Section 3:
Retrieval-based LM: Architecture
Categorization of retrieval-based LMs
Categorization of retrieval-based LMs

What to retrieve?

Query
What to retrieve?

Query

Text chunks (passages)?
Categorization of retrieval-based LMs

**What** to retrieve?

- Query

```
OF
FRIENDS

OBIX
```

- Text chunks (passages)?
- Tokens?
Categorization of retrieval-based LMs

What to retrieve?

Query

Text chunks (passages)?
- Tokens?
- Something else?
Categorization of retrieval-based LMs

**What** to retrieve?
- Query
- Text chunks (passages)?
  - Tokens?
  - Something else?

**How** to use retrieval?
- Input
- LM
- Output
Categorization of retrieval-based LMs

**What** to retrieve?
- Query
- Text chunks (passages)?
- Tokens?
- Something else?

**How** to use retrieval?
- Input
- LM
- Output
Categorization of retrieval-based LMs

**What** to retrieve?
- Query
- Text chunks (passages)?
- Tokens?
- Something else?

**How** to use retrieval?
- Input
- LM
- Output
Categorization of retrieval-based LMs

What to retrieve?
- Query
- Text chunks (passages)?
- Tokens?
- Something else?

How to use retrieval?
- Input
- LM
- Output
Categorization of retrieval-based LMs

**What to retrieve?**
- Query
  - Text chunks (passages)?
  - Tokens?
  - Something else?

**How to use retrieval?**
- Input
- LM
- Output

**When to retrieve?**
Categorization of retrieval-based LMs

**What** to retrieve?

- Query
- Text chunks (passages)?
- Tokens?
- Something else?

**How** to use retrieval?

- Input
- Output
- LM

**When** to retrieve?

- w/ retrieval
- The capital city of Ontario is Toronto.
Categorization of retrieval-based LMs

**What to retrieve?**
- Query
  - Text chunks (passages)?
  - Tokens?
  - Something else?

**How to use retrieval?**
- Input
- Output

**When to retrieve?**
- w/ retrieval
  - The capital city of Ontario is Toronto.
  - w/ retrieval w/r w/r w/r w/r w/r w/r
  - The capital city of Ontario is Toronto.
Categorization of retrieval-based LMs

**What to retrieve?**

- Query

  Text chunks (passages)?
  Tokens?
  Something else?

**How to use retrieval?**

- Input

  LM

  Output

**When to retrieve?**

- w/ retrieval
- w/r
- w/r
- w/r
- w/r
- w/r
- w/r

The capital city of Ontario is Toronto.
Roadmap
Roadmap

What to retrieve? → Text chunks
Roadmap

What to retrieve? → Text chunks → Input layer (concatenation)
Roadmap

What to retrieve? → Text chunks → Input layer (concatenation) → Once

How to use retrieval?

When to retrieve?
**Roadmap**

- **What to retrieve?**
  - Text chunks
  - Input layer (concatenation)
- **How to use retrieval?**
- **When to retrieve?**
  - Once

REALM (Guu et al. 2020)
Roadmap

**What to retrieve?**

Text chunks

**How to use retrieval?**

Input layer (concatenation)

**When to retrieve?**

REALM (Guu et al. 2020)

Every n tokens

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

Once
Roadmap

What to retrieve?

Text chunks → Input layer (concatenation) → Intermediate layers (soft incorporation) → Once

REALM (Guu et al. 2020)

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

How to use retrieval?

When to retrieve?

RETRO (Borgeaud et al. 2022)

How to use retrieval?

When to retrieve?
Roadmap

**What to retrieve?**

- Text chunks
  - Tokens
  - kNN-LM (Khandelwal et al. 2020)

**How to use retrieval?**

- Input layer (concatenation)
  - Intermediate layers (soft incorporation)
    - RETRO (Borgeaud et al. 2022)

**When to retrieve?**

- Once
  - REALM (Guu et al. 2020)
- Every n tokens
  - Retrieve-in-context (Ram et al. 2023, Shi et al. 2023)
Roadmap

**What to retrieve?**
- Tokens
  - kNN-LM (Khandelwal et al. 2020)
  - Adaptively
    - He et al. 2021, Drozdov et al. 2022, Alon et al. 2022

**How to use retrieval?**
- Text chunks

**Input layer**
- (concatenation)

**Intermediate layers**
- (soft incorporation)

**When to retrieve?**
- Once
  - REALM (Guu et al. 2020)
- Every n tokens
  - Retrieve-in-context
    - (Ram et al. 2023, Shi et al. 2023)
    - Adaptively
      - Jiang et al. 2023

**Retrieve its own input**
- Wu et al. 2022, Bertsch et al. 2023, Rubin & Brent, 2023
Roadmap

- **What to retrieve?**
  - Entities or entity mentions
  - Tokens
    - kNN-LM (Khandelwal et al. 2020)
    - Adaptively
    - He et al. 2021, Drozdov et al. 2022, Alon et al. 2022

- **How to use retrieval?**
  - Text chunks
  - Input layer (concatenation)
    - Intermediate layers (soft incorporation)
      - RETRO (Borgeaud et al. 2022)

- **When to retrieve?**
  - Once
    - REALM (Guu et al. 2020)
  - Every n tokens
    - Retrieve-in-context (Ram et al. 2023, Shi et al. 2023)
    - Adaptively
    - Jiang et al. 2023
    - Wu et al. 2022, Bertsch et al. 2023, Rubin & Brent, 2023

**This is only about “architecture”**
**Section 4 will categorize & discuss “training”**
REALM (Guu et al 2020)
REALM (Guu et al 2020)

\[ x = \text{World Cup 2022 was the last with 32 teams before the increase to [MASK] in 2026.} \]
REALM (Guu et al. 2020)

$x = \text{World Cup 2022 was the last with 32 teams before the increase to [MASK] in 2026.}$

World Cup 2022 was … the increase to [MASK] in 2026.
REALM (Guu et al 2020)

\[ x = \text{World Cup 2022 was the last with 32 teams before the increase to [MASK] in 2026.} \]

REALM (Guu et al 2020)

\[ x = \text{World Cup 2022 was the last with 32 teams before the increase to [MASK] in 2026.} \]

REALM (Guu et al 2020)

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

$k$ chunks of text (passages)

FIFA World Cup 2026 will expand to 48 teams.

