Efficient LLMs: Retrieval Augmentation

Chenyan Xiong

11-667

Outline

Retrieval-Augmentation in Pretraining

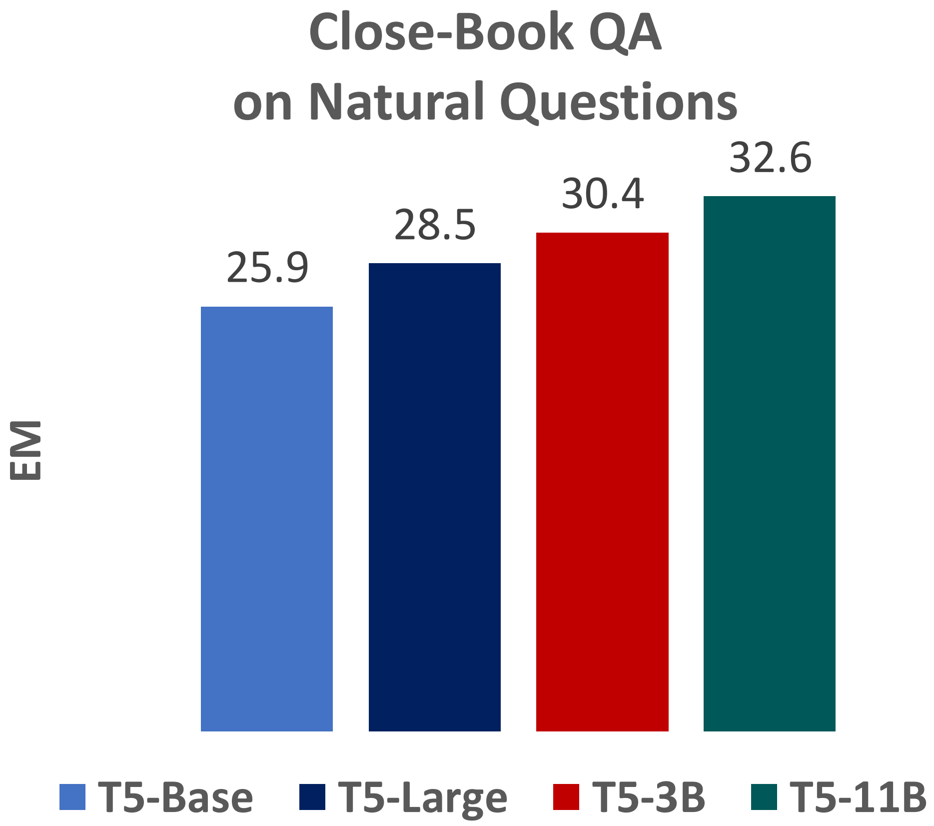
- Overview
- Popular Retrieval Augmented LLMs: KNN-LM, REALM, and RETRO
- Empirical Results
- Recap

Retrieval-Augmentation after Pretraining

- Overview
- Augmenting Training Data Points
- Augmenting Knowledge/Information
- Adapting Retriever for LLM

Retrieval-Augmented Pretraining: Motivation

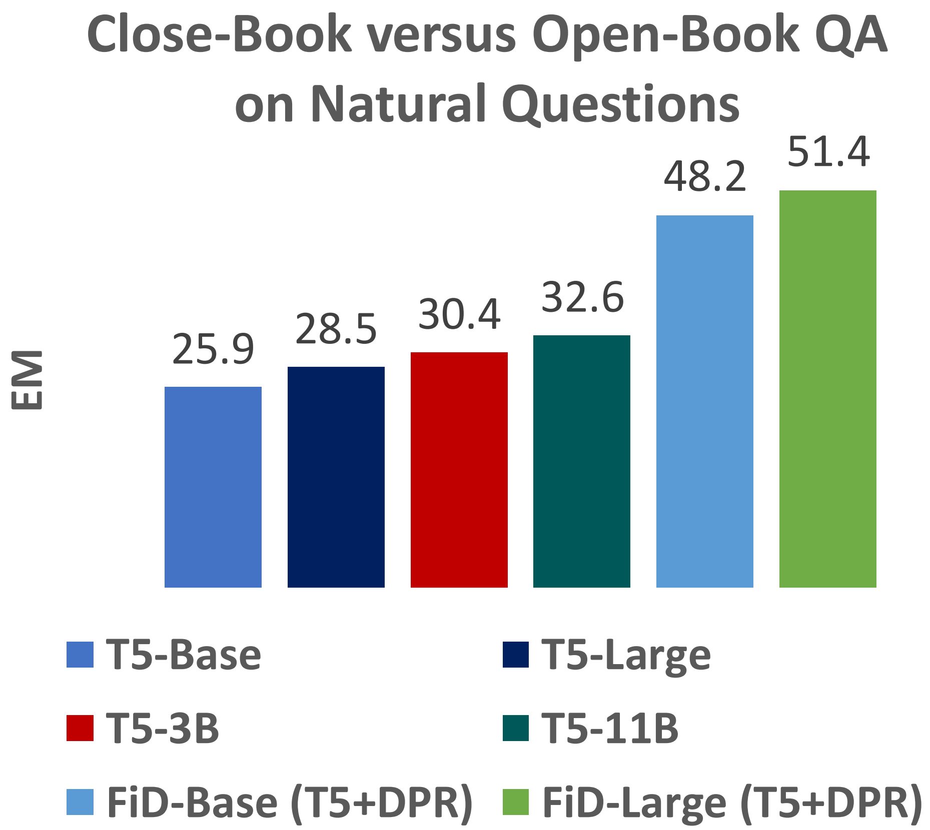
LLMs can memorize lots of knowledge in its parameters [1]



[1] Roberts et al. How Much Knowledge Can You Pack Into the Parameters of a Language Model? EMNLP 2020

Retrieval-Augmented Pretraining: Motivation

LLMs can memorize lots of knowledge in its parameters [1]



Clear benefits from adding external information from a retrieval system on knowledge-intensive tasks

- Better overall accuracy
- Significantly better parameter efficiency
- More reliable/explicit source of information

[1] Roberts et al. How Much Knowledge Can You Pack Into the Parameters of a Language Model? EMNLP 2020

Memorization or Understanding?

Memorization:

- Memorize pretraining data in parameter

Understanding:

- Learn to consume and utilize information

Memorization or Understanding?

Memorization:

- Memorize pretraining data in parameter

Understanding:

- Learn to consume and utilize information

Retrieval-Augmentation

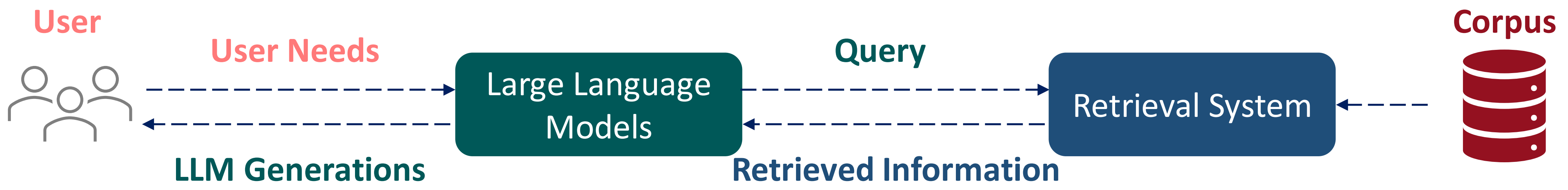
Can we alleviate LLMs from costly parametric memory by augmenting them with a retrieval system?

- Retrieval system serves as an external memory
- LLMs learn to leverage retrieved information

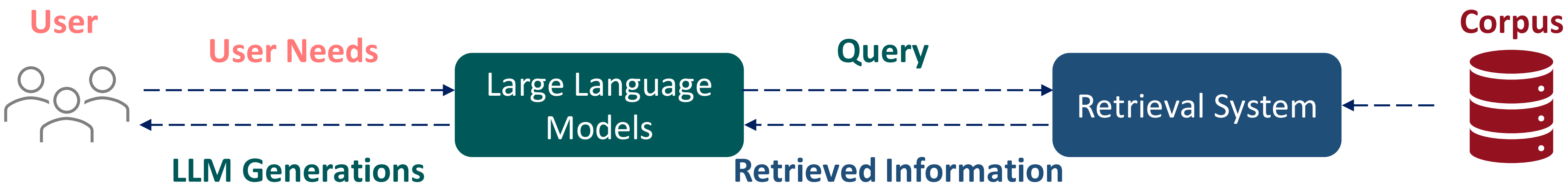
Ideally:

- More efficient parameter usage
- Better generalization ability by switching external memory (retrieval corpus)
- Clear separation of memorization and understanding for transparency and control

Retrieval-Augmented LM: Overview



Retrieval-Augmented LM: Overview



Core design questions:

- **When** to retrieve?
- **What** to retrieve?
- **How** to retrieve?
- **Where** to retrieve?
- **How** to leverage retrieved information?

Outline

Retrieval-Augmentation in Pretraining

- Overview
- **Popular Retrieval Augmented LLMs: KNN-LM, REALM, and RETRO**
- Empirical Results
- Recap

Retrieval-Augmentation after Pretraining

- Overview
- Augmenting Training Data Points
- Augmenting Knowledge/Information
- Adapting Retriever for LLM

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

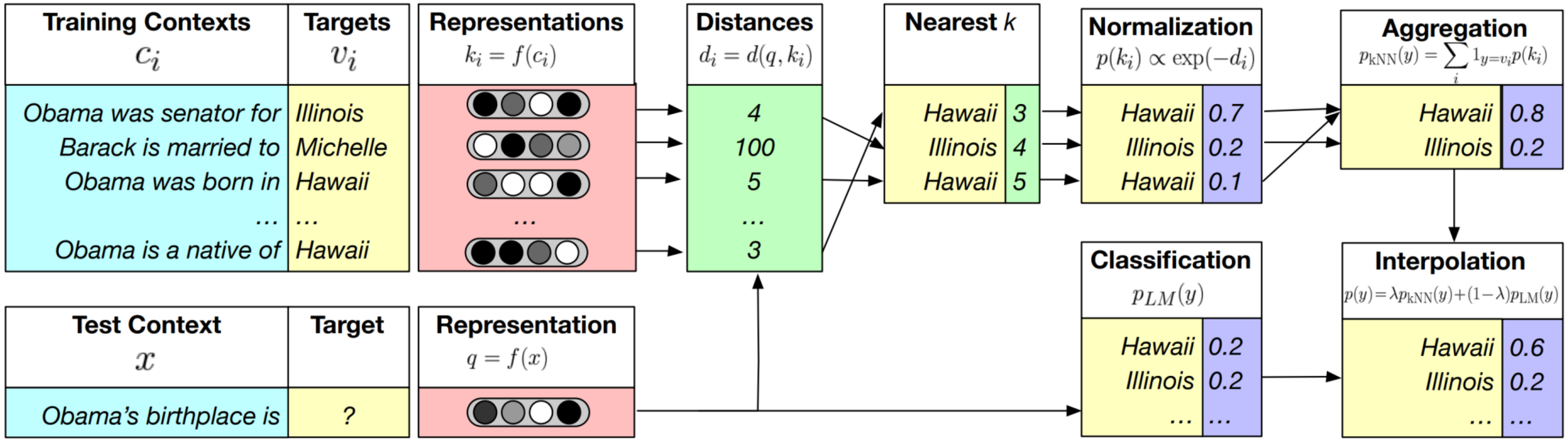


Figure 1: The pipeline of KNN-LM [2]

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

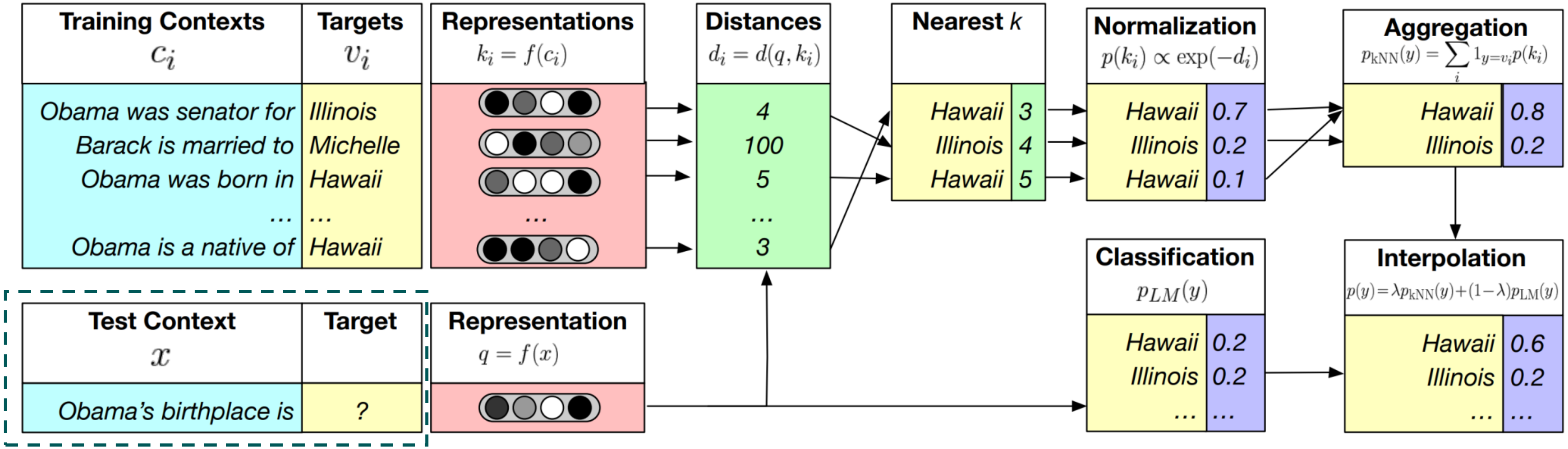


Figure 1: The pipeline of KNN-LM [2]

When to retrieve?

- Every token position at inference time

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

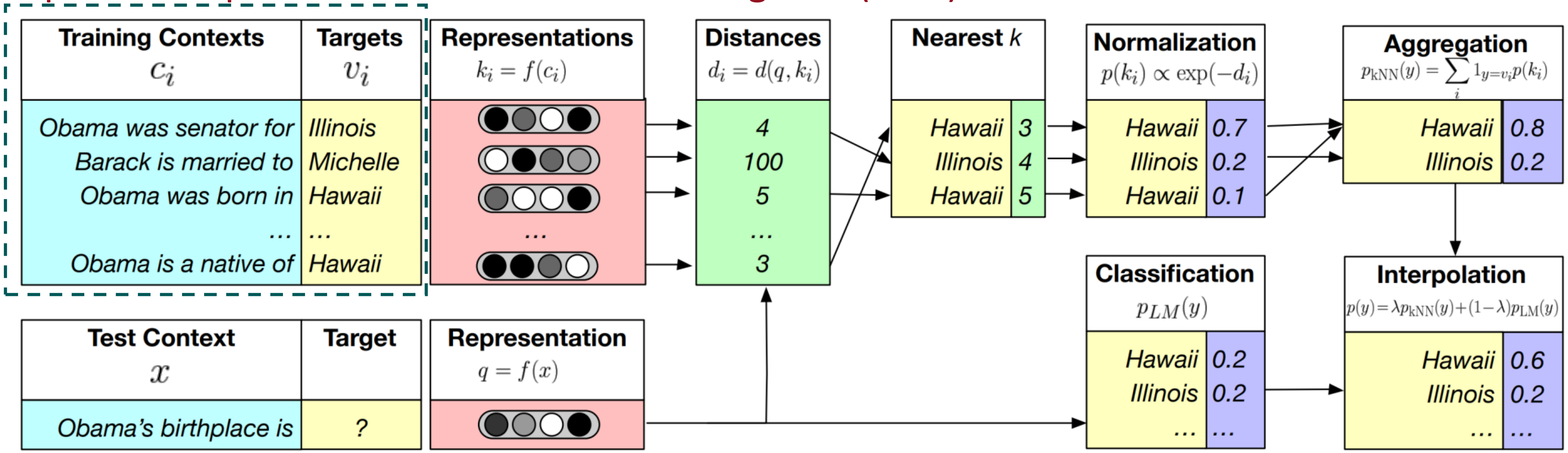


Figure 1: The pipeline of KNN-LM [2]

What to retrieve?

