Lectures on Natural Language Processing

10. Retrieval-Augmented Models

Karl Stratos

Review: Prompting Large Language Models (LLMs)

Example: GPT-4 (OpenAI, 2023)

User prompt (appended to a general prompt). Explain the plot of Cinderella in a sentence where each word has to begin with the next letter in the alphabet from A to Z, without repeating any letters.

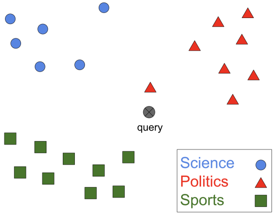
GPT-4 generation. A beautiful Cinderella, dwelling eagerly, finally gains happiness; inspiring jealous kin, love magically nurtures opulent prince; quietly rescues, slipper triumphs, uniting very wondrously, xenial youth zealously.

LLMs will likely continue to improve in "steerability" and knowledge through model/data scaling, incorporating human feedback

- **BUT** the same limitations persist, including false claims, silly reasoning errors, biases
- Promising solution: grounding the model on knowledge bases

Review: Information Retrieval (IR)

- Want "similar" documents closer to each other than "unsimilar" ones under some notion of distance/similarity



- Classical IR techniques: TFIDF, BM25
  - Sparse term-based representation, a term downweighted by how often it appears in documents
  - Representation is corpus-specific! Need to "train", i.e., build an **index** (data structure to store the corpus that allows for efficient search)
  - Fast and effective for general semantic search, but not trainable for specific types of search

Knowledge Bases (KBs)

- **KB**: dataset storing complex information
  - In general, can be highly structured (e.g., tabular information)
  - For retrieval, can be thought of as a set $\mathcal{Y}$ of items to retrieve
- The KB depends on the task:

$$\mathcal{Y}_{\mathrm{ER}} = \{\text{6 million Wikipedia articles}\} \qquad \text{(entity retrieval)}$$
$$\mathcal{Y}_{\mathrm{QA}} = \{\text{20 million passages in Wikipedia}\} \qquad \text{(open-domain QA)}$$
$$\mathcal{Y}_{\mathrm{Web}} = \{\text{billions of passages on the web}\} \qquad \text{(retrieval augmentation)}$$

- Sources of KB
  - **Wikipedia**: Crowdsourced, > 270 languages, millions of articles per language, natural annotations (hyperlinks, tables)
  - Specialized QA websites (e.g., Stack Overflow, Reddit, Quora)
  - Domain-specific resources (e.g., PubMed—32 million citations for biomedical literature)
  - **The web** (subsuming Wikipedia, GitHub, etc.), treated as a gigantic set of passages

Text Representation of KB Elements

- No matter what KB $\mathcal{Y}$ we have, we will represent each element $y \in \mathcal{Y}$ as a piece of text.
- E.g., in entity retrieval, Wikipedia entity represented by the first T words in the article

$=$ Barack Hussein Obama II is an American former politician who served as the 44th president of the United States from 2009 to 2017... $\in \mathcal{V}^T$

- If KB is a set of length-T passages, each element is already text.

proliferation of minor wartime regulations. Parts of the scripts were rewritten in the hours before the broadcast, to ensure topicality. ITMA was an important contributor to British morale during the war... $\in \mathcal{V}^T$

- Can handle any item in KB by "reading" its description

Retrieval = Text Matching

- The search problem: Given query $x \in \mathcal{X}$, find K highest scoring items in $\mathcal{Y}$

$$(y_1 \ldots y_K) = K\text{-argmax}_{y \in \underbrace{\mathcal{Y}}_{\text{huge}}} \underbrace{\textbf{score}_\theta(x, y)}_{\text{"relevance" of texts } x \text{ and } y}$$

- "Retriever": a mapping from a text-pair to a relevance score

$$\textbf{score}_\theta : \mathcal{V}^+ \times \mathcal{V}^+ \rightarrow \mathbb{R}$$

(e.g., BM25 defines **score**(x, y) as a function of $x \cap y$.)

- Important questions
  1. How should we parameterize **score**$_\theta$ so that search is efficient?
  2. How can we *learn* **score**$_\theta$?
     - Assign high score to correct pair, low score to incorrect pair

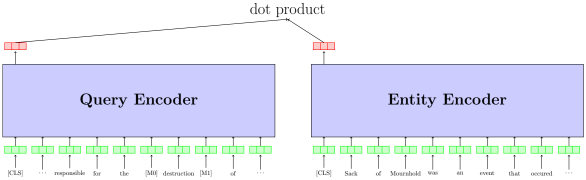
Dual Encoders

- Simplest form of parametric retriever, has two learnable modules

$$\textbf{enc}^X_\phi : \mathcal{V}^+ \to \mathbb{R}^d \qquad \text{(for queries)}$$
$$\textbf{enc}^Y_\psi : \mathcal{V}^+ \to \mathbb{R}^d \qquad \text{(for KB elements)}$$

  (e.g., transformer encoders with some pooling at the top)
- If $\textbf{enc}^X_\phi = \textbf{enc}^Y_\psi$ called "siamese" network.
- Defines the similarity between two texts x, y by (here $\theta = \{\phi, \psi\}$)

$$\textbf{score}_\theta(x, y) = \textbf{enc}^X_\phi(x) \cdot \textbf{enc}^Y_\psi(y)$$



- Crucially, similarity search with dense vectors can be done very efficiently

Nearest Neighbor Search in Vector Space

- With a dual encoder, we can embed all KB items *offline* (using the target encoder)

$$\textbf{precomputed}(y) = \textbf{enc}^{Y}_{\psi}(y) \qquad\qquad \forall y \in \mathcal{Y}$$