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

FIFA World Cup 2026 will expand to 48 teams.

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

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

**REALM** (Guu et al 2020)

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

---

**Retrieve stage**

- \( x \)
- Retrieval
- \( k \) chunks of text (passages)
- FIFA World Cup 2026 will expand to 48 teams.

**Read stage**

- FIFA World Cup 2026 will expand to 48 teams.
- ...World Cup 2022 was … the increase to [MASK] in 2026.
- LM
- 48

REALM: (1) Retrieve stage

FIFA World Cup 2026 will expand to 48 teams.

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

Team USA celebrated after winning its match against Iran ...

Wikipedia
13M chunks (passages) (called documents in the paper)
REALM: (1) Retrieve stage

FIFA World Cup 2026 will expand to 48 teams.

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

Team USA celebrated after winning its match against Iran ...

Wikipedia 13M chunks (passages) (called documents in the paper)
REALM: (1) Retrieve stage

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

FIFA World Cup 2026 will expand to 48 teams.

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

Team USA celebrated after winning its match against Iran …

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

\[ z = \text{Encoder}(z) \]
REALM: (1) Retrieve stage

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

FIFA World Cup 2026 will expand to 48 teams.

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

Team USA celebrated after winning its match against Iran ...

Wikipedia
13M chunks (passages) (called documents in the paper)
FIFA World Cup 2026 will expand to 48 teams.

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

Team USA celebrated after winning its match against Iran ...

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

REALM: (1) Retrieve stage

\[ x = \text{Encoder}(x) \]

\[ z = \text{Encoder}(z) \]

\[ x = \text{World Cup 2022 was ... the increase to [MASK] in 2026.} \]
REALM: (I) Retrieve stage

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

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

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

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

$k$ retrieved chunks

FIFA World Cup 2026 will expand to 48 teams.

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

Team USA celebrated after winning its match against Iran ...

Wikipedia
13M chunks (passages) (called documents in the paper)
REALM: (2) Read stage

\[
[MASK] z_1 [SEP] x \rightarrow \text{LM} \rightarrow P(y \mid x, z_1)
\]

\[
[MASK] z_2 [SEP] x \rightarrow \text{LM} \rightarrow P(y \mid x, z_2)
\]

\[
[MASK] z_k [SEP] x \rightarrow \text{LM} \rightarrow P(y \mid x, z_k)
\]
REALM: (2) Read stage

\[ [\text{MASK}] z_1 \ [\text{SEP}] x \rightarrow \text{LM} \rightarrow P(y \mid x, z_1) \]

\[ [\text{MASK}] z_2 \ [\text{SEP}] x \rightarrow \text{LM} \rightarrow P(y \mid x, z_2) \]

\[ \vdots \]

\[ [\text{MASK}] z_k \ [\text{SEP}] x \rightarrow \text{LM} \rightarrow P(y \mid x, z_k) \]

Weighted average
REALM: (2) Read stage

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

Weighted average
REALM: (2) Read stage

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

from the retrieve stage

Weighted average
REALM: (2) Read stage

\[ \sum_{z \in \mathcal{D}} P(z | x) P(y | x, z) \]

from the retrieve stage

from the read stage
REALM: (2) Read stage

\[
\sum_{z \in \mathcal{D}} P(z | x) P(y | x, z)
\]

Need to approximate
→ Consider top \( k \) chunks only

Weighted average
from the retrieve stage
from the read stage
REALM: (2) Read stage

\[ \sum_{z \in \mathcal{D}} P(z | x)P(y | x, z) \]

Need to approximate
→ Consider top \( k \) chunks only

Weighted average

0 if not one of top \( k \)
## REALM (Guu et al 2020)

<table>
<thead>
<tr>
<th>What to retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers</td>
<td>- Every $n$ tokens ($n&gt;1$)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
<tr>
<td><strong>What to retrieve?</strong></td>
<td><strong>How to use retrieval?</strong></td>
<td><strong>When to retrieve?</strong></td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>- <strong>Chunks</strong> ✓</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers</td>
<td>- Every $n$ tokens ($n&gt;1$)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>
**REALM** *(Guu et al 2020)*

<table>
<thead>
<tr>
<th>What to retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks ✓</td>
<td>- Input layer ✓</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers</td>
<td>- Every n tokens (n&gt;1)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>
REALM \hspace{2mm} (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 and subsequent work

* REALM (Guu et al 2020): MLM followed by fine-tuning on open-domain QA
REALM and subsequent work

* **REALM (Guu et al 2020):** MLM followed by fine-tuning on open-domain QA
* **DPR (Karpukhin et al 2020):** Pipeline training instead of joint training, fine-tuned on open-domain QA (no explicit language modeling)
REALM and subsequent work

* **REALM (Guu et al 2020)**: MLM followed by fine-tuning on open-domain QA

* **DPR (Karpukhin et al 2020)**: Pipeline training instead of joint training, fine-tuned on open-domain QA (no explicit language modeling)

* **RAG (Lewis et al 2020)**: “Generative” instead of “masked language modeling”, fine-tuned on open-domain QA & knowledge intensive tasks (no explicit language modeling)
REALM and subsequent work

* REALM (Guu et al 2020): MLM followed by fine-tuning on open-domain QA
* DPR (Karpukhin et al 2020): Pipeline training instead of joint training, fine-tuned on open-domain QA (no explicit language modeling)
* RAG (Lewis et al 2020): “Generative” instead of “masked language modeling”, fine-tuned on open-domain QA & knowledge intensive tasks (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), fine-tuned on open-domain QA & other QA tasks

Papers that follow this approach focusing on LM perplexity have come out quite recently — Shi et al. 2023, Ram et al. 2023
REALM and subsequent work

* REALM (Guu et al 2020): MLM followed by fine-tuning on open-domain QA
* DPR (Karpukhin et al 2020): Pipeline training instead of joint training, fine-tuned on open-domain QA (no explicit language modeling)
* RAG (Lewis et al 2020): “Generative” instead of “masked language modeling”, fine-tuned on open-domain QA & knowledge intensive tasks (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), fine-tuned on open-domain QA & other QA tasks

For a while, mainly evaluated on knowledge-intensive tasks (e.g. open-domain QA) with fine-tuning (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 & knowledge intensive tasks (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 & knowledge intensive tasks
* 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”
Retrieval-in-context LM

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

Ram et al. 2023. “In-Context Retrieval-Augmented Language Models”
Retrieval-in-context LM

\[ x = \text{World Cup 2022 was the last with 32 teams, before the increase to} \]

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 papers!)

