Retrieval-based Language Models

Tianle Cai and Beiqi Zou
Nov 9th, 2022
Outline

1). Motivation: why retrieval-based LMs?
2). Related Work: existing retrieval-based LMs
3). Method: RETRO (Borgeaud et al., 2022)
4). Results
Prior work: GPT-2 & GPT-3

- GPT-3 is massive!
- 175B parameters (~117x GPT-2)

Figure from blog post
Motivation

- It seems scaling larger and larger models is the main way of improving the performance...
Motivation

- It seems scaling larger and larger models is the main way of improving the performance...

  But with a tremendous increase in training energy cost!
Motivation

- It seems scaling larger and larger models is the main way of improving the performance...

  But with a tremendous increase in training energy cost!

  1). Additional computations at training and inference time

  2). Increased memorization of the training data
Motivation

- It seems scaling larger and larger models is the main way of improving the performance...

But with a tremendous increase in training energy cost!

1). Additional computations at training and inference time

2). Increased memorization of the training data

Can we separate language information from world knowledge information?
Retrieval-based Language Models

- Knowledge is encoded explicitly
- The model **learns to search** for relevant passages, then **use the retrieved information** for crafting knowledgeable response

Figure from [stanford blog post](https://example.com)
Why is retrieval important?

- Tackling inefficiency
  - Retrieval-based models can be much **smaller and faster**
Why is retrieval important?

- Tackling inefficiency
  - Retrieval-based models can be much **smaller and faster**

- Tackling static knowledge
  - The retrieval knowledge store can be **efficiently updated or expanded** by modifying the text corpus

---

**GPT-3**

Who is the president of the United States?

The current president of the United States is Donald Trump.

Who is the president of the United States in 2022?

The current president of the United States is Donald Trump. In 2022, the president will be either Trump or his successor.

*Figure from Danqi’s talk*
Why is retrieval important?

- **Tackling inefficiency**
  - Retrieval-based models can be much **smaller and faster**

- **Tackling static knowledge**
  - The retrieval knowledge store can be **efficiently updated or expanded** by modifying the text corpus

- **Tackling Opaqueness**
  - We are able to inspect the sources the model retrieved, which is more **transparent**
## Existing methods

<table>
<thead>
<tr>
<th></th>
<th># Retrieval tokens</th>
<th>Granularity</th>
<th>Retriever training</th>
<th>Retrieval integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Cache</td>
<td>$O \left( 10^3 \right)$</td>
<td>Token</td>
<td>Frozen (LSTM)</td>
<td>Add to probs</td>
</tr>
<tr>
<td>kNN-LM</td>
<td>$O \left( 10^9 \right)$</td>
<td>Token</td>
<td>Frozen (Transformer)</td>
<td>Add to probs</td>
</tr>
<tr>
<td>SPALM</td>
<td>$O \left( 10^9 \right)$</td>
<td>Token</td>
<td>Frozen (Transformer)</td>
<td>Gated logits</td>
</tr>
<tr>
<td>DPR</td>
<td>$O \left( 10^9 \right)$</td>
<td>Prompt</td>
<td>Contrastive proxy</td>
<td>Extractive QA</td>
</tr>
<tr>
<td>REALM</td>
<td>$O \left( 10^9 \right)$</td>
<td>Prompt</td>
<td>End-to-End</td>
<td>Prepend to prompt</td>
</tr>
<tr>
<td>RAG</td>
<td>$O \left( 10^9 \right)$</td>
<td>Prompt</td>
<td>Fine-tuned DPR</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>FiD</td>
<td>$O \left( 10^9 \right)$</td>
<td>Prompt</td>
<td>Frozen DPR</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>EMDR$^2$</td>
<td>$O \left( 10^9 \right)$</td>
<td>Prompt</td>
<td>End-to-End (EM)</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>RETRO (ours)</td>
<td>$O \left( 10^{12} \right)$</td>
<td>Chunk</td>
<td>Frozen (BERT)</td>
<td>Chunked cross-attention</td>
</tr>
</tbody>
</table>

**Type 1:** Token-level Retrieval (mainly) for LM – augmenting prediction of next token

**Type 2:** Passage-level Retrieval (mainly) for QA – retrieving passages relevant to the question
Type 1: *Token*-level retrieval for *LM*

- Augment LM model with *kNN*-based model.
- Target is the next token.

