Section 4: Retrieval-based LMs: Training
Retrieval-based LMs

Datastore

Query

Index

LM

Input

Output
Training retrieval-based LMs

Datastore

Input

Query

Index

LM

Output

Back-propagate
Why is training challenging?

Datastore

Query

Index

LM

Input

Output

Back-propagate

Training LMs can be very expensive!
Why is training challenging?

Datastore

Query

Index

LM

Input

Output

⚠ Too large! Expensive to update index during training!

⚠ Training LMs can be very expensive!
Challenges of updating retrieval models

Datastore

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \ldots \]

Encoder

Index

We may encode a lot of (>100M) text chunks using the encoder!
Challenges of updating retrieval models

During training, we will update the encoder
Challenges of updating retrieval models

Datastore

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \ldots \]

Encoder

Index

Re-indexing will be very expensive!
Why is training challenging?

- Too large! Expensive to update index during training!

- Training LMs can be very expensive!
Training methods for retrieval-based LMs

- Independent training
- Sequential training
- Joint training w/ asynchronous index update
- Joint training w/ in-batch approximation
Training methods for retrieval-based LMs

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

Retrieval models and language models are trained independently

- Training language models
  
  Input → **LM** → Output

- Training retrieval models
  
  **Retriever**

  Datastore → **Retriever** → Chunks/tokens

  Query
Independent training

Retrieval models and language models are trained independently.

- Training language models
  
  \[\text{Input} \rightarrow \text{LM} \rightarrow \text{Output}\]

- Training retrieval models

  \[\text{Datastore} \rightarrow \text{Retriever} \rightarrow \text{Chunks/tokens}\]

  Query
Training language models

Input $\rightarrow$ LM $\rightarrow$ Output

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

Back-propagate
Training language models

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

Input $\rightarrow$ LM $\rightarrow$ Output

Back-propagate

GPT
PaLM
LLaMA
GPT-J

......
Independent training

Retrieval models and language models are trained independently

- Training language models

    Input → LM → Output

- Training retrieval models

    Datastore → Retriever → Chunks/tokens

    Query
Sparse retrieval models: TF-IDF / BM25

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

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

[0, 0, 0.4, 0, 0.8, 0.7, …]

Lexical overlap

[0, 1.2, 0.4, 0, 0.8, 0, …]

Text chunks

Sparse vectors

No training needed!

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

Inner Product Similarity

Dense vectors

Encoder

$q$

Query

Encoder

Text chunks

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

Encoder → Corpus → Encoder → Index

Query → Encoder → Index

Maximum inner-product search
Dense retrievers: Inference

How to train dense retrieval models?
Training dense retrieval models: DPR

Inner Product Similarity

Encoder

Query $q$

Encoder

Text chunks
Training dense retrieval models: DPR

\[ L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-) = - \log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))} \]
Training dense retrieval models: DPR

Encoder

Query

Text chunks

Inner Product Similarity

$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-) = - \log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))}$

Contrastive learning
Training dense retrieval models: DPR

Inner Product Similarity

\[ L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-) = -\log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))} \]

Contrastive learning
Training dense retrieval models: DPR

Inner Product Similarity

$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-) = -\log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))}$$

Positive passage
Training dense retrieval models: DPR

Encoder

Query

Text chunks

Inner Product Similarity

Encoder

L(q, [p⁺, p⁻₁, p⁻₂, ..., p⁻ₙ])

Positive passage

Negative passages
Too expensive to consider all negatives!

exp(sim(q, p⁺))

exp(sim(q, p⁺)) + ∑ᵢ₌₁ⁿ exp(sim(q, p⁻ᵢ))

= − log

q
Training with “in-batch” negatives

\[
L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-) = -\log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))}
\]
Training with “in-batch” negatives

\[ L(q, p^+, p_1^−, p_2^−, \ldots, p_n^−) = - \log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))} \]

Training batch

- Who founded Apple?
- What is the name of Spain’s most famous soccer team?
- Who was the first ministry head of state in Nigeria?
- 12-year-old Spanish football club Real Madrid is undoubtedly the best football club Spain has ever…
- Thomas Umunnakwe Aguiyi-Ironsi seized power during the ensuing chaos after the 15 January …
Training with “in-batch” negatives

\[
L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-) = - \log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))}
\]

Back-propagation to all in-batch negatives!
In 1997, Apple merged with NeXT, and Steve Jobs became CEO of his former company. He became the saviour of his company and was largely responsible ...
Retrieval-in-context in LM (Ram et al. 2023)

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

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

- Retrieval Model

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

- LM

48 in the 2026 tournament.

