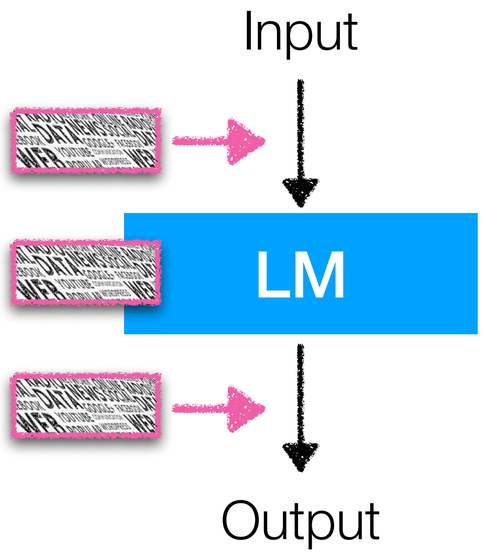
# Section 3:
# Retrieval-based LMs: Architecture

# Categorization of retrieval-based LMs

**What** to retrieve?

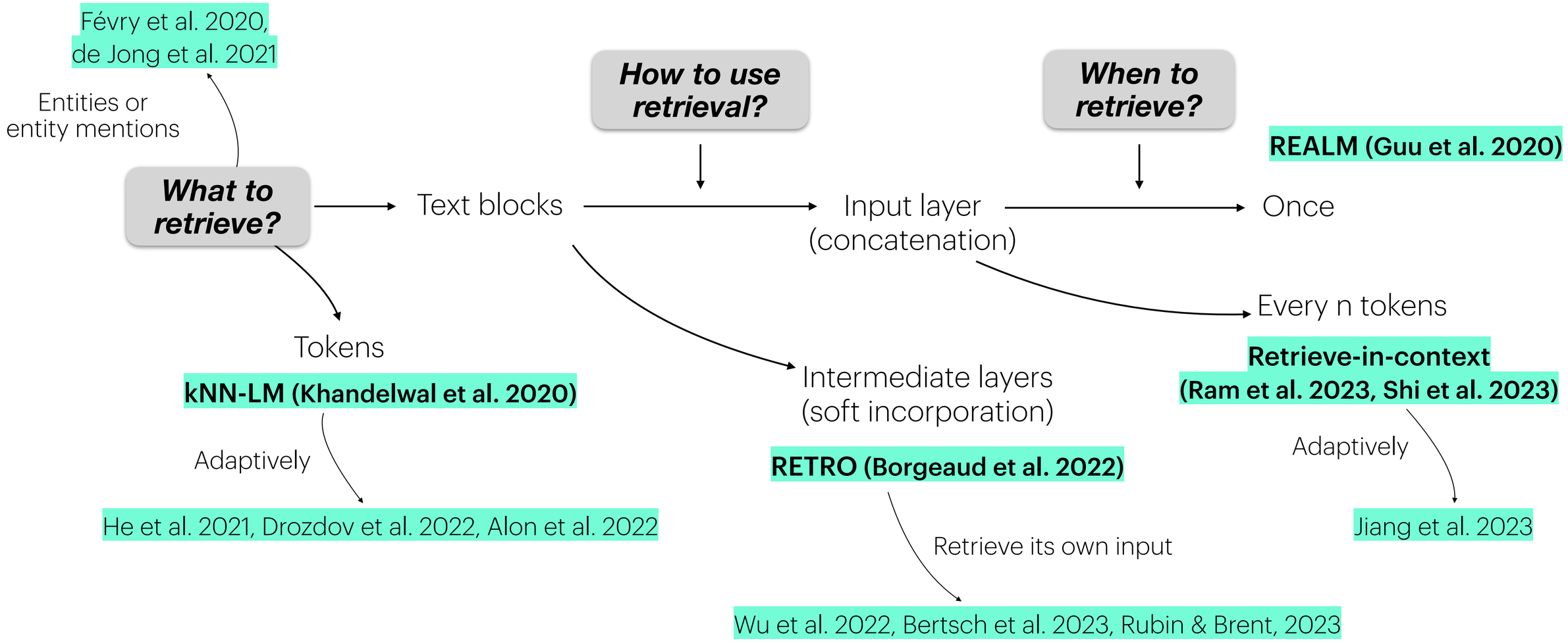

Chunks of text?
Tokens?
Something else?

**How** to use retrieval?




**When** to retrieve?

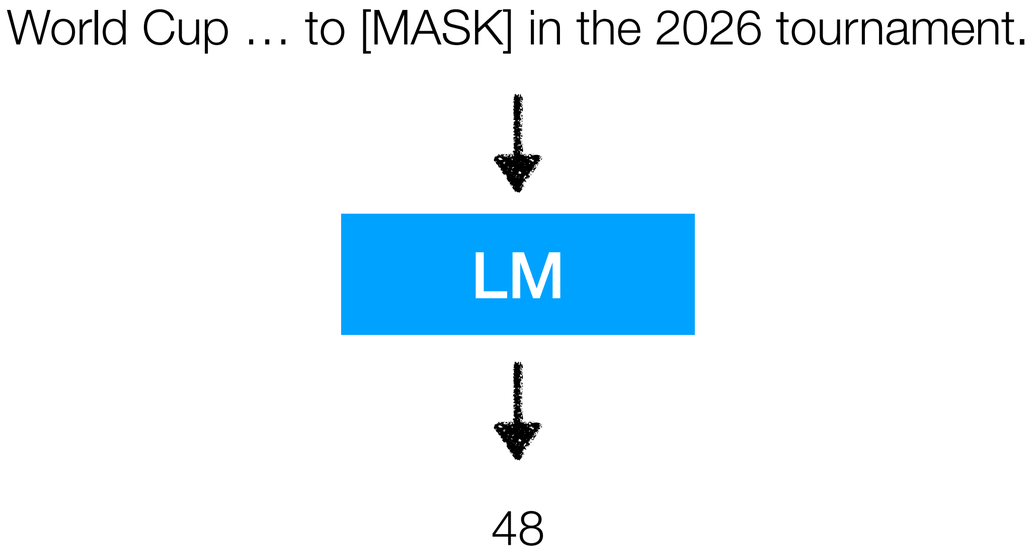
# Roadmap




**This is only about "architecture"**
**Section 4 will categorize & discuss "training"**

# REALM (Guu et al 2020)

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

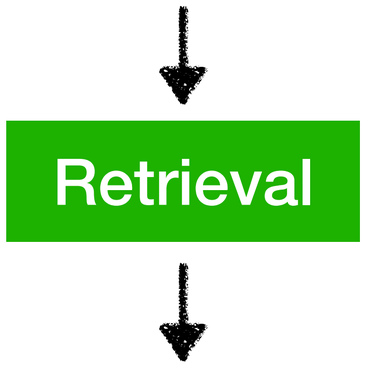



Guu et al. 2020. “REALM: Retrieval-Augmented Language Model Pre-Training”

# REALM (Guu et al 2020)

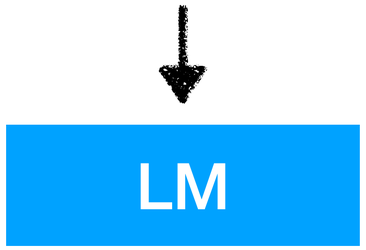
***x*** = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.

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




***k*** chunks of text


World Cup … to [MASK] in the 2026 tournament.




Guu et al. 2020. “REALM: Retrieval-Augmented Language Model Pre-Training”

# REALM (Guu et al 2020)

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


Guu et al. 2020. “REALM: Retrieval-Augmented Language Model Pre-Training”

# REALM: (1) Retrieve stage

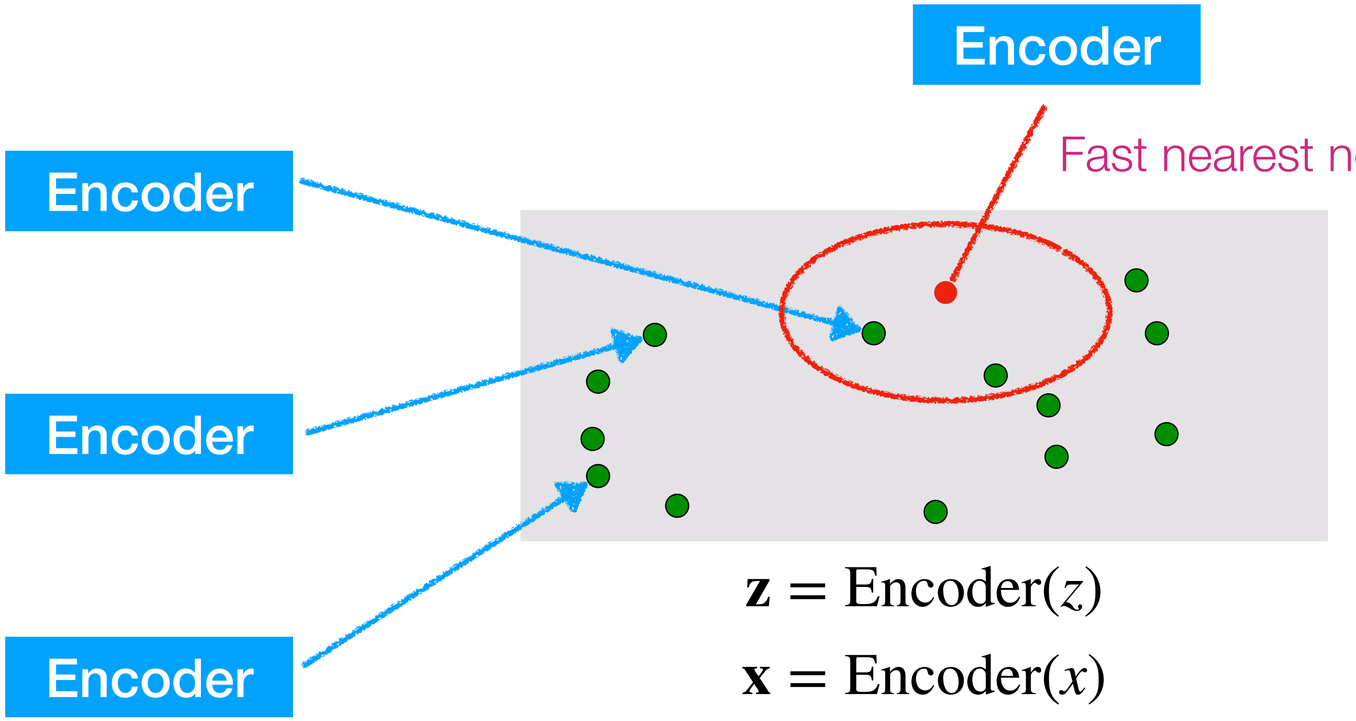
World Cup … to [MASK] in the 2026 tournament.

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
(called *documents* in the paper)



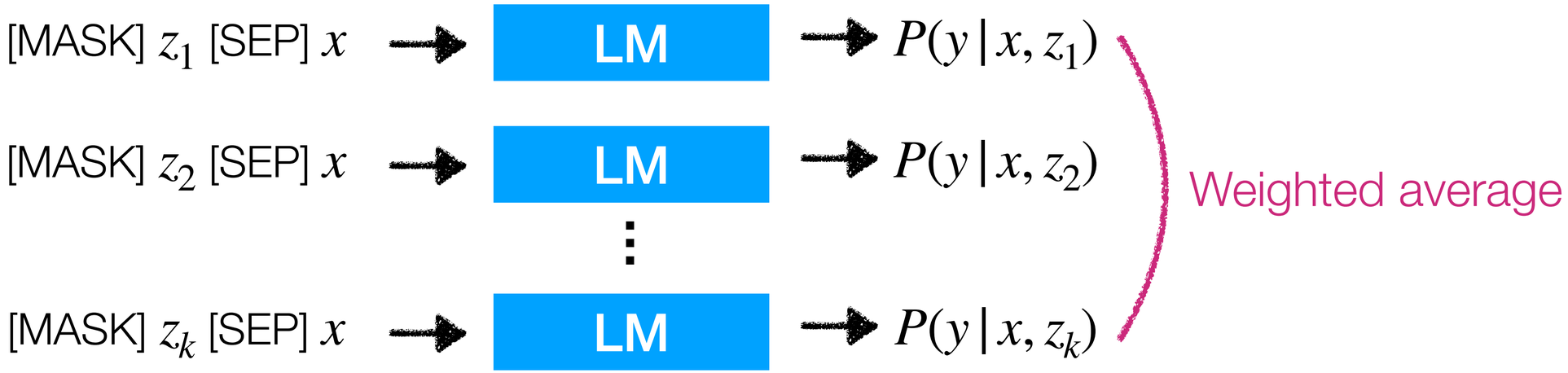

$$\mathbf{z} = \text{Encoder}(z)$$

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

$$z_1, \ldots, z_k = \text{argTopk} P(z \mid x) = \text{argTopk} \mathbf{x} \cdot \mathbf{z}$$

***k*** retrieved blocks

# REALM: (2) Read stage




0 if not one of top $k$

$$\sum_{z \in \mathcal{D}} P(z \mid x) P(y \mid x, z)$$

Need to approximate
→ Consider top $k$ chunks only

from the retrieve stage

from the read stage

# 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

# 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

# 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

# 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

# REALM and subsequent work

* REALM (Guu et al 2020): MLM followed by fine-tuning, focusing on open-domain QA
* DPR (Karpukhin et al 2020): Pipeline training instead of joint training, focusing on open-domain QA (no explicit language modeling)
* RAG (Lewis et al 2020): “Generative” instead of “masked language modeling”, focusing on open-domain QA (no explicit language modeling)
* Atlas (Izcard et al 2022): Combine RAG with retrieval-based language model pre-training based on the encoder-decoder architecture (more to come in Section 4), focusing on open-domain QA

For a while, mainly evaluated on knowledge-intensive tasks, e.g. open-domain QA

(more context in Section 5)

# REALM and subsequent work

* REALM (Guu et al 2020): MLM followed by fine-tuning, focusing on open-domain QA
* DPR (Karpukhin et al 2020): Pipeline training instead of joint training, focusing on open-domain QA (no explicit language modeling)
* RAG (Lewis et al 2020): “Generative” instead of “masked language modeling”, focusing on open-domain QA (no explicit language modeling)
* Atlas (Izcard et al 2022): Combine RAG with retrieval-based language model pre-training based on the encoder-decoder architecture (more to come in Section 4), focusing on open-domain QA
* Papers that follow this approach focusing on **LM perplexity** have come out quite recently (Shi et al. 2023, Ram et al. 2023)

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

# Retrieval-in-context in LM

***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


FIFA World Cup 2026 will expand to 48 teams.

* Can use multiple text blocks too (see the paper!)

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

# Retrieval-in-context in LM

***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


* Can use multiple text blocks too (see the paper!)

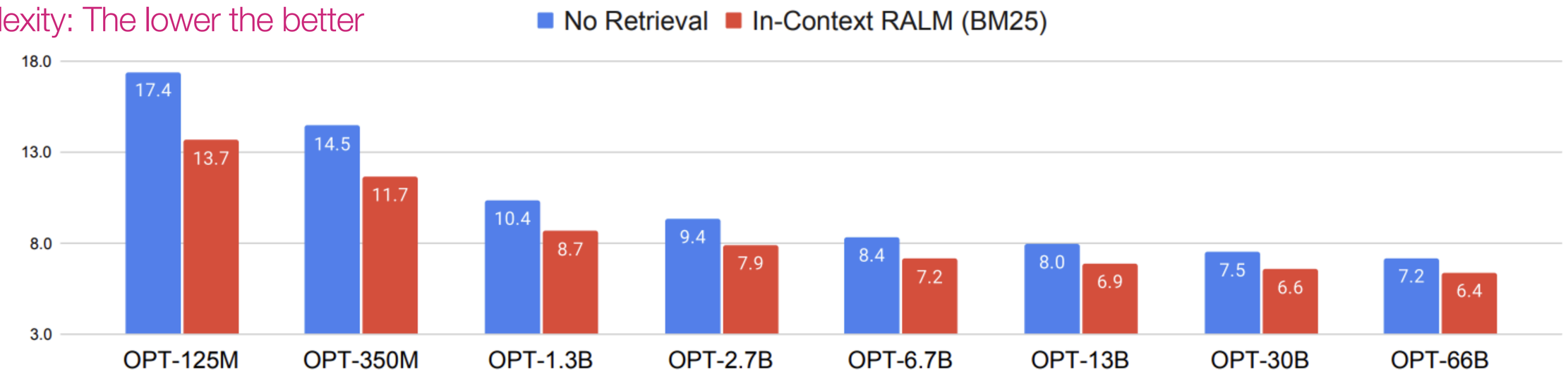
FIFA World Cup 2026 will expand to 48 teams. World Cup 2022 was the last with 32 teams, before the increase to


48 in the 2026 tournament.

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

# Retrieval-in-context in LM

Perplexity: The lower the better




Varying sizes of LMs

**Retrieval helps over all sizes of LMs**

Graphs from Ram et al. 2023

# Retrieval-in-context in LM

**Is $q=x$ necessary?**

***x*** = Team USA celebrates after winning its match against Iran at Al Thumama Stadium in Group B play of the FIFA World Cup 2022 on Nov. 29, 2022. (..) World Cup 2022 was the last with 32 teams, before the increase to

Team USA celebrates after winning its match against Iran at Al Thumama Stadium in Group B play of the FIFA World Cup 2022 on Nov. 29, 2022. (..) World Cup 2022 was the last with 32 teams, before the increase to

**Retrieval**

The U.S. national team defeated Iran 1-0.

Does not cover "tokens that will come next"

# Retrieval-in-context in LM

**Is $q=x$ necessary?**

$x$ = Team USA celebrates after winning its match against Iran at Al Thumama Stadium in Group B play of the FIFA World Cup 2022 on Nov. 29, 2022. (..) World Cup 2022 was the last with 32 teams, before the increase to

Team USA celebrates after winning its match against Iran at Al Thumama Stadium in Group B play of the FIFA World Cup 2022 on Nov. 29, 2022. (..) World Cup 2022 was the last with 32 teams, before the increase to

**Retrieval**

The U.S. national team defeated Iran 1-0.