![Type 1: Token-level retrieval for LM](image_url)
Type 1: *Token*-level retrieval for *LM*

\[
p_{\text{kNN}}(y|x) \propto \sum_{(k_i,v_i) \in \mathcal{N}} \mathbb{1}_{y=v_i} \exp(-d(k_i, f(x)))
\]

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

- No interaction between context encoder (for retrieval) and LM during training.
- What’s the relationship between lambda and the size of database?
Type 1: *Token*-level retrieval for *LM*

> No interaction between Context encoder and *LM* during training.

How to train them together?

- **SPALM**: Adding an extra gating network to post-process the retrieved data.
- **TRIME**: Training with in-batch memories.
  - Incorporating retrieval into the training objective:
    
    \[
    P(w \mid c) \propto \exp(E_w^T f_\theta(c)) + \sum_{(c_j, x_j) \in M_{\text{train}} : x_j = w} \exp(\text{sim}(g_\theta(c), g_\theta(c_j))).
    \]
Type 1: *Token*-level retrieval for *LM*

Prediction (target: “Apple”)

Jobs became CEO of _

Figure from TRIME paper (Zhong et al. 2022)
## Existing methods

<table>
<thead>
<tr>
<th></th>
<th># Retrieval tokens</th>
<th>Granularity</th>
<th>Retriever training</th>
<th>Retrieval integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Cache</td>
<td>$O(10^3)$</td>
<td>Token</td>
<td>Frozen (LSTM)</td>
<td>Add to probs</td>
</tr>
<tr>
<td>kNN-LM</td>
<td>$O(10^9)$</td>
<td>Token</td>
<td>Frozen (Transformer)</td>
<td>Add to probs</td>
</tr>
<tr>
<td>SPALM</td>
<td>$O(10^9)$</td>
<td>Token</td>
<td>Frozen (Transformer)</td>
<td>Gated logits</td>
</tr>
<tr>
<td>DPR</td>
<td>$O(10^9)$</td>
<td>Prompt</td>
<td>Contrastive proxy</td>
<td>Extractive QA</td>
</tr>
<tr>
<td>REALM</td>
<td>$O(10^9)$</td>
<td>Prompt</td>
<td>End-to-End</td>
<td>Prepend to prompt</td>
</tr>
<tr>
<td>RAG</td>
<td>$O(10^9)$</td>
<td>Prompt</td>
<td>Fine-tuned DPR</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>FiD</td>
<td>$O(10^9)$</td>
<td>Prompt</td>
<td>Frozen DPR</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>EMDR²</td>
<td>$O(10^9)$</td>
<td>Prompt</td>
<td>End-to-End (EM)</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>RetRo (ours)</td>
<td>$O(10^{12})$</td>
<td>Chunk</td>
<td>Frozen (BERT)</td>
<td>Chunked cross-attention</td>
</tr>
</tbody>
</table>

**Type 1:** *Token-level Retrieval (mainly) for LM* – augmenting prediction of next token

**Type 2:** *Passage-level Retrieval (mainly) for QA* – retrieving passages relevant to the question
Type 2: *Passage*-level Retrieval for QA

- Contrastively train the retriever.
- Can be plugged into a QA system for retrieving context.

Figure from the talk of DRP paper (Karpukhin et al. 2020)
Type 2: *Passage*-level Retrieval for QA

