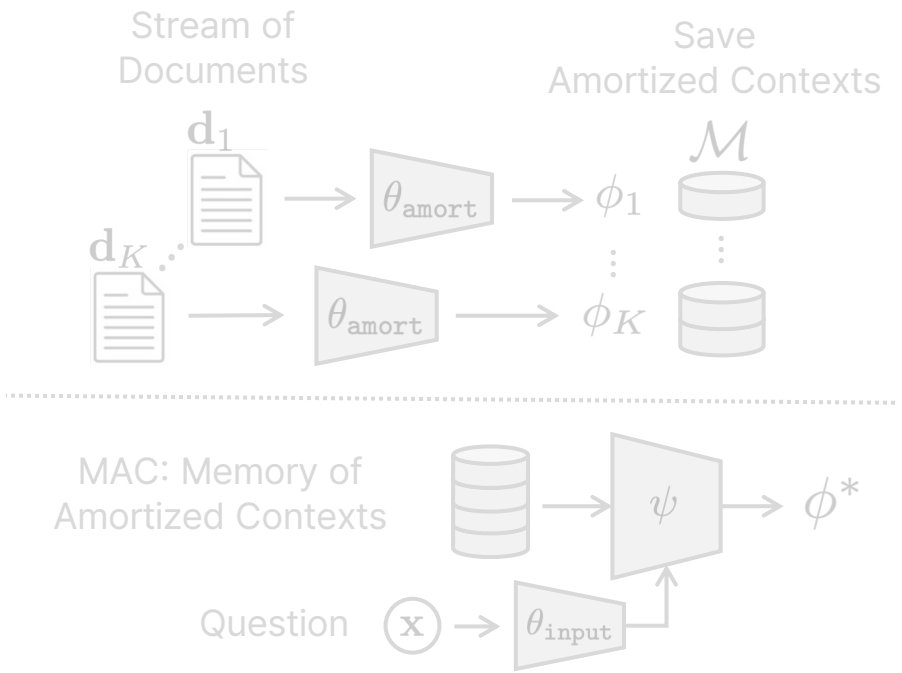




1) Efficient adaptation: **Only requires a single forward pass (no gradient computation)**
2) Efficient inference
3) Strong retention of the learned knowledge
4) No hyper-parameter during the adaptation
5) Possibility of using the knowledge from similar documents