Does not cover "tokens that will come next"

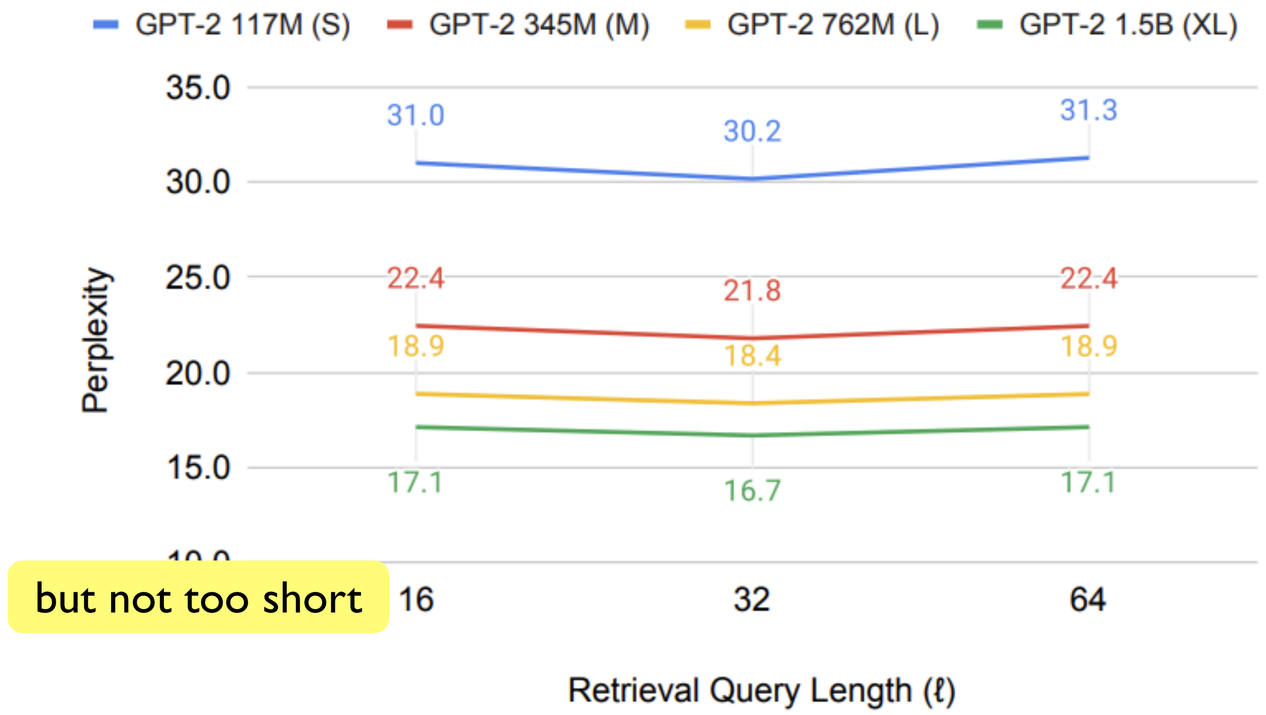
World Cup 2022 was the last with 32 teams, before the increase to

**Retrieval**

FIFA World Cup 2026 will expand to 48 teams.

more relevant to what will come next

# Retrieval-in-context in LM




but not too short

Shorter prefix (more recent tokens) as a query helps

Graphs from Ram et al. 2023

# Retrieval-in-context in LM

How frequent should retrieval be?

World Cup 2022 was the last with


The 2022 FIFA World Cup (...) The 32 national teams involved in the tournament.

# Retrieval-in-context in LM

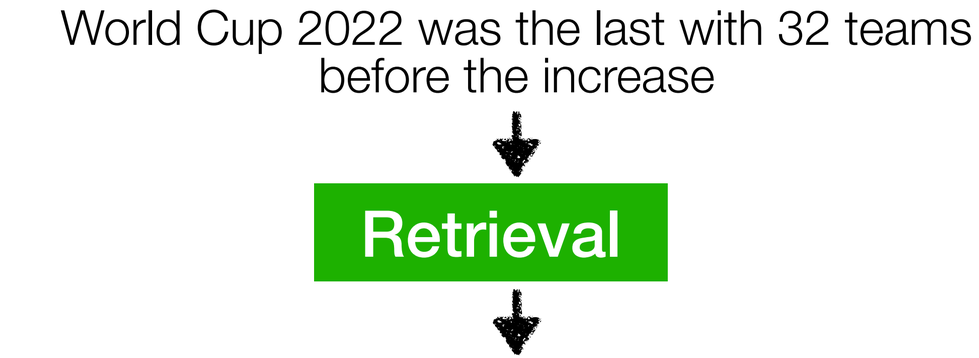
How frequent should retrieval be?

# Retrieval-in-context in LM

How frequent should retrieval be?

# Retrieval-in-context in LM

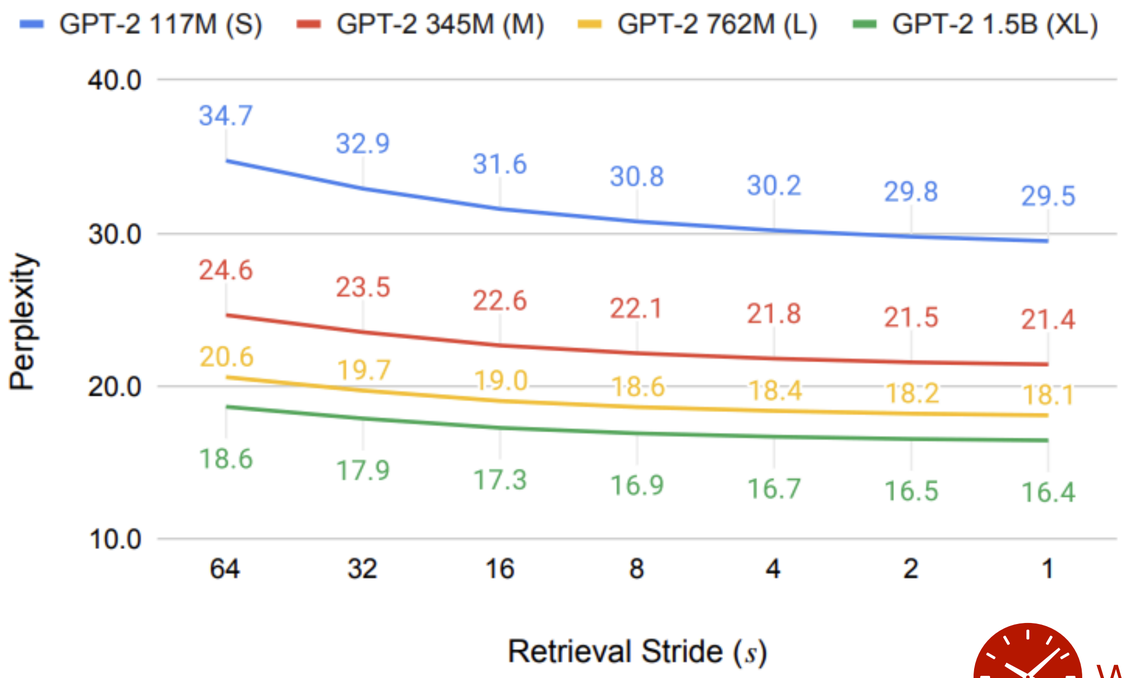
How frequent should retrieval be?

# Retrieval-in-context in LM

**How frequent should retrieval be?**


# Retrieval-in-context in LM




with cost in inference time

Retrieving more frequently helps

Graphs from Ram et al. 2023

# Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)

**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

# Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)

**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

# Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)

**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

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |

*can be very inefficient to retrieve many text blocks, frequently*

# RETRO (Borgeaud et al. 2021)

✓ Incorporation in the “intermediate layer” instead of the “input” layer
→ designed for *many* blocks, *frequently*, more *efficiently*

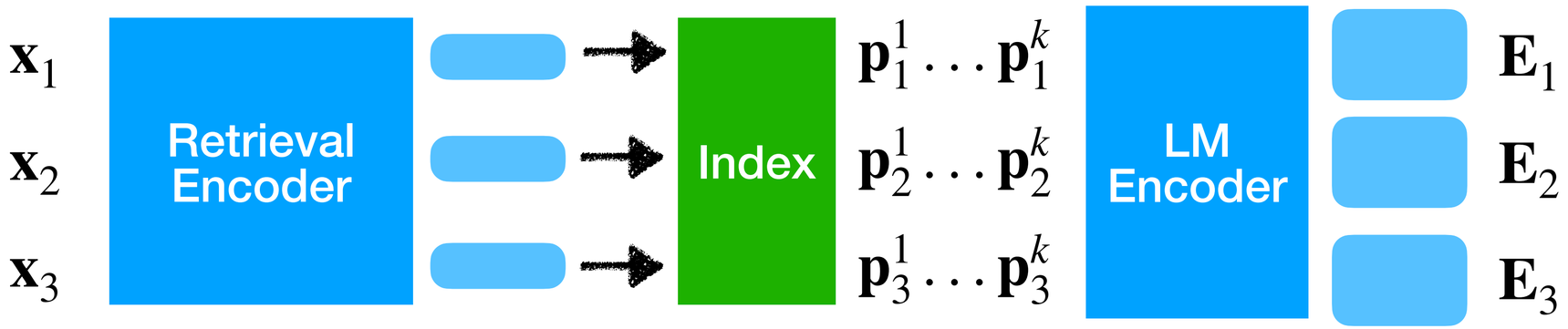
✓ Scale the datastore to retrieve from

Borgeaud et al. 2021. “Improving language models by retrieving from trillions of tokens”

# RETRO (Borgeaud et al. 2021)

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

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



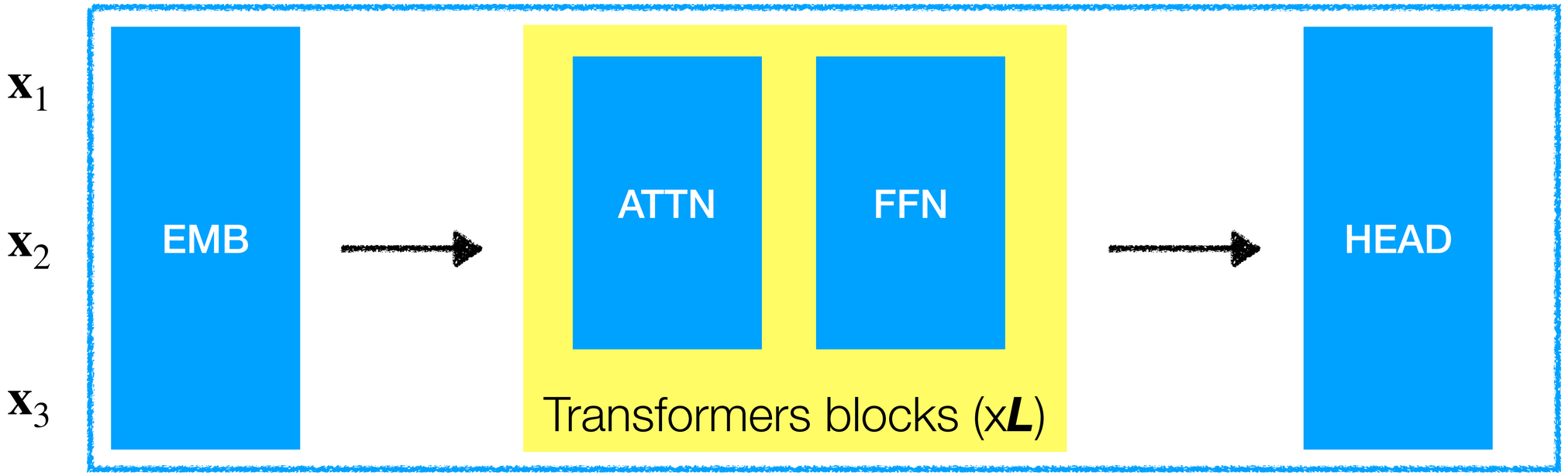

(A $r \times k \times d$ matrix)

($r$ = # tokens per text block)

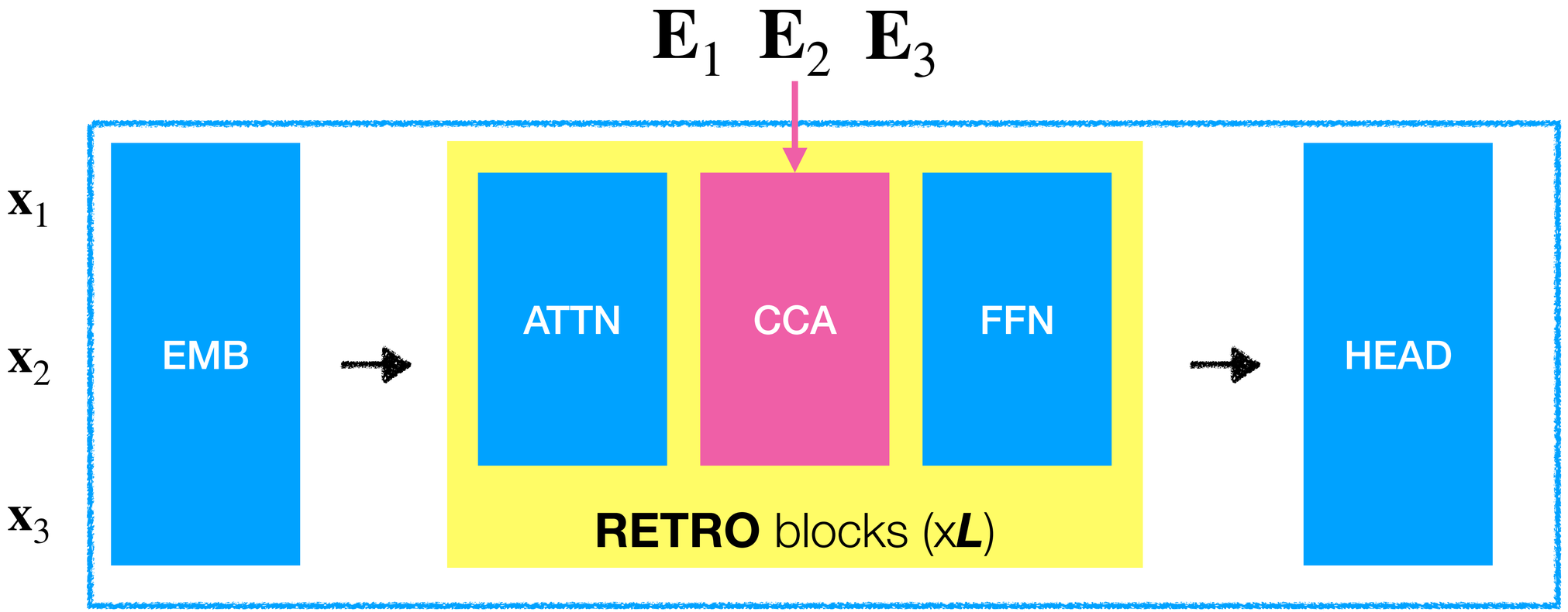
($d$ = hidden dimension)

($k$ = # retrieved blocks per split)

# Regular decoder

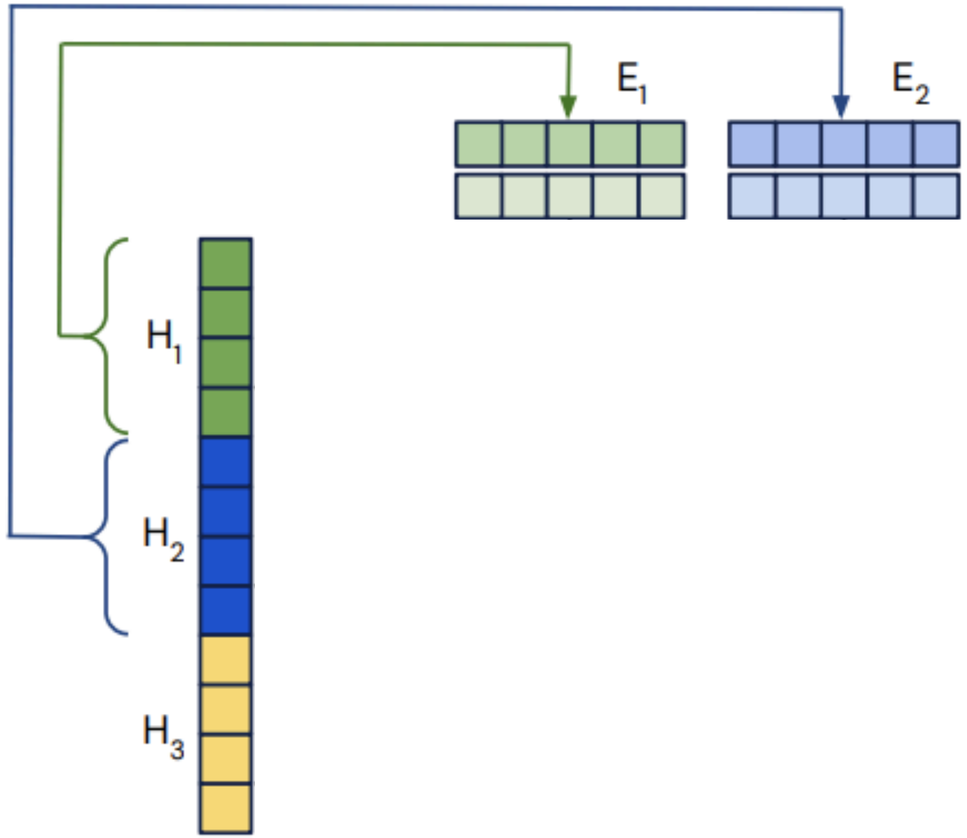


# Decoder in RETRO



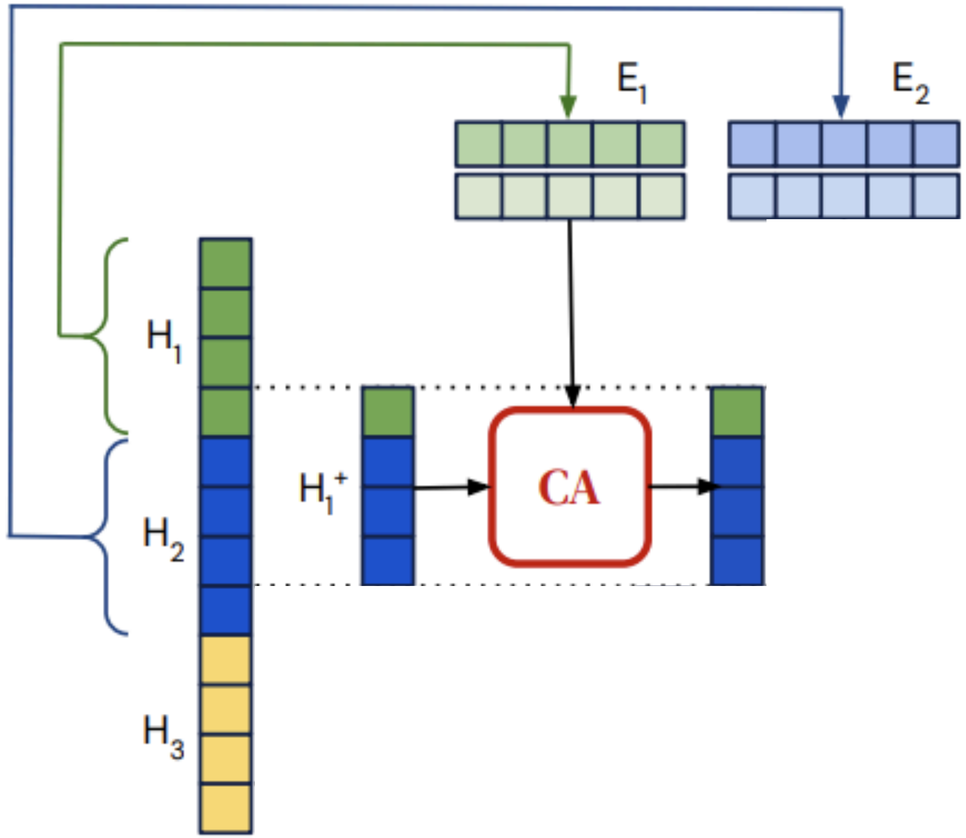

Chunked Cross Attention (CCA)