Figure from the talk of DRP paper (Karpukhin et al. 2020)
## Existing methods

<table>
<thead>
<tr>
<th>Method</th>
<th># Retrieval tokens</th>
<th>Granularity</th>
<th>Retriever training</th>
<th>Retrieval integration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Continuous Cache</td>
<td>$O\left(10^3\right)$</td>
<td>Token</td>
<td>Frozen (LSTM)</td>
<td>Add to probs</td>
</tr>
<tr>
<td>kNN-LM</td>
<td>$O\left(10^9\right)$</td>
<td>Token</td>
<td>Frozen (Transformer)</td>
<td>Add to probs</td>
</tr>
<tr>
<td>SPALM</td>
<td>$O\left(10^9\right)$</td>
<td>Token</td>
<td>Frozen (Transformer)</td>
<td>Gated logits</td>
</tr>
<tr>
<td>DPR</td>
<td>$O\left(10^9\right)$</td>
<td>Prompt</td>
<td>Contrastive proxy</td>
<td>Extractive QA</td>
</tr>
<tr>
<td>REALM</td>
<td>$O\left(10^9\right)$</td>
<td>Prompt</td>
<td>End-to-End</td>
<td>Prepend to prompt</td>
</tr>
<tr>
<td>RAG</td>
<td>$O\left(10^9\right)$</td>
<td>Prompt</td>
<td>Fine-tuned DPR</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>FiD</td>
<td>$O\left(10^9\right)$</td>
<td>Prompt</td>
<td>Frozen DPR</td>
<td>Cross-attention</td>
</tr>
<tr>
<td>EMDR²</td>
<td>$O\left(10^9\right)$</td>
<td>Prompt</td>
<td>End-to-End (EM)</td>
<td>Cross-attention</td>
</tr>
<tr>
<td><strong>Retro (ours)</strong></td>
<td>$O\left(10^{12}\right)$</td>
<td>Chunk</td>
<td>Frozen (BERT)</td>
<td>Chunked cross-attention</td>
</tr>
</tbody>
</table>

“Limited” scale:
- Datasets are up to billions of tokens.
- Models are ~100M parameters.
Improving language models by retrieving from trillions of tokens

Sebastian Borgeaud†, Arthur Mensch†, Jordan Hoffmann†, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae‡, Erich Elsen‡ and Laurent Sifre†,‡

All authors from DeepMind, †Equal contributions, ‡Equal senior authorship
This paper: RETRO architecture (for LM)
Component 1: Frozen BERT encoder for retrieval

Why frozen the encoders given that training them is helpful (as shown in previous works like DPR)?
Component 1: Frozen BERT encoder for retrieval

Why frozen the encoders given that training them is helpful (as shown in previous works like DPR)?

> “avoid having to periodically re-compute embeddings over the entire database during training”
Component 1: Frozen BERT encoder for retrieval

- Format of the retrieval neighbors: \([N, F]\) where \(N\) is used as key and \(F\) is the continuation of \(N\).
- Metric: \(d(C, N) = \|\text{BERT}(C) - \text{BERT}(N)\|\).
- \(\text{RET}(C) = ([N^1, F^1], ..., [N^k, F^k])\).
Component 2: Chunked cross-attention (CCA)

Background

- Input chunks: Divide input of length 2048 into chunks of length 64.
- N, F in the retrieval database are also of length 64.
Component 2: Chunked cross-attention (CCA)

How to maintain causality?

- Chunk-wise autoregressive
- Adding (encoded) neighbor of chunk $i$ to the last token of chunk $i$ and chunk $i+1$.
- Intuition: Ideally if neighbor is exactly same as chunk, its continuation will be the next chunk.
Miscellaneous

- Encoder for post-processing neighbors: A small (19M params) BERT encoder for conditioning neighbors on query.
- In the implementation, the retrieval models contain one RETRO-block every 3 blocks, starting from layer 6. (Why?)
Q1. Describe how the text is stored in RETRO's database (keys and values) and how they are encoded and integrated into the language model.

- **Format of the retrieval neighbors:**
  - \([N, F]\) where \(N\) is used as key and \(F\) is the continuation of \(N\).
- **Chunked cross-attention.**
Experiments Outline

1). Models and Datasets
2). Scaling on Models and Data
3). RETRO-fitting
4). RETRO on Question Answering
5). Evaluations on leakage filtering
Models

1). Baseline Transformer
   - Replace LayerNorm with RMSNorm
   - Relative position encodings

2). RETRO [Off]
   - Without retrieval data

3). RETRO [On]
Models

1). Baseline Transformer
   - Replace LayerNorm with RMSNorm
   - Relative position encodings