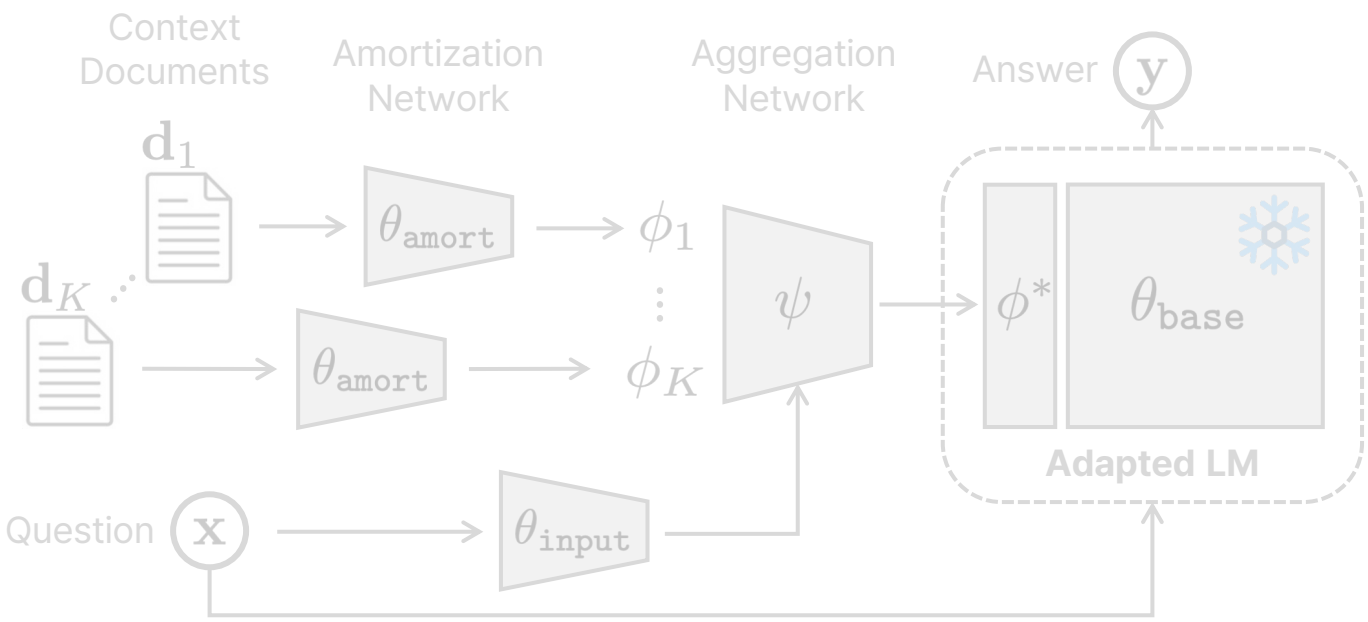
Back to the original question

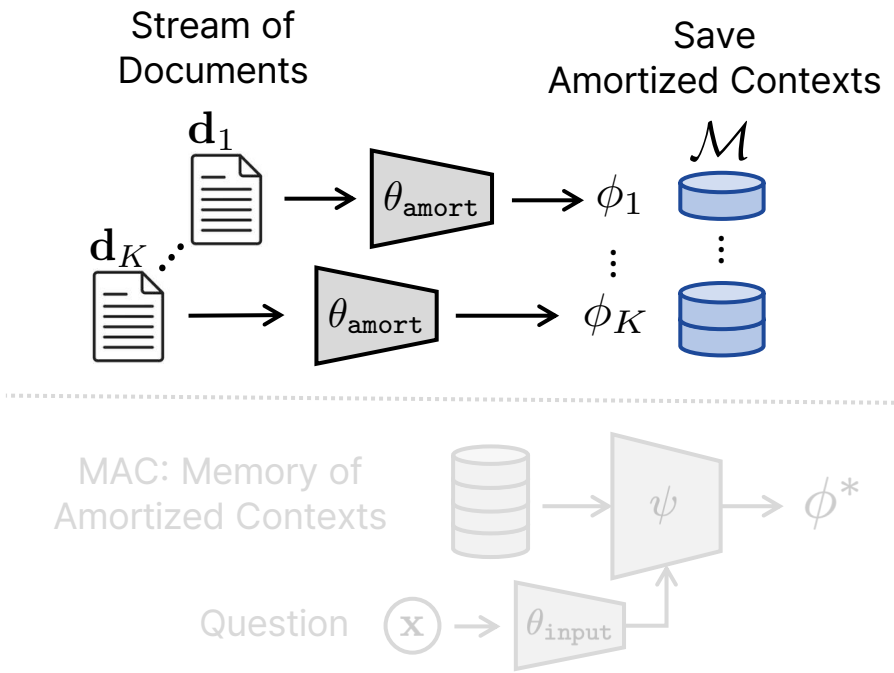




1) Efficient adaptation
2) Efficient inference: **LoRA or P-tuning V2 (i.e., ϕ) does not increase the context window size**
3) Strong retention of the learned knowledge
4) No hyper-parameter during the adaptation
5) Possibility of using the knowledge from similar documents

