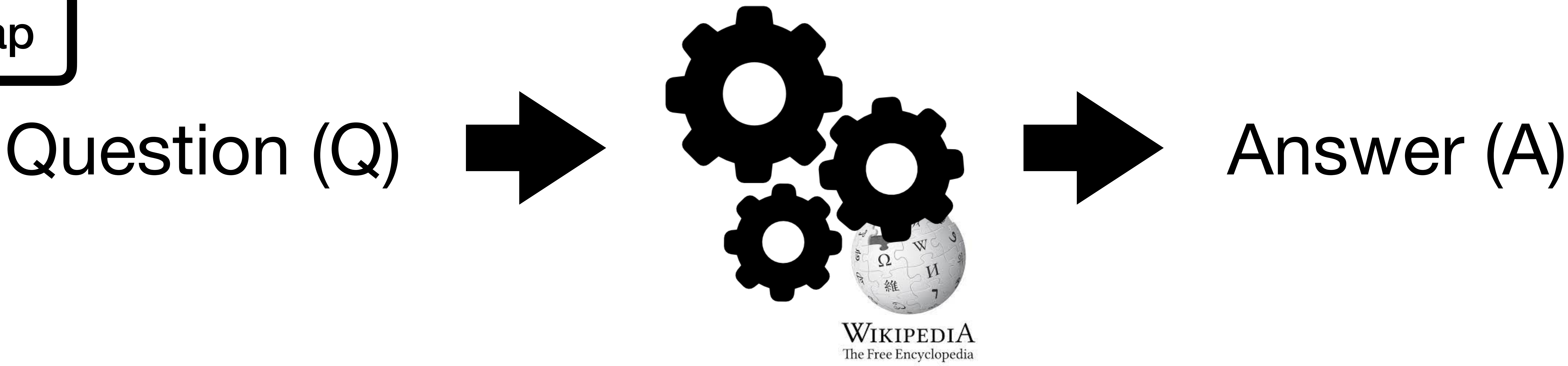
Natural Language Understanding, Generation, and Machine Translation

Lecture 23: Retrieval Augmented Generation

Pasquale Minervini
p.minervini@ed.ac.uk
March 13th, 2024

Open Domain Question Answering

Recap

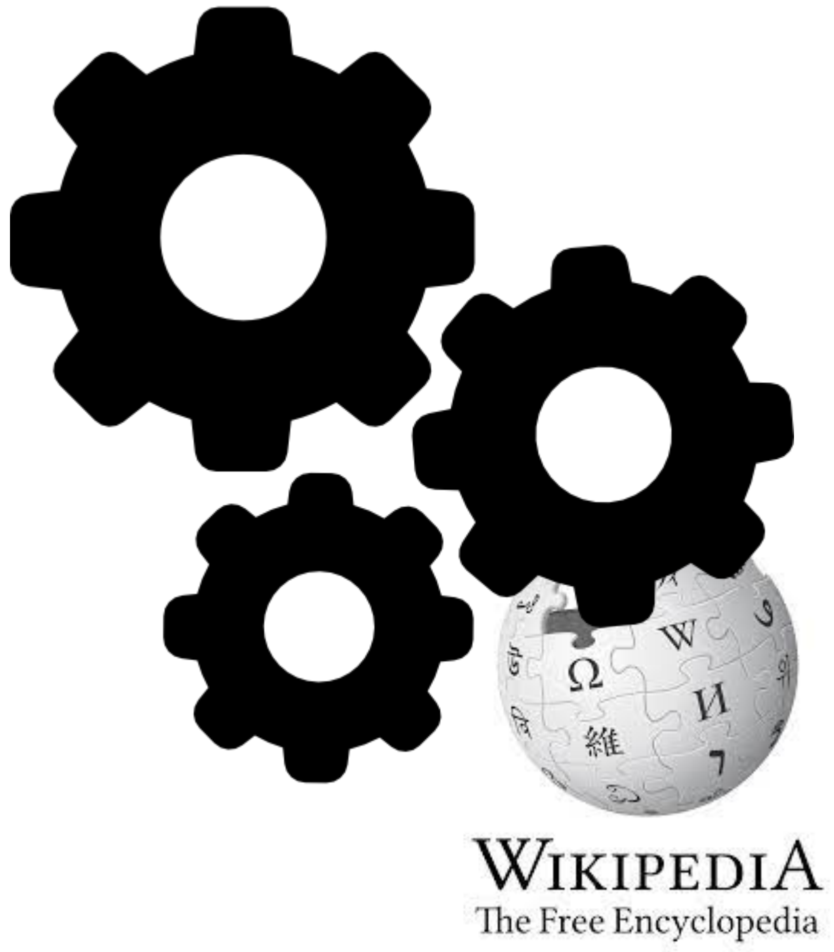


Open-Domain Question Answering (ODQA):

Open Domain Question Answering

Recap

Question (Q)



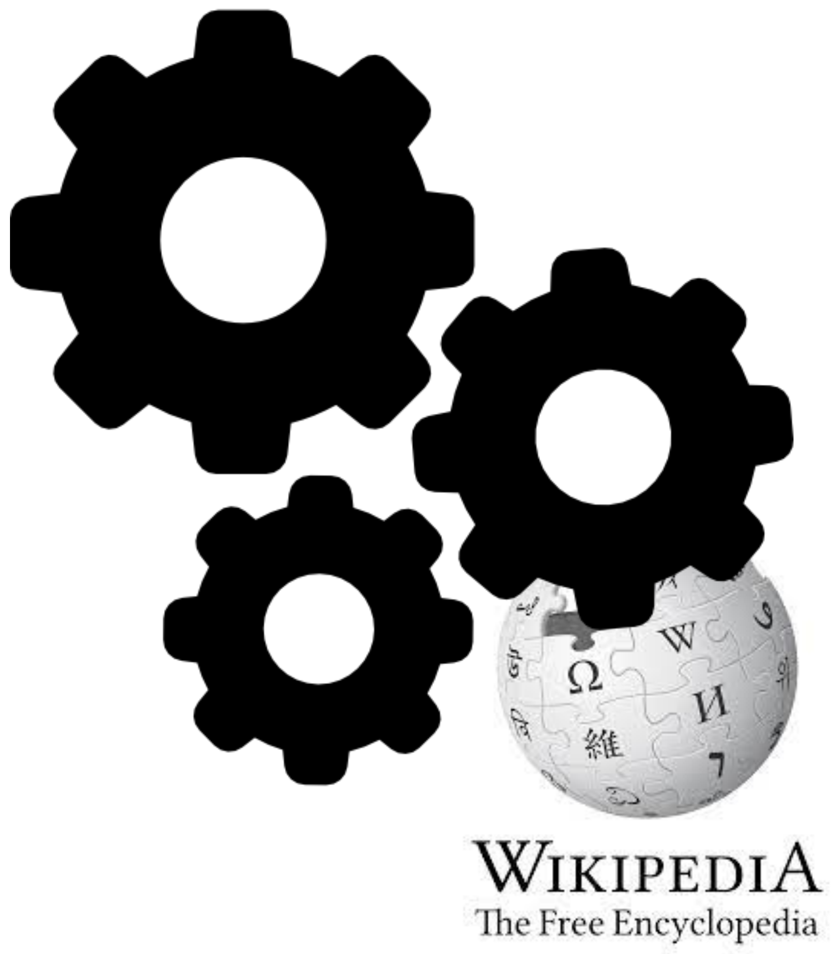
Answer (A)

Open-Domain Question Answering (ODQA):

We do not assume we are given a passage together with the question

Open Domain Question Answering

Recap

Question (Q) →  → Answer (A)

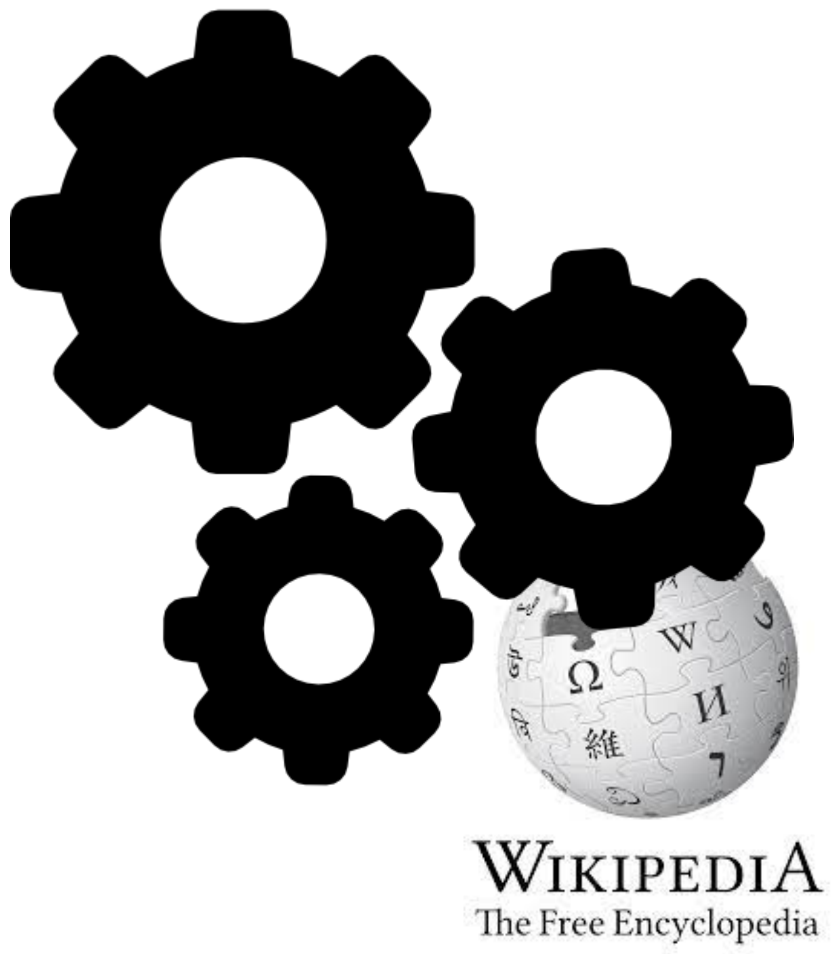
Open-Domain Question Answering (ODQA):

We do not assume we are given a passage together with the question

We can only access a large collection of documents (e.g., Wikipedia) — we don't know which document contains the answer, and the goal is to answer any open-domain questions.

Open Domain Question Answering

Recap

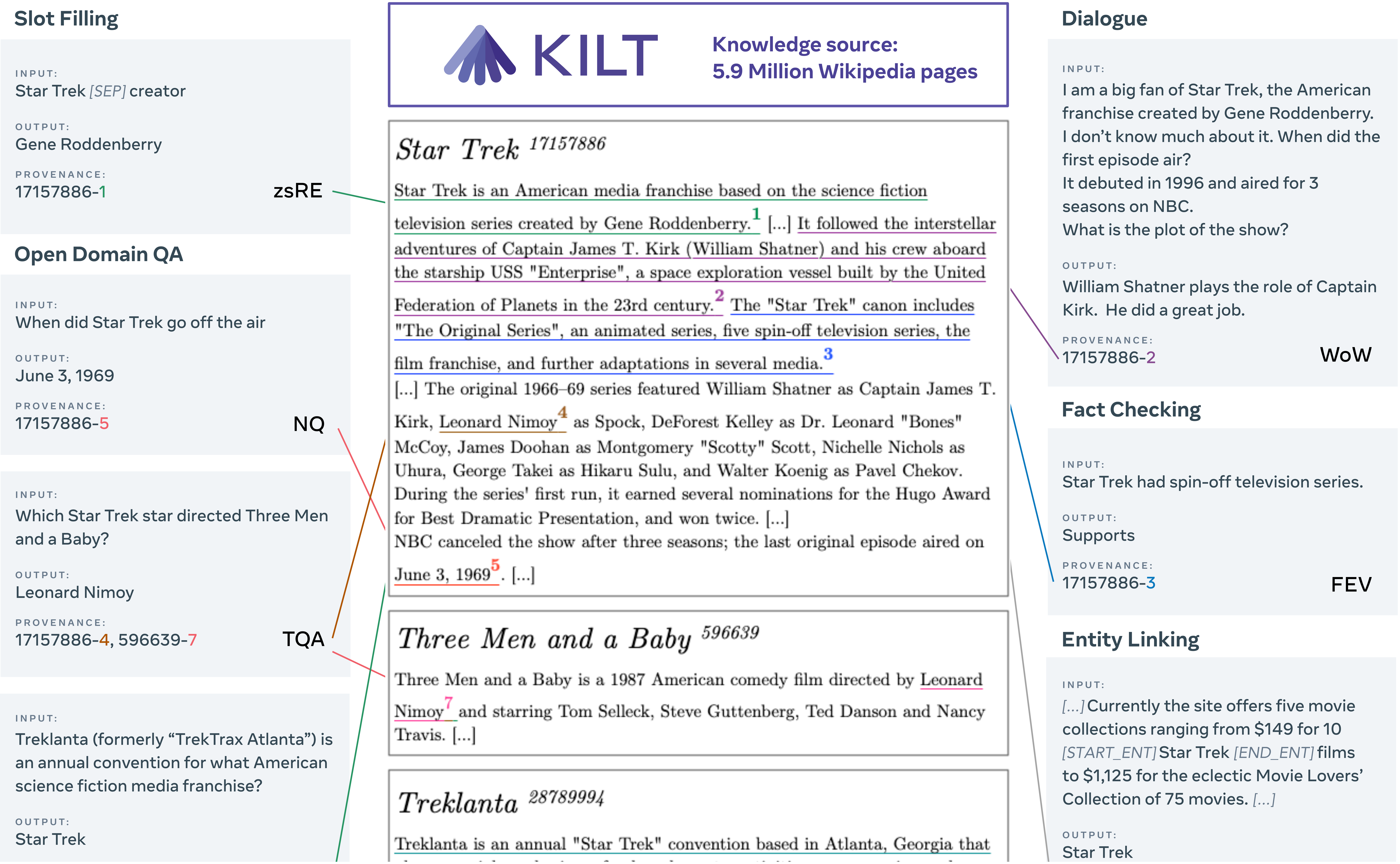
Question (Q) →  → Answer (A)

Open-Domain Question Answering (ODQA):

We do not assume we are given a passage together with the question

We can only access a large collection of documents (e.g., Wikipedia) — we don't know which document contains the answer, and the goal is to answer any open-domain questions.

Both more challenging and more practical/useful!



The KILT Benchmark

- Open-Domain Question Answering (Natural Questions, TriviaQA, HotPotQA, ELI5)
- Fact-Checking (FEVER)
- Slot Filling (T-REx, zsRE)
- Dialogue (Wizard of Wikipedia)
- Entity Linking (AIDA, WNED-WIKI, WNED-CWEB)

LLMs and their Limitations

LLMs are Extremely Impressive —

- ✅ They can store vast amounts of knowledge in their parameters/activations
- ✅ Very strong results on many tasks, even in few-shot learning settings
- ✅ Very flexible — applicable on a variety of tasks

LLMs and their Limitations

LLMs are Extremely Impressive —

- ✅ They can store vast amounts of knowledge in their parameters/activations
- ✅ Very strong results on many tasks, even in few-shot learning settings
- ✅ Very flexible — applicable on a variety of tasks

However —

- ❌ It can be difficult to update and control their knowledge/memory
- ❌ LLMs are black-boxes — no provenance or interpretability
- ❌ Very large and expensive

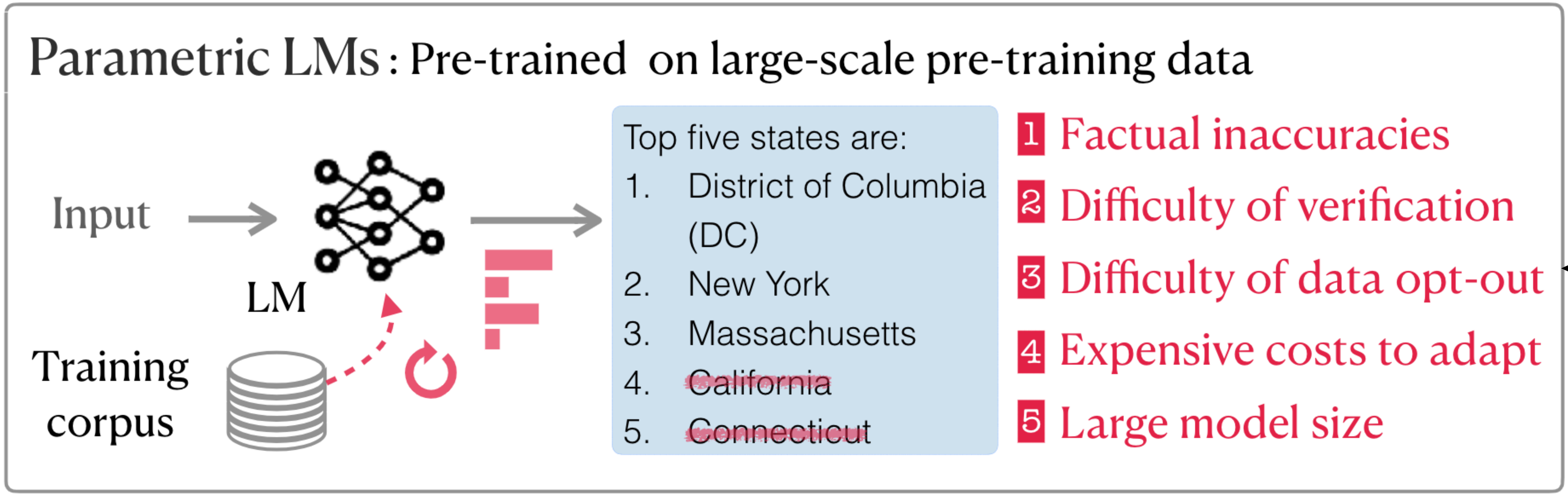
LLMs and their Limitations

Input: List the top five US states with the highest per-capita GDP, in order.