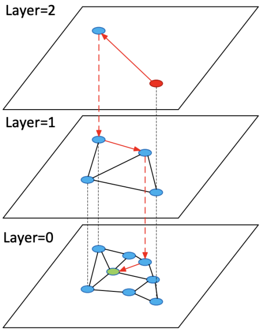
- At test time, given a query x find

$$y^* = \underset{y \in \mathcal{Y}}{\arg\min} \ \left|\left|\textbf{enc}^{X}_{\phi}(x) - \textbf{precomputed}(y)\right|\right|$$

- Naively, this will take $O(|\mathcal{Y}|)$, not tractable if KB is large
- Fortunately, ways to "index" precomputed embeddings so that *approximate* search can be done drastically faster
  - E.g., $O(\log |\mathcal{Y}|)$ expected runtime (catch: might incur a large memory overhead)
  - Important in real-world applications, not so much in academic research (i.e., just do exact search and get better results)

Example: Hierarchical Navigable Small World (HNSW)

(Malkov and Yashunin, 2018)



- Graph-based search (nodes: KB embeddings)
- Multiple layers increasingly connected (once in, always in)
- Start from the top, do local search until convergence, move down to next layer
- Idea: top-layers have long-range edges to "zoom around" really fast, bottom-layers for fine-grained search
- Stochastic graph construction, expected max depth $O(\log |\mathcal{Y}|)$

Experiments with Natural Questions comparing with exact search (21 million passages, runtime amortized over 3,600 questions)

- Runtime: 46→0.4 (ms/query), performance: 46.3→46.1 (recall@1)

Retrieval for Nearest Neighbor Classifiers

- **Nearest-neighbor classifier**: canonical application of retrieval in ML
  - "Nonparameteric": Instead of training a classifier with parameters, store all the input-label pairs in training data.
  - At test time, given a new input x, retrieve top-K most similar inputs ("neighbors") $x_1 \ldots x_K$
  - Predict its label by ensembling the labels of the neighbors $y_1 \cdots y_K$

- E.g., nearest-neighbor language models
  - Store all context-word pairs in training corpus
  - Given a new context, retrieve top-K most similar contexts and ensemble their words to predict

K-Nearest-Neighbor Language Models

- Training corpus $\mathcal{D} \equiv$ set of all context-word pairs (c', w'), e.g.,

$$(c', w') = (\text{the dog saw the}, \text{cat})$$

- Given distance $d_\theta : \mathcal{V}^+ \times \mathcal{V}^+ \rightarrow \mathbb{R}_{\geq 0}$ between texts (e.g., Euclidean distance between their embeddings), a nearest neighbor LM defines

$$p_\theta^{\text{NN}}(w|c) \propto \sum_{(c',w') \in \mathcal{D}:\ w'=w} \exp\left(-d_\theta(c, c')\right)$$

Problem: $\mathcal{D}$ too big

K-Nearest-Neighbor Language Models

- Training corpus $\mathcal{D} \equiv$ set of all context-word pairs (c', w'), e.g.,

$$(c', w') = (\text{the dog saw the}, \text{cat})$$

- Given distance $d_\theta : \mathcal{V}^+ \times \mathcal{V}^+ \rightarrow \mathbb{R}_{\geq 0}$ between texts (e.g., Euclidean distance between their embeddings), a nearest neighbor LM defines

$$p_\theta^{\mathrm{NN}}(w|c) \propto \sum_{(c',w') \in \mathcal{D}:\ w'=w} \exp\left(-d_\theta(c, c')\right)$$

  Problem: $\mathcal{D}$ too big

- Solution: approximate $\mathcal{D}$ for each context c by its K nearest neighbors

$$p_\theta^{K\text{-NN}}(w|c) \propto \sum_{(c',w') \in \left(\underset{(c',w') \in \mathcal{D}}{K\text{-argmin}}\ d_\theta(c,c')\right):\ w'=w} \exp\left(-d_\theta(c, c')\right)$$

K-NN LM: Illustration ($K = 3$)

- Current context $c =$ the dog saw the
- 3 neareast context-word pairs retrieved from $\mathcal{D}$

$$(c_1, w_1) = (\text{Ralph saw a}, \text{cat})$$
$$(c_2, w_2) = (\text{a puppy looked at the}, \text{kitten})$$
$$(c_3, w_3) = (\text{the dog chased the}, \text{cat})$$

- Conditional word distribution

$$p_\theta^{K\text{-NN}}(\text{cat}|c) = \frac{e^{-d_\theta(c,c_1)} + e^{-d_\theta(c,c_3)}}{e^{-d_\theta(c,c_1)} + e^{-d_\theta(c,c_2)} + e^{-d_\theta(c,c_3)}}$$
$$p_\theta^{K\text{-NN}}(\text{kitten}|c) = \frac{e^{-d_\theta(c,c_2)}}{e^{-d_\theta(c,c_1)} + e^{-d_\theta(c,c_2)} + e^{-d_\theta(c,c_3)}}$$
$$p_\theta^{K\text{-NN}}(w|c) = 0 \qquad\qquad \forall w \notin \{\text{cat}, \text{kitten}\}$$

K-NN LM in Practice

- Can reduce perplexity of base LM $p_\theta(w|c)$ "for free" (i.e., no training) by interpolating with a K-NN LM derived from θ

$$p_\theta^{\text{final}}(w|c) = \lambda p_\theta^{K\text{-NN}}(w|c) + (1-\lambda) p_\theta(w|c)$$

  (e.g., using the final hidden state of θ to define context distance $d_\theta(c, c') = ||\mathbf{emb}_\theta(c) - \mathbf{emb}_\theta(c')||^2$)
- Form of regularization (specifically label smoothing). λ needs to be tuned: larger $\mathcal{D}$, larger λ.
- Lots of iterations on this idea, e.g.,
  - Retrieving on the fly at test time from previously encountered context-word pairs ("continuous cache", Grave et al., 2017)
  - Retrieving from the whole training corpus (Khandelwal et al., 2020)
- Wikitext-103 perplexity drops by 2-3 points
- Downsides (common in nearest neighbor models): slow inference, large memory overhead

