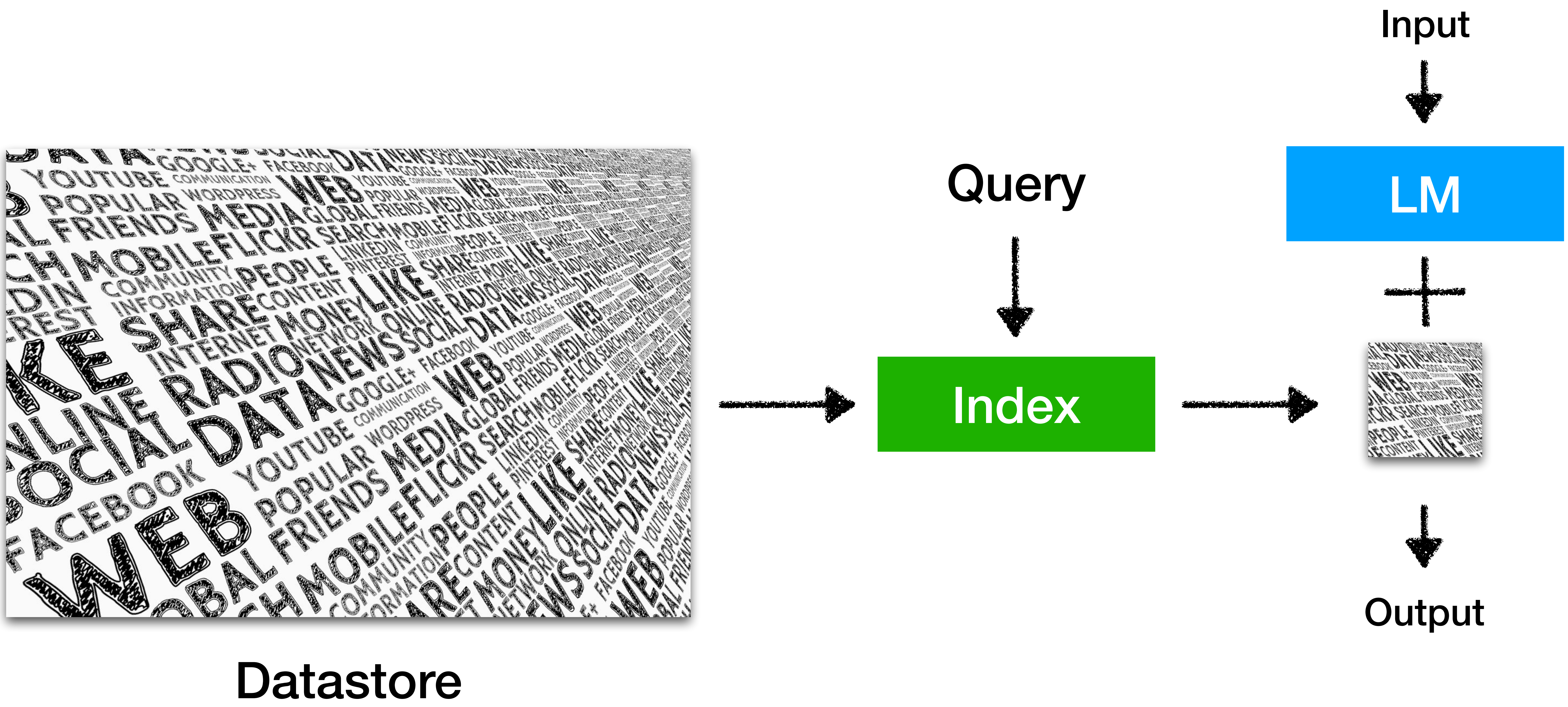
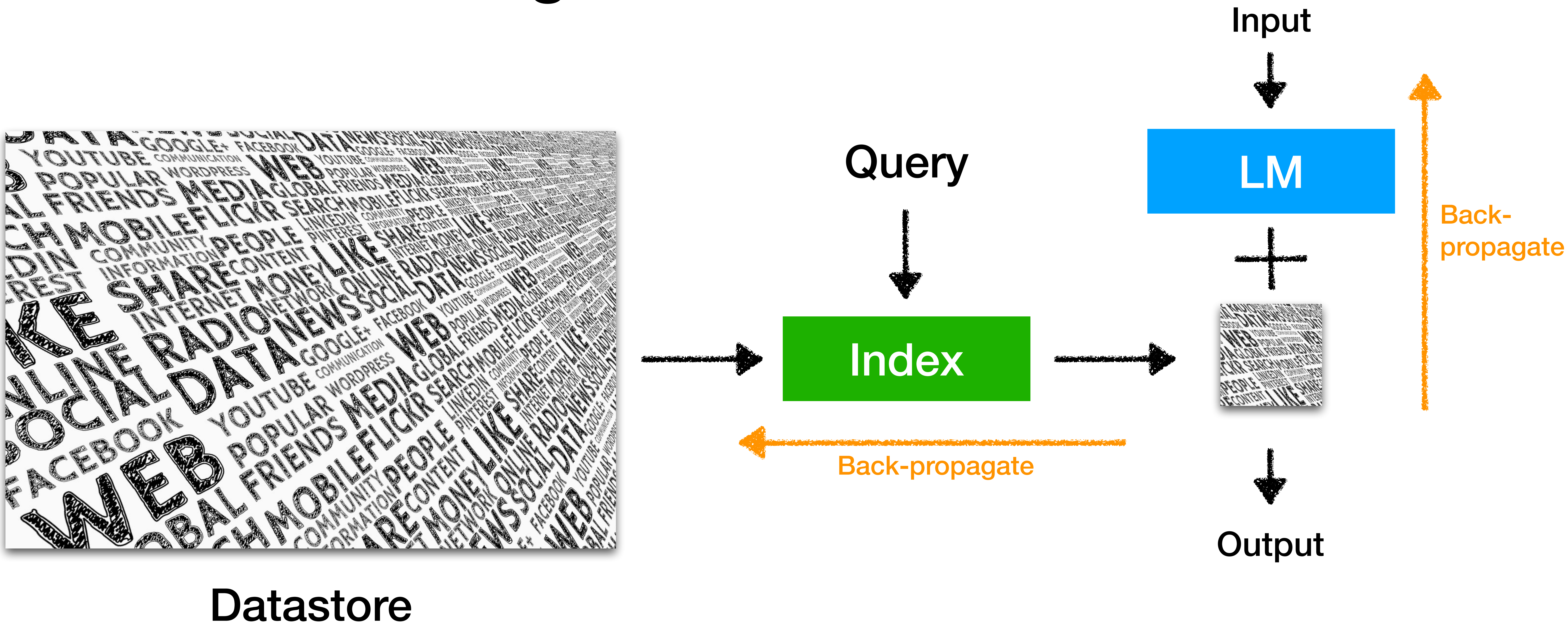
Section 4:
Retrieval-based LMs: Training

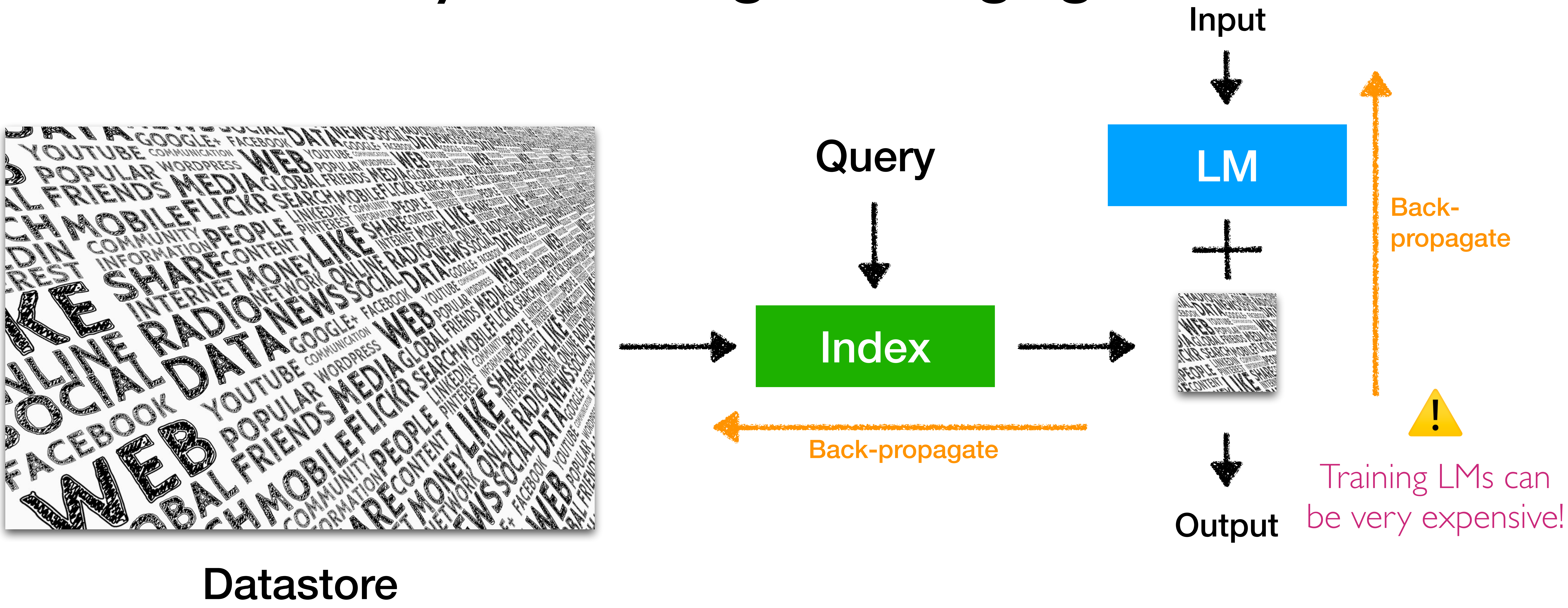
Retrieval-based LMs



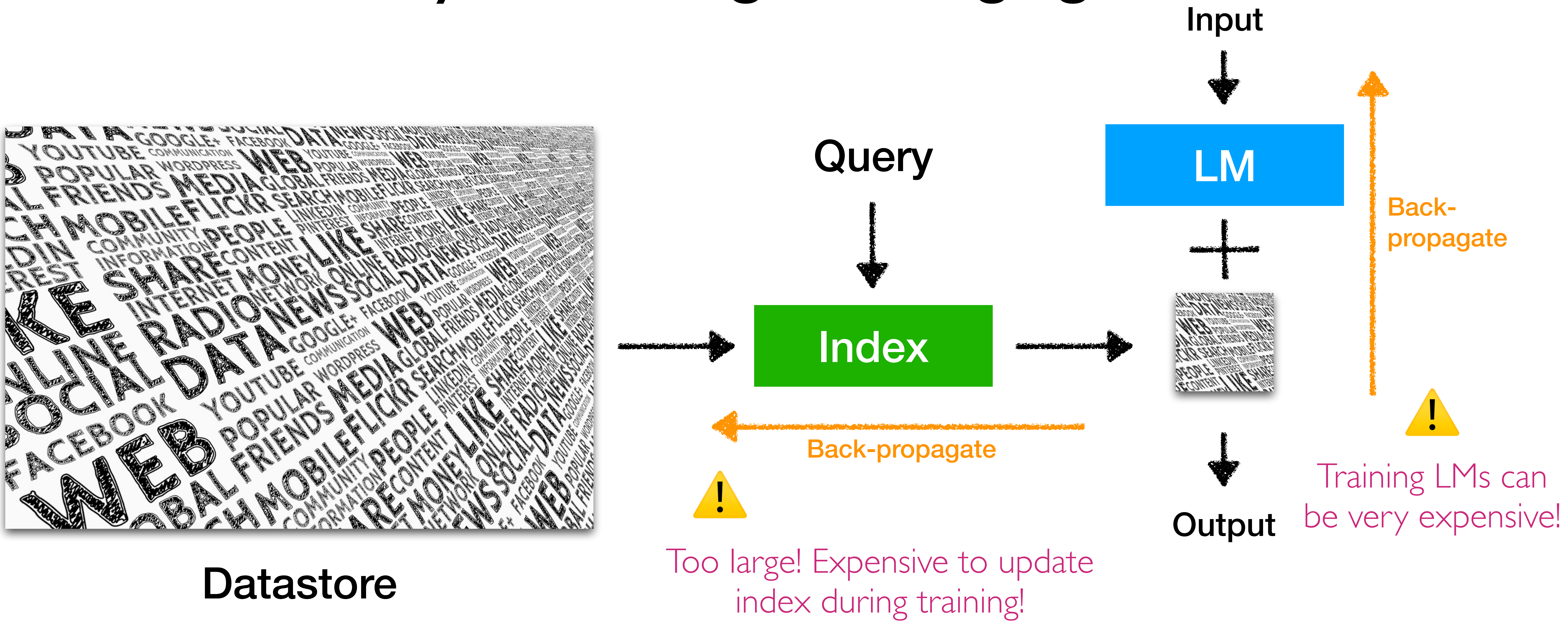
Training retrieval-based LMs



Why is training challenging?

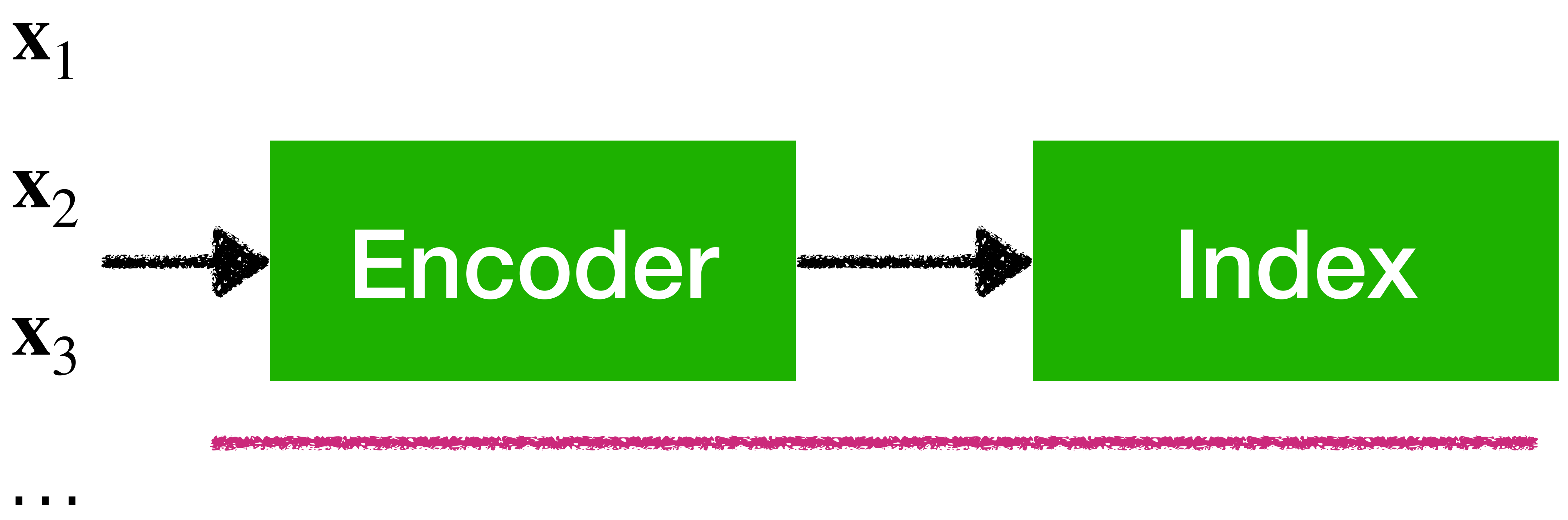


Why is training challenging?



Challenges of updating retrieval models

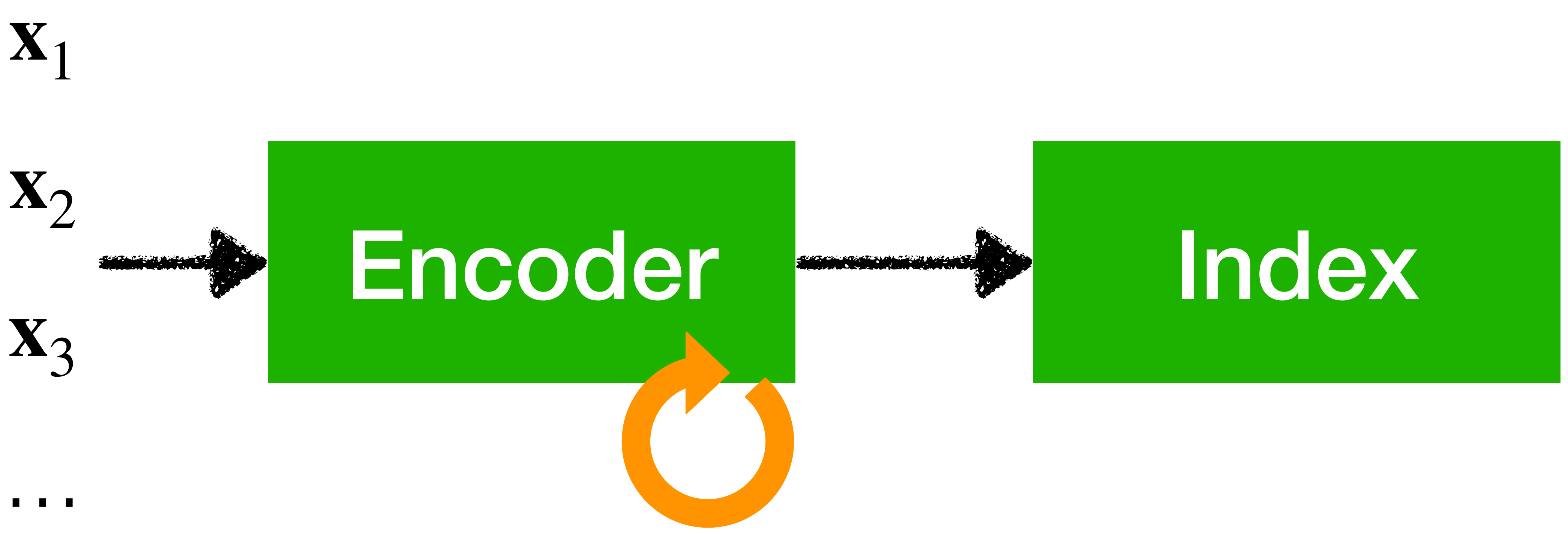
Datastore



We may encode a lot of (>1M) text chunks using the encoder!

Challenges of updating retrieval models

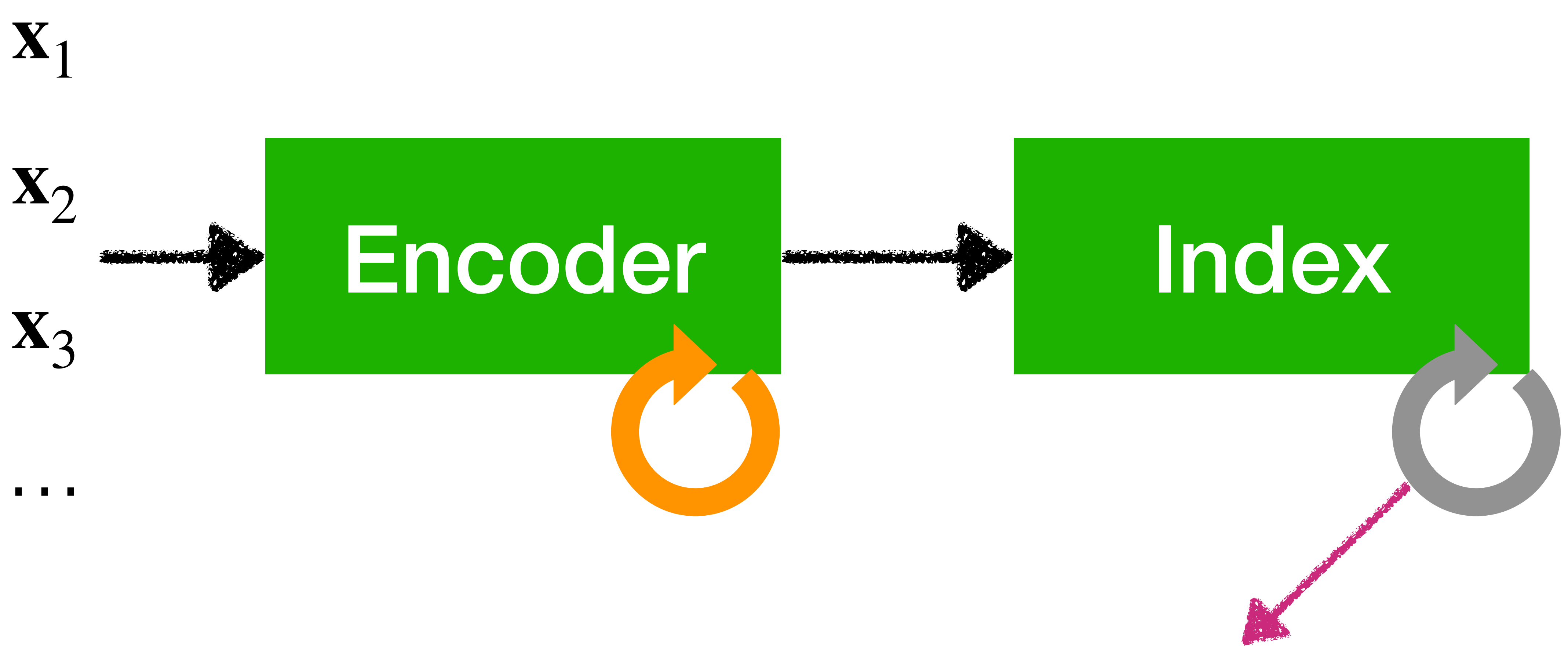
Datastore



During training, we will update the encoder

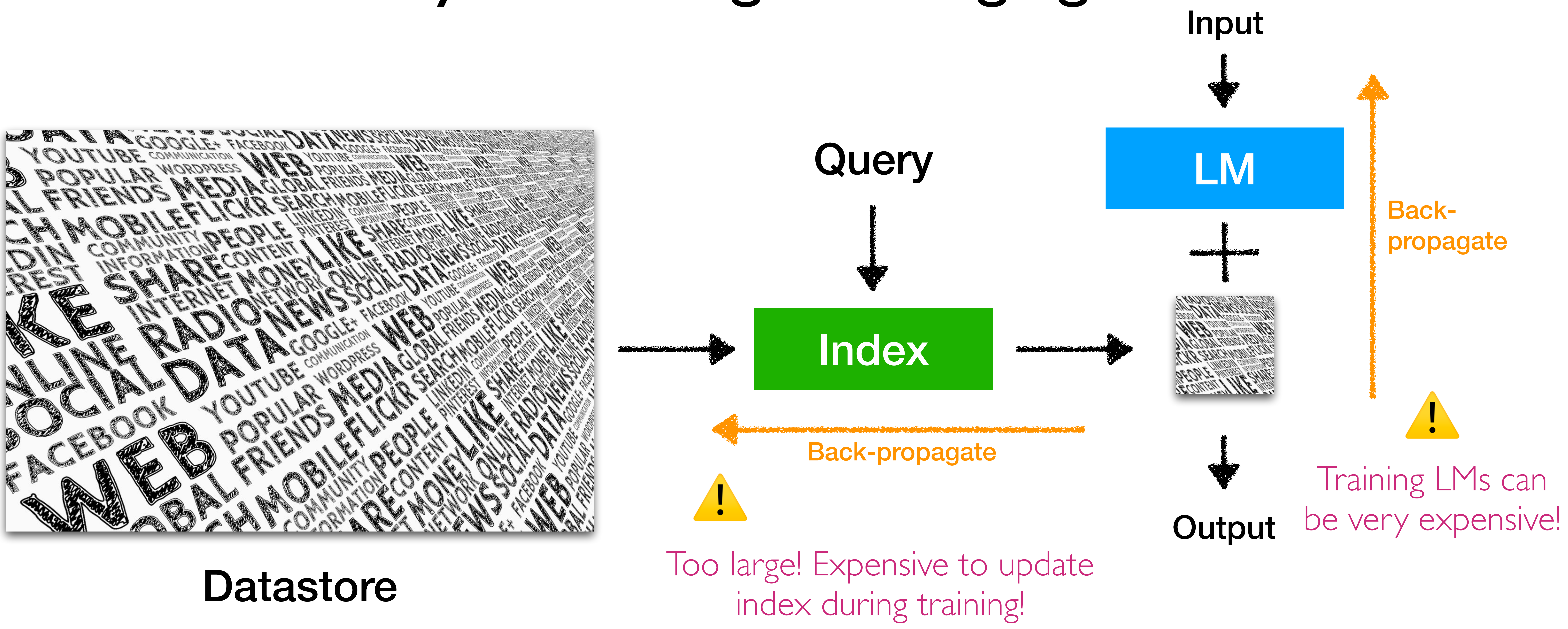
Challenges of updating retrieval models

Datastore



Re-indexing will be very expensive!

Why is training challenging?



Training methods for retrieval-based LMs

- Independent training
- Sequential training
- Joint training w/ asynchronous index update
- Joint training w/ in-batch approximation

Training methods for retrieval-based LMs

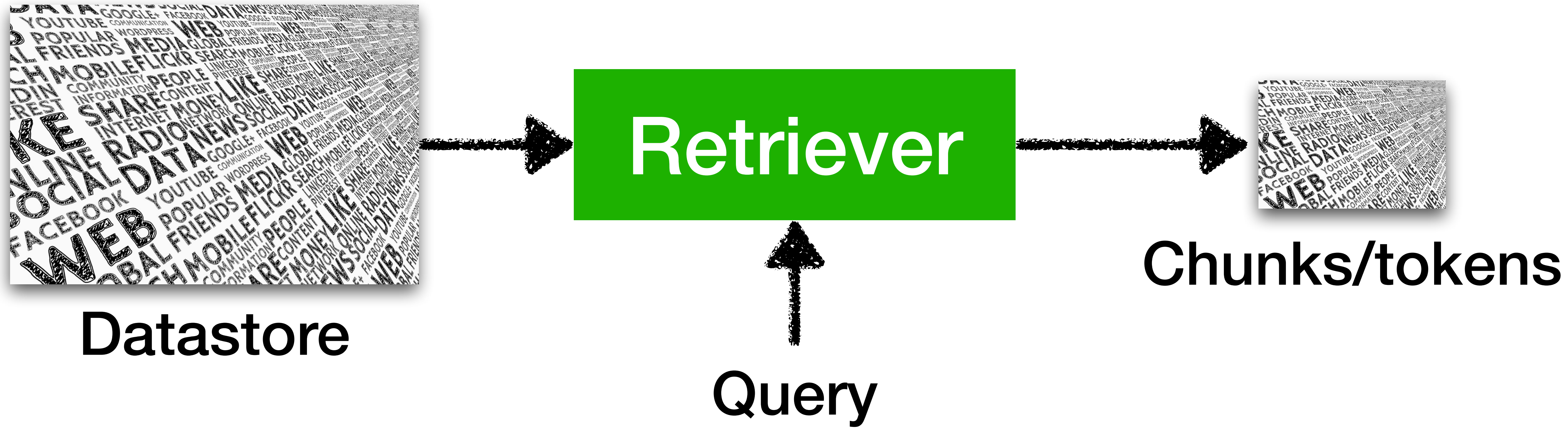
- **Independent training**
- Sequential training
- Joint training w/ asynchronous index update
- Joint training w/ in-batch approximation

Independent training

Retrieval models and language models are trained **independently**

- Training language models

- Training retrieval models



Independent training

Retrieval models and language models are trained **independently**

- Training language models

Input → LM → Output

- Training retrieval models

Datastore → Retriever →  Chunks/tokens

Query → Retriever

Training language models

Minimize $\quad -\log P_{\mathrm{LM}}(y \mid x)$

Training language models

Minimize $\quad -\log P_{\text{LM}}(y \mid x)$

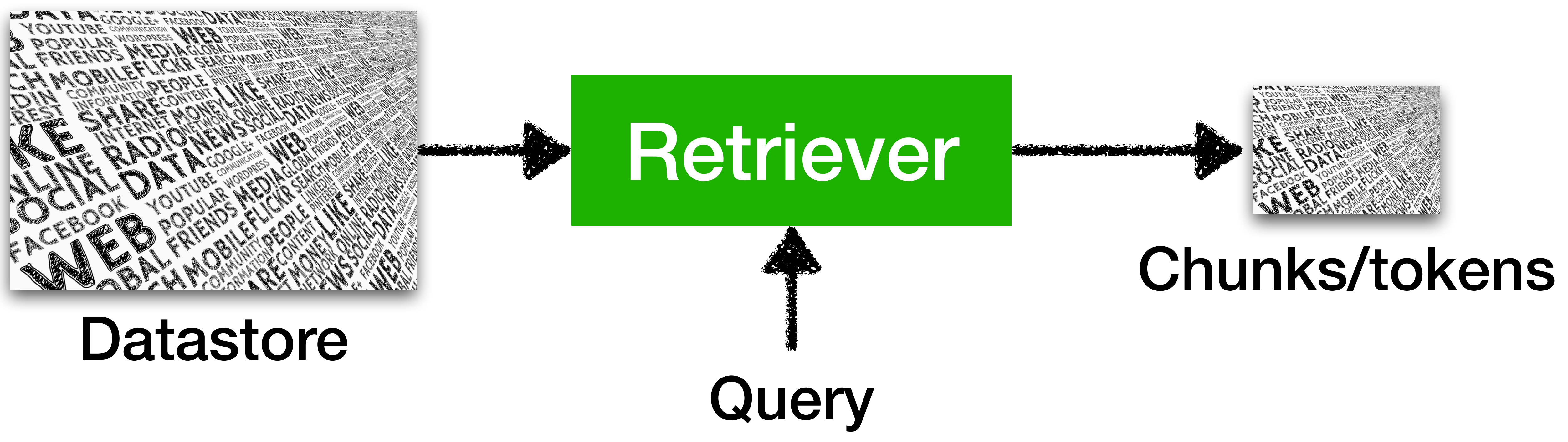
GPT PaLM LLaMA GPT-J ……

Independent training

Retrieval models and language models are trained **independently**

- Training language models

- Training retrieval models



Sparse retrieval models: TF-IDF / BM25

In 1997, Apple merged with NeXT, and Steve Jobs became CEO of …

Jobs returned to Apple as CEO after the company's acquisition …

Text chunks

[0, 0, 0.4, 0, 0.8, 0.7, …]

[0, 1.2, 0.4, 0, 0.8, 0, …]

Sparse vectors

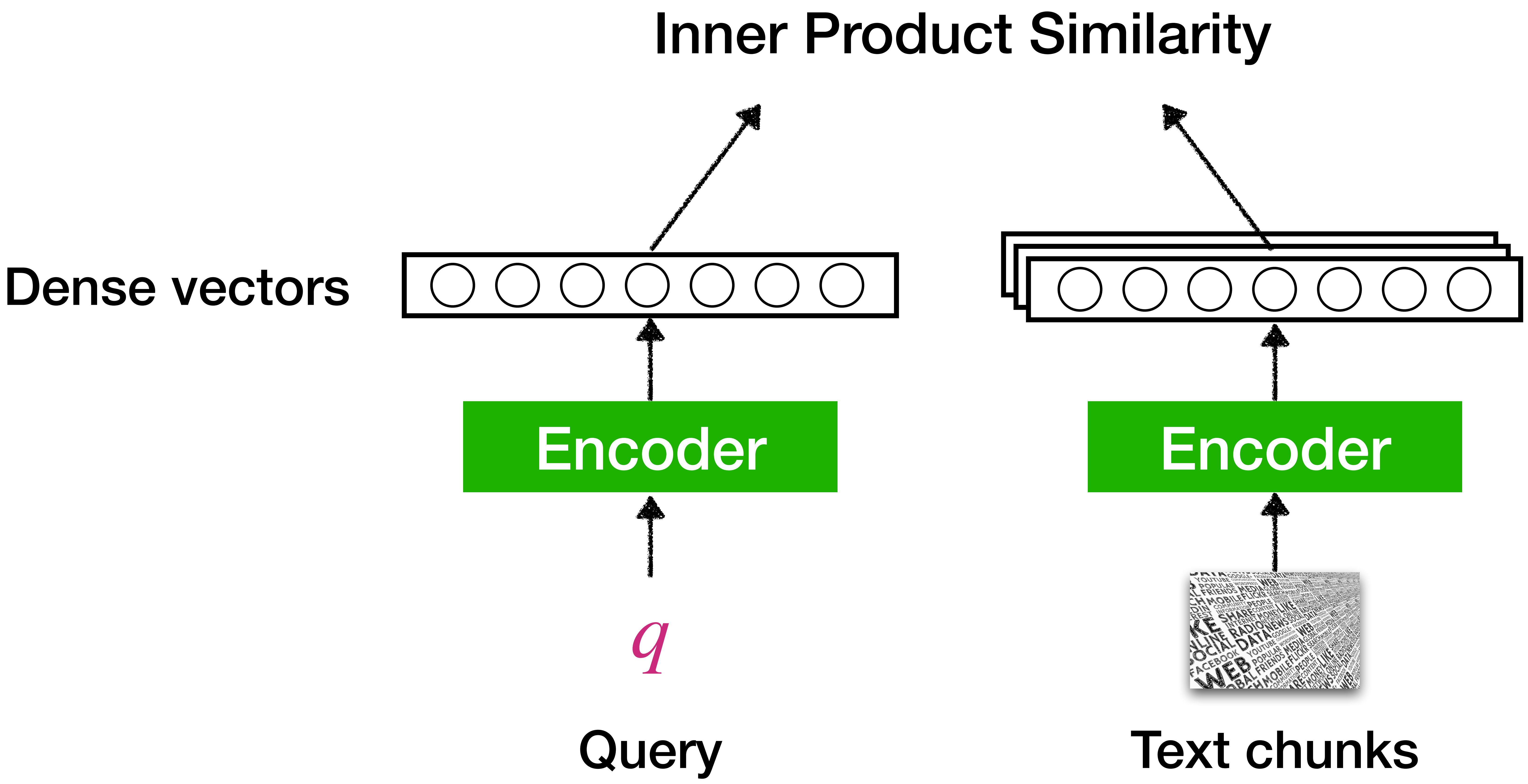
Lexical overlap

No training needed!