[Asai et al., 2024]

LLMs and their Limitations

Input: List the top five US states with the highest per-capita GDP, in order.

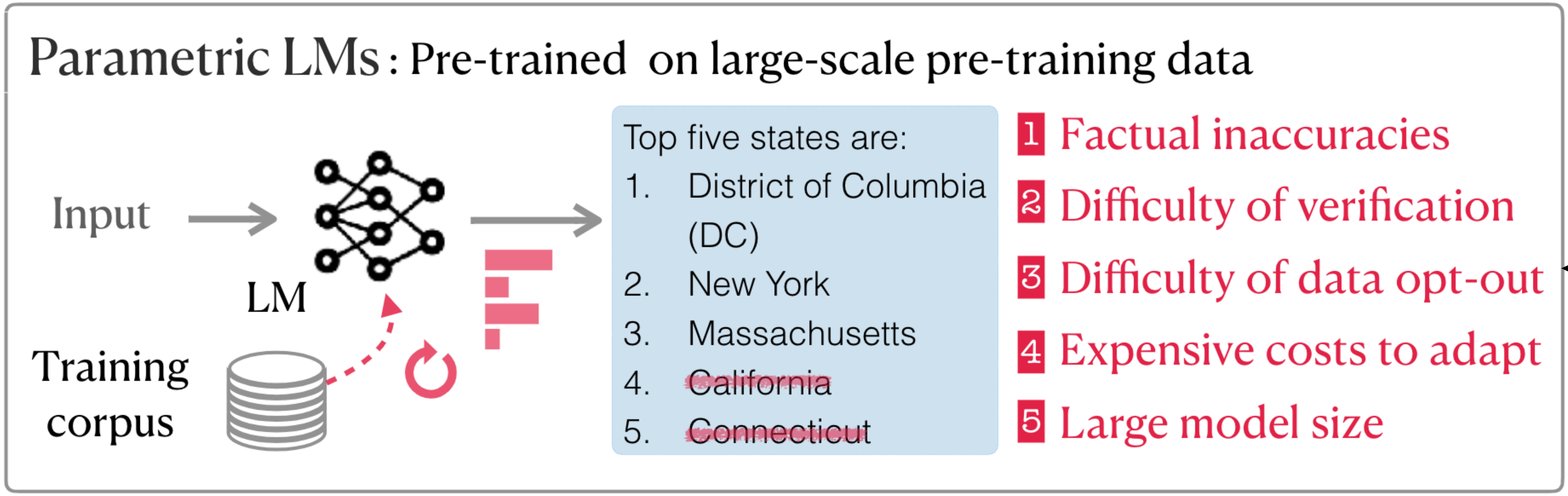


LLMs and, more generally, neural models

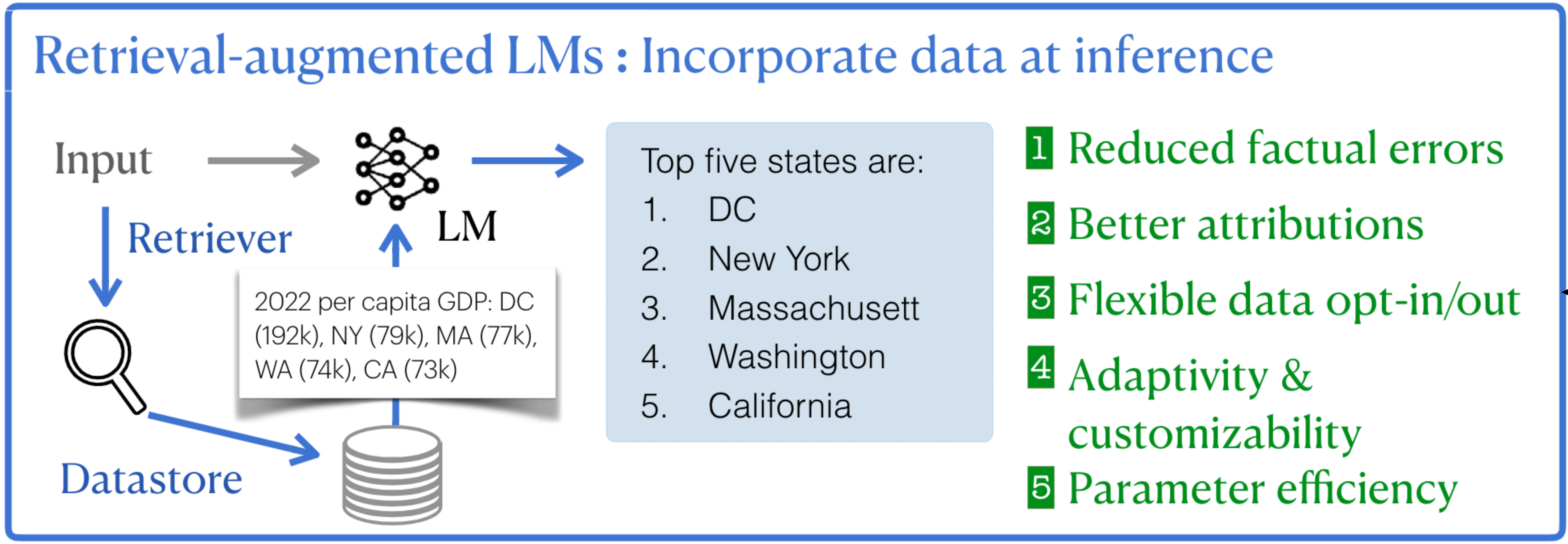
[Asai et al., 2024]

LLMs and their Limitations

Input: List the top five US states with the highest per-capita GDP, in order.



LLMs and, more generally, neural models



Retrieval-Augmented Generation models

[Asai et al., 2024]

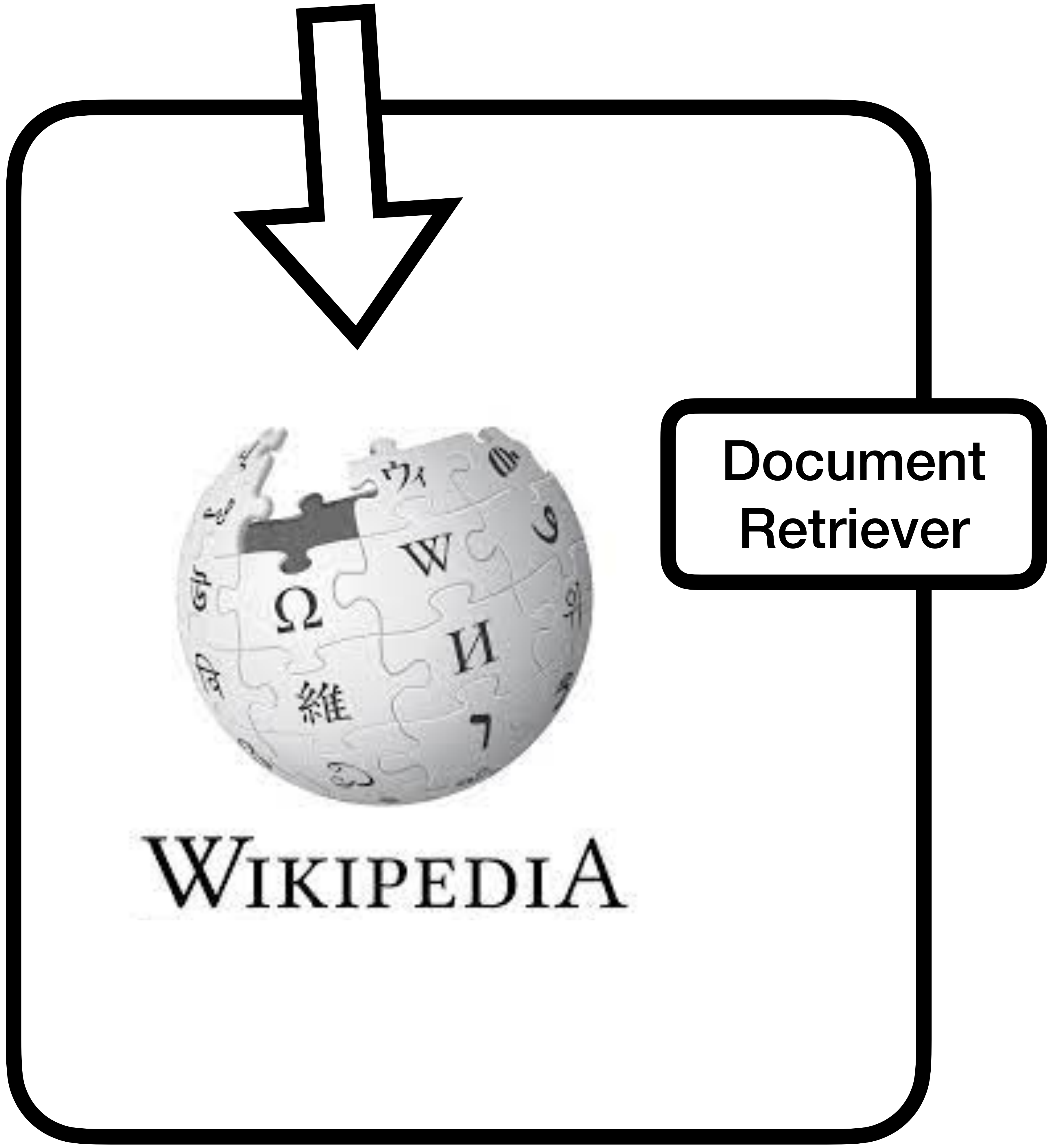
The Retriever-Reader Framework

“In what city is the University of Edinburgh located?”

[Chen et al., 2017]

The Retriever-Reader Framework

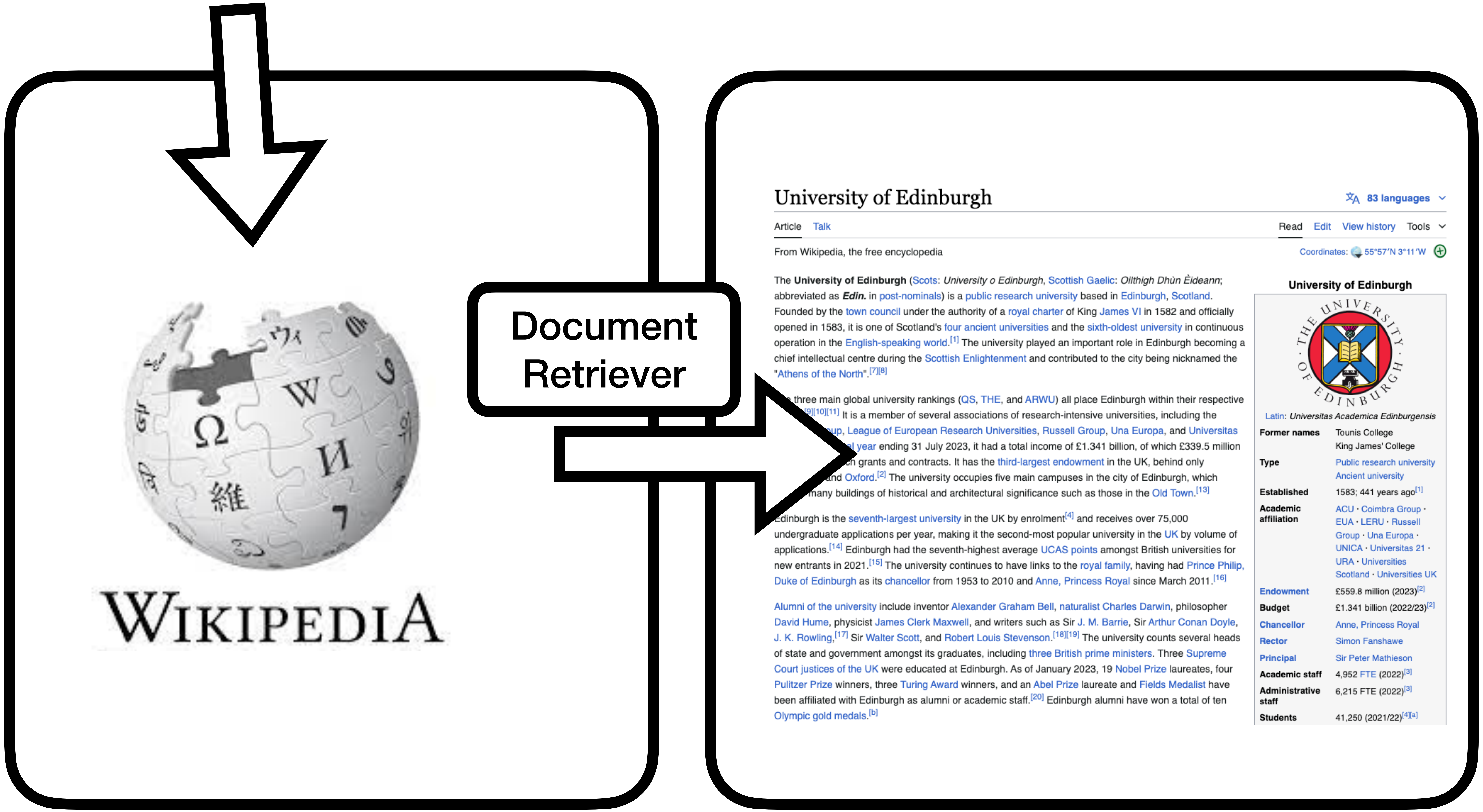
"In what city is the University of Edinburgh located?"



[Chen et al., 2017]

The Retriever-Reader Framework

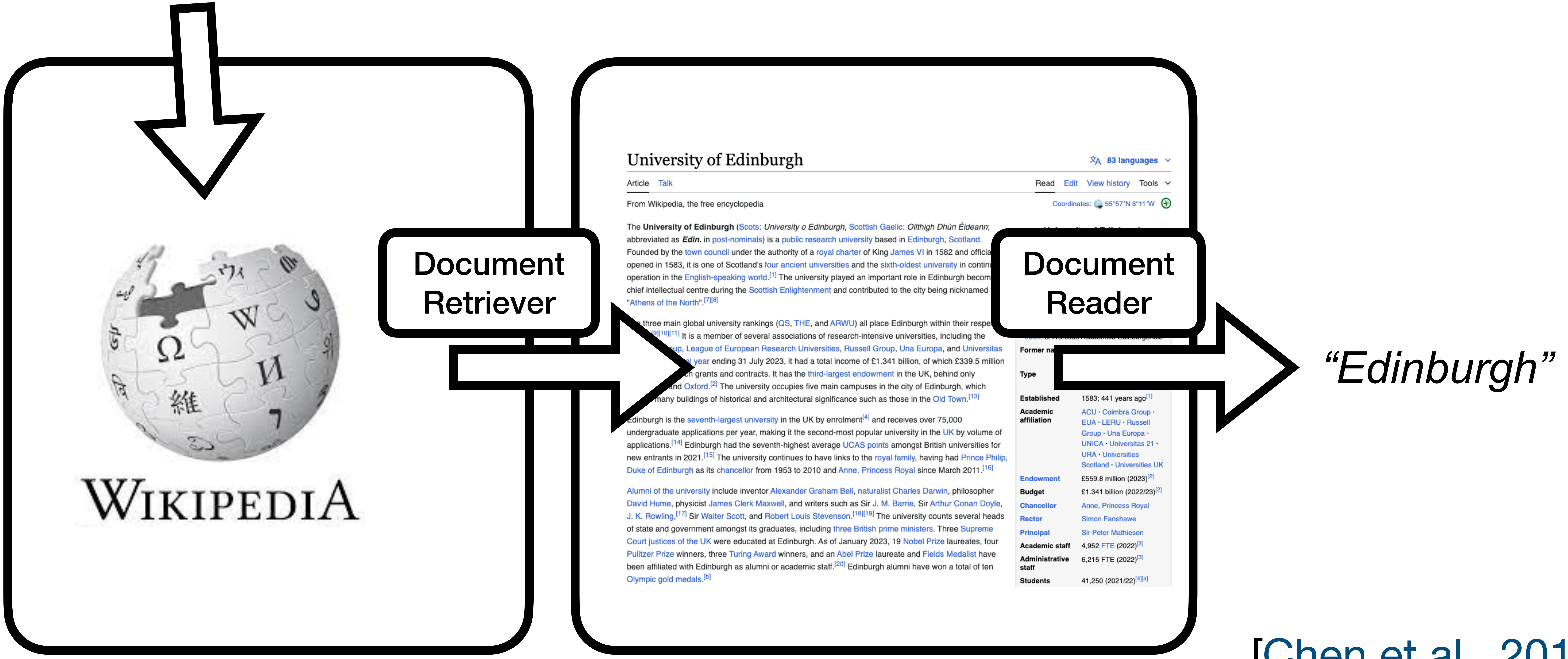
"In what city is the University of Edinburgh located?"