Trainable K-NN LM

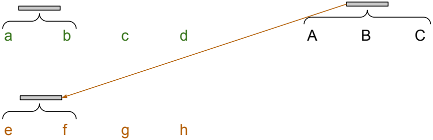
- Instead of just interpolating K-NN LM post hoc, can we *train* the LM to use the retrieved neighbors?
- Example: **RETRO** (Borgeaud et al., 2022)
  - Trains on & retrieves from MassiveText (trillions of tokens from the web, books; multilingual)
  - Frozen dual encoder based on BERT embeddings: text prefixes used as keys, but suffixes also retrieved
  - Retrieval performed every "chunk" (64 tokens)
  - LM simply conditions on all previous tokens and retrieved passages

$$p_\theta(w_t | w_{<t}, \textbf{passages}_{<t})$$

    Must be careful to stay autoregressive
  - Can yield dramatic improvement in perplexity

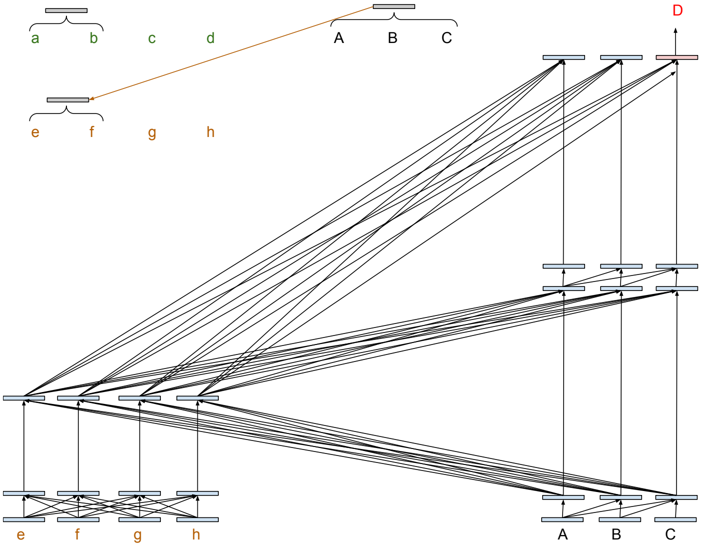
RETRO Illustration

RETRO Illustration

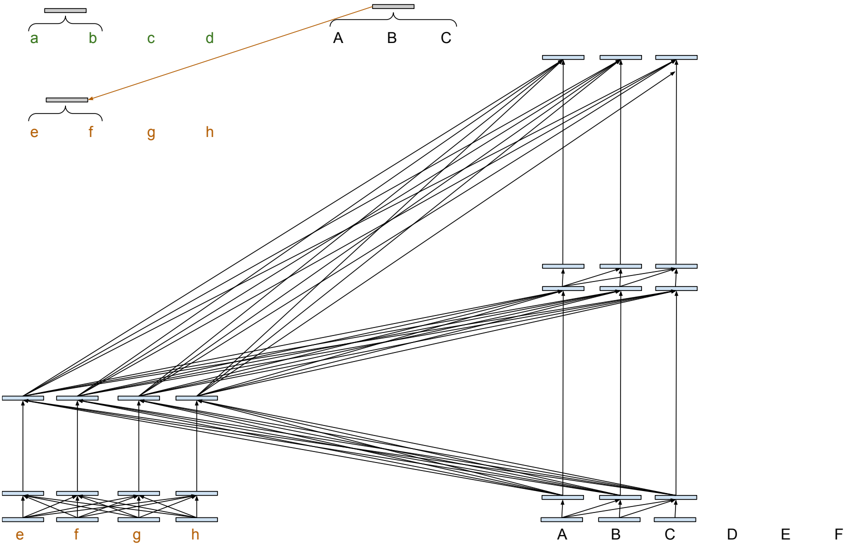


A B C

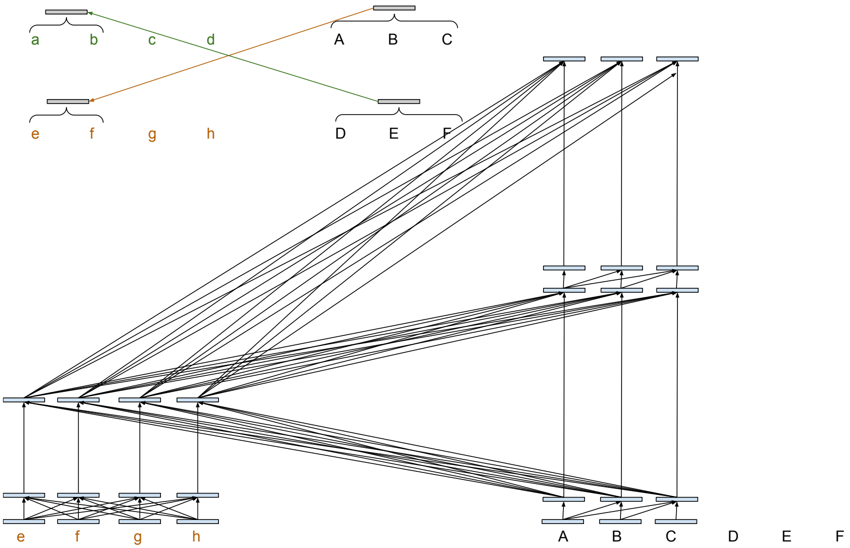
RETRO Illustration



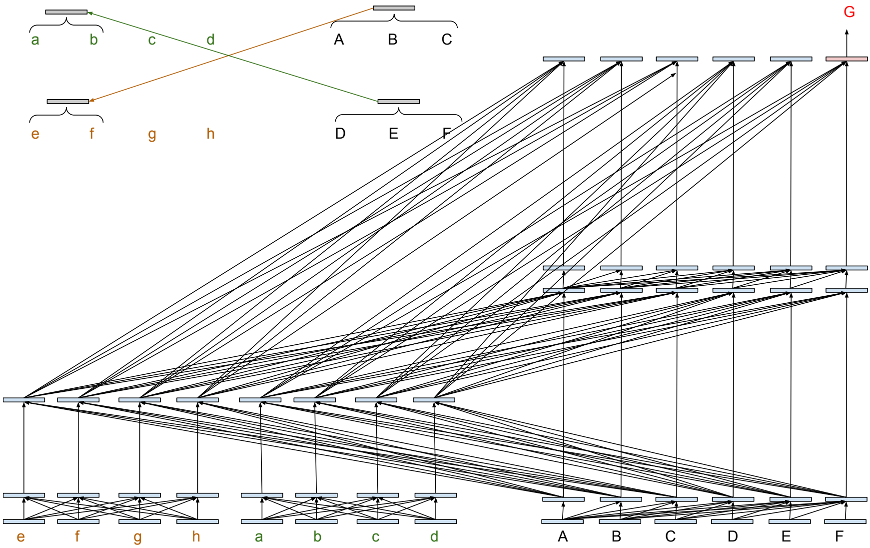
RETRO Illustration



RETRO Illustration



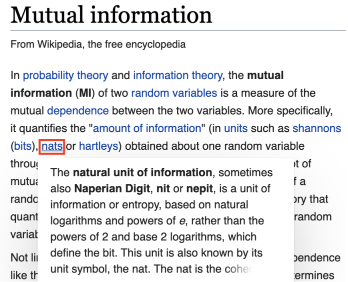
RETRO Illustration



Labeled Data for Supervised Retrieval

We assume that each query x is annotated with correct KB target $y \in \mathcal{Y}$. Can often leverage task-specific natural annotations/heuristics, e.g.,

- Entity retrieval: Crowdsourced KBs like Wikipedia already have gold mention-entity annotations in the form of hyperlinks (i.e., Wikilinks)



- Weak supervision for QA: For question x with an answer string a, can treat any KB passage $y \in \mathcal{Y}$ such that $a \in y$ as "gold".

Training Retrievers by Noise Contrastive Estimation

- Training data: N labeled queries $(x_1, y_1) \ldots (x_N, y_N) \in \mathcal{X} \times \mathcal{Y}$
- Goal: Learn $\textbf{score}_{\theta^\star}$ s.t. $\textbf{score}_{\theta^\star}(x_i, y_i) > \textbf{score}_{\theta^\star}(x_i, y)$ for $y \neq y_i$
- Naively, can train by supervised classification: maximize the usual log-likelihood

$$\theta^\star = \arg\max_{\theta} \sum_{i=1}^{N} \log\left(\frac{\exp(\textbf{score}_\theta(x_i, y_i))}{\sum_{y \in \mathcal{Y}} \exp(\textbf{score}_\theta(x_i, y))}\right)$$

Problem: $\mathcal{Y}$ too big

Training Retrievers by Noise Contrastive Estimation