- (Context, Target Word) pairs: (c_i, v_i)
- Key: $k = f(c_i)$ the hidden representation of context from the LM
- Value: the actual ground truth word for the context

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

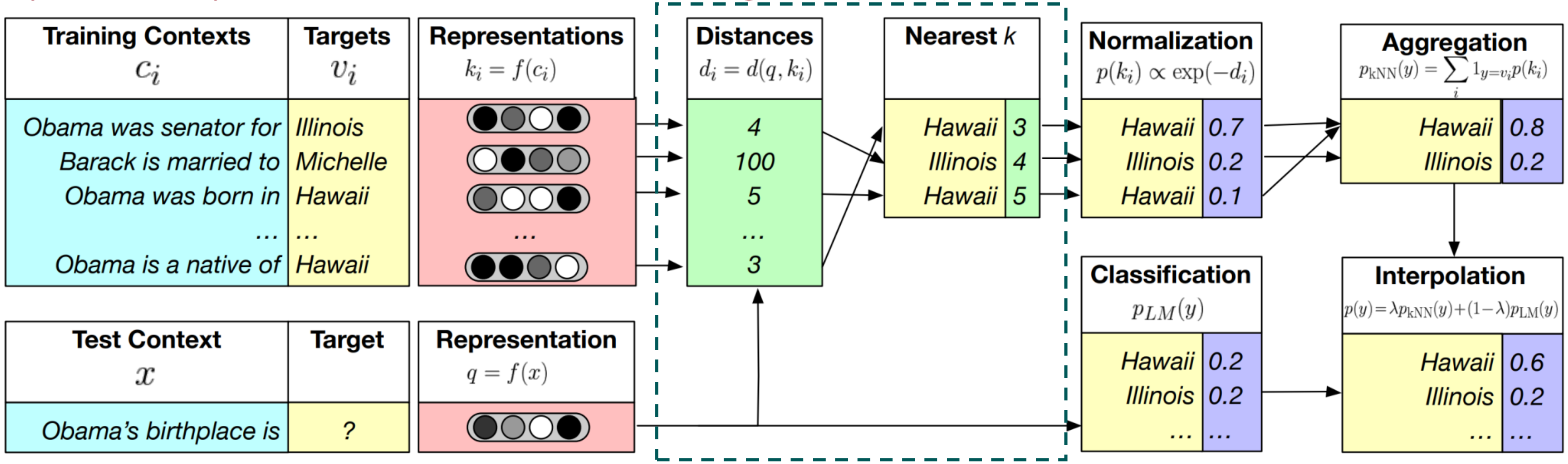


Figure 1: The pipeline of KNN-LM [2]

How to retrieve?

- K nearest neighbor search using current context x
- Query: $q = f(x)$ the hidden representation of the inference context
- Retrieval Function: standard KNN search with L2 distance

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

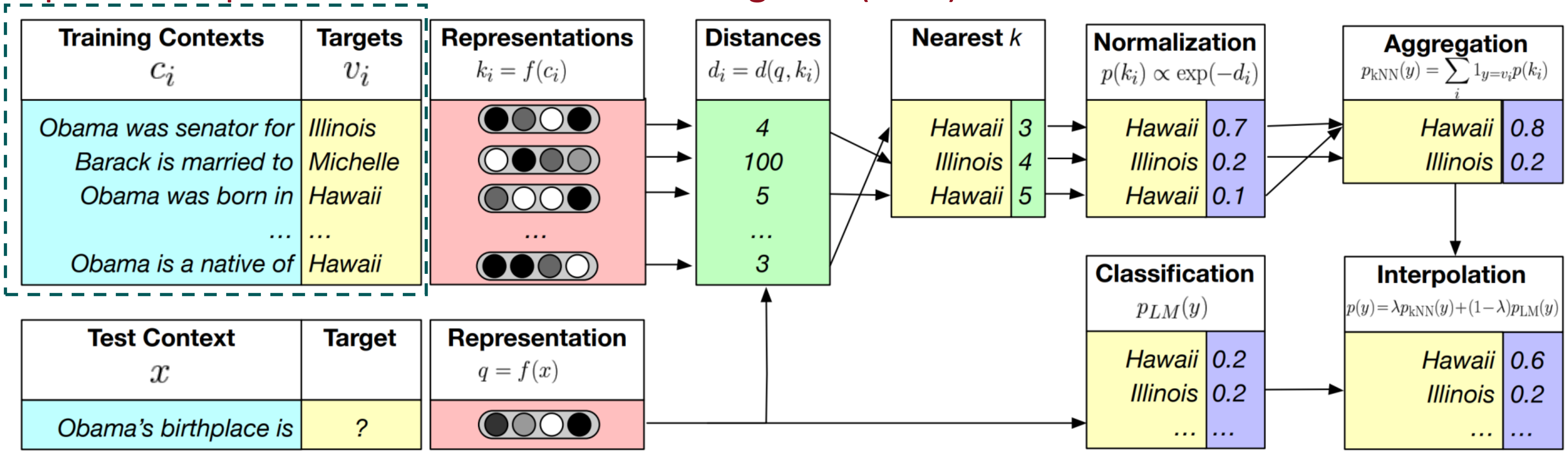


Figure 1: The pipeline of KNN-LM [2]

Where to retrieve?

- The training corpus with the pretrained LM to formulate the key-value pairs
- Or plug-in a new corpus to generalize to the new domain. E.g. Wiki→Book

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

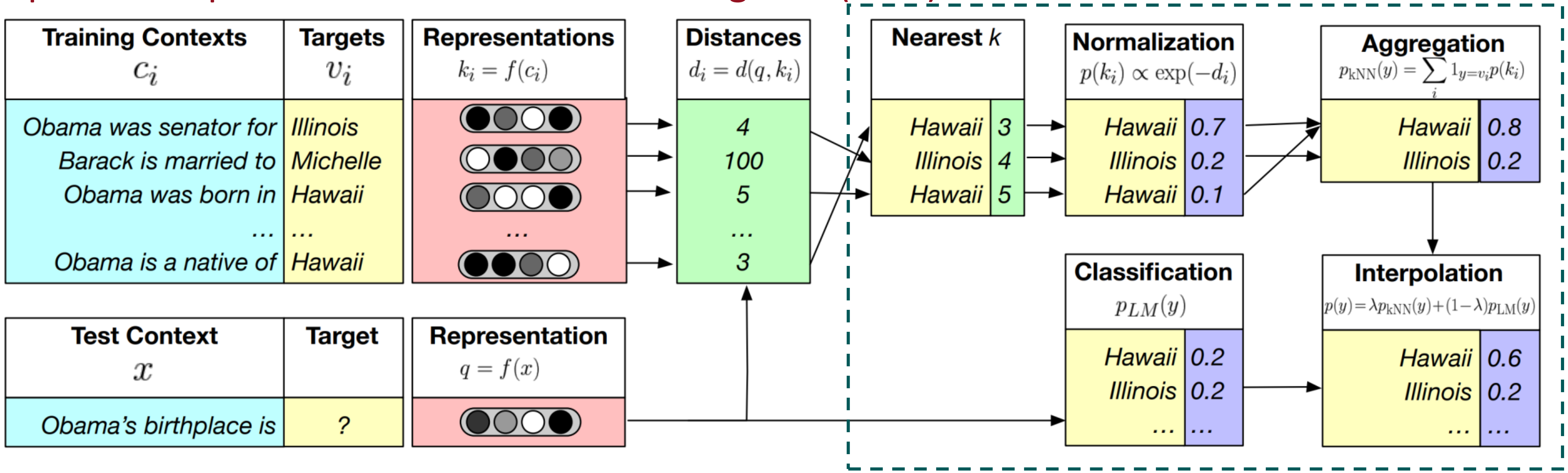


Figure 1: The pipeline of KNN-LM [2]

How to leverage retrieved information?

- Normalize and aggregate retrieved scores to obtain the KNN output
- Linearly interpolate the original LM output with KNN output

$$p(y) = \lambda p_{\text{knn}}(y) + (1 - \lambda) p_{\text{lm}}(y)$$

[2] Khandelwal et al. Generalization through Memorization: Nearest Neighbor Language Models. ICLR 2020

Popular Retrieval-Augmented LLM: KNN-LM

KNN-LM: Interpolate LM prediction with a k-nearest neighbor (KNN) model

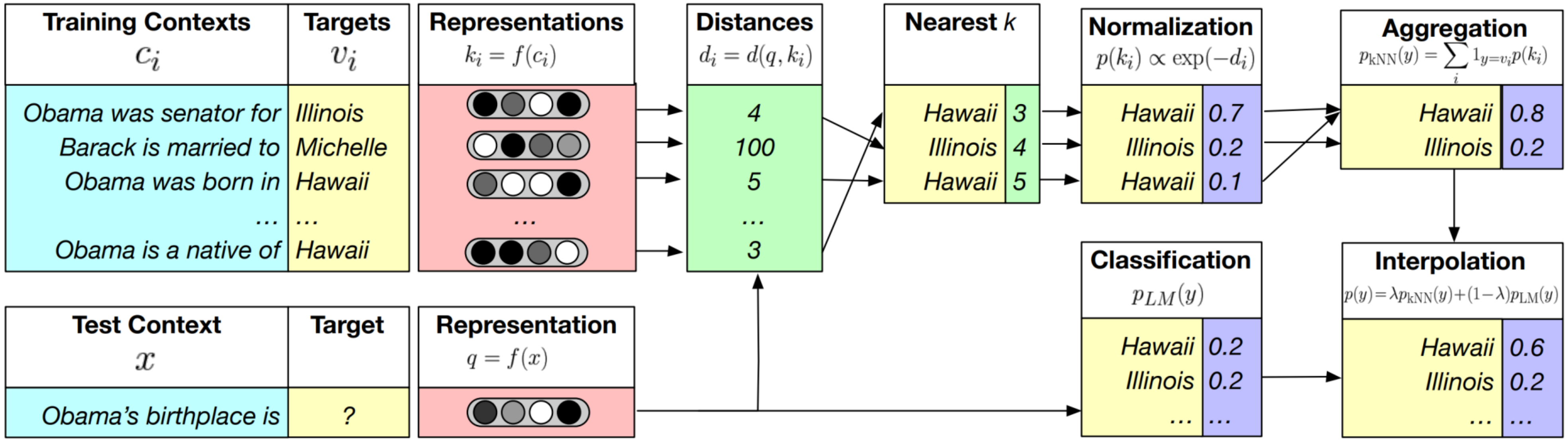


Figure 1: The pipeline of KNN-LM [2]

Recap

1. Build retrieval corpus in the format of context-target, essentially (x, y) pairs from training data
2. Retrieve nearest neighbors of test x at inference time
3. Interpolate model output with retrieved nearest neighbors' target y

- No training performed, all retrieval at testing time

[2] Khandelwal et al. Generalization through Memorization: Nearest Neighbor Language Models. ICLR 2020

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

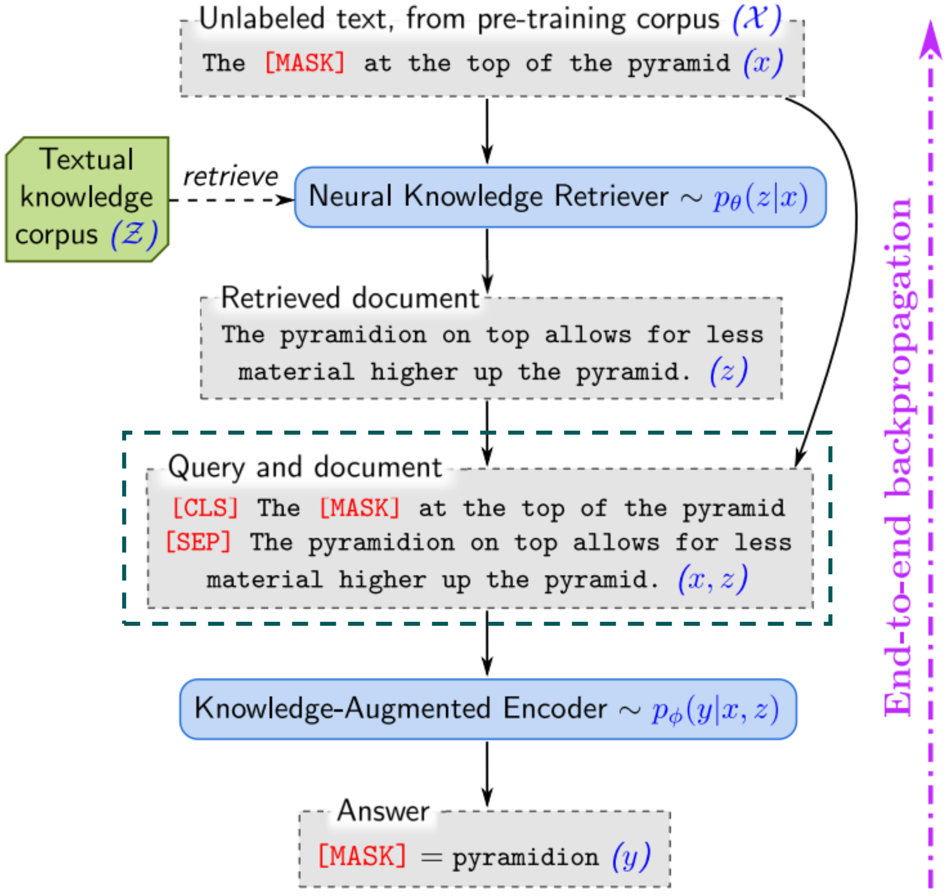


Figure 2: The pipeline of REALM [3]

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

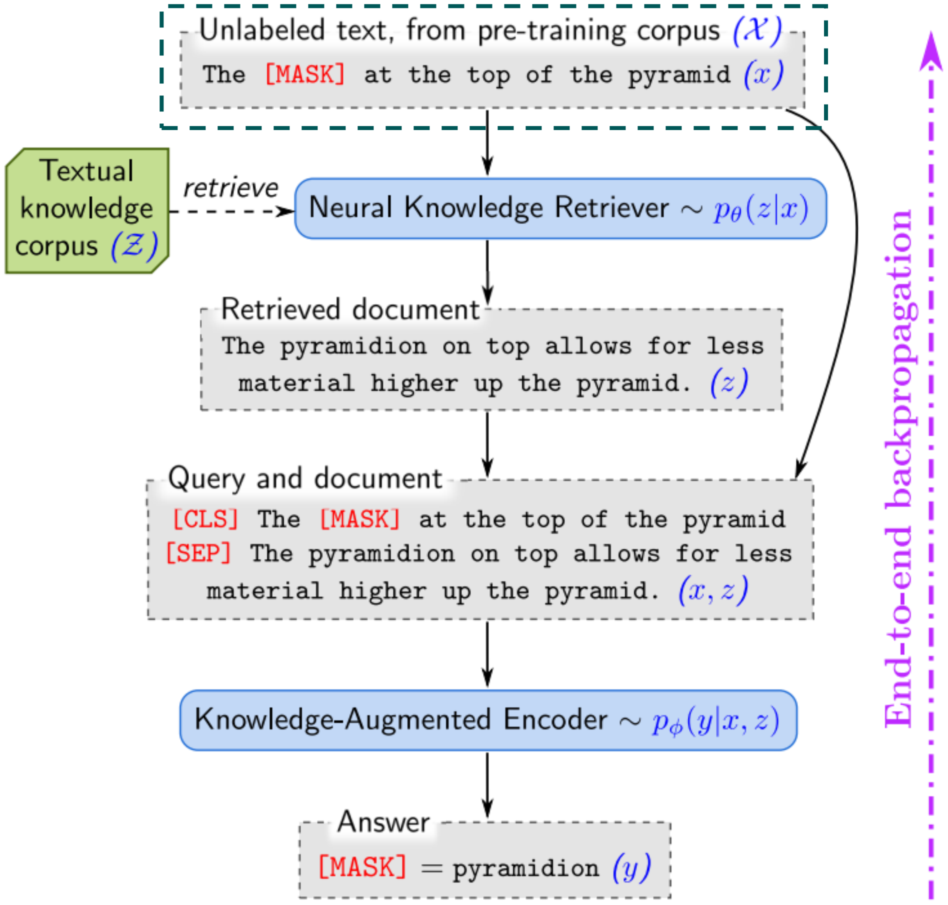


Figure 2: The pipeline of REALM [3]

When to retrieve?

- Once every training/testing sequence

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

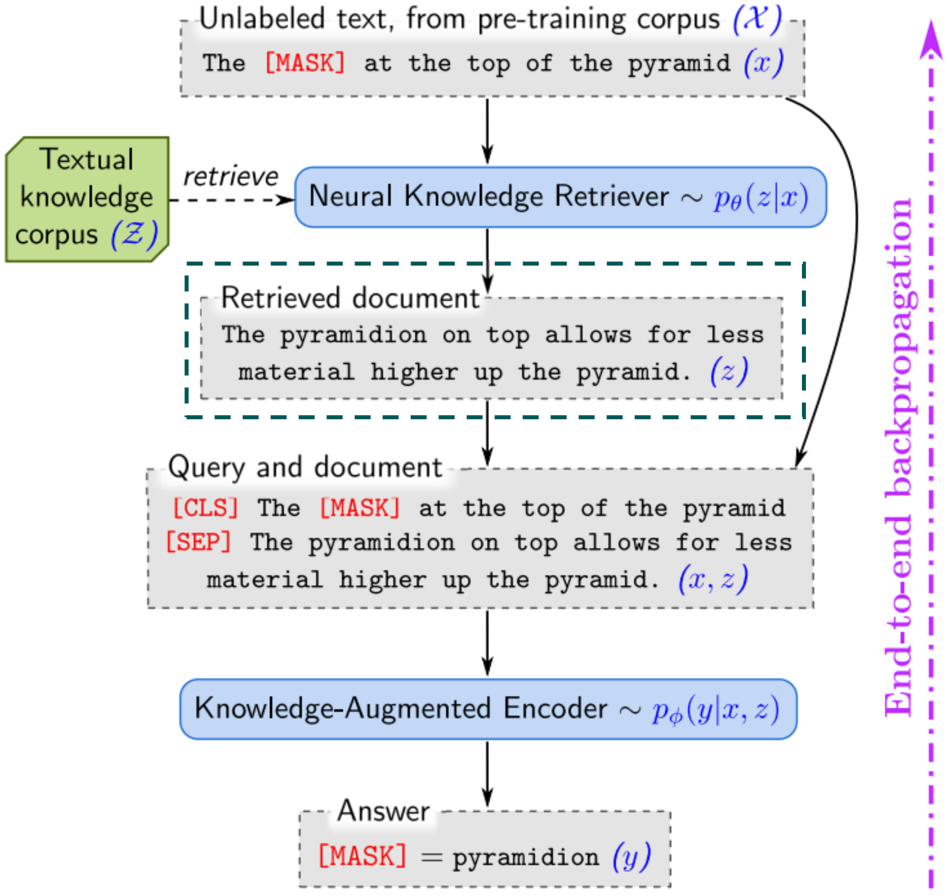


Figure 2: The pipeline of REALM [3]

When to retrieve?