[Chen et al., 2017]

The Retriever-Reader Framework

“In what city is the University of Edinburgh located?”



[Chen et al., 2017]

The Retriever-Reader Framework

Input: a large collection of documents $\mathscr{D} = \{D_1, \ldots, D_n\}$ and a question Q

Output: an answer A

The Retriever-Reader Framework

Input: a large collection of documents $\mathscr{D} = \{D_1, \ldots, D_n\}$ and a question Q

Output: an answer A

Retriever: retriever$(\mathscr{D}, Q) \rightarrow P_1, \ldots, P_k$, where $k \in \mathbb{N}$ is pre-defined (e.g., 100)

Reader: reader$(Q, \{P_1, \ldots, P_k\}) \rightarrow A$, similar to reading comprehension

The Retriever-Reader Framework

Input: a large collection of documents $\mathscr{D} = \{D_1, \ldots, D_n\}$ and a question Q

Output: an answer A

Retriever: retriever($\mathscr{D}, Q$) $\rightarrow P_1, \ldots, P_k$, where $k \in \mathbb{N}$ is pre-defined (e.g., 100)

Reader: reader($Q, \{P_1, \ldots, P_k\}$) $\rightarrow A$, similar to reading comprehension

An early retriever-reader system is **DrQA** [Chen et al., 2017]:

Retriever: a standard, “classic” TF-IDF information retrieval module (fixed)

Reader: a neural reading comprehension model, trained on SQuAD via distant supervision (i.e., by using retrieved paragraphs rather than gold ones)

Dense and Sparse Retrievers

Goal: find a small subset of elements (e.g., documents, paragraphs) in a datastore that are the most similar/related/relevant to the query

$\text{sim}(Q, P)$: similarity score between a query Q and a paragraph P

Dense and Sparse Retrievers

Goal: find a small subset of elements (e.g., documents, paragraphs) in a datastore that are the most similar/related/relevant to the query

sim(Q, P): similarity score between a query Q and a paragraph P

Example: TF-IDF similarity (sparse)

$$\text{sim}(Q_i, P_j) = \boxed{\text{cosine}(\mathbf{q}, \mathbf{p})} \text{ with } \mathbf{q}, \mathbf{p} \in \mathbb{R}^{|V|}$$

Dense and Sparse Retrievers

Goal: find a small subset of elements (e.g., documents, paragraphs) in a datastore that are the most similar/related/relevant to the query

sim(Q, P): similarity score between a query Q and a paragraph P

Example: TF-IDF similarity (sparse)

$$\text{sim}(Q_i, P_j) = \boxed{\text{cosine}(\mathbf{q}, \mathbf{p})} \text{ with } \mathbf{q}, \mathbf{p} \in \mathbb{R}^{|V|}$$

$$\mathbf{q}_w = \underbrace{\text{TF}(w, Q)}_{\text{Term Frequency}} \cdot \underbrace{\text{IDF}(w, \mathcal{D})}_{\text{Inverse Document Frequency}}$$

$$\text{TF}(w, Q) = \frac{\text{freq}(w, Q)}{\sum_{w'} \text{freq}(w', Q)}$$

$$\text{IDF}(w, \mathcal{D}) = \log \frac{|\mathcal{D}|}{|\{P \in \mathcal{D} \wedge w \in P\}|}$$

Dense Retrieval in Practice

Goal: find a small subset of elements (e.g., documents, paragraphs) in a datastore that are the most similar/related/relevant to the query

$\text{sim}(Q, P)$: similarity score between a query Q and a paragraph P

Example: Dense Retrieval

$$\text{sim}(Q_i, P_j) = \boxed{\mathbf{q}_i^\top \mathbf{p}_j} \text{ with } \mathbf{q}_i, \mathbf{p}_j \in \mathbb{R}^d$$

Neural network is used to calculate the query embedding and the paragraph embedding

Dense Retrieval in Practice

Goal: find a small subset of elements (e.g., documents, paragraphs) in a datastore that are the most similar/related/relevant to the query

sim(Q, P): similarity score between a query Q and a paragraph P

Example: Dense Retrieval

$$\text{sim}(Q_i, P_j) = \boxed{\mathbf{q}_i^\top \mathbf{p}_j} \text{ with } \mathbf{q}_i, \mathbf{p}_j \in \mathbb{R}^d$$

$$\mathbf{q}_i = \text{Encode}(Q_i)$$

$$\mathbf{p}_j = \text{Encode}(P_j)$$

Entire research on how to improve or learn the similarity function!

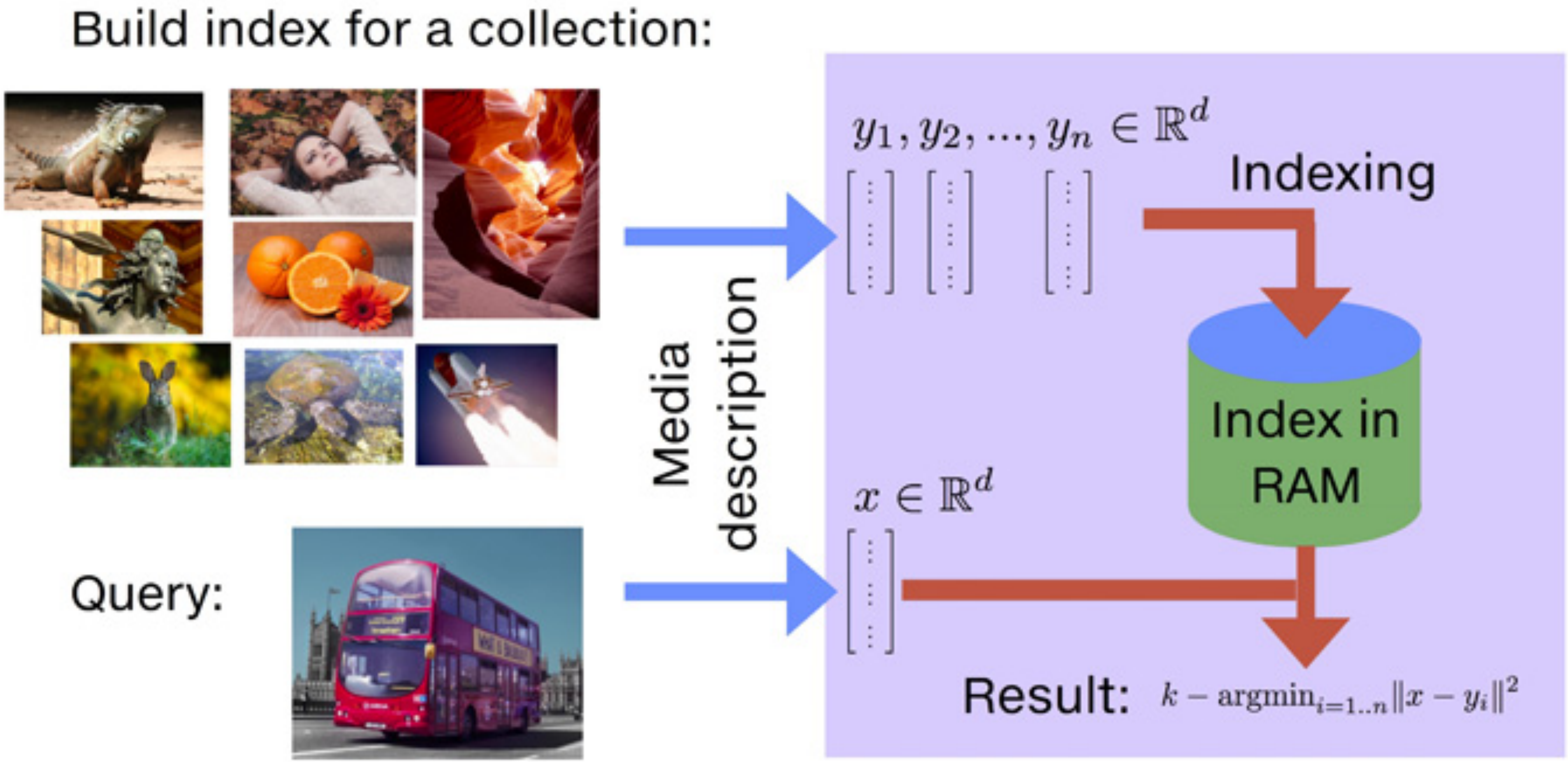
Dense Retrieval in Practice

Goal: find a small subset of elements (e.g., documents, paragraphs) in a datastore that are the most similar/related/relevant to the query

sim(Q, P): similarity score between a query Q and a paragraph P

Index: given a query embedding $\mathbf{q}_i \in \mathbb{R}^d$, returns the top-$k$ paragraph embeddings $\mathbf{p}_1, \ldots, \mathbf{p}_k \in \mathbb{R}^d$ via **maximum inner-product search** (MIPS)

Software: `FAISS`, `SCaNN`, `Annoy`, …



Software: `FAISS`, `SCaNN`, `Annoy`, …

Summary of methods

The basic indexes are given hereafter:

Method	Class name	`index_factory`	Main parameters	Bytes/vector	Exhaustiv
Exact Search for L2	`IndexFlatL2`	`"Flat"`	`d`	`4*d`	yes
Exact Search for Inner Product	`IndexFlatIP`	`"Flat"`	`d`	`4*d`	yes
Hierarchical Navigable Small World graph exploration	`IndexHNSWFlat`	`"HNSW,Flat"`	`d`, `M`	`4*d + x * M * 2 * 4`	no
Inverted file with exact post-verification	`IndexIVFFlat`	`"IVFx,Flat"`	`quantizer`, `d`, `nlists`, `metric`	`4*d + 8`	no
Locality-Sensitive Hashing (binary flat index)	`IndexLSH`	-	`d`, `nbits`	`ceil(nbits/8)`	yes
Scalar quantizer (SQ) in flat mode	`IndexScalarQuantizer`	`"SQ8"`	`d`	`d`	yes
Product quantizer (PQ) in flat mode	`IndexPQ`	`"PQx"`, `"PQ"M"x"nbits`	`d`, `M`, `nbits`	`ceil(M * nbits / 8)`	yes

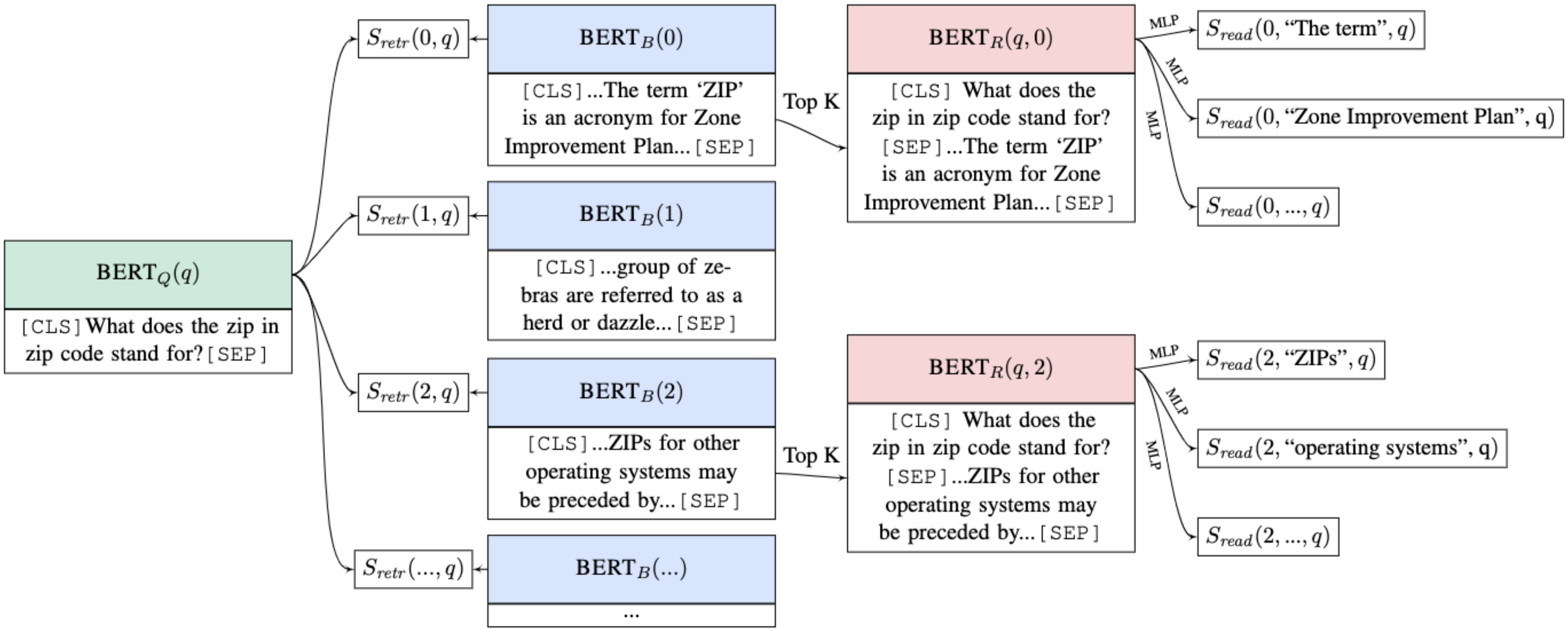
Exact Search

Approximate Search
(Scales to Billions of vectors)

CPU vs. GPU

Early End-to-End Trainable Reader-Retriever Models

Early method for training the retrieval component proposed by Lee et al., 2019:

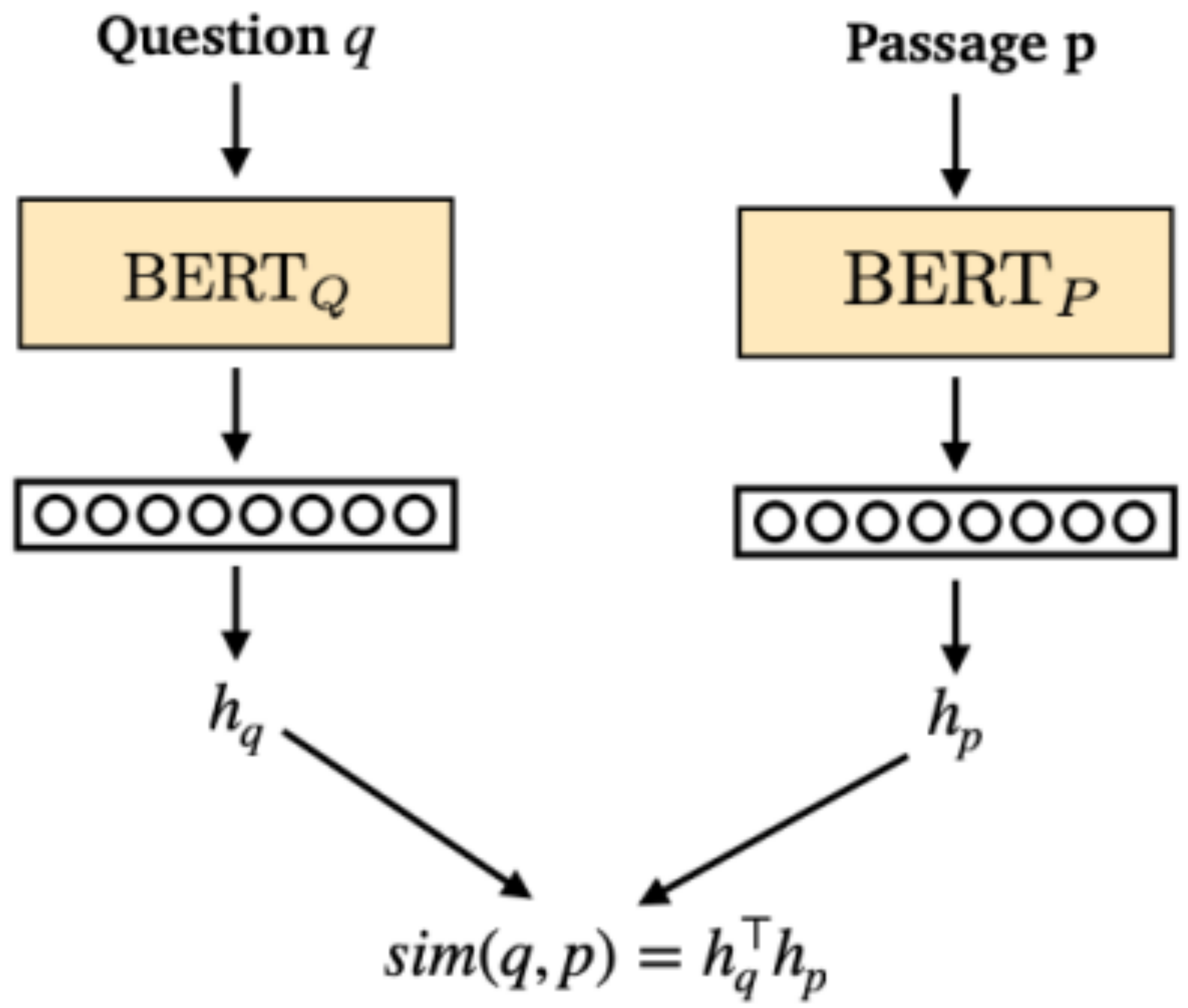


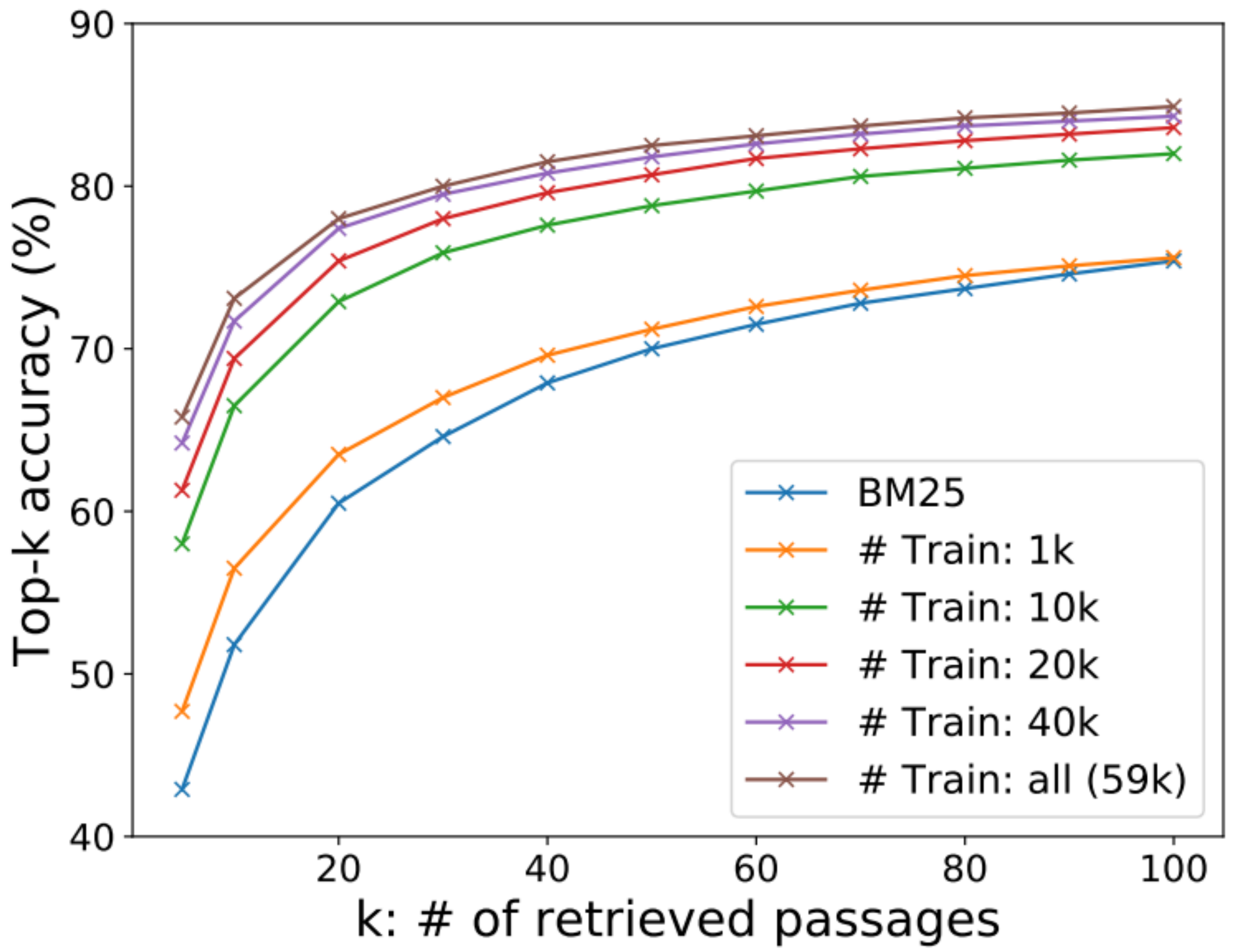
Each passage can be encoded as a vector using BERT and the retriever score can be measured as the dot product between the question and passage representations

Not easy to model as there are a huge number of passages (21M in Eng. Wikipedia)

Early End-to-End Trainable Reader-Retriever Models

Later, Dense Passage Retrieval [DPR, Karpukhin et al., 2020] authors propose to **train the retriever using question-answer pairs**:

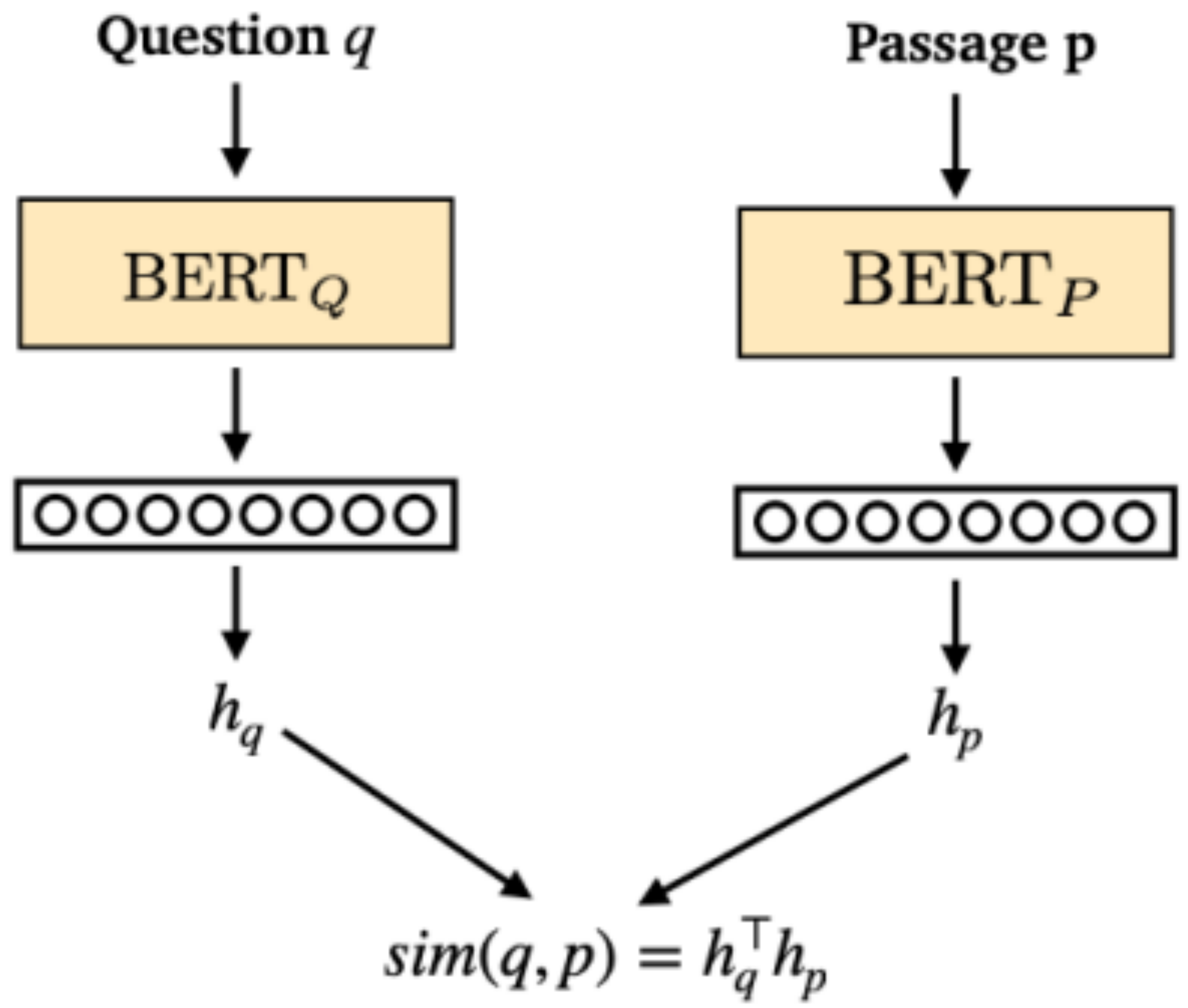


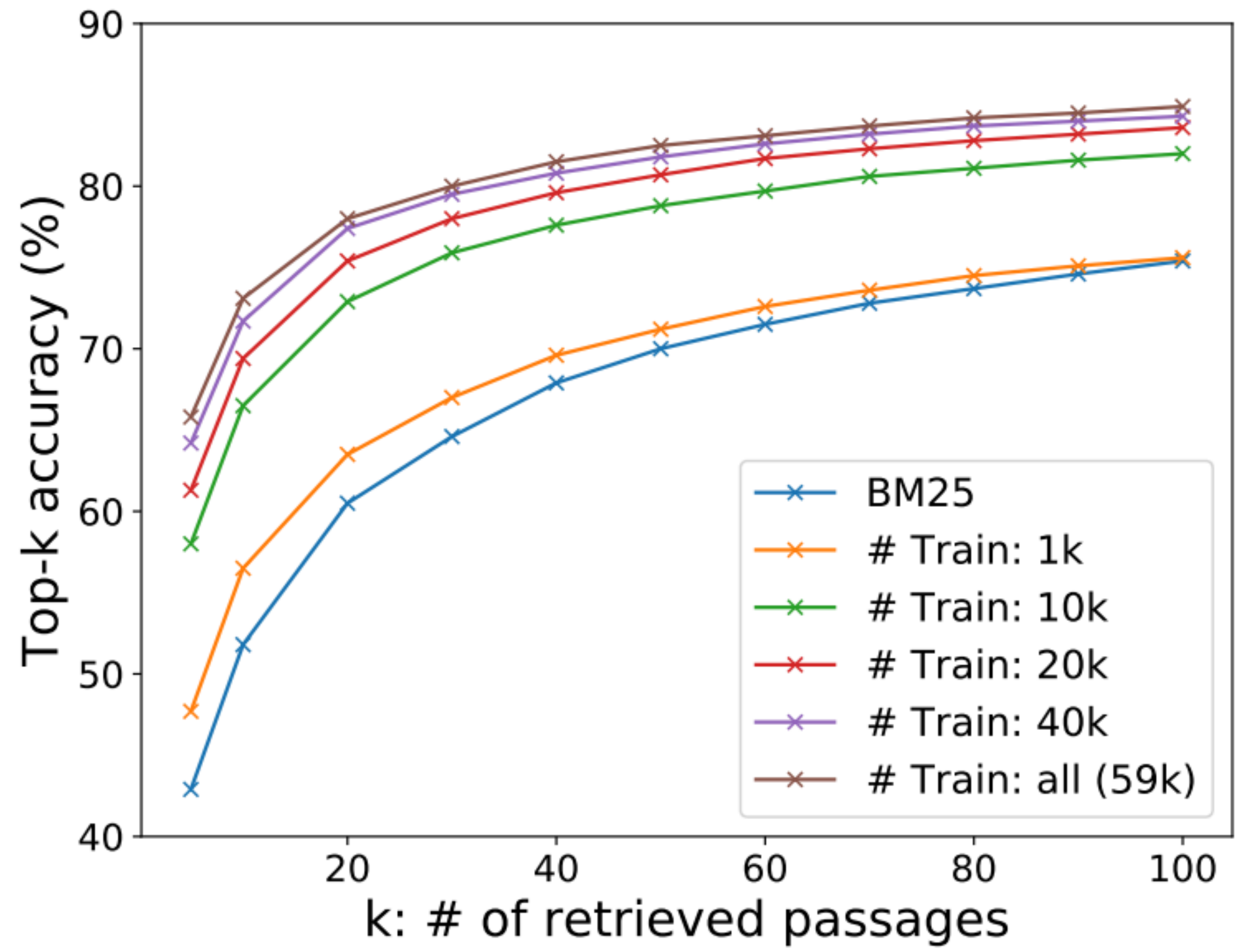


Trainable retriever (using BERT) can produce more accurate results than traditional IR models, such as BM25 and TF-IDF

Early End-to-End Trainable Reader-Retriever Models

Later, Dense Passage Retrieval [DPR, Karpukhin et al., 2020] authors propose to **train the retriever using question-answer pairs**:



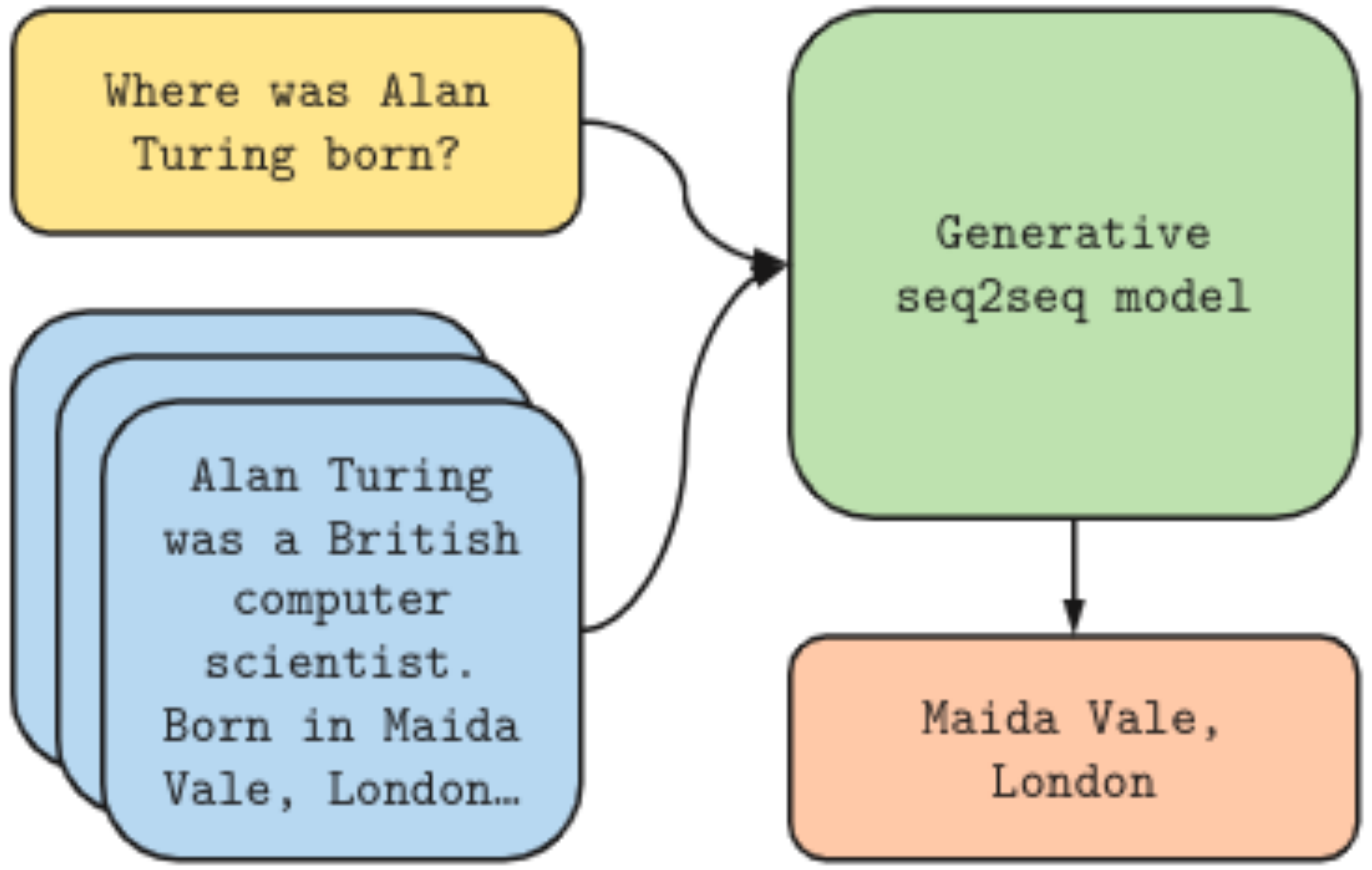


Trainable retriever (using BERT) can produce more accurate results than traditional IR models, such as BM25 and TF-IDF