- Training data: N labeled queries $(x_1, y_1) \ldots (x_N, y_N) \in \mathcal{X} \times \mathcal{Y}$
- Goal: Learn $\textbf{score}_{\theta^\star}$ s.t. $\textbf{score}_{\theta^\star}(x_i, y_i) > \textbf{score}_{\theta^\star}(x_i, y)$ for $y \neq y_i$
- Naively, can train by supervised classification: maximize the usual log-likelihood

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{N} \log \left(\frac{\exp(\textbf{score}_\theta(x_i, y_i))}{\sum_{y \in \mathcal{Y}} \exp(\textbf{score}_\theta(x_i, y))} \right)$$

  Problem: $\mathcal{Y}$ too big

- **Noise contrastive estimation (NCE)**: approximate $\mathcal{Y}$ by a small set of **negative examples** $\mathcal{N}_i \subset \{y \in \mathcal{Y} : y \neq y_i\}$

$$\theta^\star \approx \arg\max_\theta \sum_{i=1}^{N} \log \left(\frac{\exp(\textbf{score}_\theta(x_i, y_i))}{\sum_{y \in \{y_i\} \cup \mathcal{N}_i} \exp(\textbf{score}_\theta(x_i, y))} \right)$$

  - Choice of negatives critical: random, "hard" (e.g., BM25 neareast neighbors to x_i), mixture of random and hard

Hard Negatives: Idea

$x =$ Succeeded by Obama, the former president expressed his view . . . $y =$ 

Distinguishing between Bush and Watermelon is too easy.

$$\max_{\theta} \log \frac{\exp(\textbf{score}_{\theta}(x, \text{[1]}))}{\exp(\textbf{score}_{\theta}(x, \text{[2]})) + \exp(\textbf{score}_{\theta}(x, \underbrace{\text{[3]}}_{\text{random negative}}))}$$

[1]: 
[2]: 
[3]: 

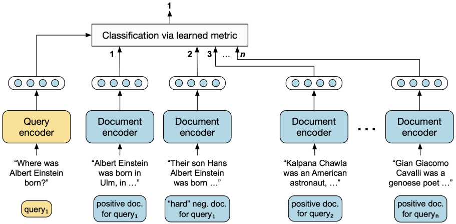
Distinguishing between Bush and Obama is harder (e.g., will fail with string matching).

$$\max_{\theta} \log \frac{\exp(\textbf{score}_{\theta}(x, \text{[1]}))}{\exp(\textbf{score}_{\theta}(x, \text{[2]})) + \exp(\textbf{score}_{\theta}(x, \underbrace{\text{[3]}}_{\text{hard negative}}))}$$

[1]: 
[2]: 
[3]: 

In-Batch Negative Sampling

- Naively, we must prepare random negatives $\mathcal{N}_i \subset \mathcal{Y}$ for every i every epoch
- **In-batch negative sampling.** Instead, use other (gold) targets y_j in the same batch as negatives (for y_i)
  - Exploits the fact that batch elements are iid
- Can easily incorporate hard negatives by batching appropriately