# Chunked Cross Attention



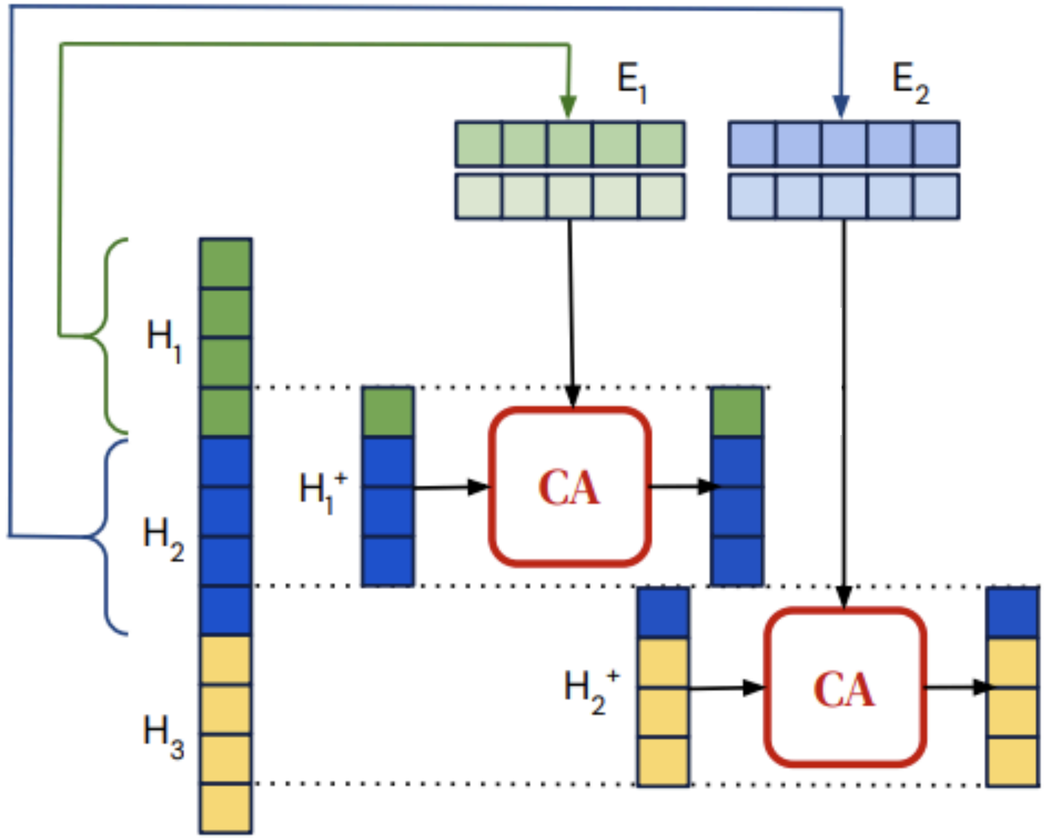

Outputs from the previous layer H

# Chunked Cross Attention



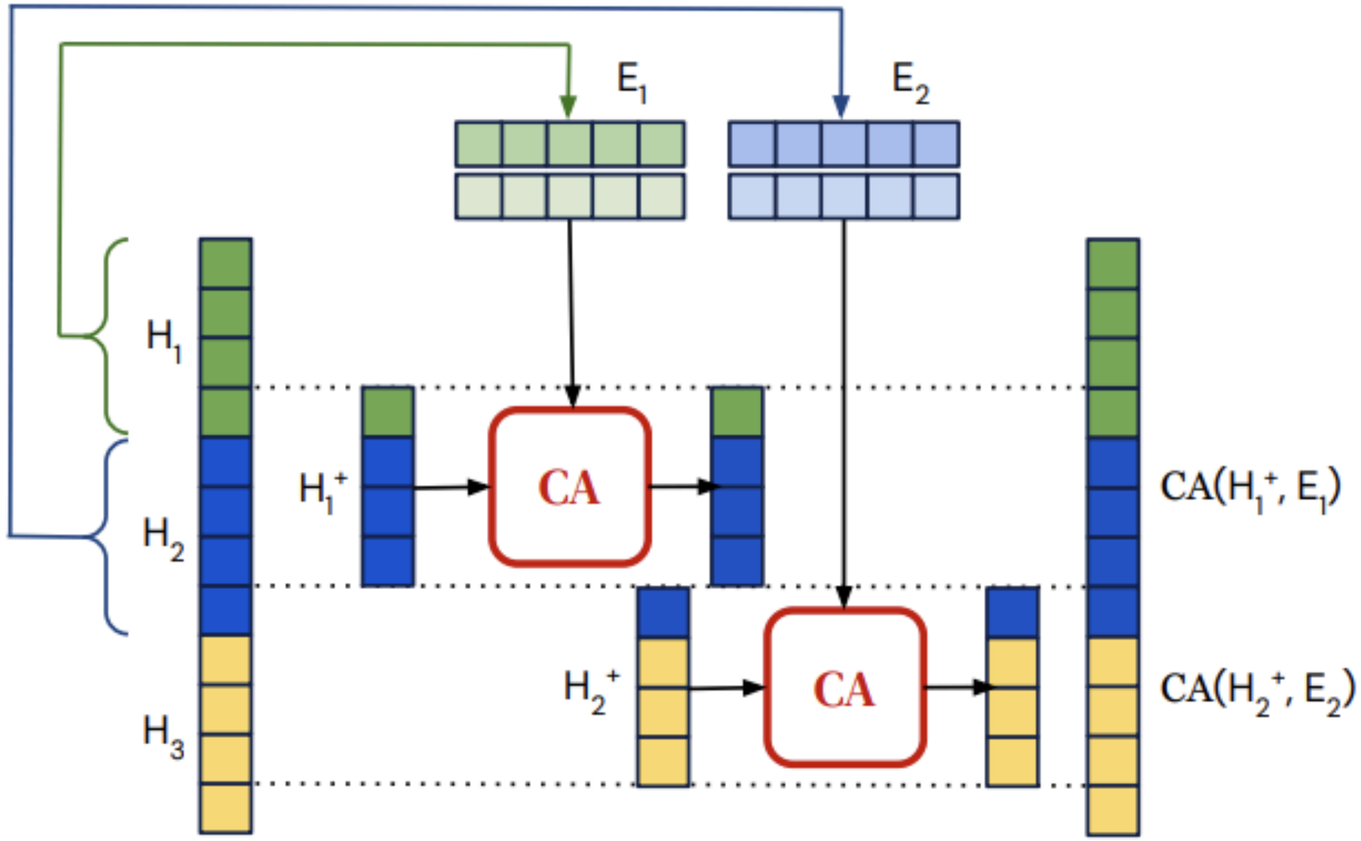

Outputs from the previous layer H

# Chunked Cross Attention




Outputs from the previous layer H

# Chunked Cross Attention




Outputs from the previous layer H

Inputs to the next layer

✓ Cross-attention can be computed in parallel

✓ Depend upon the set of retrieved blocks of all previous tokens

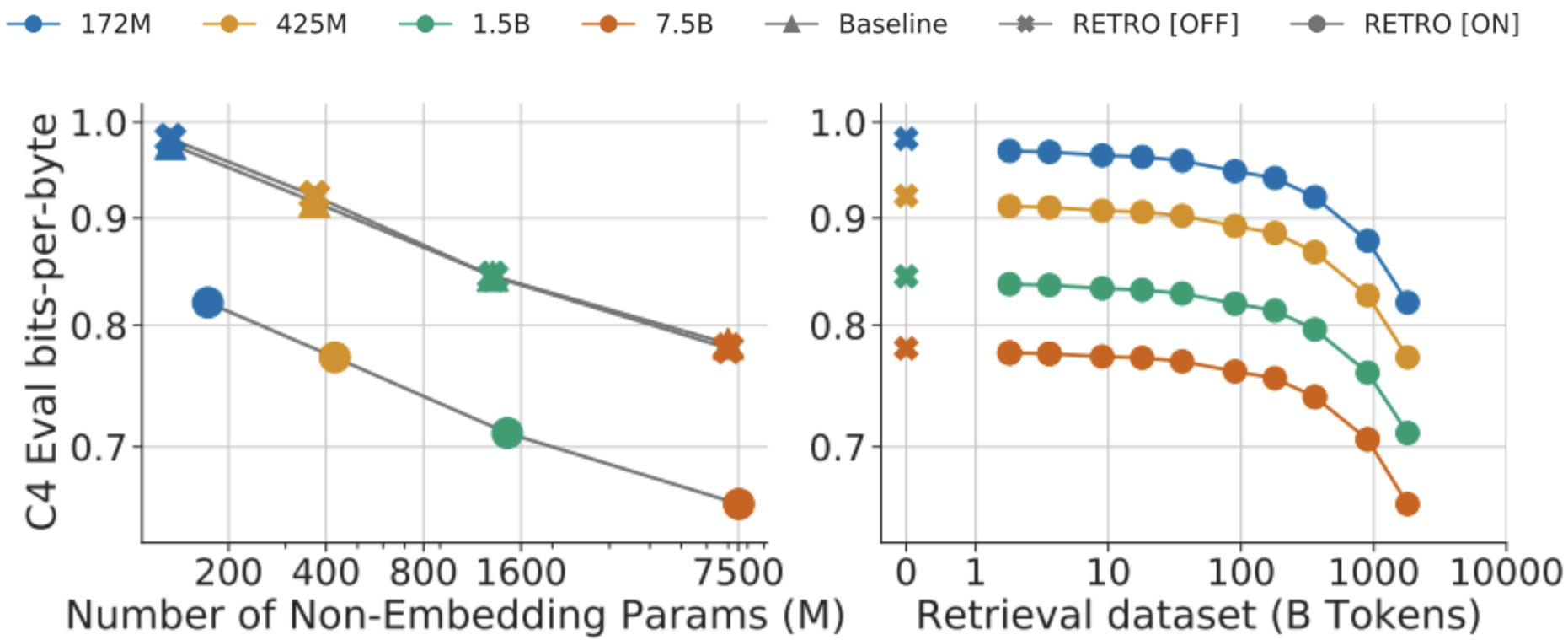
# Results

Perplexity: The lower the better

| Model | Retrieval Set | #Database tokens | #Database keys | Valid | Test |
|---|---|---|---|---|---|
| Adaptive Inputs (Baevski and Auli, 2019) | - | - | - | 17.96 | 18.65 |
| SPALM (Yogatama et al., 2021) | Wikipedia | 3B | 3B | 17.20 | 17.60 |
| *k*NN-LM (Khandelwal et al., 2020) | Wikipedia | 3B | 3B | 16.06 | 16.12 |
| Megatron (Shoeybi et al., 2019) | - | - | - | - | 10.81 |
| Baseline transformer (ours) | - | - | - | 21.53 | 22.96 |
| *k*NN-LM (ours) | Wikipedia | 4B | 4B | 18.52 | 19.54 |
| RETRO | Wikipedia | 4B | 0.06B | 18.46 | 18.97 |
| RETRO | C4 | 174B | 2.9B | 12.87 | 10.23 |
| RETRO | MassiveText (1%) | 18B | 0.8B | 18.92 | 20.33 |
| RETRO | MassiveText (10%) | 179B | 4B | 13.54 | 14.95 |
| RETRO | MassiveText (100%) | 1792B | 28B | **3.21** | **3.92** |

Significant improvements by retrieving from 1.8 trillion tokens

# Results




Gains are constant with model scale

The larger datastore is, the better

# RETRO (Borgeaud et al. 2021)

**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

# RETRO (Borgeaud et al. 2021)

**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

# RETRO (Borgeaud et al. 2021)

**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

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |

Can use many blocks, more frequently, more efficiently

Additional complexity; Can't be used without training (more in section 4)

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |

*What else?*

# kNN-LM (Khandelwal et al. 2020)

✓ A different way of using retrieval, where the LM outputs a nonparametric distribution over every distinct token in the data.

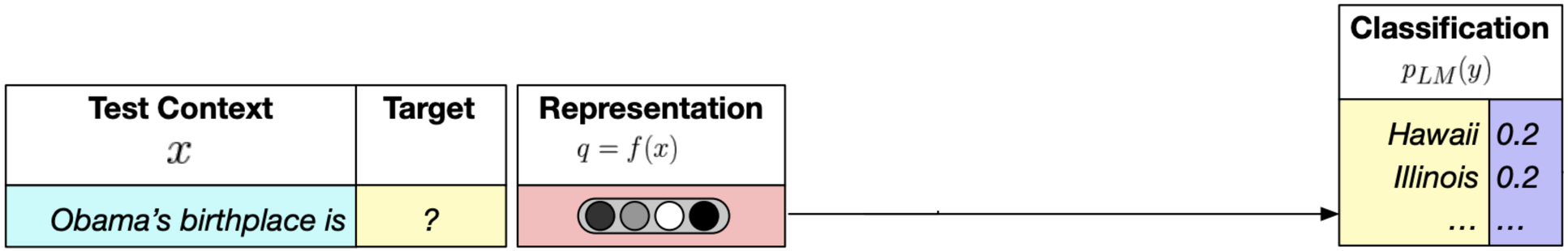
✓ Can be seen as an incorporation in the “output” layer

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

# kNN-LM (Khandelwal et al. 2020)

| Test Context $x$ | Target |
|---|---|
| *Obama's birthplace is* | *?* |

# kNN-LM (Khandelwal et al. 2020)



# kNN-LM (Khandelwal et al. 2020)

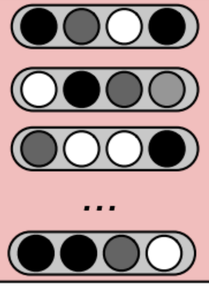
*The size of the datastore = # of distinct tokens in the corpus*

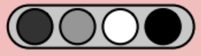
| Training Contexts $c_i$ | Targets $v_i$ |
|---|---|
| *Obama was senator for* | *Illinois* |
| *Barack is married to* | *Michelle* |
| *Obama was born in* | *Hawaii* |
| ... | ... |
| *Obama is a native of* | *Hawaii* |

... Obama was senator for Illinois from 1997 to 2005, .... Barack is Married to Michelle and their first daughter, ... Obama was born in Hawaii, and graduated from Columbia University. ... Obama is a native of Hawaii, ....

| Test Context $x$ | Target | Representation $q = f(x)$ |
|---|---|---|
| *Obama's birthplace is* | *?* |  |

# kNN-LM (Khandelwal et al. 2020)

| Training Contexts $c_i$ | Targets $v_i$ | Representations $k_i = f(c_i)$ |
|---|---|---|
| *Obama was senator for* | *Illinois* |  |
| *Barack is married to* | *Michelle* | |
| *Obama was born in* | *Hawaii* | |
| *...* | *...* | |
| *Obama is a native of* | *Hawaii* | |

| Test Context $x$ | Target | Representation $q = f(x)$ |
|---|---|---|
| *Obama's birthplace is* | *?* |  |