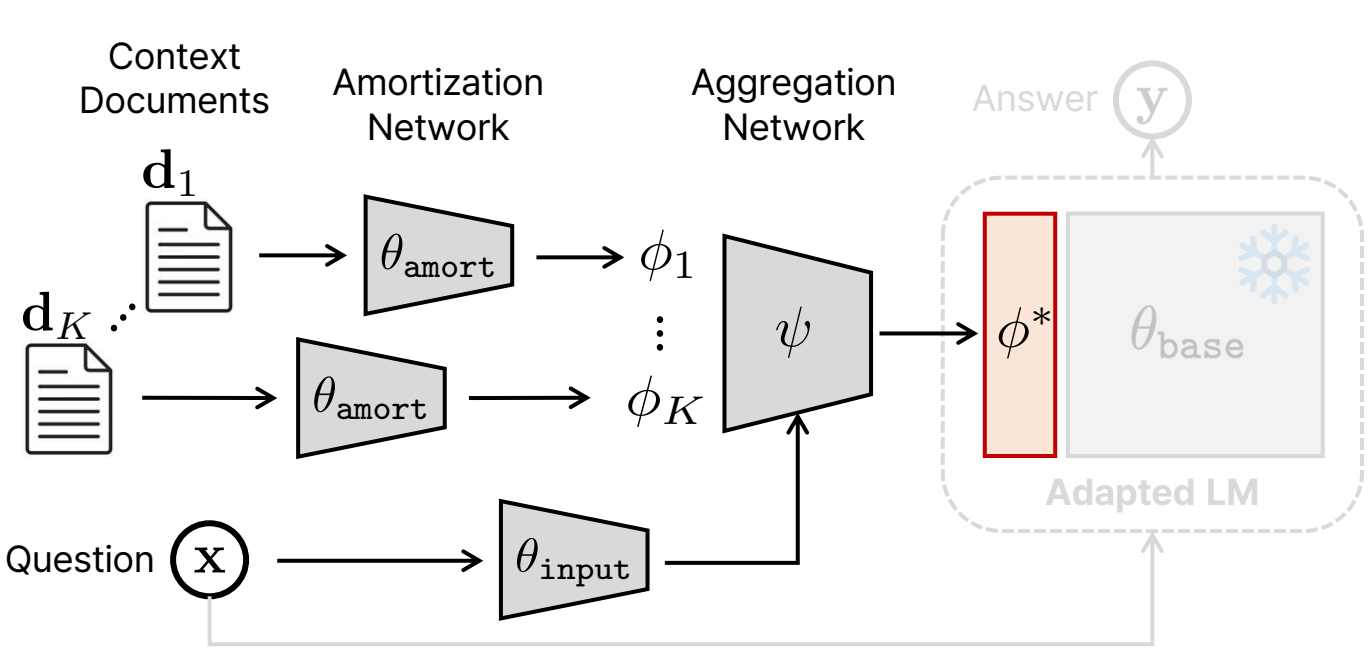
Back to the original question

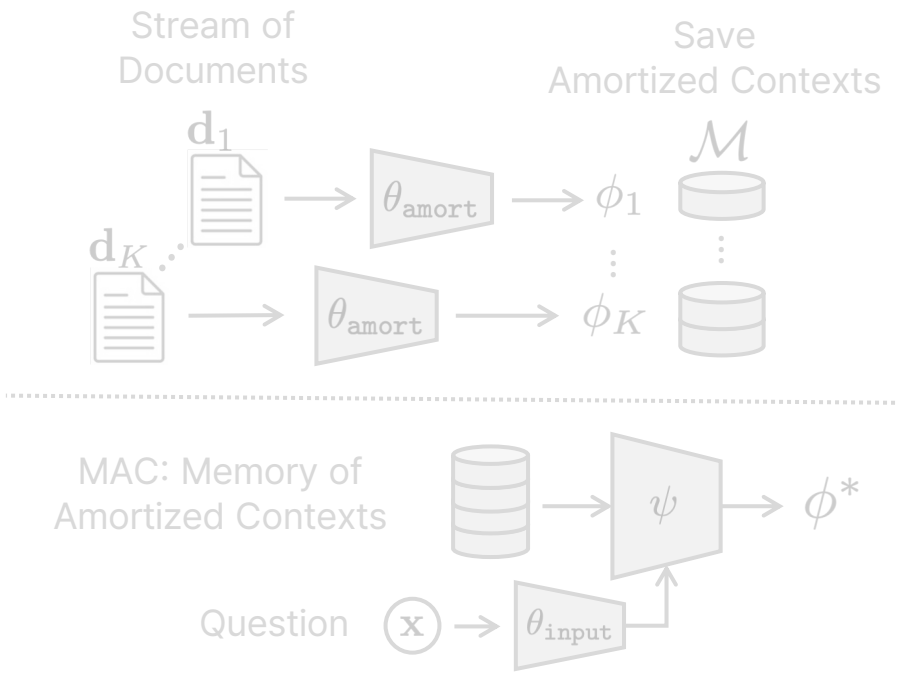




1) Efficient adaptation
2) Efficient inference
3) Strong retention of the learned knowledge: **the compact knowledge is saved into the memory bank**
4) No hyper-parameter during the adaptation
5) Possibility of using the knowledge from similar documents

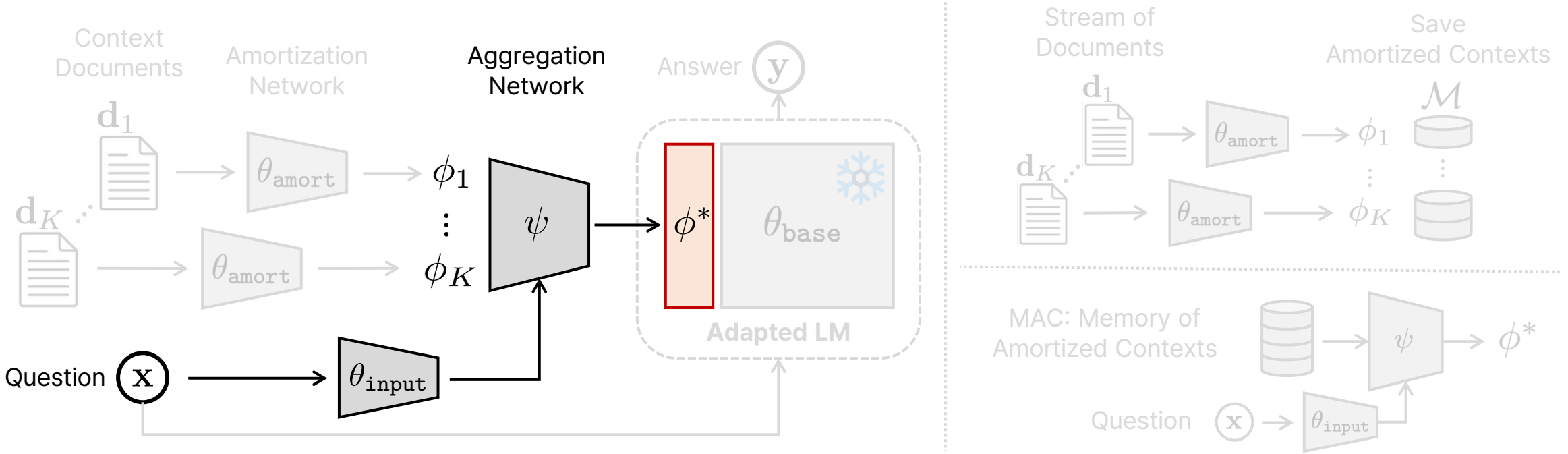
Back to the original question





1) Efficient adaptation
2) Efficient inference
3) Strong retention of the learned knowledge
4) No hyper-parameter during the adaptation: **Only requires a single forward pass +** **no document labels**
5) Possibility of using the knowledge from similar documents

Back to the original question



1) Efficient adaptation
2) Efficient inference
3) Strong retention of the learned knowledge
4) No hyper-parameter during the adaptation
5) Possibility of using the knowledge from similar documents: **Aggregate can use share the knowledge**

Model Soup: Worth to note that merging LoRAs for same checkpoints can even do better than individual LoRA

Two memory efficient techniques (Methods are in the Appendix)

