# CSE 156 | Lecture 11: Retrieval Augmented Generation (RAG)

Ndapa Nakashole

November 05, 2024

# Administrative matters

- **PA2**: due on Wednesday, November 6th, on Gradescope
- **PA3**: released on Wednesday, November 6th
- **Midterm review**: November 7th, Thursday, regular class time
- **Midterm**: next Tuesday, November 12th, 2024. Bring the following:
  - Your student ID
  - One double-sided note sheet
  - A pen

# Recap: GPT-3: Few-shot Learning

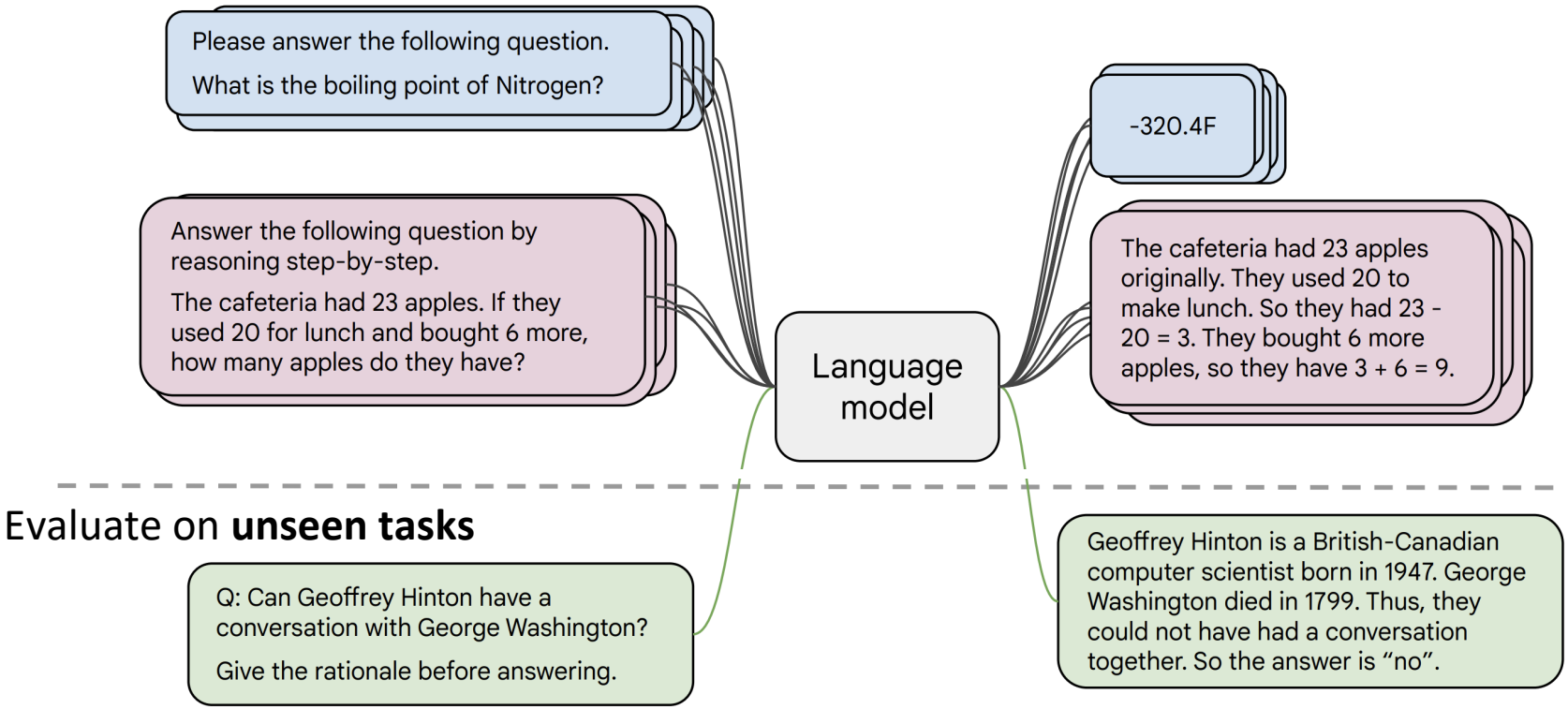
- GPT-3 **in-context learning**: Just uses the off-the-shelf model no gradient updates


Brown et al. (2020)

- This procedure depends heavily on the examples you pick as well as the prompt ("Translate English to French")

# Recap: Instruction finetuning - from LMs to AI assistants

**Collect examples** of (instruction, output) pairs across many tasks and finetune an LM




[FLAN-T5; Chung et al., 2022]