Ram et al. 2023. “In-Context Retrieval-Augmented Language Models”
Retrieval-in-context LM

\[ x = \text{World Cup 2022 was the last with 32 teams, before the increase to} \]

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

\[ \downarrow \]

**Retrieval**

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

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

\[ \downarrow \]

**LM**

48 in the 2026 tournament.

Ram et al. 2023. “In-Context Retrieval-Augmented Language Models”
Retrieval-in-context 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 LM

Is $q=x$ necessary?
Retrieval-in-context 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
Retrieval-in-context 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

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

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

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

Does not cover “tokens that will come next”
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

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

Does not cover “tokens that will come next”
The U.S. national team defeated Iran 1-0.

FIFA World Cup 2026 will expand to 48 teams.

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

Retrieval-in-context LM

Is $q=x$ necessary?
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 FIFA World Cup 2026 will expand to 48 teams. World Cup 2022 was the last with 32 teams, before the increase to

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

Does not cover “tokens that will come next”

FIFA World Cup 2026 will expand to 48 teams. more relevant to what will come next
Retrieval-in-context LM

Graphs from Ram et al. 2023

Shorter prefix (more recent tokens) as a query helps
Retrieval-in-context LM

Shorter prefix (more recent tokens) as a query helps

Graphs from Ram et al. 2023
Retrieval-in-context LM

How frequent should retrieval be?
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with

Retrieval

The 2022 FIFA World Cup (...) 32 national teams involved in the tournament.
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with
Retrieval

The 2022 FIFA World Cup (...) 32 national teams involved in the tournament. World Cup 2022 was the last with
Retrieval-in-context LM

How frequent should retrieval be?

Retrieval

The 2022 FIFA World Cup (...) 32 national teams involved in the tournament. World Cup 2022 was the last with 32 teams before the increase to 48 in the 2026 tournament.
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with

Retrieval

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

LM

32 teams before the increase to 48 in the 2026 tournament.

explained by retrieval
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with

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

32 teams before the increase to 48 in the 2026 tournament. Explained by retrieval, not really covered.
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with

Retrieval

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

LM

32 teams before the increase
How frequent should retrieval be?

World Cup 2022 was the last with 32 national teams involved in the tournament. World Cup 2022 was the last with 32 teams before the increase.

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

FIFA World Cup 2026 will expand to 48 teams.

32 teams before the increase
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with

Retrieval

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

LM

32 teams before the increase

FIFA World Cup 2026 will expand to 48 teams. World Cup 2022 was the last with 32 teams, before the increase
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with

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

32 teams before the increase

Retrieval

LM

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.
Retrieval-in-context LM

How frequent should retrieval be?

World Cup 2022 was the last with
The 2022 FIFA World Cup (...) 32 national teams involved in the tournament. World Cup 2022 was the last with
32 teams before the increase

Retrieval

LM

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.

Retrieval results from a new query explain them!
Retrieval-in-context LM

Retrieving more frequently helps

Graphs from Ram et al. 2023
Retrieval-in-context LM

Retrieving more frequently helps

Graphs from Ram et al. 2023

with cost in inference time
## Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)

<table>
<thead>
<tr>
<th>What to retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>- <strong>Chunks</strong> ✔️</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers</td>
<td>- Every n tokens (n&gt;1)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>

- **Interpretation:**
  - **Chunks** is the recommended choice for what to retrieve.
  - Retrieval can be used at different layers: 
    - Input layer
    - Intermediate layers
    - Output layer
  - Timing for retrieval:
    - Once
    - Every n tokens (n>1)
    - Every token
### Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)

<table>
<thead>
<tr>
<th>What to retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks ✓</td>
<td>- Input layer ✓</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers</td>
<td>- Every n tokens (n&gt;1)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>
### Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)

<table>
<thead>
<tr>
<th><strong>What to retrieve?</strong></th>
<th><strong>How to use retrieval?</strong></th>
<th><strong>When to retrieve?</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Chunk** ✓**</td>
<td>Input layer✓</td>
<td>Once</td>
</tr>
<tr>
<td>Tokens</td>
<td>Intermediate layers</td>
<td>Every n tokens (n&gt;1)✓</td>
</tr>
<tr>
<td>Others</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>

**Notes:**
- Retrieval can be used in different ways depending on the specific needs of the task.
- Input layer retrieval typically occurs once per document.
- Intermediate layers and output layer retrievals can occur at regular intervals such as every n tokens or every token.
Summary

<table>
<thead>
<tr>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
</tbody>
</table>

Applying the same approach to LM raised new questions which mattered less in prior work (e.g. REALM) with short inputs & short outputs
### Summary

<table>
<thead>
<tr>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
</tbody>
</table>

*can be very inefficient to retrieve many text chunks, frequently*
RETRO (Borgeaud et al. 2021)

RETRO (Borgeaud et al. 2021)

✔ Incorporation in the “intermediate layer” instead of the “input” layer → designed for many chunks, frequently, more efficiently
**RETRO** (Borgeaud et al. 2021)

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

- ✓ Scale the datastore (1.8T tokens)

RETRO (Borgeaud et al. 2021)

$\mathbf{x} =$ World Cup 2022 was the last with 32 teams, before the increase to
RETRO (Borgeaud et al. 2021)

$x = \text{World Cup 2022 was the last with 32 teams, before the increase to } x_1 x_2 x_3$
RETRO (Borgeaud et al. 2021)

\[ x = \text{World Cup 2022 was the last with 32 teams, before the increase to } \]

\[ x_1 \quad x_2 \quad x_3 \]