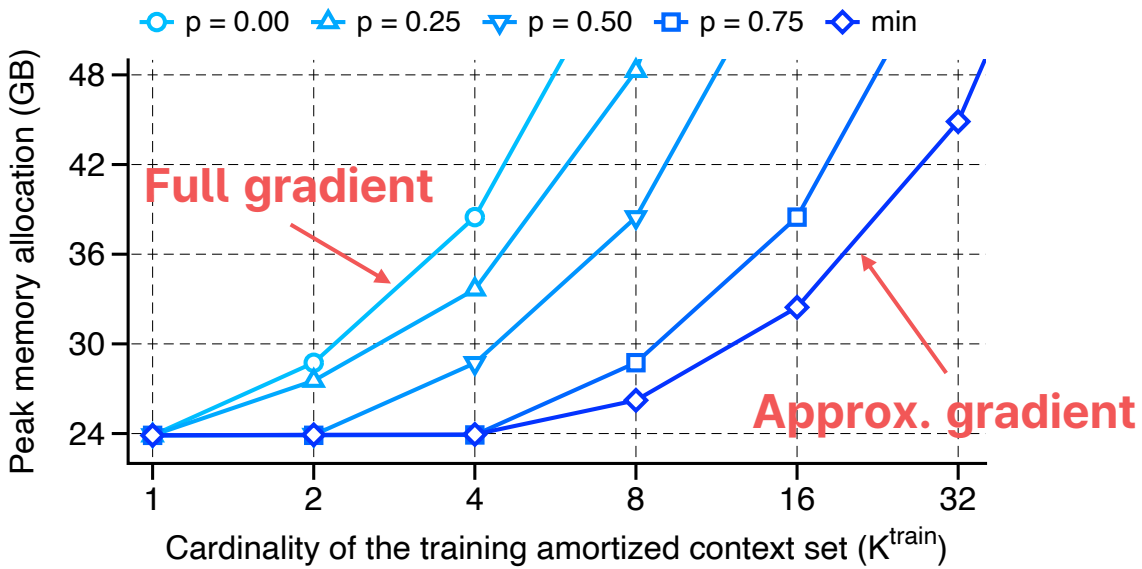
Memory efficient large batch training while the approximation is an unbiased gradient estimate

Memory efficient inference for a large memory bank while almost maintaining the performance

Two memory efficient techniques (Methods are in the Appendix)

Memory efficient large batch training while the approximation is an unbiased gradient estimate

Memory efficient inference for a large memory bank while almost maintaining the performance



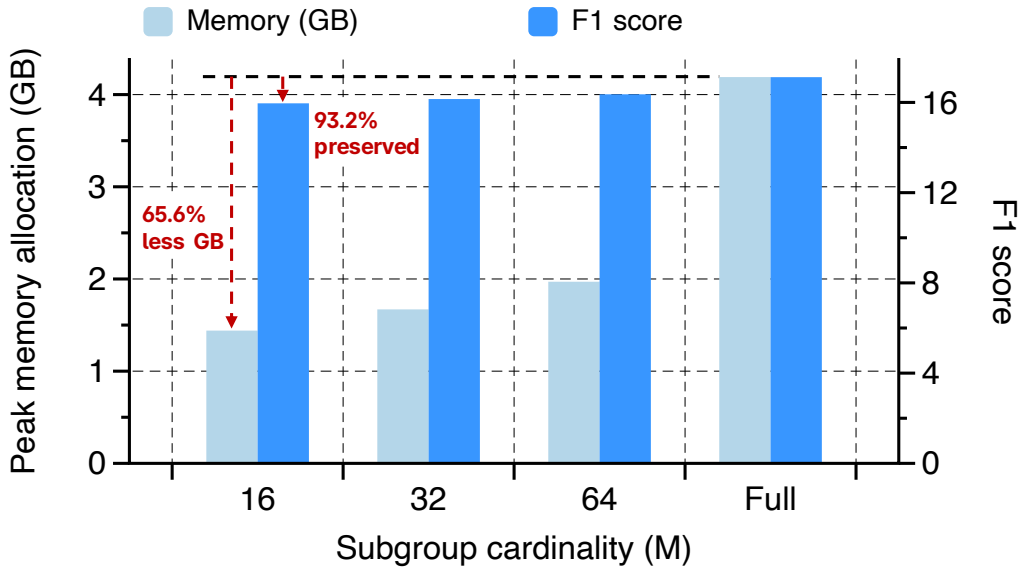
Training memory efficiency

Two memory efficient techniques (Methods are in the Appendix)

Memory efficient large batch training while the approximation is an unbiased gradient estimate

Memory efficient inference for a large memory bank while almost maintaining the performance

Training memory efficiency



Inference memory efficiency

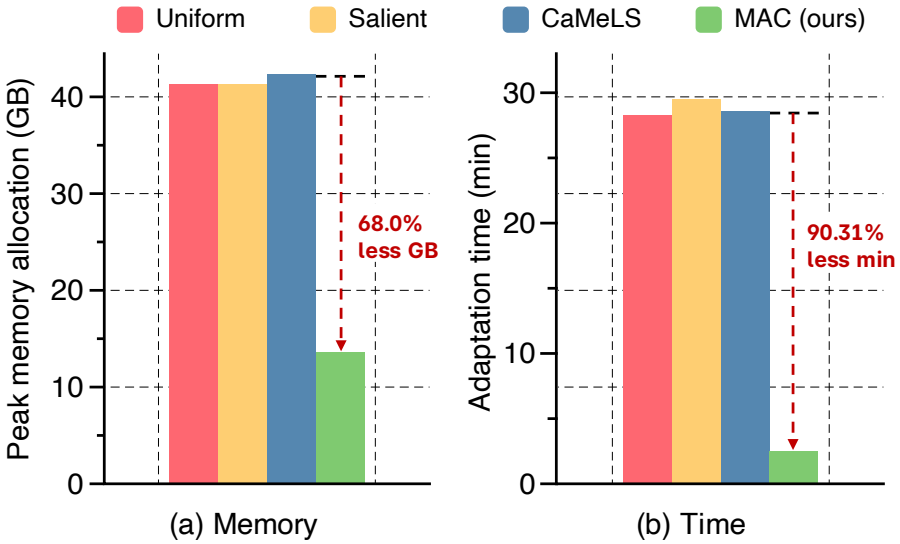
Experimental results: Comparison with online finetuning

Our method **outperforms online finetuning methods** in all benchmarks

- We adapt the LM on a stream of 1,665 documents and then perform QA

Table 1. Comparison of the online adaptation performance between MAC and online finetuning baselines. We report the exact match (EM) and F1 score by adapting the LM on a stream of documents and then performing QA based on the learned data. * denotes the adaptation results of CaMeLS's using a proxy token weighting model (i.e., a smaller model than the base LM) due to memory consumption, and OOM denotes unavailable results due to the running out-of-memory on a single NVIDIA A100 80GB GPU (even with a batch size of 1). The **bold** indicates the best result within the group.