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

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

World Cup 2022 was the last with 32 teams, before the increase to 48 in the 2026 tournament.

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

\[ \text{Retrieval Model} \quad \text{BM25, DPR, Contriever, …} \]

\[ \text{LM} \quad \text{GPT, OPT, LLaMA, …} \]
Retrieval-in-context in LM

Better retrieval model
Better base LMs
Better retrieval-based LMs
Retrieval-in-context in LM

Better retrieval model
Better base LMs

Each component can be improved separately

Better retrieval-based LMs
Inference

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

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

Re-use the LM encoder. No training needed!

\[
P_{\text{kNN}}(y|x) \propto \sum_{(k,v) \in \mathcal{D}} [v = y] \exp(-d(\text{Enc}(k), \text{Enc}(x))
\]

\[
P_{\text{kNN-LM}}(y|x) = (1 - \lambda)P_{\text{LM}}(y|x) + \lambda P_{\text{kNN}}(y|x)
\]
kNN-LM (Khandelwal et al. 2020)

**Inference**

Re-use the LM encoder. No training needed!

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

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

**Training**

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

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

👍 Work with off-the-shelf models (no extra training required)

👍 Each part can be improved independently
Independent training

👍 Work with off-the-shelf models (no extra training required)

👍 Each part can be improved independently

👎 LMs are not trained to leverage retrieval

👎 Retrieval models are not optimized for LM tasks/domains
Training methods for retrieval-based LMs

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

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

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

- One component is first trained independently and then fixed

- The other component is trained with an objective that depends on the first one
Sequential training

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

- **Retrieval models** are first trained independently and then fixed
- **Language models** are trained with an objective that depends on the retrieval
RETRO (Borgeaud et al. 2021)

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

\[ x_1 \quad x_2 \quad x_3 \]
RETRO (Borgeaud et al. 2021)

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

$x_1 \quad x_2 \quad x_3$

$x_1$ Retrieval Encoder $\Rightarrow$ Index $p_1 \ldots p_1^k$

$x_2$ $\Rightarrow$ $p_2 \ldots p_2^k$

$x_3$ $\Rightarrow$ $p_3 \ldots p_3^k$
RETRO (Borgeaud et al. 2021)

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

$x_1 \quad x_2 \quad x_3$

$(k \text{ chunks of text per split})$

$x_1 \quad \text{Retrieval Encoder} \quad p_1 \ldots p_k$

$x_2 \quad \text{Index} \quad p_2 \ldots p_k$

$x_3 \quad \text{LM Encoder} \quad p_3 \ldots p_k$

$E_1 \quad E_2 \quad E_3$
x = World Cup 2022 was the last with 32 teams, before the increase to

(k chunks of text per split)
RETRO (Borgeaud et al. 2021)

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

Very large!
600B tokens during training;
1.8T tokens during inference.

$x_1 \rightarrow \text{Index} \rightarrow x_2 \rightarrow x_3$

$(k \text{ chunks of text per split})$

$x_1$ Retrieval Encoder $x_2$ Index $x_3$ LM Encoder

$E_1 \rightarrow E_2 \rightarrow E_3$

$x_1 \rightarrow \text{EMB} \rightarrow \text{ATTN} \rightarrow \text{CCA} \rightarrow \text{FFW} \rightarrow \text{HEAD}$

$\text{RETRO blocks (}xL\text{)}$
RETRO: Training
RETRO: Training

\[ x_1 \xrightarrow{\text{Retrieval Encoder}} \text{Index} \xrightarrow{\text{LM Encoder}} E_1 \]

\[ x_2 \xrightarrow{\text{Index}} p_1^1 \ldots p_k^1 \]

\[ x_3 \xrightarrow{\text{Index}} p_1^2 \ldots p_k^2 \]

\[ p_1^3 \ldots p_k^3 \]

\[ E_2 \]

\[ E_3 \]

---

\[ x_1 \xrightarrow{\text{Emb}} \text{ATTN} \xrightarrow{\text{CCA}} \text{FFW} \xrightarrow{\text{HEAD}} \]

\[ \text{RETRO blocks (xL)} \]

Back-propagate
Updaiting an index with 600B is extremely \textbf{expensive}!!
RETRO: Training