Ramos, 2003. "Using TF-IDF to Determine Word Relevance in Document Queries"

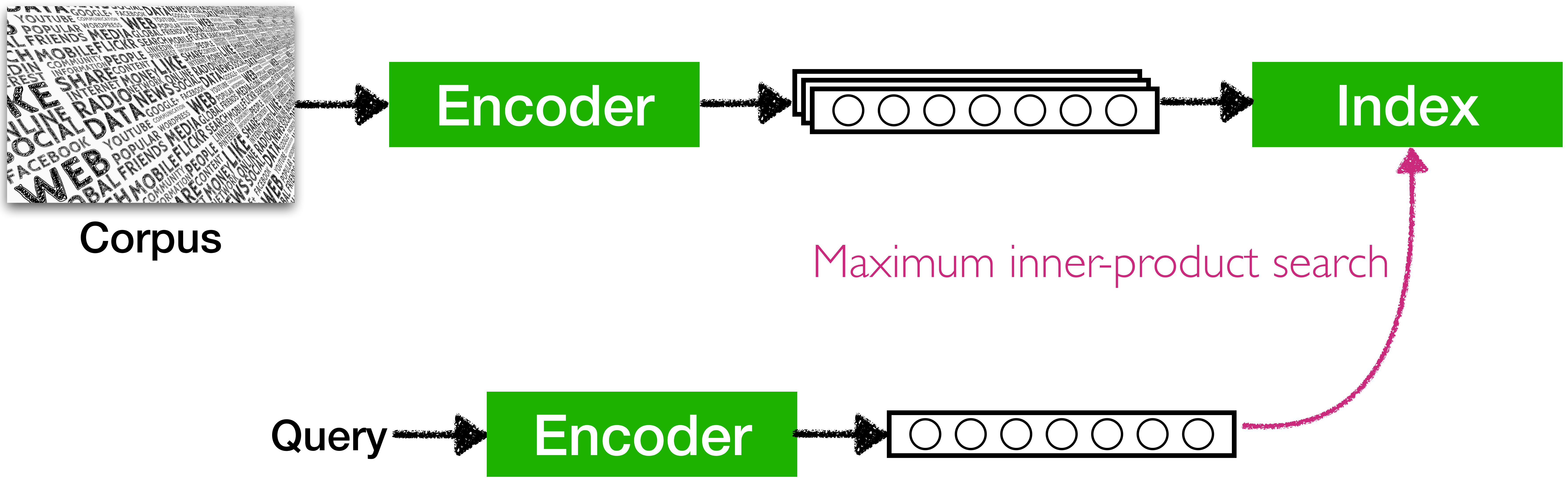
Robertson and Zaragoza, 2009. "The Probabilistic Relevance Framework: BM25 and Beyond"

Dense retrieval models: DPR (Karpukhin et al. 2020)

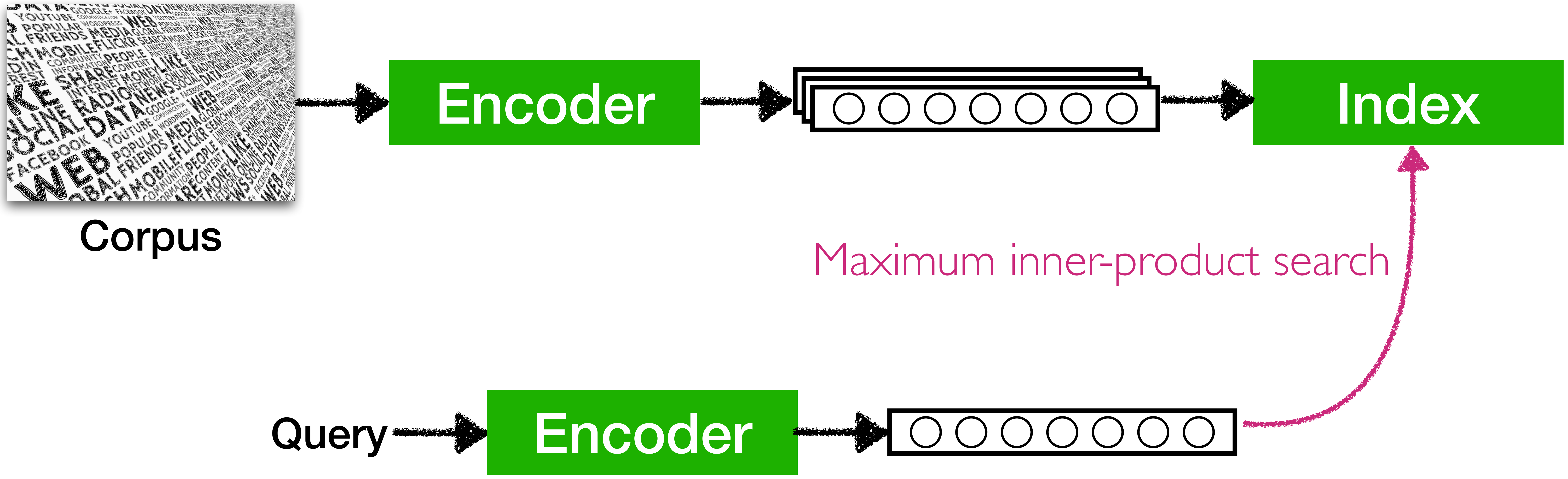


Karpukhin et al., 2020. “Dense Passage Retrieval for Open-Domain Question Answering”

Dense retriever: Inference



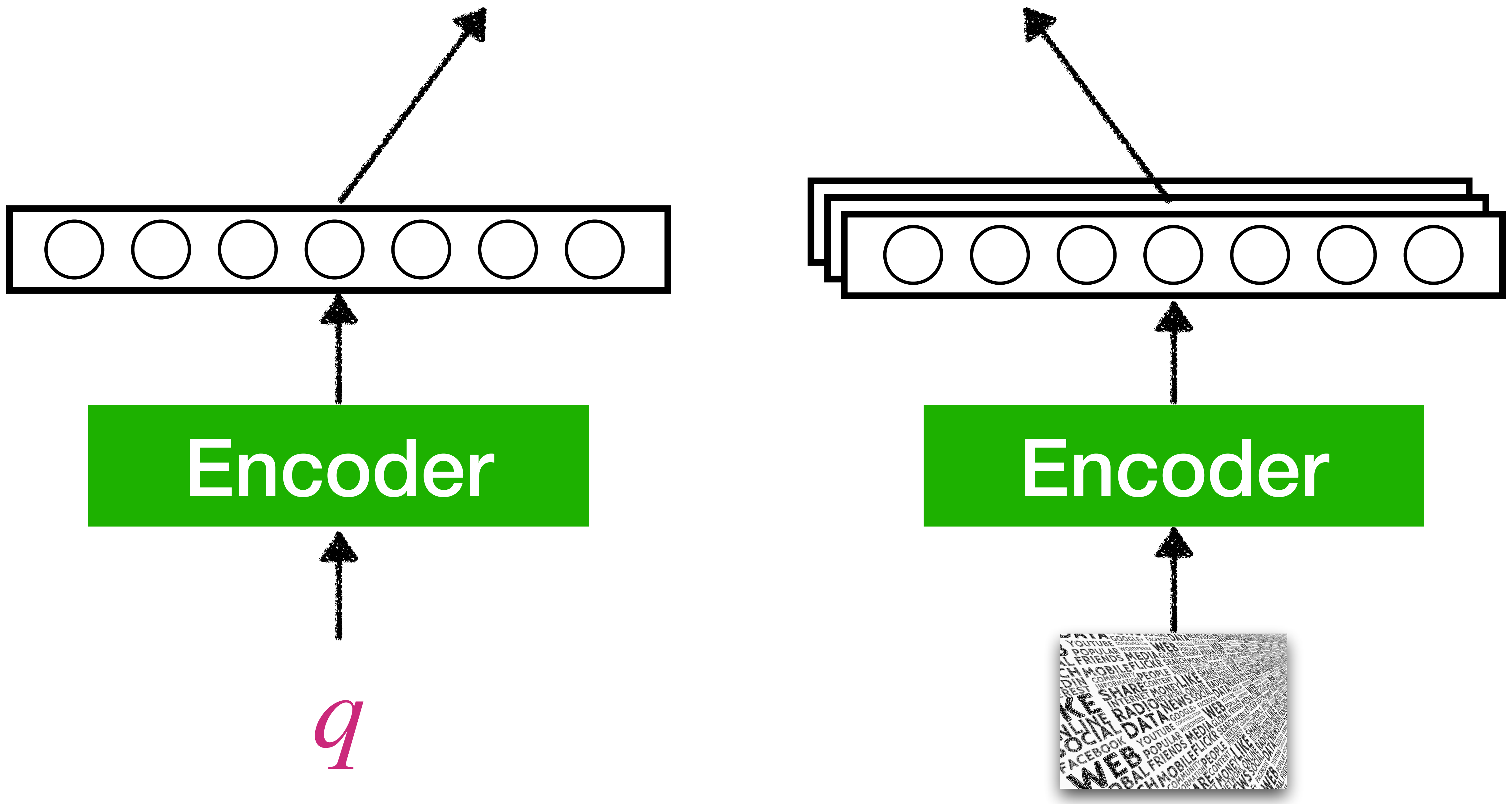
Dense retriever: Inference



How to train dense retrieval models?

Training dense retrieval models: DPR

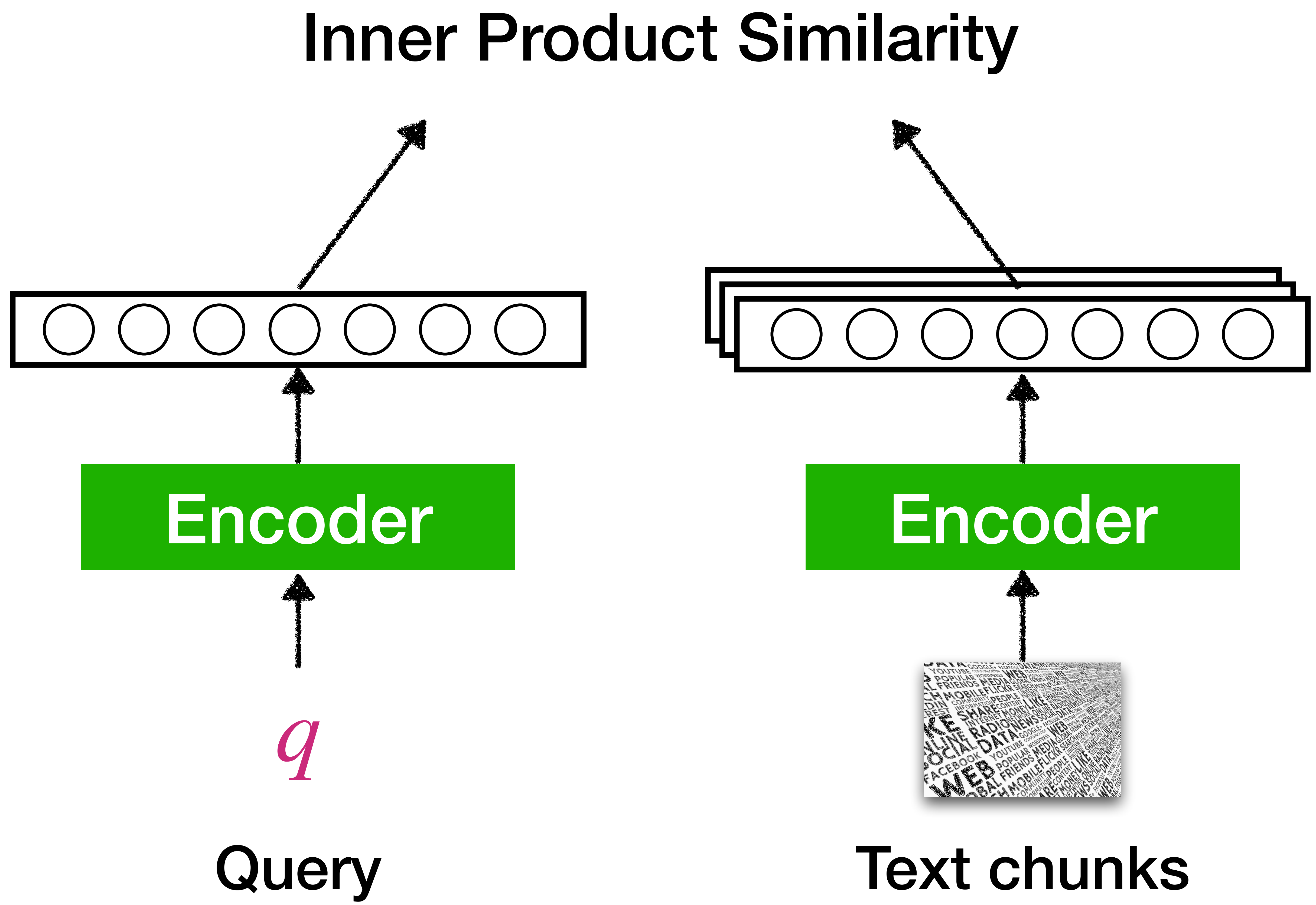
Inner Product Similarity



Query

Text chunks

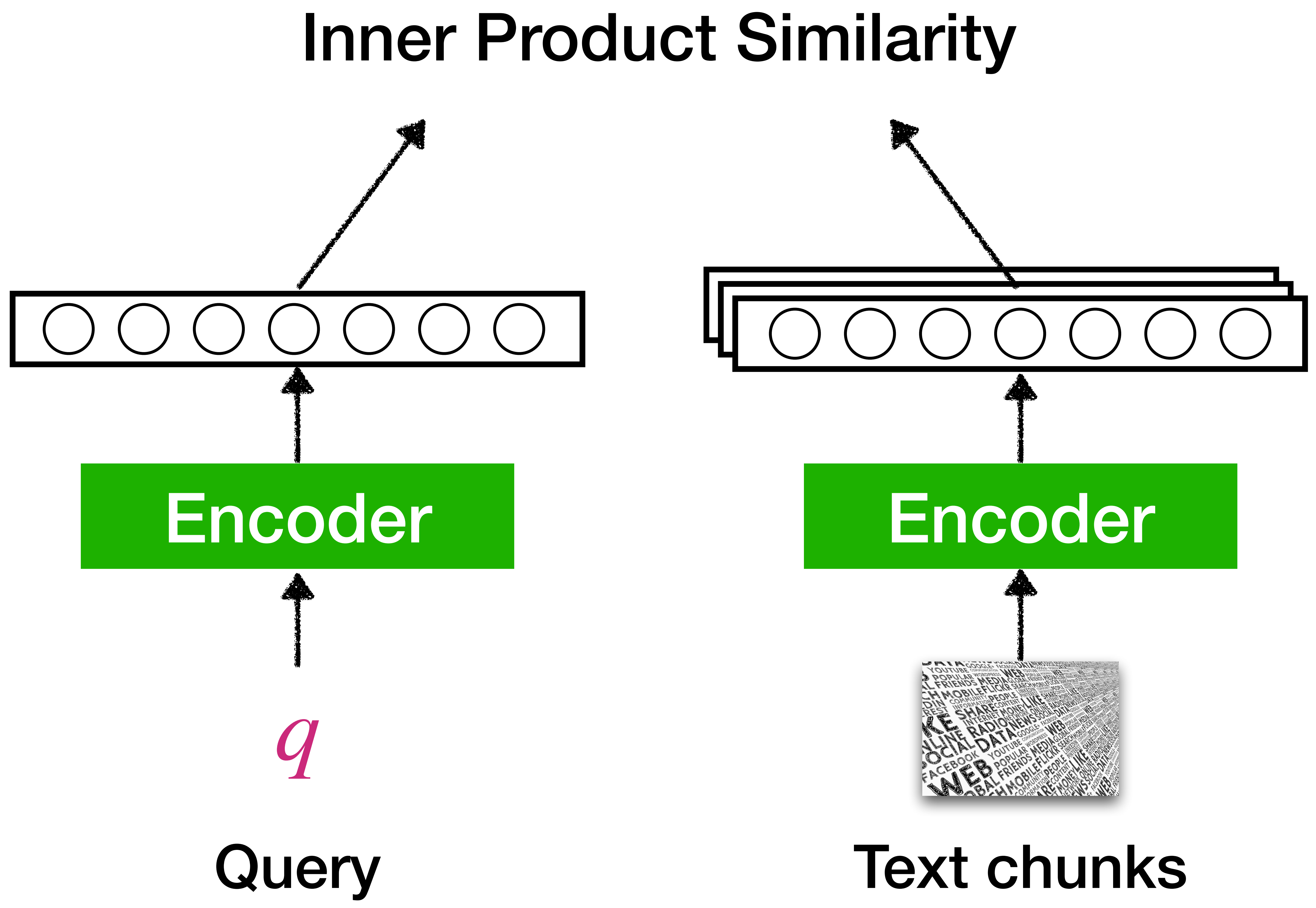
Training dense retrieval models: DPR



$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-)$$

$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$

Training dense retrieval models: DPR

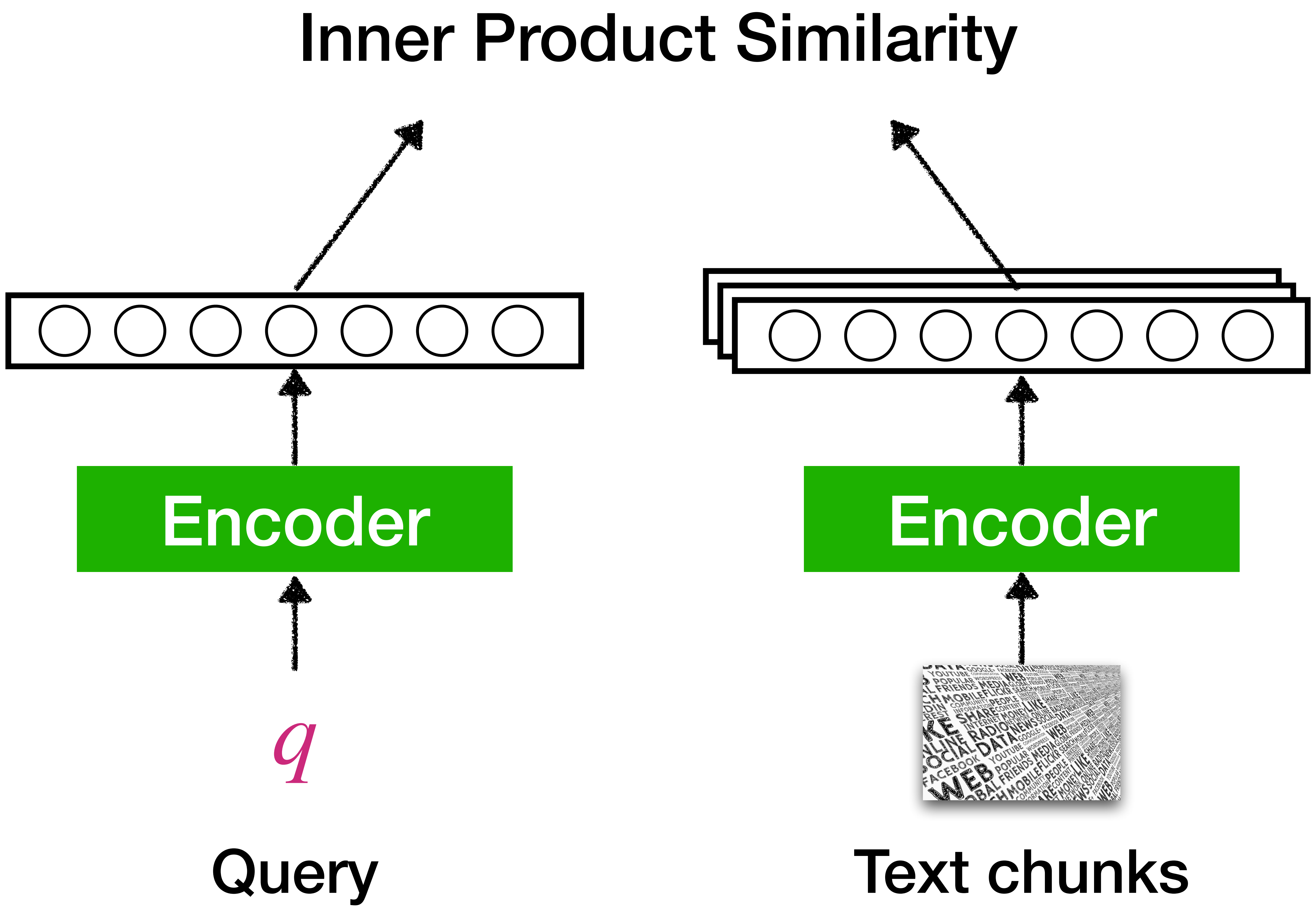


$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-)$$

$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$

Contrastive loss

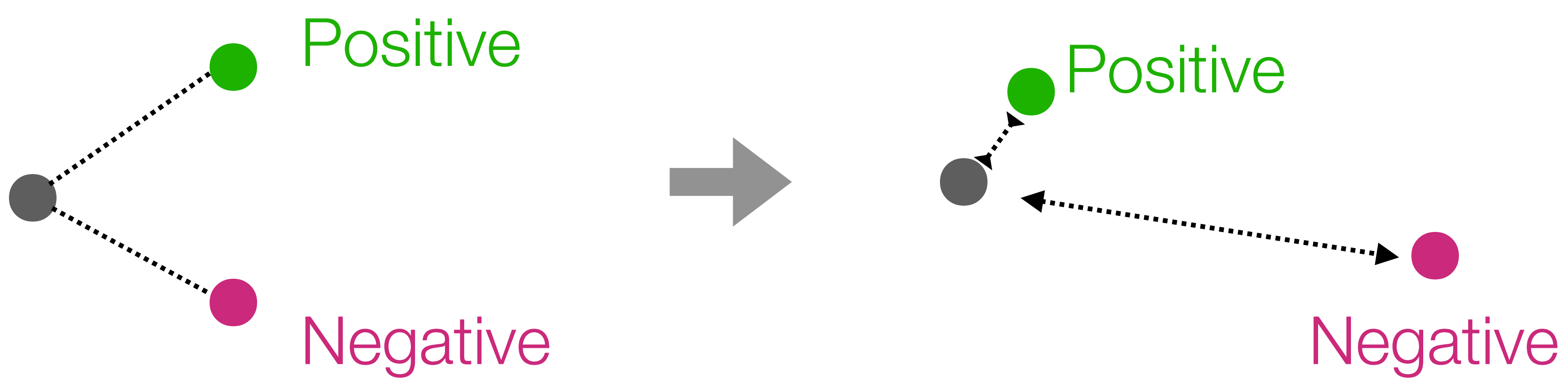
Training dense retrieval models: DPR



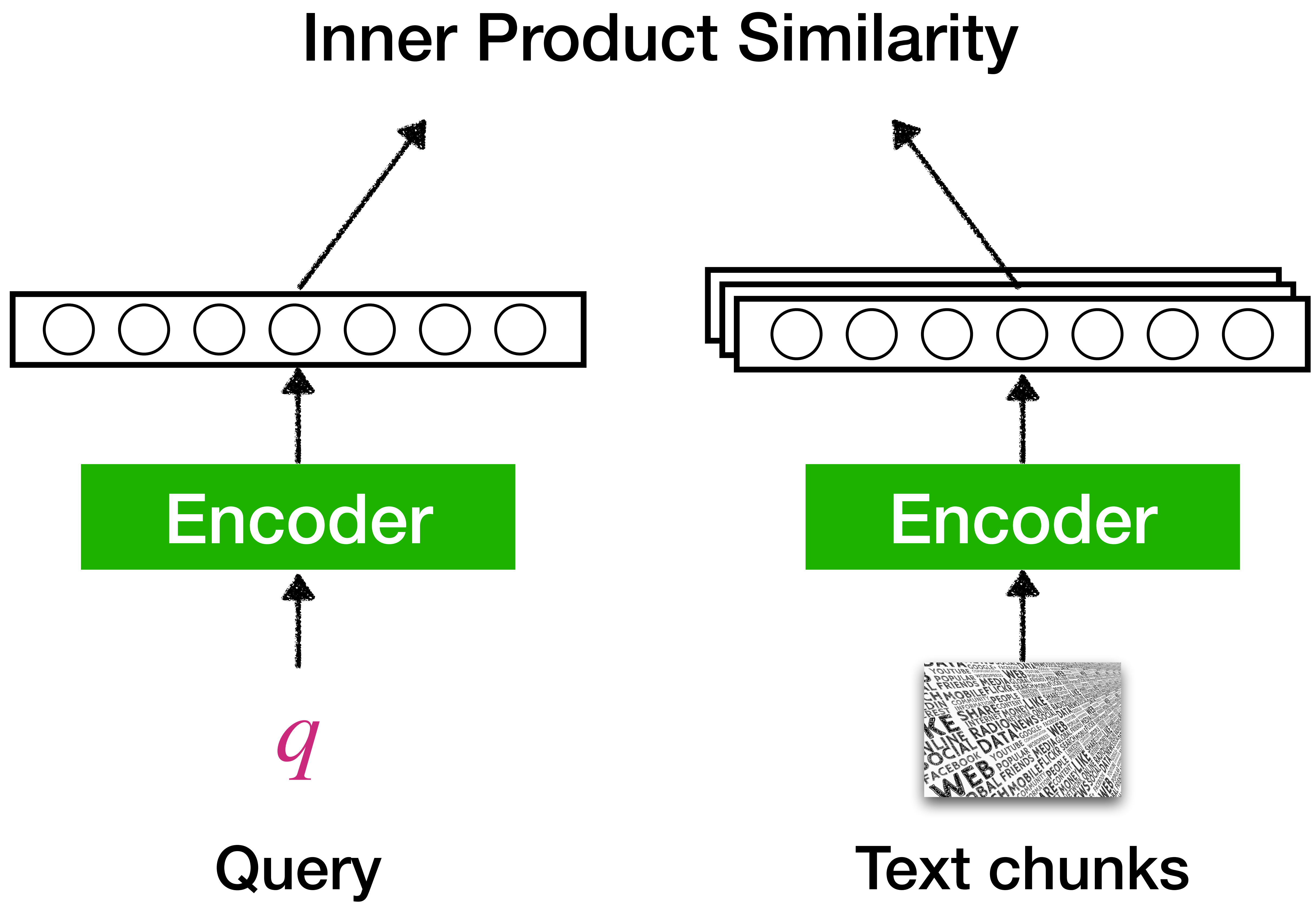
$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-)$$

$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$

Contrastive loss



Training dense retrieval models: DPR

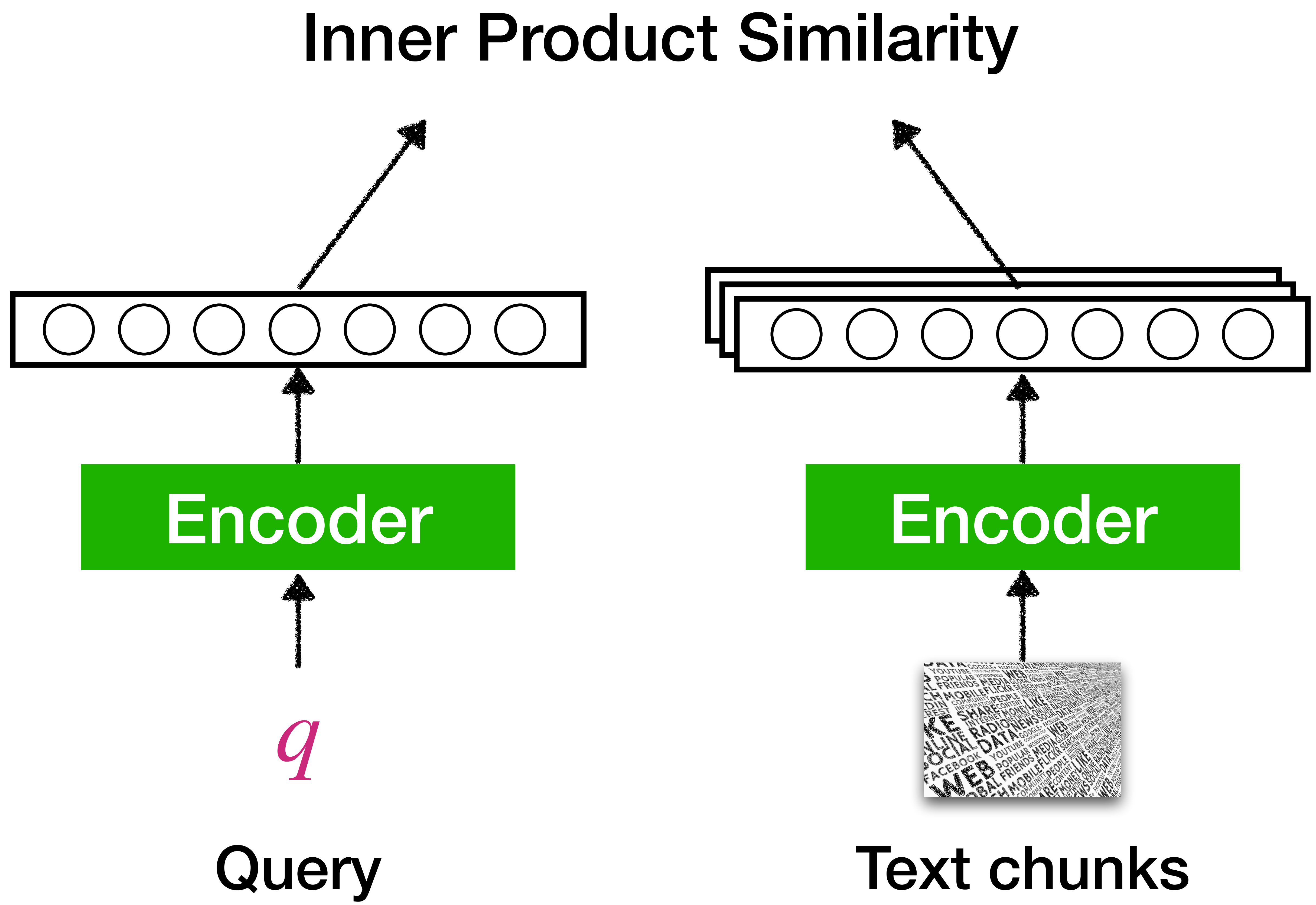


$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-)$$

Positive passage

$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$

Training dense retrieval models: DPR



Negative passages
Too expensive to consider all negatives!