…although this was slightly controversial — see e.g., "A Replication Study of DPR" **https://arxiv.org/abs/2104.05740**

Dense Retrieval and Generative Models

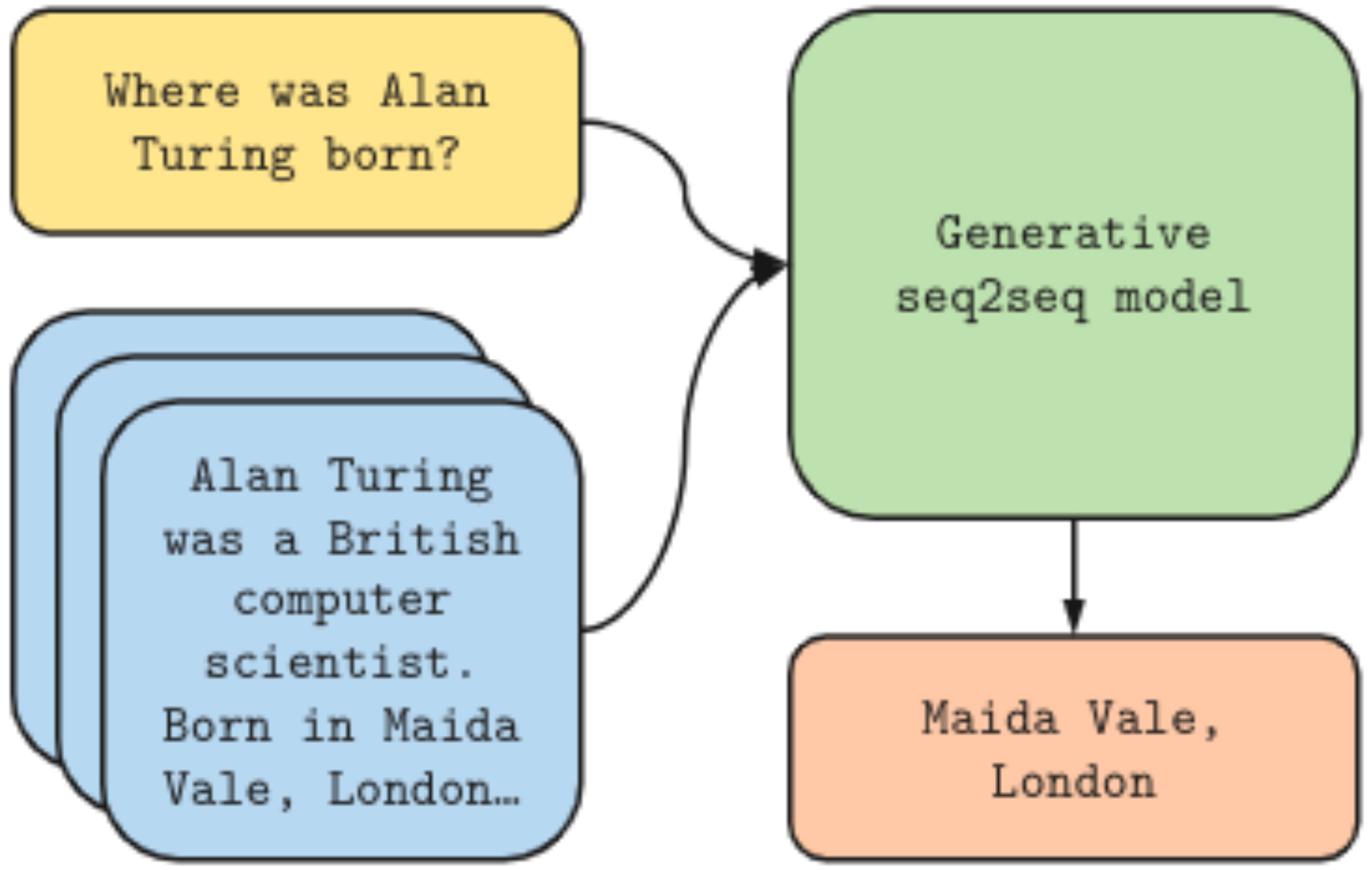
Recent works show that it can be beneficial to **generate answers** rather than **extracting them** from retrieved passages, e.g., Fusion-in-Decoder [Izacard et al., 2021]



Fusion-in-Decoder (FiD):
DPR & T5

Dense Retrieval and Generative Models

Recent works show that it can be beneficial to **generate answers** rather than **extracting them** from retrieved passages, e.g., Fusion-in-Decoder [Izacard et al., 2021]

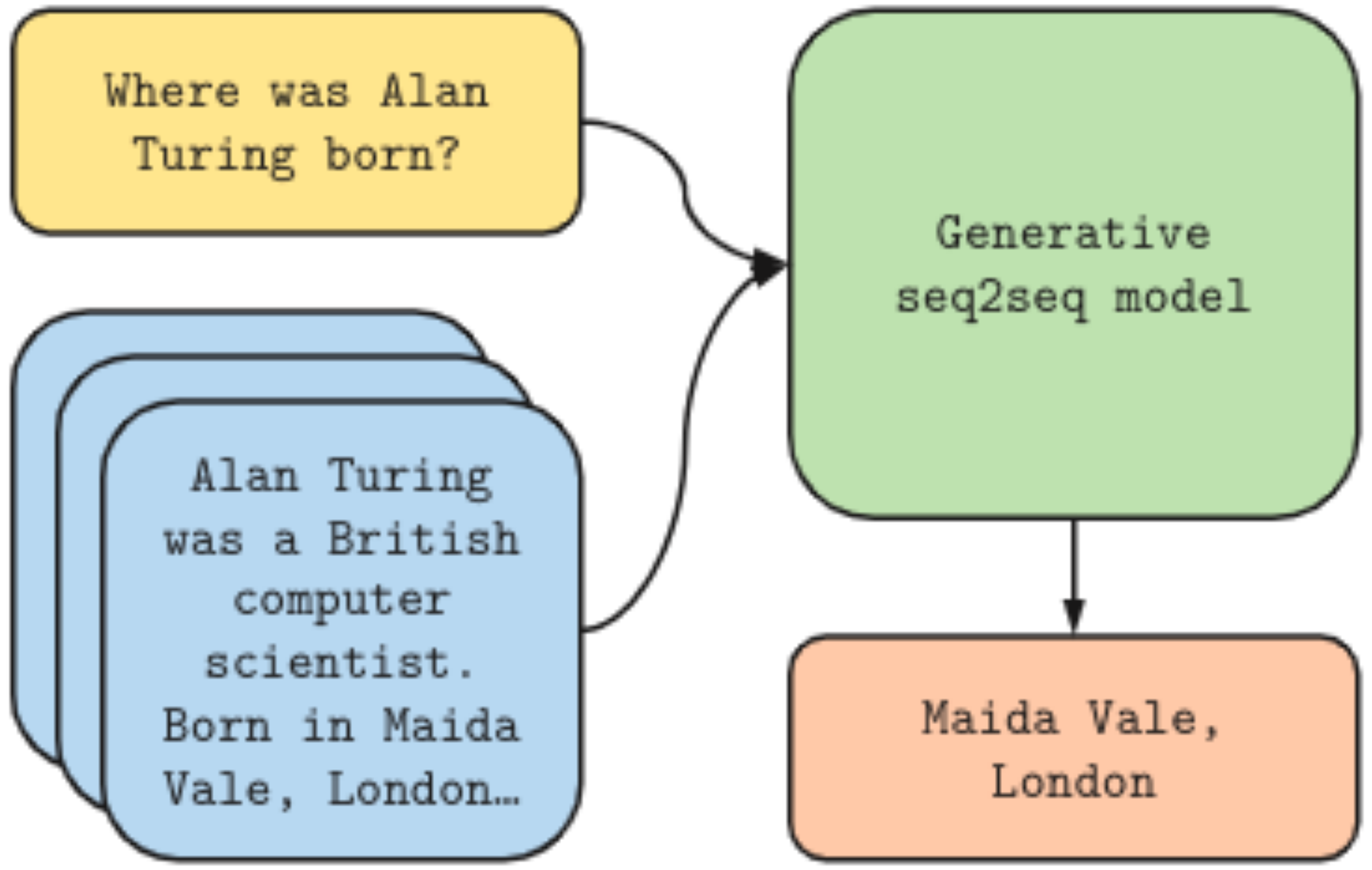


Fusion-in-Decoder (FiD):
DPR & T5

Model	NQ	TriviaQA		SQuAD Open	
	EM	EM	EM	EM	F1
DrQA (Chen et al., 2017)	-	-	-	29.8	-
Multi-Passage BERT (Wang et al., 2019)	-	-	-	53.0	60.9
Path Retriever (Asai et al., 2020)	31.7	-	-	**56.5**	**63.8**
Graph Retriever (Min et al., 2019b)	34.7	55.8	-	-	-
Hard EM (Min et al., 2019a)	28.8	50.9	-	-	-
ORQA (Lee et al., 2019)	31.3	45.1	-	20.2	-
REALM (Guu et al., 2020)	40.4	-	-	-	-
DPR (Karpukhin et al., 2020)	41.5	57.9	-	36.7	-
SpanSeqGen (Min et al., 2020)	42.5	-	-	-	-
RAG (Lewis et al., 2020b)	44.5	56.1	68.0	-	-
T5 (Roberts et al., 2020)	36.6	-	60.5	-	-
GPT-3 few shot (Brown et al., 2020)	29.9	-	71.2	-	-
Fusion-in-Decoder (base)	48.2	65.0	77.1	53.4	60.6
Fusion-in-Decoder (large)	**51.4**	**67.6**	**80.1**	**56.7**	63.2

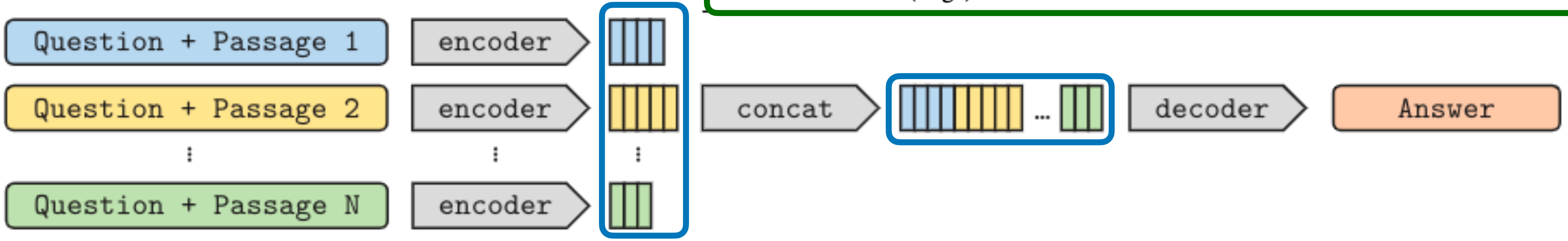
Dense Retrieval and Generative Models

Recent works show that it can be beneficial to **generate answers** rather than **extracting them** from retrieved passages, e.g., Fusion-in-Decoder [Izacard et al., 2021]



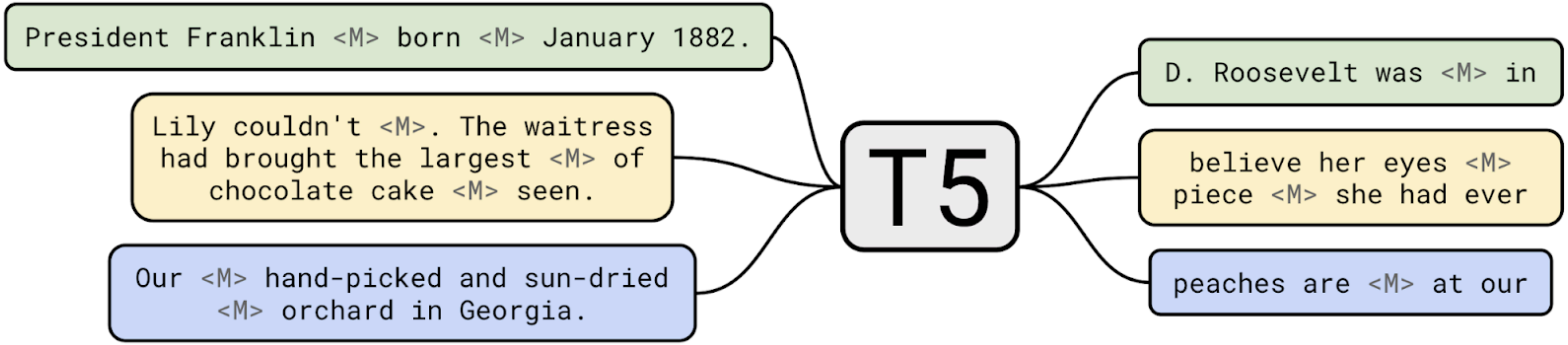
Fusion-in-Decoder (FiD):
DPR & T5

Model	NQ	TriviaQA		SQuAD Open	
	EM	EM	EM	EM	F1
DrQA (Chen et al., 2017)	-	-	-	29.8	-
Multi-Passage BERT (Wang et al., 2019)	-	-	-	53.0	60.9
Path Retriever (Asai et al., 2020)	31.7	-	-	**56.5**	**63.8**
Graph Retriever (Min et al., 2019b)	34.7	55.8	-	-	-
Hard EM (Min et al., 2019a)	28.8	50.9	-	-	-
ORQA (Lee et al., 2019)	31.3	45.1	-	20.2	-
REALM (Guu et al., 2020)	40.4	-	-	-	-
DPR (Karpukhin et al., 2020)	41.5	57.9	-	36.7	-
SpanSeqGen (Min et al., 2020)	42.5	-	-	-	-
RAG (Lewis et al., 2020b)	44.5	56.1	68.0	-	-
T5 (Roberts et al., 2020)	36.6	-	60.5	-	-
GPT-3 few shot (Brown et al., 2020)	29.9	-	71.2	-	-
Fusion-in-Decoder (base)	48.2	65.0	77.1	53.4	60.6
Fusion-in-Decoder (large)	**51.4**	**67.6**	**80.1**	**56.7**	63.2