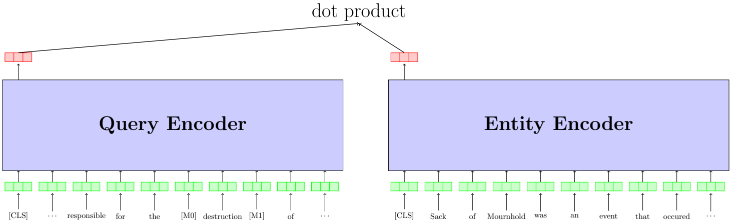
(Image credit: Maillard et al., 2021)

(1 postive, $2B - 1$ negatives (1 hard) where B batch size)

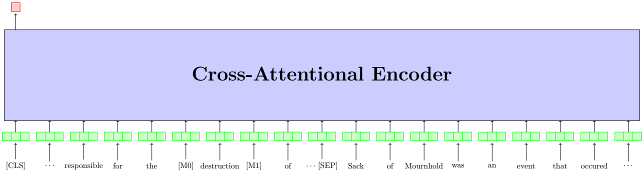
- Batch size controls # negatives: important hyperparameter

Optional: Cross-Attentional Reranking

Step 1. Train a dual encoder, do fast retrieval

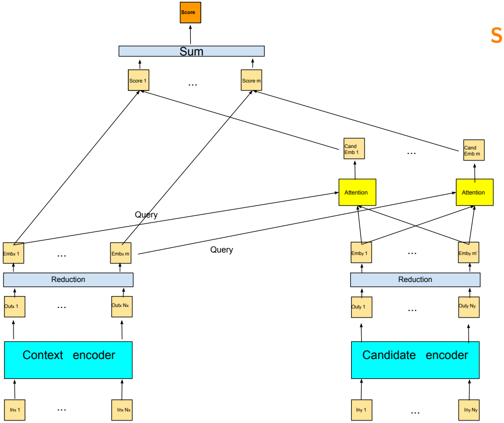


Step 2. Train a cross-attentional encoder to rerank top-K candidates (e.g., $K = 100$) from the (fixed) dual encoder from step 1.



Significant improvement in recall if (x, y) have complex interactions

Still Fast(-ish) Extensions of Dual Encoder



(Image: Zhang and Stratos, 2021)

Special cases

- **Dual encoder**: Take first embeddings
- **Poly-encoder** (Humeau et al., 2019): Learn m "code" embeddings and apply soft attention
- **Sum-of-max (ColBERT)** (Khattab and Zaharia, 2020): Take first m embeddings and apply hard attention
- **Multi-vector** (Luan et al., 2020): Same as sum-of-max but take only 1 embedding on one side

Different Settings for Training a Retriever

1. **Supervised.** Train a supervised retriever from labeled queries.
2. **Self-supervised.** Train a general-purpose retriever from raw text.
   - Essentially in-batch NCE where a "positive pair" is perturbations of the same text
     - ORQA (Lee et al., 2019): inverse cloze task
     - Contriever (Izacard et al., 2021): random spans
   - Not very performant (hard to beat BM25), but useful for initializing retrievers in end-to-end learning
   - Contrastive representation learning was more successful in vision, e.g., SimCLR (Chen et al., 2020)
3. **Indirectly supervised.** Train a task-specific retriever from annotations *for that task*
   - Sensible setting: retrieval is a means to an end
   - Mainly developed for QA: learn a passage retriever given only question-answer pairs (x, a)

Inverse Cloze Task (ICT) for Self-Supervised Retrieval

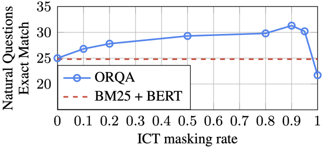
- Raw paragraph

  ... Zebras have four gaits: walk, trot, canter and gallop. They are generally slower than horses, but their great stamina helps them outrun predators. When chased, a zebra will zigzag from side to side. ...

- Positive "query-target" pair

  ... They are generally slower than horses, but their great stamina helps them outrun predators. $\mapsto$ ... Zebras have four gaits: walk, trot, canter and gallop. When chased, a zebra will zigzag from side to side. ...

- Pretrain a retriever by in-batch NCE
  - Skip masking the query sentence 10% of the time to allow lexical matching



Review: Open-Domain QA

Task: Map question x (e.g., "What does the zip in zip code stand for?") to answer a (e.g., "Zone Improvement Plan"). Evalution by exact match

- **Retriever** module. Retrieve top-K relevant passages/contexts $(y_1 \ldots y_K)$ from a KB $\mathcal{Y}$ (set of text chunks)
- **Reader** module. 2 categories:
  - **Extractive.** Encoder only: encode $(x, y_k) \in \mathcal{V}^{T+T'}$, softmax over tokens in $y_k \in \mathcal{V}^{T'}$, predict start/end span indices

    What does the zip in zip code stand for? ... The term ZIP is an acronym for <u>Zone Improvement Plan</u> ... $\mapsto (22, 24)$

  - **Generative.** Encoder-decoder: encode $(x, y_k) \in \mathcal{V}^{T+T'}$, decode $a \in \mathcal{V}^A$

    What does the zip in zip code stand for? ... The term ZIP is an acronym for <u>Zone Improvement Plan</u> ... $\mapsto$ Zone Improvement Plan

Review: Fusion-in-Decoder (FiD) (Izacard and Grave, 2020)

- Want to make the generative reader condition on as many passages as possible (i.e., large K).
- Naive solution: Concatenate question and candidate passages

$$\mathbf{dec}_\theta(\underbrace{\mathbf{enc}_\theta(x, y_1, y_2, \ldots, y_K)}_{(|x|+K|y|)\times d}) \mapsto a$$