*Which "distinct" tokens <u>in a datastore</u> are close to the next token?*

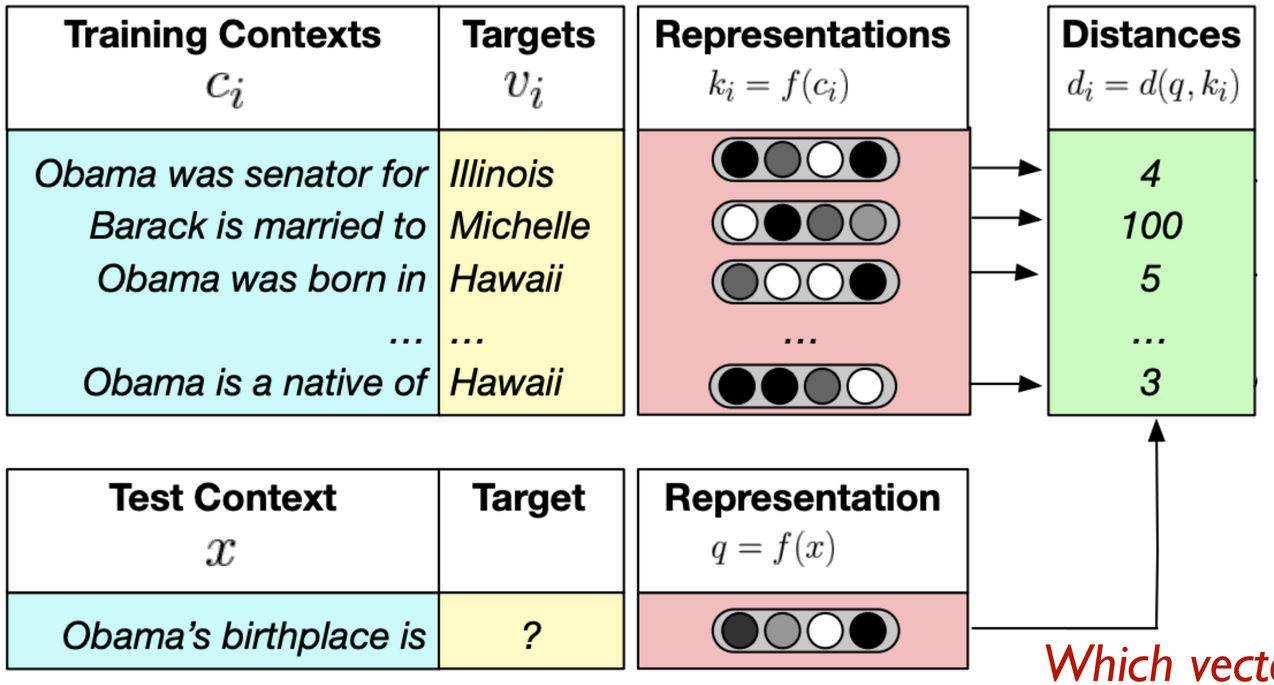
**=**

*Which prefixes in a datastore are close to the prefix we have?*

**=**

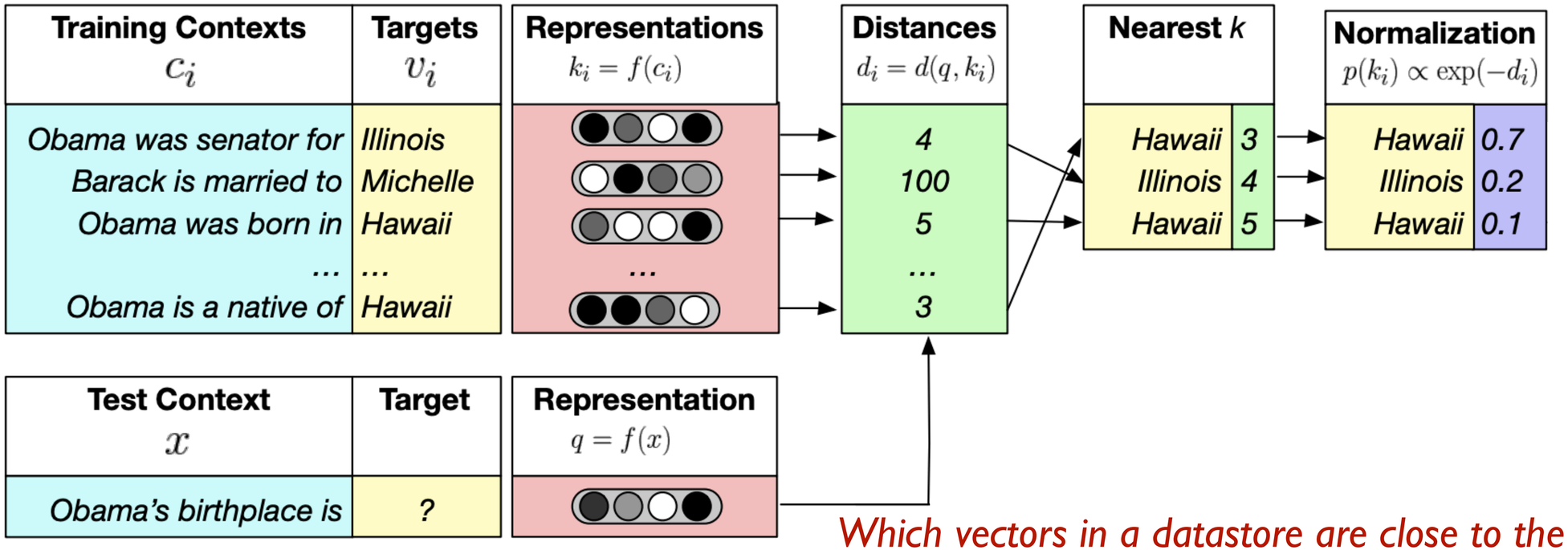
*Which vectors in a datastore are close to the vector we have?*

# kNN-LM (Khandelwal et al. 2020)



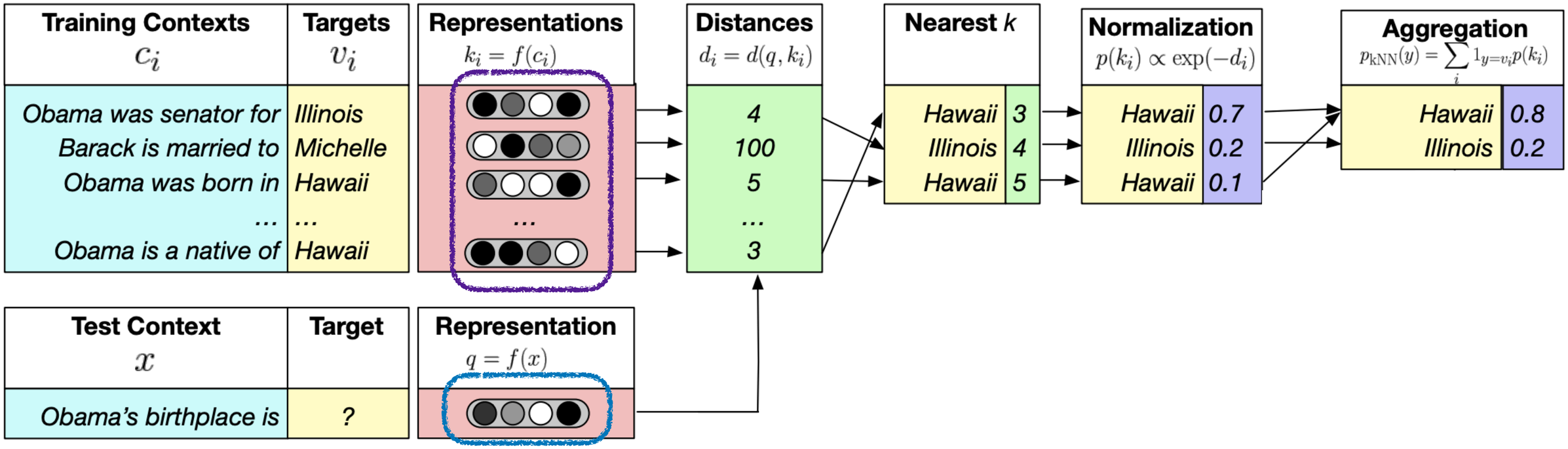

*Which vectors in a datastore are close to the vector we have?*

# kNN-LM (Khandelwal et al. 2020)



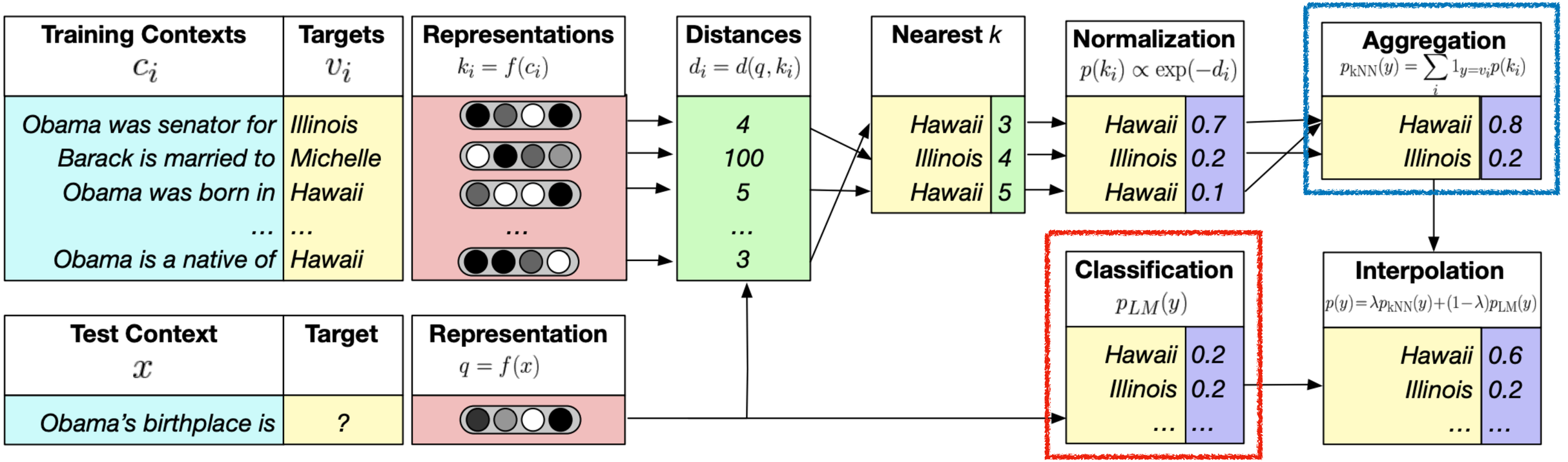

*Which vectors in a datastore are close to the vector we have?*

# kNN-LM (Khandelwal et al. 2020)




$$P_{k\text{NN}}(y \mid x) \propto \sum_{(k,v)\in\mathcal{D}} \mathbb{I}[v = y]\text{sim}(k, x) \qquad \text{sim}(k, x) = \exp\left(-d(\text{Enc}(k), \text{Enc}(x))\right)$$

# kNN-LM (Khandelwal et al. 2020)




$\lambda$: hyperparameter

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

Later work, e.g., NPM (Min et al. 2023) removed interpolation (more in Section 4)

# kNN-LM - why?

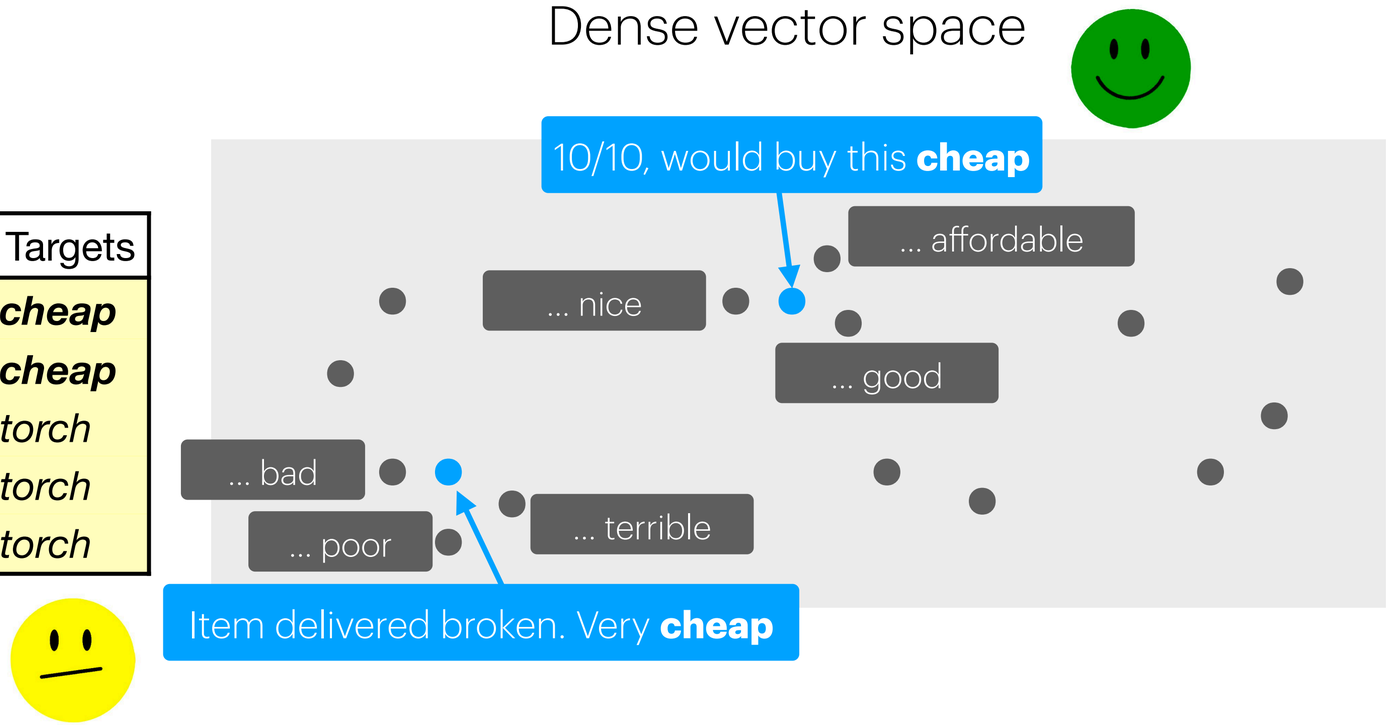
| Training contexts | Targets |
|---|---|
| *10/10, would buy this* | *cheap* |
| *Item delivered broken. Very* | *cheap* |
| *To check the version of PyTorch, you can use* | *torch* |
| *You are permitted to bring a* | *torch* |
| *A group of infections … one of the* | *torch* |

# kNN-LM - why?

| Training contexts | Targets |
|---|---|
| ***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* |

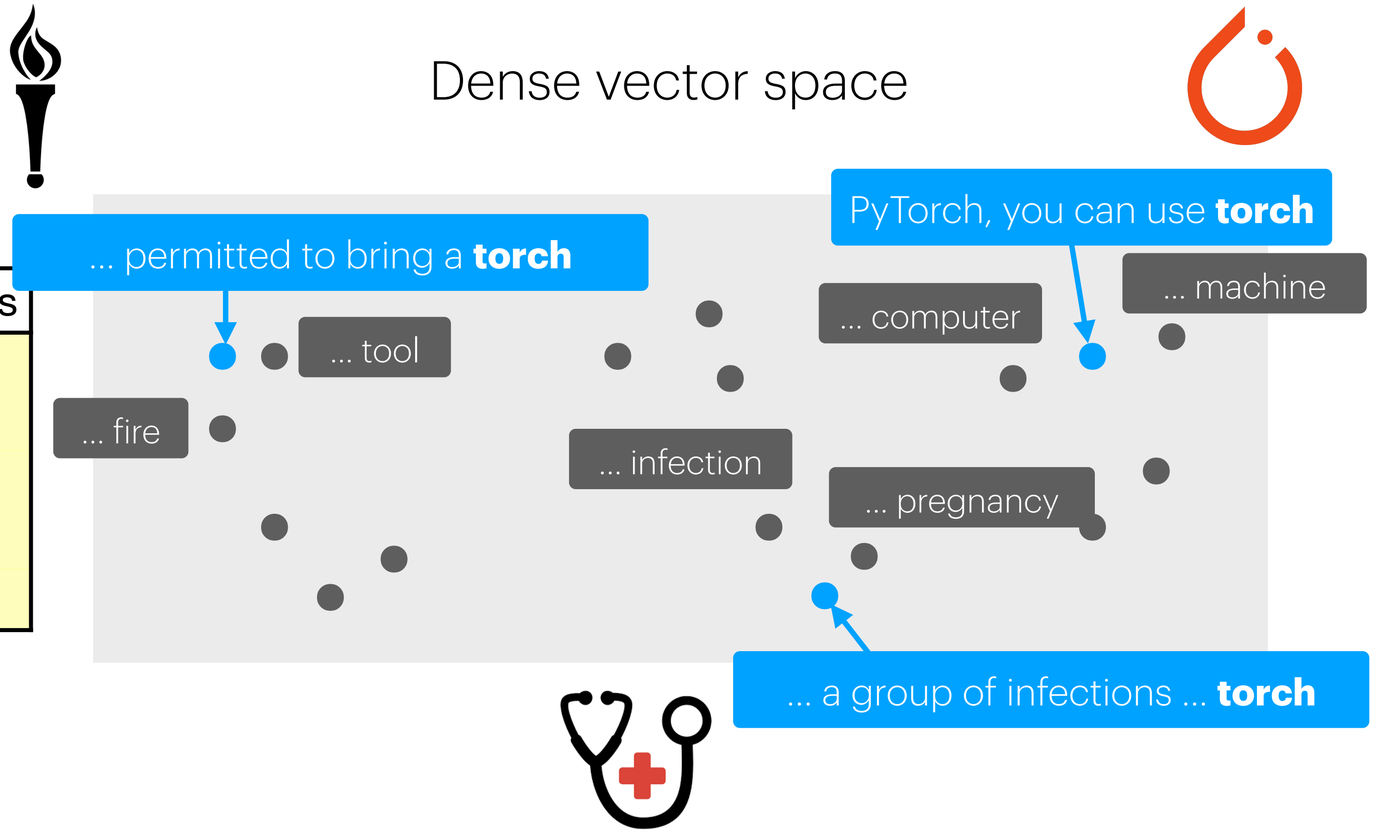
# kNN-LM - why?

| Training contexts | Targets |
|---|---|
| ***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* |

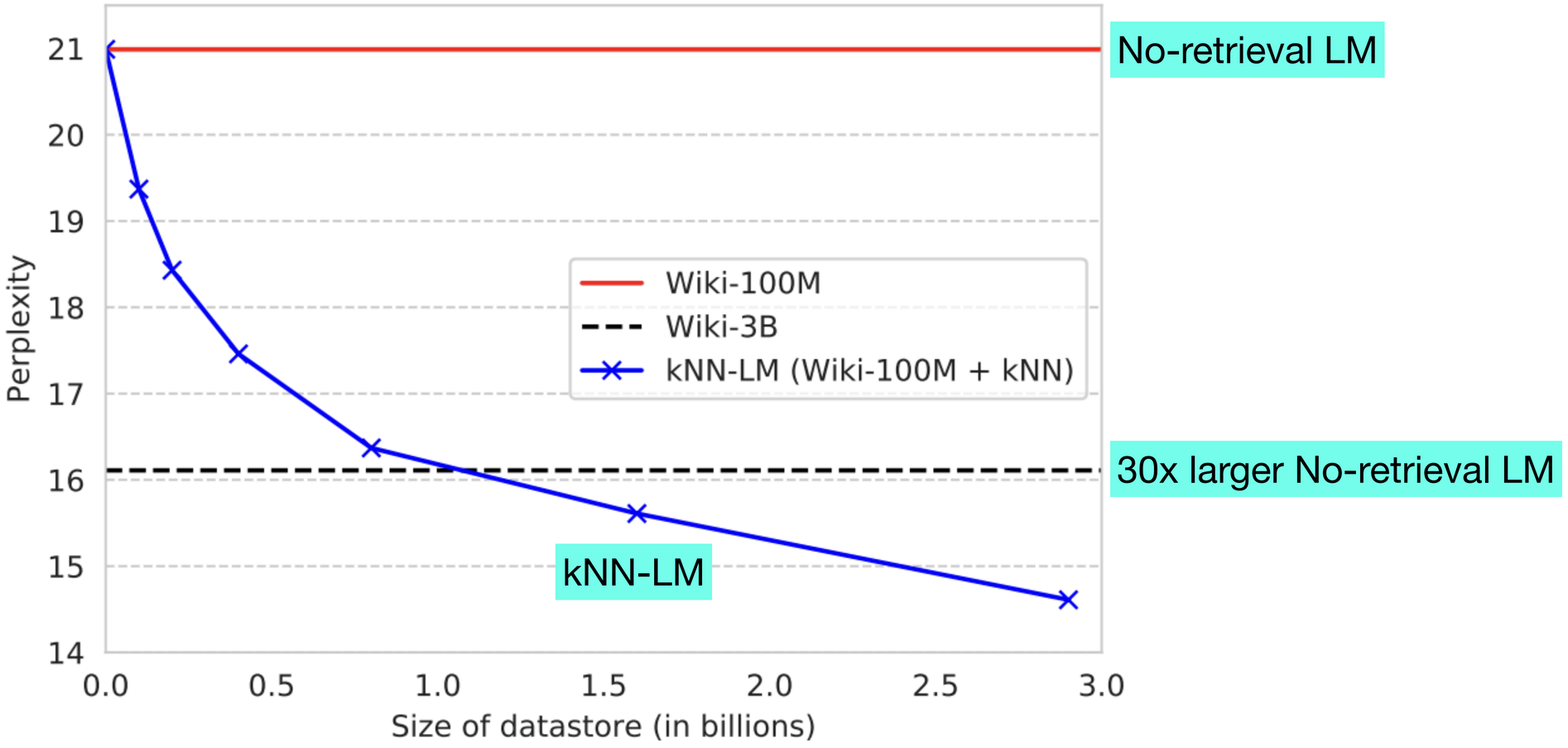


# kNN-LM - why?

| Training contexts | Targets |
|---|---|
| *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* |