LLMs Can Do Open-Domain QA

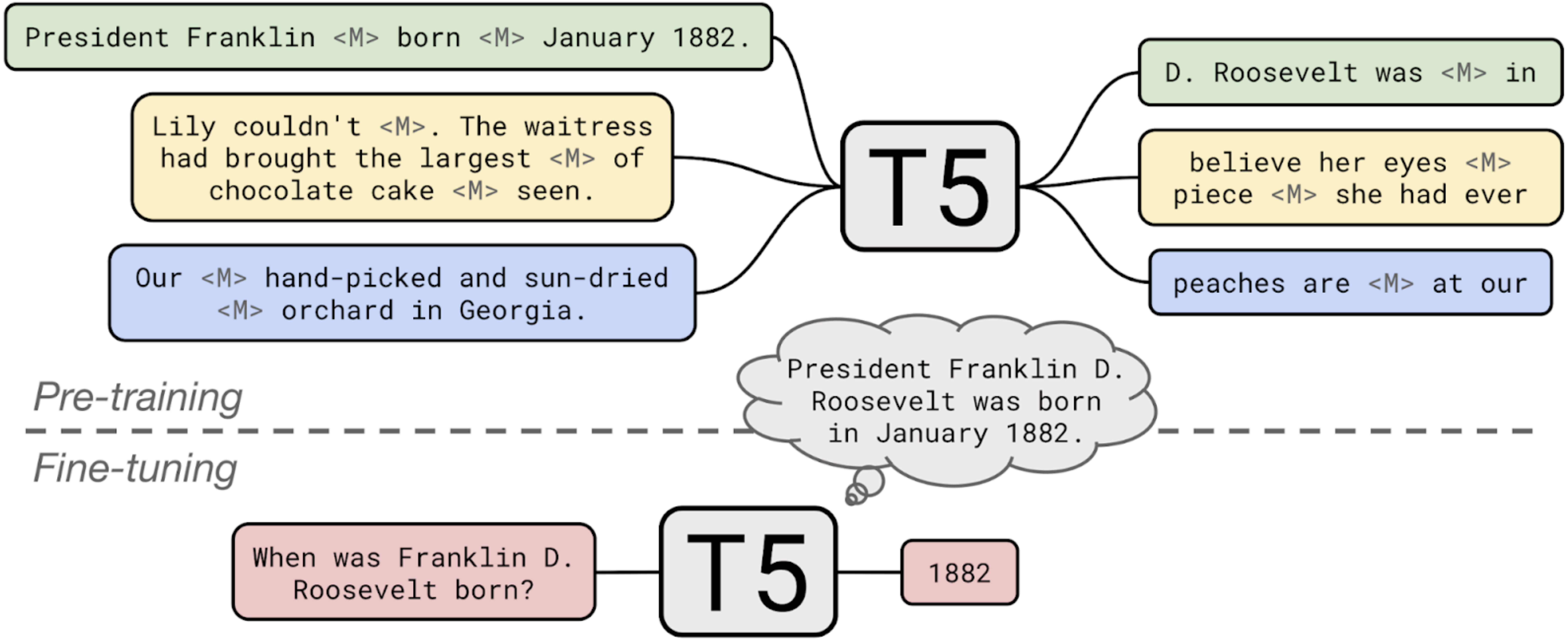
LLMs — without an (explicit) retrieval component — can be used to solve Open-Domain Question Answering tasks; knowledge about the world is encoded in their parameters and activations, rather than in a corpus:



[Roberts et al., 2020]

LLMs Can Do Open-Domain QA

LLMs — without an (explicit) retrieval component — can be used to solve Open-Domain Question Answering tasks; knowledge about the world is encoded in their parameters and activations, rather than in a corpus:



[Roberts et al., 2020]

LLMs vs. RAG — Generalisation

How do LLMs **really** compare with RAG models, in terms of accuracy and generalisation, on open-domain question answering tasks?

Model	Open Natural Questions				TriviaQA				WebQuestions			
	Total	Question Overlap	Answer Overlap Only	No Overlap	Total	Question Overlap	Answer Overlap Only	No Overlap	Total	Question Overlap	Answer Overlap Only	No Overlap

The question appears in the training set

The answer appears in the training set

[Lewis et al., 2020]

LLMs vs. RAG — Generalisation

How do LLMs **really** compare with RAG models, in terms of accuracy and generalisation, on open-domain question answering tasks?

Model		Open Natural Questions				TriviaQA				WebQuestions			
		Total	Question Overlap	Answer Overlap Only	No Overlap	Total	Question Overlap	Answer Overlap Only	No Overlap	Total	Question Overlap	Answer Overlap Only	No Overlap
Closed book	T5-11B+SSM	36.6	77.2	22.2	9.4	-	-	-	-	44.7	82.1	44.5	22.0
	BART	26.5	67.6	10.2	0.8	26.7	67.3	16.3	0.8	27.4	71.5	20.7	1.6

The question appears in the training set

The answer appears in the training set

[Lewis et al., 2020]

LLMs vs. RAG — Generalisation

How do LLMs **really** compare with RAG models, in terms of accuracy and generalisation, on open-domain question answering tasks?

Model		Open Natural Questions				TriviaQA				WebQuestions			
		Total	Question Overlap	Answer Overlap Only	No Overlap	Total	Question Overlap	Answer Overlap Only	No Overlap	Total	Question Overlap	Answer Overlap Only	No Overlap
Open book	RAG	44.5	70.7	34.9	24.8	56.8	82.7	54.7	29.2	45.5	81.0	45.8	21.1
	DPR	41.3	69.4	34.6	19.3	57.9	80.4	59.6	31.6	42.4	74.1	39.8	22.2
	FID	51.4	71.3	48.3	34.5	67.6	87.5	66.9	42.8	-	-	-	-
Closed book	T5-11B+SSM	36.6	77.2	22.2	9.4	-	-	-	-	44.7	82.1	44.5	22.0
	BART	26.5	67.6	10.2	0.8	26.7	67.3	16.3	0.8	27.4	71.5	20.7	1.6

[Lewis et al., 2020]

LLMs vs. RAG — Generalisation

How do LLMs **really** compare with RAG models, in terms of accuracy and generalisation, on open-domain question answering tasks?

Model		Natural Questions				TriviaQA			
		Total	Overlap	Comp -gen	Novel -entity	Total	Overlap	Comp -gen	Novel -entity
Parametric	T5-11B+SSM	36.59	**81.48**	17.47	12.56	-	-	-	-
	BART	26.54	76.34	5.88	3.35	26.78	78.38	11.37	10.09

Unseen entity-relation pair

New entity

[Liu et al., 2021]

LLMs vs. RAG — Generalisation

How do LLMs **really** compare with RAG models, in terms of accuracy and generalisation, on open-domain question answering tasks?

Model		Natural Questions				TriviaQA			
		Total	Overlap	Comp -gen	Novel -entity	Total	Overlap	Comp -gen	Novel -entity
Non-parametric	RAG	44.49	75.75	30.41	37.69	56.83	87.12	47.58	47.81
	FiD	**53.13**	78.85	**40.00**	**47.74**	**67.69**	**90.39**	**58.10**	**66.23**
	DPR	41.27	71.33	25.88	33.84	57.91	82.31	46.11	58.99
	RePAQ	47.26	78.61	34.21	36.85	52.06	89.08	42.95	38.38
Parametric	T5-11B+SSM	36.59	**81.48**	17.47	12.56	-	-	-	-
	BART	26.54	76.34	5.88	3.35	26.78	78.38	11.37	10.09

[Liu et al., 2021]

LLMs vs. RAG — Updateability

		2017 Test Set Acc.		2020 Test Set Acc.	
Train Set	Test-time Index	Closed-book	ATLAS	Closed-book	ATLAS
2017 answers	2017				
	2020				
2020 answers	2017				
	2020				

T5-XXL

Retrieval-augmented

[Izacard et al., 2022]

LLMs vs. RAG — Updateability

		2017 Test Set Acc.		2020 Test Set Acc.	
Train Set	Test-time Index	Closed-book	ATLAS	Closed-book	ATLAS
2017 answers	2017	12.1		2.9	
	2020	12.1		2.9	
2020 answers	2017	4.8		3.6	
	2020	4.8		3.6	

Retrieval-augmented

[Izacard et al., 2022]

LLMs vs. RAG — Updateability

		2017 Test Set Acc.		2020 Test Set Acc.	
Train Set	Test-time Index	Closed-book	ATLAS	Closed-book	ATLAS
2017 answers	2017	12.1	57.7	2.9	1.5
	2020	12.1	10.2	2.9	53.1
2020 answers	2017	4.8	50.1	3.6	4.2
	2020	4.8	3.5	3.6	60.5

[Izacard et al., 2022]

LLMs vs. RAG — Accuracy

	NQ		TriviaQA filtered		TriviaQA unfiltered	
Model	64-shot	Full	64-shot	Full	64-shot	Full
GPT-3 (Brown et al., 2020)	29.9	-	-	-	71.2	-
Gopher (Rae et al., 2021)	28.2	-	57.2	-	61.3	-
Chinchilla (Hoffmann et al., 2022)	35.5	-	64.6	-	72.3	-
PaLM (Chowdhery et al., 2022)	39.6	-	-	-	81.4	-
RETRO (Borgeaud et al., 2021)	-	45.5	-	-	-	-
FiD (Izacard & Grave, 2020)	-	51.4	-	67.6	-	80.1
FiD-KD (Izacard & Grave, 2021)	-	54.7	-	73.3	-	-
R2-D2 (Fajcik et al., 2021)	-	55.9	-	69.9	-	-

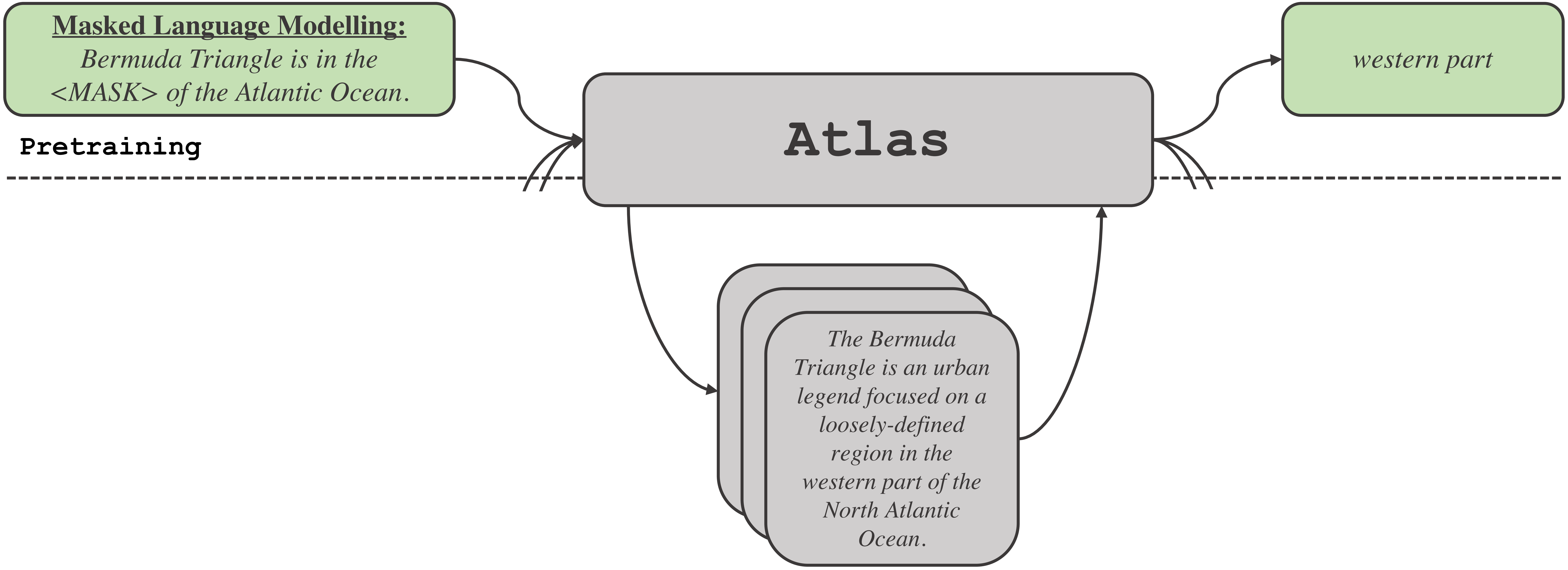
[Izacard et al., 2022]

LLMs vs. RAG — Accuracy

	NQ		TriviaQA filtered		TriviaQA unfiltered	
Model	64-shot	Full	64-shot	Full	64-shot	Full
GPT-3 (Brown et al., 2020)	29.9	-	-	-	71.2	-
Gopher (Rae et al., 2021)	28.2	-	57.2	-	61.3	-
Chinchilla (Hoffmann et al., 2022)	35.5	-	64.6	-	72.3	-
PaLM (Chowdhery et al., 2022)	39.6	-	-	-	81.4	-
RETRO (Borgeaud et al., 2021)	-	45.5	-	-	-	-
FiD (Izacard & Grave, 2020)	-	51.4	-	67.6	-	80.1
FiD-KD (Izacard & Grave, 2021)	-	54.7	-	73.3	-	-
R2-D2 (Fajcik et al., 2021)	-	55.9	-	69.9	-	-
ATLAS	**42.4**	**60.4**	**74.5**	**79.8**	**84.7**	**89.4**

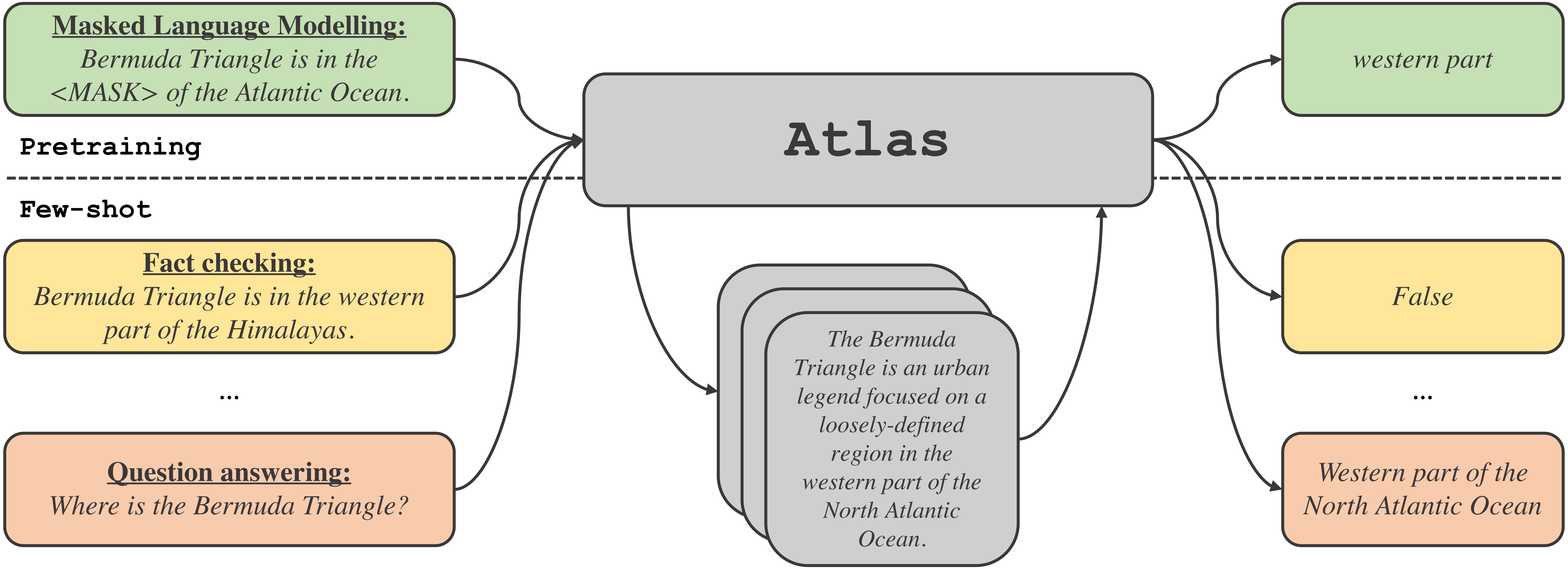
[Izacard et al., 2022]

ATLAS — A Retrieval-Augmented LM



[Izacard et al., 2022]

ATLAS — A Retrieval-Augmented LM



[Izacard et al., 2022]

Retrieval-Augmented LMs

What to retrieve?