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

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

- **Language models** are first trained independently and then fixed

- **Retrieval models** are trained/fine-tuned with supervisions from LMs
REPLUG (Shi et al. 2023)
REPLUG (Shi et al. 2023)

Shi et al., 2023. “REPLUG: Retrieval-Augmented Black-Box Language Models”
Shi et al., 2023. “REPLUG: Retrieval-Augmented Black-Box Language Models”
REPLUG LSR (LM-Supervised Retrieval)
REPLUG LSR (LM-Supervised Retrieval)

2. Computing LM likelihood $Q(d_i \mid x) \propto P_{LM(\text{apple} \mid d_i, x) / \beta}$
REPLUG LSR (LM-Supervised Retrieval)

1. Computing Retriever Likelihood $P_R(d_i | x)$

2. Computing LM likelihood $Q(d_i | x) \propto P_{LM}(\text{apple} | d_i, x) / \beta$

3. KL Divergence ($P||Q$)
REPLUG LSR (LM-Supervised Retrieval)

1. **Computing Retriever Likelihood** $P_R(d_i | x)$

2. **Computing LM likelihood** $Q(d_i | x) \propto P_{LM}(apple | d_i, x) / \beta$

3. **KL Divergence** (P||Q)

Updating retrieval encoder → Retrieval Index becomes "stale"

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

"Asynchronous update"
# REPLUG results

Bits per byte (BPB): The lower the better

<table>
<thead>
<tr>
<th>Model</th>
<th># Parameters</th>
<th>Original</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>GPT-2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>117M</td>
<td>1.33</td>
</tr>
<tr>
<td>Medium</td>
<td>345M</td>
<td>1.20</td>
</tr>
<tr>
<td>Large</td>
<td>774M</td>
<td>1.19</td>
</tr>
<tr>
<td>XL</td>
<td>1.5B</td>
<td>1.16</td>
</tr>
<tr>
<td><strong>GPT-3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(black-box)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ada</td>
<td>350M</td>
<td>1.05</td>
</tr>
<tr>
<td>Babbage</td>
<td>1.3B</td>
<td>0.95</td>
</tr>
<tr>
<td>Curie</td>
<td>6.7B</td>
<td>0.88</td>
</tr>
<tr>
<td>Davinci</td>
<td>175B</td>
<td>0.80</td>
</tr>
</tbody>
</table>
# REPLUG Results

With Contriever, “independent training”

<table>
<thead>
<tr>
<th>Model</th>
<th># Parameters</th>
<th>Original</th>
<th>+ REPLUG</th>
<th>Gain %</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPT-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Small</td>
<td>117M</td>
<td>1.33</td>
<td>1.26</td>
<td>5.3</td>
</tr>
<tr>
<td>Medium</td>
<td>345M</td>
<td>1.20</td>
<td>1.14</td>
<td>5.0</td>
</tr>
<tr>
<td>Large</td>
<td>774M</td>
<td>1.19</td>
<td>1.15</td>
<td>3.4</td>
</tr>
<tr>
<td>XL</td>
<td>1.5B</td>
<td>1.16</td>
<td>1.09</td>
<td>6.0</td>
</tr>
<tr>
<td>GPT-3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ada</td>
<td>350M</td>
<td>1.05</td>
<td>0.98</td>
<td>6.7</td>
</tr>
<tr>
<td>Babbage</td>
<td>1.3B</td>
<td>0.95</td>
<td>0.90</td>
<td>5.3</td>
</tr>
<tr>
<td>Curie</td>
<td>6.7B</td>
<td>0.88</td>
<td>0.85</td>
<td>3.4</td>
</tr>
<tr>
<td>Davinci</td>
<td>175B</td>
<td>0.80</td>
<td>0.77</td>
<td>3.8</td>
</tr>
</tbody>
</table>
# REPLUG results

## Fine-tuning Contriever with LM-supervised training

“Sequential training”

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

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

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

Let’s jointly train retrieval models and LMs!
We’ll be back at 4:00pm!
Section 4:
Retrieval-based LMs: Training (cont’d)
Why is training challenging?