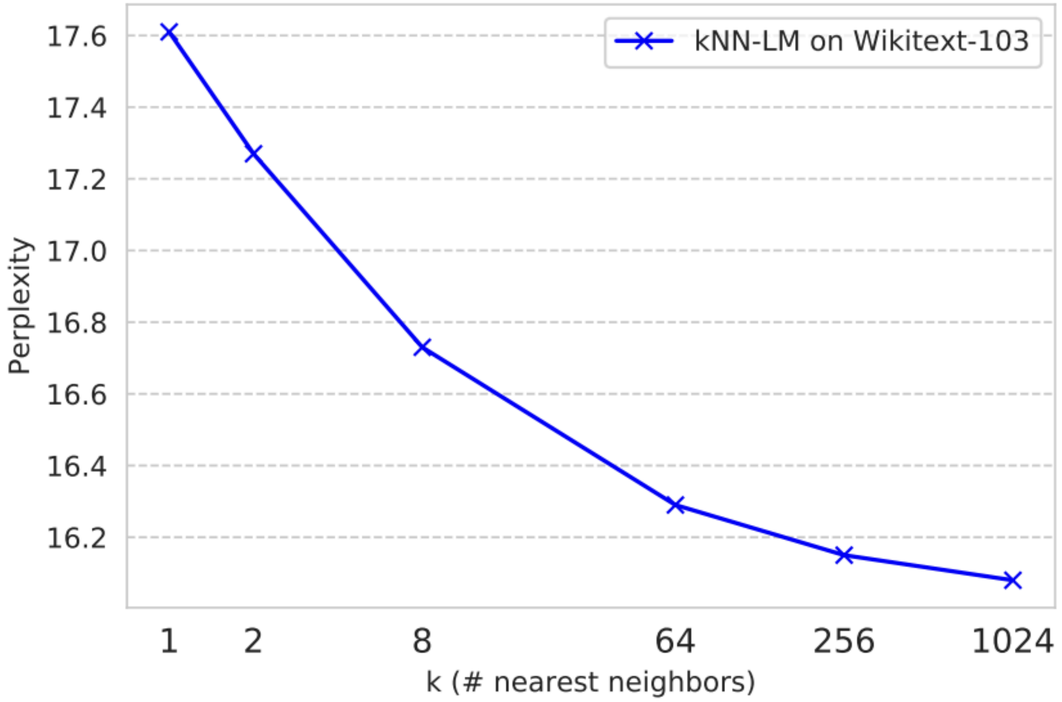
# kNN-LM - results



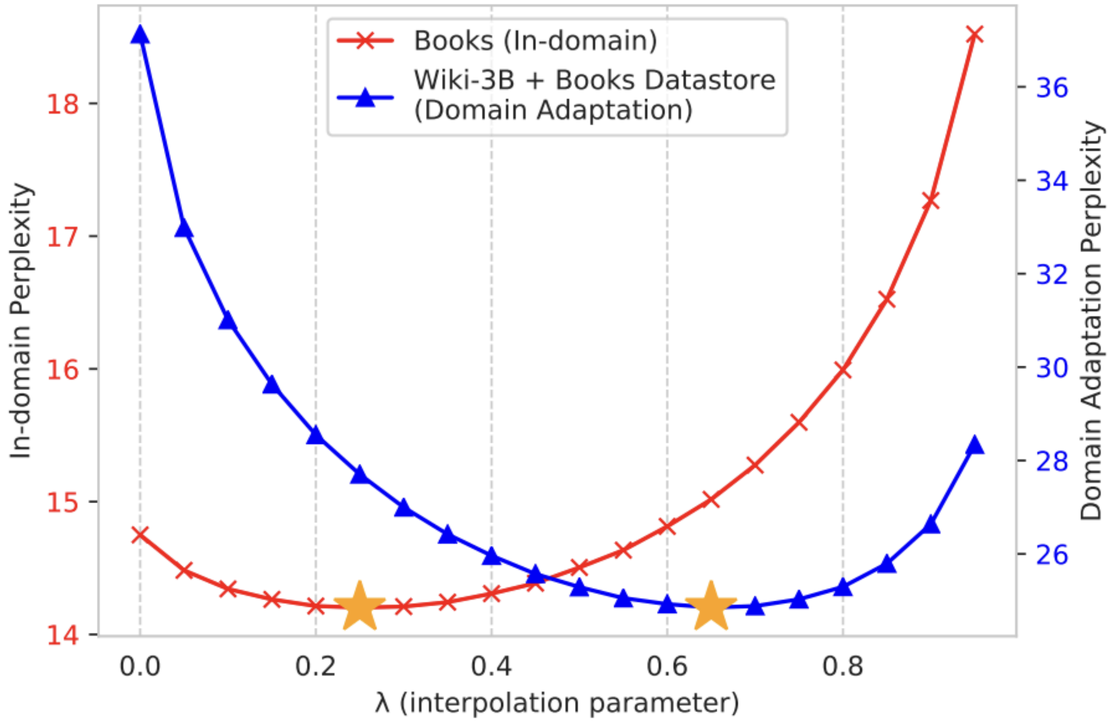

Outperforms no-retrieval LM

Better with bigger datastore

# kNN-LM - results




Better with bigger ***k***




Helps more out-of-domain

# kNN-LM (Khandelwal 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

# kNN-LM (Khandelwal 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

# kNN-LM (Khandelwal 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

# kNN-LM (Khandelwal 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** ✓

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |

More fine-grained; Can be better

Can be very efficient (as long as

**Wikipedia: 13 millions vs. 4 billions**

Datastore is expensive in space: given the same data, # text blocks vs. # tokens

No cross attention between input and retrieval results

# Extensions

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |

*It's fixed! Can we do adaptively?*

# Adaptive retrieval for efficiency

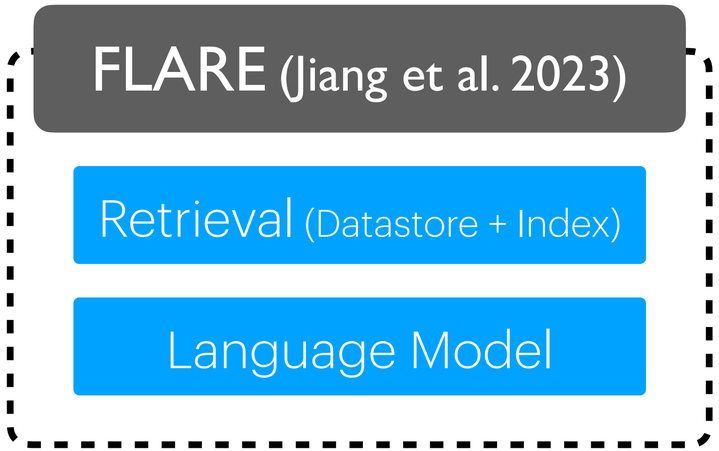
Adaptive retrieval of text blocks
(following retrieve-in-context)

Adaptive retrieval of tokens
(following kNN-LM)

# Adaptive retrieval of *blocks*

*- Judge necessity*

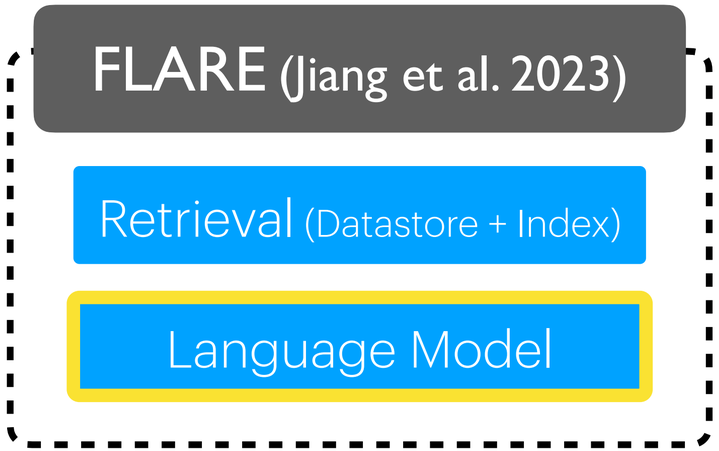
**Input:** Generate a summary about Joe Biden.




Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

*- Judge necessity*



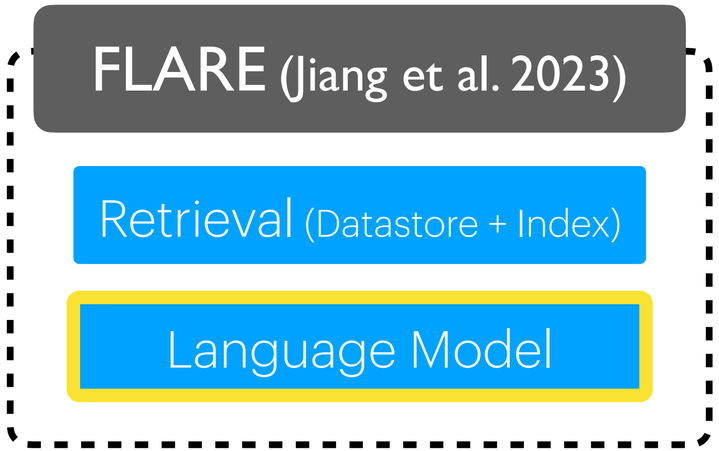

**Input:** Generate a summary about Joe Biden.

Joe Biden (born November 20, 1942) is the 46th president of the United States.

Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

*- Judge necessity*



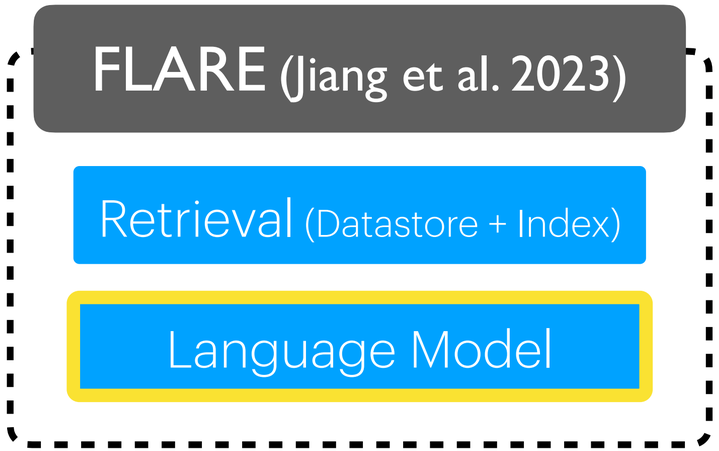

**Input:** Generate a summary about Joe Biden.

Joe Biden (born November 20, 1942) is the 46th president of the United States.

Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

*- Judge necessity*



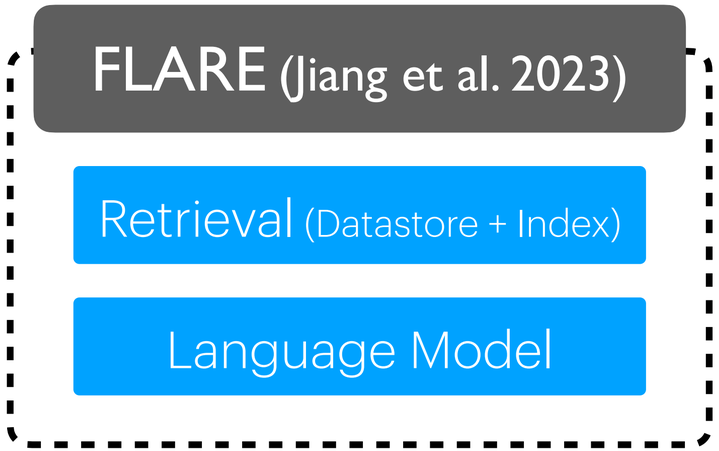

**Input:** Generate a summary about Joe Biden.

Joe Biden (born November 20, 1942) is the 46th president of the United States. Joe Biden attended **the University of Pennsylvania**, where he earned **a law degree**.

Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

*- Judge necessity*



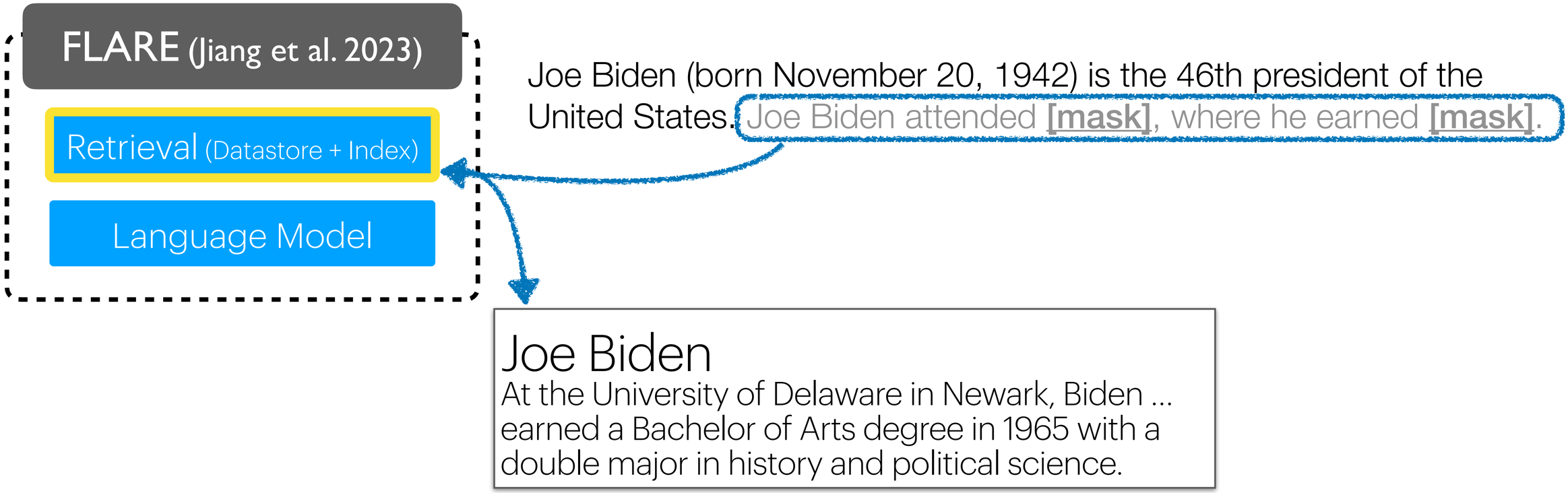

**Input:** Generate a summary about Joe Biden.

Joe Biden (born November 20, 1942) is the 46th president of the United States. Joe Biden attended **[mask]**, where he earned **[mask]**.

Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

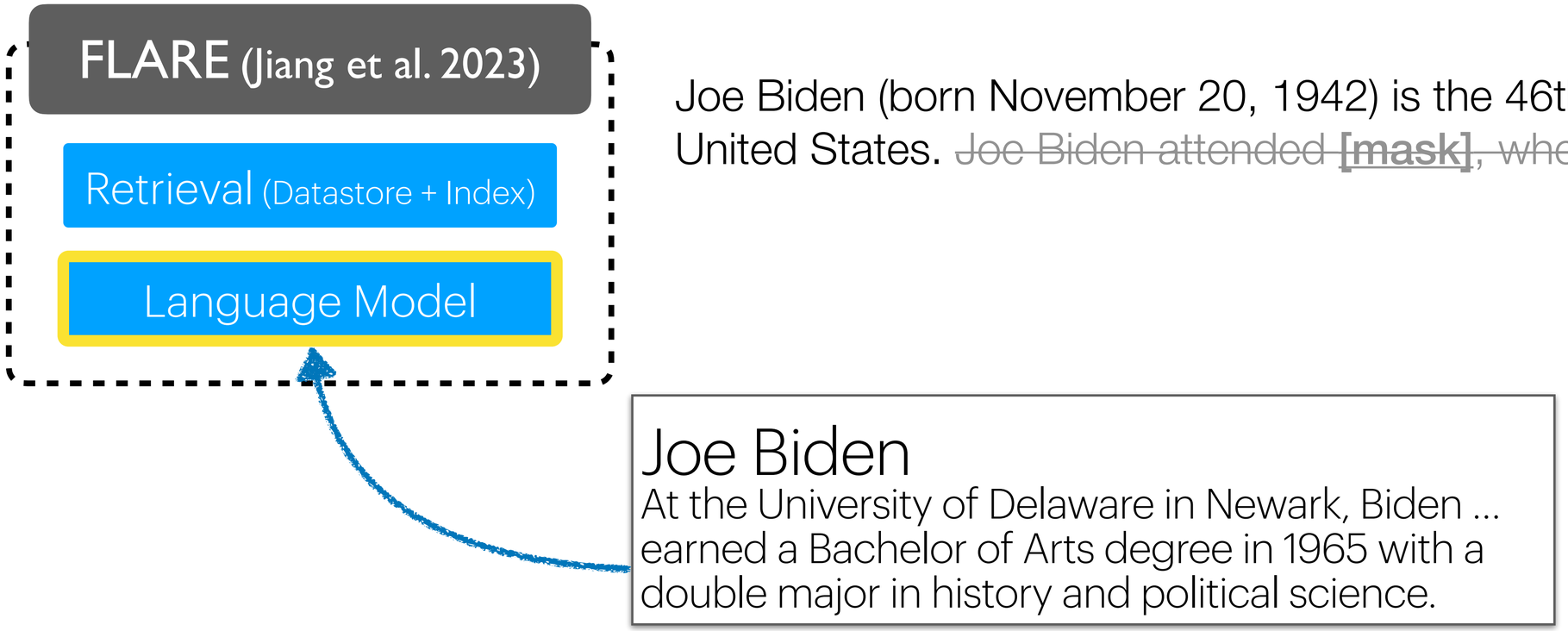
*- Judge necessity*




Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

*- Judge necessity*



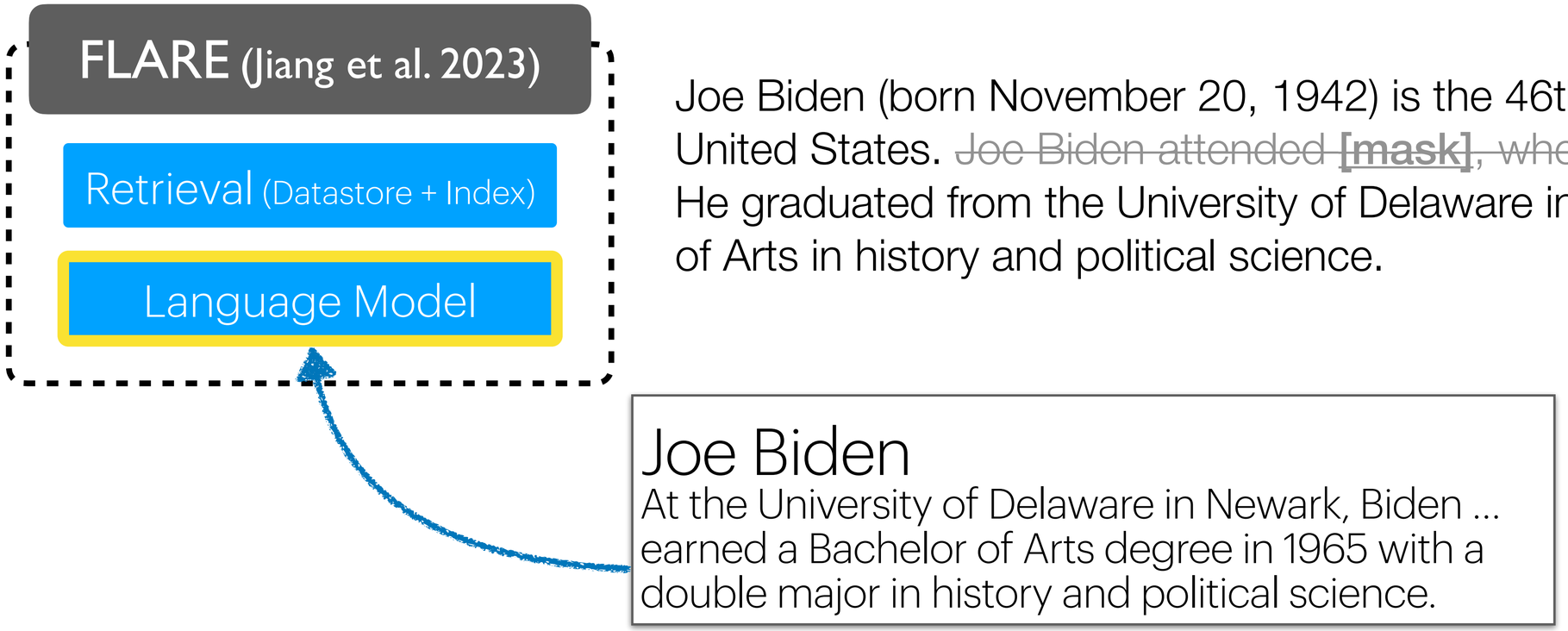

**Input:** Generate a summary about Joe Biden.

Joe Biden (born November 20, 1942) is the 46th president of the United States. ~~Joe Biden attended **[mask]**, where he earned **[mask].**~~

Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *blocks*

*- Judge necessity*



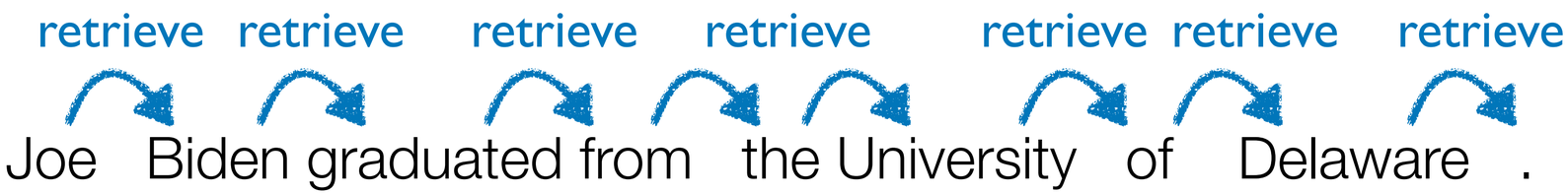

**Input:** Generate a summary about Joe Biden.

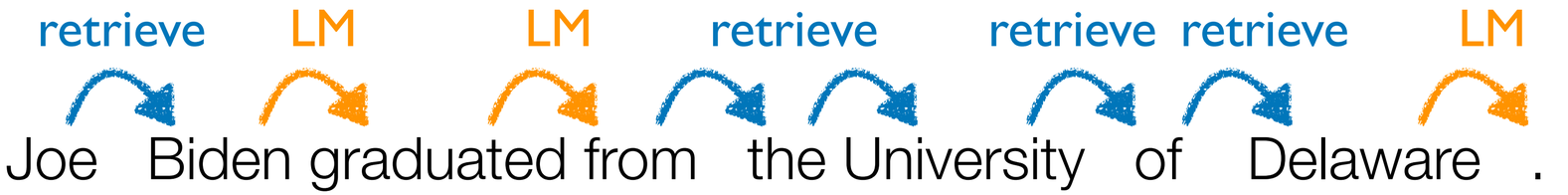
Joe Biden (born November 20, 1942) is the 46th president of the United States. ~~Joe Biden attended **[mask]**, where he earned **[mask]**.~~ He graduated from the University of Delaware in 1965 with a Bachelor of Arts in history and political science.

Jiang et al. “Active Retrieval Augmented Generation”

# Adaptive retrieval of *tokens*

*- Judge necessity*




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

*A function of **x** and retrieved contexts*

$$\rightarrow \ \lambda = 0 \quad \textit{if} \ \ \lambda < \gamma$$

He et al. 2021. “Efficient Nearest Neighbor Language Models”

Drozdov et al. 2022. “You can't pick your neighbors, or can you? When and how to rely on retrieval in the kNN-LM”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

Joe Biden graduated from

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

retrieve

Joe Biden graduated from the

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

retrieve retrieve

Joe Biden graduated from the University

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |


Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

retrieve retrieve retrieve retrieve

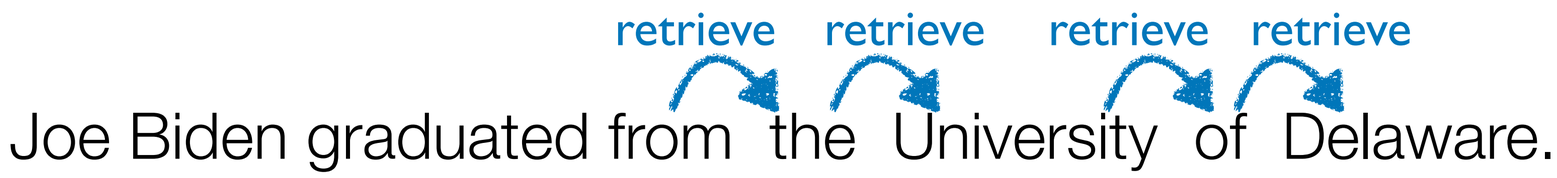
Joe Biden graduated from the University of Delaware.

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |




Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

retrieve

Joe Biden graduated from the

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

pointer

retrieve pointer

Joe Biden graduated from the University

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

pointer

retrieve pointer pointer

Joe Biden graduated from the University of

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Adaptive retrieval of *tokens*

*- Use local info*

| Training contexts | Targets |
|---|---|
| *At* | *the* |
| *At the* | *University* |
| *At the Universty* | *of* |
| *At the University of* | *Delaware* |
| *At the University of Delaware* | *in* |
| *At the University of Delaware in* | *Newark* |

pointer

retrieve pointer pointer pointer

Joe Biden graduated from the University of Delaware.

Retrieve once, and save other searches!

Alon et al. 2022. “Neuro-Symbolic Language Modeling with Automaton-augmented Retrieval”

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |
| FLARE (Jiang et al. 2023) | Text blocks | Input layer | Every n tokens ***(adaptive)*** |
| Adaptive kNN-LM (He et al 2021, Drozdov et al. 2022, Alon et al 2022, etc) | Tokens | Output layer | Every n tokens ***(adaptive)*** |

 More efficient

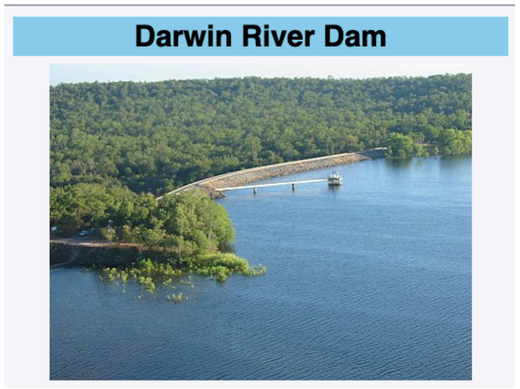
 Additional complexity

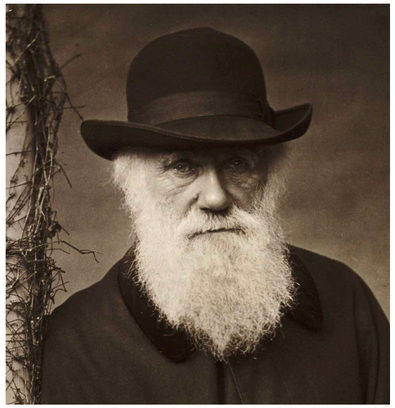
# Summary

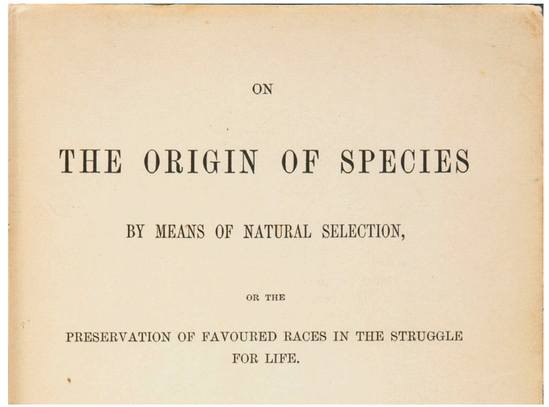
| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |
| FLARE (Jiang et al. 2023) | Text blocks | Input layer | Every n tokens *(adaptive)* |
| Adaptive kNN-LM (He et al 2021, Drozdov et al. 2022, Alon et al 2022, etc) | Tokens | Output layer | Every n tokens *(adaptive)* |

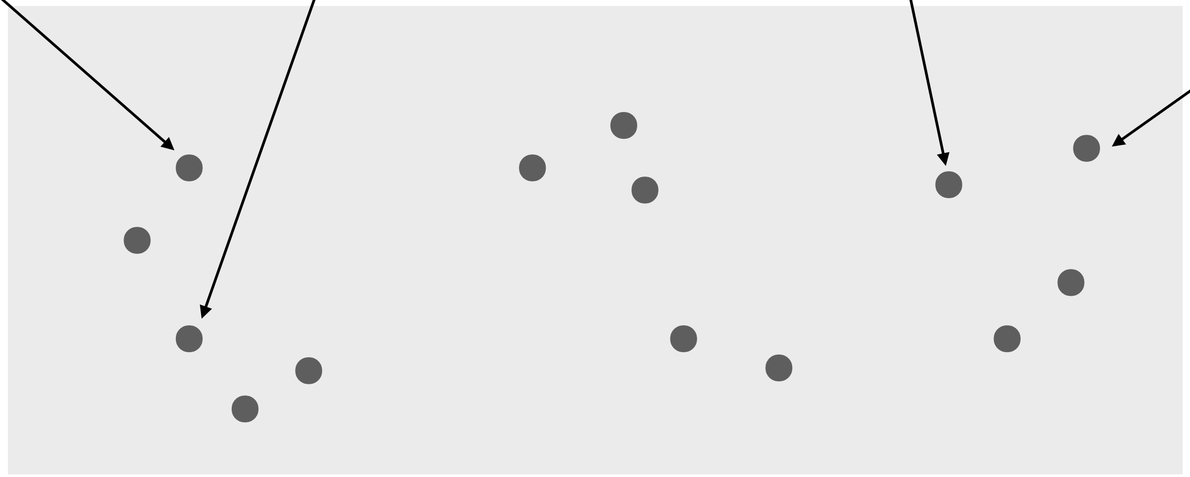
*What else beyond text blocks and tokens?*

# Entities as Experts (Fevry et al. 2020)

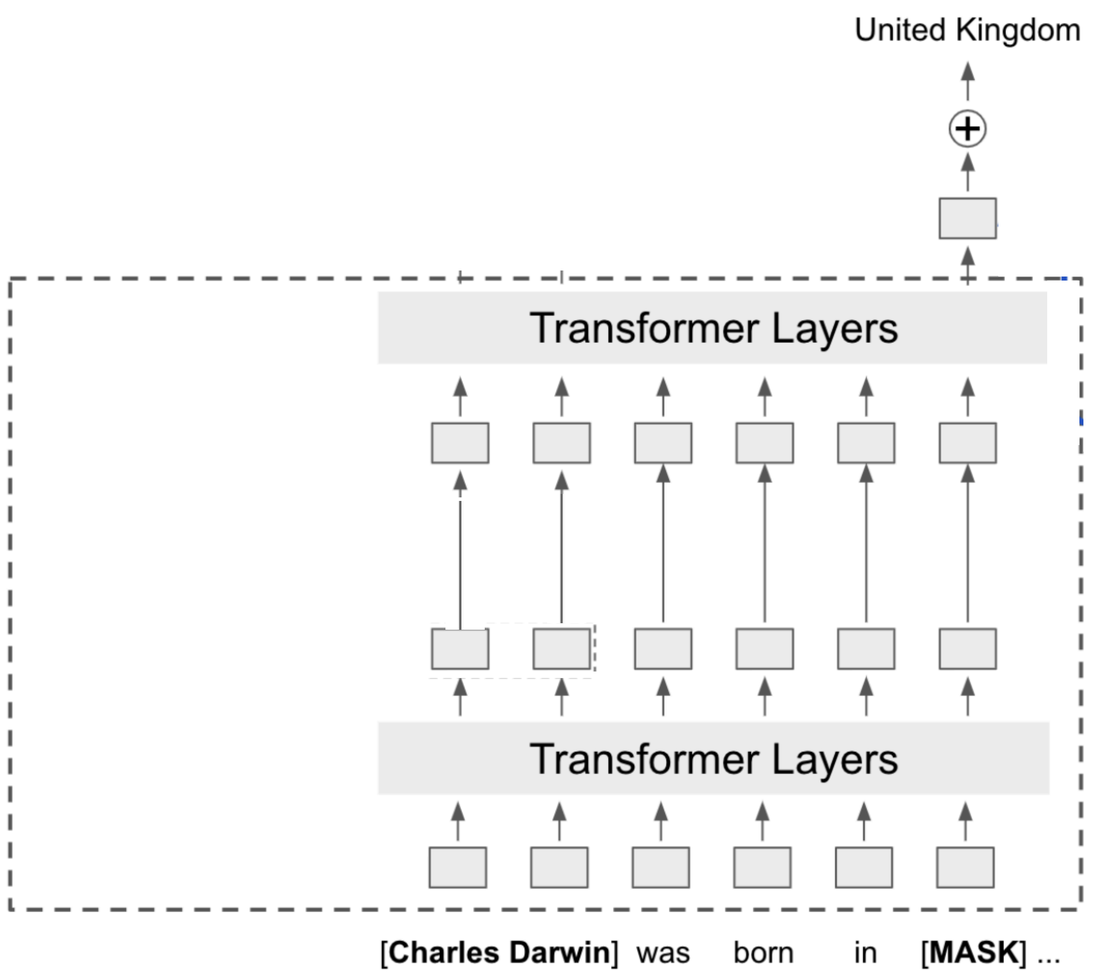




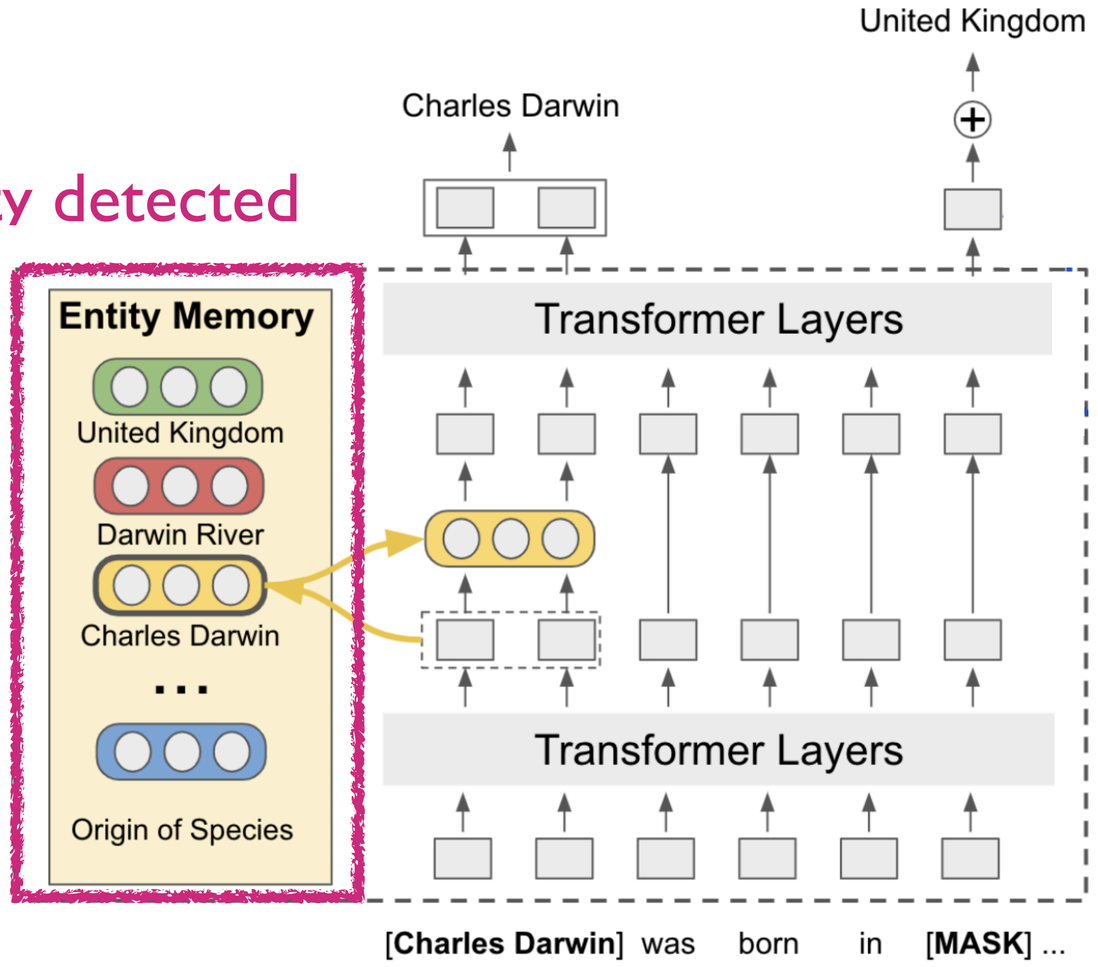
# Entities as Experts (Fevry et al. 2020)