$$L(q, p^+, p_1^-, p_2^-, \dots, p_n^-)$$

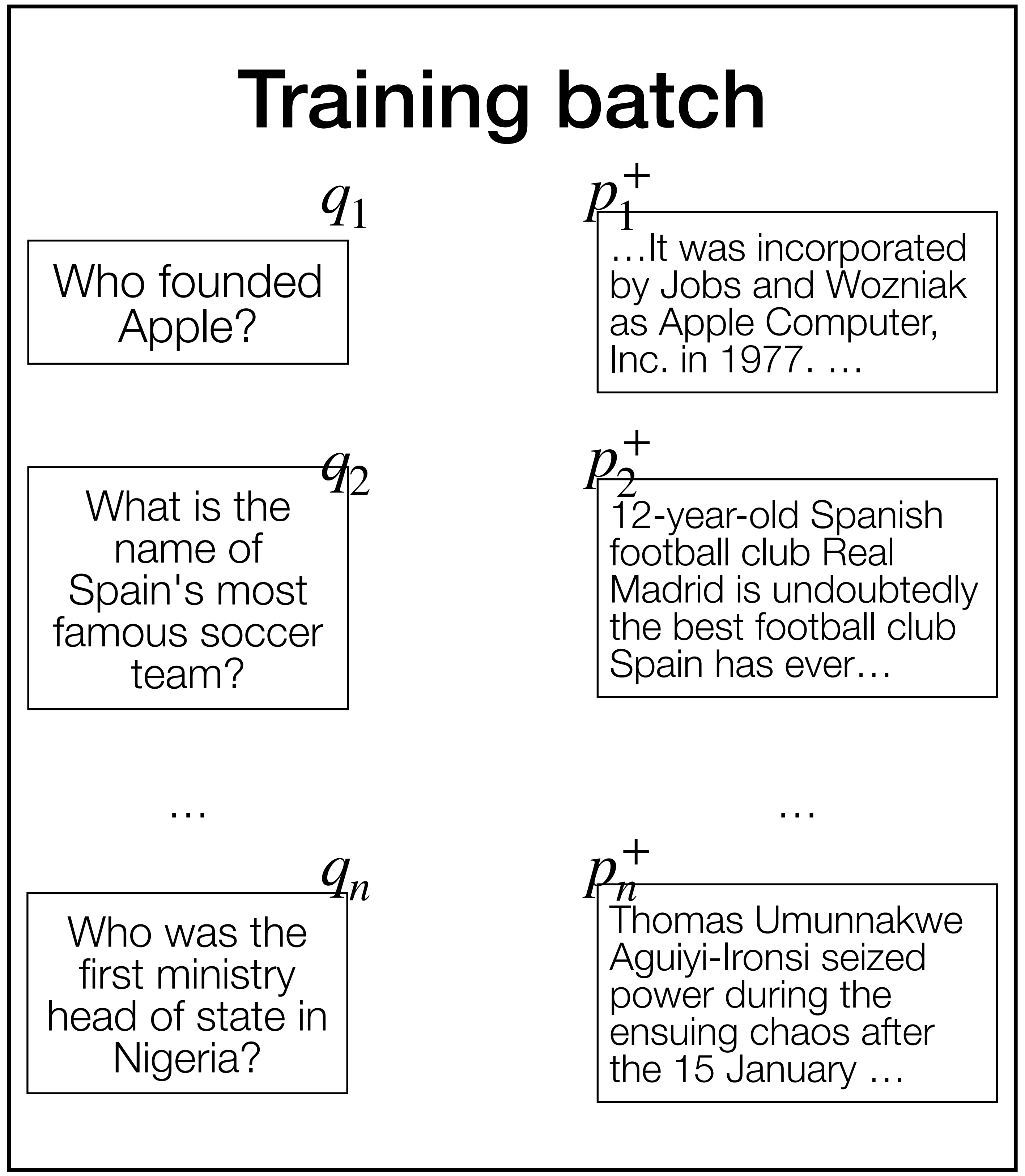
Positive passage

$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$

Training with “in-batch” negatives

$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-)$$

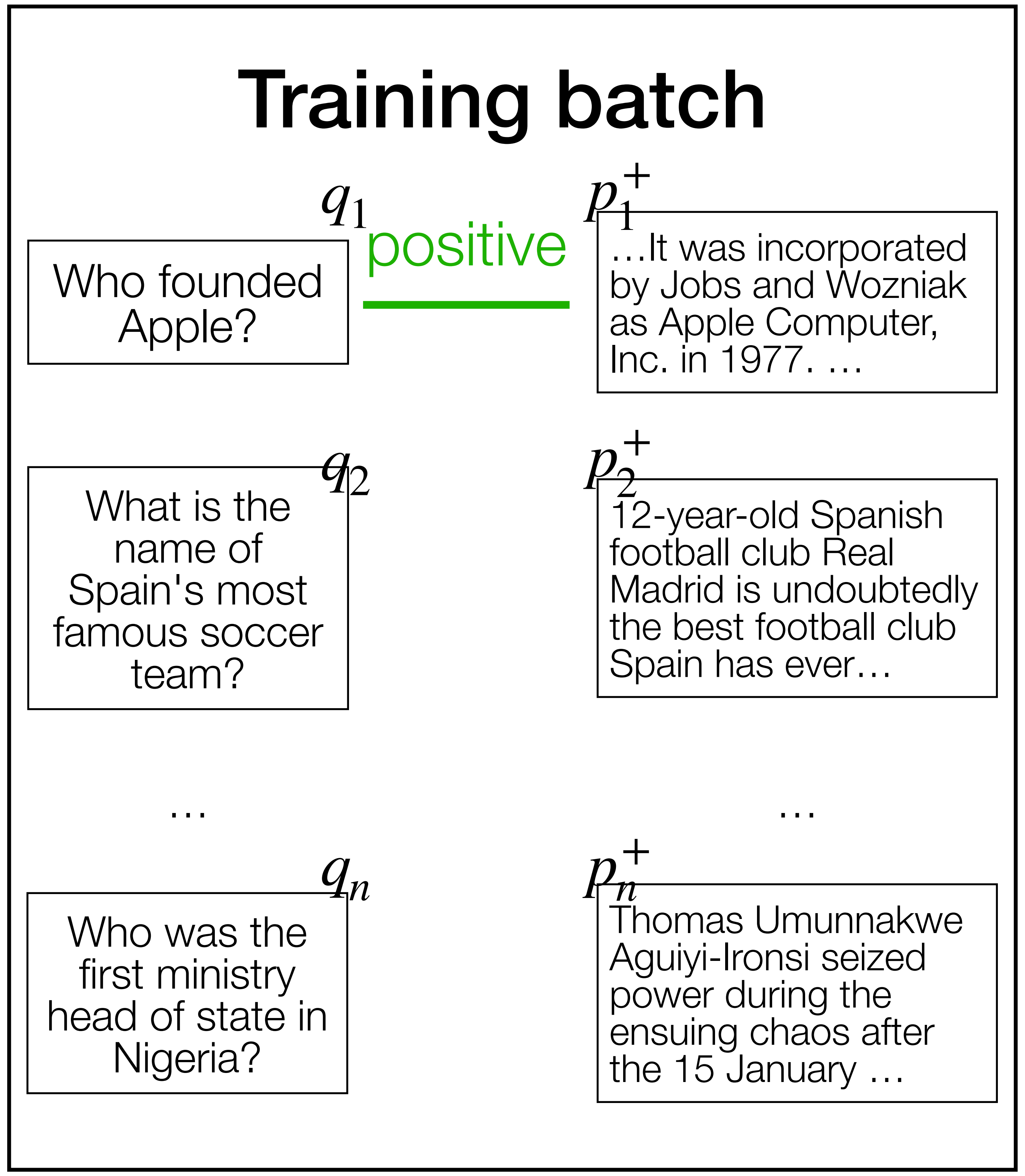
$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$



Training with "in-batch" negatives

$$L(q, \boxed{p^+}, p_1^-, p_2^-, \ldots, p_n^-)$$

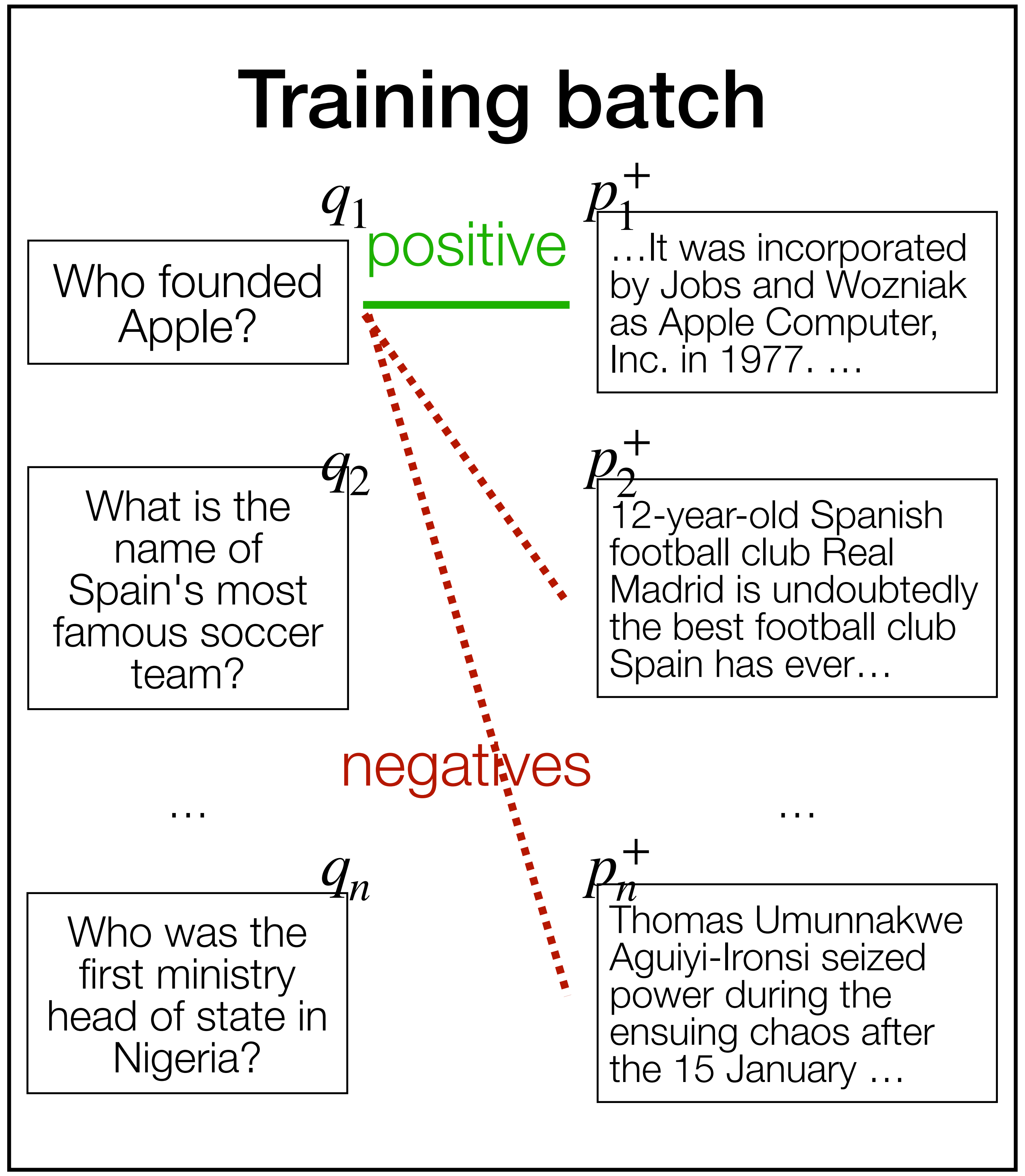
$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$



Training with "in-batch" negatives

$$L(q, \boxed{p^+}, \boxed{p_1^-, p_2^-, \ldots, p_n^-})$$

$$= -\log \frac{\exp(\mathrm{sim}(q, p^+))}{\exp(\mathrm{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\mathrm{sim}(q, p_j^-))}$$



Back-propagation to all in-batch negatives!

Contriever (Izacard et al. 2022)

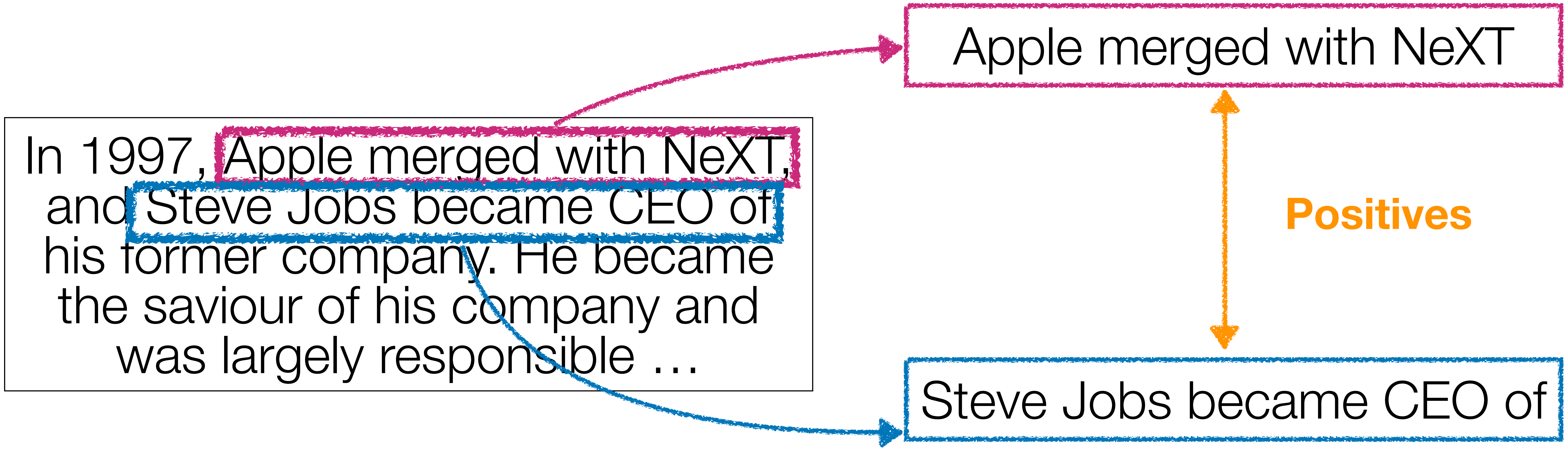
Unsupervised dense retrieval model!

- How to get **positive** samples in unlabeled text?
- How to use more **negative** samples during training?

Izacard et al., 2022. "Unsupervised Dense Information Retrieval with Contrastive Learning"

Contriever (Izacard et al. 2022)

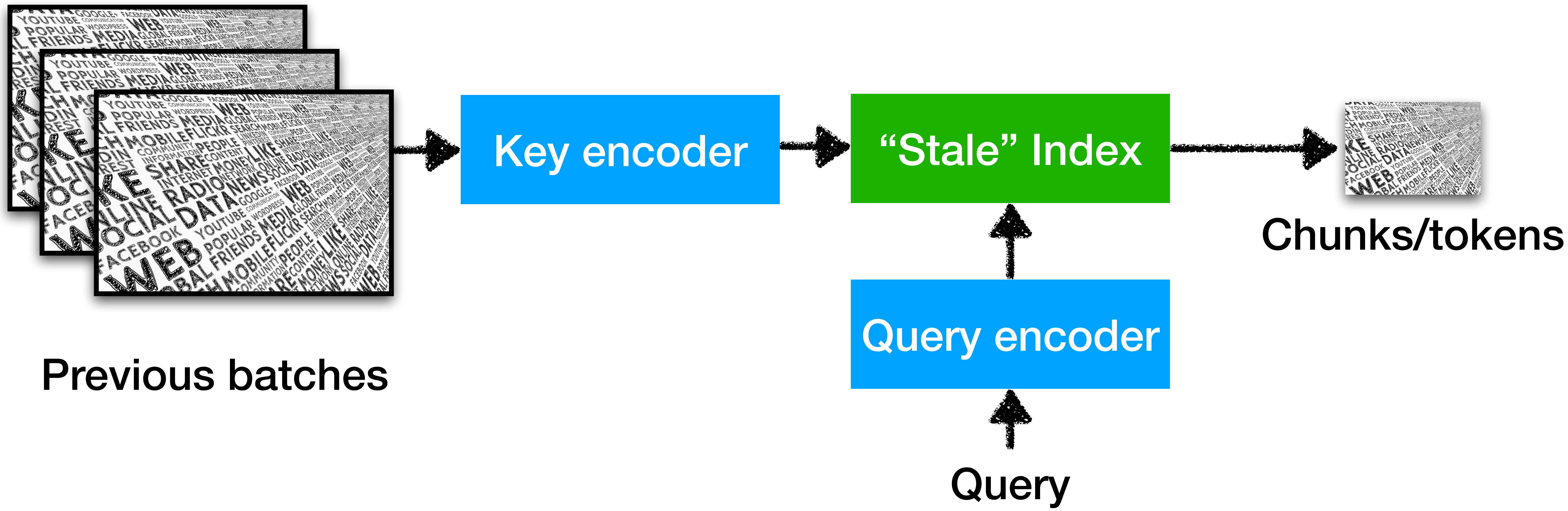
Independent Cropping



Positive pairs from unlabeled text!

Contriever (Izacard et al. 2022)

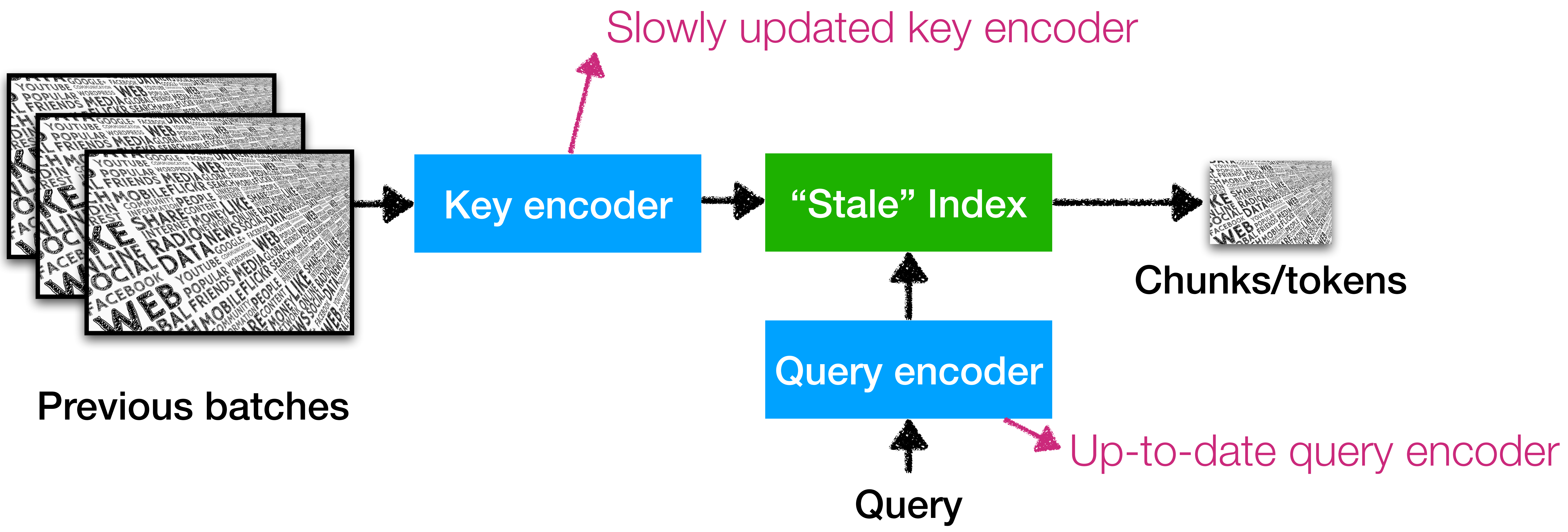
MoCo (Momentum Contrast)



He et al., 2020. "Momentum Contrast for Unsupervised Visual Representation Learning"

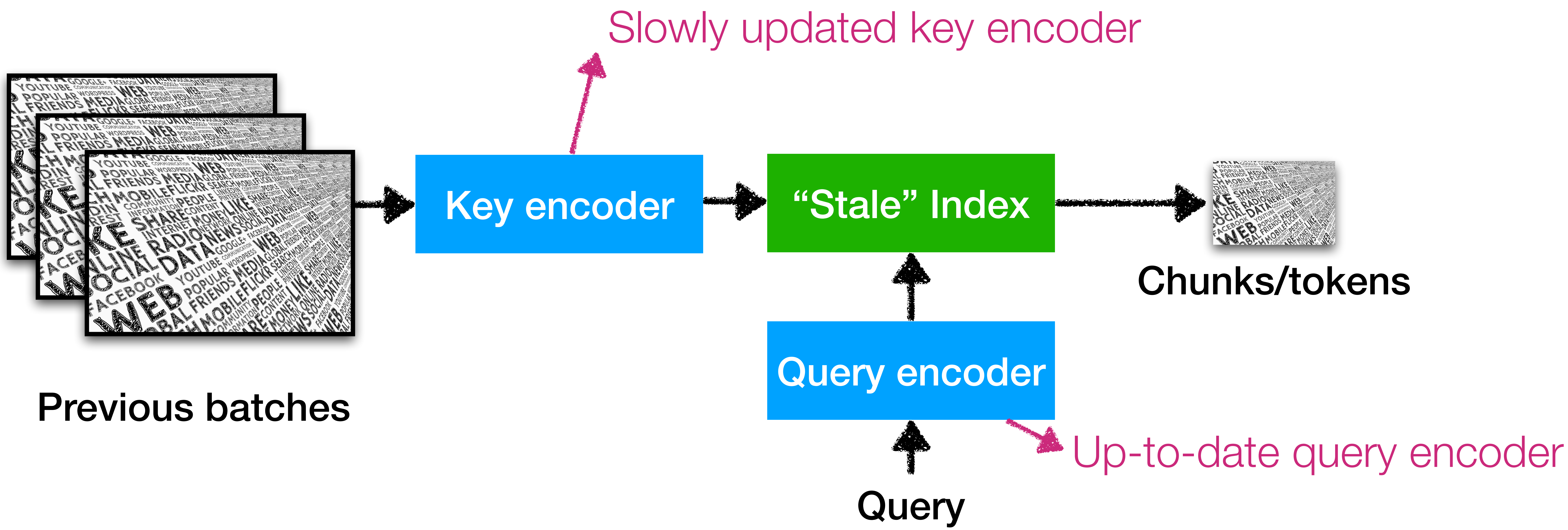
Contriever (Izacard et al. 2022)

MoCo (Momentum Contrast)



Contriever (Izacard et al. 2022)

MoCo (Momentum Contrast)



Get more negatives from previous batches!

Retrieval-in-context in LM (Ram et al. 2023)

x = World Cup 2022 was the last with 32 teams, before the increase to

World Cup 2022 was the last with 32 teams, before the increase to

↓

Retrieval Model

↓

FIFA World Cup 2026 will expand to 48 teams. World Cup 2022 was the last with 32 teams, before the increase to

↓

LM

↓

48 in the 2026 tournament.

Ram et al. 2023. "In-Context Retrieval-Augmented Language Models"

Retrieval-in-context in LM (Ram et al. 2023)

$\boldsymbol{x}$ = World Cup 2022 was the last with 32 teams, before the increase to

World Cup 2022 was the last with 32 teams, before the increase to

↓

Retrieval Model | BM25, DPR, Contriever, …

↓

FIFA World Cup 2026 will expand to 48 teams. World Cup 2022 was the last with 32 teams, before the increase to

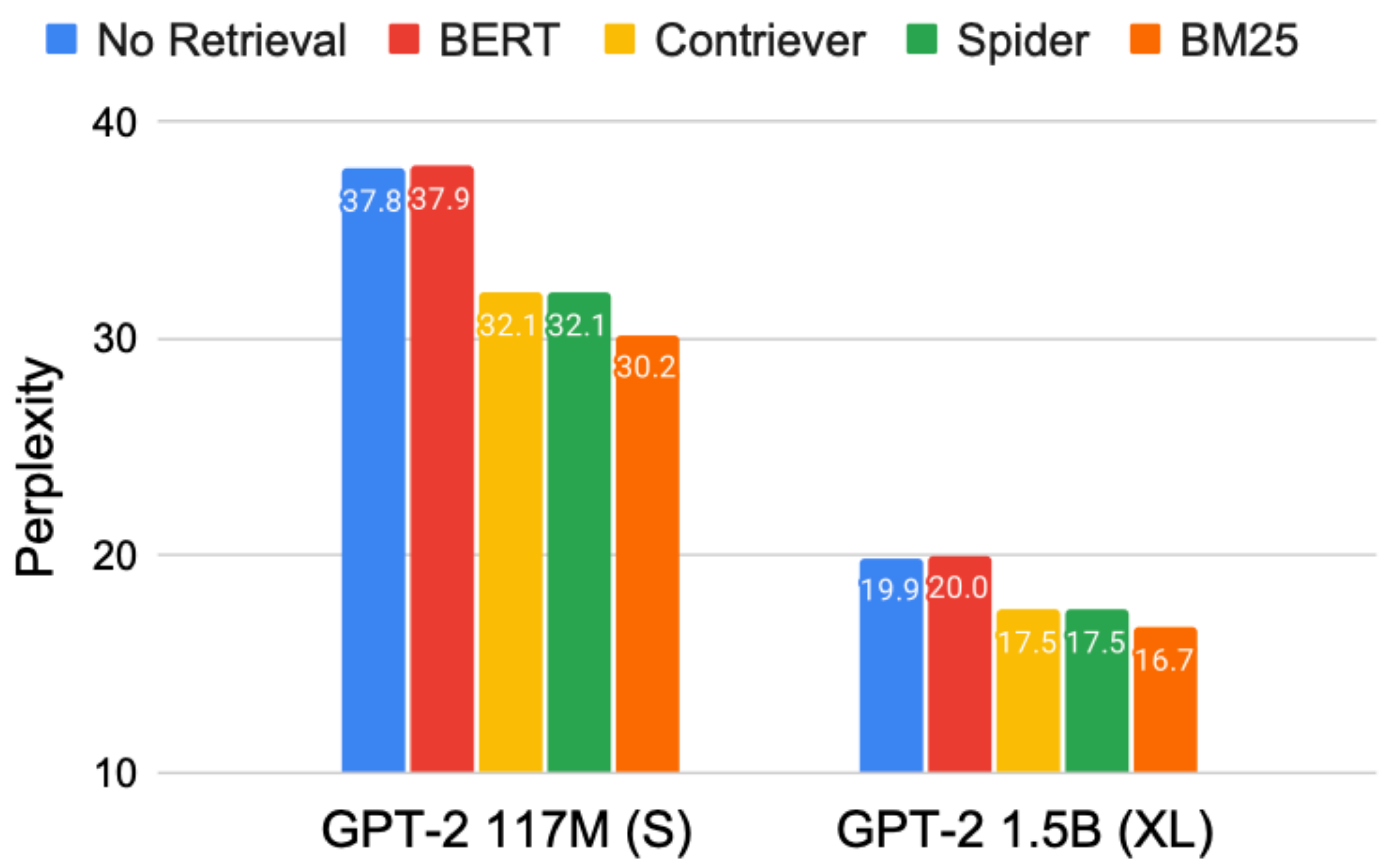
↓

LM | GPT, OPT, LLaMA, …

↓

48 in the 2026 tournament.