- Too large! Expensive to update index during training!
- Training LMs can be very expensive!
Training methods for retrieval-based LMs

- **Independent** training
- **Sequential** training
- Joint training w/ asynchronous index update
- Joint training w/ in-batch approximation
Training methods for retrieval-based LMs

- Independent training
- Sequential training
- Joint training w/ asynchronous index update
- Joint training w/ in-batch approximation
Challenges of updating retrieval models

During training, we will update the encoder
Challenges of updating retrieval models

Datastore

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \ldots \]

Encoder

Index

Re-indexing will be very expensive!
Training methods for retrieval-based LMs

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

- **Retrieval models** and **language models** are trained jointly

- Allow the index to be “**stale**”; rebuild the retrieval index every $T$ steps
Asynchronous index update

Datastore

\[ x_1 \quad x_2 \quad x_3 \quad \ldots \]

Encoder

Index
Asynchronous index update
Asynchronous index update

Datastore

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]
\[ \ldots \]

Updated Encoder

Updated Index

\( T \) updates

Refresh
**REALM** (Guu et al. 2020)

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

\[ q (=x) \]

Index

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

\[ P(z \mid x) \]

\[ P(y \mid x, z) \]

The pyramidion on top ... the pyramid.

... The [MASK] at the top of the pyramid.

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

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

$q (=x)$

Index

$\mathcal{I}_\theta$: top-K retrieved chunks

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

$P_\theta(z \mid x)$

The pyramidion on top … the pyramid.

…

The [MASK] at the top of the pyramid.

LM

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

pyramid
REALM: Training

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

$q (=x)$

$\mathcal{I}_\theta :$ top-K retrieved chunks

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

$P_\theta(z \mid x)$

The pyramidion on top … the pyramid.

…

The [MASK] at the top of the pyramid.

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

Back-propagation
REALM: Training

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

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

\( q (=x) \)

The pyramidion on top … the pyramid.

\( \mathcal{Z}_\theta \): top-K retrieved chunks

The [MASK] at the top of the pyramid.

Up-to-date parameters

Stale index; Update every \( T \) steps

\( \mathcal{Z}_\theta \): top-K retrieved chunks

Objective:

\[
\text{maximize } \sum_{z \in \mathcal{Z}_\theta} P_{\theta}(z \mid q)P_{\theta}(y \mid q, z)
\]

Index

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

LM

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

pyramid
REALM: Index update rate

How often should we update the retrieval index?
- Frequency too high: expensive
- Frequency too slow: out-dated
REALM: Index update rate

How often should we update the retrieval index?
- Frequency too high: expensive
- Frequency too slow: out-dated

REALM: updating the index every 500 training steps
REALM: Index update rate

How often should we update the retrieval index?
- Frequency too high: expensive
- Frequency too slow: out-dated

REALM: updating the index every 500 training steps

EM score on NQ

- REALM: 38.20
- 30x slower update: 28.70
Atlas (Izacard et al. 2022)

Masked Language Modelling: 
Bermuda Triangle is in the <MASK> of the Atlantic Ocean.

Pretraining

Few-shot

Fact checking: 
Bermuda Triangle is in the western part of the Himalayas.

Question answering: 
Where is the Bermuda Triangle?

The Bermuda Triangle is an urban legend focused on a loosely-defined region in the western part of the North Atlantic Ocean.

Western part of the North Atlantic Ocean

False

Atlas (Izacard et al. 2022)

Retrieval-based encoder-decoder model

Jobs is CEO of __

Jobs was raised ...

Jobs is CEO of ...

Steve Jobs passed ...

Jobs is CEO of ...

Jobs cofounded ...

Jobs is CEO of ...

Jobs is CEO of __

Index

Jobs was raised … Jobs is CEO of
Steve Jobs passed … Jobs is CEO of
Jobs cofounded … Jobs is CEO of

Encoder

Decoder

Retrieve docs & Process each doc independently using “Fusion-in-Decoder”

Atlas (Izacard et al. 2022)

Adapted to a lot of downstream tasks! (Section 5)
Atlas: Retriever training

**Perplexity Distillation**

Retrieve the text that can help LM encoders improve perplexity

\[
P_{\text{retr}}(z \mid q) = \frac{\exp(s(z, q))}{\sum_{k=1}^{K} \exp(s(z_k, q))}
\]

How likely each document is retrieved

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

How much each document improves the ppl
Atlas: Retriever training

Similarity based on retrieval encoder

\[ P_{\text{retr}}(z \mid q) = \frac{\exp(s(z, q))}{\sum_{k=1}^{K} \exp(s(z_k, q))} \]

How likely each document is retrieved

KL Divergence

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

How much each document improves the ppl

Prob of the gold labels if augmenting this text chunk

Perplexity Distillation
Atlas: Asynchronous index update

Update the index every T steps

30% overhead for asynchronous update on Wikipedia
Atlas: Asynchronous index update

Update the index every $T$ steps

30% overhead for asynchronous update on Wikipedia

How can we get rid of this?
Training methods for retrieval-based LMs

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

- Retrieval models and language models are trained jointly
- Use “in-batch index” instead of full index
In-batch approximation

Full corpus

Encoder

Full index
In-batch approximation

Full corpus → Encoder → Full index

Re-indexing will be very expensive!
In-batch approximation

Full corpus

Encoder

Full index

Re-indexing will be very expensive!