  Not scalable: $\mathbf{enc}_\theta$ complexity quadratic in input length

- **FiD.** Encode each question-candidate pair separately, decoder conditions on the concatenation of these encodings

$$\mathbf{dec}_\theta(\underbrace{\mathbf{enc}_\theta(x, y_1), \mathbf{enc}_\theta(x, y_2), \ldots, \mathbf{enc}_\theta(x, y_K)}_{K(|x|+|y|)\times d}) \mapsto a$$

  Can be seen as QA-specific constrained self-attention

- Easy to implement (reshape the input tensor from $TK \times d$ to $T \times K \times d$), scalable in K, can be initialized from any pretrained encoder-decoder (e.g., T5)

Approaches to Open-Domain QA

- **Pipeline.** Train a supervised retriever. Then, train a reader using that retriever fixed.
  - Very strong baseline: DPR (Karpukhin et al., 2020), FiD
- **End-to-end.** Train a retriever ϕ and a reader θ jointly. 2 options

Approaches to Open-Domain QA

- **Pipeline.** Train a supervised retriever. Then, train a reader using that retriever fixed.
  - Very strong baseline: DPR (Karpukhin et al., 2020), FiD
- **End-to-end.** Train a retriever ϕ and a reader θ jointly. 2 options
  1. **Marginalized log-likelihood (MLL)** (ORQA: Lee et al., 2019)

$$\max_{\phi,\theta} \log \sum_{y \in \mathcal{Y}} \underbrace{q_\phi(y|x)}_{\propto \exp(\textbf{score}_\phi(x,y))} \times p_\theta(a|x,y)$$

Sum over $\mathcal{Y}$ approximated by $\textbf{top}_{\phi,K}(x) = (y_1 \dots y_K) \in \mathcal{Y}^K$

Approaches to Open-Domain QA

- **Pipeline.** Train a supervised retriever. Then, train a reader using that retriever fixed.
  - Very strong baseline: DPR (Karpukhin et al., 2020), FiD
- **End-to-end.** Train a retriever ϕ and a reader θ jointly. 2 options
  1. **Marginalized log-likelihood (MLL)** (ORQA: Lee et al., 2019)

$$\max_{\phi,\theta} \log \sum_{y \in \mathcal{Y}} \underbrace{q_{\phi}(y|x)}_{\propto \exp(\textbf{score}_{\phi}(x,y))} \times p_{\theta}(a|x,y)$$

     Sum over $\mathcal{Y}$ approximated by $\textbf{top}_{\phi,K}(x) = (y_1 \ldots y_K) \in \mathcal{Y}^K$

  2. **Knowledge distillation (KD)** (FiD-KD: Izacard and Grave, 2021)

$$\max_{\phi,\theta} \log p_{\theta}(a|x, \textbf{top}_{\phi,K}(x)) - \mathrm{Distil}(\underbrace{\textbf{SG}(\theta)}_{\text{teacher}}, \underbrace{\phi}_{\text{student}}, x)$$

     (**SG**: "stop-gradient", i.e., no backpropagation)

What Does the Retriever Learn in MLL?

- Marginal log-likelihood = "log-RL"

$$\underbrace{\log\left(\sum_{y\in\mathcal{Y}} q_{\phi}(y|x) \times p_{\theta}(a|x,y)\right)}_{p_{\phi,\theta}(a|x)} = \log \mathop{\mathbf{E}}_{y\sim q_{\phi}(\cdot|x)}[\underbrace{p_{\theta}(a|x,y)}_{\text{reward}}]$$

- Gradient with respect to ϕ (Gu et al., 2020)

$$\sum_{y\in\mathcal{Y}} \underbrace{\left(\frac{p_{\theta}(a|x,y)}{p_{\phi,\theta}(a|x)} - 1\right)}_{①} q_{\phi}(y|x)\nabla\textbf{score}_{\phi}(x,y)$$

- If y makes predicting a from x more likely, ① is large positive: **score**$_{\phi}(x,y)$ will increase after gradient step

Knowledge Distillation for End-to-End QA

- Idea: For question x, the *relevance* of passage y is determined by how much the reader *uses* y to generate an answer.
- Define a conditional distribution over the K passages under the reader, e.g., FiD-KD defines

$$q_{\theta}(y_k|x, y_1 \ldots y_K) \propto \exp\left(\operatorname*{Mean}_{\substack{i \in \text{positions}(y_k) \\ l \in \{1 \ldots L\} \\ h \in \{1 \ldots H\}}} \alpha_{0,i,l,h} \right)$$

$\alpha_{0,i,l,h}$: decoder attention score from target step 0 to source step i, at layer l, under head h

- Distillation loss: compute $(y_1 \ldots y_K) = \textbf{top}_{\phi,K}(x)$ and minimize

$$-\sum_{k=1}^{K} q_{\theta}(y_k|x, y_1 \ldots y_K) \log \left(\frac{\exp(\textbf{score}_{\phi}(x, y_k))}{\sum_{k'} \exp(\textbf{score}_{\phi}(x, y_{k'}))} \right)$$

FiD-KD: Iterative Training

- Initialize
  - Retriever ϕ (e.g., raw BERT, trained DPR)
  - Passage candidates $\mathrm{cands}(x) \in \mathcal{Y}^K$ (e.g., using the initial retriever, or BM25)

FiD-KD: Iterative Training

- Initialize
  - Retriever ϕ (e.g., raw BERT, trained DPR)
  - Passage candidates $\mathrm{cands}(x) \in \mathcal{Y}^K$ (e.g., using the initial retriever, or BM25)
- For T iterations:
  1. Use the current candidates to train FiD θ from scratch:

$$\min_{\theta} - \sum_{(x,a)} \log p_{\theta}(a|x, \mathrm{cands}(x))$$

FiD-KD: Iterative Training

- Initialize
  - Retriever ϕ (e.g., raw BERT, trained DPR)
  - Passage candidates $\mathrm{cands}(x) \in \mathcal{Y}^K$ (e.g., using the initial retriever, or BM25)
- For T iterations:
  1. Use the current candidates to train FiD θ from scratch:

$$\min_{\theta} \; - \sum_{(x,a)} \log p_{\theta}(a|x, \mathrm{cands}(x))$$

  2. Use the distillation loss to update retriever ϕ:

$$\min_{\phi} \; - \sum_{x} \sum_{k=1}^{K} q_{\theta}(y_k|x, \mathrm{cands}(x)) \log \left(\frac{\exp(\textbf{score}_{\phi}(x, \mathrm{cands}_k(x)))}{\sum_{k'} \exp(\textbf{score}_{\phi}(x, \mathrm{cands}_{k'}(x)))} \right)$$