Fevry et al. 2020. “Entities as Experts: Sparse Memory Access with Entity Supervision”

# Entities as Experts (Fevry et al. 2020)

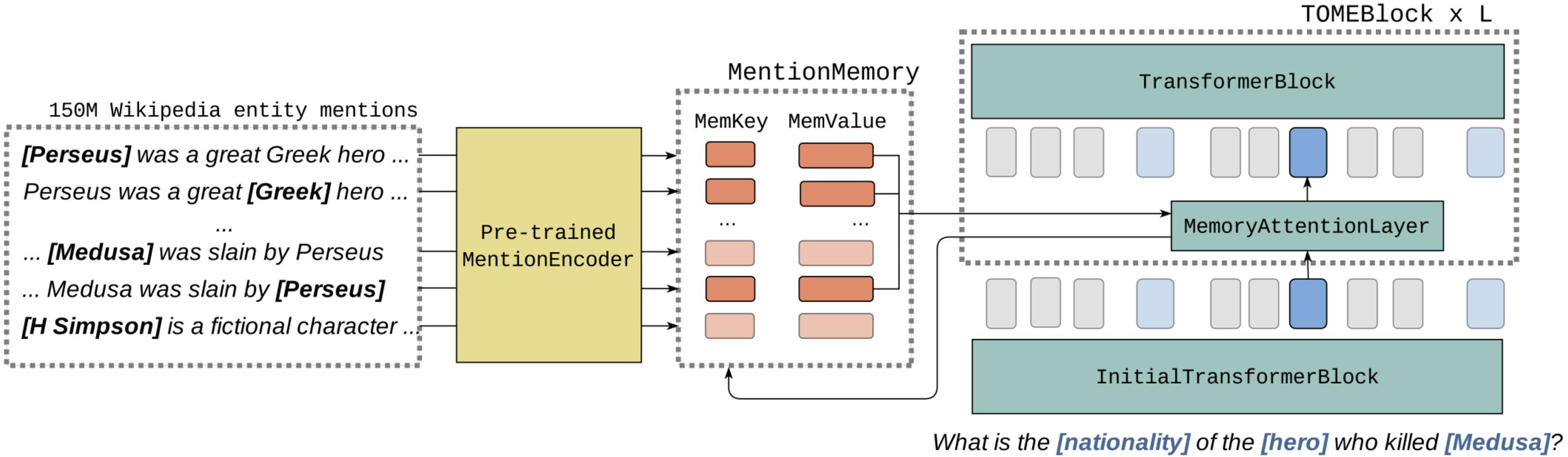
Need text with entity detected




Fevry et al. 2020. “Entities as Experts: Sparse Memory Access with Entity Supervision”

# Mention Memory (de Jong et al. 2022)

One vector per entity → One vector per entity *mention*




de Jong et al. 2022. "Mention Memory:
incorporating textual knowledge into Transformers through entity mention attention"

# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |
| FLARE (Jiang et al. 2023) | Text blocks | Input layer | Every n tokens *(adaptive)* |
| Adaptive kNN-LM (He et al 2021, Drozdov et al. 2022, Alon et al 2022, etc) | Tokens | Output layer | Every n tokens *(adaptive)* |
| Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022) | Entities or entity mentions | Intermediate layers | Every entity mentions |


Most effective with entity-centric tasks & efficient


Additional entity detection required

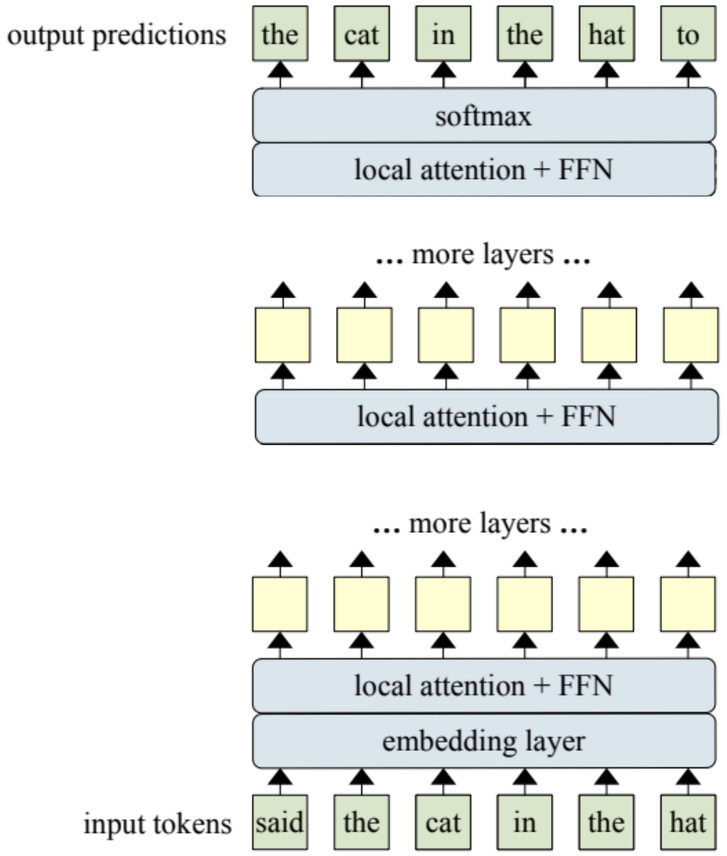
# Summary

| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |
| FLARE (Jiang et al. 2023) | Text blocks | Input layer | Every n tokens *(adaptive)* |
| Adaptive kNN-LM (He et al 2021, Drozdov et al. 2022, Alon et al 2022, etc) | Tokens | Output layer | Every n tokens *(adaptive)* |
| Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022) | Entities or entity mentions | Intermediate layers | Every entity mentions |

*All models retrieve from the external text*

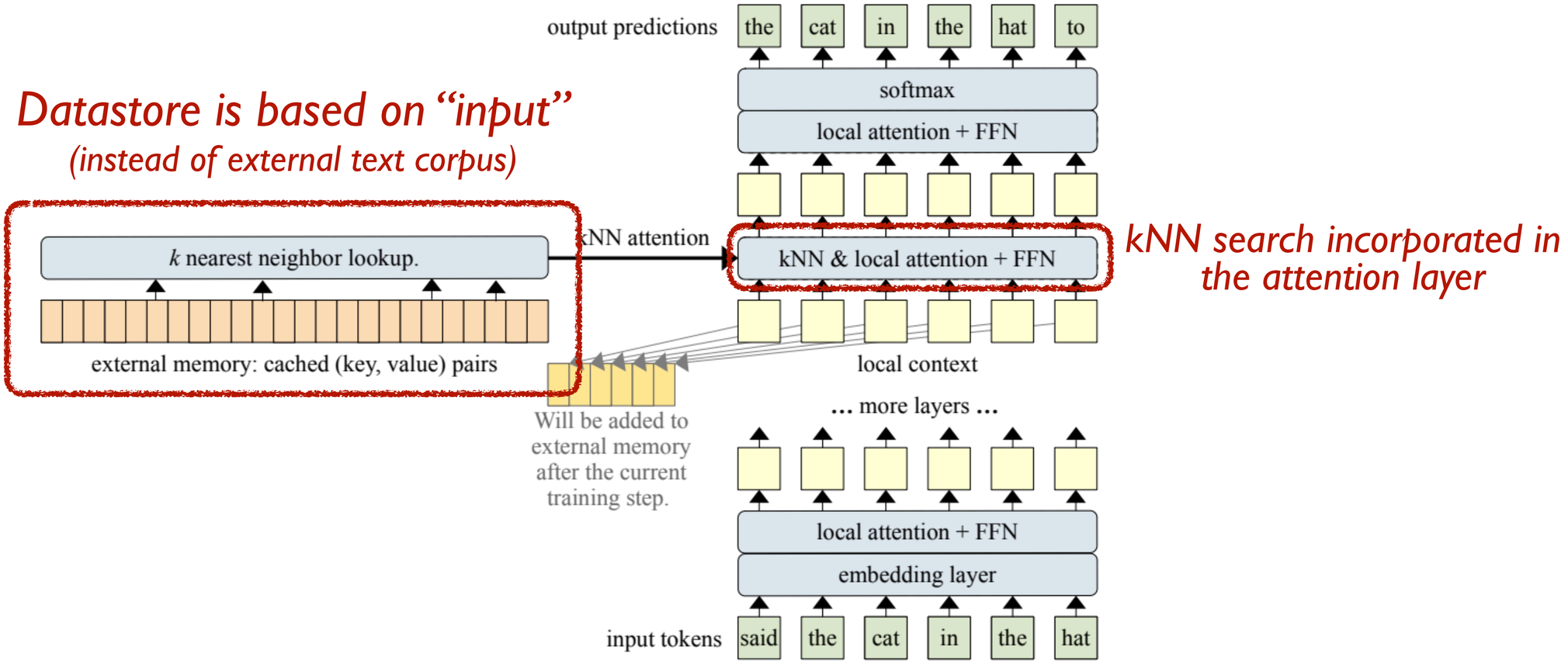
*What else can we do with these models?*

# Retrieval for long-range LM



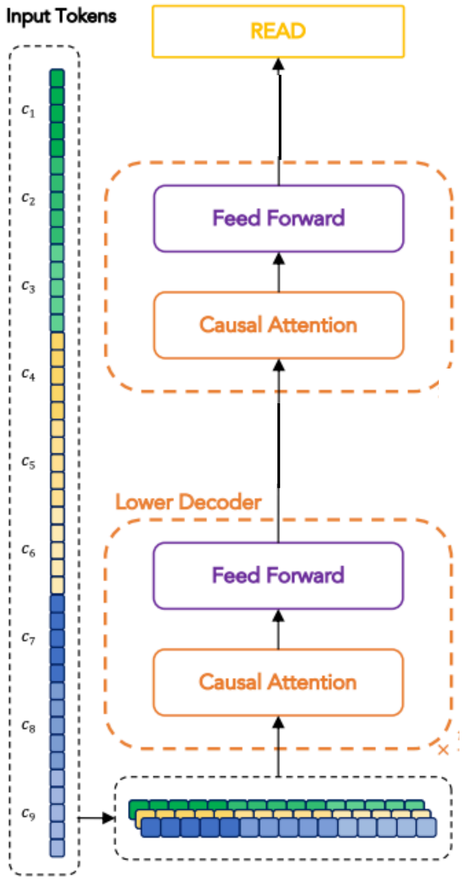

Wu et al. 2022. Memorizing Transformers **(Figure source)**
Bertsch et al. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input
Rubin & Berant. 2023. Long-range Language Modeling with Self-retrieval

# Retrieval for long-range LM



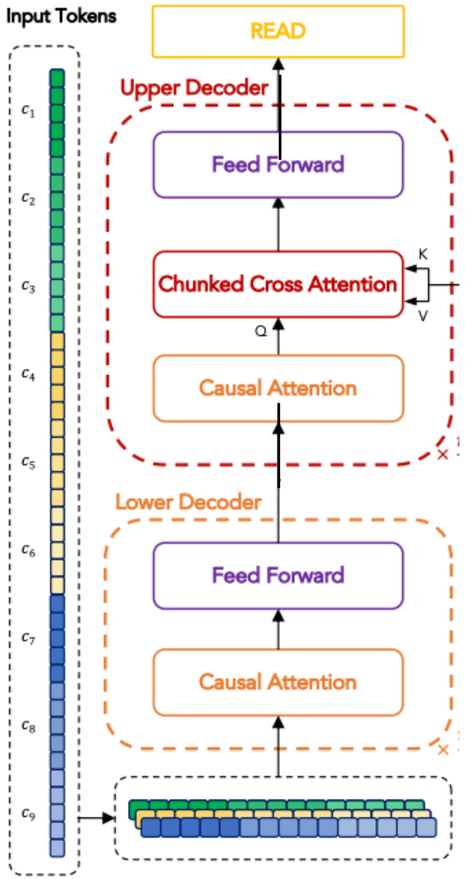

Wu et al. 2022. Memorizing Transformers **(Figure source)**
Bertsch et al. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input
Rubin & Berant. 2023. Long-range Language Modeling with Self-retrieval

# Retrieval for long-range LM




Wu et al. 2022. Memorizing Transformers
Bertsch et al. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input
Rubin & Berant. 2023. Long-range Language Modeling with Self-retrieval **(Figure source)**

# Retrieval for long-range LM

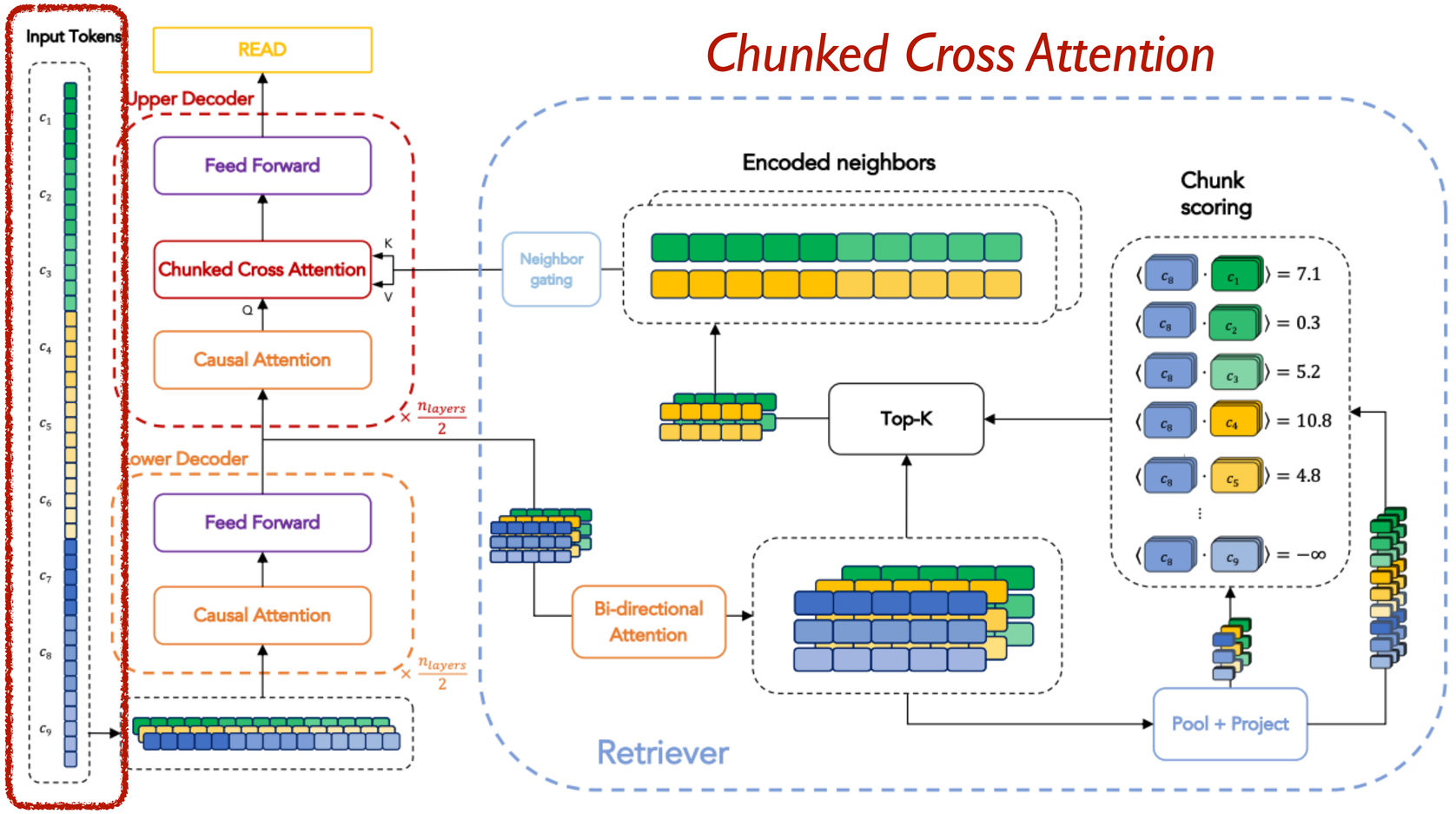



*Chunked Cross Attention*

Wu et al. 2022. Memorizing Transformers
Bertsch et al. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input
Rubin & Berant. 2023. Long-range Language Modeling with Self-retrieval **(Figure source)**

# Retrieval for long-range LM

*Datastore based on "input"*

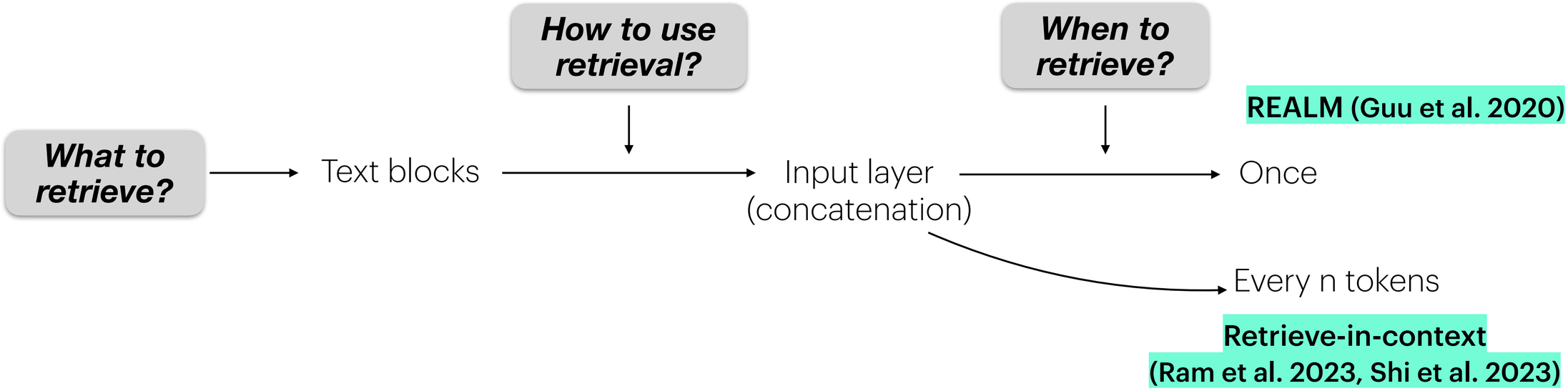



Wu et al. 2022. Memorizing Transformers
Bertsch et al. 2023. Unlimiformer: Long-Range Transformers with Unlimited Length Input
Rubin & Berant. 2023. Long-range Language Modeling with Self-retrieval **(Figure source)**