2). RETRO [Off]
   - Without retrieval data

3). RETRO [On]
Models

1). Baseline Transformer
   - Replace LayerNorm with RMSNorm
   - Relative position encodings

2). RETRO [Off]
   - Without retrieval data

3). RETRO [On]
Models

1). Baseline Transformer

2). RETRO [Off]

3). RETRO [On]

<table>
<thead>
<tr>
<th>Baseline parameters</th>
<th>RETRO</th>
<th>d</th>
<th>$d_{ffw}$</th>
<th># heads</th>
<th>Head size</th>
<th># layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>132M</td>
<td>172M (+30%)</td>
<td>896</td>
<td>3,584</td>
<td>16</td>
<td>64</td>
<td>12</td>
</tr>
<tr>
<td>368M</td>
<td>425M (+15%)</td>
<td>1,536</td>
<td>6,144</td>
<td>12</td>
<td>128</td>
<td>12</td>
</tr>
<tr>
<td>1,309M</td>
<td>1,451M (+11%)</td>
<td>2,048</td>
<td>8,192</td>
<td>16</td>
<td>128</td>
<td>24</td>
</tr>
<tr>
<td>6,982M</td>
<td>7,532M (+8%)</td>
<td>4,096</td>
<td>16,384</td>
<td>32</td>
<td>128</td>
<td>32</td>
</tr>
</tbody>
</table>
# Models

1). Baseline Transformer

2). RETRO [Off]

3). RETRO [On]

<table>
<thead>
<tr>
<th>Baseline parameters</th>
<th>RETRO</th>
<th>(d)</th>
<th>(d_{ffw})</th>
<th># heads</th>
<th>Head size</th>
<th># layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>132M</td>
<td>172M (+30%)</td>
<td>896</td>
<td>3,584</td>
<td>16</td>
<td>64</td>
<td>12</td>
</tr>
<tr>
<td>368M</td>
<td>425M (+15%)</td>
<td>1,536</td>
<td>6,144</td>
<td>12</td>
<td>128</td>
<td>12</td>
</tr>
<tr>
<td>1,309M</td>
<td>1,451M (+11%)</td>
<td>2,048</td>
<td>8,192</td>
<td>16</td>
<td>128</td>
<td>24</td>
</tr>
<tr>
<td>6,982M</td>
<td>7,532M (+8%)</td>
<td>4,096</td>
<td>16,384</td>
<td>32</td>
<td>128</td>
<td>32</td>
</tr>
</tbody>
</table>

Less percentage increase for larger models
Datasets

- Multilingual version of MassiveText (Rae et al., 2021) for both training and retrieval data

<table>
<thead>
<tr>
<th>Source</th>
<th>Token count (M)</th>
<th>Documents (M)</th>
<th>Multilingual</th>
<th>Sampling frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web</td>
<td>977,563</td>
<td>1,208</td>
<td>Yes</td>
<td>55%</td>
</tr>
<tr>
<td>Books</td>
<td>3,423,740</td>
<td>20</td>
<td>No</td>
<td>25%</td>
</tr>
<tr>
<td>News</td>
<td>236,918</td>
<td>398</td>
<td>No</td>
<td>10%</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>13,288</td>
<td>23</td>
<td>Yes</td>
<td>5%</td>
</tr>
<tr>
<td>GitHub</td>
<td>374,952</td>
<td>143</td>
<td>No</td>
<td>5%</td>
</tr>
</tbody>
</table>
Datasets

- C4 (Raffel et al., 2020)
- The Pile (Gao et al., 2020)
- Curation Corpus (Curation, 2020)
- A set of manually selected Wikipedia articles

- WikiText-103 (Merity et al., 2017)
- Lambada (Paperno et al., 2016)

Bits-per-byte (bpb)

Perplexity

Accuracy
Example Data from LAMBADA

- Designed to evaluate the capabilities of computational models for text understanding by means of a word prediction task
- Models must be able to keep track of information in the broader discourse
- Measured in *accuracy*

*Context:* “Why?” “I would have thought you’d find him rather dry,” she said. “I don’t know about that,” said Gabriel. “He was a great craftsman,” said Heather. “That he was,” said Flannery.