  3. Update $\mathrm{cands}(x) \leftarrow \textbf{top}_{\phi,K}(x)$.

Pretraining with Latent Retrieval

- **REALM** (Gu et al., 2020) (self-supervised)
  - BERT pretraining (span-level mask), retriever trained by MLL

$$\text{[CLS]}\ \underbrace{\text{The}\ \overbrace{\text{[MASK]}}^{\textbf{pyramidion}}\ \text{at the top of the pyramid}}_{\text{masked query text}}\ \text{[SEP]}\ \underbrace{\text{The pyramidion on top allows for less material higher up the pyramid}}_{\text{retrieved passage}}$$

  - Heuristics to mask useful spans, disallow trivial retrieval
  - KB: Wiki/CC-News. Retriever initialized by ICT (pre-DPR), asynchronous index refresh with passage encoder updates
  - Pretraining followed by ORQA (extractive reader) finetuning

Pretraining with Latent Retrieval

- **REALM** (Gu et al., 2020) (self-supervised)
  - BERT pretraining (span-level mask), retriever trained by MLL

$$\text{[CLS]}\ \underbrace{\text{The}\ \overbrace{\text{[MASK]}}^{\textbf{pyramidion}}\ \text{at the top of the pyramid}}_{\text{masked query text}}\ \text{[SEP]}\ \underbrace{\text{The pyramidion on top allows for less material higher up the pyramid}}_{\text{retrieved passage}}$$

  - Heuristics to mask useful spans, disallow trivial retrieval
  - KB: Wiki/CC-News. Retriever initialized by ICT (pre-DPR), asynchronous index refresh with passage encoder updates
  - Pretraining followed by ORQA (extractive reader) finetuning

- **RAG** (Lewis et al., 2020) (multitask)
  - DPR + BART (generative reader, pre-FiD) finetuned on (x, a) pairs from Wiki-KB tasks (QA, fact verification, ...) by MLL
  - DPR passage encoder not learned, all passage embeddings precomputed and fixed

Atlas (Izacard et al., 2022)

- Self-supervised FiD with latent retrieval
- Denoising autoencoder: given masked text (A $\underbrace{\;?\;}$ C), condition on K retrieved passages $y_1 \dots y_K$ to predict the missing span (B)

$$\mathbf{dec}_\theta(\mathbf{enc}_\theta(\text{A} \underbrace{\;?\;} \text{C}, y_1), \dots, \mathbf{enc}_\theta(\text{A} \underbrace{\;?\;} \text{C}, y_K)) \mapsto \text{B}$$

  Same intuition as REALM, but is *generative* and uses *all* passages

- Retriever trained by KD using "perplexity distillation": usefulness of the k-th passage is how much it helps with "unmasking"

$$q_\theta(y_k | x, y_1 \dots y_K) \propto \exp(\log p_\theta(\text{B} | \text{A} \underbrace{\;?\;} \text{C}, y_k))$$

  Other variants of MLL and KD perform similarly, but this is simple to compute (e.g., already have query-passage encodings from FiD)

Atlas (Cont.)

- Training data: Wikipedia + Common Crawl, doubled as both training data and KB (~ 400m passages)
  - Retrieving the query passage disallowed for pretraining
- FiD initialized from T5 (11B, trained only on unlabeled data), conditioning on $K = 20$ candidates
- Retriever initialized from Contriever, different schemes to train the passage encoder efficiently
  1. Refresh passage embeddings frequently (e.g., every 1000 steps)
  2. Refresh less frequently (e.g., 2500), but rerank top-100 using current up-to-date retriever
  3. Freeze passage encoder

  Findings: freezing passage encoder is fine for few-shot training, but performance on larger datasets is better with training
- NQ: 42.4 when finetuned on 64 examples (64-shot), **60.4** when finetuned on all training data

NQ Test Performance (Exact Match)

Finetuned on all question-answer pairs in training data

	retriever	reader	end-to-end	pretraining	EM
	BM25	extract (BERT)	✗	✗	26.5
ORQA	dual enc	extract (BERT)	MLL	✗	33.3
REALM	dual enc	extract (BERT)	MLL	masked LM	40.4
DPR	dual enc	extract (BERT)	✗	✗	41.5
RAG	dual enc	BART	MLL	multitasking	44.5
FiD	dual enc	FiD (T5)	✗	✗	51.4
FiD-KD	dual enc	FiD (T5)	KD	✗	54.7
Atlas	dual enc	FiD (T5)	KD	masked LM	60.4
(closed-book)	✗	T5 (11B)	✗	✗	32.8

- Retrieval is important for QA.
- Training a QA-specific retriever is important.
- Conditioning on a large number of passages is important.
- Can go a long way with pipelines.
- Jointly training retriever+reader (esp. w/ end-to-end pretraining) can yield large gains.