- Once every training/testing sequence

What to retrieve?

- Similar text sequences

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

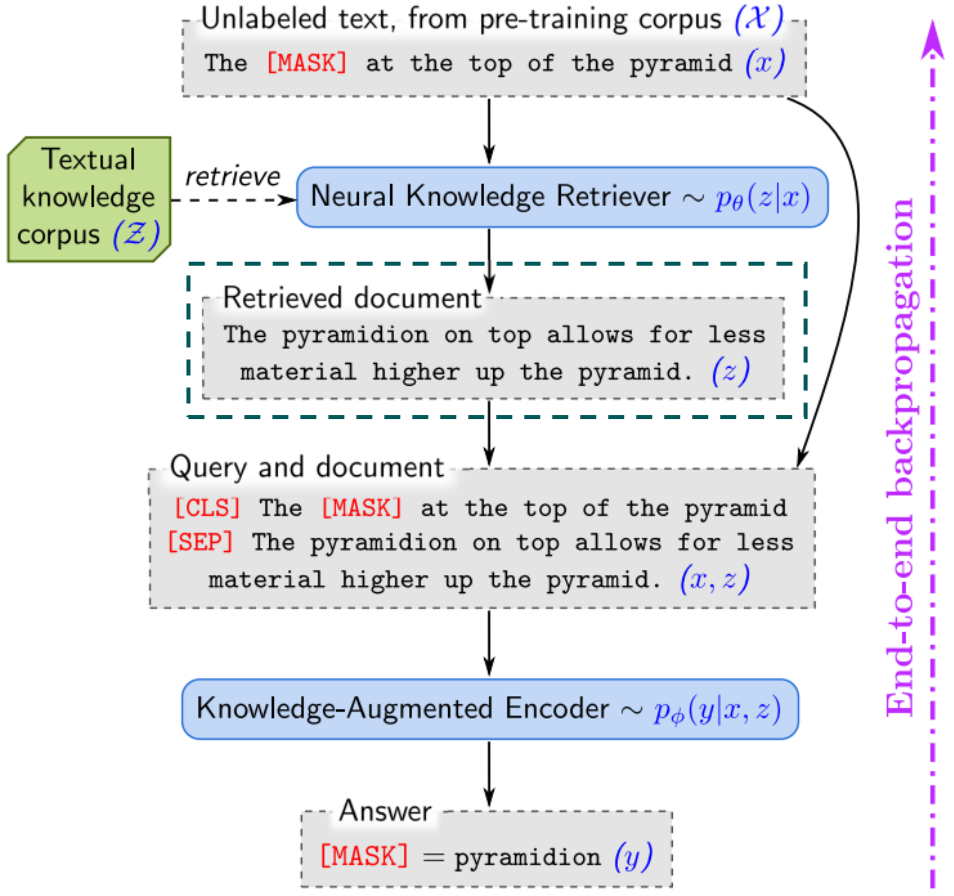


Figure 2: The pipeline of REALM [3]

When to retrieve?

- Once every training/testing sequence

What to retrieve?

- Similar text sequences

How to retrieve?

- Dense retriever (BERT here) using current sequence as the query

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

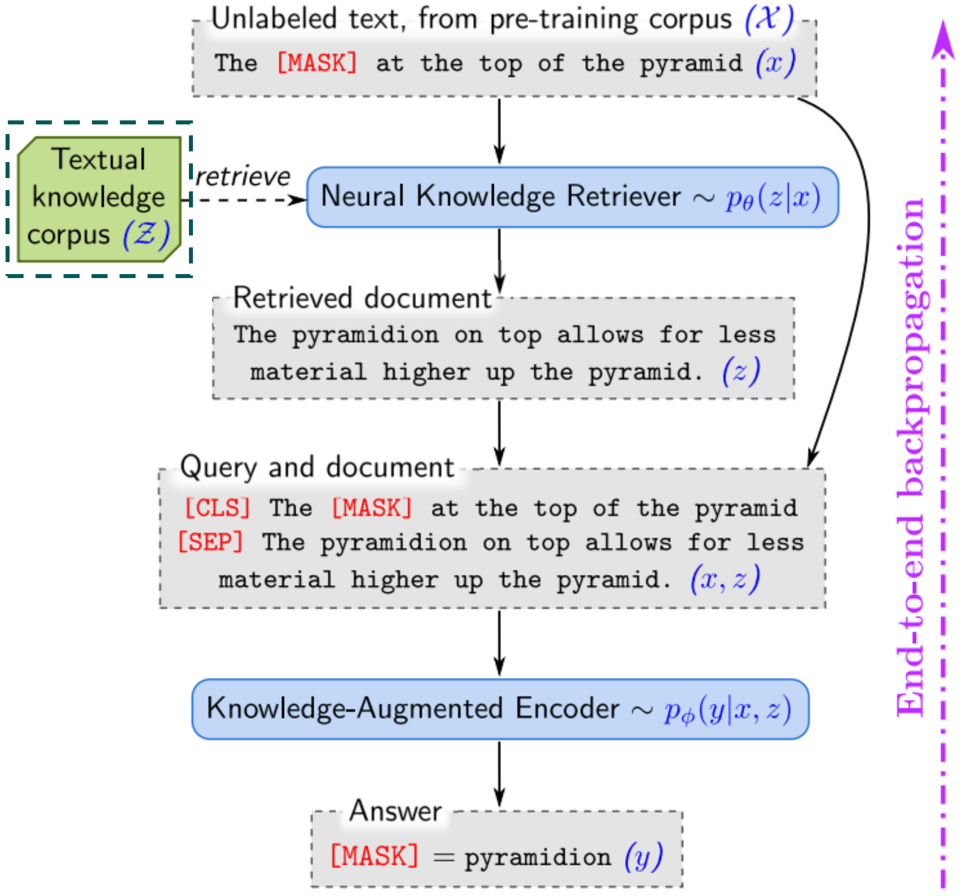


Figure 2: The pipeline of REALM [3]

When to retrieve?
- Once every training/testing sequence

What to retrieve?
- Similar text sequences

How to retrieve?
- Dense retriever (BERT here) using current sequence as the query

Where to retrieve?
- The same pretraining corpus

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

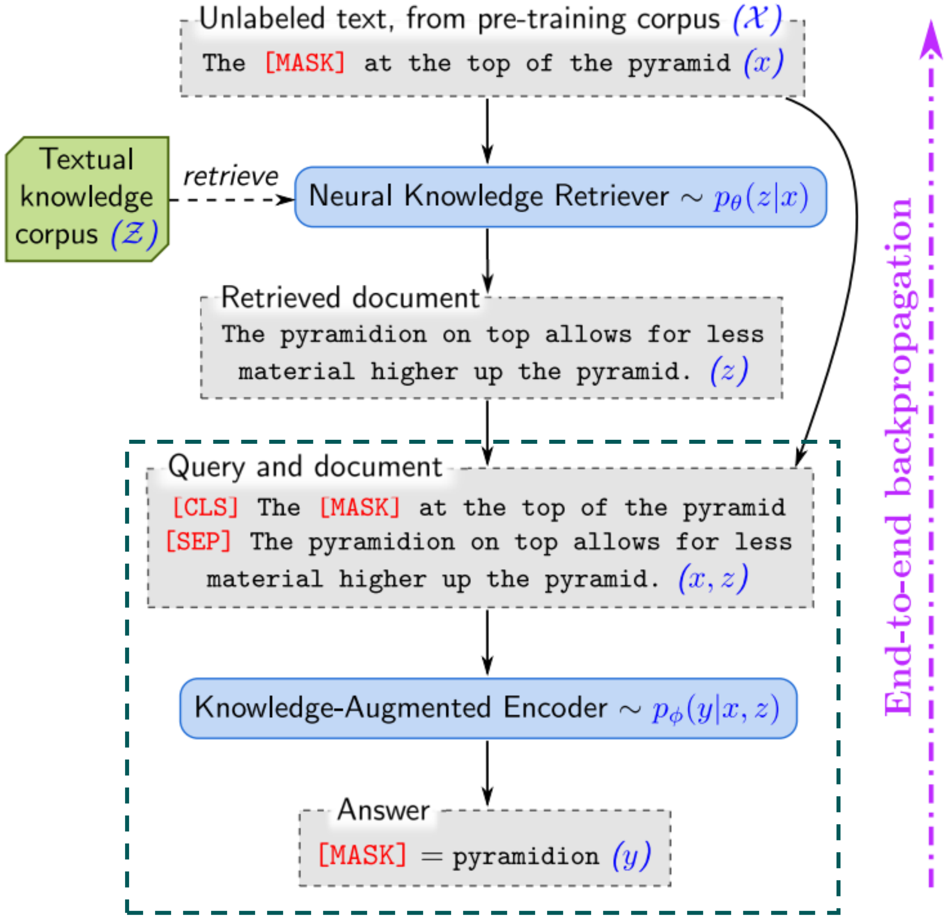


Figure 2: The pipeline of REALM [3]

When to retrieve?
- Once every training/testing sequence

What to retrieve?
- Similar text sequences

How to retrieve?
- Dense retriever (BERT here) using current sequence as the query

Where to retrieve?
- The same pretraining corpus

How to leverage retrieved information?
- Add retrieved sequence as extra inputs
- Pretrain LM to learn how to use these extra information (hopefully)

Popular Retrieval-Augmented LLM: REALM

REALM: Introduce retrieval augmentation in LM pretraining

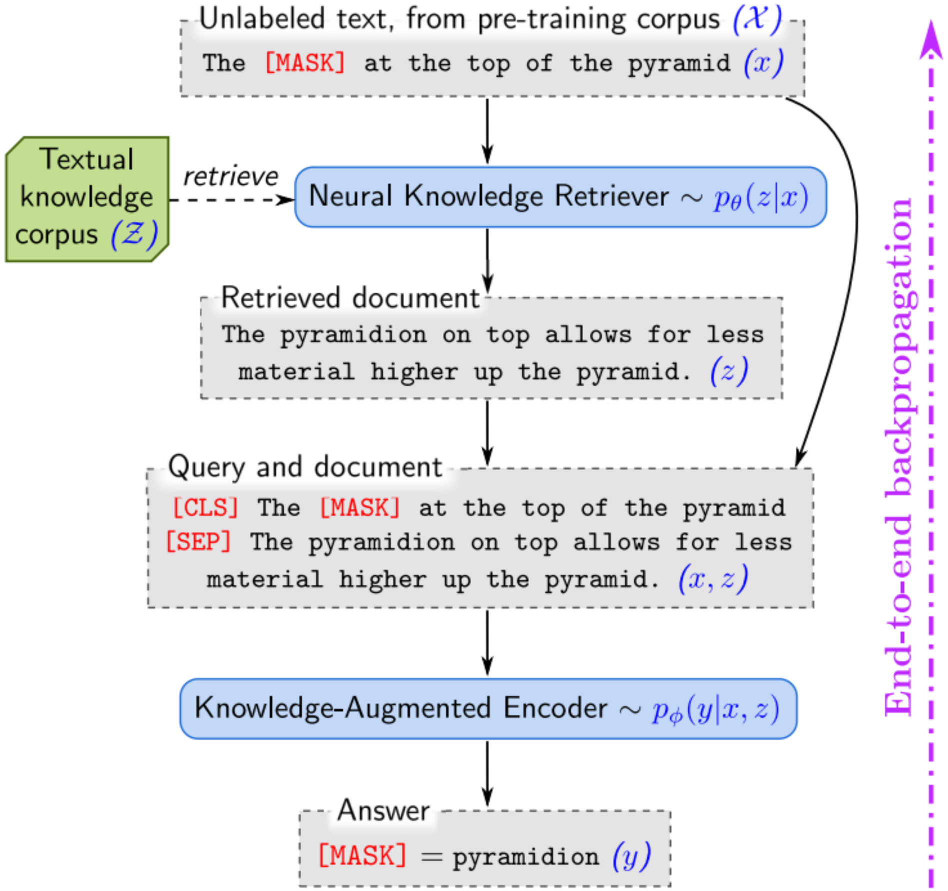


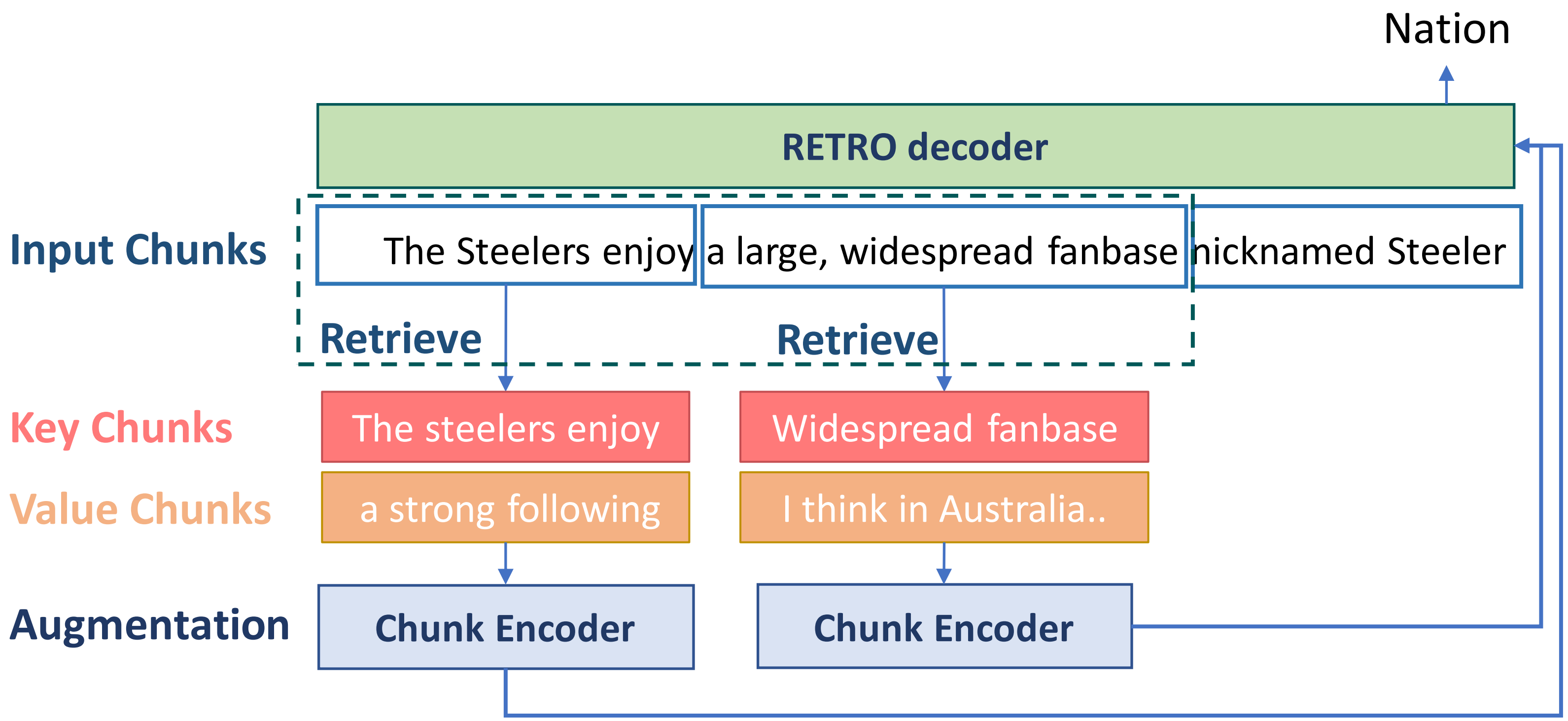
Figure 2: The pipeline of REALM [3]

Recap:

- Retrieval augmentation at the training instance level
- Retrieve related information (x)
- Pretrain end-to-end for LM to learn how to leverage these related information

Popular Retrieval-Augmented LLM: RETRO

RETRO: Pretraining Decoder Language Models by Retrieving from Trillions of Tokens [4].