*Target sentence:* “And Polish, to boot,” said ______.

*Target word:* Gabriel
Evaluation Metric: Bits-per-bytes

\[ \forall \alpha \in [0, 1], \quad C_\alpha \triangleq \{ C \in C, r(C) \leq \alpha \}, \quad \text{bpb}(\alpha) \triangleq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)} \]
Evaluation Metric: Bits-per-bytes

\[ \forall \alpha \in [0, 1], \quad C_\alpha \triangleq \{ C \in C, \text{r}(C) \leq \alpha \}, \quad \text{bpb}(\alpha) \triangleq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)} \]
Evaluation Metric: Bits-per-bytes

1). Split the evaluation sequences into chunks of length $m \leq 64$

2). For each evaluation chunk $C$, retrieve 10 closest neighbours in the training data

3). Compute the longest token substring common to both the evaluation chunk and its neighbours

$$r(C) = \frac{s}{m}$$
Evaluation Metric: Bits-per-bytes

1). Split the evaluation sequences into chunks of length \( m \leq 64 \)

2). For each evaluation chunk \( C \), retrieve 10 closest neighbours in the training data

3). Compute the longest token substring common to both the evaluation chunk and its neighbours

\[ r(C) = \frac{s}{m} \]

- Ranges from 0 (chunk never seen) to 1 (chunk entirely seen)
- Indicates how much overlap there is between the evaluation chunk and training data
Evaluation Metric: Bits-per-bytes

4). Obtain the log-likelihood of each chunk $C$, and the number of bytes it encodes

*Filtered bits-per-bytes (bpb)* as follows:

$$\forall \alpha \in [0, 1], \quad C_\alpha \triangleq \{C \in C, r(C) \leq \alpha\}, \quad \text{bpb}(\alpha) \triangleq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)}$$
Evaluation Metric: Bits-per-bytes

4). Obtain the log-likelihood of each chunk $C$, and the number of bytes it encodes

Filtered bits-per-bytes (bpb) as follows:

$$\forall \alpha \in [0, 1], \quad C_\alpha \doteq \{ C \in C, r(C) \leq \alpha \}, \quad \text{bpb}(\alpha) \doteq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)}$$

Shows bpb on the set of chunks that overlap less than $\alpha\%$ with the training chunks
Evaluation Metric: Bits-per-bytes

4). Obtain the log-likelihood of each chunk $C$, and the number of bytes it encodes

**Filtered bits-per-bytes (bpb)** as follows:

$$\forall \alpha \in [0, 1], \quad C_\alpha \triangleq \{C \in C, r(C) \leq \alpha\}, \quad bpb(\alpha) \triangleq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)}$$

Shows bpb on the set of chunks that overlap less than $\alpha\%$ with the training chunks

Full evaluation **bits-per-bytes (bpb)** performance is recovered by bpb(1)

C4, The Pile, Curation, Manually selected Wiki articles
Evaluation Metric: Bits-per-bytes

4). Obtain the log-likelihood of each chunk C, and the number of bytes it encodes

Filtered bits-per-bytes (bpb) as follows:

\[ \forall \alpha \in [0, 1], \quad C_\alpha \triangleq \{ C \in C, r(C) \leq \alpha \}, \quad \text{bpb}(\alpha) = \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)} \]

Shows bpb on the set of chunks that overlap less than \( \alpha \)% with the training chunks

Full evaluation bits-per-bytes (bpb) performance is recovered by bpb(1)

Tokenizer agnostic The lower, the better
Model Scaling

- On all datasets, RETRO outperforms the baseline at all model sizes.
- Improvements do not diminish as we scale the models.
Model Scaling

- On all datasets, RETRO outperforms the baseline at all model sizes
- Improvements do not diminish as we scale the models
Data Scaling

- Scaling the retrieval database at evaluation improves performance
Relative bpb improvement on the Pile

- RETRO outperforms baseline on almost all datasets except \textit{dm\_mathematics} and \textit{ubuntu\_irc}