# Today

1. RAG Overview
2. Classical Retrieval
3. Neural Retrieval
4. Unsupervised Neural Retrieval

# How to scale up LLMs without breaking the bank?

- Transformer LMs are monolithic - store knowledge **inside the parameters**, but current **memory capacity is too small**, and scaling up is expensive!
- Costs, not including data preparation, comparison runs at smaller scale, architecture, experiments, etc.:

| | params. | hardware | days | est. AWS cost |
|---|---|---|---|---|
| Llama2 (Meta) | 70 B | 6,000 GPUs | 12 | $2M |
| GPT-3 (OpenAI) | 175 B | 1,500 GPUs | 60 | $3M |
| PaLM (Google) | 540 B | 6,144 TPUs | 57 | $25M |

- **Open question:** Is it possible to achieve results on par with the best seen, but with **less data, less parameters, less computationally-intensive training**?

# Wish List

- Fast and modular knowledge editing
  - Robustly update the model $\mathrm{N}$ times without breaking its behavior on other tasks
- Attribution and interpretability
  - Trace a model's knowledge back to a particular document / training example
- Efficient scaling
  - Increase the model's memory size by $10x$ without paying $10x$ more compute.

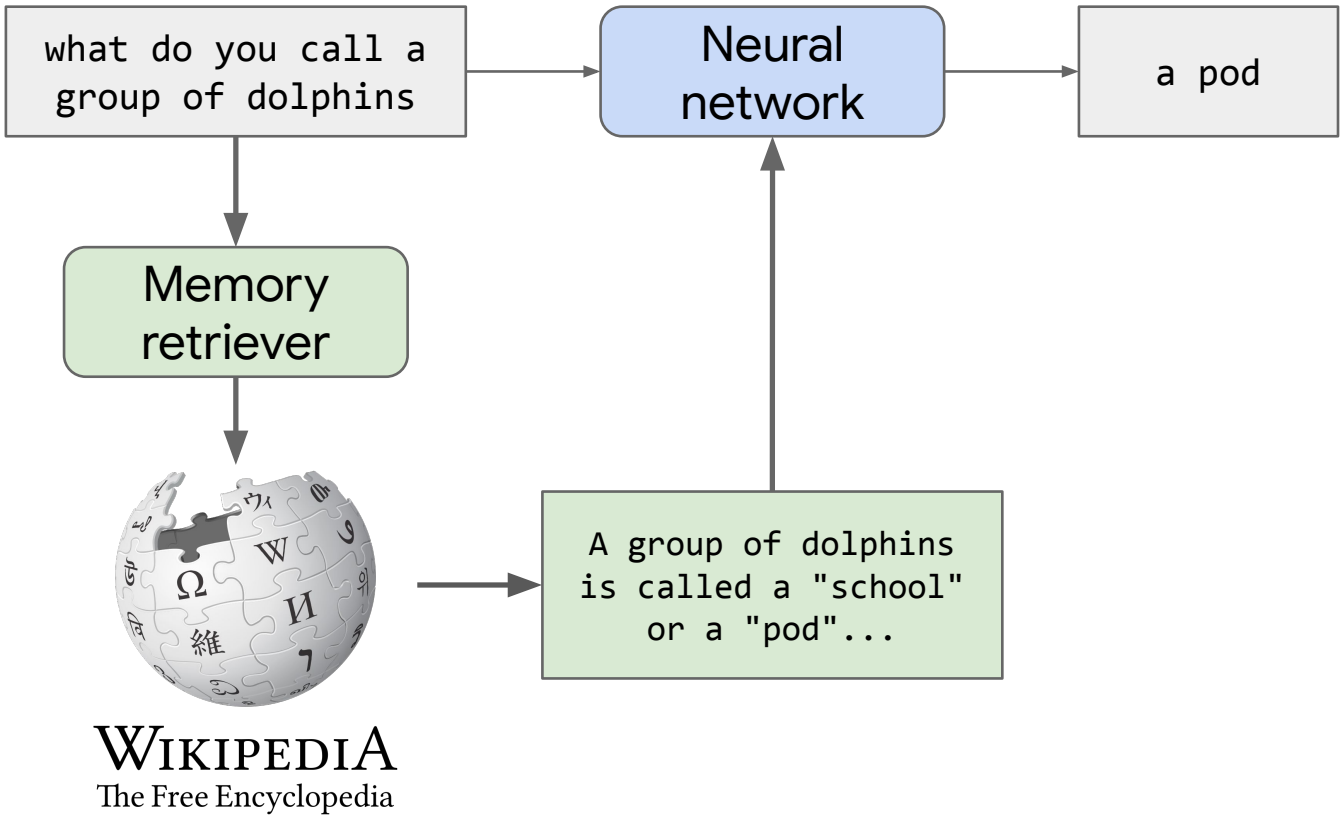
# Memory-Augmented Models

Allow neural model (LM, QA, etc) to interact with external memory (e.g., a search engine, database, ...).

