Defⁿ

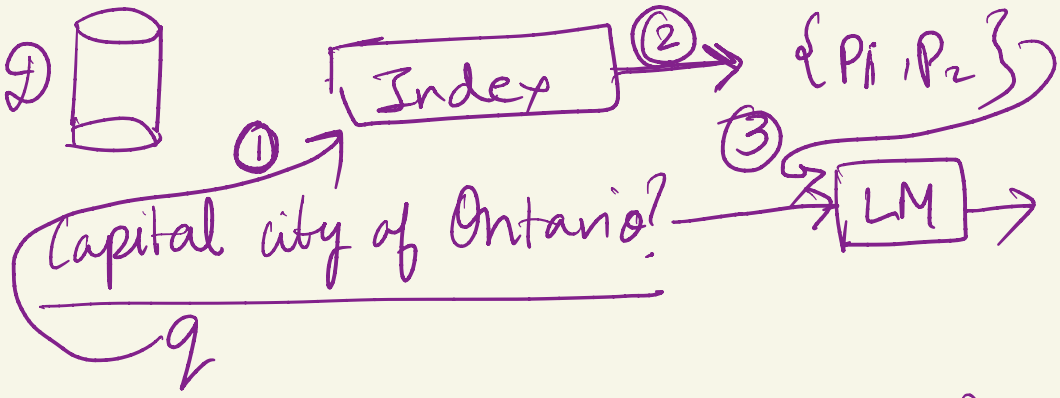
An LM that uses an external datastore at test time

Corpus of documents

$\mathcal{D} = \{p_1, p_2 \ldots \ldots \ldots p_N\}$

Why do we need retrieval?

- LLMs are static
- LLMs cannot memorize all knowledge
-



Standard $y \sim P(\cdot \mid q)$

Retrieval-based LM $y \sim P(\cdot \mid q, \{P_1, P_2\})$

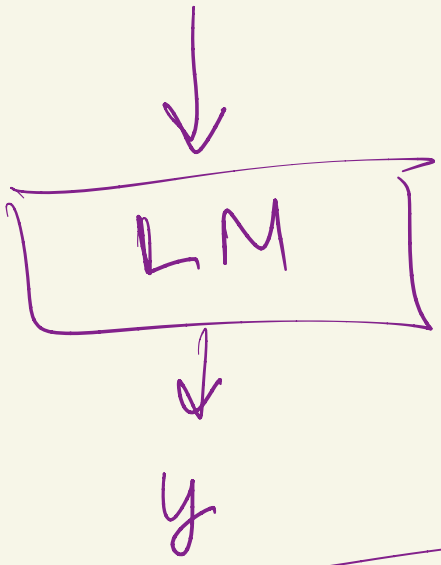
"Retrieval Model"

Creates an index of docs in $\mathcal{D}$

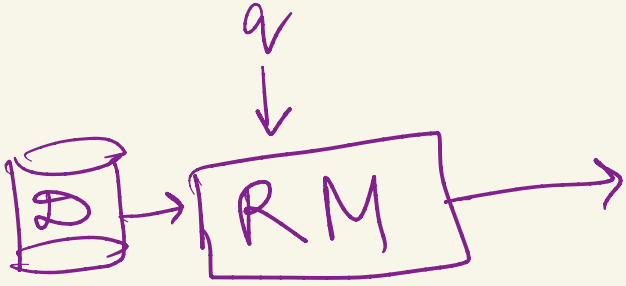
Given q at test time, retrieves the most relevant docs.

<P1>
<P2>

What is the capital of Ontario?



The function of the retrival model RM is to returns this ranked list of docs.



Simple count-based technique.

Represent any text (q or p_i) with a vector of size |V|.

$q = [0 \; \underset{\text{capital}}{1} \; 0 \; \ldots \; 0 \; \underset{\text{Ontario}}{1}]$

Limitations?

① Similarity comp. will be dominated by common words

② Vectors are sparse.

We can fix ① using tf-idf.

We can fix ② using a dense representation of documents / queries

Pre-processing

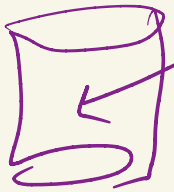
$E \rightarrow$ embedder

$E : t \rightarrow \mathbb{R}^d$

t ↙ any text

$\mathcal{D} \rightarrow$ for each $d \in \mathcal{D}$

$e^d = E(d)$



store key e^d in the index

At test

$E(q) = e^q$

- Compute sim. b/w e^q and all docs.

$$\begin{bmatrix} e^{d_1} \\ e^{d_2} \\ \vdots \end{bmatrix} [e^q] = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_N \end{bmatrix}$$

extract top-k docs to add to the context.

Dense-retrievers-

What we want.

— e^q should be close to documents that answer q. and disimilar to irrelevat documents.