Retrieval-in-context in LM

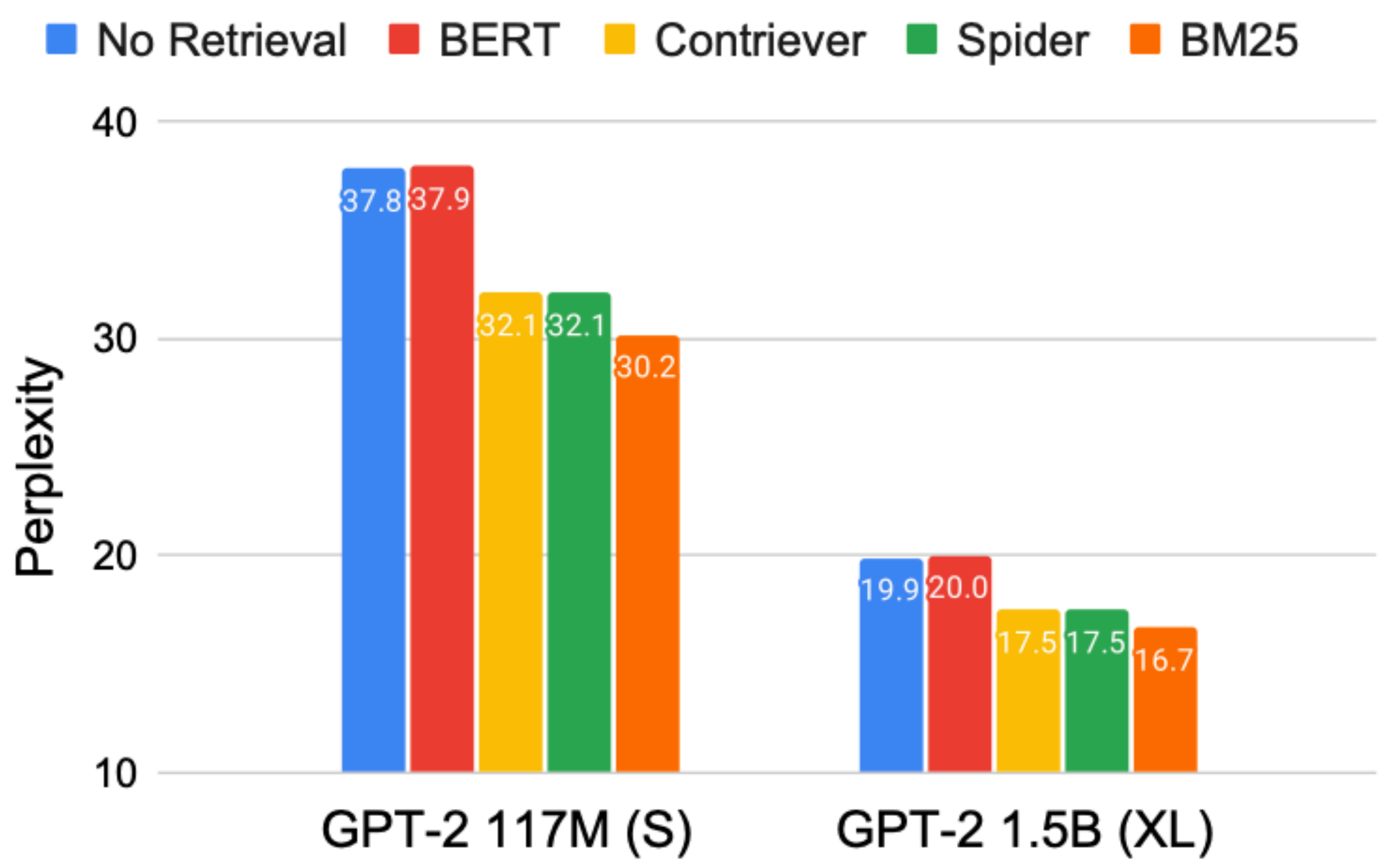


Better retrieval model

Better base LMs

→ Better **retrieval-based LMs**

Retrieval-in-context in LM



Better retrieval model

Better base LMs

→ Better **retrieval-based LMs**

Each component can be improved separately

kNN-LM (Khandelwal et al. 2020)

Inference

$$P_{k\mathrm{NN}}(y|x) \propto \sum_{(k,v)\in\mathcal{D}} \mathbb{I}[v=y]\exp(-d(\mathrm{Enc}(k),\mathrm{Enc}(x)))$$

$$P_{k\mathrm{NN-LM}}(y|x) = \lambda P_{\mathrm{LM}}(y|x) + (1-\lambda)P_{k\mathrm{NN}}(y|x)$$

Khandelwal et al., 2020. “Generalization through Memorization: Nearest Neighbor Language Models”

kNN-LM (Khandelwal et al. 2020)

Inference

Use the same encoder as the base LMs!

$$P_{k\mathrm{NN}}(y \mid x) \propto \sum_{(k,v)\in\mathcal{D}} \mathbb{1}[v = y]\exp(-d(\mathrm{Enc}(k), \mathrm{Enc}(x)))$$

$$P_{k\mathrm{NN-LM}}(y \mid x) = \lambda P_{\mathrm{LM}}(y \mid x) + (1 - \lambda) P_{k\mathrm{NN}}(y \mid x)$$

Khandelwal et al., 2020. “Generalization through Memorization: Nearest Neighbor Language Models”

kNN-LM (Khandelwal et al. 2020)

Inference

Use the same encoder as the base LMs!

$$P_{kNN}(y \mid x) \propto \sum_{(k,v) \in \mathcal{D}} \mathbb{1}[v = y] \exp(-d(\mathrm{Enc}(k), \mathrm{Enc}(x)))$$

$$P_{kNN-LM}(y \mid x) = \lambda P_{LM}(y \mid x) + (1 - \lambda) P_{kNN}(y \mid x)$$

Training

Minimize $-\log P_{LM}(y \mid x)$

Khandelwal et al., 2020. “Generalization through Memorization: Nearest Neighbor Language Models”

Independent training

- Work with off-the-shelf models (no extra training required)
- Each part can be improved independently

Independent training

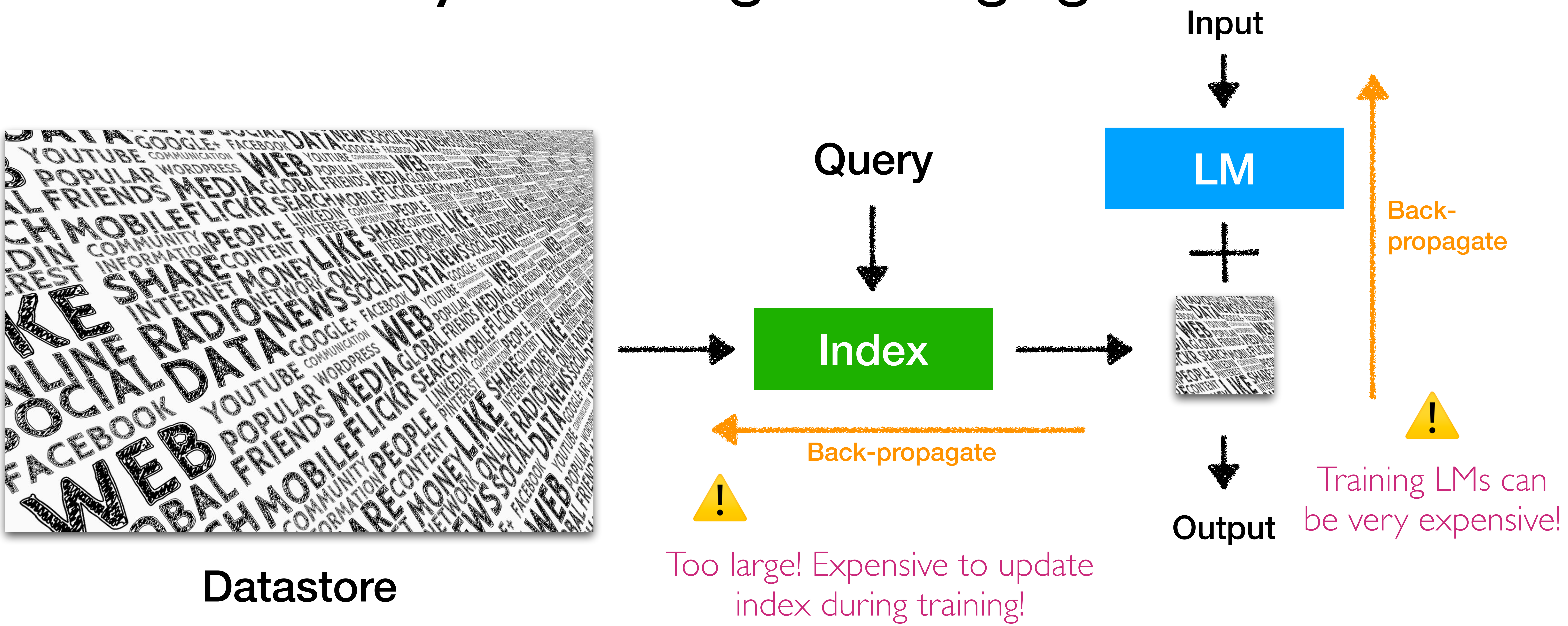
- 👍 Work with off-the-shelf models (no extra training required)
- 👍 Each part can be improved independently
- 👎 LMs are not trained to leverage retrieval
- 👎 Retrieval models are not optimized for LM tasks/domains

Q & A

We'll be back soon!

Section 4:
Retrieval-based LMs: Training (cont'd)

Why is training challenging?



Training methods for retrieval-based LMs

- Independent training
- Sequential training
- Joint training w/ asynchronous index update
- Joint training w/ In-batch approximation

Training methods for retrieval-based LMs

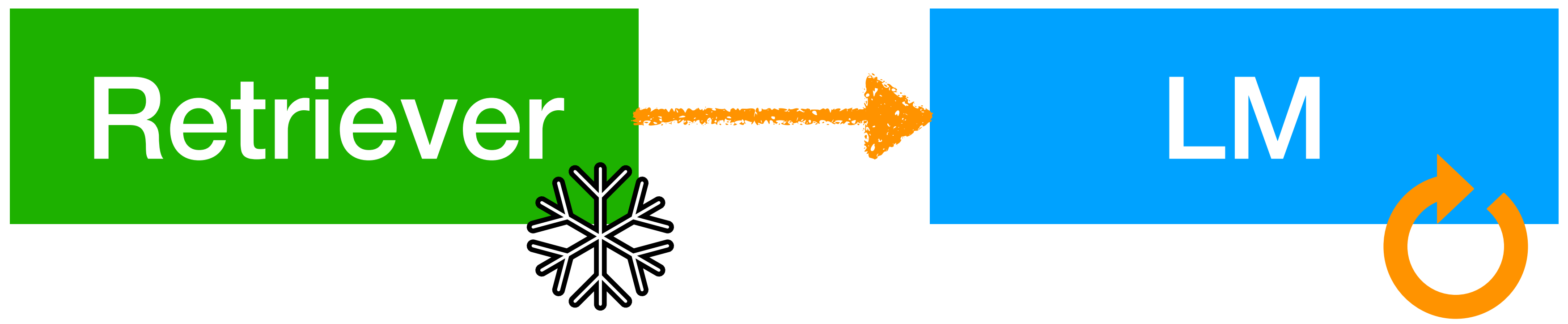
- Independent training
- **Sequential training**
- Joint training w/ asynchronous index update
- Joint training w/ in-batch approximation

Sequential training

- One component is first trained independently and then fixed
- The other component is trained with an objective that depends on the first one

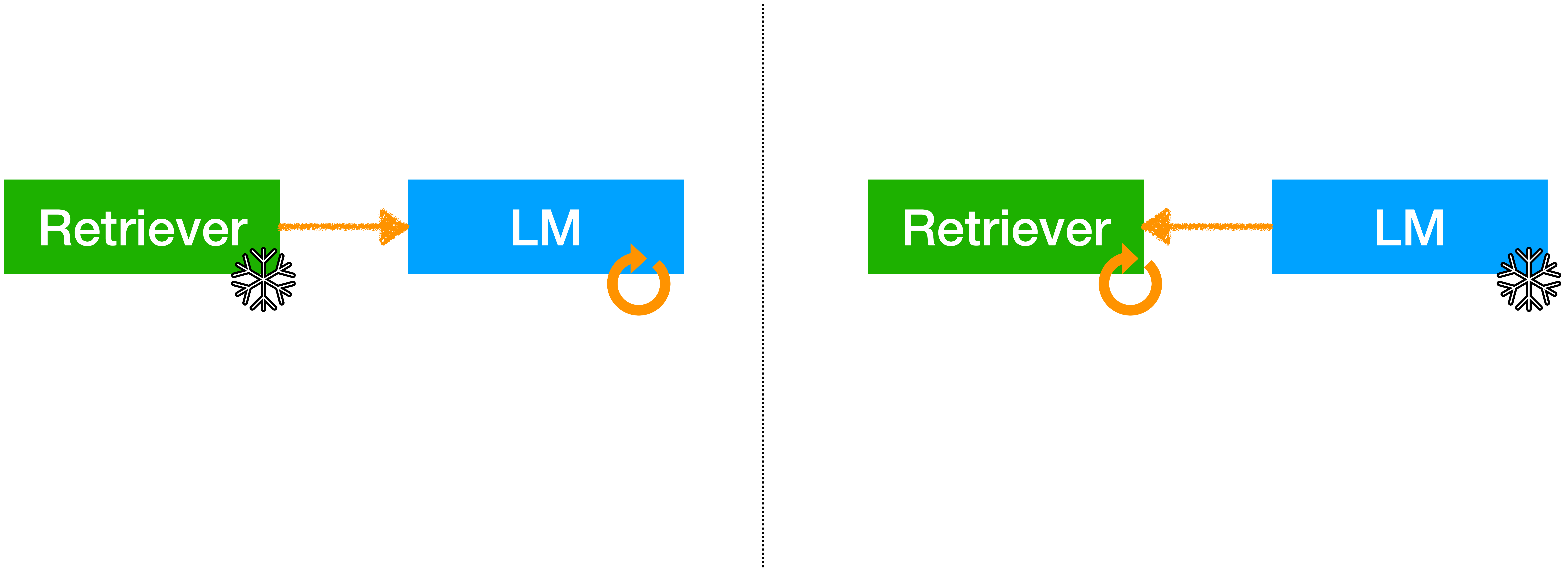
Sequential training

- One component is first trained independently and then fixed
- The other component is trained with an objective that depends on the first one



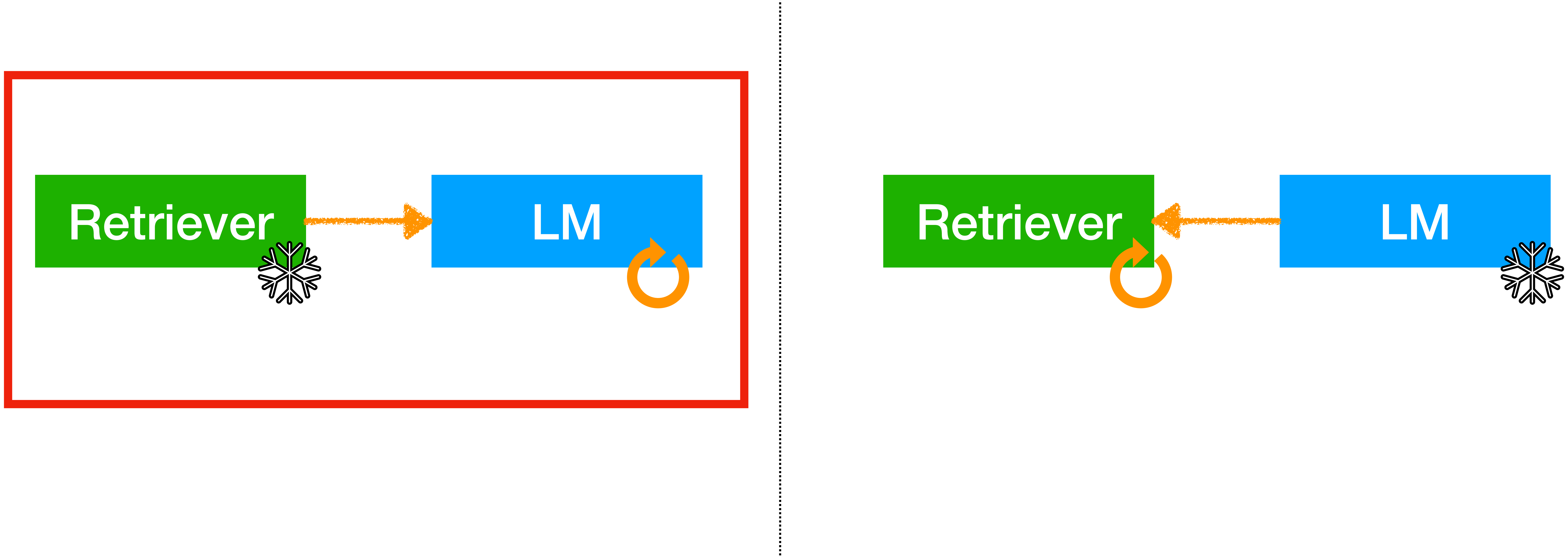
Sequential training

- One component is first trained independently and then fixed
- The other component is trained with an objective that depends on the first one



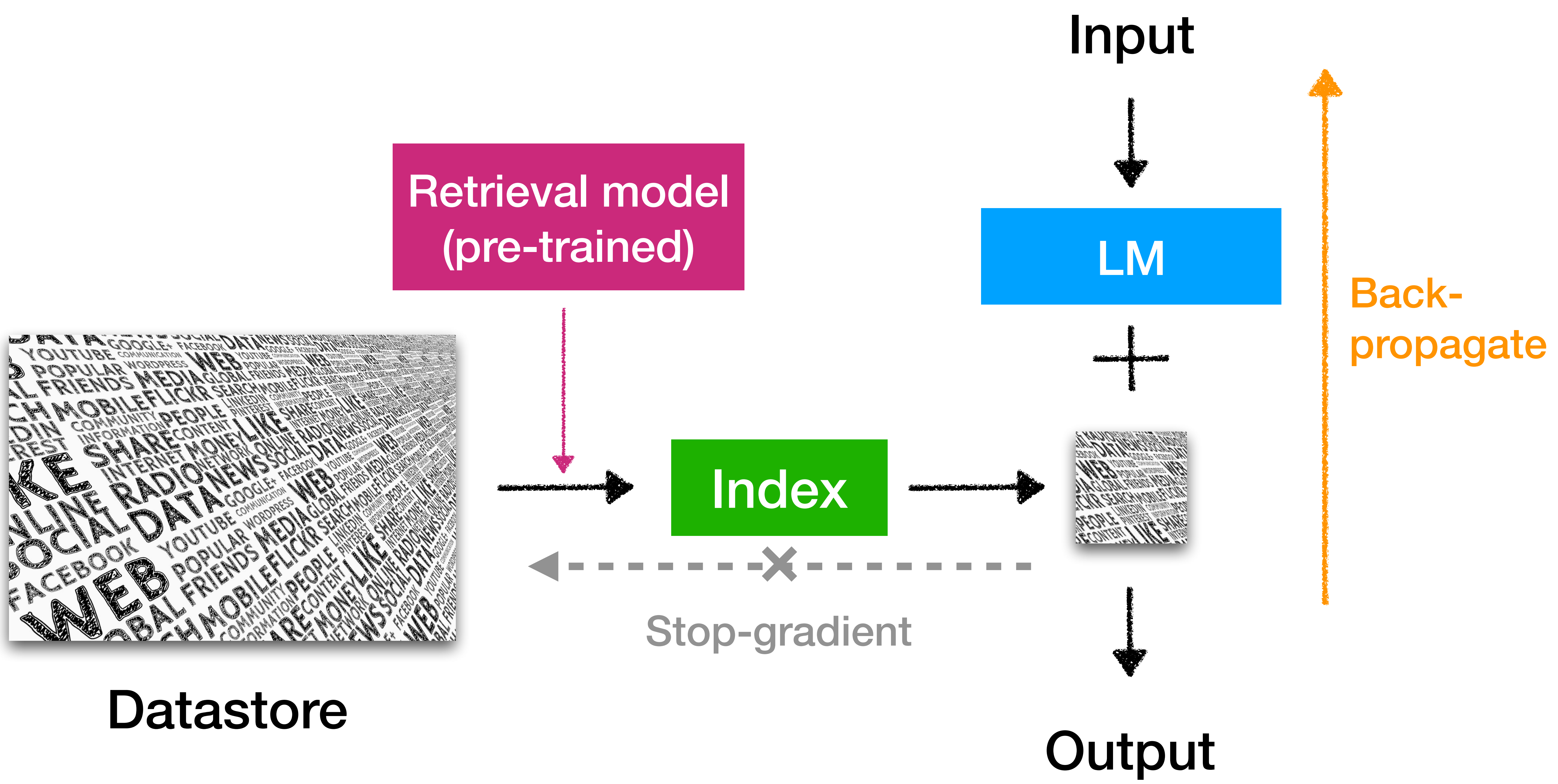
Sequential training

- One component is first trained independently and then fixed
- The other component is trained with an objective that depends on the first one



Sequential training

- Retrieval models are first trained independently and then fixed
- Language models are trained with an objective that depends on the retrieval



RETRO (Borgeaud et al. 2021)

$\boldsymbol{x}$ = World Cup 2022 was / the last with 32 teams, / before the increase to

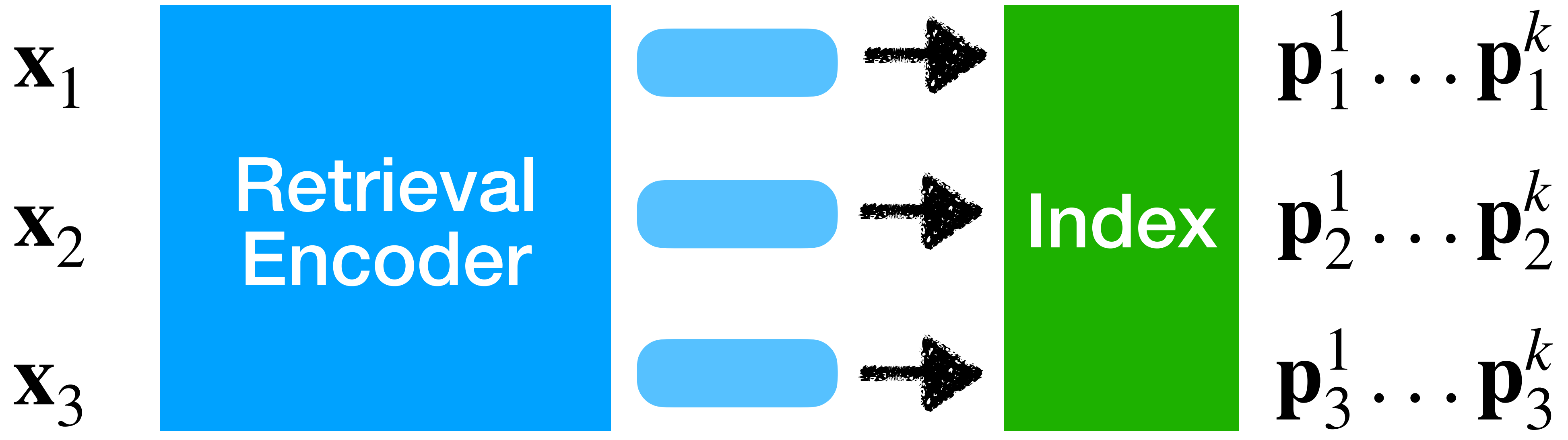
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$

Borgeaud et al., 2021. "Improving language models by retrieving from trillions of tokens"

RETRO (Borgeaud et al. 2021)

$\boldsymbol{x}$ = World Cup 2022 was the last with 32 teams, before the increase to

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$

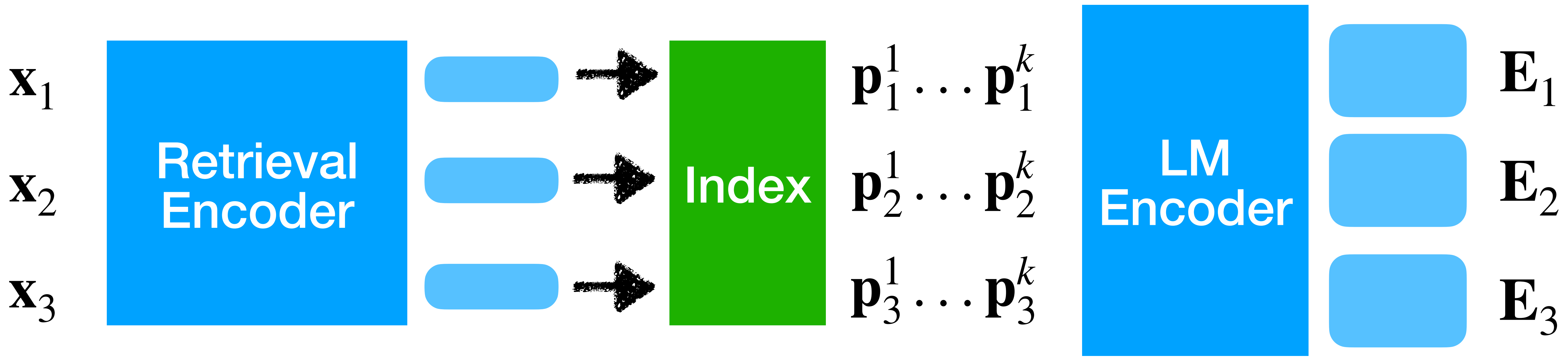


RETRO (Borgeaud et al. 2021)

$\boldsymbol{x}$ = World Cup 2022 was the last with 32 teams, before the increase to

$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$

(k chunks of text per split)

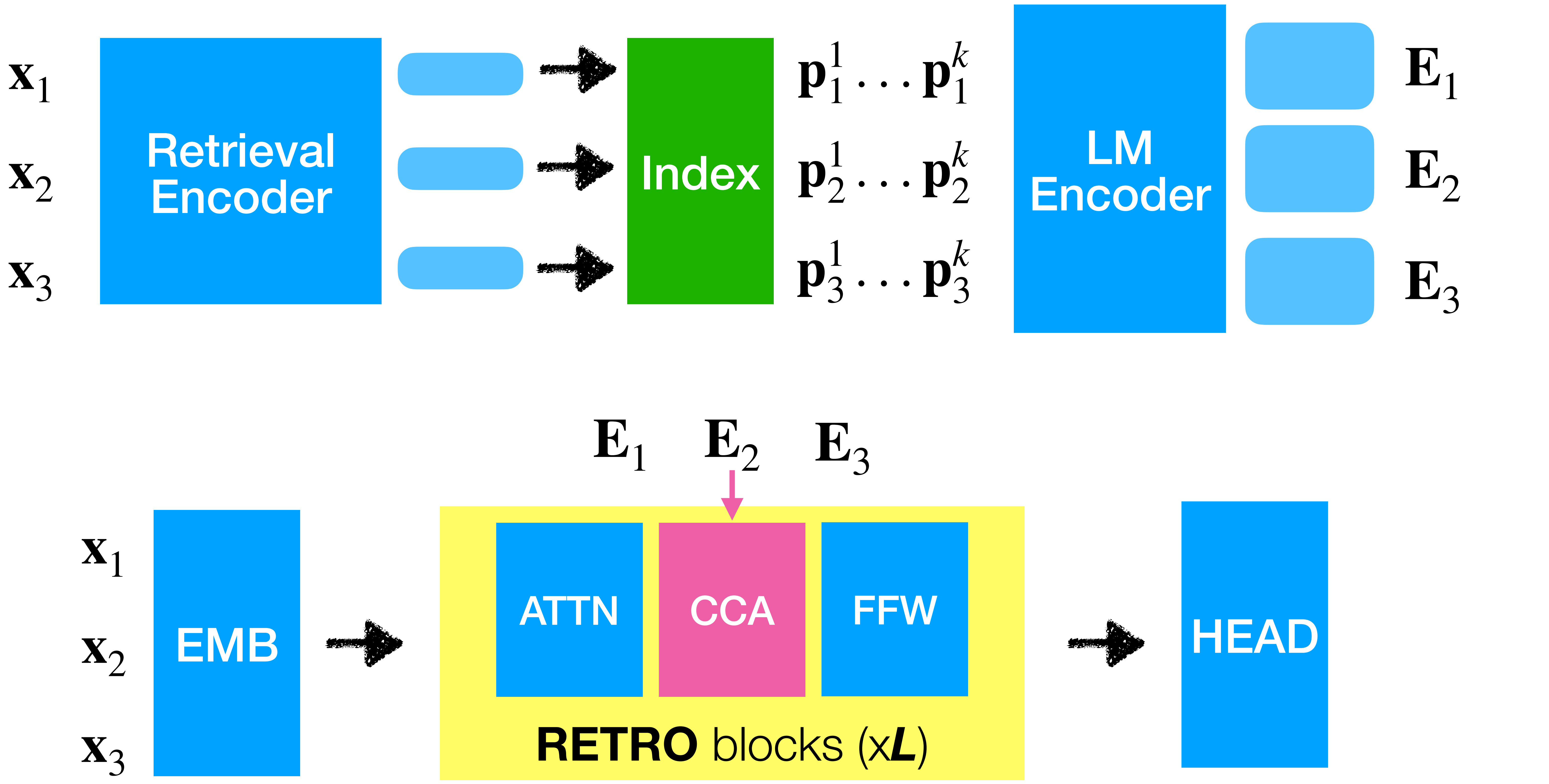


RETRO (Borgeaud et al. 2021)

$\boldsymbol{x}$ = World Cup 2022 was the last with 32 teams, before the increase to

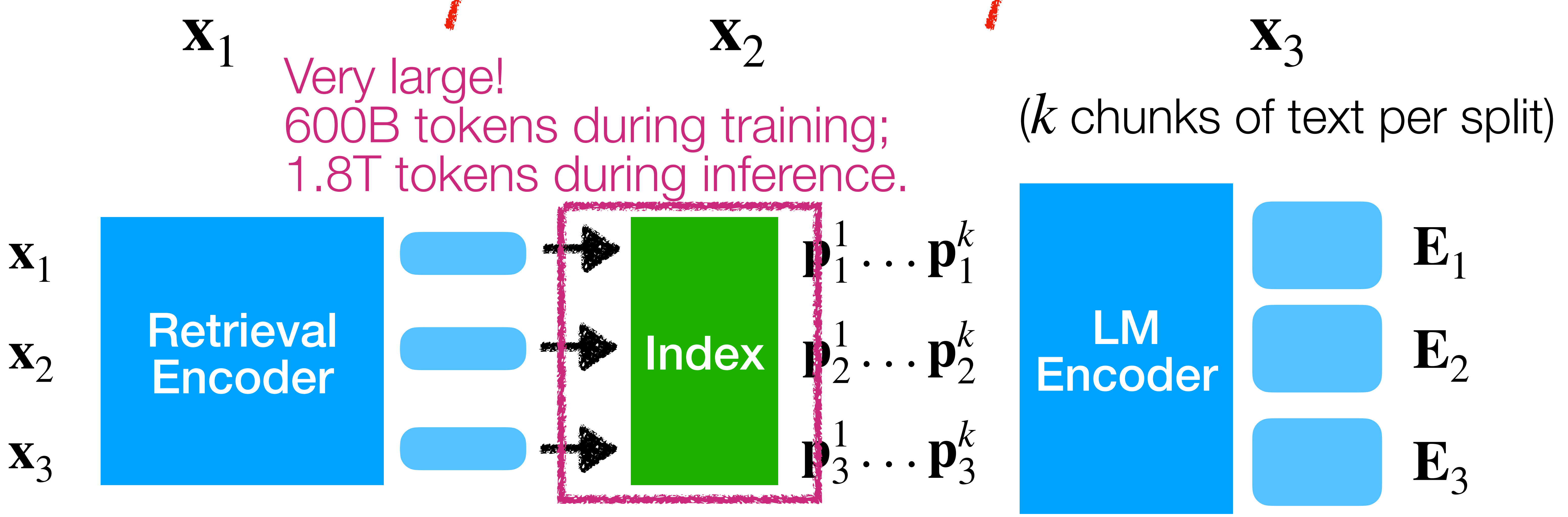
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$

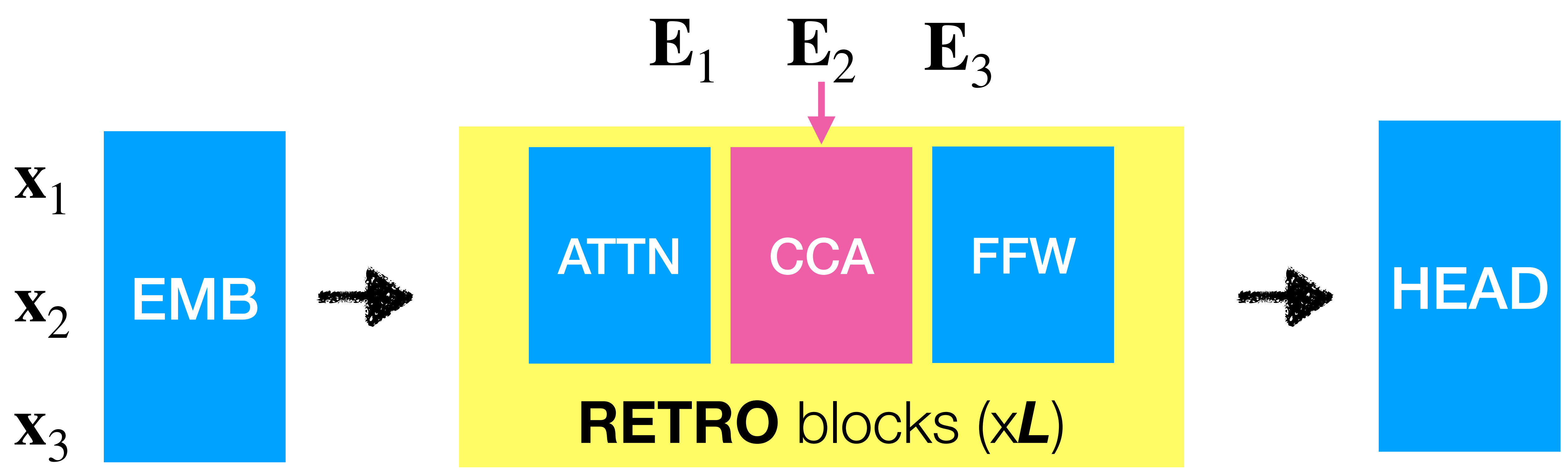
(k chunks of text per split)