Training batch

Encoder

In-batch index

Computed on the fly for each batch!

~10K

>100M
TRIME: Training with in-batch memory \cite{zhong2022training}

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

Similar to kNN-LM

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</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</td>
<td>torch</td>
</tr>
<tr>
<td>A group of infections … one of the</td>
<td>torch</td>
</tr>
</tbody>
</table>
TRIME: Training with in-batch memory (Zhong et al. 2022)

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

Similar to kNN-LM

```
<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</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>
```

Inference

$$P(y \mid x) \propto \exp(E^T f(x)) + \sum_{(k,v) \in \mathcal{D}} \mathbb{1}[v = y] \exp(-d(Enc(k), Enc(x)))$$

- Output embedding (same as standard LMs)
- Datastore (very large!)
TRIME: Training with in-batch memory (Zhong et al. 2022)

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

- context (chunk)
- next token

Inference

$$P(y \mid x) \propto \exp(E^T f(x)) + \sum_{(k,v) \in \mathcal{D}} \mathbb{1}[v = y] \exp(-d(\text{Enc}(k), \text{Enc}(x)))$$

1. Aligning the output representations with static embeddings
2. Aligning input chunks with all the chunks in datastore that share the same next token

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

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

\[ P(y | x) \propto \exp(E^\top f(x)) + \sum_{(k,v) \in \mathcal{D}} \mathbb{1}[v = y] \exp(-d(\text{Enc}(k), \text{Enc}(x))) \]

Jobs become CEO of **Apple** → **He moves to Apple**

Positive chunks → pull together

Jobs become CEO of **Apple** → **She works at Microsoft**

Negative chunks → push away
TRIME: Training with in-batch memory (Zhong et al. 2022)

\[ P(y \mid x) \propto \exp(E^\top f(x)) + \sum_{(k,v) \in \mathcal{D}} \mathbb{1}[v = y] \exp(-d(\text{Enc}(k), \text{Enc}(x))) \]

Jobs become CEO of Apple \(\leftrightarrow\) He moves to Apple

**Positive** chunks \(\rightarrow\) pull together

Jobs become CEO of Apple \(\leftrightarrow\) She works at Microsoft

**Negative** chunks \(\rightarrow\) push away
TRIME: Training

Key idea: build a temporary index from **same training batch** on the fly

\[ P(y \mid x) \propto \exp(E^T f(x)) + \sum_{(k,v) \in \mathcal{D}_{\text{train}}} \mathbb{1}[v = y](-d(Enc(k), Enc(x))) \]

**In-batch approximation**
(built from in-batch examples on the fly)
TRIME: Training

Key idea: build a temporary index from same training batch on the fly

\[
P(y \mid x) \propto \exp(E^T f(x)) + \sum_{(k,v) \in \mathcal{D}_{train}} \mathbb{1}[v = y](-d(\text{Enc}(k), \text{Enc}(x)))
\]

In-batch approximation
(built from in-batch examples on the fly)

We can back-propagate to all the representations in datastore \(\mathcal{D}_{train} \!\)
TRIME: Full index vs. in-batch index

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>To check the version of PyTorch, you can use</td>
<td>torch</td>
</tr>
<tr>
<td>Item delivered broken. Very</td>
<td>cheap</td>
</tr>
<tr>
<td>He moves to</td>
<td>Apple</td>
</tr>
<tr>
<td>Apple merged with NeXT, and</td>
<td>CEO</td>
</tr>
<tr>
<td>Jobs became</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Full corpus

Full index (used during inference)
TRIME: Full index vs. in-batch index

Full corpus

Full index (used during inference)

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

Values
- torch
- cheap
- Apple
- CEO

Compute on the fly!

Training batch

In-batch index (used during training)

Keys
- Apple merged with NeXT, and ...
- VS Code was developed by Microsoft for Windows in 2015 ...
- He moves to Apple ...