Suppose we have a training data

q_1 , p_1

q_2 , p_2

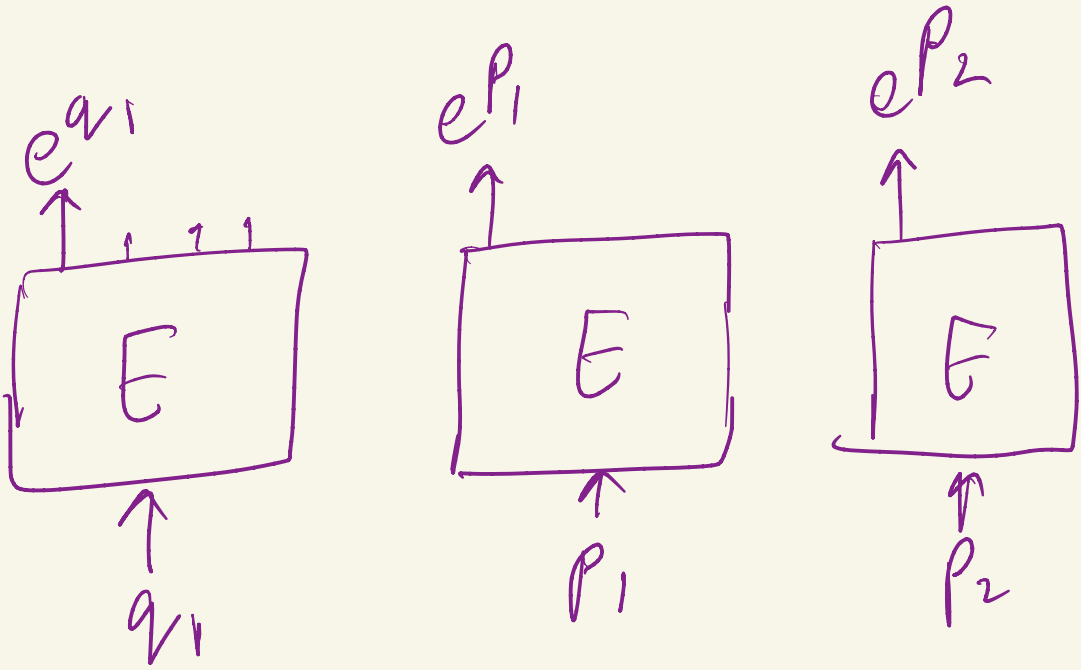
q_3 , p_3

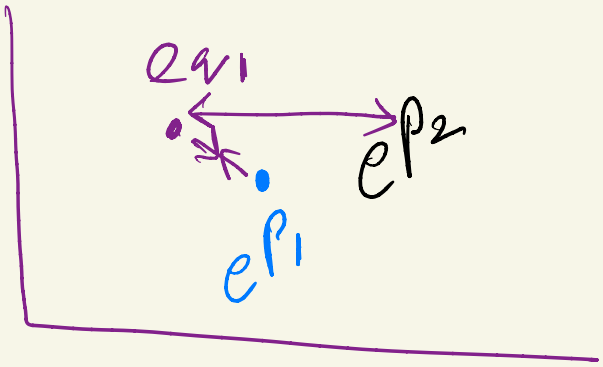
q_4 , p_4

⋮

Here, p_i is a relevant doc. for answering query q_i

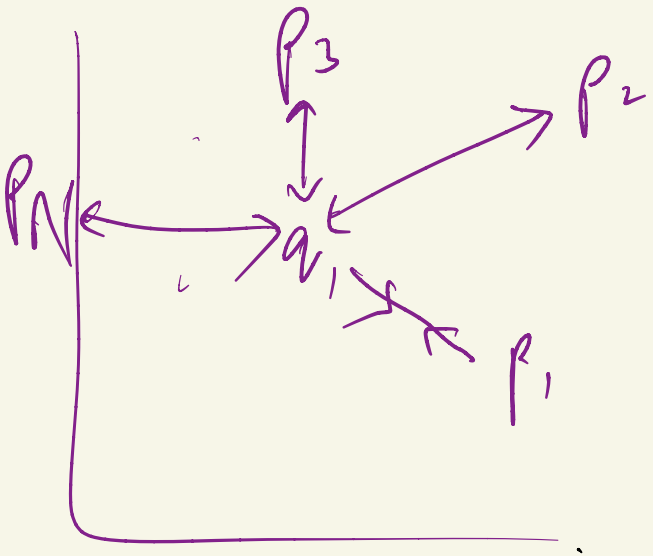
We use a BERT-style model as our embedder to represent text





Want to push e^{q_1} close to e^{p_1} and far from e^{p_2} (irrelevant doc)

$q_1, \overset{+}{p_1}, \overset{-}{p_2}, \overset{-}{p_3} \cdots \overset{-}{p_N}$



(using q_i to represent e^{q_i}, p_i to represent e^{p_i} for simplicity)

Minimizing contrastive loss achives this

$$\text{loss} = -\log \sum_i \frac{e^{\text{sim}(q_i, p_i)}}{e^{\text{sim}(q_i, p_i)} + \sum_{j \neq i} e^{\text{sim}(q_i, p_j)}}$$

Want the numerator to be high

similarity b/w q_i and all irrelevant docs. Want this to be low.

In-batch negative sampling

Consider the above loss formulation for one query-para pair (q_i, p_i).

Computing the denominator requires computing sim. with all paragraphs / documents in the corpus

Very Expensive !!.

Solution : Only consider in-batch documents as the -ve documents.

In-batch negative sampling

q_1 p_1

q_2 p_2

batch size = 10

.

.

q_{1000} p_{1000}

for each batch

$$loss = -\sum \log \frac{e^{sim(q_i, p_i)}}{e^{sim(q_i, p_i)} + \sum_{j \neq i \text{ in batch}} e^{sim(q_i, p_j)}}$$

only consider in-batch docs as -ve samples

More comp. efficient!