		StreamingQA		SQuAD-Seq		ArchivalQA-Seq	
Model (# params)	Method	EM (↑)	F1 (↑)	EM (↑)	F1 (↑)	EM (↑)	F1 (↑)
DistilGPT2 (82M)	Uniform (Hu et al., 2023)	1.62	3.76	1.24	2.54	4.86	4.08
	Salient Spans (Hu et al., 2023)	1.44	4.67	1.03	2.47	4.52	3.76
	CaMeLS (Hu et al., 2023)	1.62	5.79	1.47	3.08	4.62	6.19
	MAC (ours)	**5.59**	**10.18**	**2.01**	**6.85**	**7.55**	**10.58**
GPT2-Large (774M)	Uniform (Hu et al., 2023)	4.74	7.00	3.64	4.97	7.66	8.71
	Salient Spans (Hu et al., 2023)	4.86	8.54	4.03	6.48	9.75	11.19
	CaMeLS* (Hu et al., 2023)	5.35	10.60	4.97	8.63	9.92	12.41
	MAC (ours)	**7.25**	**13.31**	**6.43**	**11.42**	**11.84**	**15.26**
GPT2-XL (1.5B)	Uniform (Hu et al., 2023)	5.11	7.48	6.10	6.78	8.61	10.78
	Salient Spans (Hu et al., 2023)	5.40	9.42	4.55	6.74	11.81	14.11
	CaMeLS* (Hu et al., 2023)	6.55	11.67	6.70	10.15	13.87	15.74
	MAC (ours)	**8.99**	**15.38**	**7.10**	**12.55**	**14.01**	**17.12**
LLaMA-2 (7B)	Uniform (Hu et al., 2023)	12.43	13.54	13.25	17.01	18.53	21.35
	Salient Spans (Hu et al., 2023)	13.33	18.97	13.74	18.66	18.97	22.75
	CaMeLS (Hu et al., 2023)	OOM					
	MAC (ours)	**14.29**	**21.79**	**15.07**	**21.14**	**20.12**	**23.90**



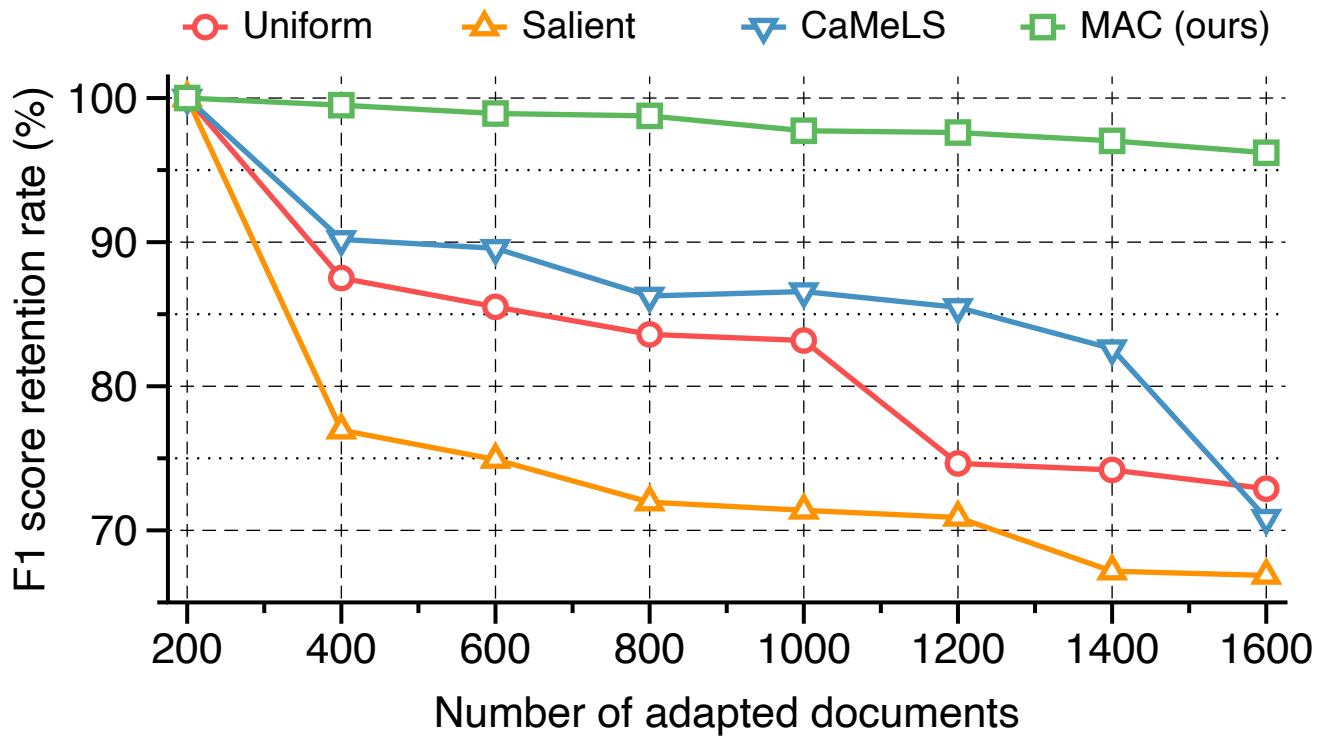
Also **much more efficient** than others