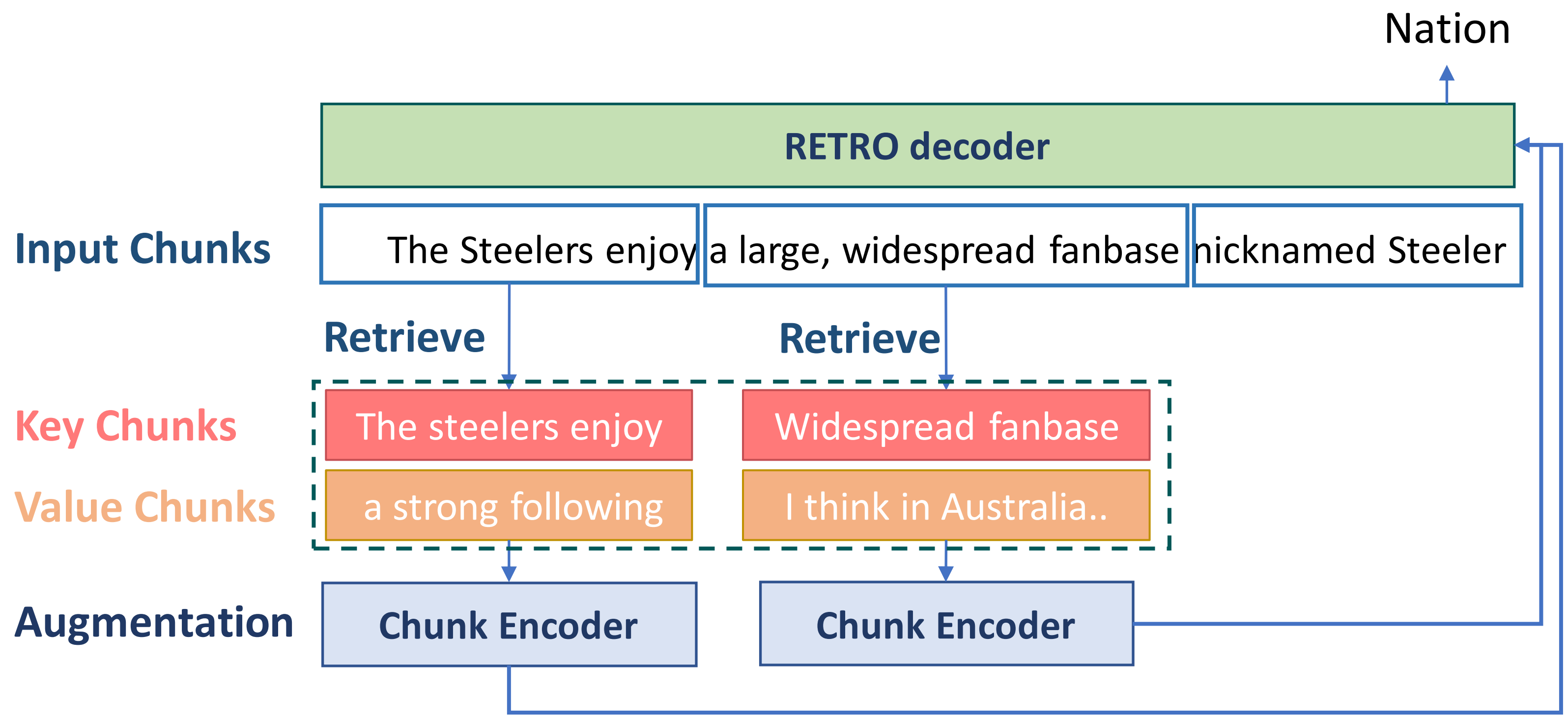


When to retrieve?

- Split a pretraining sequence into chunks (e.g. 64 tokens per chunk)
- Retrieve similar chunks for each chunk

Popular Retrieval-Augmented LLM: RETRO

RETRO: Pretraining Decoder Language Models by Retrieving from Trillions of Tokens [4].



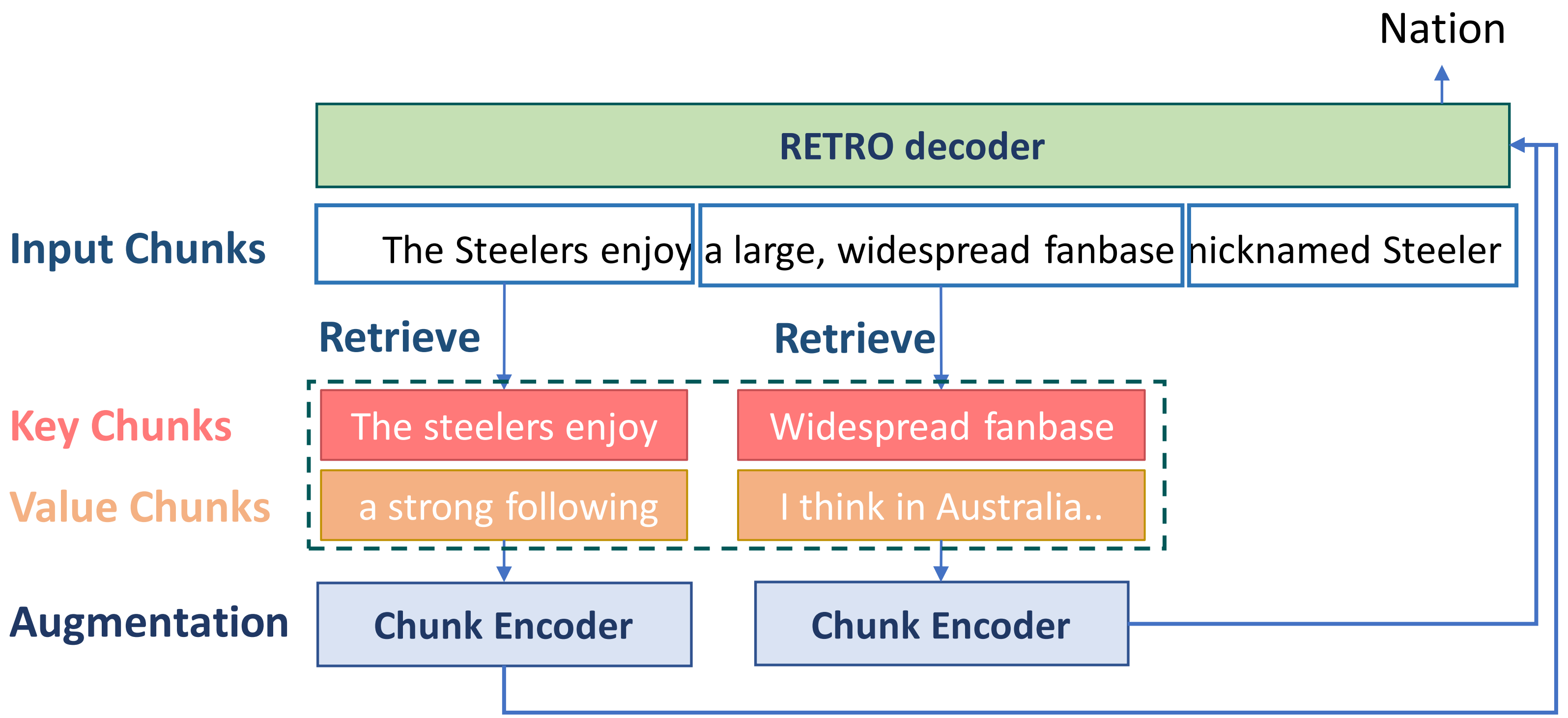
What to retrieve?

- Forming key-value pairs as (this chunk, next chunk) from documents of the corpus
- Retrieve key chunks (x), augment value chunk (y).

[4] Borgeaud et al. Improving Language Models by Retrieving from Trillions of Tokens. ICML 2022

Popular Retrieval-Augmented LLM: RETRO

RETRO: Pretraining Decoder Language Models by Retrieving from Trillions of Tokens [4].

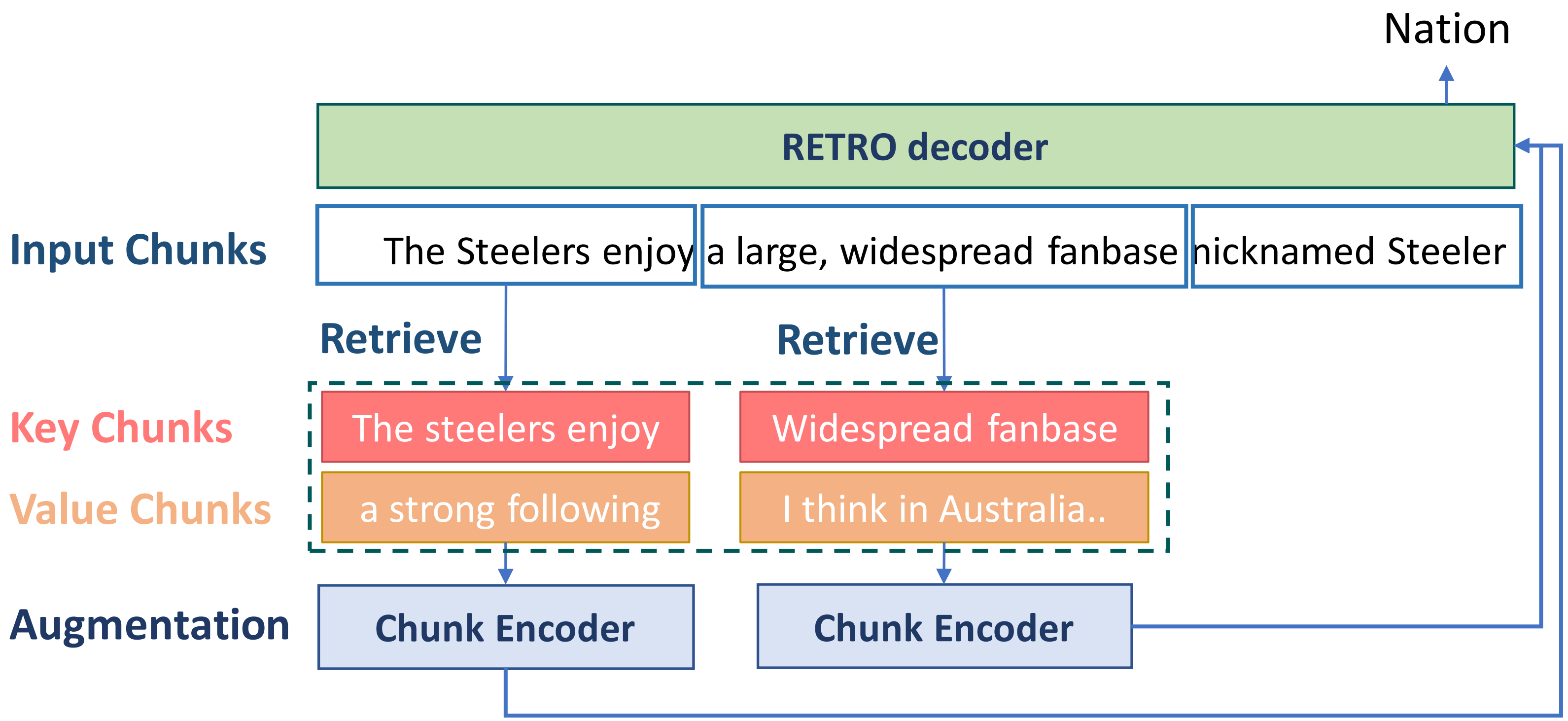


How to retrieve?

- Represent chunks by average BERT embeddings across its token positions
- Retrieval by L2 distance in their representations from ANNS index
- Use SCANN to enable 10 million second latency per querying from 2 trillion tokens (31 Billion Embeddings)

Popular Retrieval-Augmented LLM: RETRO

RETRO: Pretraining Decoder Language Models by Retrieving from Trillions of Tokens [4].

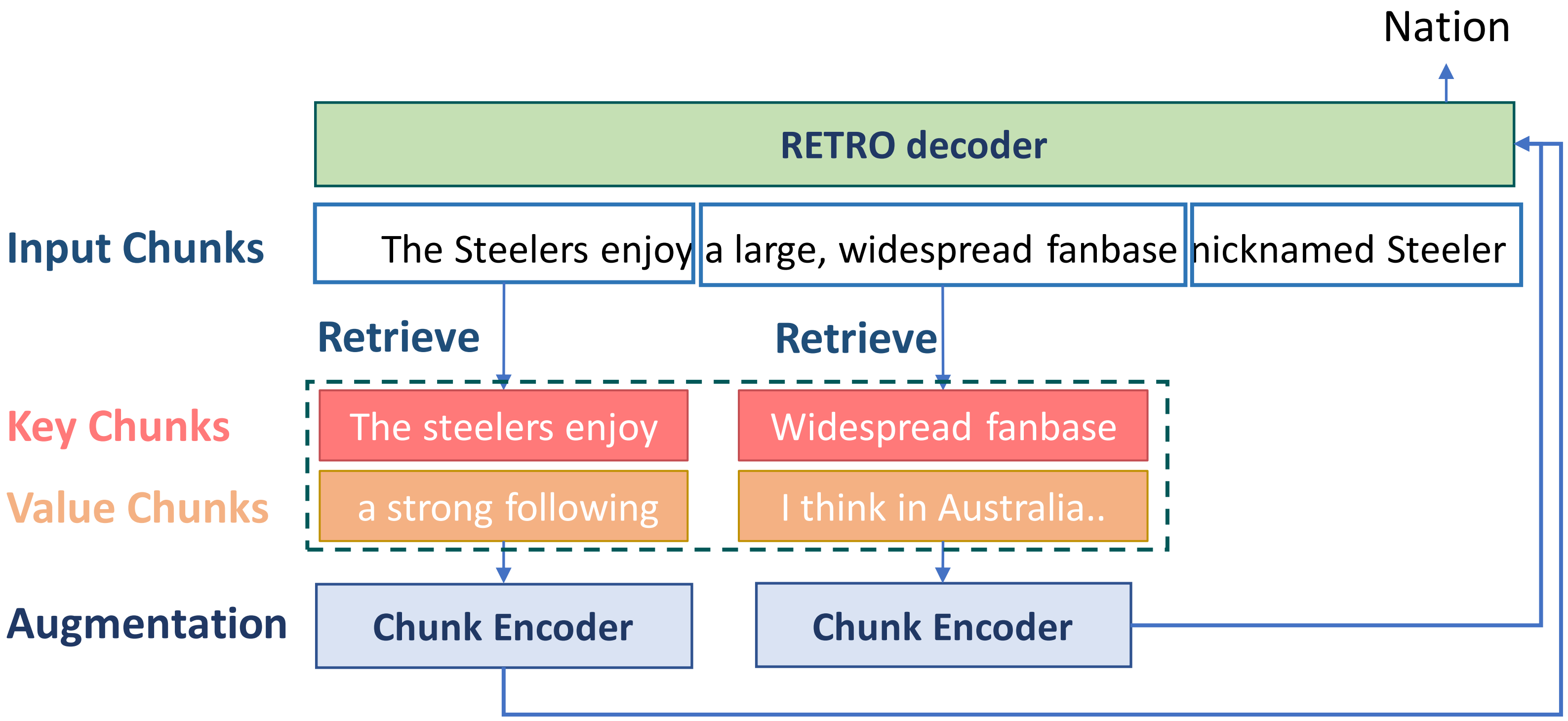


Where to retrieve?

- All chunks from the pretraining corpus, embedded by frozen BERTs
- Again, can switch the retrieval corpus at inference time

Popular Retrieval-Augmented LLM: RETRO

RETRO: Pretraining Decoder Language Models by Retrieving from Trillions of Tokens [4].