Query

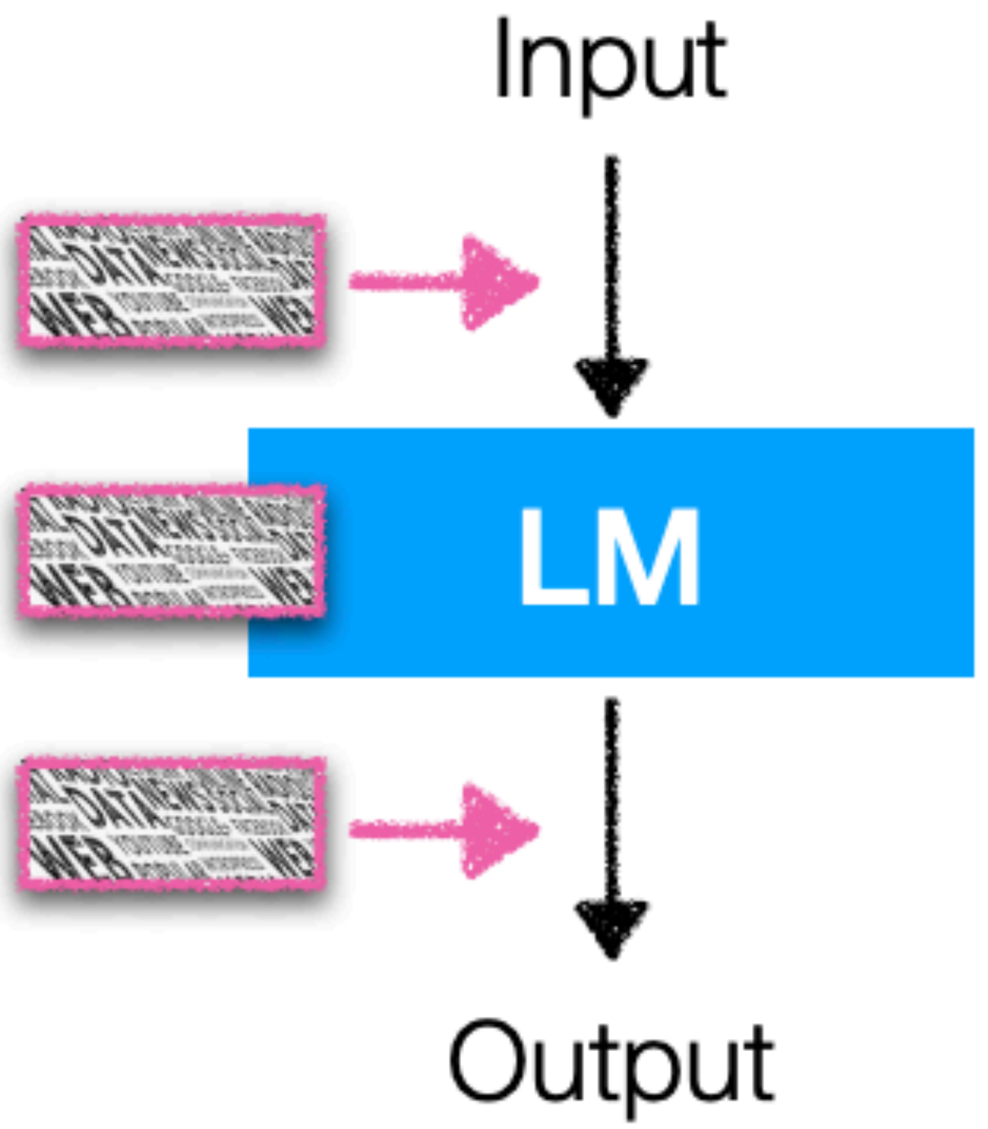
Text chunks (passages)?
Tokens?
Something else?

Retrieval-Augmented LMs

What to retrieve?

Text chunks (passages)?
Tokens?
Something else?

How to use retrieval?

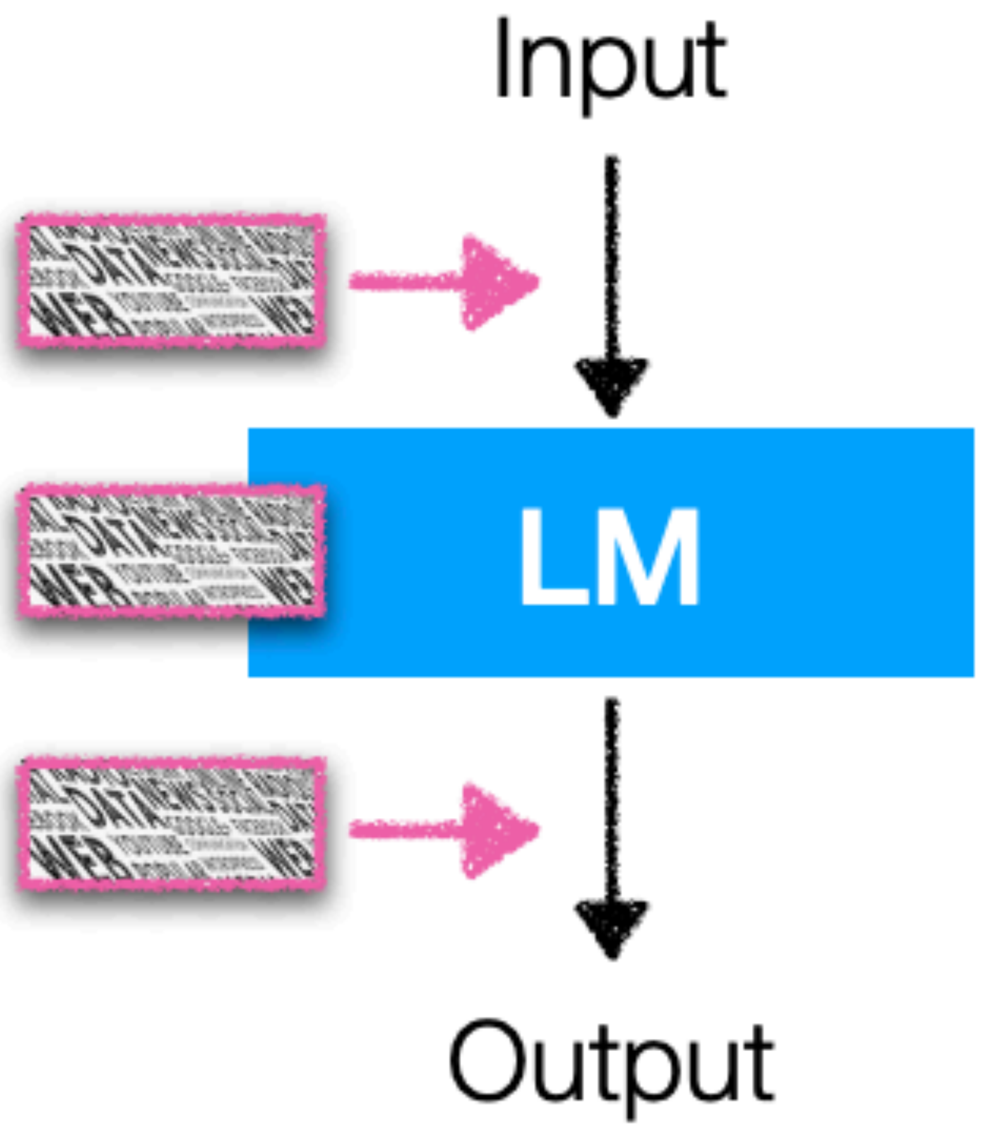


Retrieval-Augmented LMs

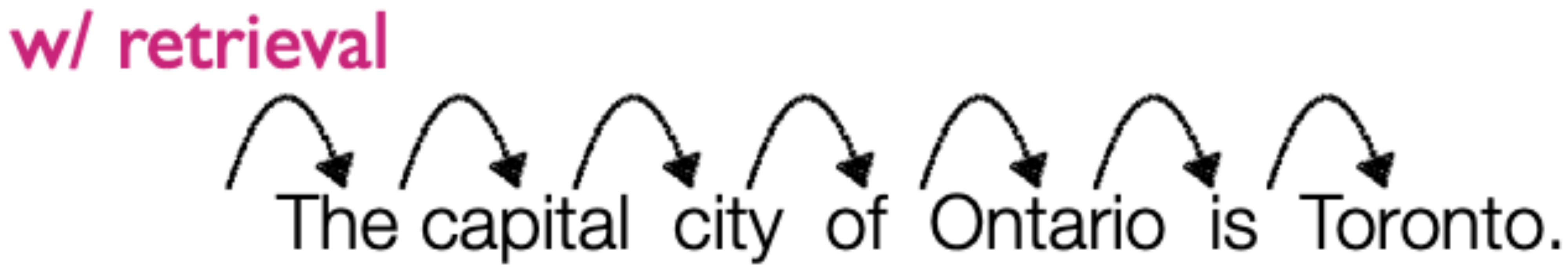
What to retrieve?

Text chunks (passages)?
Tokens?
Something else?

How to use retrieval?



When to retrieve?

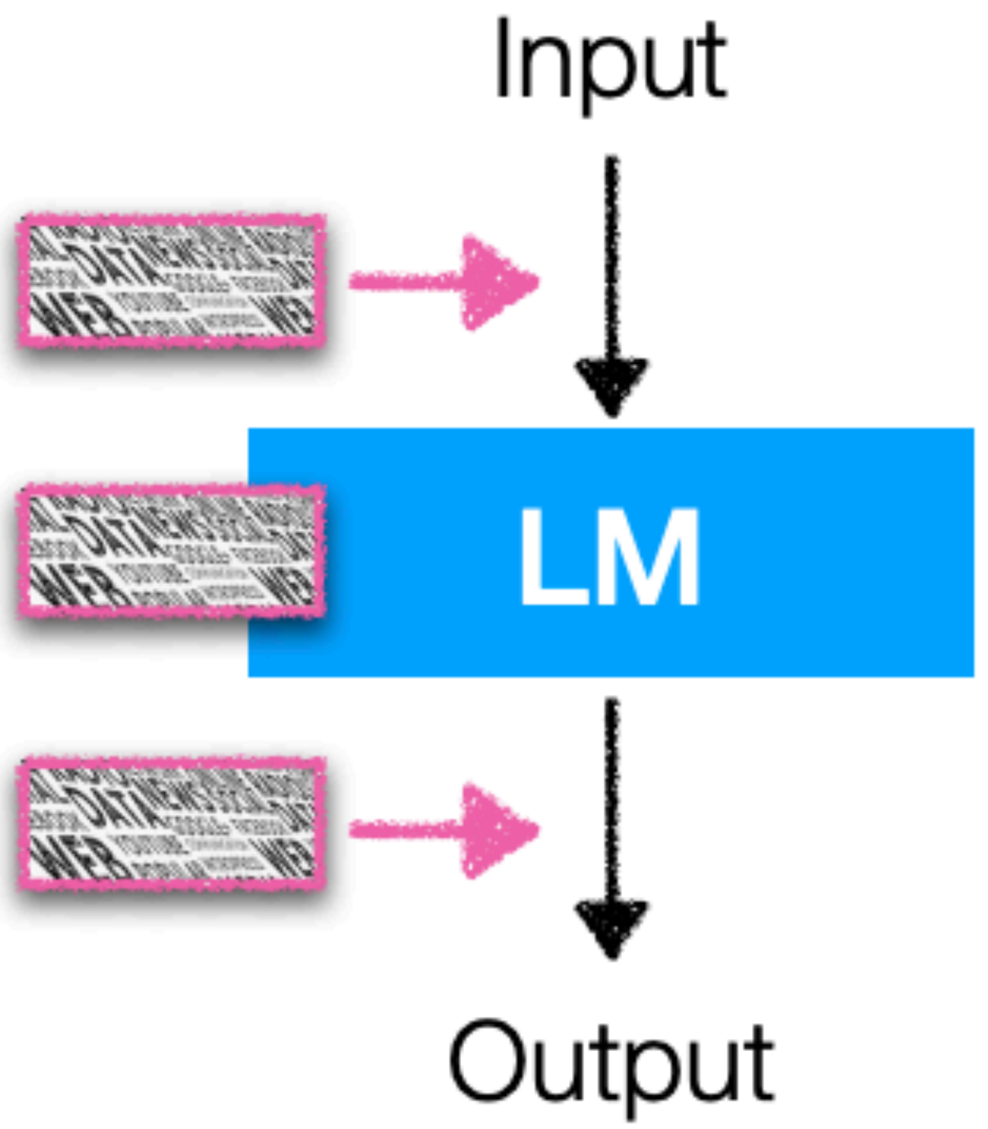


Retrieval-Augmented LMs

What to retrieve?

Text chunks (passages)?
Tokens?
Something else?

How to use retrieval?



When to retrieve?

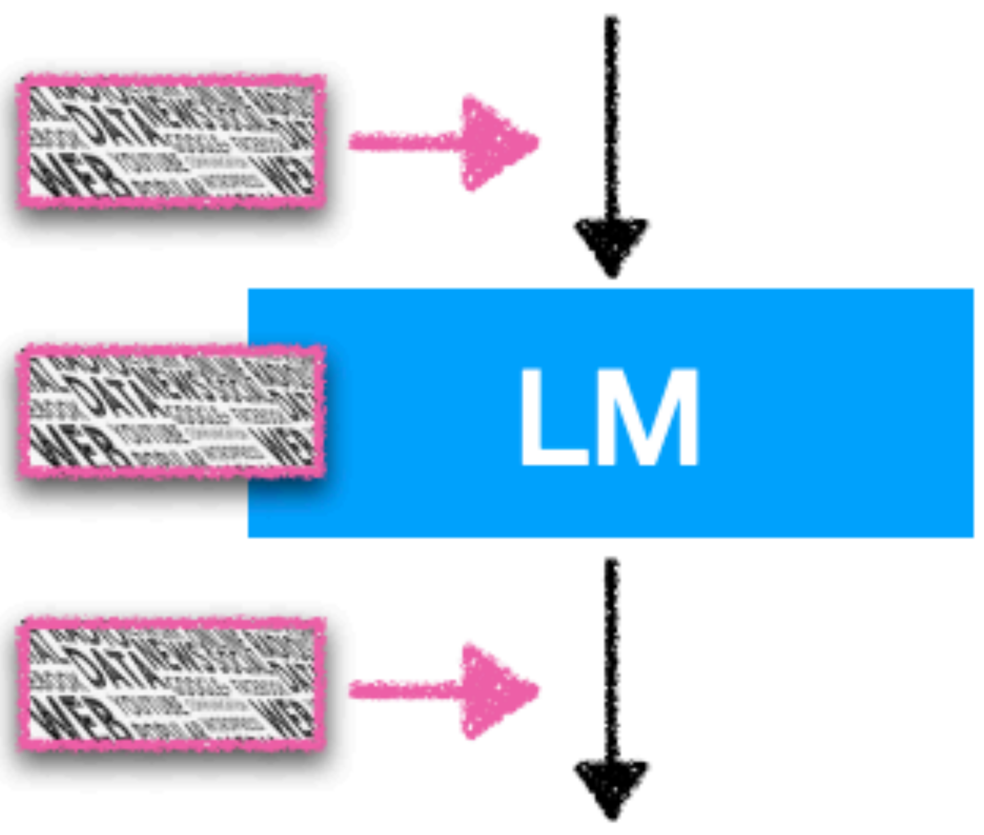
Retrieval-Augmented LMs

What to retrieve?

Query

Text chunks (passages)?
Tokens?
Something else?

How to use retrieval?



When to retrieve?

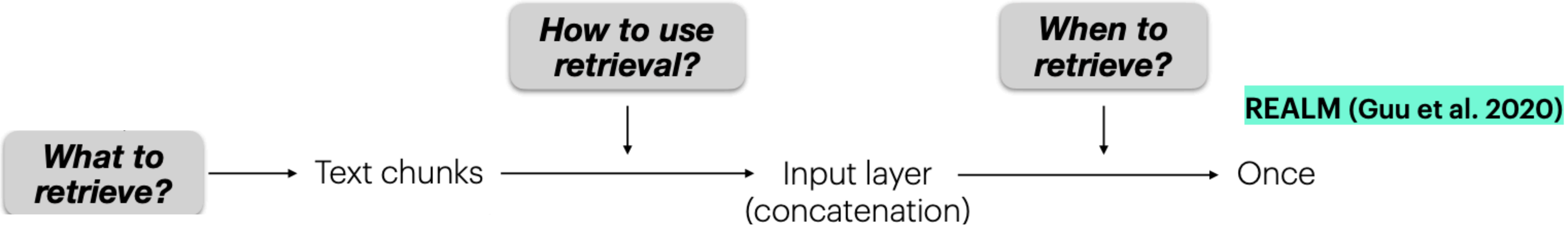
w/ retrieval
The capital city of Ontario is Toronto.

w/ retrieval w/ r w/r w/r w/ r w/r w/r
costly
The capital city of Ontario is Toronto.

w/ retrieval w/r w/r
The capital city of Ontario is Toronto.

When we encounter named entity, then retreival can be performed

Retrieval-Augmented LMs



[Asai et al., 2023]

Retrieval-Augmented LMs

[Asai et al., 2023]

REALM [Guu et al., 2020]

x = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.

World Cup 2022 was … the increase to [MASK] in 2026.