Experimental results: Comparison with online finetuning

Our method **outperforms online finetuning methods** in all benchmarks

- We adapt the LM on a stream of 1,665 documents and then perform QA

Also, the **knowledge retention is much better** than the online finetuning methods



Experimental results: Joint usage with retrieval augmentation

Our method can also be **jointly** used with **retrieval augmentation methods**

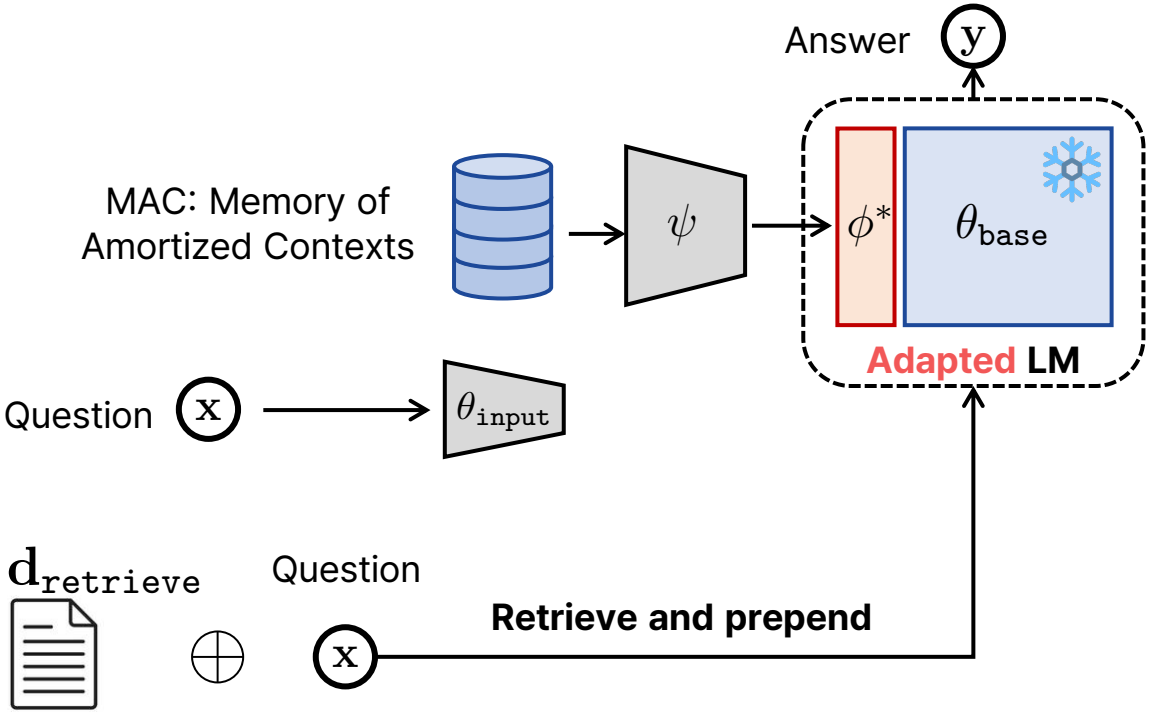
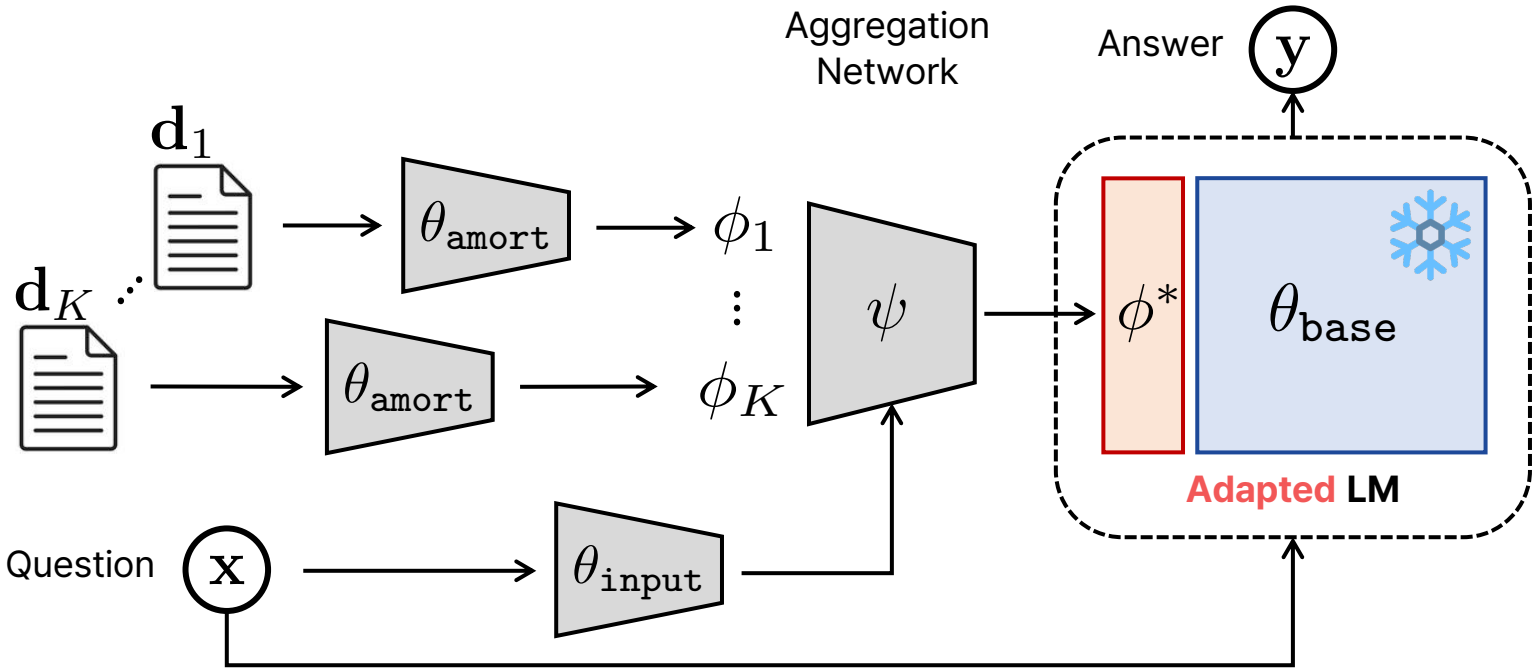