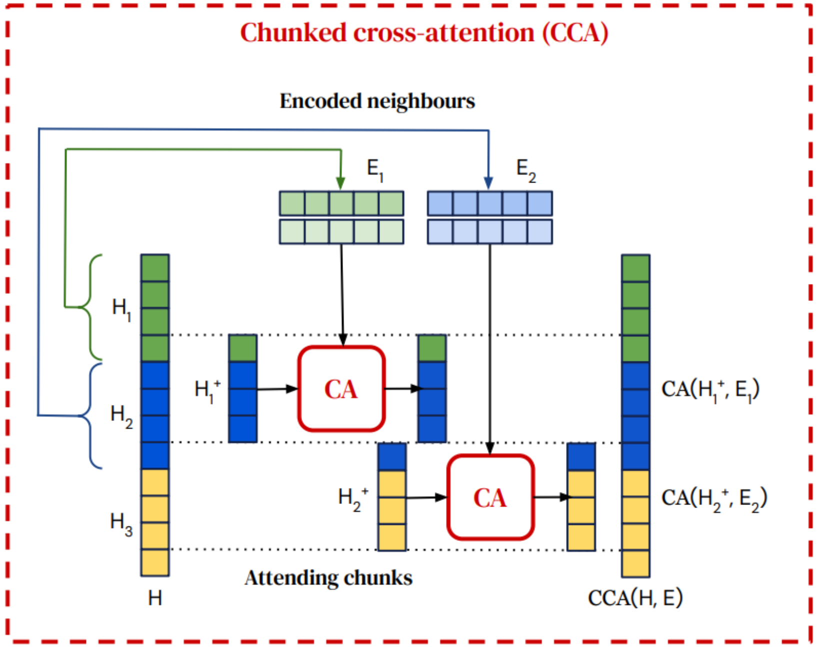


Figure 3: Chunked Cross-Attention [4]

How to leverage retrieved information

- New Chunked Cross-Attention (CCA) blocks: each chunk attends to retrieved chunks of the previous chunk
- Inter leaving CCA blocks and normal decoder attention blocks in each RETRO attention layer
- Pretraining end-to-end at large scale

Popular Retrieval-Augmented LLM: Recap

Model	KNN-LM	REALM	RETRO
Retrieval Frequency	Each Token	Full Sequence	Sub Sequence Chunks
(Key, Value)	(Context, Target Token) (X,Y)	(Text Sequence) (X)	(This Chunk, Next Chunk) (X,Y)
Retrieval Model	Current LM	BERT Embedding	BERT Embedding
Corpus	Pretrain/Plug-In	Pretrain/Plug-In	Pretrain/Plug-In
Utilization	Interpolate KNN probability with LM output at Inference Only	Pretraining as additional inputs of this sequence	Attention to previous chunk's retrieval in pretraining

Table 1: Recap of Popular Retrieval-Augmented LLM Designs

Outline

Retrieval-Augmentation in Pretraining

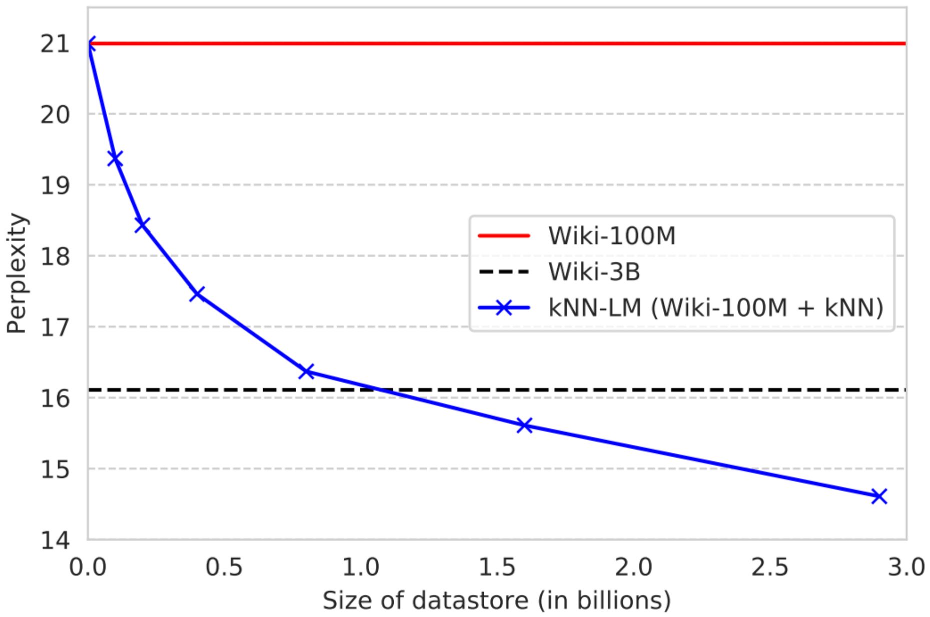
- Overview
- Popular Retrieval Augmented LLMs: KNN-LM, REALM, and RETRO
- **Empirical Results**
- Recap

Retrieval-Augmentation after Pretraining

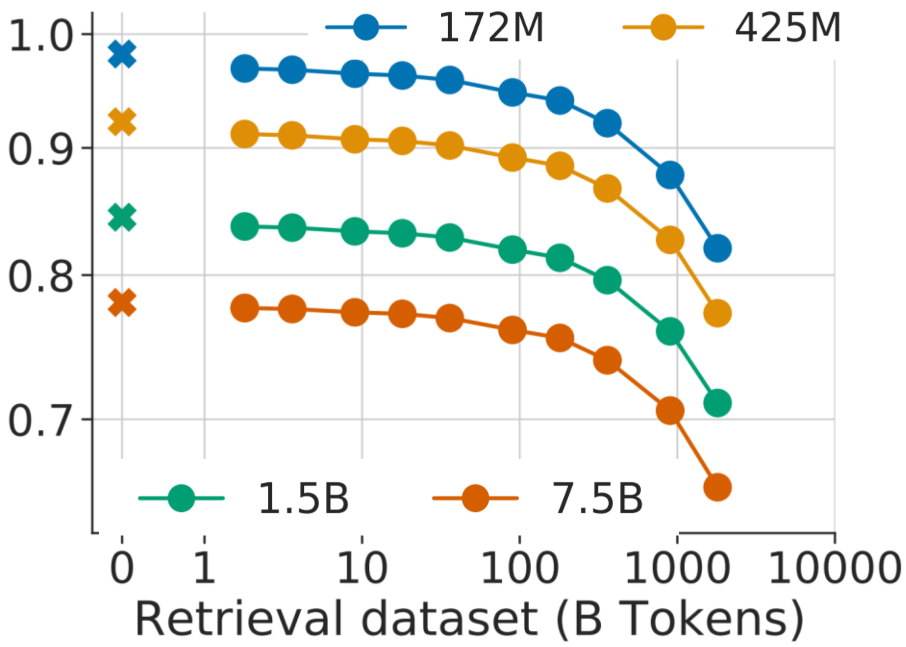
- Overview
- Augmenting Training Data Points
- Augmenting Knowledge/Information
- Adapting Retriever for LLM

Retrieval-Augmented Pretraining: Results

Retrieval-Augmented LMs are great at the language modeling task



(a) KNN-LM Wiki Perplexity

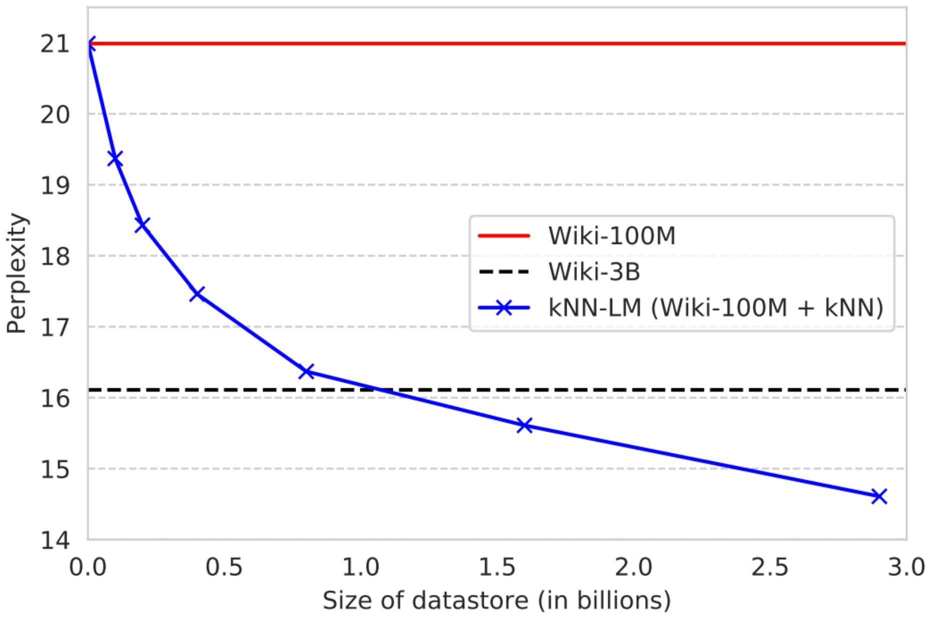


(b) RETRO C4 Bits-per-Byte

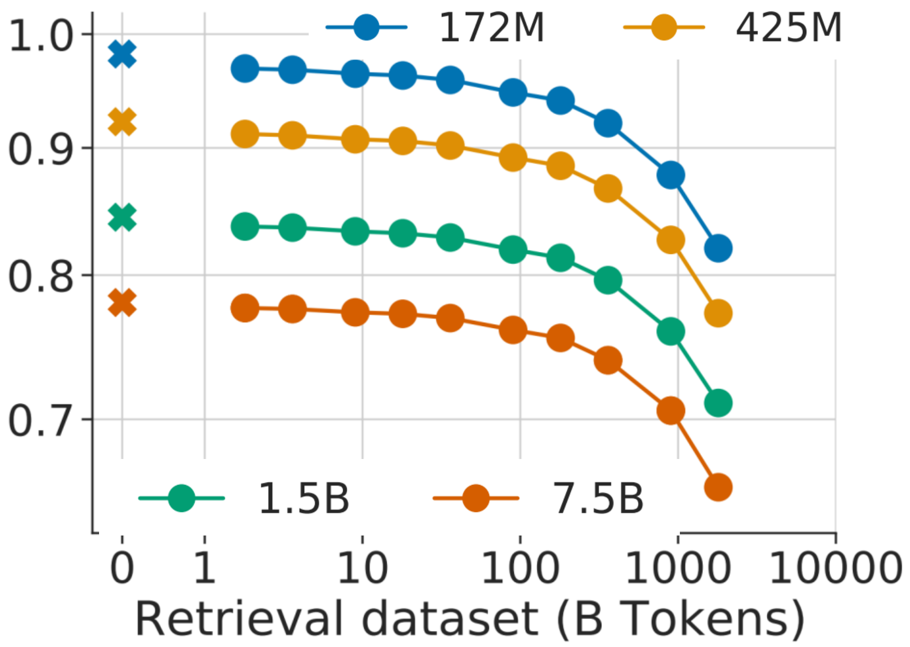
Figure 4: Language model accuracy (perplexity or bits-per-byte) of KNN-LM and RETRO

Retrieval-Augmented Pretraining: Results

Retrieval-Augmented LMs are great at the language modeling task



(a) KNN-LM Wiki Perplexity



(b) RETRO C4 Bits-per-Byte

Figure 4: Language model accuracy (perplexity or bits-per-byte) of KNN-LM and RETRO

- Hugely improved language model accuracy with retrieval-augmentation
- Benefit significantly with bigger retrieval corpus

[2] Khandelwal et al. Generalization through Memorization: Nearest Neighbor Language Models. ICLR 2020
[4] Borgeaud et al. Improving Language Models by Retrieving from Trillions of Tokens. ICML 2022

Retrieval-Augmented Pretraining: Results

REALM claimed effective on knowledge-intensive tasks (QA mostly)

Name	Architectures	Pre-training	NQ (79k/4k)
BERT-Baseline (Lee et al., 2019)	Sparse Retr.+Transformer	BERT	26.5
T5 (base) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	27.0
T5 (large) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	29.8
T5 (11b) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	34.5
DrQA (Chen et al., 2017)	Sparse Retr.+DocReader	N/A	-
HardEM (Min et al., 2019a)	Sparse Retr.+Transformer	BERT	28.1
GraphRetriever (Min et al., 2019b)	GraphRetriever+Transformer	BERT	31.8
PathRetriever (Asai et al., 2019)	PathRetriever+Transformer	MLM	32.6
ORQA (Lee et al., 2019)	Dense Retr.+Transformer	ICT+BERT	33.3
Ours ($\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	39.2
Ours ($\mathcal{X}$ = CC-News, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	**40.4**

Table 2: REALM fine-tuned on Natural Questions [3]

Retrieval-Augmented Pretraining: Results

REALM claimed effective on knowledge-intensive tasks (QA mostly)

- But the source of effectiveness is unclear

Ablation	**Exact Match**
REALM	38.2
REALM retriever+Baseline encoder	37.4
Baseline retriever+REALM encoder	35.3
Baseline (ORQA)	31.3
REALM with random uniform masks	32.3
REALM with random span masks	35.3
$30\times$ stale MIPS	28.7

Not much difference in the pretrained LM

Huge benefits from salient span masking

Table 3: REALM Ablations on Natural Questions [3]

Retrieval-Augmented Pretraining: Results

Downstream performance of RETRO on knowledge-intensive tasks

Model	Test Accuracy
REALM (Guu et al., 2020)	40.4
DPR (Karpukhin et al., 2020)	41.5
RAG (Lewis et al., 2020)	44.5
EMDR^2 (Sachan et al., 2021)	52.5
FiD (Izacard and Grave, 2021)	51.4
FiD + Distill. (Izacard et al., 2020)	**54.7**
Baseline 7B (closed book)	30.4
RETRO 7.5B (DPR retrieval)	45.5

Table 4: RETRO finetuning performance on Natural Questions [3]

Good benefits from retrieval augmentation:

- **Close-Book QA versus Open-Book QA**

[3] Guu et al. Retrieval-Augmented Language Model Pre-Training. ICML 2020

Retrieval-Augmented Pretraining: Results

Downstream performance of RETRO on knowledge-intensive tasks

Model	Test Accuracy
REALM (Guu et al., 2020)	40.4
DPR (Karpukhin et al., 2020)	41.5
RAG (Lewis et al., 2020)	44.5
EMDR^2 (Sachan et al., 2021)	52.5
FID (Izacard and Grave, 2021)	51.4
FID + Distill. (Izacard et al., 2020)	**54.7**
Baseline 7B (closed book)	30.4
RETRO 7.5B (DPR retrieval)	45.5

Table 4: RETRO finetuning performance on Natural Questions [3]

Not much difference between:

- **Retrieval-augmentation in pretraining**
- **Retrieval-augmentation in downstream usage**

Retrieval-Augmented Pretraining: Results

Downstream performance of RETRO on knowledge-nonintensive tasks

Tasks	Small		Medium		XL		XXL	
	GPT	RETRO	GPT	RETRO	GPT	RETRO	GPT	RETRO
Knowledge-nonintensive Tasks								
Lambada	41.7	41.4 ↓0.3	54.1	55.0 ↑0.9	63.9	64.0 ↑0.1	73.9	72.7 ↓1.2
RACE	34.6	32.5 ↓2.1	37.3	37.3 ↑0.0	40.8	39.9 ↓0.9	44.3	43.2 ↓1.1
PiQA	64.3	64.8 ↑0.5	70.2	68.7 ↓1.5	73.7	74.1 ↑0.4	78.5	77.4 ↓1.1
WinoGrande	52.4	52.0 ↓0.4	53.8	55.2 ↑1.4	59.0	60.1 ↑1.1	68.5	65.8 ↓2.7
ANLI-R2	35.1	36.2 ↑1.1	33.5	33.3 ↓0.2	34.3	35.3 ↑1.0	32.2	35.5 ↑3.3
HANS	51.5	51.4 ↓0.1	50.5	50.5 ↑0.0	50.1	50.0 ↓0.1	50.8	56.5 ↑5.7
WiC	50.0	50.0 ↑0.0	50.2	50.0 ↓0.2	47.8	49.8 ↑2.0	52.4	52.4 ↑0.0

Table 4: Nvidia Megatron-RETRO zero-shot performances versus GPT at variant scales [5]

[5] Wang et al. Shall We Pretrain Autoregressive Language Models with Retrieval? A Comprehensive Study. EMNLP 2023

Retrieval-Augmented Pretraining: Results

Downstream performance of RETRO on knowledge-nonintensive tasks

Tasks	Small		Medium		XL		XXL	
	GPT	RETRO	GPT	RETRO	GPT	RETRO	GPT	RETRO
Knowledge-nonintensive Tasks								
Lambada	41.7	41.4 ↓0.3	54.1	55.0 ↑0.9	63.9	64.0 ↑0.1	73.9	72.7 ↓1.2
RACE	34.6	32.5 ↓2.1	37.3	37.3 ↑0.0	40.8	39.9 ↓0.9	44.3	43.2 ↓1.1
PiQA	64.3	64.8 ↑0.5	70.2	68.7 ↓1.5	73.7	74.1 ↑0.4	78.5	77.4 ↓1.1
WinoGrande	52.4	52.0 ↓0.4	53.8	55.2 ↑1.4	59.0	60.1 ↑1.1	68.5	65.8 ↓2.7
ANLI-R2	35.1	36.2 ↑1.1	33.5	33.3 ↓0.2	34.3	35.3 ↑1.0	32.2	35.5 ↑3.3
HANS	51.5	51.4 ↓0.1	50.5	50.5 ↑0.0	50.1	50.0 ↓0.1	50.8	56.5 ↑5.7
WiC	50.0	50.0 ↑0.0	50.2	50.0 ↓0.2	47.8	49.8 ↑2.0	52.4	52.4 ↑0.0

Table 4: Nvidia Megatron-RETRO zero-shot performances versus GPT at variant scales [4]

Hard to say which one works better

- Which is a loss given the complication of retrieval augmentation in pretraining

Retrieval-Augmented Pretraining: Recap

Model	**KNN-LM**	**REALM**	**RETRO**
Retrieval Frequency	Each Token	Full Sequence	Sub Sequence Chunks
(Key, Value)	(Context, Target Token) (X,Y)	(Text Sequence) (X)	(This Chunk, Next Chunk) (X,Y)
Retrieval Model	Current LM	BERT Embedding	BERT Embedding
Corpus	Pretrain/Plug-In	Pretrain/Plug-In	Pretrain/Plug-In
Utilization	Interpolate KNN probability with LM output at Inference Only	Pretraining as additional inputs of this sequence	Attention to previous chunk’s retrieval in pretraining

Table 1: Recap of Popular Retrieval-Augmented LLM Designs

Retrieval-augmented pretraining improves language model accuracy significantly

- Plus ability to plug-in new corpus for better transfer

Generalization ability to downstream tasks more in question

- Retrieval-augmentation help knowledge-intensive tasks, but not necessarily needed at pretraining phrase
- Ambivalent performances on knowledge-nonintensive tasks

Retrieval-Augmented Pretraining: Recap

Why pretraining with retrieval augmentation not helping LLM generalization?

- At least multi million $$$ question, e.g., the pretraining cost of RETRO and Megatron-RETRO

Some guesses:

- We do not have a good mechanism to control the learning of understanding and memorization in pretraining
- We are not clear when and where LLMs need external information in pretraining
  - Some parametric memory is necessary, but not all of them?
- Retrieval is not as effective in retrieval-augmented pretraining
  - Query is coarse
  - Retrieval system is not designed for retrieval-augmented pretraining
  - LLMs are good at ignoring noisy signals and focusing on easy context relations

Afterall, we only know 1.5 effective ways (Autoregressive LM + 0.5 denoising LM) to pretrain strong LLMs, perhaps learning to utilize retrieved information is not the skillset of these 1.5 ways.