# Summary

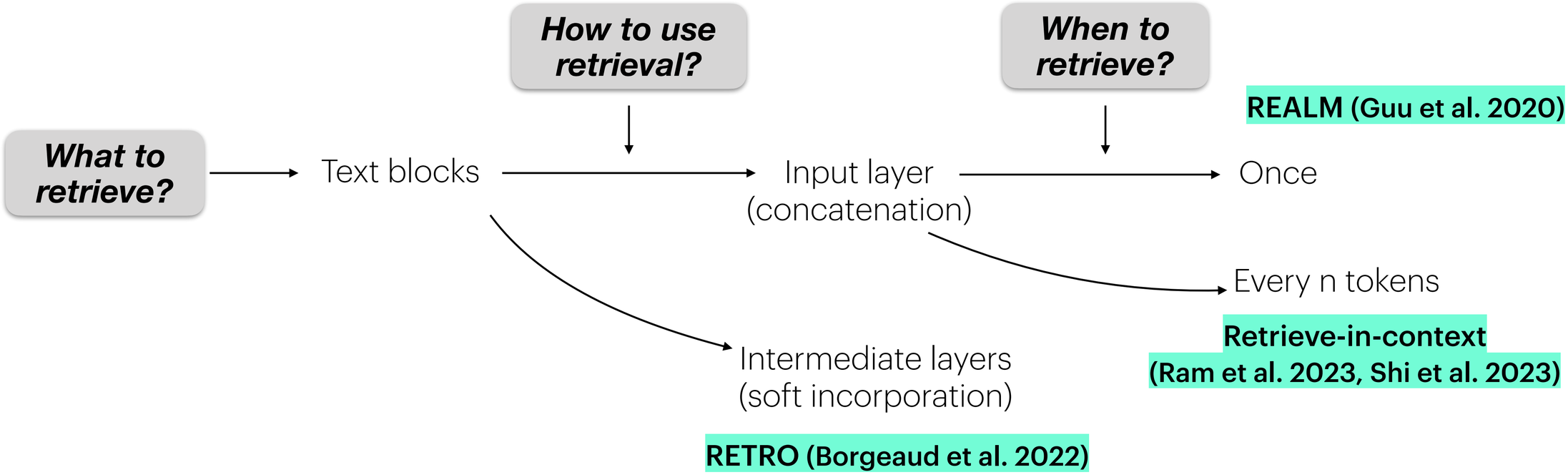
| | What do retrieve? | How to use retrieval? | When to retrieve? |
|---|---|---|---|
| REALM (Guu et al 2020) | Text blocks | Input layer | Once |
| Retrieve-in-context LM (Shi et al 2023, Ram et al 2023) | Text blocks | Input layer | Every n tokens |
| RETRO (Borgeaud et al. 2021) | Text blocks | Intermediate layers | Every n tokens |
| kNN-LM (Khandelwal et al. 2020) | Tokens | Output layer | Every token |
| FLARE (Jiang et al. 2023) | Text blocks | Input layer | Every n tokens *(adaptive)* |
| Adaptive kNN-LM (He et al 2021, Drozdov et al. 2022, Alon et al 2022, etc) | Tokens | Output layer | Every n tokens *(adaptive)* |
| Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022) | Entities or entity mentions | Intermediate layers | Every entity mentions |
| Wu et al. 2022, Bertsch et al. 2023, Rubin & Berant. 2023 | Text blocks ***from the input*** | Intermediate layers | Once or every n tokens |

# Wrapping up



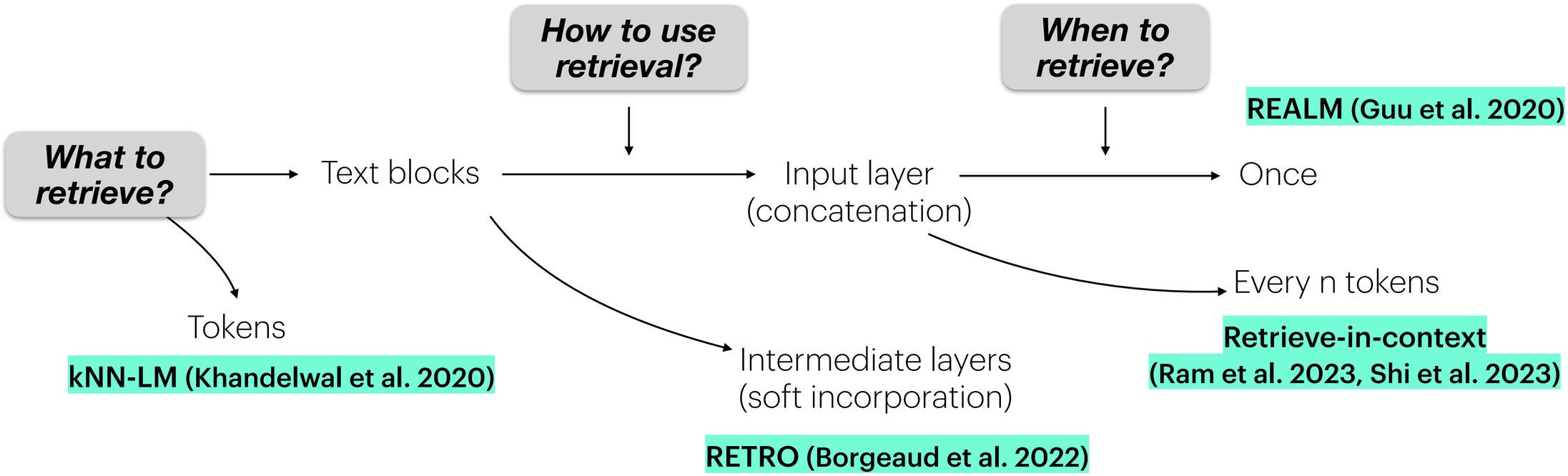

More frequent retrieval = better in performance, but slower

# Wrapping up



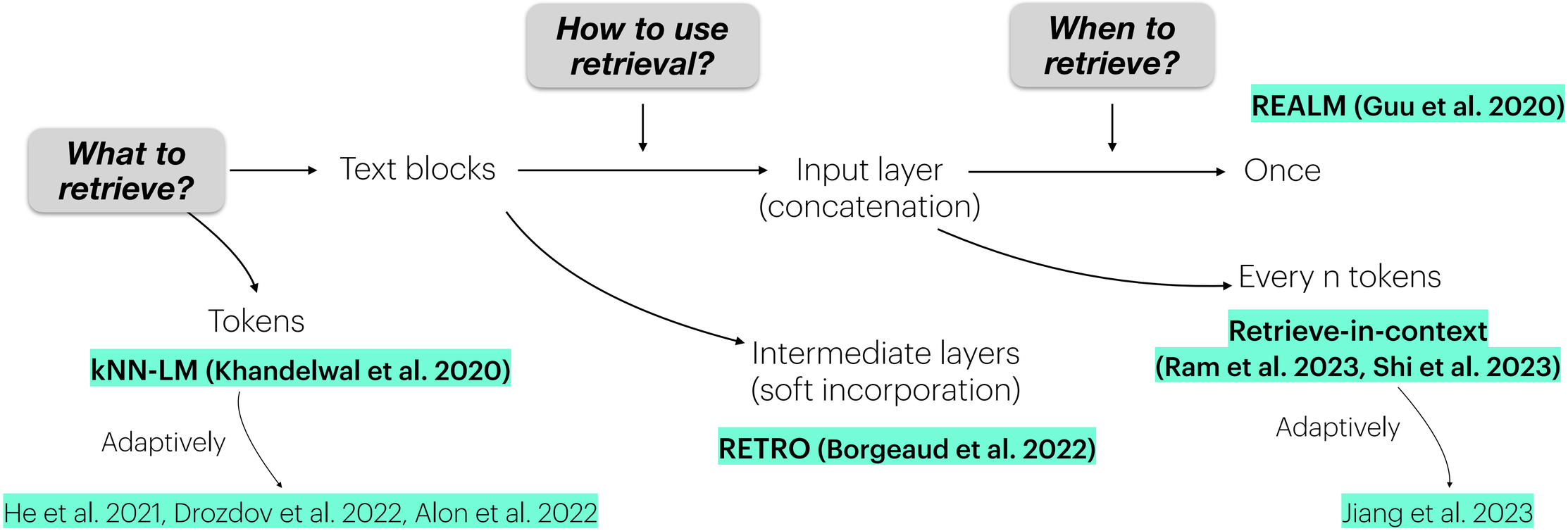

- Input layer: Simple but can be slower
- Intermediate layers: More complex (need training) but can be designed to be more efficient

# Wrapping up



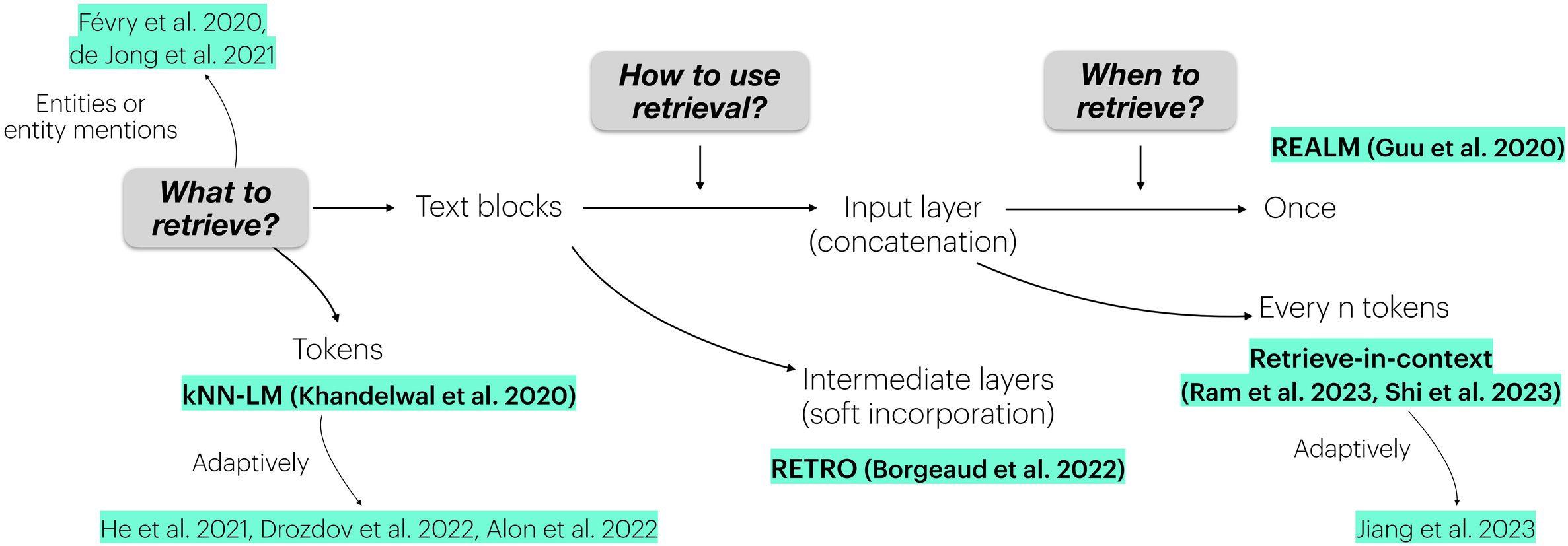

- Text blocks: Datastore can be space-efficient, more computation
- Tokens: More fine-grained, compute-efficient, but datastore can be space-expensive

# Wrapping up



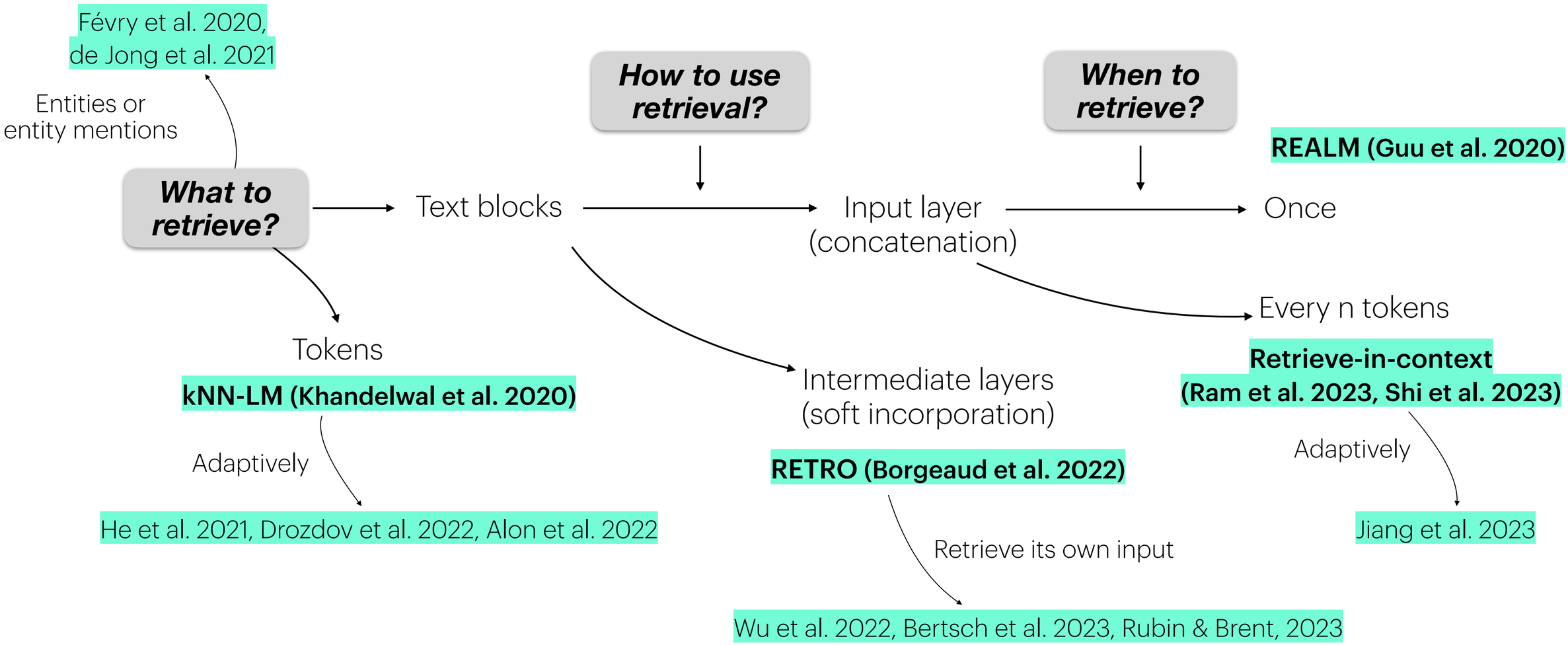

Adaptive retrieval can improve efficiency

# Wrapping up



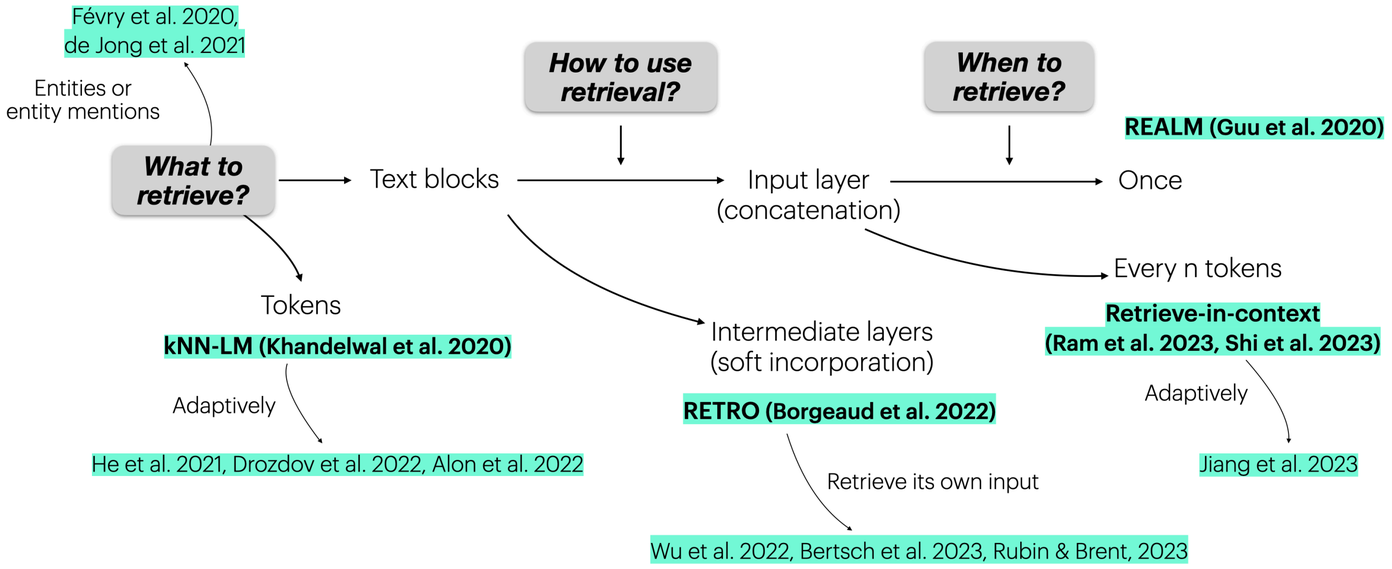

Entities or entity mentions instead of every token or chunk

# Wrapping up




We can use the same model for long-sequence modeling

# Wrapping up




We didn't cover anything about training → **Section 4**!

We briefly saw some results but not extensively on downstream tasks → **Section 5**!