(k chunks of text per split)

\[ x_1 \quad \text{Retrieval Encoder} \quad \rightarrow \quad p^1_1 \ldots p^k_1 \]

\[ x_2 \quad \rightarrow \quad p^1_2 \ldots p^k_2 \]

\[ x_3 \quad \rightarrow \quad p^1_3 \ldots p^k_3 \]
RETRO (Borgeaud et al. 2021)

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

$x_1 \quad x_2 \quad x_3$

$(k \text{ chunks of text per split})$
RETRO (Borgeaud et al. 2021)

\[ x = \text{World Cup 2022 was the last with 32 teams, before the increase to} \]

\[ x_1 \quad x_2 \quad x_3 \]

(k chunks of text per split)

\[ \mathbf{p}_1^1 \ldots \mathbf{p}_1^k \quad \mathbf{p}_2^1 \ldots \mathbf{p}_2^k \quad \mathbf{p}_3^1 \ldots \mathbf{p}_3^k \]
**RETRO** (Borgeaud et al. 2021)

\[ x = \text{World Cup 2022 was the last with 32 teams, before the increase to} \]

\[ x_1 \quad x_2 \quad x_3 \]

\[ (k \text{ chunks of text per split}) \]

\[ x_1 \xrightarrow{} \text{Retrieval} \quad \quad \text{Index} \quad \quad \text{LM Encoder} \]

\[ p_1^1 \ldots p_r^k \quad p_2^1 \ldots p_r^k \quad p_3^1 \ldots p_r^k \]

\[ E_1 \quad E_2 \quad E_3 \]

\[ (A \quad r \times k \times d \text{ matrix}) \]

\[ (r = \# \text{ tokens per text chunk}) \]

\[ (d = \text{hidden dimension}) \]

\[ (k = \# \text{retrieved chunks per split}) \]
Regular decoder

Transformers blocks (xL)
Decoder in RETRO

\[ E_1 \quad E_2 \quad E_3 \]

\[ x_1 \quad x_2 \quad x_3 \]

EMB \quad ATTN \quad CCA \quad FFN \quad HEAD

RETRO blocks \((xL)\)

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 $CA(H^+, E)$ $CA(H^+, E)$
Chunked Cross Attention

Outputs from the previous layer $H$ to the next layer $H'$.

Cross-attention can be computed \textit{in parallel}.
Chunked Cross Attention

Cross-attention can be computed \textit{in parallel}

If you generated until here
Chunked Cross Attention

Outputs from the previous layer $H$  
Inputs to the next layer

Cross-attention can be computed \textit{in parallel}

You get this

If you generated until here

$CA(H^+_1, E_1)$  
$CA(H^+_2, E_2)$
Chunked Cross Attention

Cross-attention can be computed in parallel
Chunked Cross Attention

Cross-attention can be computed \textit{in parallel}

Outputs from the previous layer \( H \)

Inputs to the next layer

This part can be re-used

If you generated until here
## Results

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

Significant improvements by retrieving from 1.8 trillion tokens

Perplexity: The lower the better
## Results

Perplexity: The lower the better

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

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)*

<table>
<thead>
<tr>
<th>What to retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks ✓</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers ✓</td>
<td>- Every n tokens ($n&gt;1$)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>
### Summary

<table>
<thead>
<tr>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
</tr>
</tbody>
</table>
**Summary**

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>

👍 Can use many blocks, more frequently, more efficiently
### Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>

- **Can use many blocks, more frequently, more efficiently**
- **Additional complexity; Can’t be used without training** *(more in section 4)*
<table>
<thead>
<tr>
<th>Model</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>

What else?
kNN-LM (Khandelwal et al. 2020)

kNN-LM (Khandelwal et al. 2020)

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

kNN-LM (Khandelwal et al. 2020)

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

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

kNN-LM (Khandelwal et al. 2020)

<table>
<thead>
<tr>
<th>Test Context</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td></td>
</tr>
<tr>
<td>Obama’s birthplace is</td>
<td>?</td>
</tr>
</tbody>
</table>
kNN-LM (Khandelwal et al. 2020)

<table>
<thead>
<tr>
<th>Test Context</th>
<th>Target</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>x</em></td>
<td>?</td>
<td><em>q = f(x)</em></td>
</tr>
<tr>
<td>Obama’s birthplace is</td>
<td>?</td>
<td>![Representation Image]</td>
</tr>
</tbody>
</table>

Classification

\[ p_{LM}(y) \]

- Hawaii: 0.2
- Illinois: 0.2
- ...: ...

...
... 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, ....

<table>
<thead>
<tr>
<th>Test Context</th>
<th>Target</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama’s birthplace is</td>
<td>?</td>
<td>$q = f(x)$</td>
</tr>
</tbody>
</table>
**kNN-LM** (Khandelwal et al. 2020)

<table>
<thead>
<tr>
<th>Training Contexts $C_i$</th>
<th>Targets $v_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama was senator for</td>
<td>Illinois</td>
</tr>
<tr>
<td>Barack is married to</td>
<td>Michelle</td>
</tr>
<tr>
<td>Obama was born in</td>
<td>Hawaii</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Obama is a native of</td>
<td>Hawaii</td>
</tr>
</tbody>
</table>

... 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, ....

<table>
<thead>
<tr>
<th>Test Context $x$</th>
<th>Target</th>
<th>Representation $q = f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama's birthplace is</td>
<td>?</td>
<td><img src="image" alt="Representation" /></td>
</tr>
</tbody>
</table>
**kNN-LM** (Khandelwal et al. 2020)

The size of the datastore = # of tokens in the corpus (> 1B)

<table>
<thead>
<tr>
<th>Training Contexts $C_i$</th>
<th>Targets $v_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama was senator for Illinois</td>
<td>Barack is married to Michelle</td>
</tr>
<tr>
<td>Barack is married to Hawaii</td>
<td>Michelle Hawaii</td>
</tr>
<tr>
<td>Obama was born in Hawaii</td>
<td>Obama is a native of Hawaii</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Context $x$</th>
<th>Target</th>
<th>Representation $q = f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama’s birthplace is</td>
<td>?</td>
<td><img src="image" alt="Representation" /></td>
</tr>
</tbody>
</table>
kNN-LM (Khandelwal et al. 2020)

<table>
<thead>
<tr>
<th>Training Contexts $C_i$</th>
<th>Targets $U_i$</th>
<th>Representations $k_i = f(c_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama was senator for</td>
<td>Illinois</td>
<td><img src="image" alt="Representation" /></td>
</tr>
<tr>
<td>Barack is married to</td>
<td>Michelle</td>
<td></td>
</tr>
<tr>
<td>Obama was born in</td>
<td>Hawaii</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Obama is a native of</td>
<td>Hawaii</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Test Context $x$</th>
<th>Target</th>
<th>Representation $q = f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Obama’s birthplace is</td>
<td>?</td>
<td><img src="image" alt="Representation" /></td>
</tr>
</tbody>
</table>
kNN-LM (Khandelwal et al. 2020)

Which tokens in a datastore are close to the next token?
Which tokens in a datastore are close to the next token?