Can then potentially support:

- Editing of knowledge
- Attribution
- Efficient scaling

# What is a Retrieval Augmented Generation (RAG) model?




- E.g.: **Open-domain dialog / question answering** - retrieve text on the web; **Fact checking** - retrieve text to support or refute a claim; **Language modeling** - retrieve text to support generation

# Classical Information Retrieval

Still a strong baseline (Contriver* only outperforms BM25 on 11 out of 15 datasets)

May be useful in combination with neural models (as in DrQA†)

*Izacard et al. 2021, "Unsupervised Dense Information Retrieval with Contrastive Learning"

†Danqi Chen, Adam Fisch, Jason Weston, Antoine Bordes. ACL 2017 "Reading Wikipedia to Answer Open-Domain Questions"

# The term-document matrix

- Term-document matrix: each row corresponds to a term, each column to a document
- Very high-dimensional, and sparse

| | d1 | d2 | d3 | d4 | d5 | d6 | d7 | d8 | d9 | d10 | ··· |
|---|---|---|---|---|---|---|---|---|---|---|---|
| against | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 2 | 3 | 0 | |
| age | 0 | 0 | 0 | 1 | 0 | 3 | 1 | 0 | 4 | 0 | |
| agent | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ages | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | |
| ago | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 3 | 0 | |
| agree | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ahead | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ain't | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| air | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| aka | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| ⋮ | | | | | | | | | | | |

# TF-IDF weighting

For a corpus of documents $D$:

- **Term frequency**: $\mathbf{TF}(w, \text{doc}) = \frac{\text{count}(w, \text{doc})}{|\text{doc}|}$
- **Document frequency**: $\mathbf{df}(w, D) = |\{\text{doc} \in D : w \in \text{doc}\}|$
- **Inverse document frequency**: $\mathbf{IDF}(w, D) = \log\left(\frac{|D|}{\mathbf{df}(w,D)}\right)$
- **TF-IDF**$(w, \text{doc}, D) = \text{TF}(w, \text{doc}) \cdot \text{IDF}(w, D)$

| | $\text{doc}_1$ | $\text{doc}_2$ | $\text{doc}_3$ | $\text{doc}_4$ |
|---|---|---|---|---|
| *A* | 10 | 10 | 10 | 10 |
| *B* | 10 | 10 | 10 | 0 |
| *C* | 10 | 10 | 0 | 0 |
| *D* | 0 | 0 | 0 | 1 |

⇒

| | IDF |
|---|---|
| *A* | 0.00 |
| *B* | 0.29 |
| *C* | 0.69 |
| *D* | 1.39 |

⇓

TF

| | $\text{doc}_1$ | $\text{doc}_2$ | $\text{doc}_3$ | $\text{doc}_4$ |
|---|---|---|---|---|
| *A* | 0.33 | 0.33 | 0.50 | 0.91 |
| *B* | 0.33 | 0.33 | 0.50 | 0.00 |
| *C* | 0.33 | 0.33 | 0.00 | 0.00 |
| *D* | 0.00 | 0.00 | 0.00 | 0.09 |