- \textbf{Jurassic-1} and \textbf{Gopher} outperform GPT-3!
RETRO-fitting

● Extend baseline models into RETRO models
● Freeze the pre-trained weights
● Only train chunked cross-attention and neighbour encoder parameters
RETRO-fitting

- RETRO-fitting Models quickly surpasses the performance of baseline models
- Close to RETRO models trained from scratch
Performance on QA

- Fine-tune on the Natural Questions dataset
- Measures exact string match accuracy

Format in: “question: {question} \n answer: {answer}”

Figure from Natural Questions
Performance on QA

- Fine-tune on the Natural Questions dataset
- Measures exact string match accuracy

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{REALM} (Guu et al., 2020)</td>
<td>40.4</td>
</tr>
<tr>
<td>\texttt{DPR} (Karpukhin et al., 2020)</td>
<td>41.5</td>
</tr>
<tr>
<td>\texttt{RAG} (Lewis et al., 2020)</td>
<td>44.5</td>
</tr>
<tr>
<td>\texttt{EMDR}^2 (Sachan et al., 2021)</td>
<td>52.5</td>
</tr>
<tr>
<td>\texttt{FiD} (Izacard and Grave, 2021)</td>
<td>51.4</td>
</tr>
<tr>
<td>\texttt{FiD + Distill.} (Izacard et al., 2020)</td>
<td>54.7</td>
</tr>
<tr>
<td>Baseline 7B (closed book)</td>
<td>30.4</td>
</tr>
<tr>
<td>\texttt{RETRO} 7.5B (DPR retrieval)</td>
<td>45.5</td>
</tr>
</tbody>
</table>
Performance on QA

- **RETRO 7.5B (DPR retrieval)**
  - Has access to the question as well as the **top 20 DPR Wiki passages** and their titles via CCA

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REALM</strong> (Guu et al., 2020)</td>
<td>40.4</td>
</tr>
<tr>
<td><strong>DPR</strong> (Karpukhin et al., 2020)</td>
<td>41.5</td>
</tr>
<tr>
<td><strong>RAG</strong> (Lewis et al., 2020)</td>
<td>44.5</td>
</tr>
<tr>
<td><strong>EMDR^2</strong> (Sachan et al., 2021)</td>
<td>52.5</td>
</tr>
<tr>
<td><strong>FiD</strong> (Izacard and Grave, 2021)</td>
<td>51.4</td>
</tr>
<tr>
<td><strong>FiD + Distill.</strong> (Izacard et al., 2020)</td>
<td>54.7</td>
</tr>
<tr>
<td><strong>Baseline 7B</strong> (closed book)</td>
<td>30.4</td>
</tr>
<tr>
<td><strong>RETRO 7.5B</strong> (DPR retrieval)</td>
<td>45.5</td>
</tr>
</tbody>
</table>
Performance on QA

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>REALM (Guu et al., 2020)</td>
<td>40.4</td>
</tr>
<tr>
<td>DPR (Karpukhin et al., 2020)</td>
<td>41.5</td>
</tr>
<tr>
<td>RAG (Lewis et al., 2020)</td>
<td>44.5</td>
</tr>
<tr>
<td>EMDR² (Sachan et al., 2021)</td>
<td>52.5</td>
</tr>
<tr>
<td>FID (Izacard and Grave, 2021)</td>
<td>51.4</td>
</tr>
<tr>
<td>FID + Distill. (Izacard et al., 2020)</td>
<td>54.7</td>
</tr>
<tr>
<td>Baseline 7B (closed book)</td>
<td>30.4</td>
</tr>
<tr>
<td>RETRO 7.5B (DPR retrieval)</td>
<td>45.5</td>
</tr>
</tbody>
</table>
## Performance on QA