Which prefixes in a datastore are close to the prefix we have?
kNN-LM (Khandelwal et al. 2020)

Which tokens in a datastore 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)

- **Training Contexts** $C_i$
  - Obama was senator for
  - Barack is married to
  - Obama was born in
    - Hawaii
  - Obama is a native of

- **Targets** $U_i$
  - Illinois
  - Michelle
  - Hawaii

- **Representations** $k_i = f(c_i)$
  - 4
  - 100
  - 5
  - 3

- **Distances** $d_i = d(q, k_i)$
  - Hawaii: 3
  - Illinois: 4
  - Hawaii: 5

- **Nearest k**
  - Hawaii: 0.7
  - Illinois: 0.2

- **Normalization** $p_k(k_i) \propto \exp(-d_i)$

- **Aggregation** $p_{kNN}(y) = \sum_{i=1}^{k} p_k(k_i)$
  - Hawaii: 0.8
  - Illinois: 0.2

- **Test Context** $x$
  - Obama’s birthplace is

- **Target**
  - ?

- **Representation** $q = f(x)$
$P_{kNN}(y | x) \propto \sum_{(k,v) \in \Omega} \mathbb{1}[v = y] \text{sim}(k, x)$
kNN-LM (Khandelwal et al. 2020)

\[ P_{kNN}(y| x) \propto \sum_{(k, v) \in \mathcal{D}} \mathbb{1}[v = y]\text{sim}(k, x) \quad \text{sim}(k, x) = \exp \left( -d(\text{Enc}(k), \text{Enc}(x)) \right) \]
**kNN-LM** (Khandelwal et al. 2020)

\[
P_{kNN}(y|x) \propto \sum_{(k,v) \in \mathcal{D}} [v = y] \text{sim}(k, x) \quad \text{sim}(k, x) = \exp \left( -d(\text{Enc}(k), \text{Enc}(x)) \right)
\]
kNN-LM \textsuperscript{(Khandelwal et al. 2020)}

\begin{equation}
P_{kNN}(y \mid x) \propto \sum_{(k,v) \in \mathcal{D}} \mathbb{1}[v = y] \text{sim}(k, x)
\text{sim}(k, x) = \exp\left(-d(\text{Enc}(k), \text{Enc}(x))\right)
\end{equation}
**kNN-LM** (Khandelwal et al. 2020)

\[ P_{k\text{NN}}(y \mid x) \propto \sum_{(k,v) \in \mathcal{D}} [v = y] \text{sim}(k, x) \]

\[ \text{sim}(k, x) = \exp \left( -d(\text{Enc}(k), \text{Enc}(x)) \right) \]
kNN-LM (Khandelwal et al. 2020)

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

Later work, e.g., NPM (Min et al. 2023) removed interpolation (more in Section 4)
**kNN-LM** (Khandelwal et al. 2020)

\[
P_{kNN-LM}(y | x) = (1 - \lambda)P_{LM}(y | x) + \lambda P_{kNN}(y | x)
\]

Later work, e.g., NPM (Min et al. 2023) removed interpolation (more in Section 4)
**kNN-LM** (Khandelwal et al. 2020)

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

Later work, e.g., NPM (Min et al. 2023) removed interpolation (more in Section 4)
**kNN-LM** (Khandelwal et al. 2020)

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

Later work, e.g., NPM (Min et al. 2023) removed interpolation (more in Section 4)
## kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very cheap</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use torch</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
# kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
## kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
## kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this Item delivered broken. Very cheap</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use torch</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a torch</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the torch</td>
<td>torch</td>
</tr>
</tbody>
</table>
kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td></td>
</tr>
</tbody>
</table>

Dense vector space
kNN-LM - why?

Dense vector space

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>

10/10, would buy this cheap

... affordable

... nice

... good

... bad

... poor

... terrible
kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>

Dense vector space

- 10/10, would buy this **cheap**
- ... affordable
- ... nice
- ... good
- ... bad
- ... poor
- ... terrible

Item delivered broken. Very **cheap**
### kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td><strong>To check the version of PyTorch, you can use</strong></td>
<td>torch</td>
</tr>
<tr>
<td><strong>You are permitted to bring a</strong></td>
<td>torch</td>
</tr>
<tr>
<td><strong>A group of infections ... one of the</strong></td>
<td>torch</td>
</tr>
</tbody>
</table>
### kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
## kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very cheap</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a torch</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very cheap</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a torch</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
### kNN-LM - why?

**Dense vector space**

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very cheap</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a torch</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
kNN-LM - why?

Dense vector space

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>

PyTorch, you can use `torch`
kNN-LM - why?

Dense vector space

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this cheap Item delivered broken. Very cheap</td>
<td>torch</td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use torch</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a torch</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the torch</td>
<td></td>
</tr>
</tbody>
</table>
kNN-LM - why?