Table 2. Online adaptation performance of MAC jointly using the retrieval augmentation under ArchivalQA-Seq dataset. We report the exact match (EM) and F1 score by adapting the LM on a stream of documents and then performing QA based on the learned data, while retrieval augmentation retrieves the top-1 document. The **bold** indicates the best results within the group.

Model	Method	EM (↑)	F1 (↑)
DistilGPT2 (82M)	Contriever (Izacard et al., 2022)	9.28	12.41
	BM25 (Robertson et al., 2009)	10.90	14.50
	BM25 + MAC (ours)	**12.22**	**16.05**
GPT2-Large (774M)	Contriever (Izacard et al., 2022)	13.30	17.13
	BM25 (Robertson et al., 2009)	16.33	21.43
	BM25 + MAC (ours)	**22.54**	**28.10**
GPT2-XL (1.5B)	Contriever (Izacard et al., 2022)	13.99	17.28
	BM25 (Robertson et al., 2009)	17.79	22.58
	BM25 + MAC (ours)	**24.23**	**29.93**
LLaMA-2 (7B)	Contriever (Izacard et al., 2022)	21.52	28.31
	BM25 (Robertson et al., 2009)	25.11	31.30
	BM25 + MAC (ours)	**31.62**	**40.11**

Appendix - 1: Backpropagation dropout

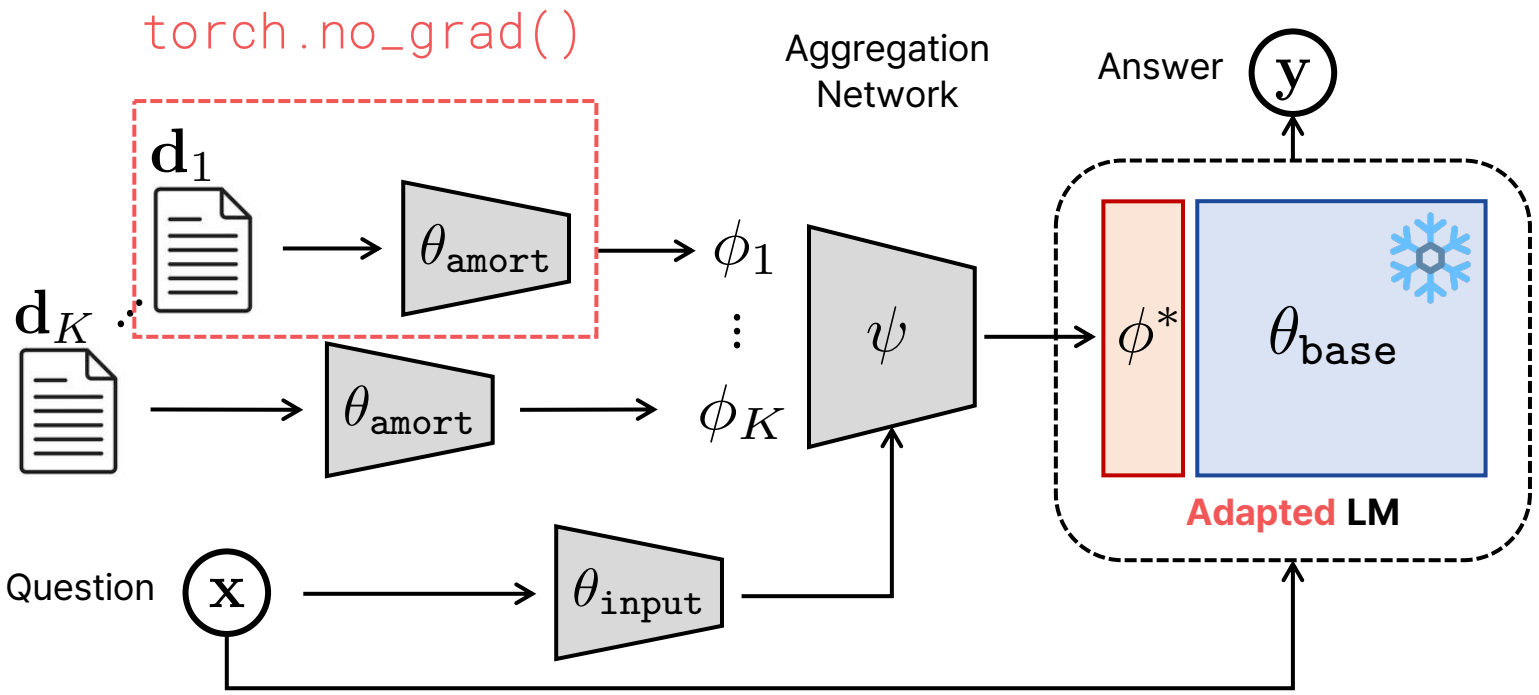
Large document batch (i.e., $K \gg 1$) training can cause a memory issue