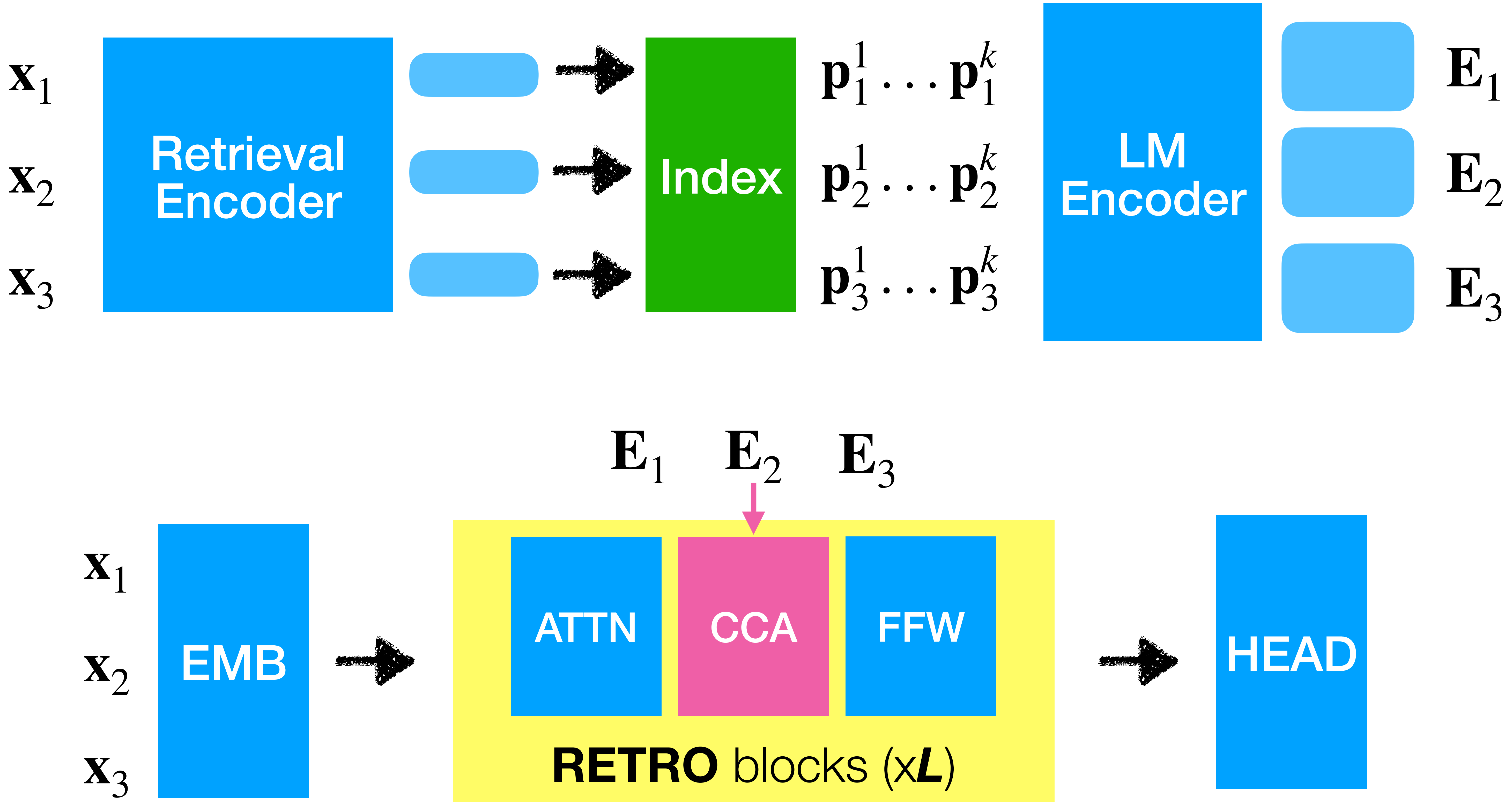
RETRO (Borgeaud et al. 2021)

$\boldsymbol{x}$ = World Cup 2022 was the last with 32 teams, before the increase to

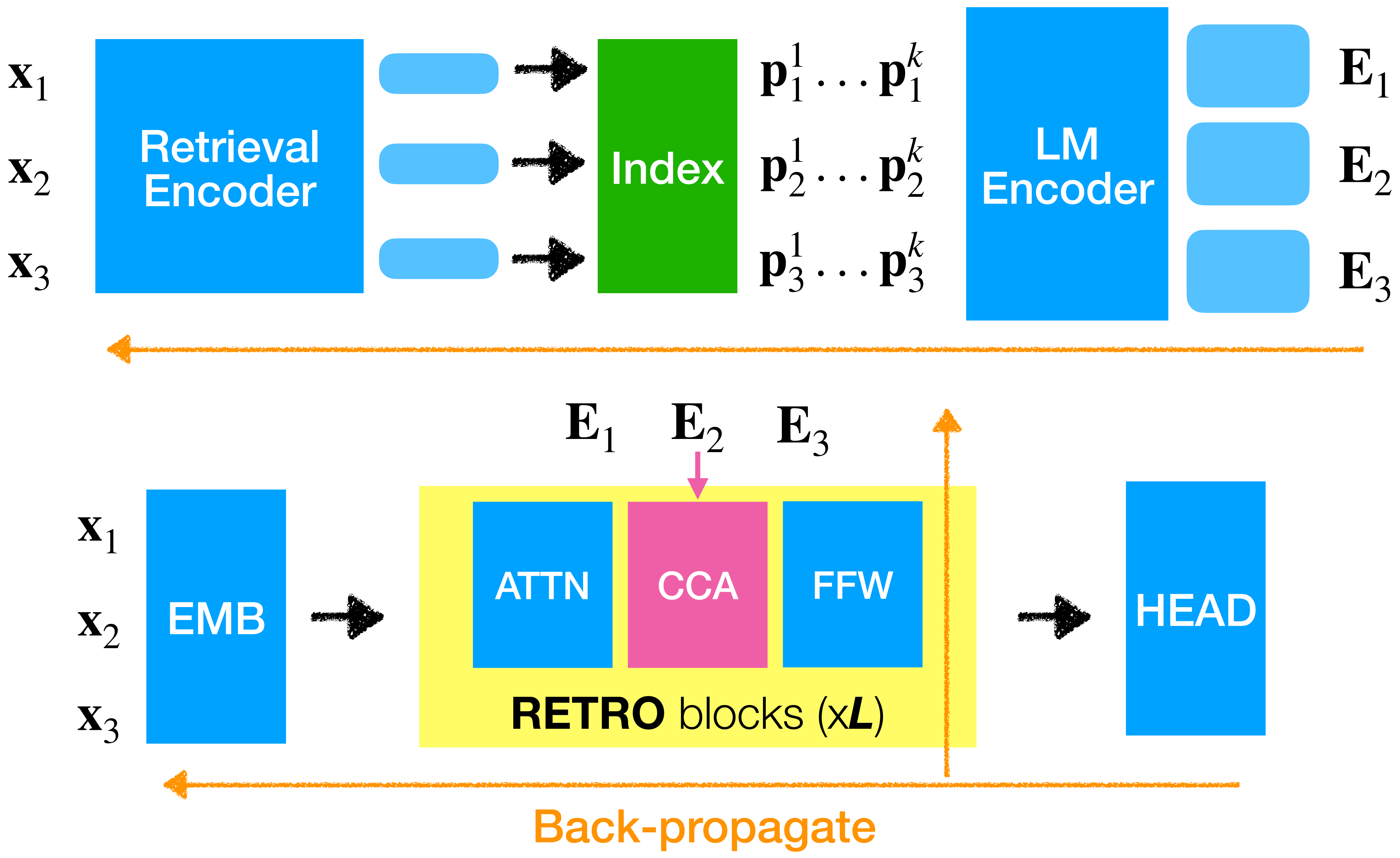




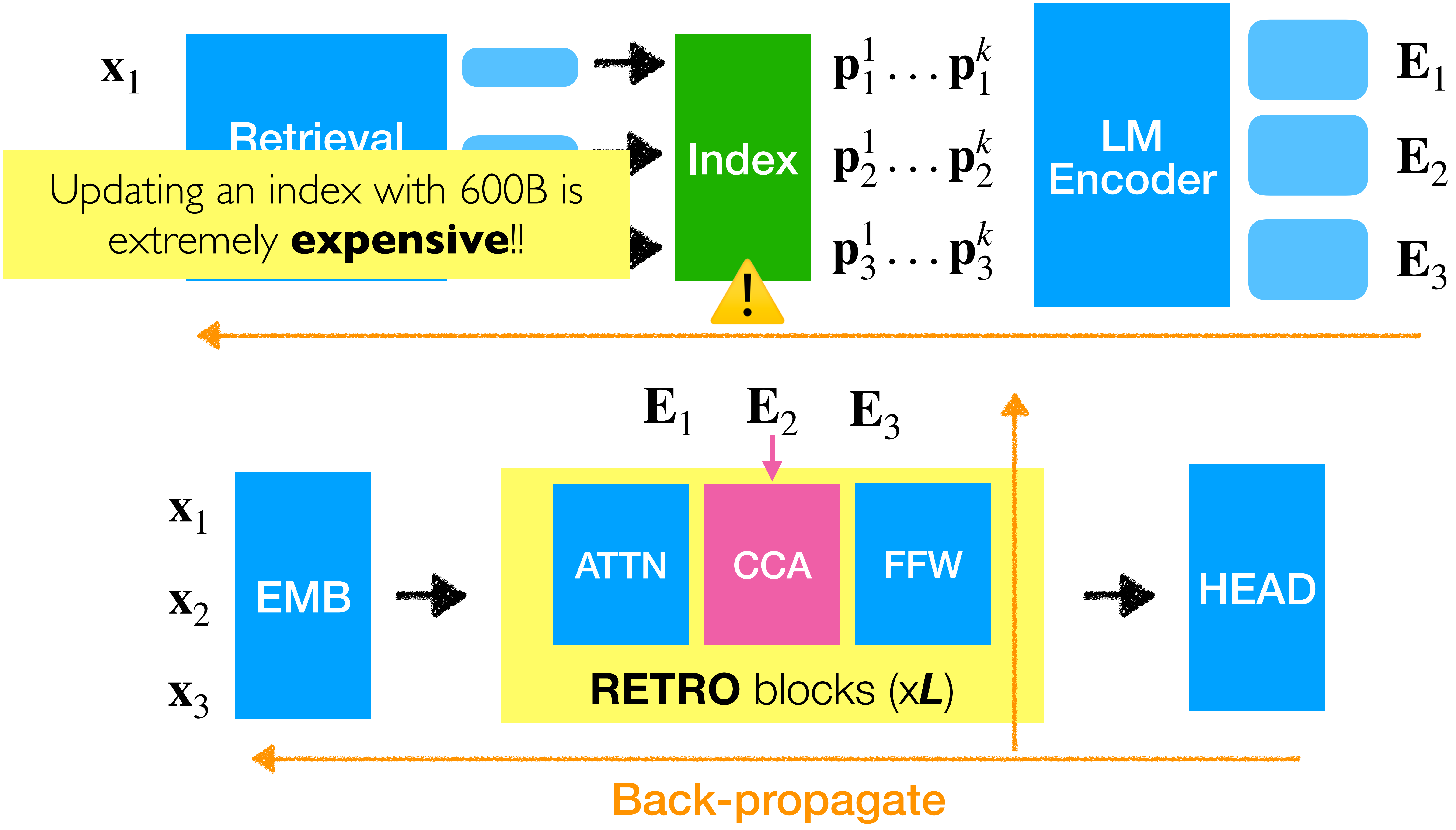
RETRO: Training



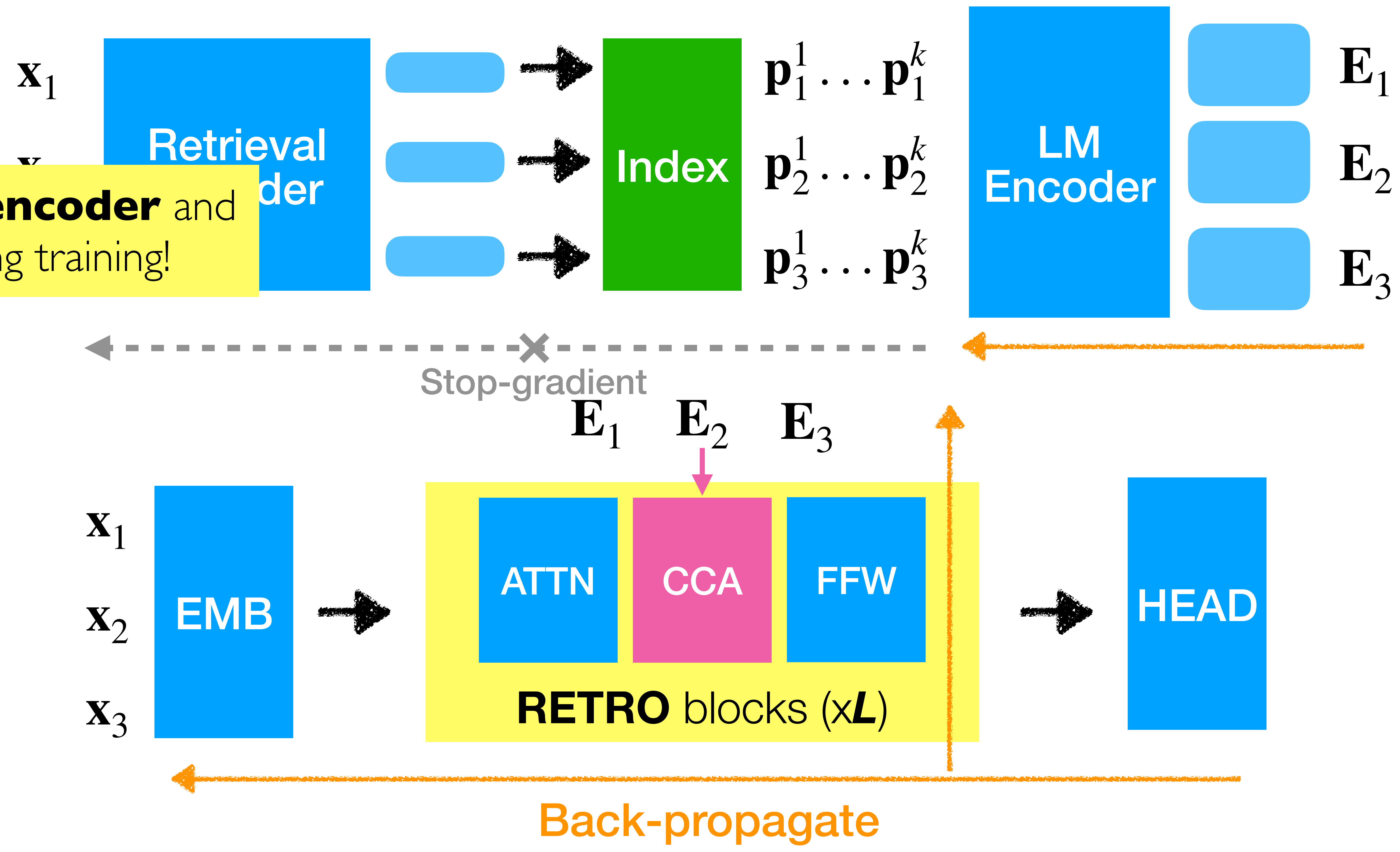
RETRO: Training



RETRO: Training



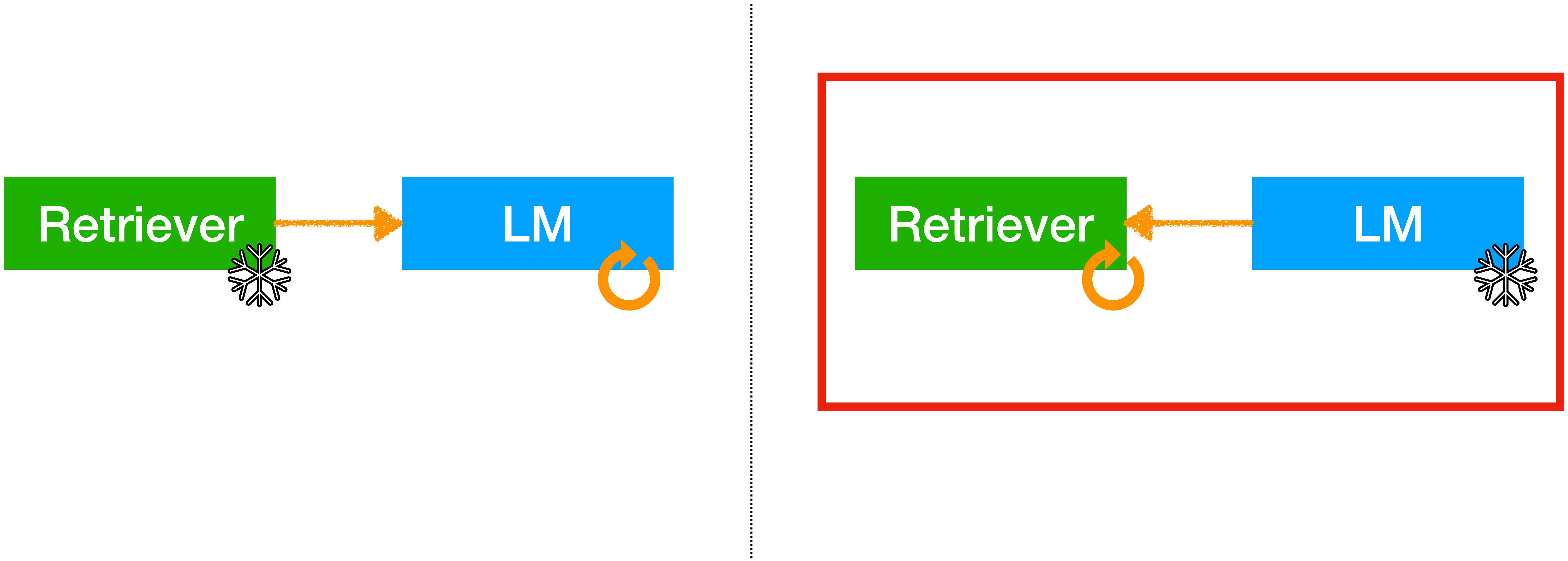
RETRO: Training



Fix the **retrieval encoder** and the **index** during training!

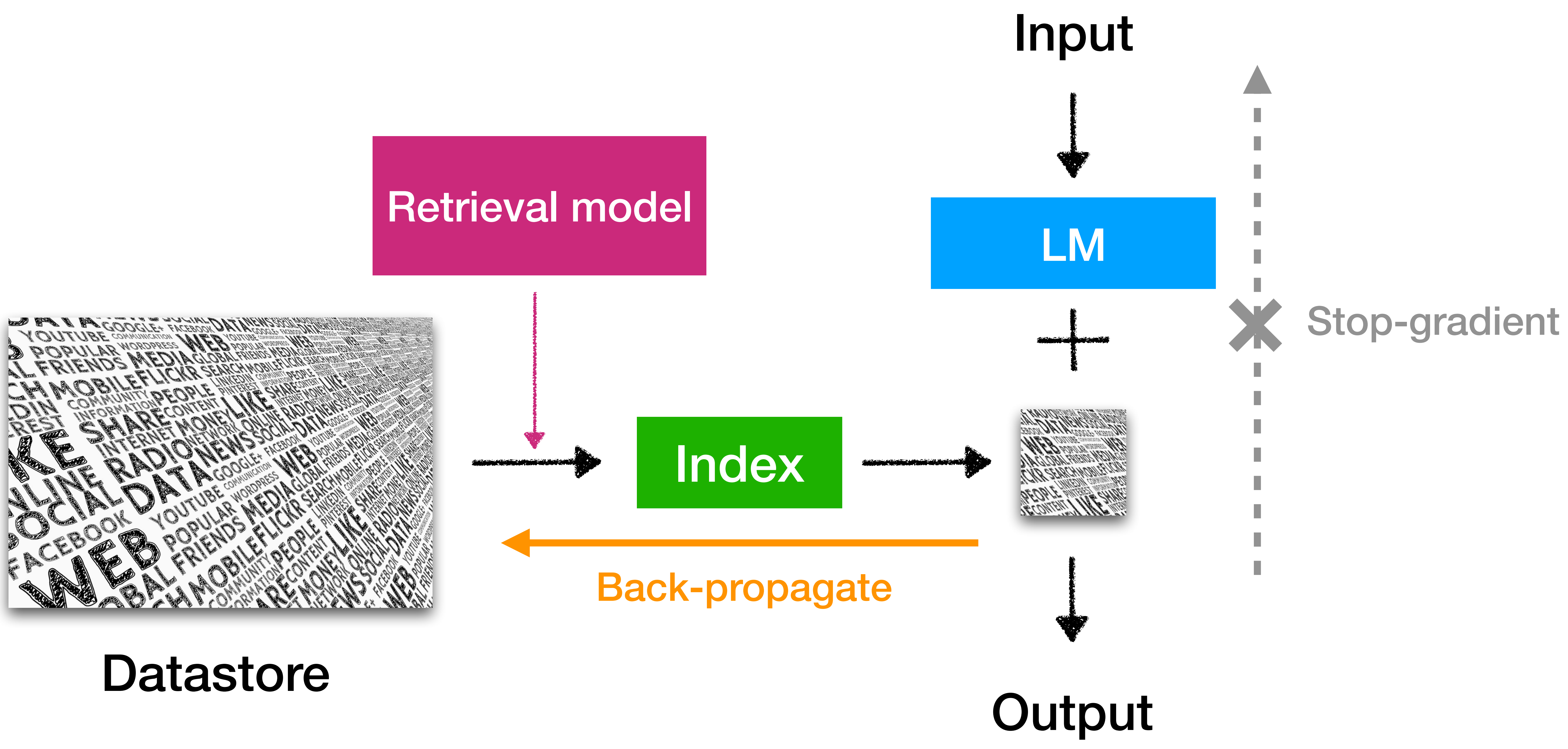
Sequential training

- One component is first trained independently and then fixed
- The other component is trained with an objective that depends on the first one

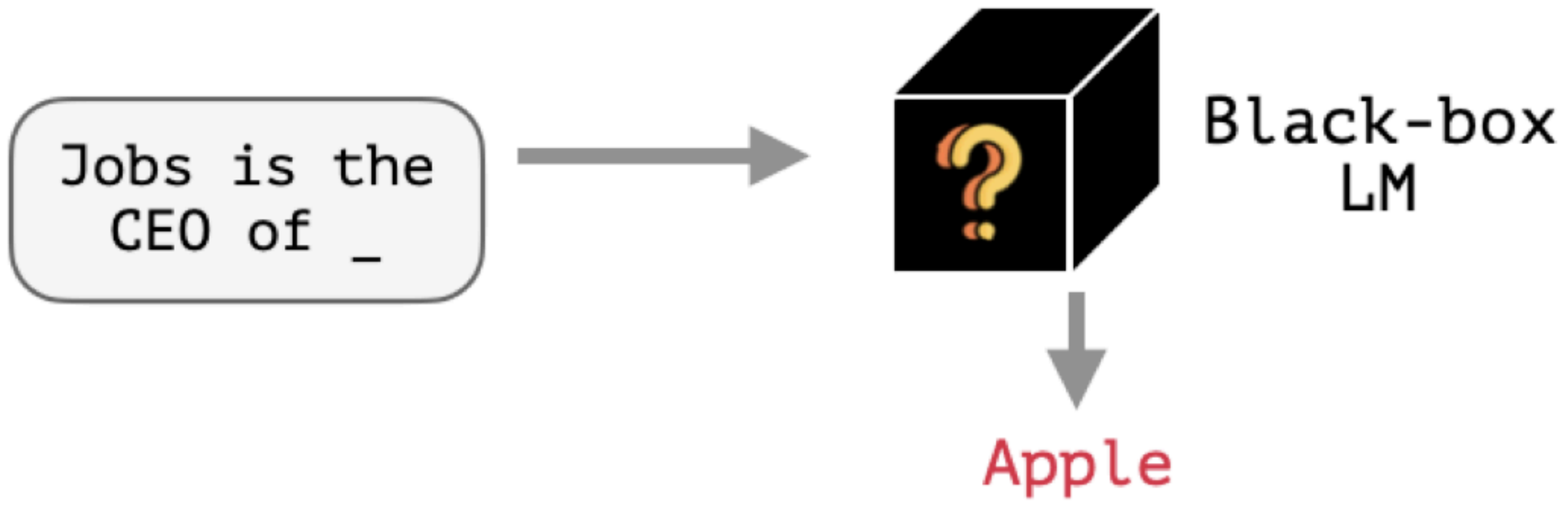


Sequential training

- Language models are first trained independently and then fixed
- Retrieval models are trained/fine-tuned with supervisions from LMs

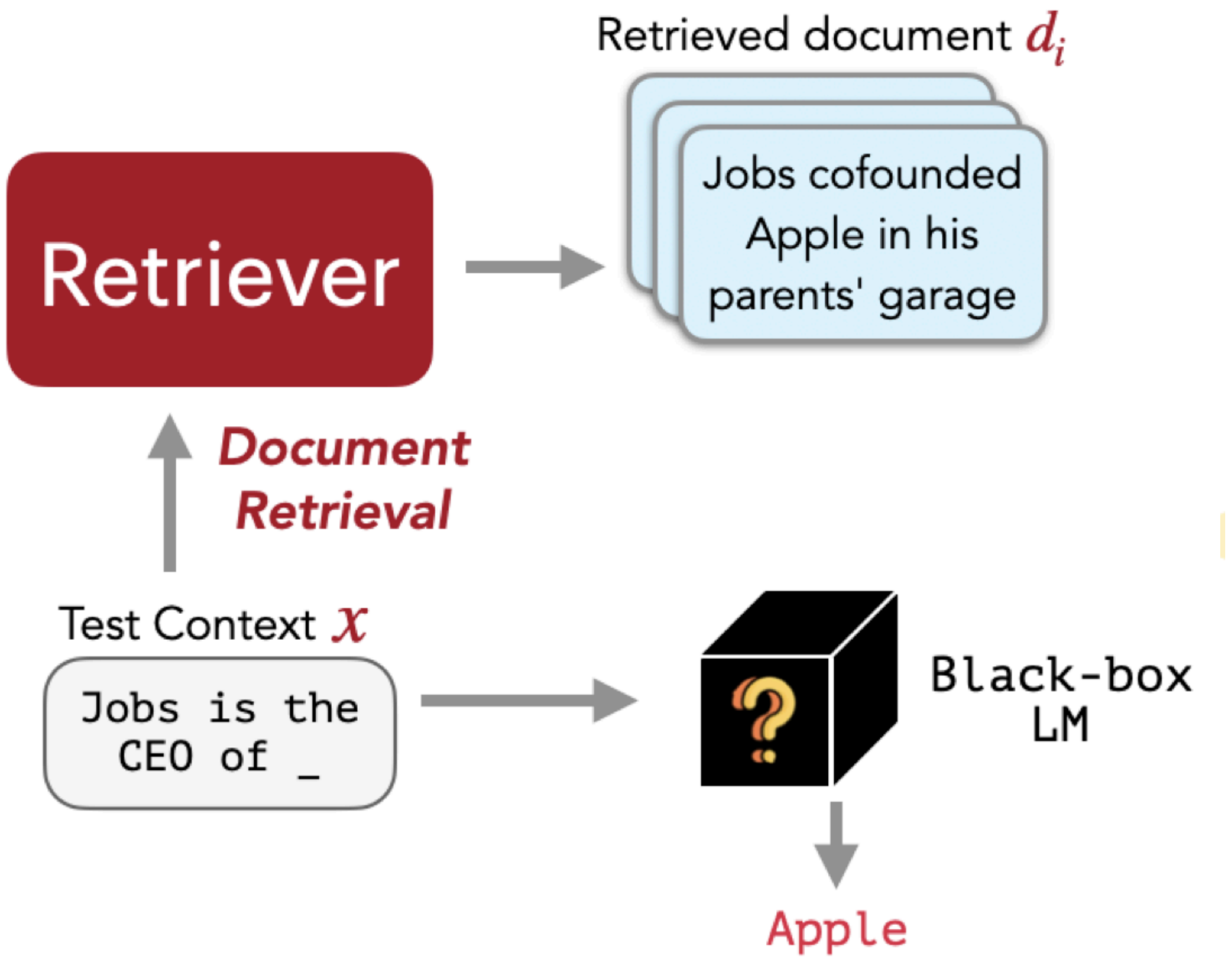


REPLUG (Shi et al. 2023)



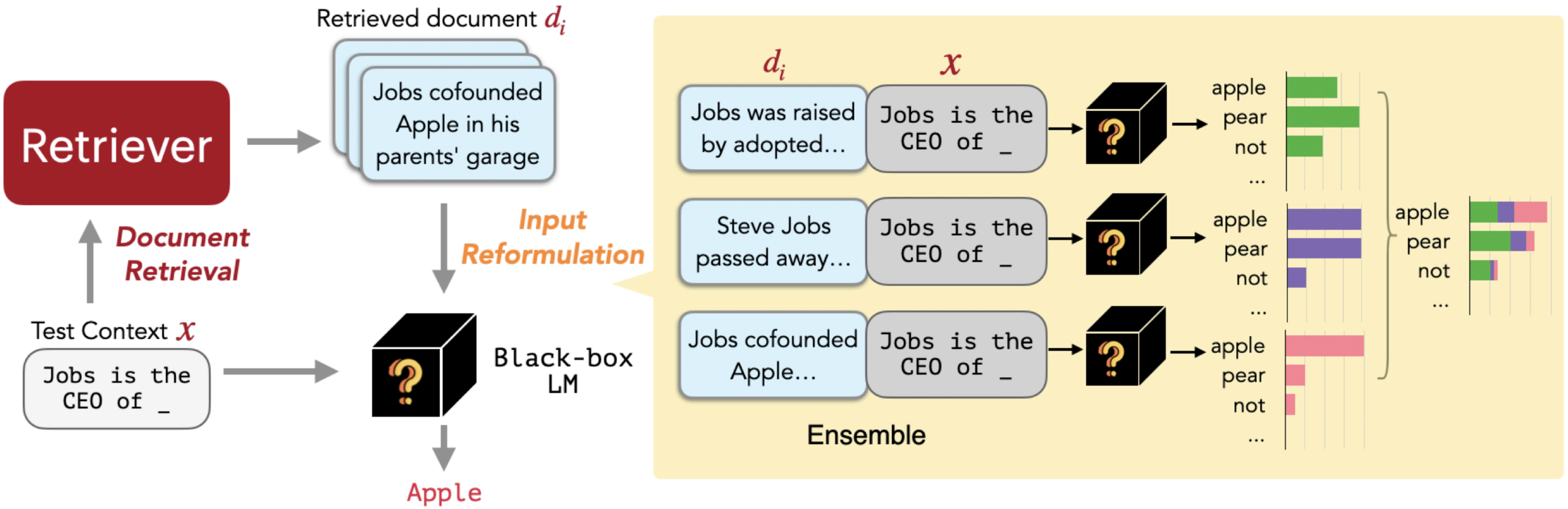
Shi et al., 2023. "REPLUG: Retrieval-Augmented Black-Box Language Models"

REPLUG (Shi et al. 2023)



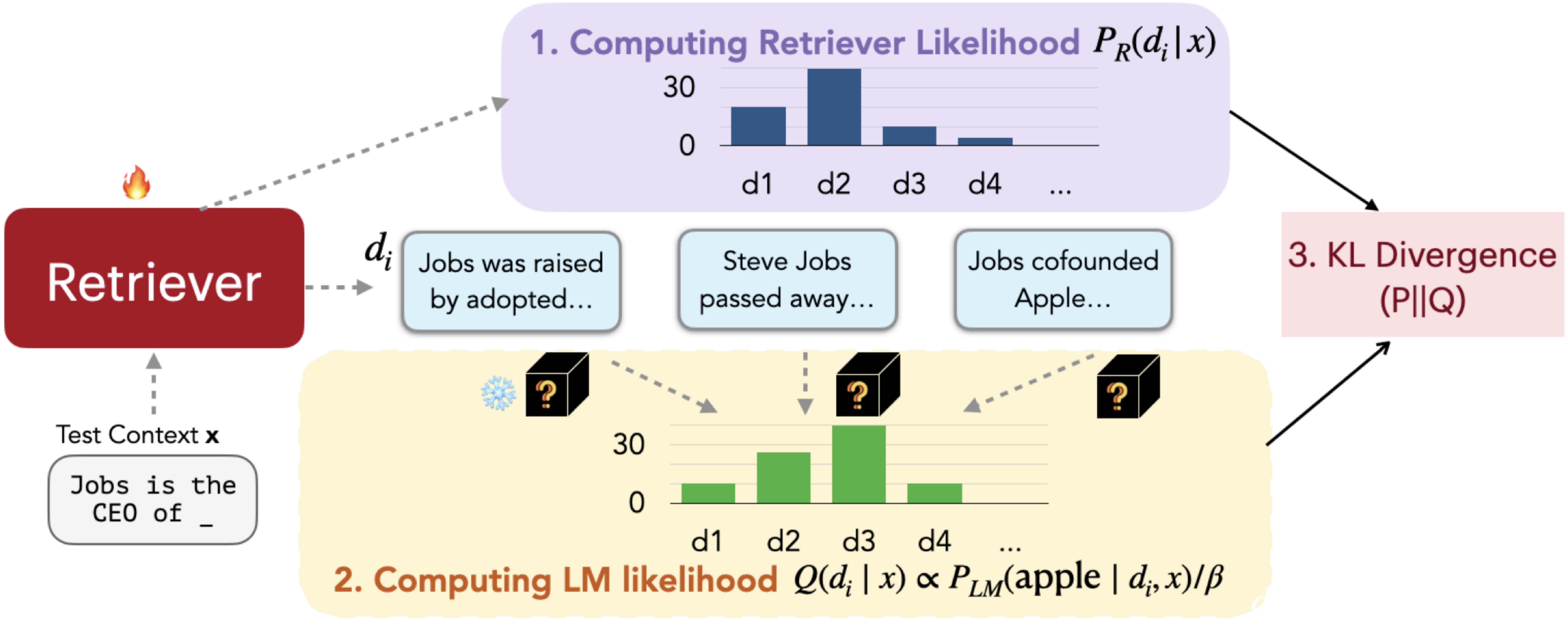
Shi et al., 2023. “REPLUG: Retrieval-Augmented Black-Box Language Models”

REPLUG (Shi et al. 2023)



Shi et al., 2023. "REPLUG: Retrieval-Augmented Black-Box Language Models"

REPLUG LSR (LM-Supervised Retrieval)



Updating retrieval encoder ⟶ Retrieval Index becomes "stale"

How to deal with this issue? We will talk about it soon!