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/10, would buy this</td>
<td>cheap</td>
</tr>
<tr>
<td>Item delivered broken. Very cheap</td>
<td></td>
</tr>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>You are permitted to bring a torch</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections ... one of the</td>
<td></td>
</tr>
</tbody>
</table>
The lower the better
kNN-LM - results

The lower the better

Perplexity

Size of datastore (in billions)

No-retrieval LM

Wiki-100M

Wiki-3B

kNN-LM (Wiki-100M + kNN)
kNN-LM - results

The lower the better

Perplexity

Size of datastore (in billions)

Wiki-100M

Wiki-3B

kNN-LM (Wiki-100M + kNN)

No-retrieval LM

30x larger No-retrieval LM
kNN-LM - results

The lower the better

![Graph showing the performance of kNN-LM compared to No-retrieval LM with different dataset sizes.](image)

- **No-retrieval LM**
- **30x larger No-retrieval LM**

The graph illustrates the perplexity of different models as a function of the size of the datastore. The lower the perplexity, the better the performance.
kNN-LM - results

The lower the better

- kNN-LM
- No-retrieval LM

Outperforms no-retrieval LM

Perplexity

Size of datastore (in billions)

Wiki-100M
Wiki-3B
kNN-LM (Wiki-100M + kNN)

30x larger No-retrieval LM
kNN-LM - results

The lower the better

- kNN-LM vs. No-retrieval LM
- Outperforms no-retrieval LM
- Better with bigger datastore

- Wiki-100M
- Wiki-3B
- kNN-LM (Wiki-100M + kNN)

30x larger No-retrieval LM
kNN-LM - results

Better with bigger $k$
**kNN-LM - results**

**Better with bigger $k$**

**Helps more out-of-domain**
kNN-LM - results

Better with bigger $k$

Can use in-domain datastore even if parameters were not trained in-domain

Helps more out-of-domain
<table>
<thead>
<tr>
<th><strong>What</strong> to retrieve?</th>
<th><strong>How</strong> to use retrieval?</th>
<th><strong>When</strong> to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- Tokens</td>
<td>- Intermediate layers</td>
<td>- Every $n$ tokens ($n&gt;1$)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>

**kNN-LM** (Khandelwal et al. 2020)
### kNN-LM (Khandelwal et al. 2020)

<table>
<thead>
<tr>
<th><strong>What to retrieve?</strong></th>
<th><strong>How to use retrieval?</strong></th>
<th><strong>When to retrieve?</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- <strong>Tokens ✔</strong></td>
<td>- Intermediate layers</td>
<td>- Every $n$ tokens ($n&gt;1$)</td>
</tr>
<tr>
<td>- Others</td>
<td>- Output layer</td>
<td>- Every token</td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th><strong>What to retrieve?</strong></th>
<th><strong>How to use retrieval?</strong></th>
<th><strong>When to retrieve?</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>- Chunks</td>
<td>- Input layer</td>
<td>- Once</td>
</tr>
<tr>
<td>- <strong>Tokens ✓</strong></td>
<td>- Intermediate layers</td>
<td>- Every n tokens ($n &gt; 1$)</td>
</tr>
<tr>
<td>- Others</td>
<td>- <strong>Output layer ✓</strong></td>
<td>- <strong>Every token ✓</strong></td>
</tr>
</tbody>
</table>
## Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>Method</td>
<td>What do retrieve?</td>
<td>How to use retrieval?</td>
<td>When to retrieve?</td>
</tr>
<tr>
<td>------------------------------</td>
<td>-------------------</td>
<td>-----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>
## Summary

<table>
<thead>
<tr>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
</tr>
</tbody>
</table>

👍 More fine-grained; Can be better at rare patterns & out-of-domain
### Summary

<table>
<thead>
<tr>
<th></th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>

👍 More fine-grained; Can be better at rare patterns & out-of-domain
Can be very efficient (as long as kNN search is fast)
## Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>

👍 More fine-grained; Can be better at rare patterns & out-of-domain
   Can be very efficient (as long as kNN search is fast)

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

<table>
<thead>
<tr>
<th>Model</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>

More fine-grained; Can be better at rare patterns & out-of-domain

Can be very efficient (as long as kNN search is fast)

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

(Wikipedia) 13M vs. 4B
### Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>(Shi et al 2023, Ram et al 2023)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>

- More fine-grained; Can be better at rare patterns & out-of-domain
- Can be very efficient (as long as kNN search is fast)

- **Datastore is expensive in space:** given the same data, # text chunks vs. # tokens
- No cross attention between input and retrieval results
## Extensions

<table>
<thead>
<tr>
<th>Model</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>
## Extensions

<table>
<thead>
<tr>
<th>Models</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
</tbody>
</table>

*It’s fixed! Can we do adaptively?*
Adaptive retrieval for efficiency

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

Adaptive retrieval of tokens (following kNN-LM)
Adaptive retrieval of chunks

- Judge necessity

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

**FLARE (Jiang et al. 2023)**

- Retrieval (Datastore + Index)
- Language Model

Jiang et al. “Active Retrieval Augmented Generation”
Adaptive retrieval of chunks
- Judge necessity

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

Joe Biden (born November 20, 1942) is the 46th president of the United States.
Adaptive retrieval of chunks
- 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 chunks
- 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 chunks
- 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 chunks
- 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.
Adaptive retrieval of chunks
- 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].

**FLARE (Jiang et al. 2023)**

- Retrieval (Datastore + Index)
- Language Model

Jiang et al. “Active Retrieval Augmented Generation”
Adaptive retrieval of chunks
- 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].

FLARE (Jiang et al. 2023)

Retrieval (Datastore + Index)

Language Model

Jiang et al. “Active Retrieval Augmented Generation”
Adaptive retrieval of chunks
- 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].

Joe Biden
At the University of Delaware in Newark, Biden ... earned a Bachelor of Arts degree in 1965 with a double major in history and political science.

Jiang et al. “Active Retrieval Augmented Generation”
Adaptive retrieval of *chunks* 
- *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].

Joe Biden
At the University of Delaware in Newark, Biden ... earned a Bachelor of Arts degree in 1965 with a double major in history and political science.

Jiang et al. “Active Retrieval Augmented Generation”
Adaptive retrieval of chunks
- 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*
Adaptive retrieval of *tokens*

- *Judge necessity*

Joe Biden graduated from the University of Delaware.
Adaptive retrieval of *tokens*

- *Judge necessity*

Joe Biden graduated from the University of Delaware.
Adaptive retrieval of tokens
- Judge necessity

Joe Biden graduated from the University of Delaware.

Adaptive retrieval of tokens

- Judge necessity

Joe Biden graduated from the University of Delaware.