Appendix - 1: Backpropagation dropout

Large document batch (i.e., $K \gg 1$) training can cause a memory issue

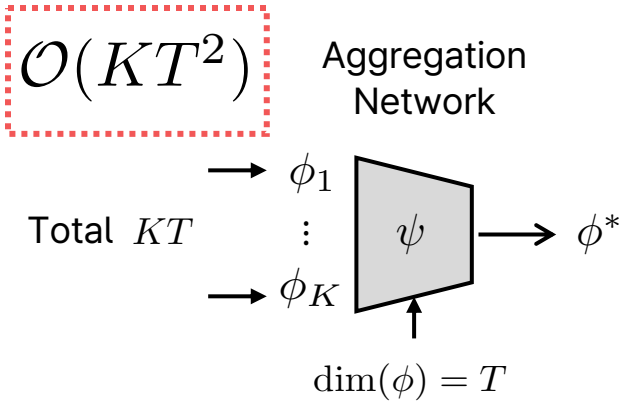
Randomly dropout the backpropagation of the amortization network



This yields an unbiased approximation of the full gradient

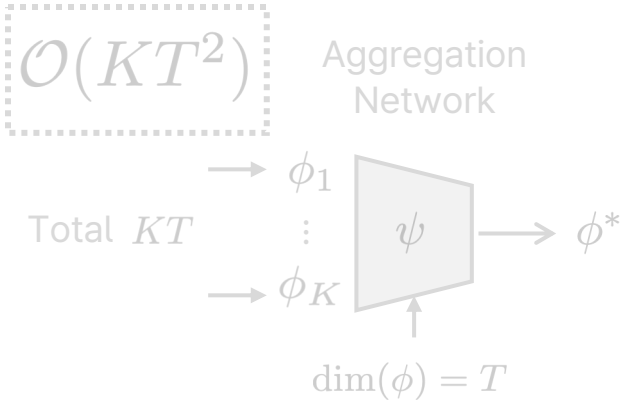
Appendix - 2: Hierarchical modulation aggregation

Memory inefficiency when aggregating a large memory bank



Appendix - 2: Hierarchical modulation aggregation

Memory inefficiency when aggregating a large memory bank



Algorithm 1 Hierarchical modulation aggregation

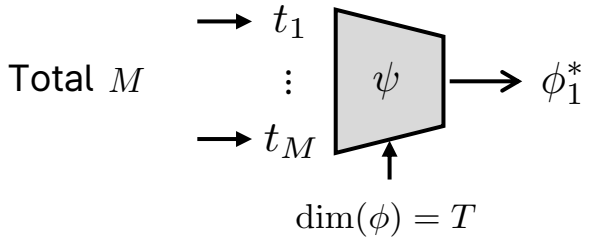
Input: $\mathcal{M}$, ψ, $\mathbf{x}$, $\theta_{\texttt{input}}$, subgroup cardinality M

while $|\mathcal{M}| > 1$ **do**
  Subgroup $\mathcal{M}$ into M tokens $\{\mathcal{M}_1, \cdots, \mathcal{M}_{\lceil \frac{|\mathcal{M}|}{M} \rceil}\}$
  Initialize new memory bank $\mathcal{M}_{\texttt{new}} := \emptyset$
  for all $i = 1$ to $\lceil \frac{|\mathcal{M}|}{M} \rceil$ **do**
    Aggregate subgroup $\phi_i \leftarrow h_\psi\left(g_{\theta_{\texttt{input}}}(\mathbf{x}), \mathcal{M}_i\right)$
    Store ϕ_i into $\mathcal{M}_{\texttt{new}}$
  end for
  Repeat by $\mathcal{M} \leftarrow \mathcal{M}_{\texttt{new}}$
end while

Output: $\mathcal{M} = \{\phi^*\}$

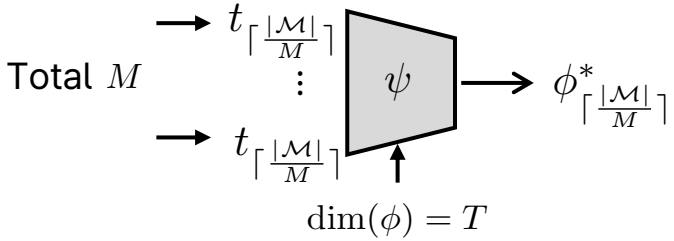
Assuming no parallelization, the memory usage is $\mathcal{O}(MT)$

$\mathcal{O}(MT)$



⋮

$\mathcal{O}(MT)$



Save into a new memory bank (**cache**)

$$\mathcal{M}_{\texttt{new}} := \{\phi_1^*, \cdots, \phi^*_{\lceil \frac{|\mathcal{M}|}{M} \rceil}\}$$

Repeat until the new memory bank size = 1