REPLUG results

Model		# Parameters	Perplexity Original
GPT-2	Small	117M	1.33
	Medium	345M	1.20
	Large	774M	1.19
	XL	1.5B	1.16
GPT-3 (black-box)	Ada	350M	1.05
	Babbage	1.3B	0.95
	Curie	6.7B	0.88
	Davinci	175B	0.80

REPLUG results

With Contriever, “**independent training**”

Model		# Parameters	Original	+ REPLUG	Gain %
GPT-2	Small	117M	1.33	1.26	5.3
	Medium	345M	1.20	1.14	5.0
	Large	774M	1.19	1.15	3.4
	XL	1.5B	1.16	1.09	6.0
GPT-3 (black-box)	Ada	350M	1.05	0.98	6.7
	Babbage	1.3B	0.95	0.90	5.3
	Curie	6.7B	0.88	0.85	3.4
	Davinci	175B	0.80	0.77	3.8

REPLUG results

Fine-tuning Contriever with
LM-supervised training
"**Sequential training**"

Model		# Parameters	Original	+ REPLUG	Gain %	+ REPLUG LSR	Gain %
GPT-2	Small	117M	1.33	1.26	5.3	1.21	9.0
	Medium	345M	1.20	1.14	5.0	1.11	7.5
	Large	774M	1.19	1.15	3.4	1.09	8.4
	XL	1.5B	1.16	1.09	6.0	1.07	7.8
GPT-3 (black-box)	Ada	350M	1.05	0.98	6.7	0.96	8.6
	Babbage	1.3B	0.95	0.90	5.3	0.88	7.4
	Curie	6.7B	0.88	0.85	3.4	0.82	6.8
	Davinci	175B	0.80	0.77	3.8	0.75	6.3

Sequential training

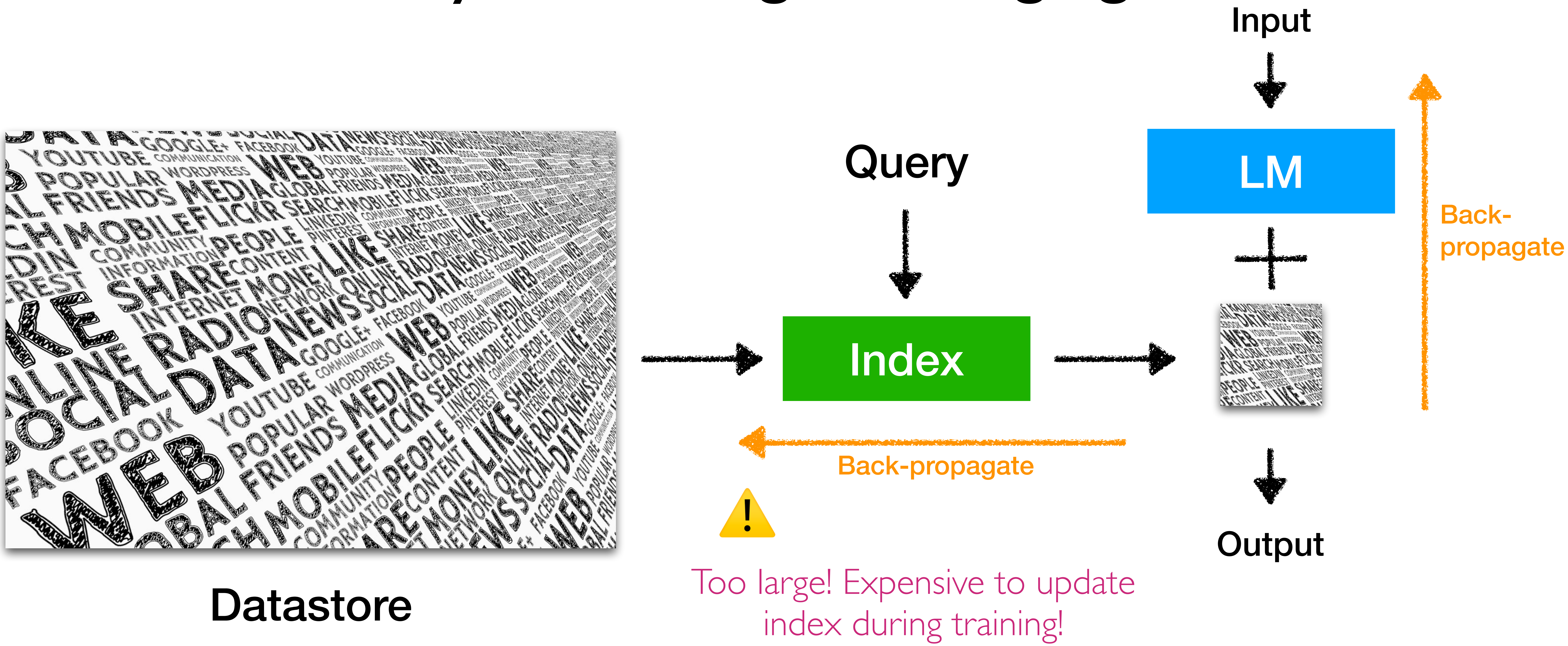
- Work with off-the-shelf components (either a large index or a powerful LM)
- LMs are trained to effectively leverage retrieval results
- Retrievers are trained to provide text that helps LMs the most
- One component is still fixed and not trained

Let's jointly train retrieval models and LMs!

Training methods for retrieval-based LMs

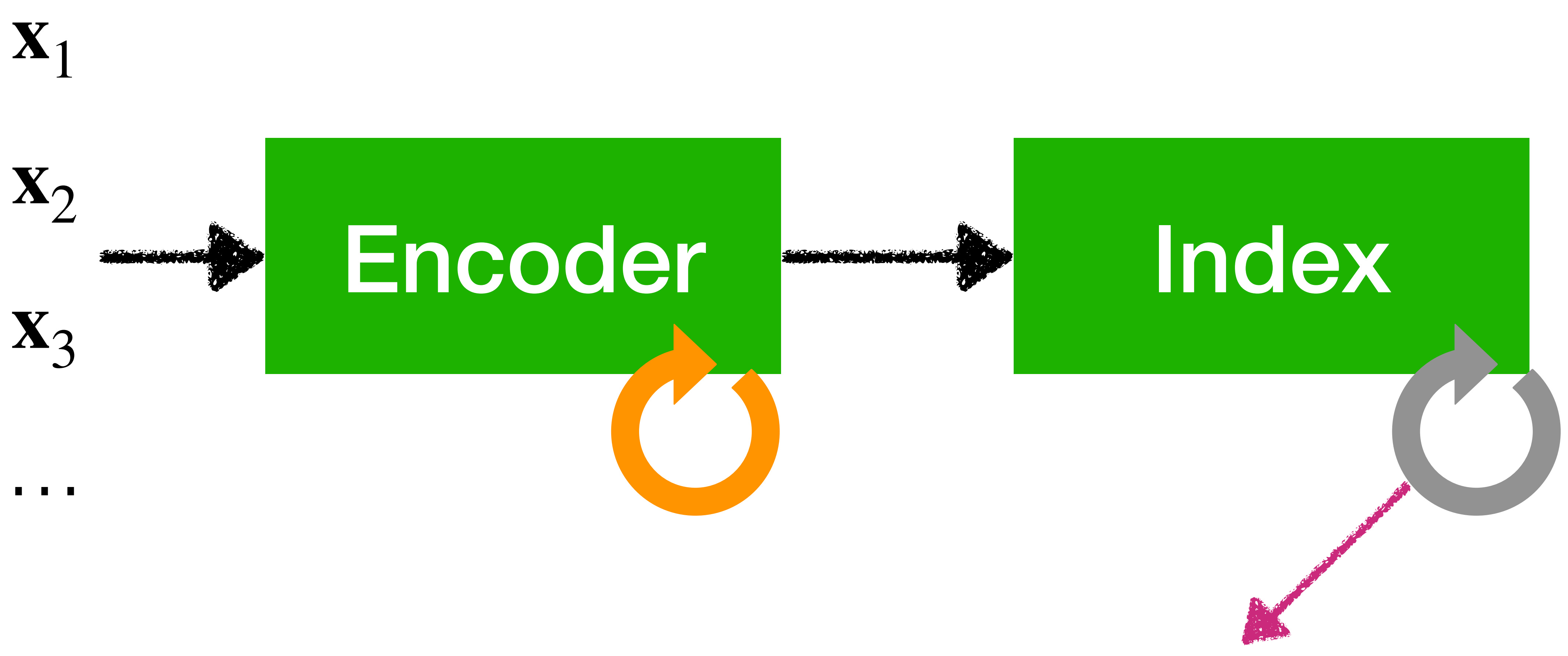
- Independent training
- Sequential training
- **Joint training w/ asynchronous index update**
- **Joint training w/ in-batch approximation**

Why is training challenging?



Challenges of updating retrieval models

Datastore



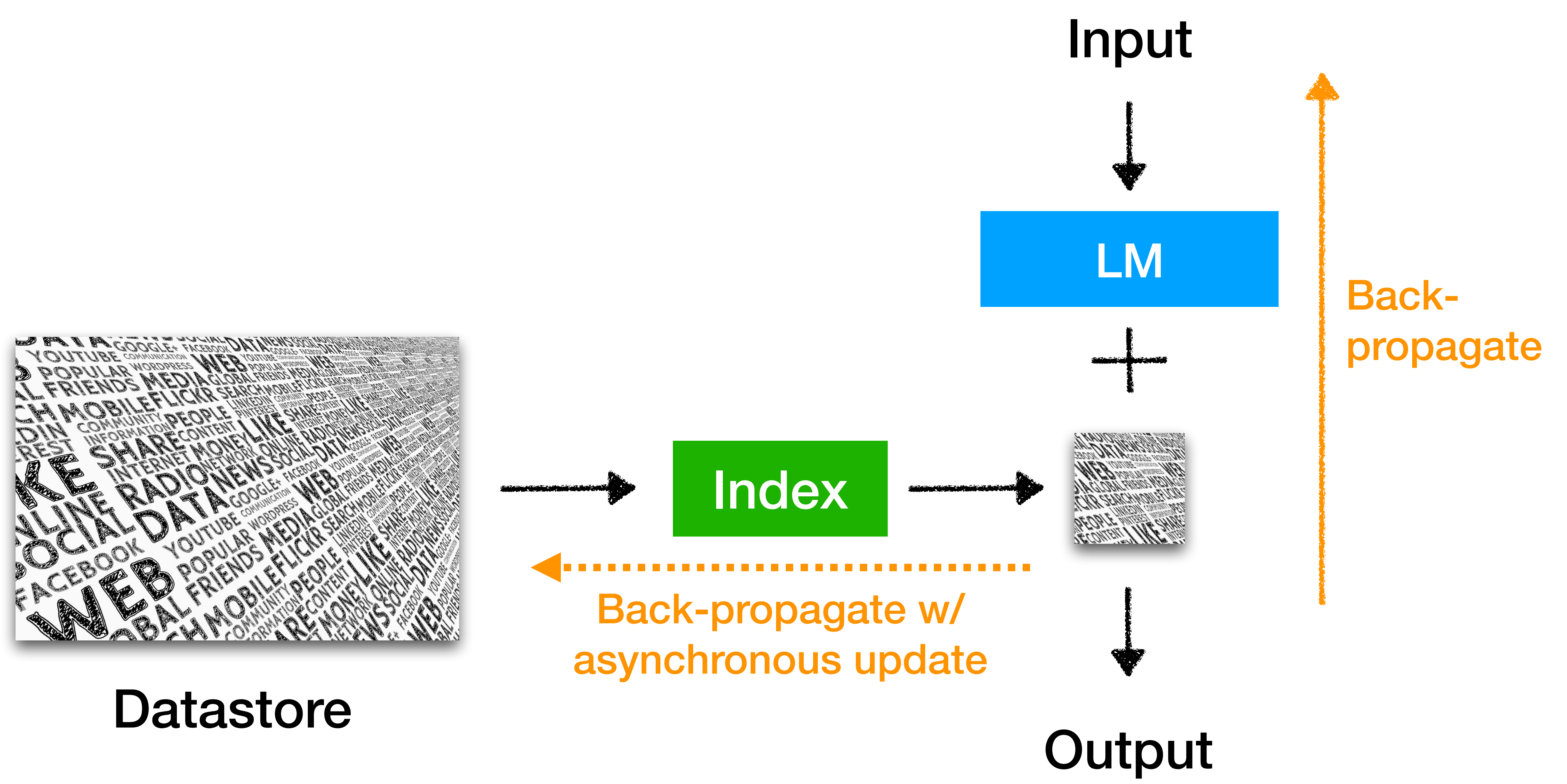
Re-indexing will be very expensive!

Training methods for retrieval-based LMs

- Independent training
- Sequential training
- **Joint training w/ asynchronous index update**
- Joint training w/ in-batch approximation

Asynchronous index update

- Retrieval models and language models are trained jointly
- Allow the index to be “**stale**”; rebuild the retrieval index every T steps



Asynchronous index update

Datastore

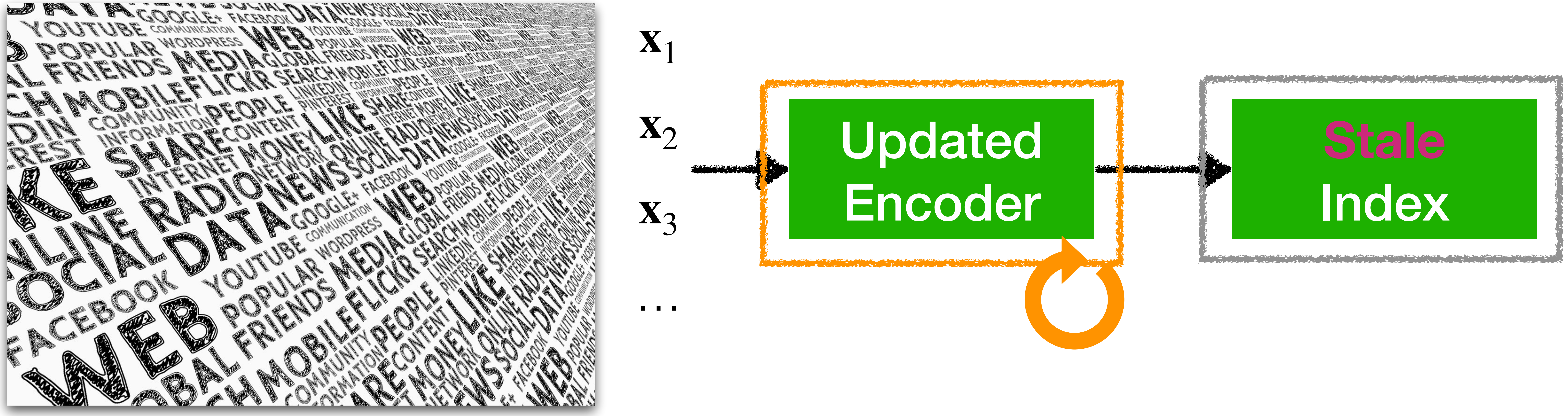
$\mathbf{x}_1$

$\mathbf{x}_2$ → Encoder → Index

$\mathbf{x}_3$

$\dots$

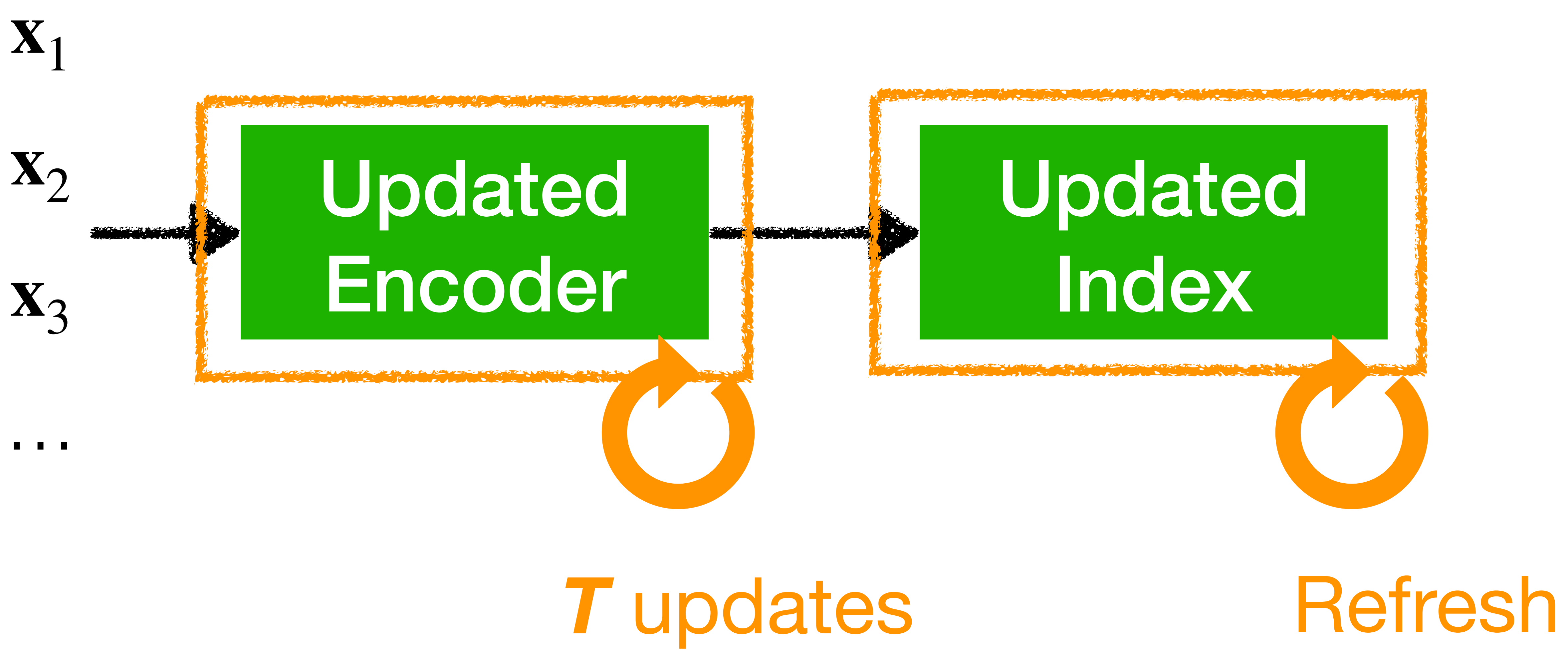
Asynchronous index update



Datastore

Asynchronous index update

Datastore



REALM (Guu et al. 2020)

$\boldsymbol{x}$ = The [MASK] at the top of the pyramid.

q (=x)

Datastore

The pyramidion on top allows for less material higher up the pyramid.

$P(z \mid x)$

The pyramidion on top … the pyramid.

…

The [MASK] at the top of the pyramid.

LM

pyramid

$P(y \mid x, z)$

Guu et al., 2020. "REALM: Retrieval-Augmented Language Model Pre-Training"

REALM: Training

Objective: $\text{maximize} \sum_{z \in \mathcal{Z}_\theta} P_\theta(z \mid q) P_\theta(y \mid q, z)$

***q* (=*x*)**

Datastore

$\mathcal{Z}_\theta$: top-K retrieved documents

The pyramidion on top allows for less material higher up the pyramid.

$P_\theta(z \mid x)$

The pyramidion on top … the pyramid.

…

The [MASK] at the top of the pyramid.

LM

pyramid

$P_\theta(y \mid x, z)$

REALM: Training

Objective: maximize $\sum_{z \in \mathcal{Z}_\theta} P_\theta(z \mid q) P_\theta(y \mid q, z)$

q (=x)

Datastore

$\mathcal{Z}_\theta$: top-K retrieved documents

The pyramidion on top allows for less material higher up the pyramid.

$P_\theta(z \mid x)$

Back-propagation

The pyramidion on top … the pyramid.

…

The [MASK] at the top of the pyramid.

LM

pyramid

$P_\theta(y \mid x, z)$

REALM: Training

Objective: maximize $\sum_{z \in \mathcal{Z}_\theta} P_\theta(z \mid q) P_\theta(y \mid q, z)$

***q* (=*x*)**

Stale index;
Update every T steps

Datastore

$\mathcal{Z}_\theta$: top-K retrieved documents

The pyramidion on top allows for less material higher up the pyramid.

$P_{\theta_{\text{new}}}(z \mid x)$

The pyramidion on top … the pyramid.

…

The [MASK] at the top of the pyramid.

LM

pyramid

$P_{\theta_{\text{new}}}(y \mid x, z)$

Up-to-date parameters

REALM: Index update rate

How often should we update the retrieval index?

- Too often: expensive
- Too slowly: out-dated

REALM: Index update rate

How often should we update the retrieval index?

- Too often: expensive
- Too slowly: out-dated

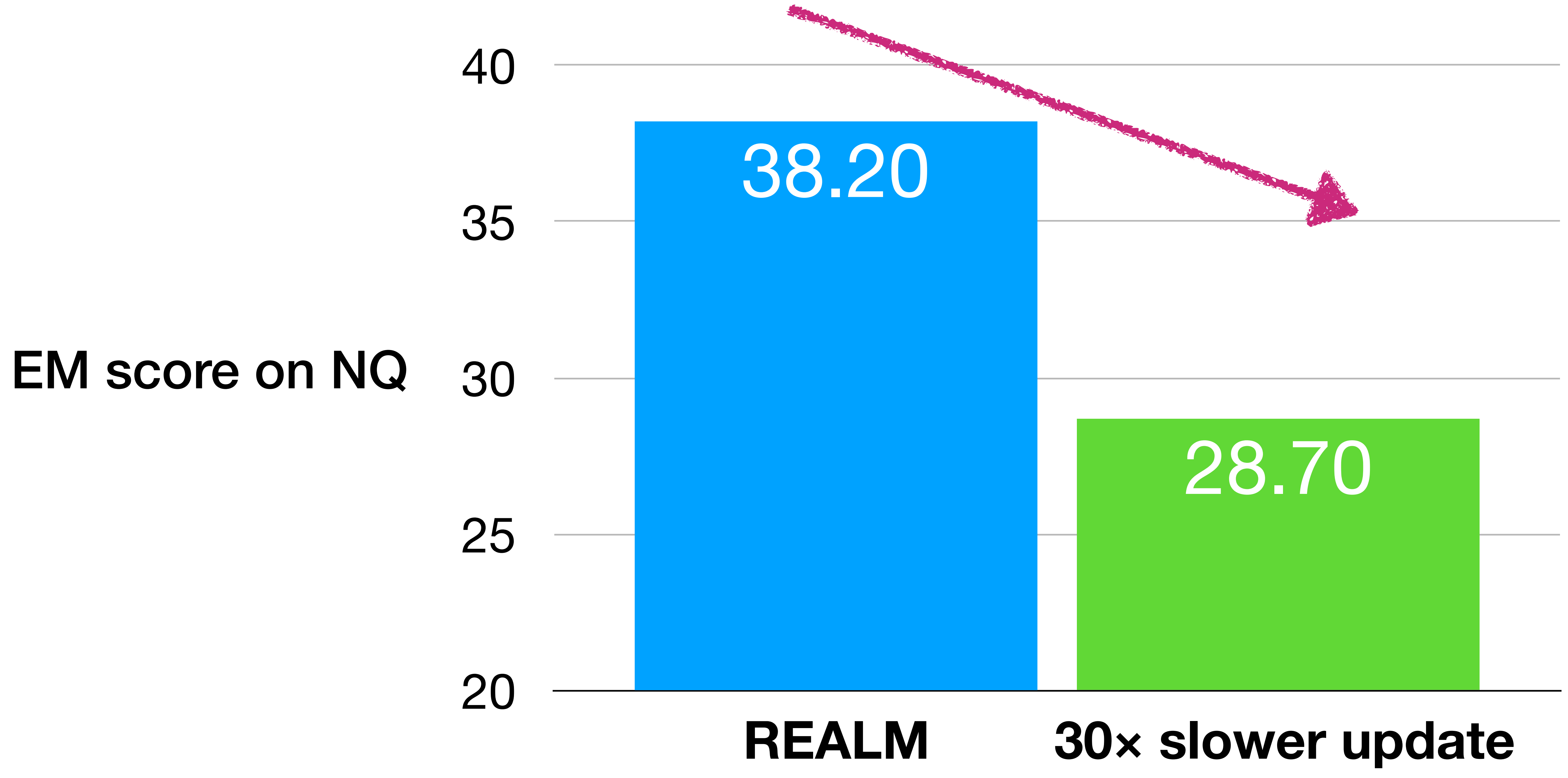
REALM: updating the index every 500 training steps

REALM: Index update rate

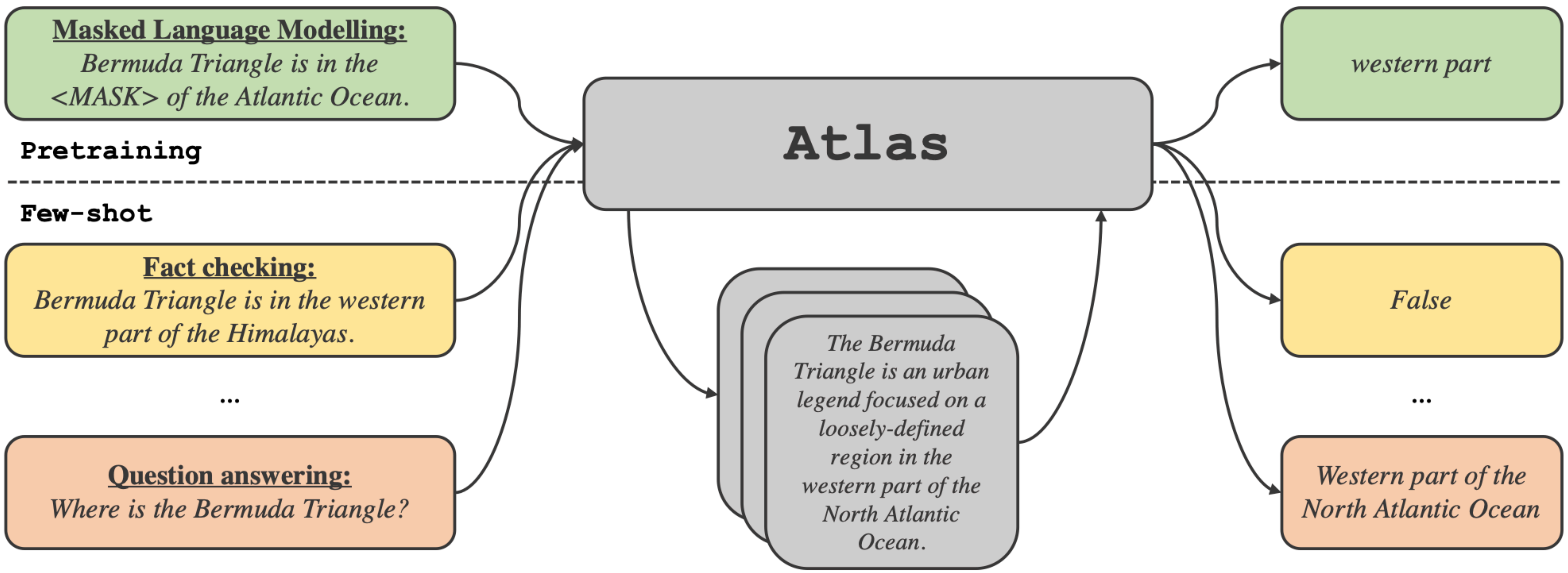
How often should we update the retrieval index?