Outline

Retrieval-Augmentation in Pretraining

- Overview
- Popular Retrieval Augmented LLMs: KNN-LM, REALM, and RETRO
- Empirical Results
- Recap

Retrieval-Augmentation after Pretraining

- Overview
- Augmenting Training Data Points
- Augmenting Knowledge/Information
- Adapting Retriever for LLM

Retrieval-Augmentation in Downstream Tasks

Two types of retrieval augmentation

Retrieving Similar Supervision Data Points

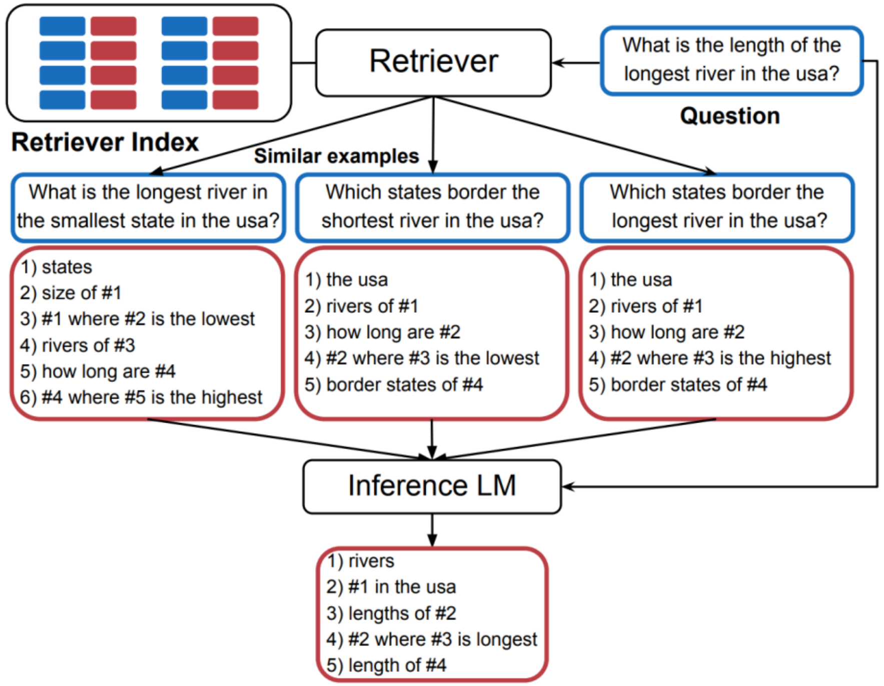


Figure 5: Retrieve Similar Prompts/In-context Examples [6]

Retrieving Additional Information

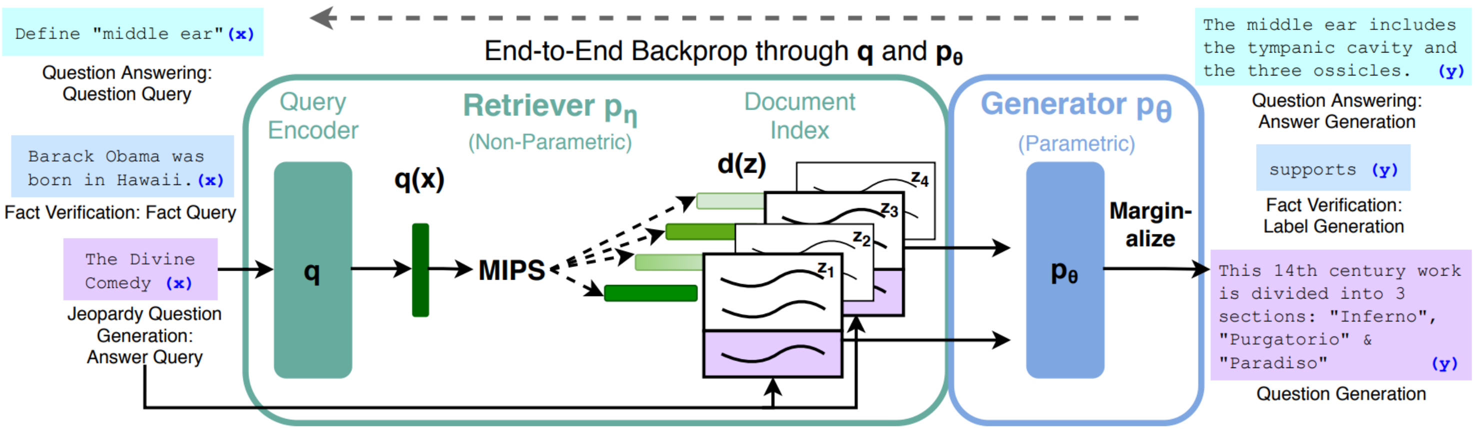


Figure 6: Retrieve additional information for generation [7]

[6] Rubin et al. Learning to Retrieve Prompts for In-Context Learning. NAACL-HLT 2022
[7] Lewis et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020

Augmenting with Retrieved In-Context Examples

Retrieval similar training data points for the current downstream task

Retrieving Similar Supervision Data Points

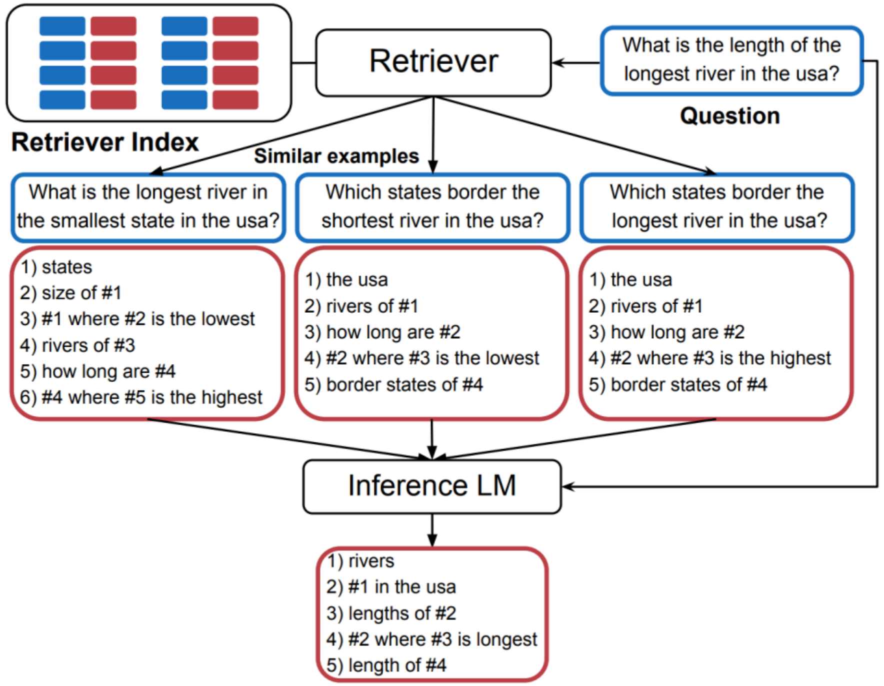


Figure 5: Retrieve Similar Prompts/In-context Examples [6]

When to retrieve?
- Per downstream data point

What to retrieve?
- Similar training data points (x, y)

How to retrieve?
- Dense retriever adapted for LLM

Where to retrieve?
- Training data

How to leverage retrieved information?
- Add in as in-context examlpes

Augmenting with Retrieved In-Context Examples

Retrieval similar training data points for the current downstream task

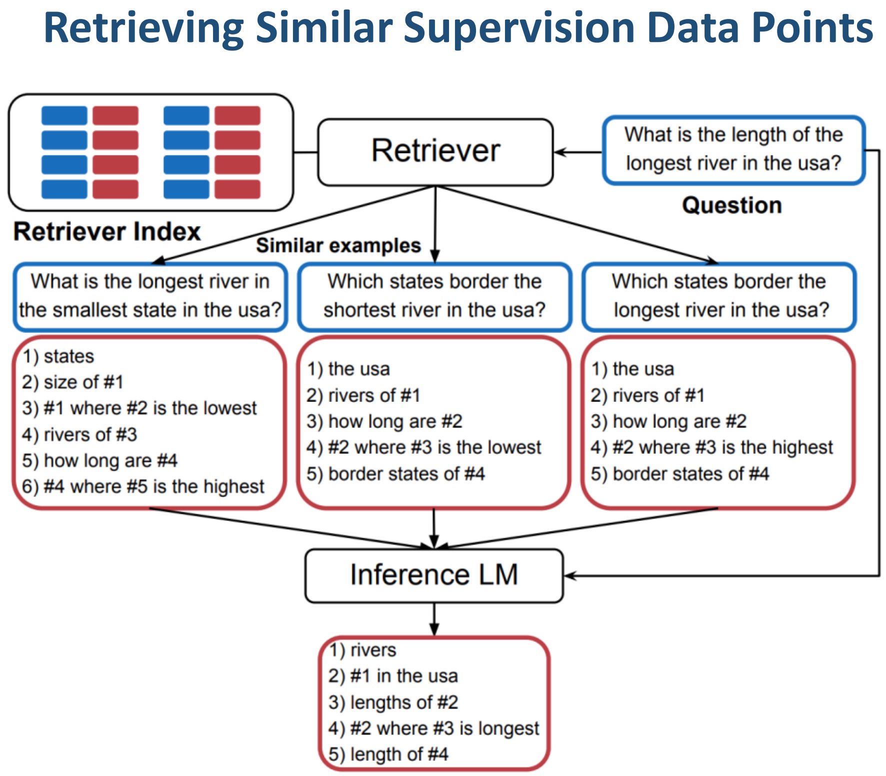


Figure 5: Retrieve Similar Prompts/In-context Examples [6]

When does this work?

- A generally better way to fine more similar in-context examples than random sample

Why does this work?

- A form of test time learning

Augmenting with Retrieved In-Context Examples

Retrieval similar training data points for the current downstream task

Retrieving Similar Supervision Data Points

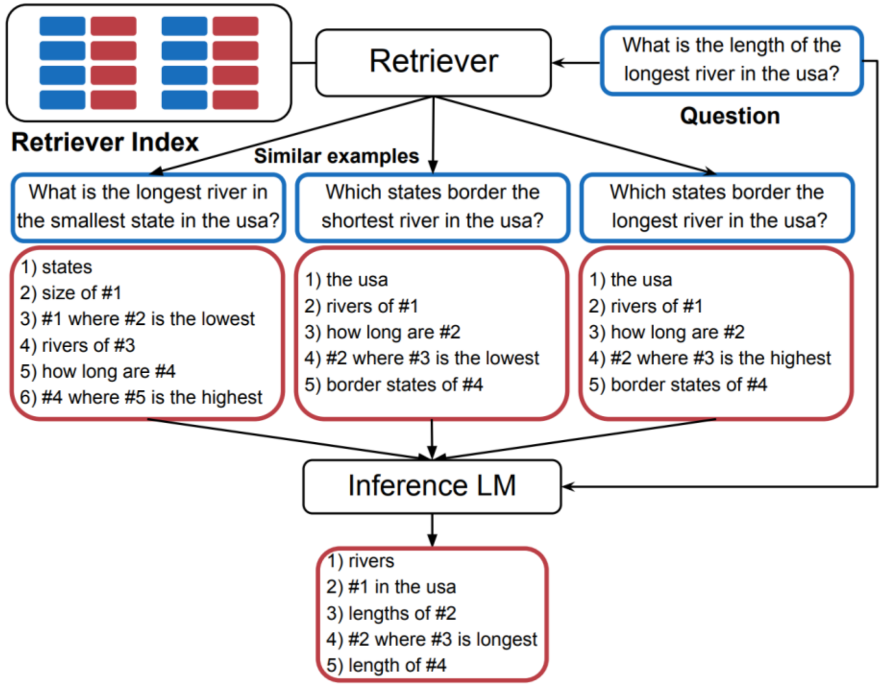


Figure 5: Retrieve Similar Prompts/In-context Examples [6]

When does this work?

- A generally better way to fine more similar in-context examples than random sample

Why does this work?

- A form of test time learning

Test Time Learning

- Find similar training data points for the current testing data point
- Focused learning (e.g., a few gradient steps) on similar training data points
- "Upweighting" training data close to the current testing data
- A classic idea

"In-context learning == SGD view"

- Performing virtual SGD on random versus retrieved similar in-context examples

[6] Rubin et al. Learning to Retrieve Prompts for In-Context Learning. NAACL-HLT 2022

Retrieval-Augmentation in Downstream Tasks

Two types of retrieval augmentation

Retrieving Additional Information

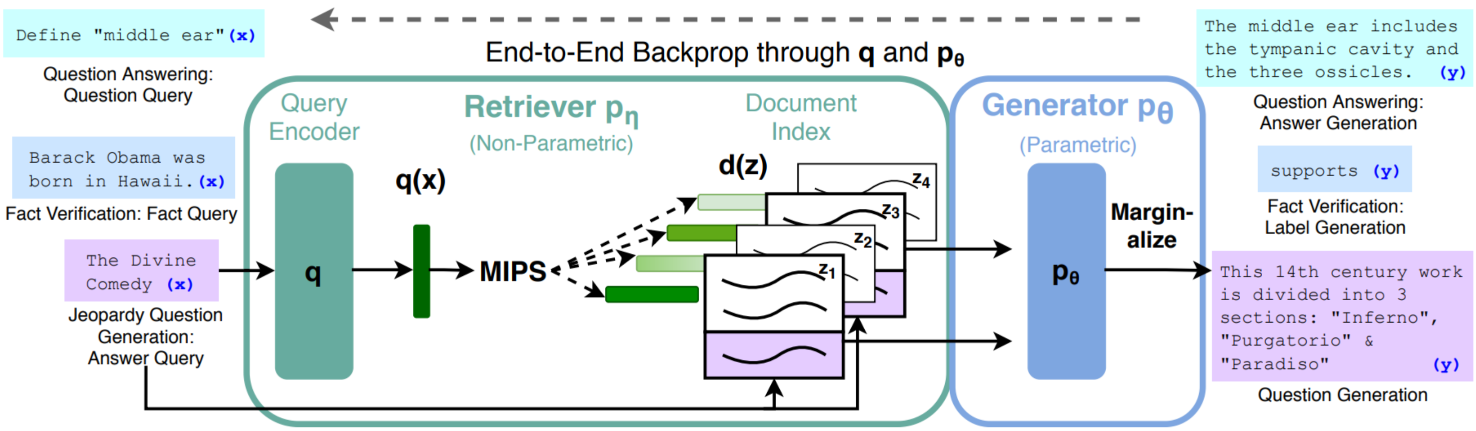


Figure 6: Retrieve additional information for generation [7]

When to retrieve?

- Per downstream data point

What to retrieve?

- Relevant documents (x)

How to retrieve?

- Dense retriever, e.g., from web search

Where to retrieve?

- Target corpus with needed information

How to leverage retrieved information?

- Used to be complex:
  - Latent space models
  - Fusion-in-Decoder
- With Decoder LLM:
  - As additional inputs (with prompts)
  - Zero-shot or finetuned

Retrieval-Augmentation in Downstream Tasks

Two types of retrieval augmentation

Retrieving Additional Information

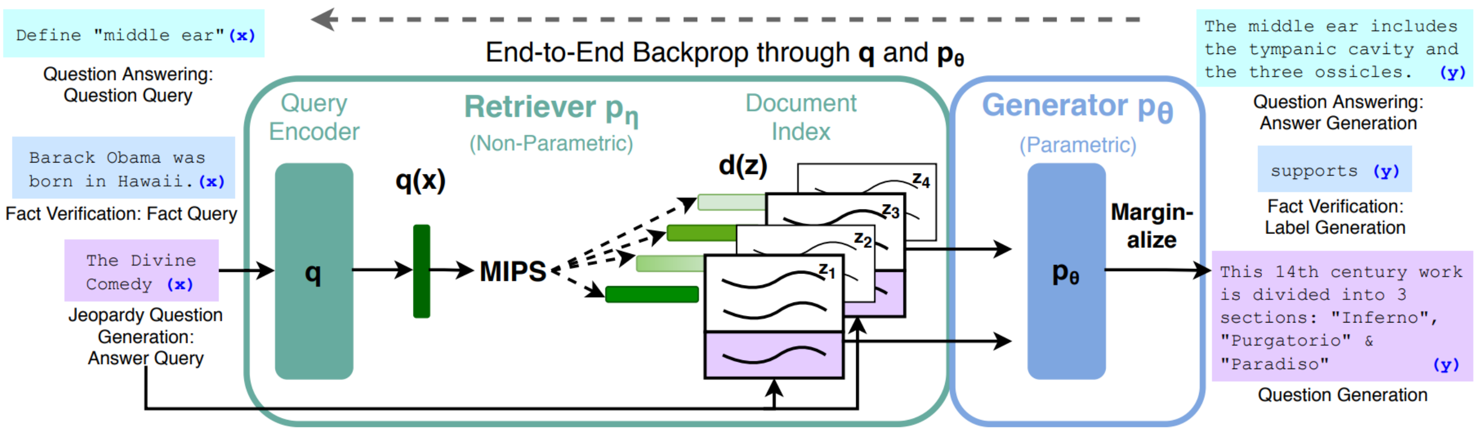


Figure 6: Retrieve additional information for generation [7]

When does this work?