Values
- Jobs became CEO
- ... ...
- He moves to Apple
- VS Code was developed by
How to batch training data — so we can have good in-batch examples?

Apple merged with NeXT, and …

VS Code was developed by Microsoft for Windows in 2015 …

He moves to Apple …

…

Training batch

In-batch index (used during training)

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple merged with NeXT, and Jobs became CEO</td>
<td>…</td>
</tr>
<tr>
<td>He moves to Apple</td>
<td>VS Code was developed by Microsoft for Windows in 2015</td>
</tr>
</tbody>
</table>

Compute on the fly!
TRIME: Data batching strategy

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

Use BM25 scores to find similar text chunks
TRIME: Results

Perplexity: The lower the better

Perplexity on Wikitext-103

- Transformer: 18.65
- kNN-LM: 16.37
- TrimeLM: 15.41

Perplexity: The lower the better
NPM: Nonparametric masked LMs (Min et al. 2023)

Reference Corpus

Item delivered **broken**. Very cheaply made and does not even function. 10/10, would buy this cheap **awesome** gaming headset again.

The Church of Saint Demetrius, or Hagios Demetrios, is the main sanctuary dedicated to Saint Demetrius, the patron saint of **Thessaloniki**.

The Banpo Bridge (Korean: 반포대교) is a major bridge in downtown Seoul.

cheaper than an iPod. It was \(<\text{mask}\>** awesome**.

cheap construction. It was \(<\text{mask}\>** broken**.

Hagios Demetrius is located in \(<\text{mask}\>** Thessaloniki**.

The Korean translation of Banpo Bridge is \(<\text{mask}\>**.

Encoder (12 tokens)
NPM: Nonparametric masked LMs (Min et al. 2023)

1. **masked** language model pretrained on >1B tokens
NPM: Nonparametric masked LMs (Min et al. 2023)

1. Masked language model pretrained on >1B tokens

2. Each mask corresponds to a phrase (instead of a token)
1. Masked language model pertained on >1B tokens

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

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

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

Key challenges
1. How to approximate the full retrieval index during training
2. How to get training signals (positive/negatives) from the index approximation

In-batch approximation with same-doc batching
NPM: Training

1. Sample sequences from the same document

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

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

For simplicity, we assume 2 sequences in a batch
NPM: Training

2. Identify co-occurring spans

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

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

For simplicity, we assume 2 sequences in a batch
NPM: Training

3. One is “positive” for the other

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

positive

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

For simplicity, we assume 2 sequences in a batch
In the 2010 NFL season, __________ made history by making it into the playoffs despite having a 7-9 record.

Positive

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

Negatives

For simplicity, we assume 2 sequences in a batch
Beyond lexical clues?

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

Positives: *co-occurring* tokens/spans

*Can we do more than that?*
RPT: Retrieval-pretrained transformer
(Rubin and Berant 2023)

Reference score  \( P(\text{“Apple”} \mid \text{“Jobs become CEO of”, “NeXT merged with …”}) \)

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

Reference score

$P(\text{"Apple" | "Jobs become CEO of", "NeXT merged with ..."})$

Reference chunk

$P(\text{"Apple" | "Jobs become CEO of", "He joined his former ..."}) >$ Reference score

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

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

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

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

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

End-to-end trained — each component is optimized

Good performance

Training is more complicated (async update, overhead, data batching, etc)

Train-test discrepancy still remains
# Summary

<table>
<thead>
<tr>
<th>Training method</th>
<th>🌟 Easy to implement: off-the-shelf models</th>
<th>🌟 Models are not end-to-end trained — suboptimal performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent training</td>
<td>🌟 Easy to improve: sub-module can be separately improved</td>
<td></td>
</tr>
<tr>
<td>(Ram et al 2023; Khandelwal et al 2020)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequential training</td>
<td>🌟 Models are not end-to-end trained — suboptimal performance</td>
<td></td>
</tr>
<tr>
<td>(Borgeaud et al 2021; Shi et al 2023)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joint training: async update</td>
<td>🌟 End-to-end trained — very good performance!</td>
<td></td>
</tr>
<tr>
<td>(Guu et al 2020; Izacard et al 2022)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Joint training: in-batch approx</td>
<td>🌟 End-to-end trained — very good performance!</td>
<td></td>
</tr>
<tr>
<td>(Zhong et al 2022; Min et al 2023; Rubin and Berant 2023)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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