- Too often: expensive
- Too slowly: out-dated

REALM: updating the index every 500 training steps



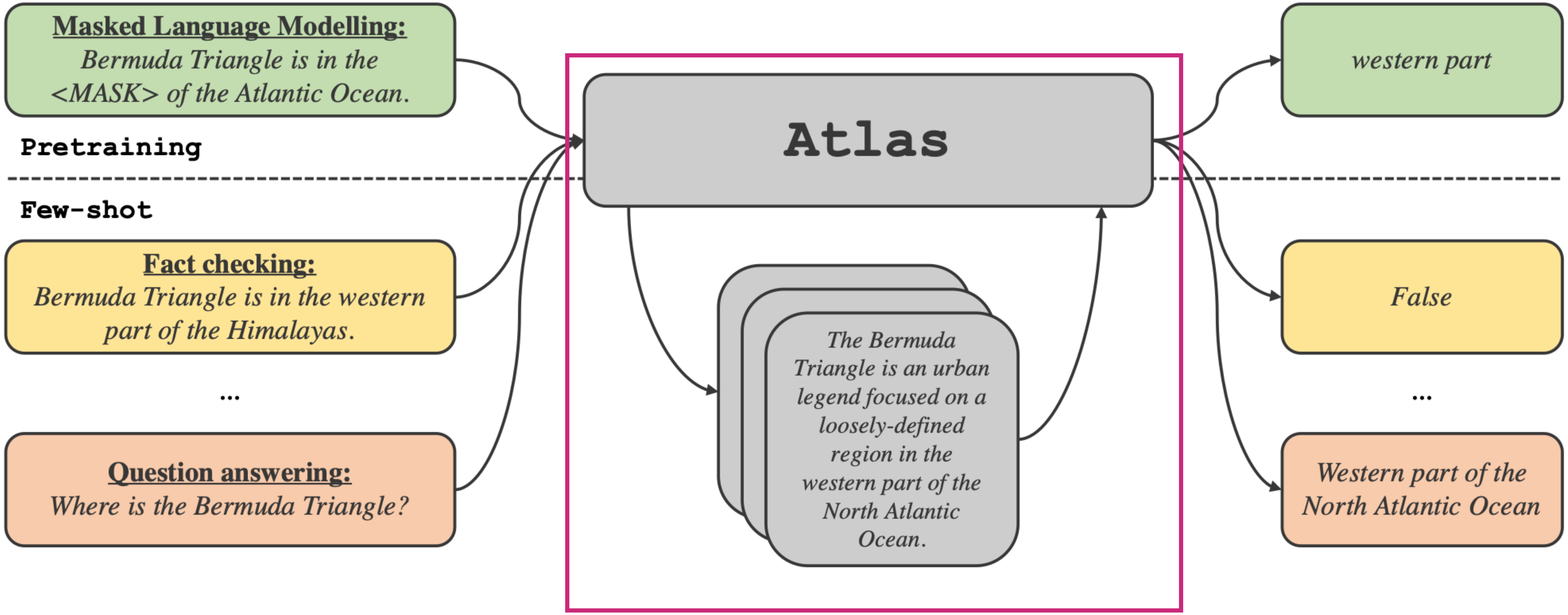
Atlas (Izacard et al. 2022)



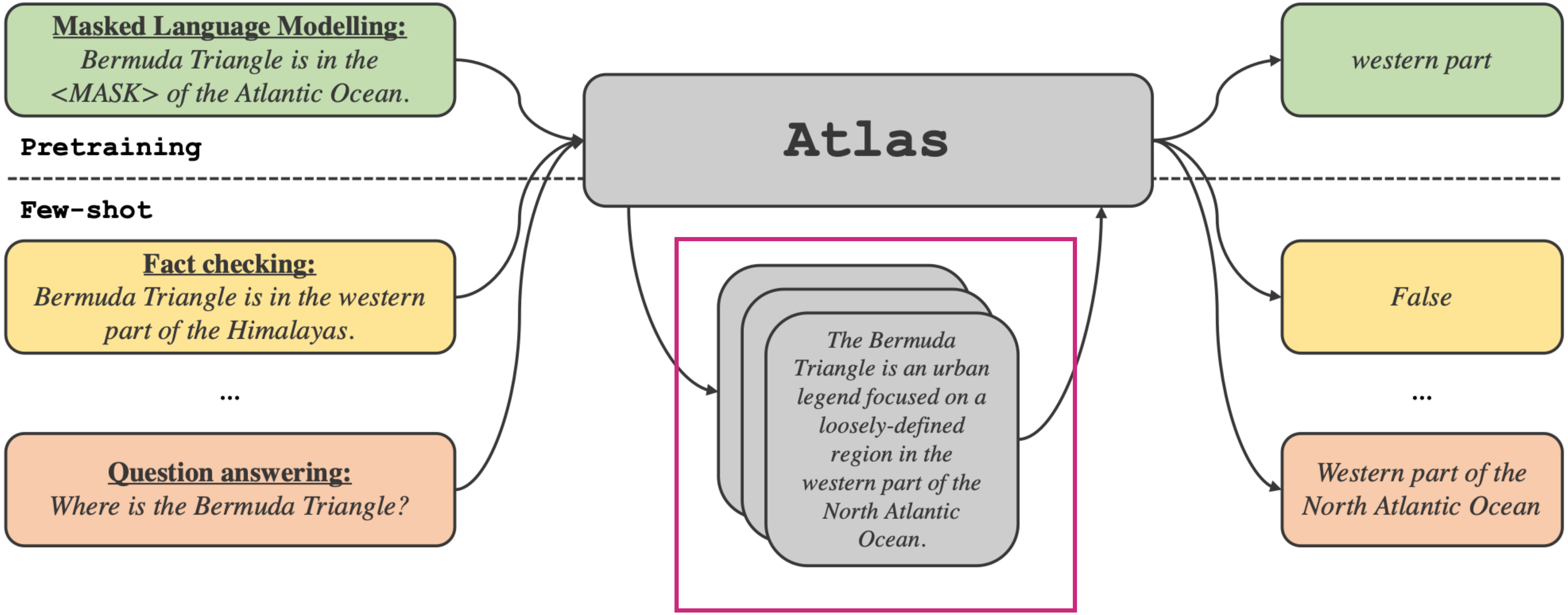
Izacard et al., 2022. "Atlas: Few-shot Learning with Retrieval Augmented Language Models"

Atlas (Izacard et al. 2022)

Retrieval-based generative seq2seq model



Atlas (Izacard et al. 2022)



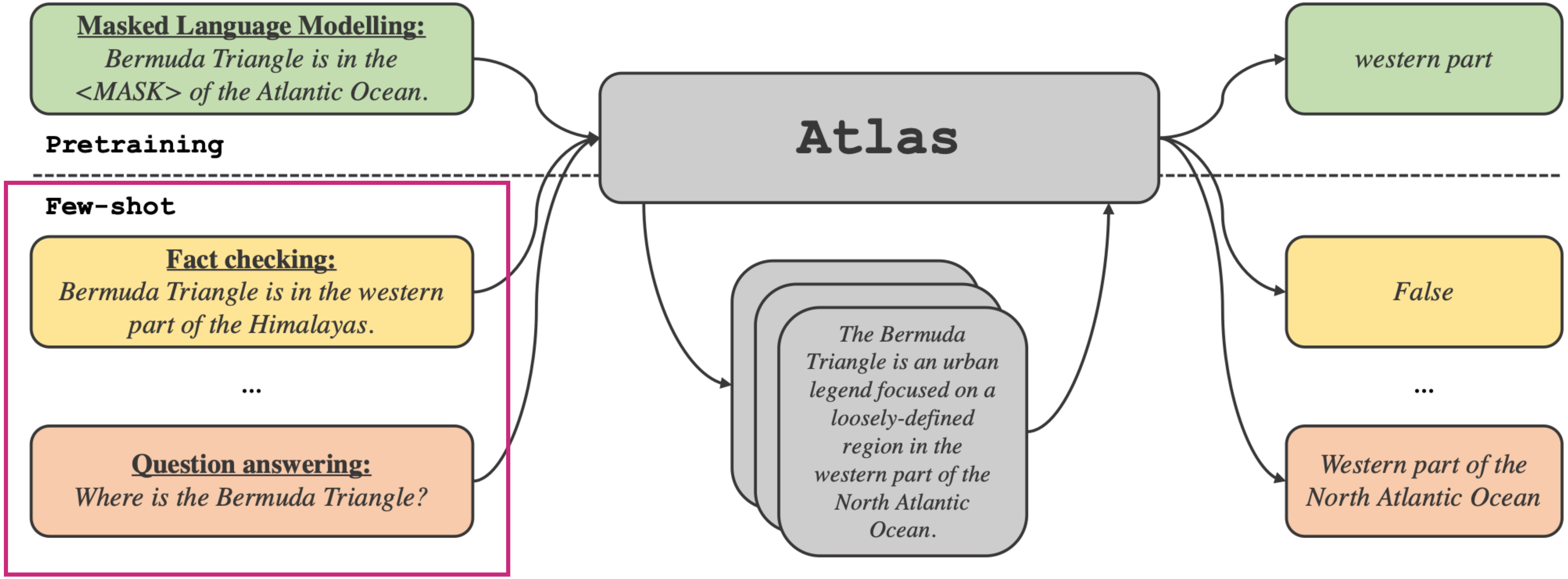
Process each doc independently using "Fusion-in-Decoder"

Atlas (Izacard et al. 2022)

"Fusion-in-decoder"

Izacard and Grave., 2020. "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering"

Atlas (Izacard et al. 2022)



Adapted to a lot of downstream tasks! (Section 5)

Atlas: Retriever training

Perplexity Distillation

Retrieve the text that can help LM encoders improve perplexity

$$P_{\text{retr}}(z \mid q) = \frac{\exp(s(z, q))}{\sum_{k=1}^{K} \exp(s(z_k, q))} \quad \longleftrightarrow \quad P_{\text{ppl}}(z \mid q, y) = \frac{\exp(\log P_{\text{LM}}(y \mid q, z))}{\sum_{k=1}^{K} \exp(\log P_{\text{LM}}(y \mid q, z_k))}$$

How likely each document is retrieved

How much each document improves the ppl

Atlas: Retriever training

Similarity based on retrieval encoder

KL Divergence

Prob of the gold labels if augmenting this text chunk

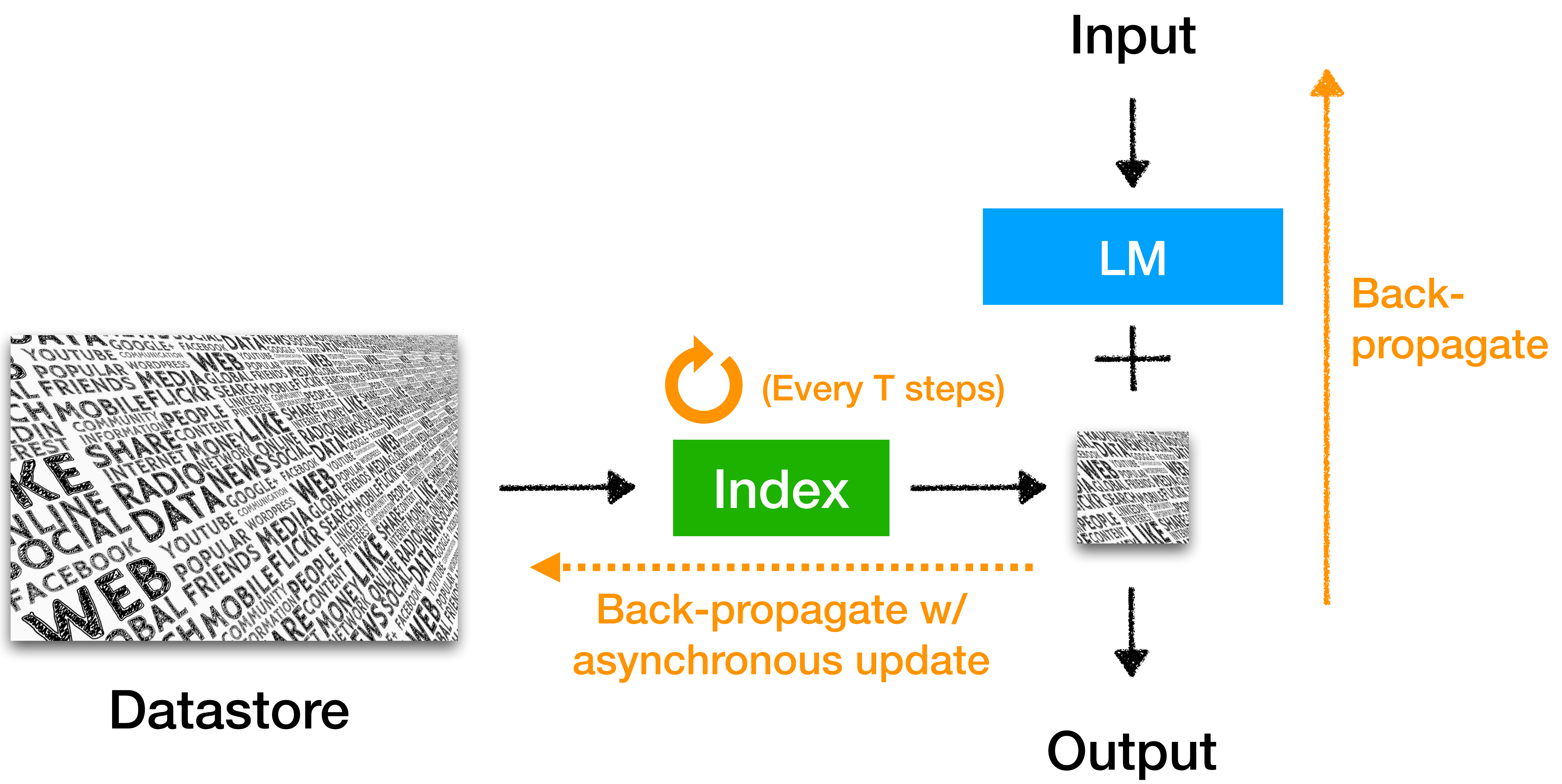
$$P_{\text{retr}}(z \mid q) = \frac{\exp(s(z, q))}{\sum_{k=1}^{K} \exp(s(z_k, q))} \longleftrightarrow P_{\text{ppl}}(z \mid q, y) = \frac{\exp(\log P_{\text{LM}}(y \mid q, z))}{\sum_{k=1}^{K} \exp(\log P_{\text{LM}}(y \mid q, z_k))}$$

How likely each document is retrieved

How much each document improves the ppl

Perplexity Distillation

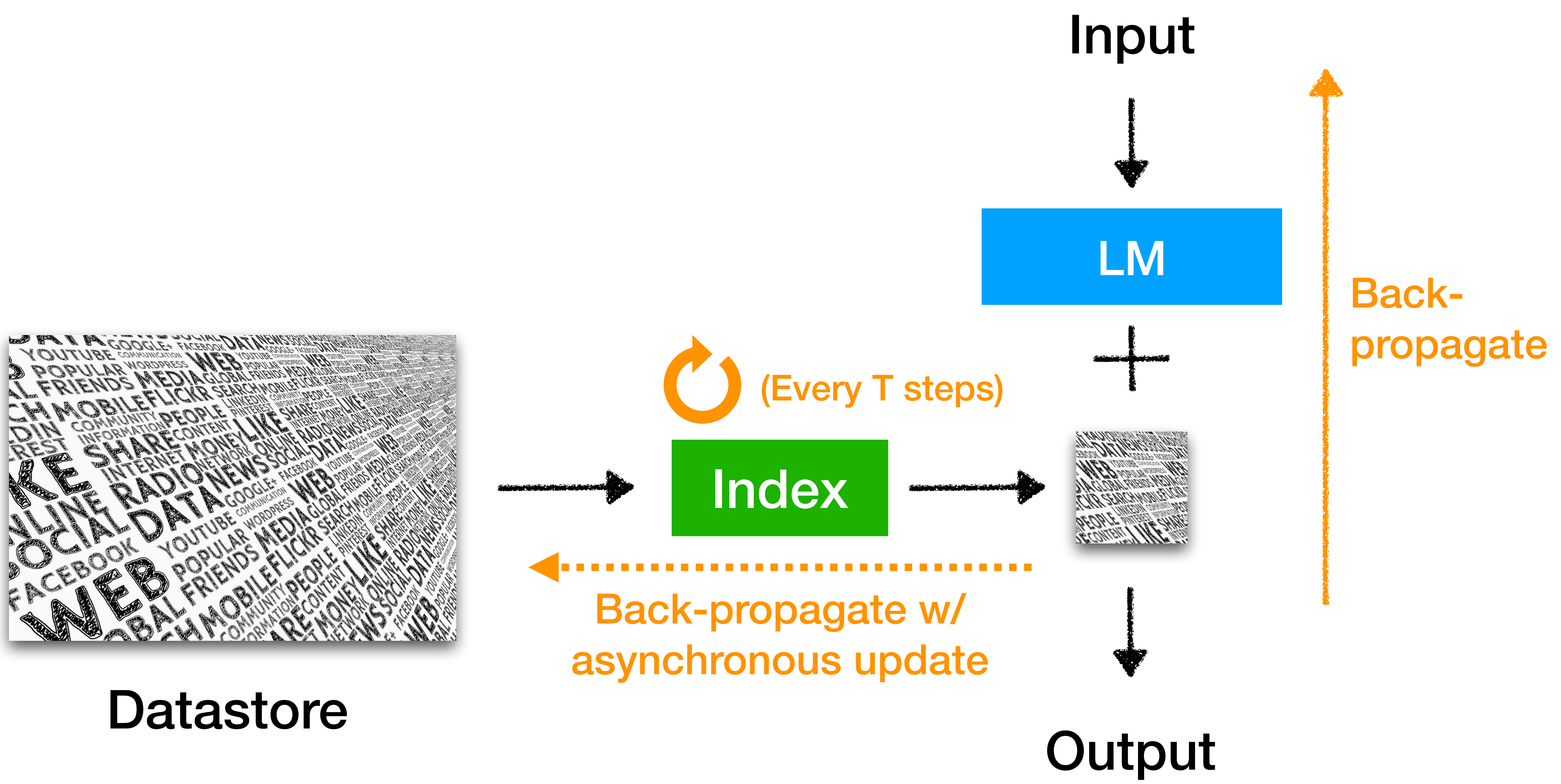
Atlas: Asynchronous index update



Update the index every T steps

30% overhead for asynchronous updating

Atlas: Asynchronous index update



Update the index every T steps

30% overhead for asynchronous updating

How can we get rid of this?

Training methods for retrieval-based LMs

- Independent training
- Sequential training
- Joint training w/ asynchronous index update
- **Joint training w/ in-batch approximation**

In-batch approximation

Full corpus

Encoder

Full index

Re-indexing will be very expensive!

Training batch

Encoder

In-batch index

Computed on the fly for each batch!

TRIME: Training with in-batch memory (Zhong et al. 2022)

Similar to kNN-LM

Datastore $\mathcal{D} = \{(k, v)\}$

context next token

Keys	Values
10/10, would buy this	*cheap*
Item delivered broken. Very	*cheap*
To check the version of PyTorch, you can use	*torch*
You are permitted to bring a	*torch*
A group of infections … one of the	*torch*

Zhong et al., 2022. "Training Language Models with Memory Augmentation"

TRIME: Training with in-batch memory (Zhong et al. 2022)

Similar to kNN-LM

Datastore $\mathscr{D} = \{(\underbrace{k}_{\text{context}}, \underbrace{v}_{\text{next token}})\}$

Keys	Values
10/10, would buy this	*cheap*
Item delivered broken. Very	*cheap*
To check the version of PyTorch, you can use	*torch*
You are permitted to bring a	*torch*
A group of infections … one of the	*torch*

Inference

$$P(y \mid x) \propto \underbrace{\boxed{\exp(E^\top f(x))}}_{\text{output embedding}} + \sum_{\underbrace{\boxed{(k,v)\in\mathscr{D}}}_{\text{datastore}}} \mathbb{I}[v = y]\exp(-d(\mathrm{Enc}(k), \mathrm{Enc}(x)))$$

output embedding

(same as standard LMs)

datastore

(very large!)

TRIME: Training with in-batch memory (Zhong et al. 2022)

Similar to kNN-LM

Datastore $\mathcal{D} = \{(k, v)\}$

context next token

Keys	Values
10/10, would buy this	*cheap*
Item delivered broken. Very	*cheap*
To check the version of PyTorch, you can use	*torch*
You are permitted to bring a	*torch*
A group of infections … one of the	*torch*

Inference

$$P(y \mid x) \propto \boxed{\exp(E^\top f(x))} + \sum_{\boxed{(k,v)\in\mathcal{D}}} \mathbb{I}[v = y]\exp(-d(\mathrm{Enc}(k), \mathrm{Enc}(x)))$$

output embedding

(same as standard LMs)

datastore

(very large!)

1. Aligning the output representations with static embeddings

2. Aligning input context with all the contexts in datastore that share the same next token

TRIME: Training

Key idea: build a temporary index from <u>same training batch</u> on the fly

$$P(y \mid x) \propto \exp(E^\top f(x)) + \sum_{(k,v)\in \boxed{\mathscr{D}_{train}}} \mathbb{I}[v = y]\big(-d(\mathrm{Enc}(k), \mathrm{Enc}(x))\big)$$

In-batch approximation

(built from in-batch examples on the fly)

TRIME: Training

Key idea: build a temporary index from <u>same training batch</u> on the fly

$$P(y \mid x) \propto \exp(E^\top f(x)) + \sum_{(k,v)\in \boxed{\mathcal{D}_{train}}} \mathbb{I}[v = y]\big(-d(\mathrm{Enc}(k), \mathrm{Enc}(x))\big)$$

In-batch approximation

(built from in-batch examples on the fly)

We can back-propagate to all the representations in datastore $\mathcal{D}_{train}$!

TRIME: full index vs. in-batch index

Full corpus

Keys	Values
To check the version of PyTorch, you can use	*torch*
Item delivered broken. Very	*cheap*
He moves to	*Apple*
Apple merged with NeXT, and Jobs became	*CEO*
…	…

Full index (used during inference)

TRIME: full index vs. in-batch index

Full corpus

Keys	Values
To check the version of PyTorch, you can use	*torch*
Item delivered broken. Very	*cheap*
He moves to	*Apple*
Apple merged with NeXT, and Jobs became	*CEO*
…	…

Full index (used during inference)

Compute on the fly!

Apple merged with NeXT, and …

VS Code was developed by Microsoft for Windows in 2015 …

He moves to Apple …

…

Training batch

Keys	Values
Jobs	*became*
Apple merged with NeXT, and Jobs became	*CEO*
…	…
He moves to	*Apple*
VS Code was developed	*By*

In-batch index (used during training)

TRIME: full index vs. in-batch index

How to batch training data —
so we can have good in-batch examples?

Compute on the fly!

Apple merged with NeXT, and …

VS Code was developed by Microsoft for Windows in 2015 …

He moves to Apple …

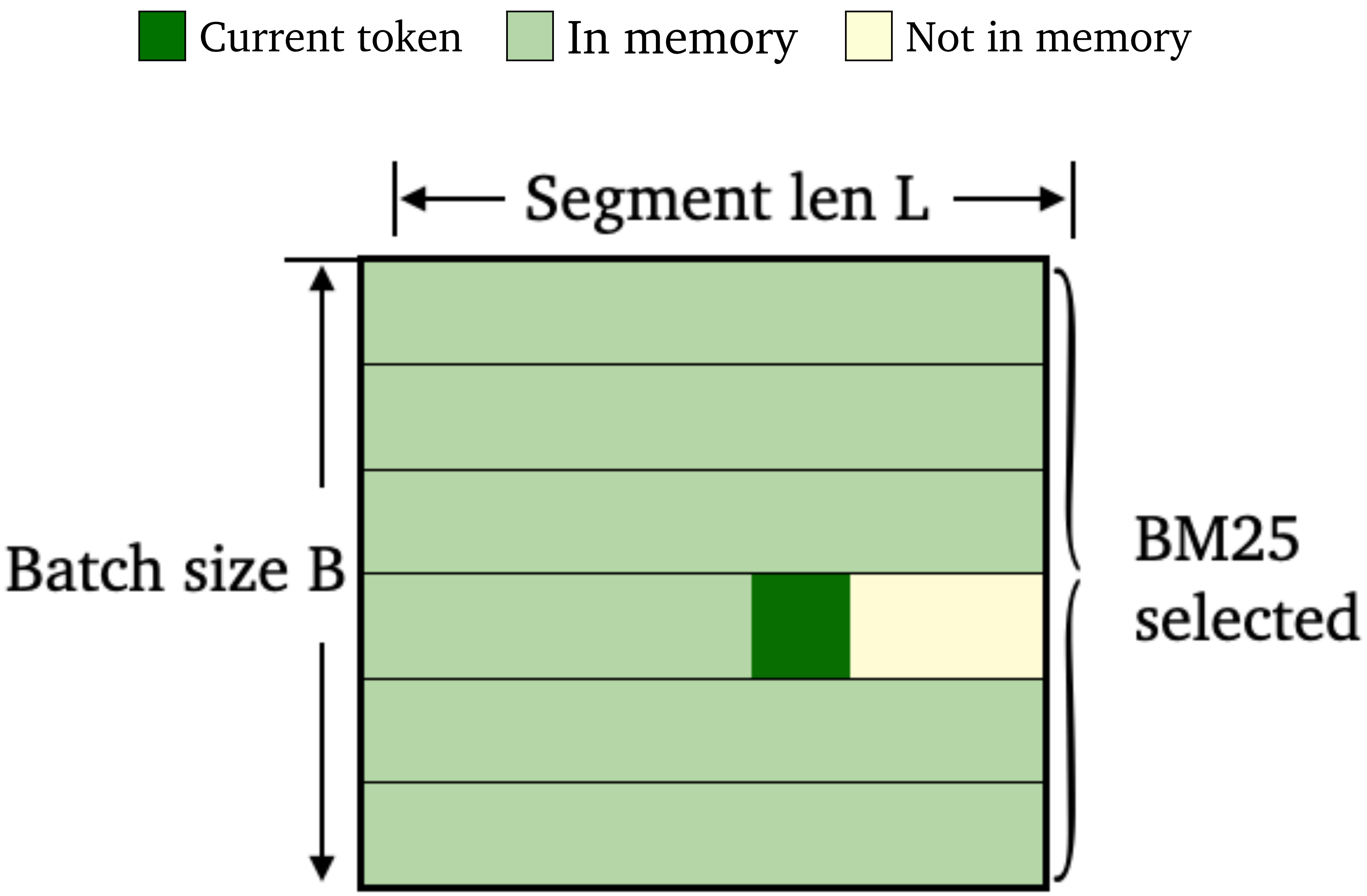
…

Training batch

Keys	Values
Jobs	*became*
Apple merged with NeXT, and Jobs became	*CEO*
…	…
He moves to	*Apple*
VS Code was developed	*By*

In-batch index (used during training)

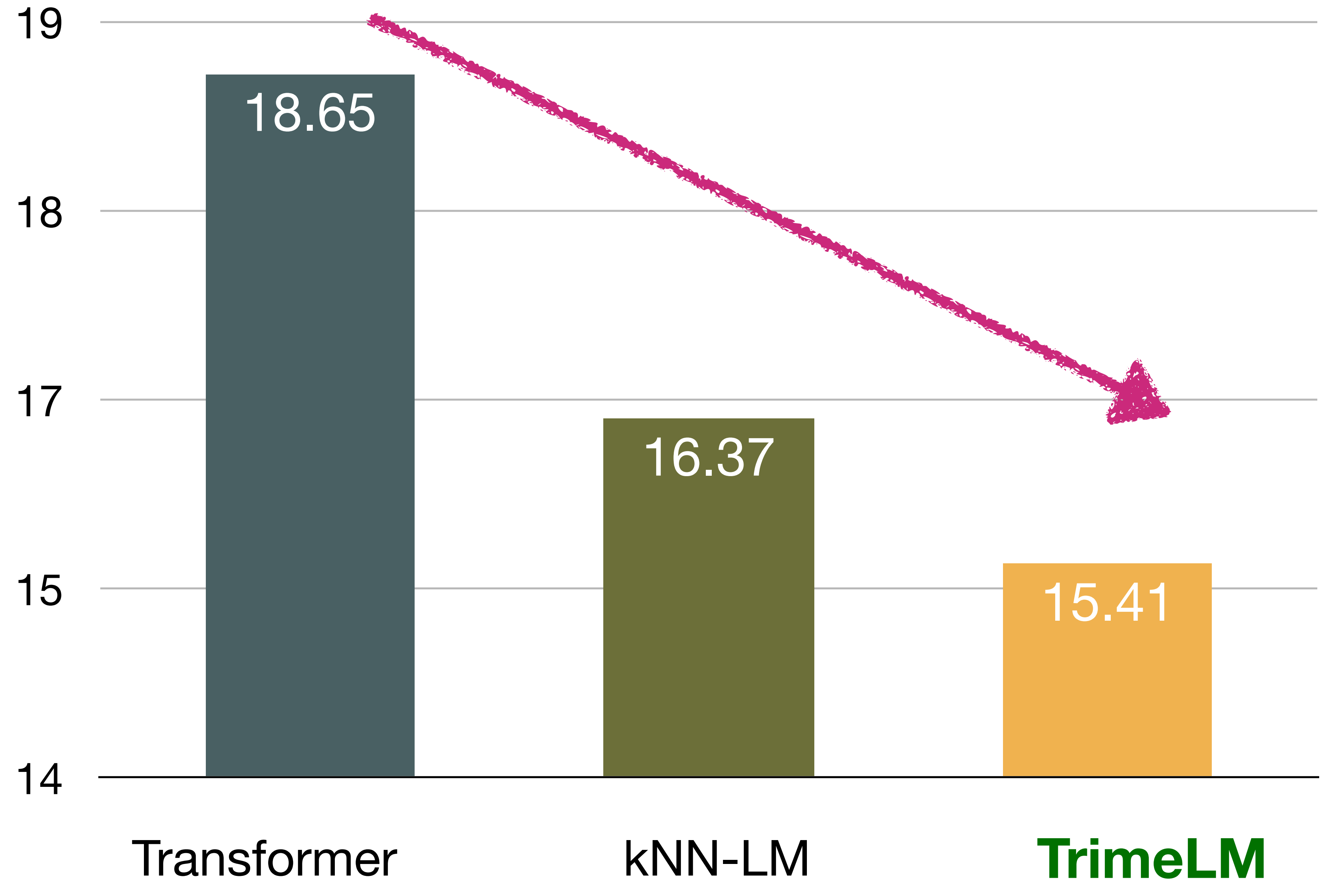
TRIME: Data batching strategy



Key idea: similar contexts — more training signals from in-batch examples!