<table>
<thead>
<tr>
<th>Model</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>REALM</strong> (Guu et al., 2020)</td>
<td>40.4</td>
</tr>
<tr>
<td><strong>DPR</strong> (Karpukhin et al., 2020)</td>
<td>41.5</td>
</tr>
<tr>
<td><strong>RAG</strong> (Lewis et al., 2020)</td>
<td>44.5</td>
</tr>
<tr>
<td><strong>EMDR²</strong> (Sachan et al., 2021)</td>
<td>52.5</td>
</tr>
<tr>
<td><strong>FiD</strong> (Izacard and Grave, 2021)</td>
<td>51.4</td>
</tr>
<tr>
<td><strong>FiD + Distill.</strong> (Izacard et al., 2020)</td>
<td>54.7</td>
</tr>
<tr>
<td>Baseline 7B (closed book)</td>
<td>30.4</td>
</tr>
<tr>
<td><strong>RETO 7.5B</strong> (DPR retrieval)</td>
<td>45.5</td>
</tr>
</tbody>
</table>
Performance wrt. Dataset Leakage

- Filtered bp as eval loss

\[ bp(\alpha) = \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)} \]
Performance wrt. Dataset Leakage

- Filtered bpb as eval loss

\[
bpb(\alpha) = \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)}
\]

More details in table 19
Performance wrt. Dataset Leakage

- Filtered bpb as eval loss

\[ \text{bpb}(\alpha) \triangleq \frac{\sum_{C \in C_{\alpha}} \ell(C)}{\sum_{C \in C_{\alpha}} N(C)} \]
Performance wrt. Dataset Leakage

- Filtered bpb as eval loss

\[ bpb(\alpha) \triangleq \frac{\sum_{C \in C_\alpha} \ell(C)}{\sum_{C \in C_\alpha} N(C)} \]
Q2. Describe how RETRO defines dataset leakage. Do retrieval-based models like RETRO actually exploit evaluation dataset leakage or not?

- Filtered bits-per-bytes.
- Yes, retrieval-based models like RETRO do exploit evaluation dataset leakage, as indicated in the figure below.
Sampling Results: “Beavers are interesting animals”

Beavers are interesting animals that live near rivers. They build dams to create ponds. Frogs are amphibians, so they can live in both land and water. They have great camouflage to hide from predators. The Golden Retriever, or Golden, is a very big dog. They are very strong, and have a thick coat to help them live in harsh environments.

Beavers build their lodges in ponds they have created in wooded areas. Like many things in nature, there is a connection between creatures in the wild. Beaver ponds cause trees to drown, but the dead trees attract the great blue heron, which often return year after year. Over time, a beaver pond can attract more than 50 nests in a colony, called a rookery. An example of this can be found in the large pond off Bradford Road at Carter Fields near the Boxford insect. Chris Leahy, an expert with the Massachusetts Audubon Society who wrote, then they mean that you are very busy. Beavers swim easily in streams, picking up rocks and sticks to build their dams. They gnaw at trees with their big front teeth to cut them down. Then they use parts of the trees to build their lodges. Beavers are clever builders. They know exactly what they need to build their beaver dams. They use mud from the stream to make their dams stay together. They use their tails to pat down the mud. Beavers put a snug room at the top of their dams for their babies. They store their food underwater. Beavers eat the bark from the ar-like tail, and two protruding teeth that are strong enough to gnaw down trees. The beaver uses trees, branches, and mud to build dams across rivers and streams. These dams create a deep pool of water in which the beaver builds its home. A beaver home is called a lodge. A baby beaver or "kit" remains in the family lodge until the age of two. Beaver fur, known as pelts, was once highly popular as a trim for hats and coats. How might the popularity of beaver fur contribute to the colonization of New York? www.WW

<table>
<thead>
<tr>
<th>Prompt and sample of RETRO[OFF]</th>
<th>Prompt and sample of RETRO[ON] colored by LCP with RET(Cu−1)</th>
<th>[N^2_{0, 1}, F^1_{1}] colored by LCP with Cu+1</th>
<th>[N^2_{0, 1}, F^2_{2}] colored by LCP with Cu+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCP = 0, 1, 2, 3, 4, &gt;5</td>
<td>LCP = 0, 1, 2, 3, 4, &gt;5</td>
<td>LCP = 0, 1, 2, 3, 4, &gt;5</td>
<td></td>
</tr>
</tbody>
</table>
Thanks for listening!
Q3: Do you think that retrieval-based LMs can work similarly as standard dense LLMs in terms of downstream applications (e.g., prompting, fine-tuning)?

What are key challenges of scaling up retrieval-based LMs?