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

Adaptive retrieval of tokens
- Judge necessity

Joe Biden graduated from the University of Delaware.

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

A function of the input \( x \)

\[ \rightarrow \lambda = 0 \text{ if } \lambda < \gamma \]

Adaptive retrieval of tokens
- Use local info

Adaptive retrieval of tokens
- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University of Delaware in Newark</td>
<td>At the University of Delaware in Newark</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
</tbody>
</table>

Joe Biden graduated from

Adaptive retrieval of tokens

- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University of Delaware</td>
<td>retrieve</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University of Delaware.

Adaptive retrieval of tokens

- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University</td>
<td>University</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>Delaware</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
<tr>
<td>At the University of Delaware in</td>
<td></td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University

Adaptive retrieval of tokens

- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University of Delaware</td>
<td>the</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>of</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>in</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>Newark</td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University of

Adaptive retrieval of tokens
- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University of Delaware</td>
<td>Joe Biden graduated from the University of Delaware</td>
</tr>
<tr>
<td></td>
<td>in Newark</td>
</tr>
</tbody>
</table>

Adaptive retrieval of *tokens*

- *Use local info*

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the</td>
<td>University of Delaware</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>Newark</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td></td>
</tr>
</tbody>
</table>

Joe Biden graduated from **the**

Adaptive retrieval of tokens
- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the</td>
<td>the</td>
</tr>
<tr>
<td>At the University</td>
<td>University</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>Delaware</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td></td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University

Adaptive retrieval of tokens

- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University</td>
<td>At the University</td>
</tr>
<tr>
<td>At the University</td>
<td>At the University</td>
</tr>
<tr>
<td>At the University</td>
<td>At the University</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>At the University of Delaware</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td>At the University of Delaware in Newark</td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University of

Adaptive retrieval of tokens
- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University of Delaware</td>
<td>Pointer</td>
</tr>
<tr>
<td>At the University of Delaware in Newark</td>
<td>Pointer</td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University of Delaware.

Adaptive retrieval of tokens

- Use local info

<table>
<thead>
<tr>
<th>Training contexts</th>
<th>Targets</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the University of Delaware in Newark</td>
<td>At the University of Delaware</td>
</tr>
<tr>
<td>At the University of Delaware</td>
<td>Retrieve once, and save other searches!</td>
</tr>
</tbody>
</table>

Joe Biden graduated from the University of Delaware.

## Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens (adaptive)</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens (adaptive)</td>
</tr>
</tbody>
</table>
### Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al. 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>(Shi et al. 2023, Ram et al. 2023)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Adaptive kNN-LM</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>

*More efficient*
## Summary

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>

- More efficient
- Decision may not always be optimal
<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens</td>
</tr>
</tbody>
</table>

What else beyond text chunks and tokens?
Entities as Experts (Fevry et al. 2020)
Entities as Experts  (Fevry et al. 2020)
Entities as Experts (Fevry et al. 2020)
Entities as Experts (Fevry et al. 2020)
Entities as Experts (Fevry et al. 2020)
Entities as Experts (Fevry et al. 2020)

Dense vector space
Entities as Experts (Fevry et al. 2020)

Entities as Experts (Fevry et al. 2020)

(Wikipedia)
chunks: 13 millions
tokens: 4 billions
entities: 6 millions

Entities as Experts (Fevry et al. 2020)

(Wikipedia)
chunks: 13 millions
tokens: 4 billions
entities: 6 millions

Entities as Experts (Fevry et al. 2020)

(Wikipedia) chunks: 13 millions
tokens: 4 billions
entities: 6 millions

Mention Memory (de Jong et al. 2022)

One vector per entity → One vector per entity mention
Mention Memory (de Jong et al. 2022)

One vector per entity → One vector per entity mention

[Perseus] was a great Greek hero ...
Perseus was a great [Greek] hero ...
... [Medusa] was slain by Perseus
... Medusa was slain by [Perseus]
[H Simpson] is a fictional character ...

What is the [nationality] of the [hero] who killed [Medusa]?

TOMEBlock x L

TransformerBlock

MemoryAttentionLayer

InitialTransformerBlock

MemKey MemValue

de Jong et al. 2022. “Mention Memory: incorporating textual knowledge into Transformers through entity mention attention”
Mention Memory (de Jong et al. 2022)

One vector per entity $\rightarrow$ One vector per entity mention

[Perseus] was a great Greek hero...
Perseus was a great [Greek] hero...
... [Medusa] was slain by Perseus
... Medusa was slain by [Perseus]
[H Simpson] is a fictional character...

What is the [nationality] of the [hero] who killed [Medusa]?

de Jong et al. 2022. “Mention Memory: incorporating textual knowledge into Transformers through entity mention attention”
Mention Memory (de Jong et al. 2022)

(Wikipedia)
chunks: 13M
tokens: 4B
etties: 6M
entity mentions: 150M

One vector per entity mention

[Perseus] was a great Greek hero ...
Perseus was a great [Greek] hero ...
... [Medusa] was slain by Perseus
... Medusa was slain by [Perseus]
[H Simpson] is a fictional character ...

TOMEBlock x L

What is the [nationality] of the [hero] who killed [Medusa]?

de Jong et al. 2022. “Mention Memory: incorporating textual knowledge into Transformers through entity mention attention”
## Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)</td>
<td>Entities or entity mentions</td>
<td>Intermediate layers</td>
<td>Every entity mentions</td>
</tr>
</tbody>
</table>
## Summary

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

Most effective for entity-centric tasks & space-efficient
<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al. 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al. 2023, Ram et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al. 2021, Alon et al. 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)</td>
<td>Entities or entity mentions</td>
<td>Intermediate layers</td>
<td>Every entity mentions</td>
</tr>
</tbody>
</table>

Most effective for entity-centric tasks & space-efficient

Additional entity detection required
# Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)</td>
<td>Entities or entity mentions</td>
<td>Intermediate layers</td>
<td>Every entity mentions</td>
</tr>
</tbody>
</table>

*All models retrieve from the external text*
## Summary

<table>
<thead>
<tr>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
</tr>
<tr>
<td>Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)</td>
<td>Entities or entity mentions</td>
<td>Intermediate layers</td>
</tr>
</tbody>
</table>