Use **BM25** scores to find similar contexts

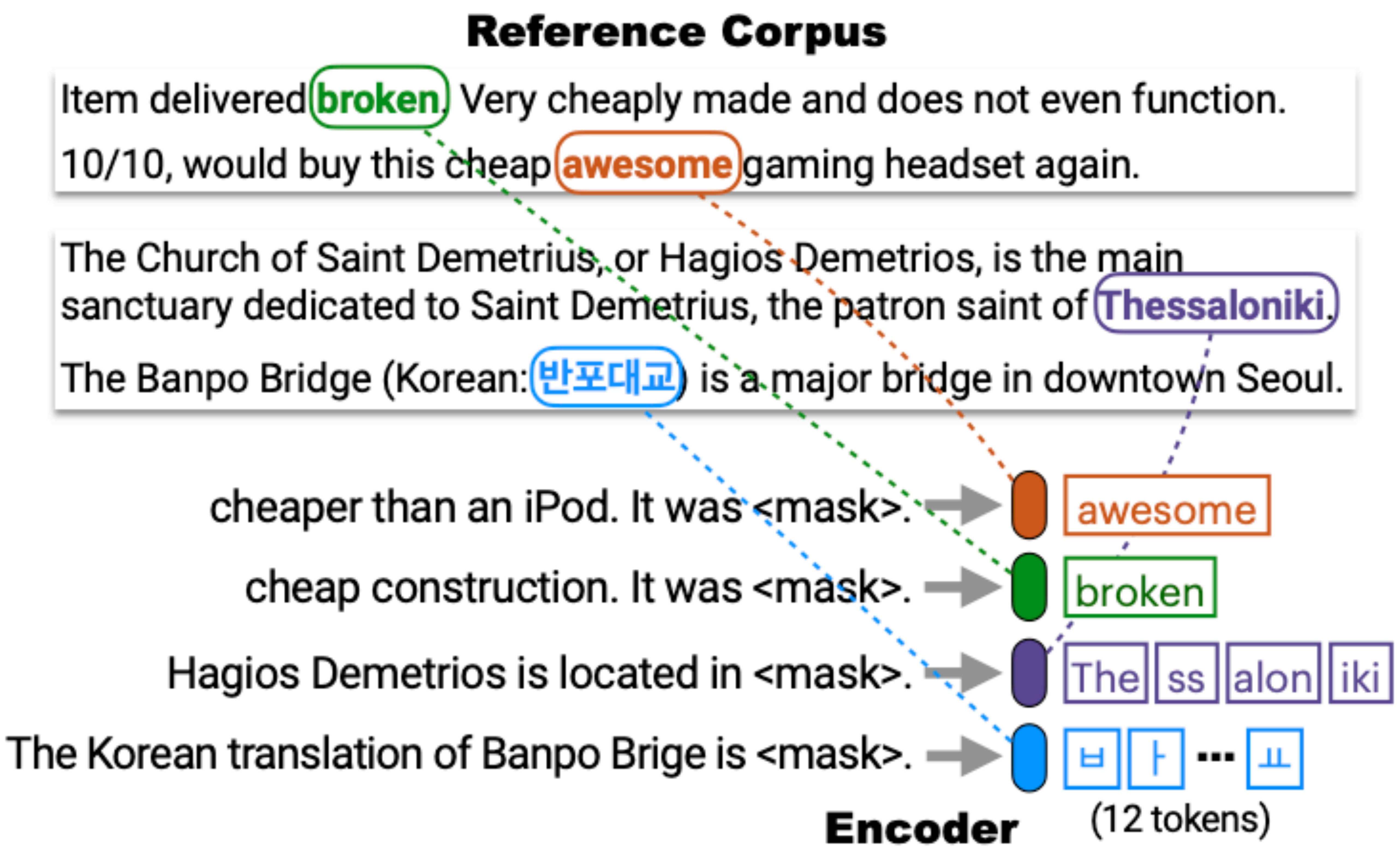
TRIME: Results



Perplexity on Wikitext-103

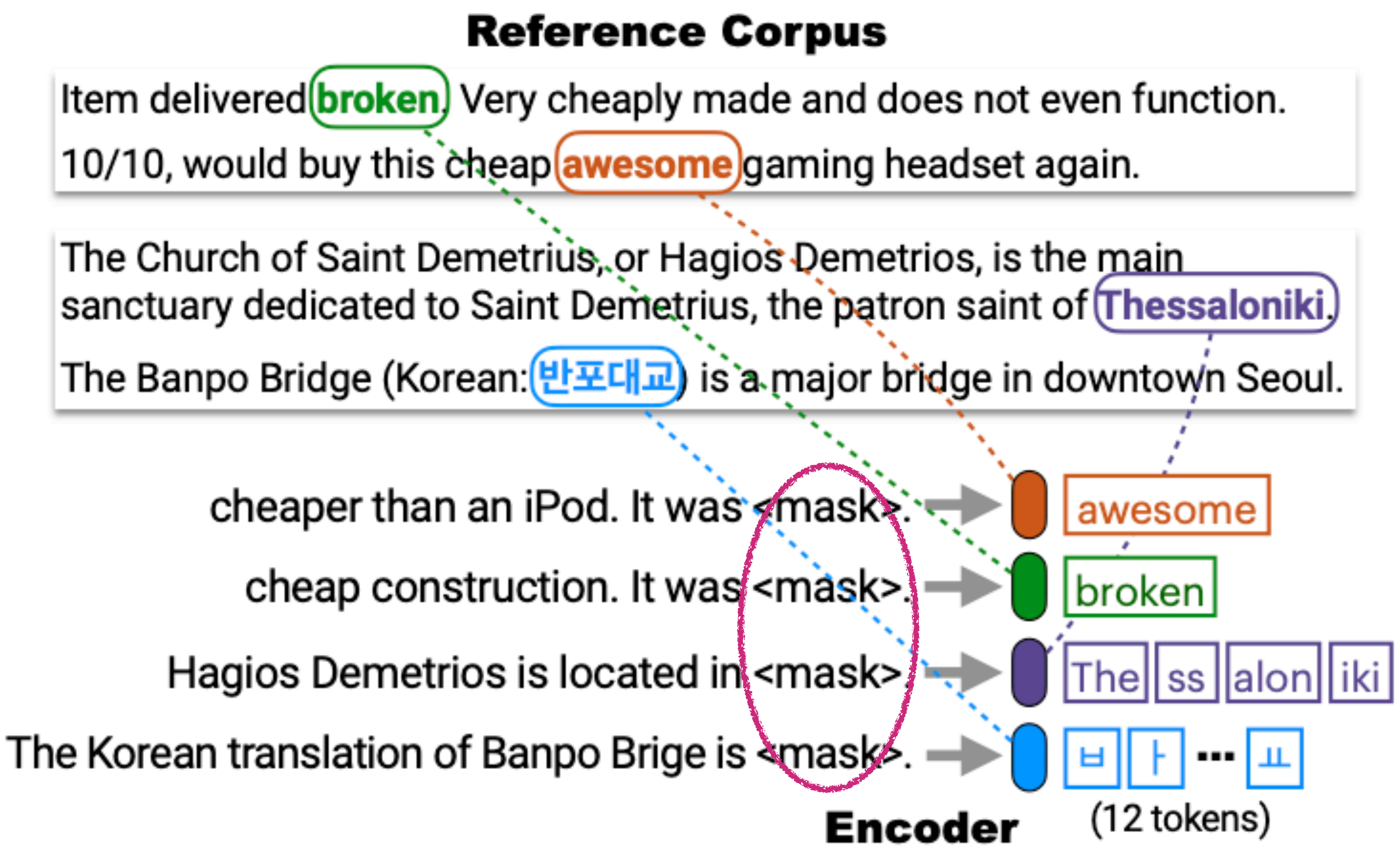
Model size: 247M
Sequence len: 3072

NPM: Nonparametric masked LMs (Min et al. 2023)



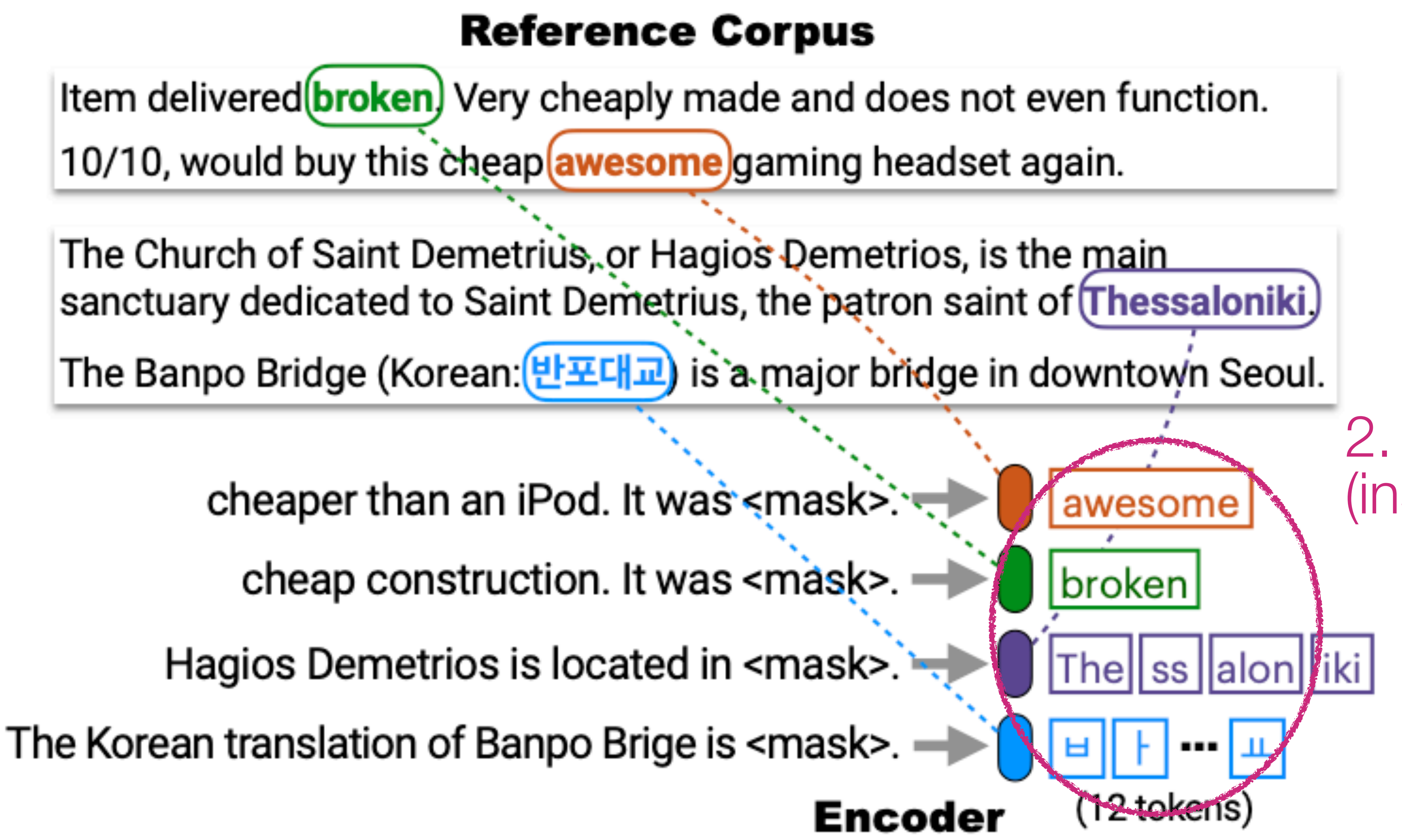
Min et al., 2023. "Nonparametric Masked Language Modeling"

NPM: Nonparametric masked LMs (Min et al. 2023)



1. **masked** language model pertained on >1B tokens

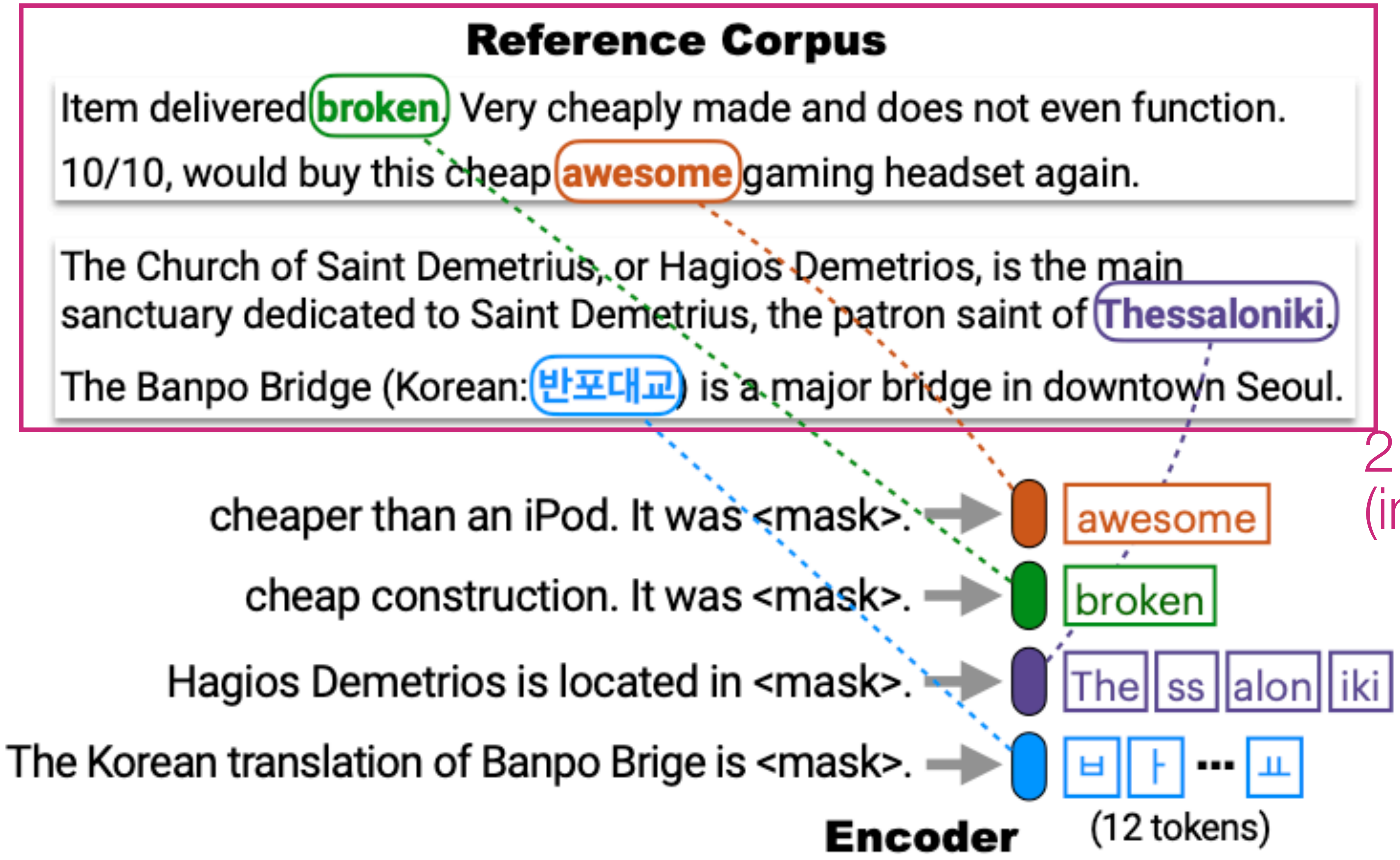
NPM: Nonparametric masked LMs (Min et al. 2023)



2. Each mask corresponds to a **phrase** (instead of a token)

1. masked language model pertained on >1B tokens

NPM: Nonparametric masked LMs (Min et al. 2023)



3. During inference, predictions are made **purely** according to retrieval results

2. Each mask corresponds to a phrase (instead of a token)

1. masked language model pertained on >1B tokens

NPM: Nonparametric masked LMs (Min et al. 2023)

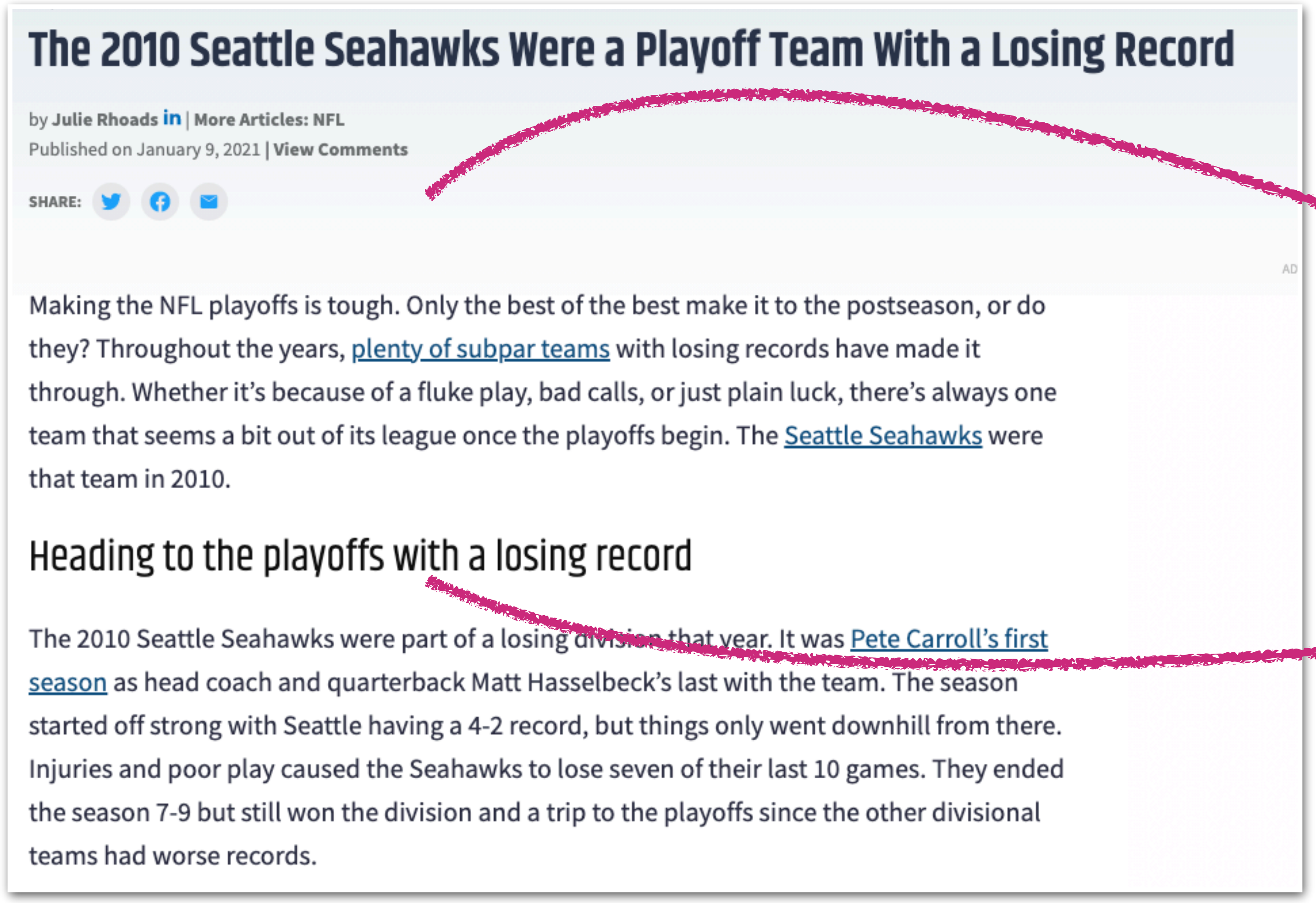
Key challenge: how to train the retrieval-based model on >1B tokens?

1. How to approximate full corpus retrieval
2. How to choose positives and negatives

In-batch approximation with same-doc batching

NPM: Training

1. Sample sequences from the same document



The 2010 Seattle Seahawks Were a Playoff Team With a Losing Record

by Julie Rhoads in | More Articles: NFL
Published on January 9, 2021 | View Comments

SHARE:

Making the NFL playoffs is tough. Only the best of the best make it to the postseason, or do they? Throughout the years, plenty of subpar teams with losing records have made it through. Whether it's because of a fluke play, bad calls, or just plain luck, there's always one team that seems a bit out of its league once the playoffs begin. The Seattle Seahawks were that team in 2010.

Heading to the playoffs with a losing record

The 2010 Seattle Seahawks were part of a losing division that year. It was Pete Carroll's first season as head coach and quarterback Matt Hasselbeck's last with the team. The season started off strong with Seattle having a 4-2 record, but things only went downhill from there. Injuries and poor play caused the Seahawks to lose seven of their last 10 games. They ended the season 7-9 but still won the division and a trip to the playoffs since the other divisional teams had worse records.

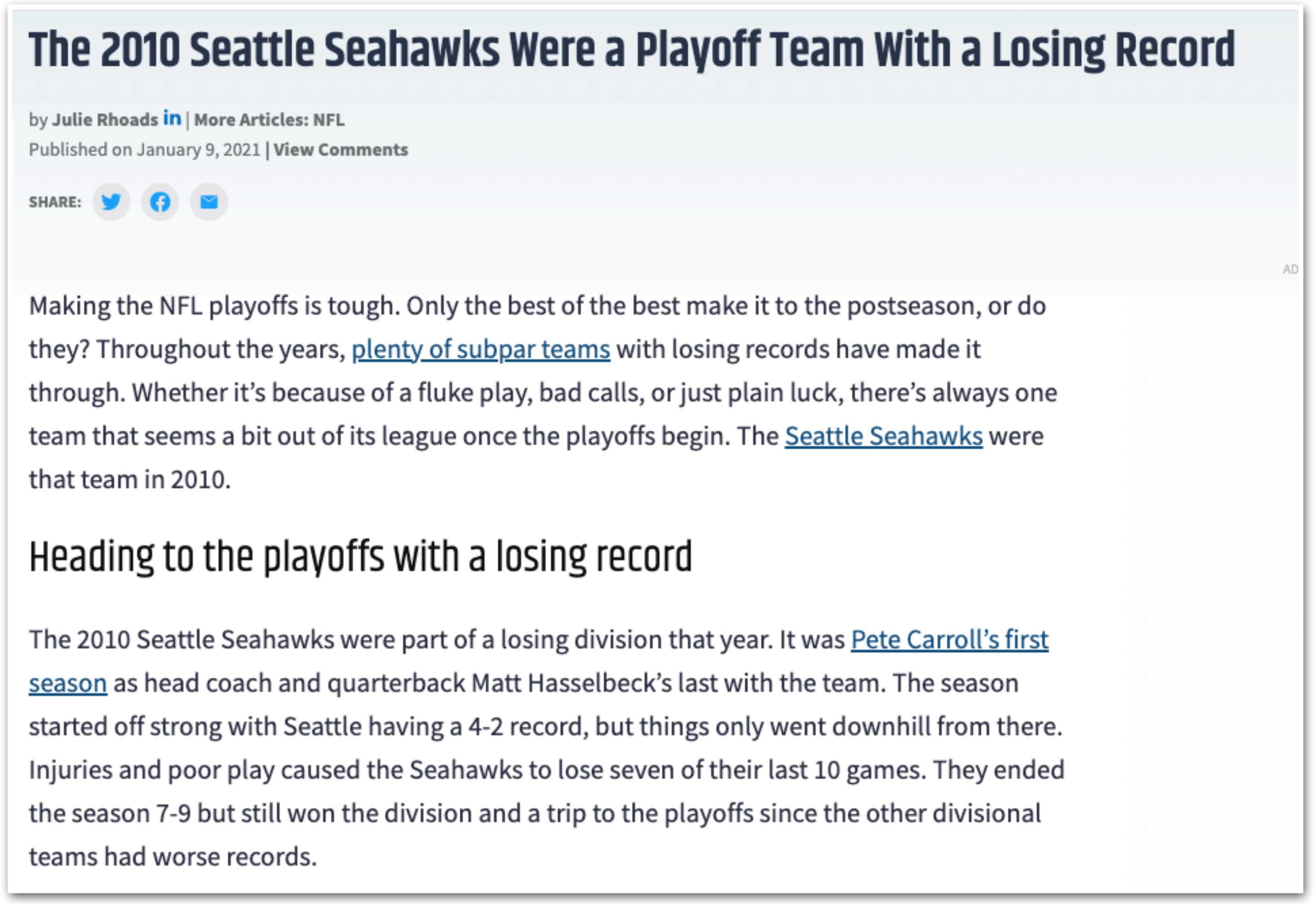
In the 2010 NFL season, the Seattle Seahawks made history by making it into the playoffs despite having a 7–9 record.

... against the Seattle Seahawks as a member of (...) In the 2010 season, the Seahawks became the first team in NFL history to ...

For simplicity, we assume 2 sequences in a batch

NPM: Training

2. Identify co-occurring spans



The 2010 Seattle Seahawks Were a Playoff Team With a Losing Record

by Julie Rhoads | More Articles: NFL
Published on January 9, 2021 | View Comments

SHARE:

Making the NFL playoffs is tough. Only the best of the best make it to the postseason, or do they? Throughout the years, plenty of subpar teams with losing records have made it through. Whether it's because of a fluke play, bad calls, or just plain luck, there's always one team that seems a bit out of its league once the playoffs begin. The Seattle Seahawks were that team in 2010.

Heading to the playoffs with a losing record

The 2010 Seattle Seahawks were part of a losing division that year. It was Pete Carroll's first season as head coach and quarterback Matt Hasselbeck's last with the team. The season started off strong with Seattle having a 4-2 record, but things only went downhill from there. Injuries and poor play caused the Seahawks to lose seven of their last 10 games. They ended the season 7-9 but still won the division and a trip to the playoffs since the other divisional teams had worse records.

In the 2010 NFL season, **the Seattle Seahawks** made history by making it into the playoffs despite having a 7–9 record.

... against **the Seattle Seahawks** as a member of (...) In the 2010 season, the Seahawks became the first team in NFL history to ...

For simplicity, we assume 2 sequences in a batch

NPM: Training

3. One is “positive” for the other

The 2010 Seattle Seahawks Were a Playoff Team With a Losing Record

by **Julie Rhoads** | **More Articles: NFL**
Published on January 9, 2021 | **View Comments**

SHARE:

Making the NFL playoffs is tough. Only the best of the best make it to the postseason, or do they? Throughout the years, plenty of subpar teams with losing records have made it through. Whether it’s because of a fluke play, bad calls, or just plain luck, there’s always one team that seems a bit out of its league once the playoffs begin. The Seattle Seahawks were that team in 2010.

Heading to the playoffs with a losing record

The 2010 Seattle Seahawks were part of a losing division that year. It was Pete Carroll’s first season as head coach and quarterback Matt Hasselbeck’s last with the team. The season started off strong with Seattle having a 4-2 record, but things only went downhill from there. Injuries and poor play caused the Seahawks to lose seven of their last 10 games. They ended the season 7-9 but still won the division and a trip to the playoffs since the other divisional teams had worse records.

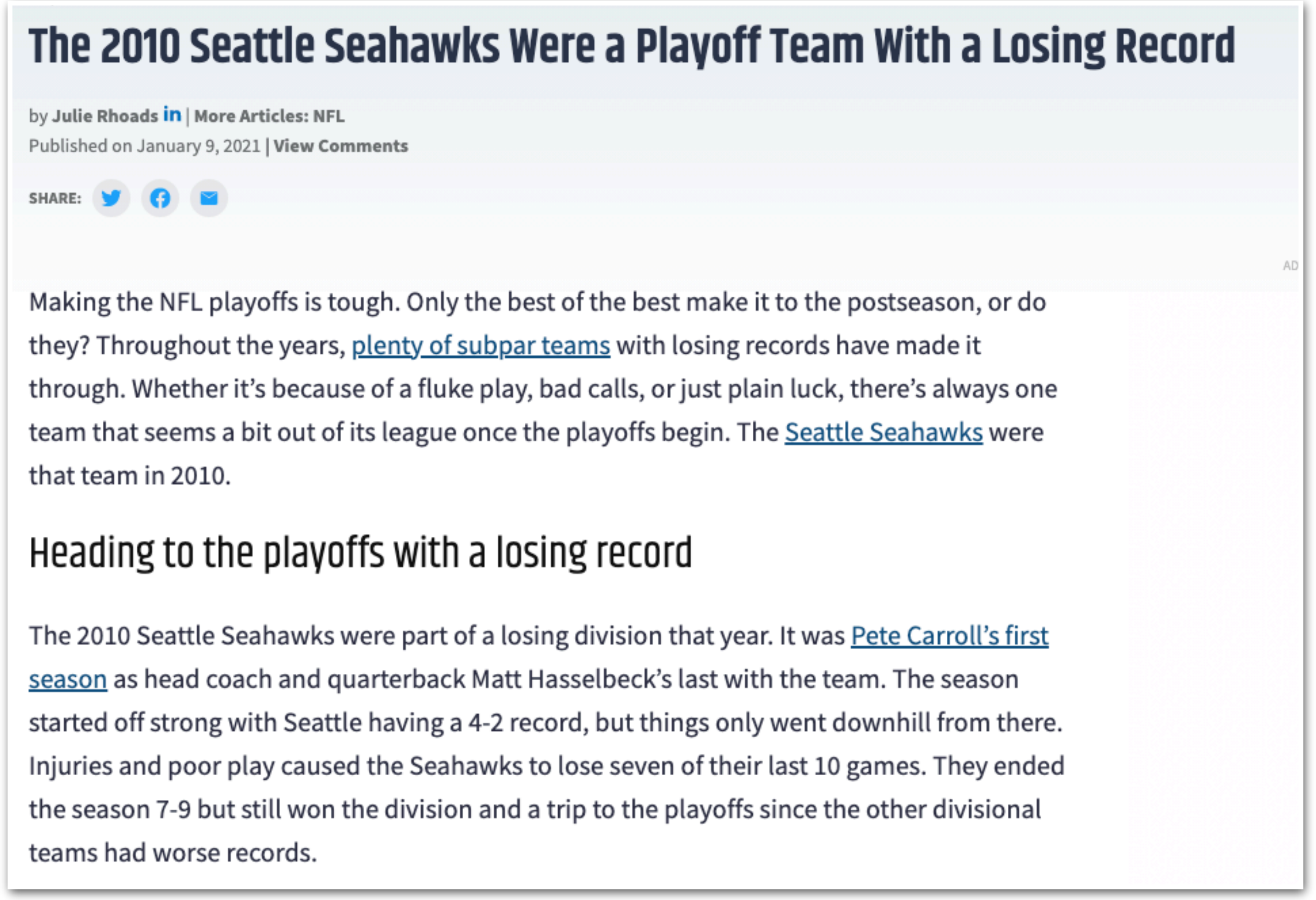
In the 2010 NFL season, ____________ made history by making it into the playoffs despite having a 7–9 record.

... against **the Seattle Seahawks** as a member of (...) In the 2010 season, the Seahawks became the first team in NFL history to ...

For simplicity, we assume 2 sequences in a batch

NPM: Training

4. The others are "negatives"



The 2010 Seattle Seahawks Were a Playoff Team With a Losing Record

by Julie Rhoads | More Articles: NFL
Published on January 9, 2021 | View Comments

SHARE:

Making the NFL playoffs is tough. Only the best of the best make it to the postseason, or do they? Throughout the years, plenty of subpar teams with losing records have made it through. Whether it's because of a fluke play, bad calls, or just plain luck, there's always one team that seems a bit out of its league once the playoffs begin. The Seattle Seahawks were that team in 2010.

Heading to the playoffs with a losing record

The 2010 Seattle Seahawks were part of a losing division that year. It was Pete Carroll's first season as head coach and quarterback Matt Hasselbeck's last with the team. The season started off strong with Seattle having a 4-2 record, but things only went downhill from there. Injuries and poor play caused the Seahawks to lose seven of their last 10 games. They ended the season 7-9 but still won the division and a trip to the playoffs since the other divisional teams had worse records.

In the 2010 NFL season, ____________ made history by making it into the playoffs despite having a 7–9 record.

positive

… against **the Seattle Seahawks** as a member of (…) In the 2010 season, the Seahawks became the first team in NFL history to …

For simplicity, we assume 2 sequences in a batch

Beyond lexical clues?

In TRIME and NPM, retrieval models are trained to use ***lexical*** information

Positives: **co-occurring** tokens/spans

Can we do more than that?

RPT: Retrieval-pretrained transformer
(Rubin and Berant 2023)

Reference score P("Apple" | "Jobs become CEO of", "NeXT merged with …")

Reference chunk

Rubin and Berant, 2023. "Long-range Language Modeling with Self-retrieval"

RPT: Retrieval-pretrained transformer
(Rubin and Berant 2023)

Reference score P(“Apple” | "Jobs become CEO of”, “NeXT merged with …”)

Reference chunk

P(“Apple” | "Jobs become CEO of”, “He joined his former …”) **>** **Reference score**

Positive chunks

Rubin and Berant, 2023. “Long-range Language Modeling with Self-retrieval”

RPT: Retrieval-pretrained transformer
(Rubin and Berant 2023)

Reference score P("Apple" | "Jobs become CEO of", "NeXT merged with …")

Reference chunk

P("Apple" | "Jobs become CEO of", "He joined his former …") **>** **Reference score**

Positive chunks

P("Apple" | "Jobs become CEO of", "Jobs was raised …") **<** **Reference score**

Negative chunks

Rubin and Brent, 2023. "Long-range Language Modeling with Self-retrieval"

Joint training

- 👍 End-to-end trained — each component is optimized
- 👍 Good performance
- 👎 Training is more complicated
- 👎 Train-test discrepancy still remains

Summary

Training method	👍	👎
Independent training (Ram et al 2023; Khandelwal et al 2020) Sequential training (Borgeaud et al 2021; Shi et al 2023)	✴ Easy to implement: off-the-shelf models ✴ Easy to improve: sub-module can be separately improved	✴ Models are not end-to-end trained — suboptimal performance
Joint training: async update (Guu et al 2020; Izacard et al 2022) Joint training: in-batch approx (Zhong et al 2022; Min et al 2023; Rubin and Berant 2023)	✴ End-to-end trained — very good performance!	✴ Training may be complicated (overhead, batching methods, etc) ✴ Train-test discrepancy still remains

How do retrieval-based language models perform on downstream tasks? → **Section 5**!