- Tasks required additional information
- E.g., Knowledge-intensive tasks, reducing hallucination, plug-in in-domain information, etc.

Why does it work?

- Additional information from retrieval
- An LLM version of OpenQA

Retrieval-Augmentation in Downstream Tasks

Two types of retrieval augmentation

Retrieving Additional Information

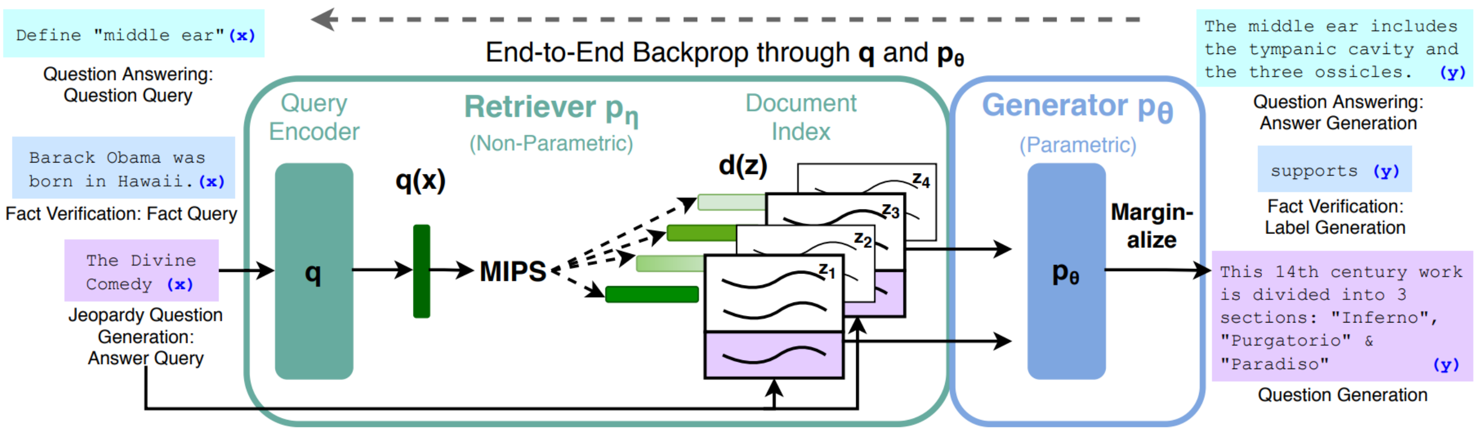


Figure 6: Retrieve additional information for generation [7]

When does this work?

- Tasks required additional information
- E.g., Knowledge-intensive tasks, reducing hallucination, plug-in in-domain information, etc.

Why does it work?

- Additional information from retrieval
- An LLM version of OpenQA

When does it **not** work?

- Tasks do not require extra information
- E.g., hard to think of what to extra information is needed for sentiment analysis or grammar check

Outline

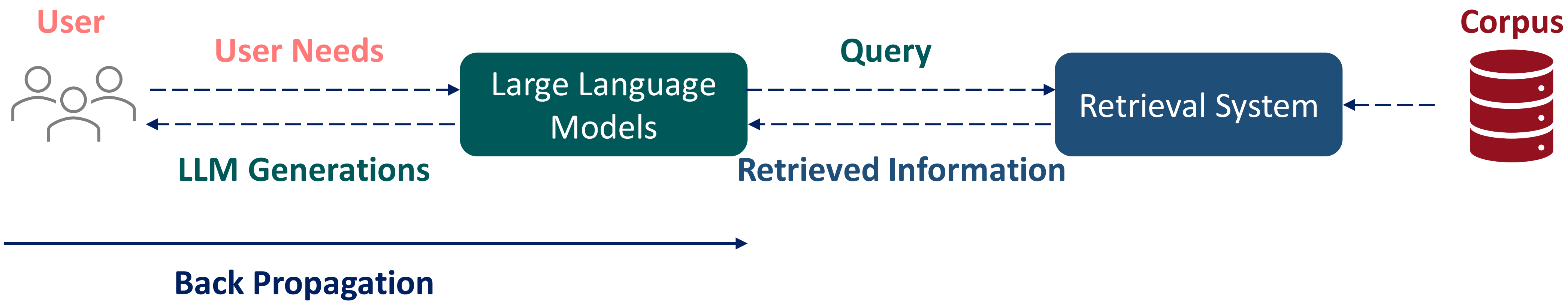
Retrieval-Augmentation in Pretraining

- Overview
- Popular Retrieval Augmented LLMs: KNN-LM, REALM, and RETRO
- Empirical Results
- Recap

Retrieval-Augmentation after Pretraining

- Overview
- Augmenting Training Data Points
- Augmenting Knowledge/Information
- **Adapting Retriever for LLM**

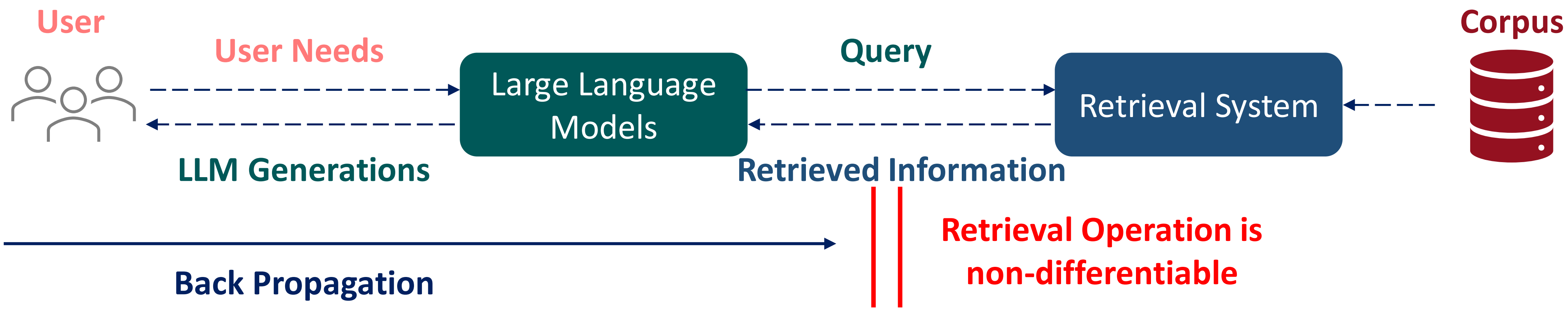
Finetuning Retrieval-Augmented LM



Finetuning language model parameters is standard

- Back propagate from user preferences/supervision labels

Finetuning Retrieval-Augmented LM

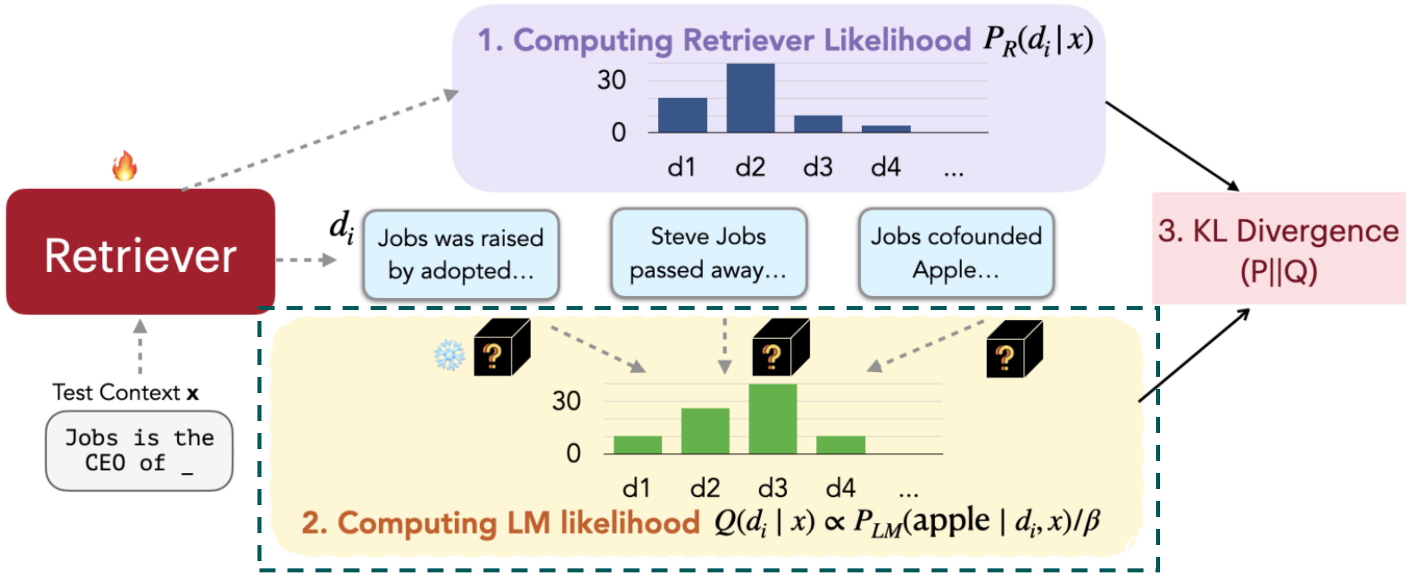


Finetuning language model parameters is standard
- Back propagate from user preferences/supervision labels

Not as trivial to fine-tune retriever end-to-end
- Retrieval is a Top-K operation which is not differentiable
- Also many of current LLMs are black-box APIs

Finetuning Retriever to Augment LM: REPLUG

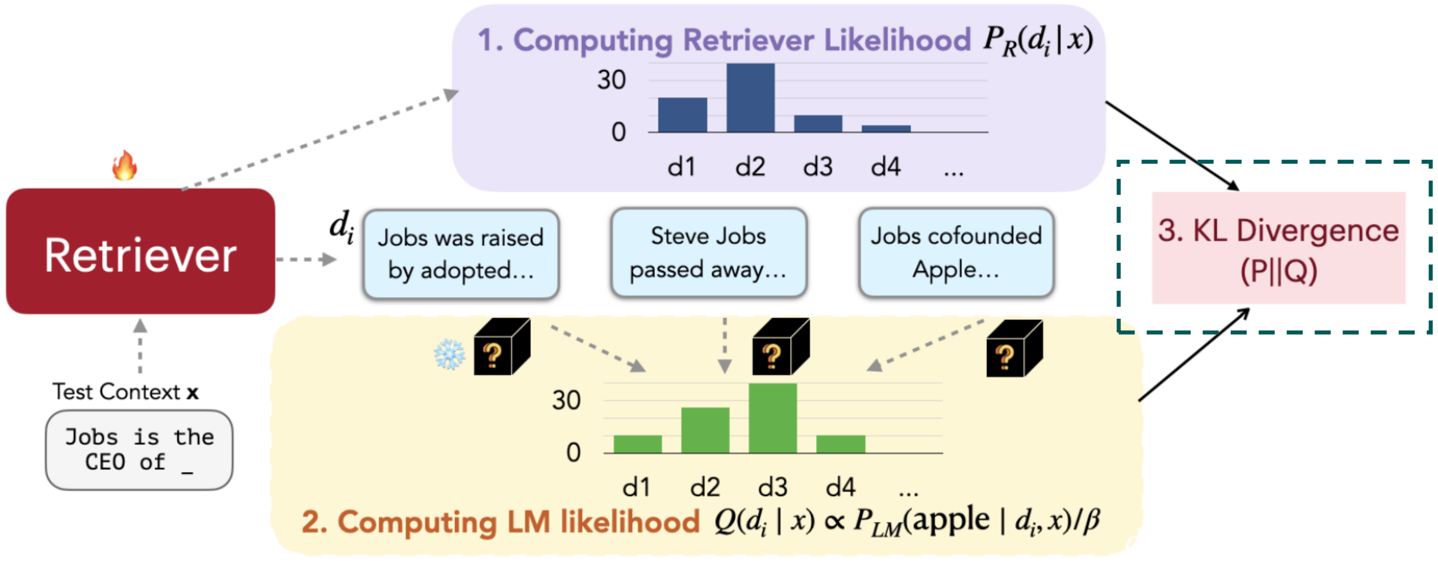
Option 1: Using influences on the outputs of LLM



1. Plug in each retrieved document to LLM and get its output probability
   - $p_{LM}(y|d,x)$: probability of generating ground truth y when x is augmented with retrieved document d
2. Get the likelihood of document d being useful for LLM: $Q(d|x,y) = \text{softmax}_d p_{LM}(y|d,x)$

Finetuning Retriever to Augment LM: REPLUG

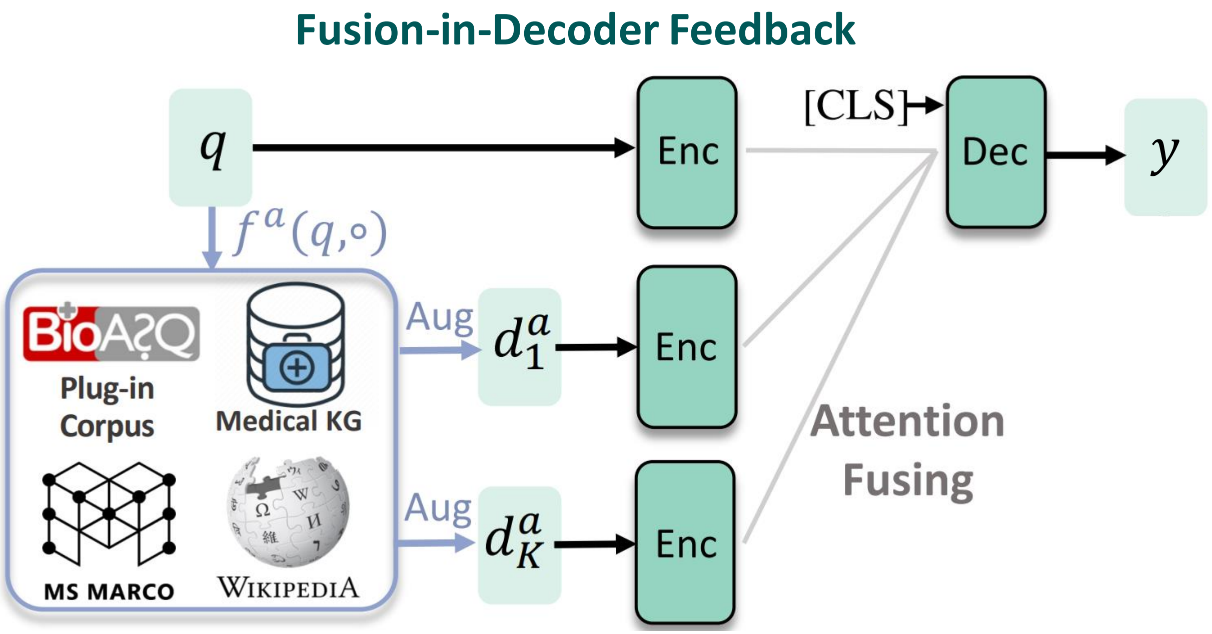
Option 1: Using influences on the outputs of LLM



1. Plug in each retrieved document to LLM and get its output probability
   - $p_{LM}(y|d,x)$: probability of generating ground truth y when x is augmented with retrieved document d
2. Get the likelihood of document d being useful for LLM: $Q(d|x,y) = \text{softmax}_d p_{LM}(y|d,x)$
3. Train retrievers to match retrieval scores of d with $Q(d|x,y)$

Finetuning Retriever to Augment LM: REPLUG

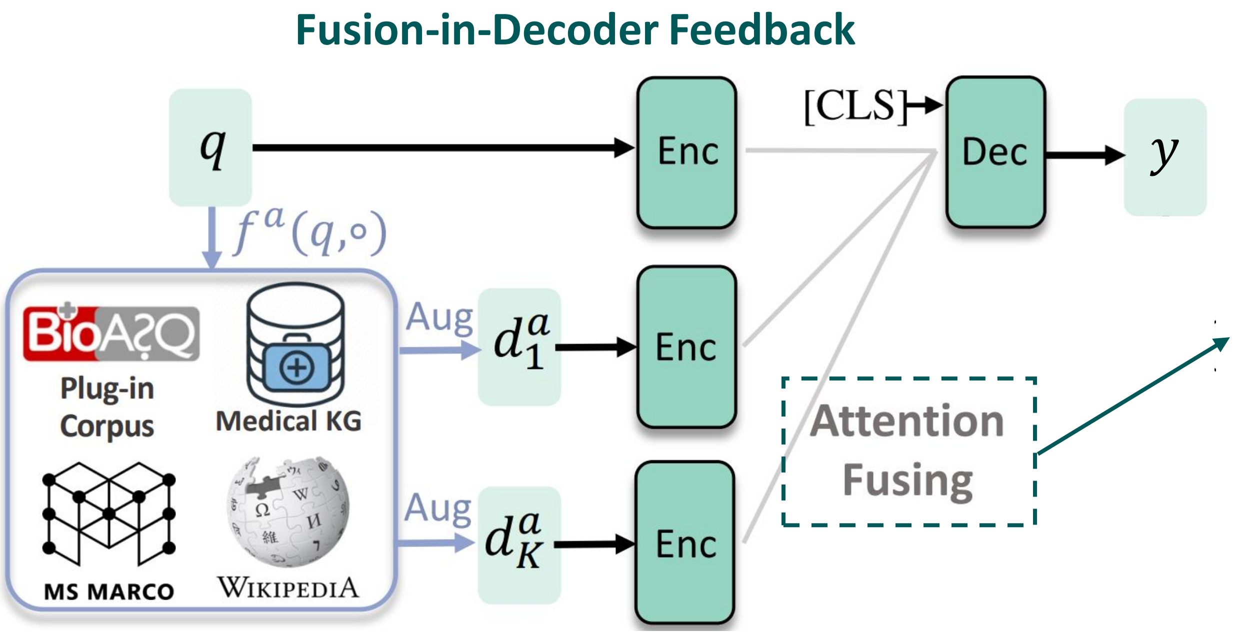
Option 2: Using fine-grained feedback signals from an open LLM to train the augmentation retriever, apply the Augmentation-Adapted Retrieval (AAR) with open or black-box LLMs



Finetuning Retriever to Augment LM: AAR

Option 2: Using fine-grained feedback signals from an open LLM to train the augmentation retriever

- Apply the Augmentation-Adapted Retrieval (AAR) with open or black-box LLMs



Use attention weights from decoder to document's encoder as LLM preferences [9]

- Sum up all attention scores from y decoding to d encoder: all heads, all layers, all d tokens
- The cumulated attention is a very effective feedback signal from the LLM in Open QA
- Better than crowd-source labels sometimes (!)