**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 (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

**Datastore is based on “input”**
*(instead of external text corpus)*

---

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

Datastore is based on “input” (instead of external text corpus)

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

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

Every chunk is assigned a similarity score

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

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

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

<table>
<thead>
<tr>
<th>Method</th>
<th>What do retrieve?</th>
<th>How to use retrieval?</th>
<th>When to retrieve?</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al 2020)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Once</td>
</tr>
<tr>
<td>Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>RETRO (Borgeaud et al. 2021)</td>
<td>Text chunks</td>
<td>Intermediate layers</td>
<td>Every n tokens</td>
</tr>
<tr>
<td>kNN-LM (Khandelwal et al. 2020)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every token</td>
</tr>
<tr>
<td>FLARE (Jiang et al. 2023)</td>
<td>Text chunks</td>
<td>Input layer</td>
<td>Every n tokens (adaptive)</td>
</tr>
<tr>
<td>Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)</td>
<td>Tokens</td>
<td>Output layer</td>
<td>Every n tokens (adaptive)</td>
</tr>
<tr>
<td>Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)</td>
<td>Entities or entity mentions</td>
<td>Intermediate layers</td>
<td>Every entity mentions</td>
</tr>
<tr>
<td>Wu et al. 2022, Bertsch et al. 2023, Rubin &amp; Berant. 2023</td>
<td>Text chunks from the input</td>
<td>Intermediate layers</td>
<td>Once or every n tokens</td>
</tr>
</tbody>
</table>
Wrapping up
Wrapping up

What to retrieve? → Text chunks → Input layer (concatenation) → When to retrieve?

How to use retrieval?

REALM (Guu et al. 2020)

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

Every n tokens

Once
Wrapping up

What to retrieve? Text chunks → Input layer (concatenation) → How to use retrieval? → When to retrieve?

REALM (Guu et al. 2020)
Retrieve-in-context (Ram et al. 2023, Shi et al. 2023)

More frequent retrieval = better in performance, but slower
Wrapping up

How to use retrieval?

What to retrieve?

Text chunks

Input layer (concatenation)

Intermediate layers (soft incorporation)

When to retrieve?

REALM (Guu et al. 2020)

Every n tokens

Retrieval-in-context (Ram et al. 2023, Shi et al. 2023)

Once

RETRO (Borgeaud et al. 2022)
Wrapping up

What to retrieve?

Text chunks → Input layer (concatenation) → Intermediate layers (soft incorporation) → Once

How to use retrieval?

REALM (Guu et al. 2020)

When to retrieve?

Every n tokens

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

RETRO (Borgeaud et al. 2022)

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

**What to retrieve?**
- Tokens
- Text chunks
- kNN-LM (Khandelwal et al. 2020)

**How to use retrieval?**
- Input layer (concatenation)
- Intermediate layers (soft incorporation)

**When to retrieve?**
- Once
- Every n tokens
- REALM (Guu et al. 2020)
- Retrieve-in-context (Ram et al. 2023, Shi et al. 2023)
- RETRO (Borgeaud et al. 2022)
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

What to retrieve?
- Text chunks
- Tokens
  - kNN-LM (Khandelwal et al. 2020)
  - Adaptively
  - He et al. 2021, Alon et al. 2022

How to use retrieval?
- Input layer (concatenation)
- Intermediate layers (soft incorporation)
  - RETRO (Borgeaud et al. 2022)

When to retrieve?
- Once
- Every n tokens
  - REALM (Guu et al. 2020)
  - Retrrieve-in-context (Ram et al. 2023, Shi et al. 2023)
  - Adaptively
  - Jiang et al. 2023

Adaptive retrieval can improve efficiency
Wrapping up

Entities or entity mentions instead of every token or chunk
Wrapping up

**What to retrieve?**
- Tokens
  - kNN-LM (Khandelwal et al. 2020)
  - Adaptively
  - He et al. 2021, Alon et al. 2022
- Entities or entity mentions

**How to use retrieval?**
- Text chunks
  - Input layer (concatenation)
  - Intermediate layers (soft incorporation)
  - RETRO (Borgeaud et al. 2022)

**When to retrieve?**
- Once
- Every n tokens
  - REALM (Guu et al. 2020)
  - Retrieve-in-context (Ram et al. 2023, Shi et al. 2023)
  - Adaptively
  - Jiang et al. 2023

We can use a similar approach for long-sequence modeling.
Wrapping up
(not covered in this section)

**What to retrieve?**
- Text chunks
  - Tokens
    - kNN-LM (Khandelwal et al. 2020)
      - Adaptively
    - He et al. 2021, Alon et al. 2022
- Entities or entity mentions
  - REALM (Guu et al. 2020)
  - REALM (Guu et al. 2020)
  - REALM (Guu et al. 2020)
  - REALM (Guu et al. 2020)

**How to use retrieval?**
- Input layer (concatenation)
  - Intermediate layers (soft incorporation)
    - RETRO (Borgeaud et al. 2022)
      - Min et al. 2023
        - Removing interpolation
      - Wu et al. 2022, Bertsch et al. 2023, Rubin & Brent, 2023
  - Retrieve-in-context
    - (Ram et al. 2023, Shi et al. 2023)
      - Adaptively
    - (Ram et al. 2023, Shi et al. 2023)
      - Adaptively

**When to retrieve?**
- Once
  - Every n tokens
    - REALM (Guu et al. 2020)
    - REALM (Guu et al. 2020)
    - REALM (Guu et al. 2020)
    - REALM (Guu et al. 2020)

**Extend to use fact triples**
- Févry et al. 2020, de Jong et al. 2021

**Soft adaptation for better expressivity**
- Drozdov et al. 2022

**Use encoder-decoder to scale # of chunks to process**
- Izcard et al. 2022

**Intermediate layers (soft incorporation)**
- Min et al. 2023
  - Removing interpolation
  - Wu et al. 2022, Bertsch et al. 2023, Rubin & Brent, 2023
Wrapping up
Wrapping up
Wrapping up

Still largely under-explored!
Wrapping up

We didn’t cover anything about training → Section 4!
We briefly saw some results but not extensively on downstream tasks → Section 5!