↓

LM

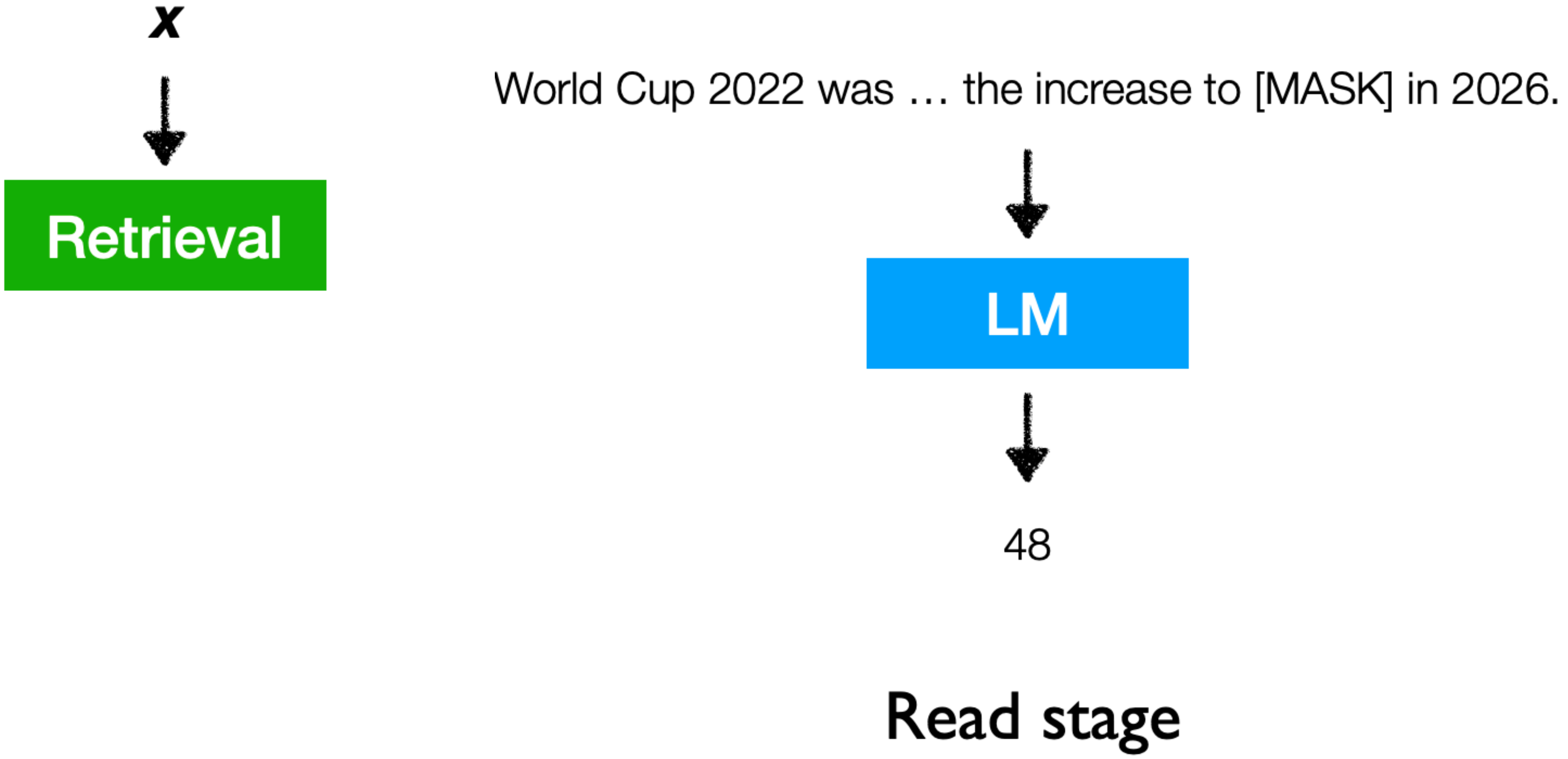
↓

48

Read stage

REALM [Guu et al., 2020]

$\boldsymbol{x}$ = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.

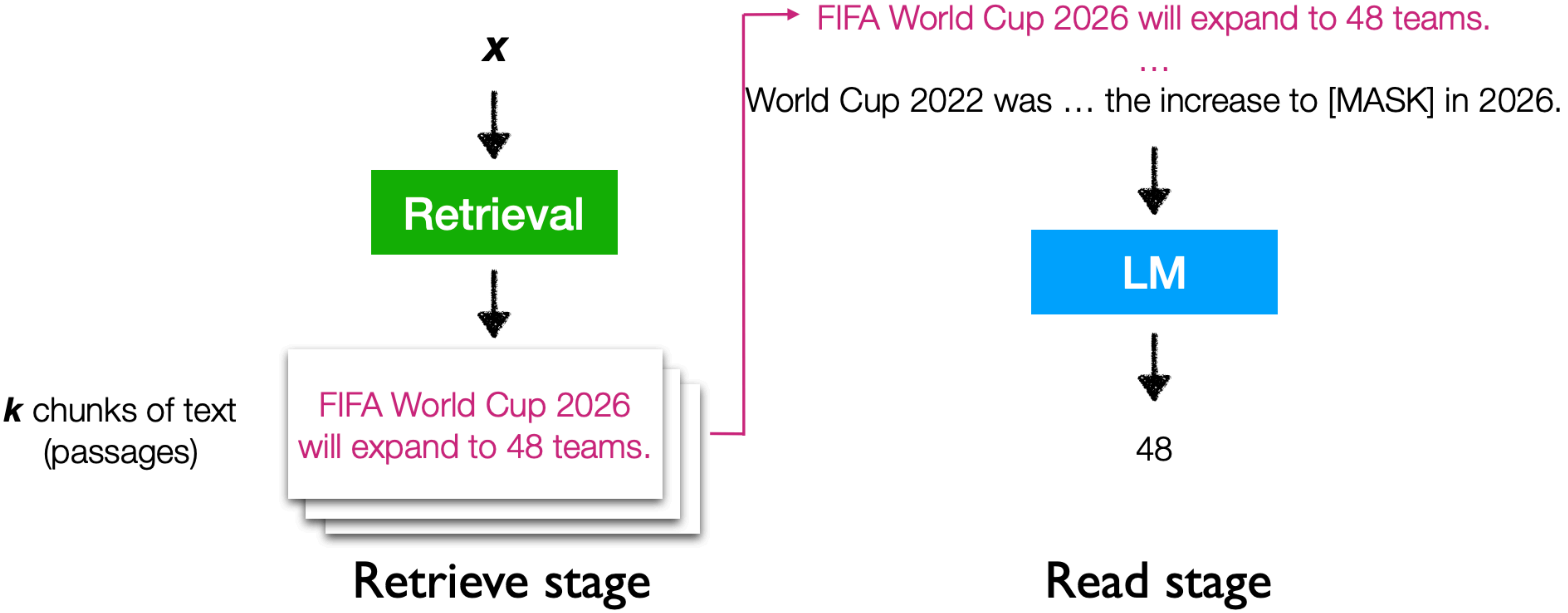


REALM [Guu et al., 2020]

x = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.

REALM [Guu et al., 2020]

x = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.



REALM — Retrieval

FIFA World Cup 2026 will expand to 48 teams.

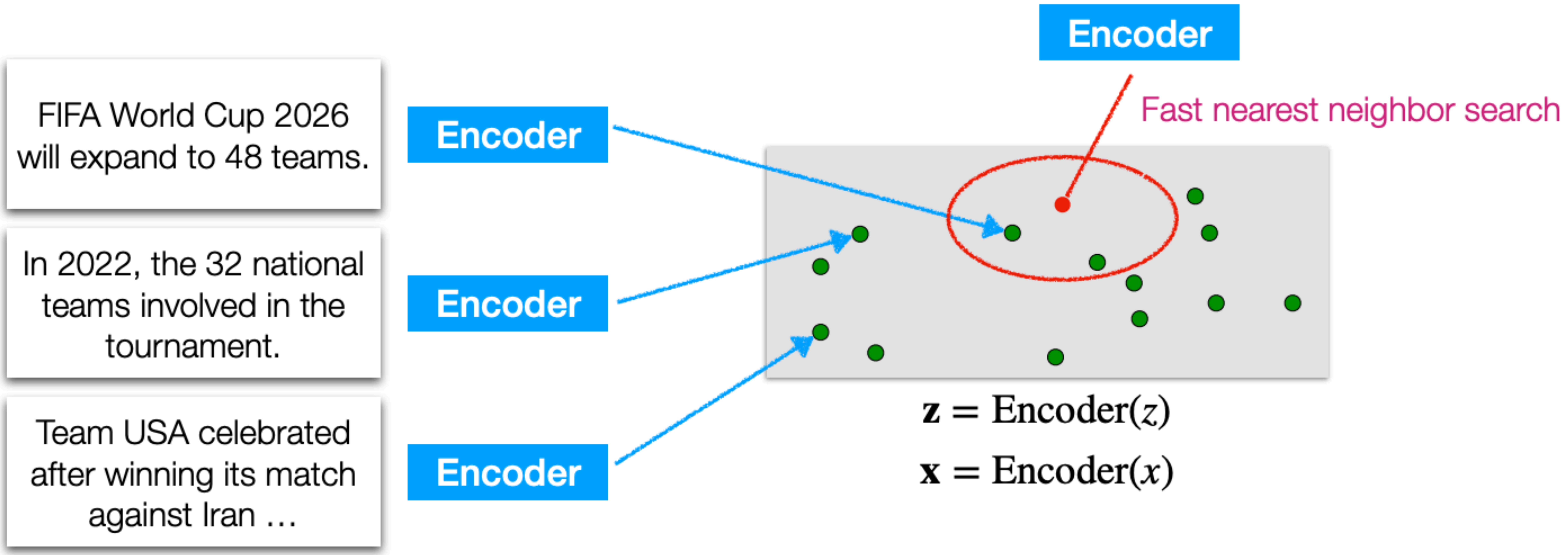
In 2022, the 32 national teams involved in the tournament.

Team USA celebrated after winning its match against Iran …

Wikipedia
13M chunks (passages)
(called *documents* in the paper)

REALM — Retrieval

$\boldsymbol{x}$ = World Cup 2022 was … the increase to [MASK] in 2026.

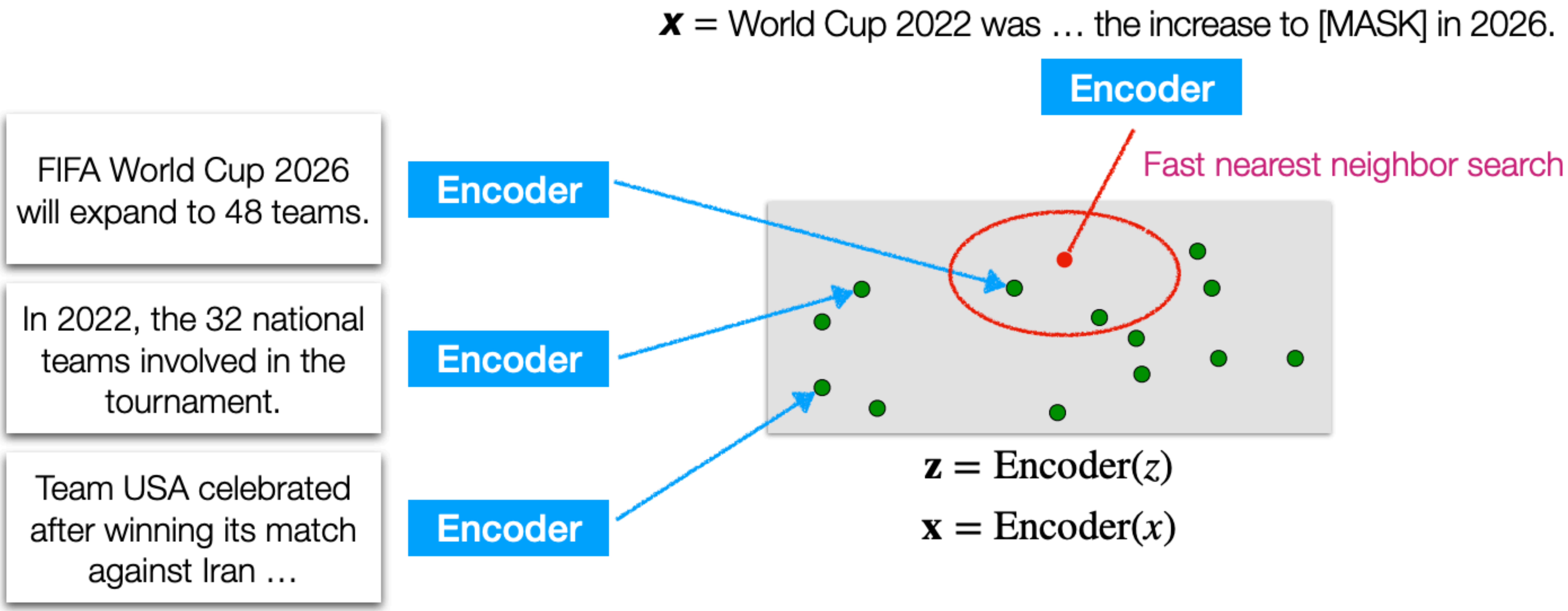


$$\mathbf{z} = \text{Encoder}(z)$$

$$\mathbf{x} = \text{Encoder}(x)$$

Wikipedia
13M chunks (passages)
(called *documents* in the paper)

REALM — Retrieval



$$\mathbf{z} = \text{Encoder}(z)$$

$$\mathbf{x} = \text{Encoder}(x)$$

Wikipedia
13M chunks (passages)
(called *documents* in the paper)

$$z_1, \ldots, z_k = \text{argTop-}k\,(\mathbf{x} \cdot \mathbf{z})$$

k retrieved chunks

REALM — Reading

[MASK] z_1 [SEP] x $\rightarrow$ LM $\rightarrow$ $P(y \mid x, z_1)$

[MASK] z_2 [SEP] x $\rightarrow$ LM $\rightarrow$ $P(y \mid x, z_2)$

$\vdots$

[MASK] z_k [SEP] x $\rightarrow$ LM $\rightarrow$ $P(y \mid x, z_k)$

REALM — Reading

[MASK] z_1 [SEP] x → LM → $P(y \mid x, z_1)$

[MASK] z_2 [SEP] x → LM → $P(y \mid x, z_2)$

⋮

[MASK] z_k [SEP] x → LM → $P(y \mid x, z_k)$

Weighted average

0 if not one of top *k*

$$\sum_{z \in \mathcal{D}} \underbrace{P(z \mid x)}_{\text{from the retrieve stage}} \underbrace{P(y \mid x, z)}_{\text{from the read stage}}$$

Need to approximate
→ Consider top *k* chunks only

REALM [Guu et al., 2020]

What to retrieve?

- **Chunks** ✓
- Tokens
- Others

REALM [Guu et al., 2020]

What to retrieve?

- **Chunks** ✓
- Tokens
- Others

How to use retrieval?

- **Input layer** ✓
- Intermediate layers
- Output layer

REALM [Guu et al., 2020]

What to retrieve?

- **Chunks** ✓
- Tokens
- Others

How to use retrieval?

- **Input layer** ✓
- Intermediate layers
- Output layer

When to retrieve?

- **Once** ✓
- Every *n* tokens (*n*>1)
- Every token

Overview — Retrieval-Augmented Models

REALM [Guu et al., 2020] — Masked Language Modeling (MLM) pre-training objective followed by fine-tuning, focusing on ODQA

DPR [Karpukhin et al., 2020] — pipeline training rather than join training, focusing on ODQA with no explicit LM training objective

RAG [Lewis et al., 2020] — Generative training objective rather than MLM, focusing on ODQA and knowledge-intensive tasks (no explicit LM objective)

ATLAS [Izacard et al., 2022] — Combine RAG with a retrieval-based LM pre-training objective and a encoder-decoder architecture, focusing on ODQA and knowledge-intensive tasks

Reading List

Reliable, Adaptable, and Attributable Language Models with Retrieval,

https://arxiv.org/abs/2403.03187

ATLAS: Few-shot Learning with Retrieval Augmented Language Models, **https://arxiv.org/abs/2208.03299**

REALM: Retrieval-Augmented Language Model Pre-Training,

https://arxiv.org/abs/2002.08909

Reading Wikipedia to Answer Open-Domain Questions,

https://arxiv.org/abs/1704.00051

Question and Answer Test-Train Overlap in Open-Domain Question Answering Datasets, **https://arxiv.org/abs/2008.02637**

Challenges in Generalisation in Open Domain Question Answering, **https://arxiv.org/abs/2109.01156**