NQ Test Performance (Exact Match)

Finetuned on all question-answer pairs in training data

	retriever	reader	end-to-end	pretraining	EM
	BM25	extract (BERT)	✗	✗	26.5
ORQA	dual enc	extract (BERT)	MLL	✗	33.3
REALM	dual enc	extract (BERT)	MLL	masked LM	40.4
DPR	dual enc	extract (BERT)	✗	✗	41.5
RAG	dual enc	BART	MLL	multitasking	44.5
FiD	dual enc	FiD (T5)	✗	✗	51.4
FiD-KD	dual enc	FiD (T5)	KD	✗	54.7
Atlas	dual enc	FiD (T5)	KD	masked LM	60.4
(closed-book)	✗	T5 (11B)	✗	✗	32.8

- Retrieval is important for QA.
- Training a QA-specific retriever is important.
- Conditioning on a large number of passages is important.
- Can go a long way with pipelines.
- Jointly training retriever+reader (esp. w/ end-to-end pretraining) can yield large gains.

NQ Test Performance (Exact Match)

Finetuned on all question-answer pairs in training data

	retriever	reader	end-to-end	pretraining	EM
	BM25	extract (BERT)	✗	✗	26.5
ORQA	dual enc	extract (BERT)	MLL	✗	33.3
REALM	dual enc	extract (BERT)	MLL	masked LM	40.4
DPR	dual enc	extract (BERT)	✗	✗	41.5
RAG	dual enc	BART	MLL	multitasking	44.5
FiD	dual enc	FiD (T5)	✗	✗	51.4
FiD-KD	dual enc	FiD (T5)	KD	✗	54.7
Atlas	dual enc	FiD (T5)	KD	masked LM	60.4
(closed-book)	✗	T5 (11B)	✗	✗	32.8

- Retrieval is important for QA.
- Training a QA-specific retriever is important.
- Conditioning on a large number of passages is important.
- Can go a long way with pipelines.
- Jointly training retriever+reader (esp. w/ end-to-end pretraining) can yield large gains.

NQ Test Performance (Exact Match)

Finetuned on all question-answer pairs in training data

	retriever	reader	end-to-end	pretraining	EM
	BM25	extract (BERT)	✗	✗	26.5
ORQA	dual enc	extract (BERT)	MLL	✗	33.3
REALM	dual enc	extract (BERT)	MLL	masked LM	40.4
DPR	dual enc	extract (BERT)	✗	✗	41.5
RAG	dual enc	BART	MLL	multitasking	44.5
FiD	dual enc	FiD (T5)	✗	✗	51.4
FiD-KD	dual enc	FiD (T5)	KD	✗	54.7
Atlas	dual enc	FiD (T5)	KD	masked LM	60.4
(closed-book)	✗	T5 (11B)	✗	✗	32.8

- Retrieval is important for QA.
- Training a QA-specific retriever is important.
- Conditioning on a large number of passages is important.
- Can go a long way with pipelines.
- Jointly training retriever+reader (esp. w/ end-to-end pretraining) can yield large gains.

NQ Test Performance (Exact Match)

Finetuned on all question-answer pairs in training data

	retriever	reader	end-to-end	pretraining	EM
	BM25	extract (BERT)	✗	✗	26.5
ORQA	dual enc	extract (BERT)	MLL	✗	33.3
REALM	dual enc	extract (BERT)	MLL	masked LM	40.4
DPR	dual enc	extract (BERT)	✗	✗	41.5
RAG	dual enc	BART	MLL	multitasking	44.5
FiD	dual enc	FiD (T5)	✗	✗	51.4
FiD-KD	dual enc	FiD (T5)	KD	✗	54.7
Atlas	dual enc	FiD (T5)	KD	masked LM	60.4
(closed-book)	✗	T5 (11B)	✗	✗	32.8

- Retrieval is important for QA.
- Training a QA-specific retriever is important.
- Conditioning on a large number of passages is important.
- Can go a long way with pipelines.
- Jointly training retriever+reader (esp. w/ end-to-end pretraining) can yield large gains.

NQ Test Performance (Exact Match)

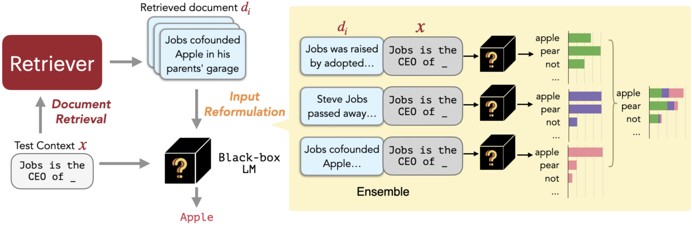
Finetuned on all question-answer pairs in training data

	retriever	reader	end-to-end	pretraining	EM
	BM25	extract (BERT)	✗	✗	26.5
ORQA	dual enc	extract (BERT)	MLL	✗	33.3
REALM	dual enc	extract (BERT)	MLL	masked LM	40.4
DPR	dual enc	extract (BERT)	✗	✗	41.5
RAG	dual enc	BART	MLL	multitasking	44.5
FiD	dual enc	FiD (T5)	✗	✗	51.4
FiD-KD	dual enc	FiD (T5)	KD	✗	54.7
Atlas	dual enc	FiD (T5)	KD	masked LM	60.4
(closed-book)	✗	T5 (11B)	✗	✗	32.8

- Retrieval is important for QA.
- Training a QA-specific retriever is important.
- Conditioning on a large number of passages is important.
- Can go a long way with pipelines.
- Jointly training retriever+reader (esp. w/ end-to-end pretraining) can yield large gains.

Retrieval-Based Prompting for LLMs

Can we skip training and just prompt "black-box" LLMs to use retrieved passages? E.g., **REPLUG** (Shi et al., 2023): "FiD + ensemble prompting"



While the LLM is frozen, the retriever can be learned (e.g., by KD).

- MMLU 5-shot prompting with Codex-175B: $68.3 \rightarrow 71.8$ (vs 69.3 with PaLM-540B without retrieval)
- NQ 16-shot promting with Codex-175B: $40.6 \rightarrow 45.5$ (vs 42.4 with Atlas-11B *finetuned* on 64 examples)
- Cons: Need a L(Large)LM, may not be as high performance as fully finetuned model (e.g., NQ 60.4 with Atlas)