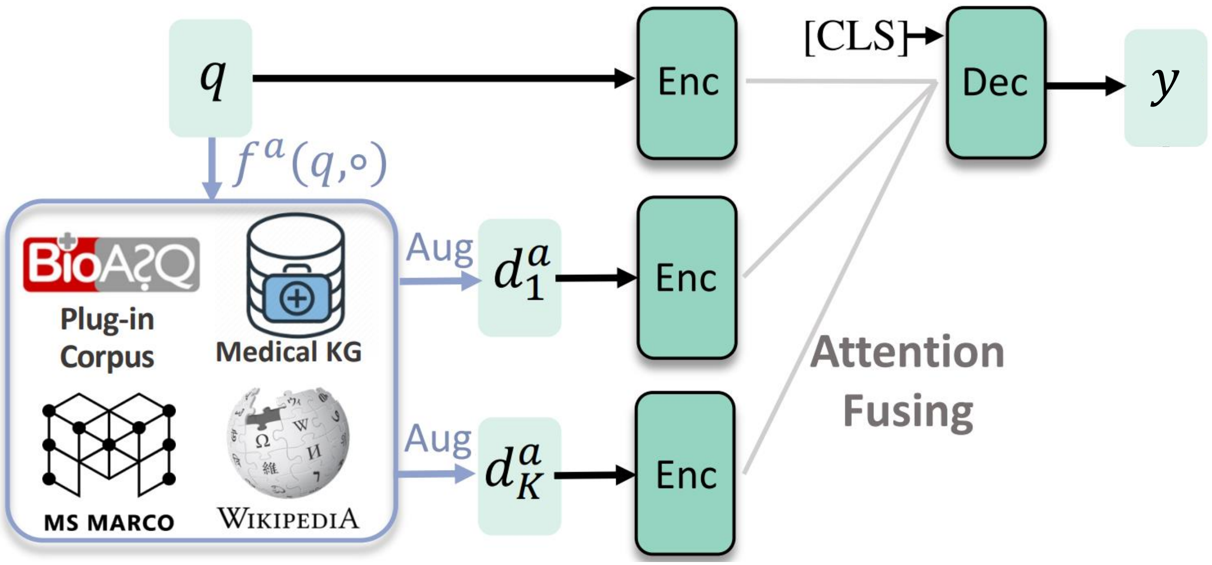
Why? Attention interpretation lecture

Finetuning Retriever to Augment LM: AAR

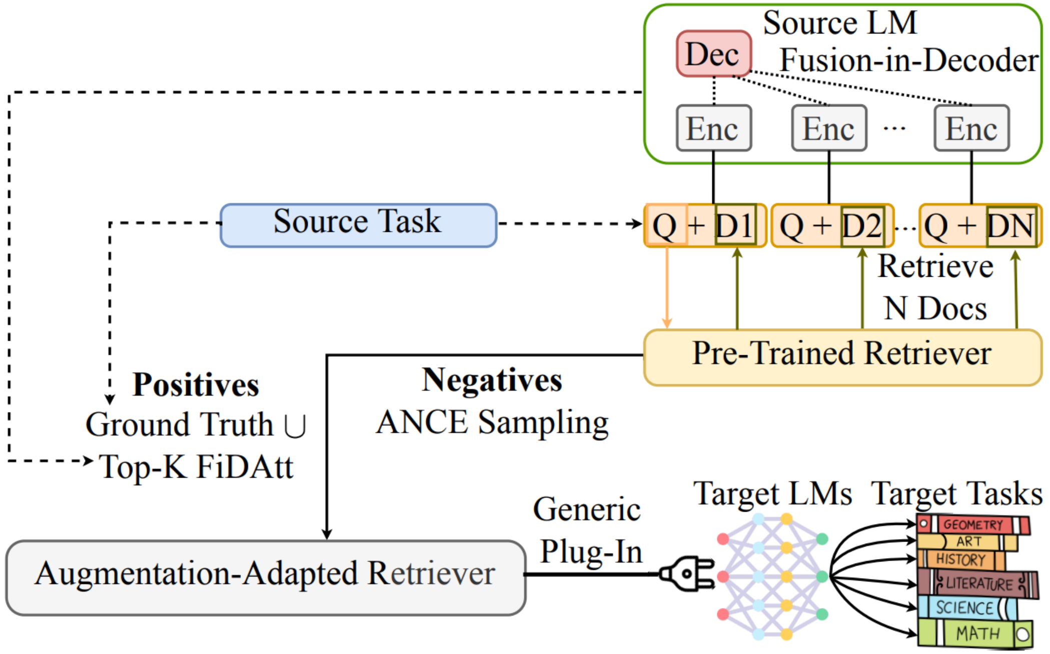
Option 2: Using fine-grained feedback signals from an open LLM to train the augmentation retriever

- Apply the Augmentation-Adapted Retrieval (AAR) with open or black-box LLMs

Fusion-in-Decoder Feedback



Plug-in Adapted Retriever to Black-box LLMs [10]



Finetuning Retriever to Augment LM: AAR

Settings	Methods	# Parameters	MMLU All	MMLU Hum.	MMLU Soc. Sci.	MMLU STEM	MMLU Other	PopQA All
Few-shot	Chinchilla (Hoffmann et al., 2022)	70B	67.5	63.6	79.3	55.0	73.9	n.a.
	OPT-IML-Max (Iyer et al., 2022)	175B	47.1	n.a.	n.a.	n.a.	n.a.	n.a.
	InstructGPT (Ouyang et al., 2022)	175B	60.5	62.0	71.8	44.3	70.1	35.2
Zero-shot	GAL (Taylor et al., 2022)	120B	52.6	n.a.	n.a.	n.a.	n.a.	n.a.
	OPT-IML-Max	175B	49.1	n.a.	n.a.	n.a.	n.a.	n.a.
	InstructGPT	175B	60.2	**65.7**	68.0	46.1	66.5	34.7
	InstructGPT w/ AR	175B	60.5	62.2	71.3	44.7	69.7	43.3
	InstructGPT w/ $\text{AAR}_{\text{Contriever}}$ (Ours)	175B	61.5	64.5	**73.1**	45.0	69.9	43.9
	InstructGPT w/ AAR_{ANCE} (Ours)	175B	**62.2**	62.0	72.0	**49.2**	**70.7**	**52.0**

Table 5: Performance of augmenting black-box GPT with retriever adapted using feedback from FLAN-T5 base [10]

[10] Yu et al. Augmentation-Adapted Retriever Improves Generalization of Language Models as Generic Plug-In. ACL 2023

Finetuning Retriever to Augment LM: AAR

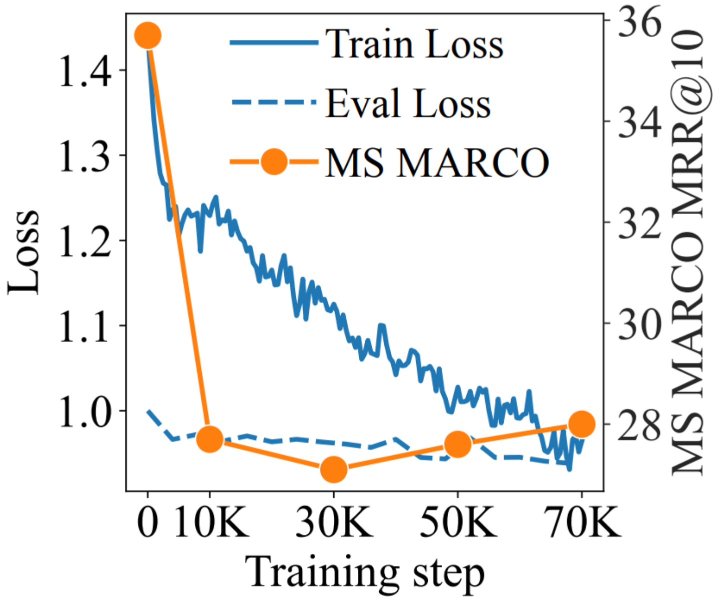
Settings	Methods	# Parameters	MMLU All	MMLU Hum.	MMLU Soc. Sci.	MMLU STEM	MMLU Other	PopQA All
Few-shot	Chinchilla (Hoffmann et al., 2022)	70B	67.5	63.6	79.3	55.0	73.9	n.a.
	OPT-IML-Max (Iyer et al., 2022)	175B	47.1	n.a.	n.a.	n.a.	n.a.	n.a.
	InstructGPT (Ouyang et al., 2022)	175B	60.5	62.0	71.8	44.3	70.1	35.2
Zero-shot	GAL (Taylor et al., 2022)	120B	52.6	n.a.	n.a.	n.a.	n.a.	n.a.
	OPT-IML-Max	175B	49.1	n.a.	n.a.	n.a.	n.a.	n.a.
	InstructGPT	175B	60.2	**65.7**	68.0	46.1	66.5	34.7
	InstructGPT w/ AR	175B	60.5	62.2	71.3	44.7	69.7	43.3
	InstructGPT w/ $AAR_{Contriever}$ (Ours)	175B	61.5	64.5	**73.1**	45.0	69.9	43.9
	InstructGPT w/ AAR_{ANCE} (Ours)	175B	**62.2**	62.0	72.0	**49.2**	**70.7**	**52.0**

Table 5: Performance of augmenting black-box GPT with retriever adapted using feedback from FLAN-T5 base [10]

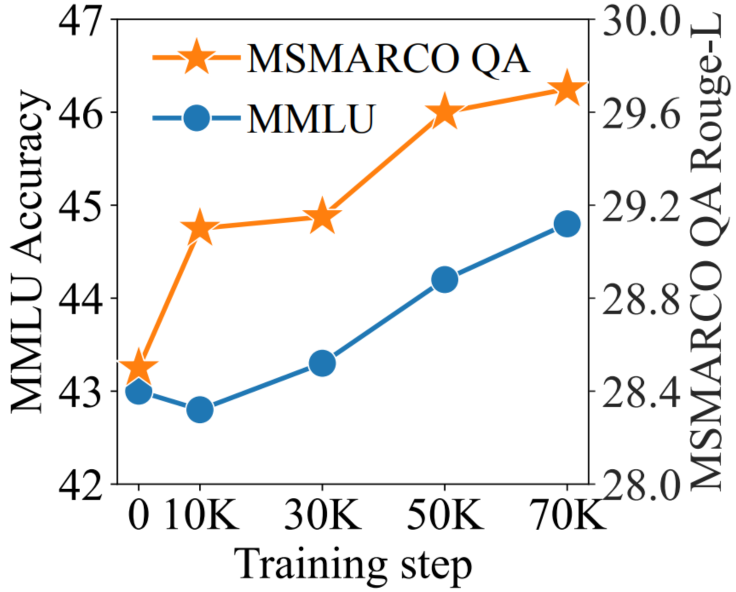
Significant improvements compared to augmenting with retrieval systems trained for web search (AAR versus AR)

[10] Yu et al. Augmentation-Adapted Retriever Improves Generalization of Language Models as Generic Plug-In. ACL 2023

Finetuning Retriever to Augment LM: Why AAR Helps?



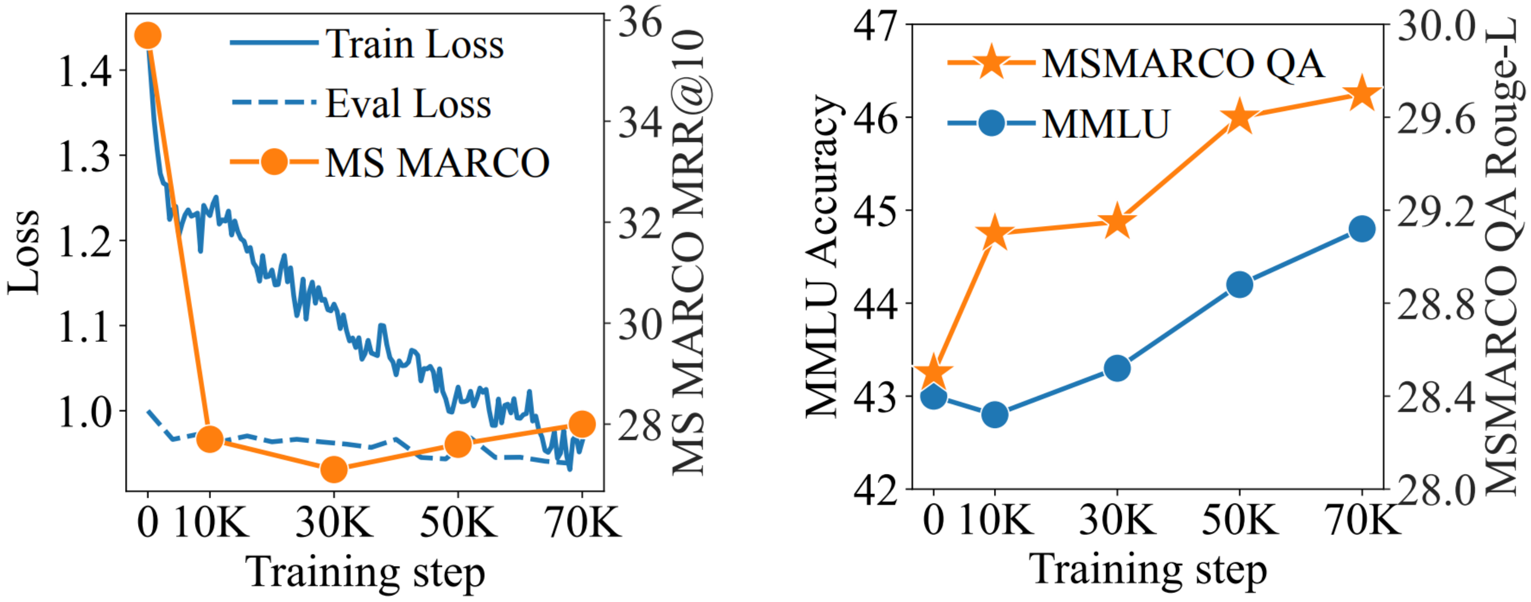
(a) Versus Human Relevance



(b) Augmented LM Performance

Figure 7: Retrieval performance w.r.t human preferences and augmented language model effective ness [10]

Finetuning Retriever to Augment LM: Why AAR Helps?



(a) Versus Human Relevance

(b) Augmented LM Performance

Figure 7: Retrieval performance w.r.t human preferences and augmented language model effective ness [10]

Figure 8: Preference agreements between FLAN-T5 variants and human labels [10]

Finetuning Retriever to Augment LM: Why AAR Helps?

Question	Human-preferred Document	LM-preferred Document
what happens if you miss your cruise ship	*If you do miss the ship, go into the cruise terminal and talk with the port agents, who are in contact with both shipboard and shoreside personnel.* They can help you decide the best way to meet your ...	*The cruise line* is not financially responsible for getting passengers to the next port if they miss the ship. Your travel to the subsequent port, or home, is on your dime, as are any necessary hotel stays and meals...
what is annexation?	*Annexation is an activity in which two things are joined together, usually with a subordinate or lesser thing being attached to a larger thing.* In strict legal terms, annexation simply involves...	Annexation (Latin ad, to, and nexus, joining) is the administrative action and concept in international law relating to the *forcible transition of one state's territory by another state*. It is generally held to be an illegal act...

Table 6: Examples of human labeled relevant documents and LM preferred augmentation odcument [10]

Search relevance requires full information

LM prefers complementary to its own knowledge?

Retrieval-Augmented LMs: Recap

Two types of retrieval augmentation:

1. Retrieving (X, Y): KNN-LM, Final version of RETRO, and Retrieval In-Context Examples
2. Retrieving (X): REALM, RAG

Pretty vanilla techniques:

- Querying with current X
- Mainly for all positions/chunks/sequences
- Pre-constructed external data store

Not much benefits in pretraining. Works very well in downstream tasks

- Retrieving (X) for additional knowledge
- Retrieval (X, Y) for better demonstrations

Still a “plug-in” to LLMs, not achieving better intelligence yet