TF-IDF

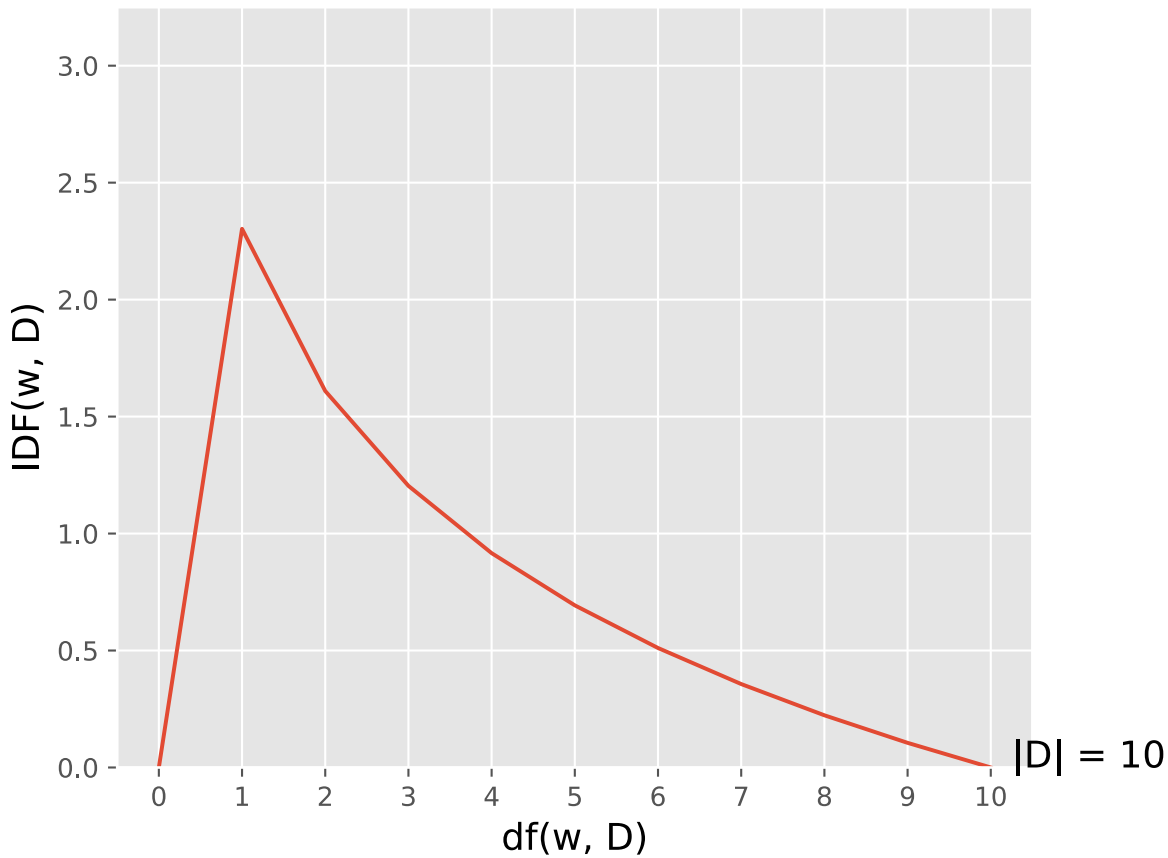
| | $\text{doc}_1$ | $\text{doc}_2$ | $\text{doc}_3$ | $\text{doc}_4$ |
|---|---|---|---|---|
| *A* | 0.00 | 0.00 | 0.00 | 0.00 |
| *B* | 0.10 | 0.10 | 0.14 | 0.00 |
| *C* | 0.23 | 0.23 | 0.00 | 0.00 |
| *D* | 0.00 | 0.00 | 0.00 | 0.13 |

# IDF value

$$\mathbf{IDF}(w, D) = \log\left(\frac{|D|}{\mathbf{df}(w, D)}\right)$$

For a corpus of 10 documents, the IDF value of a term $w$ is as follows:

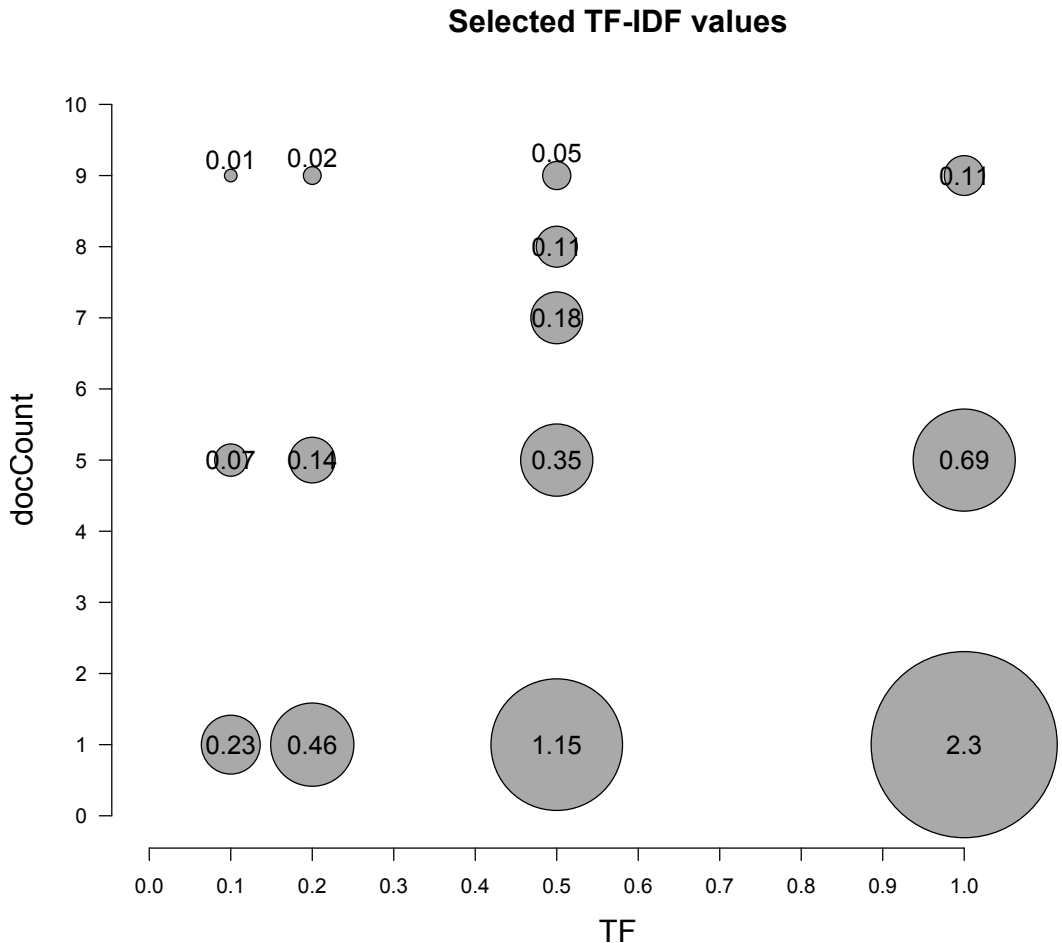
- $\text{df}_w = 1$: $\log_e\left(\frac{10}{1}\right) = 2.3$
- $\text{df}_w = 10$: $\log_e\left(\frac{10}{10}\right) = 0$



# Selected TF-IDF values

$$\mathrm{TF-IDF}(w, \mathrm{doc}, D) = \mathrm{TF}(w, \mathrm{doc}) \cdot \mathrm{IDF}(w, D)$$

- Peaks for terms occur very frequently in a document but are rare in the corpus.
- Want words that are distinguishing indicators of particular documents.



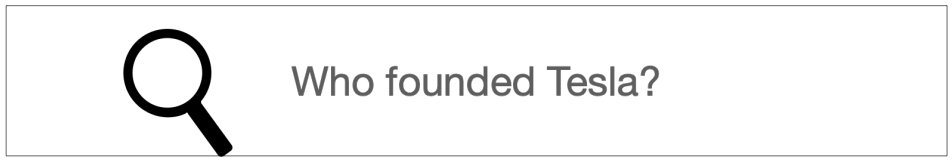
# Relevance Score

$$\text{RelevanceScore}\ (q, \text{doc}, D) = \sum_{w \in q_0} \textbf{Weight}(w, \text{doc}, D)$$

where **Weight** is often TF-IDF, but can be other functions as well (e.g., **BM25** an extension of TF-IDF with TF smoothing and document length normalization).

# Inverted Index

- **Inverted index**: precomputed for each term-document pair: **very scalable**




Term look-up

| | | |
|---|---|---|
| founded | $(\text{doc}_{47}, 0.90), (\text{doc}_{39}, 0.76), (\text{doc}_{41}, 0.76), \ldots$ | 0.12 |
| fountain | $(\text{doc}_{21}, 0.65), (\text{doc}_{64}, 0.60), (\text{doc}_{16}, 0.10), \ldots$ | 0.88 |
| $\vdots$ | | |
| Test | $(\text{doc}_{21}, 0.91), (\text{doc}_{11}, 0.89), (\text{doc}_{17}, 0.50), \ldots$ | 0.01 |
| Tesla | $(\text{doc}_{47}, 0.29), (\text{doc}_{39}, 0.01), (\text{doc}_{68}, 0.10), \ldots$ | 0.56 |
| $\vdots$ | | |
| Inc. | $(\text{doc}_{21}, 0.91), (\text{doc}_{39}, 0.90), (\text{doc}_{68}, 0.76), \ldots$ | 0.01 |

Document scoring

| | |
|---|---|
| doc39 | History of Tesla, Inc. |
| doc47 | Tesla, Inc. Wikipedia |
| doc64 | Tesla About Page |

# Beyond term matching

- Query and document expansion
- Phrase search
- Different document fields (e.g., title, body)
- Link analysis (e.g., PageRank)
- ...

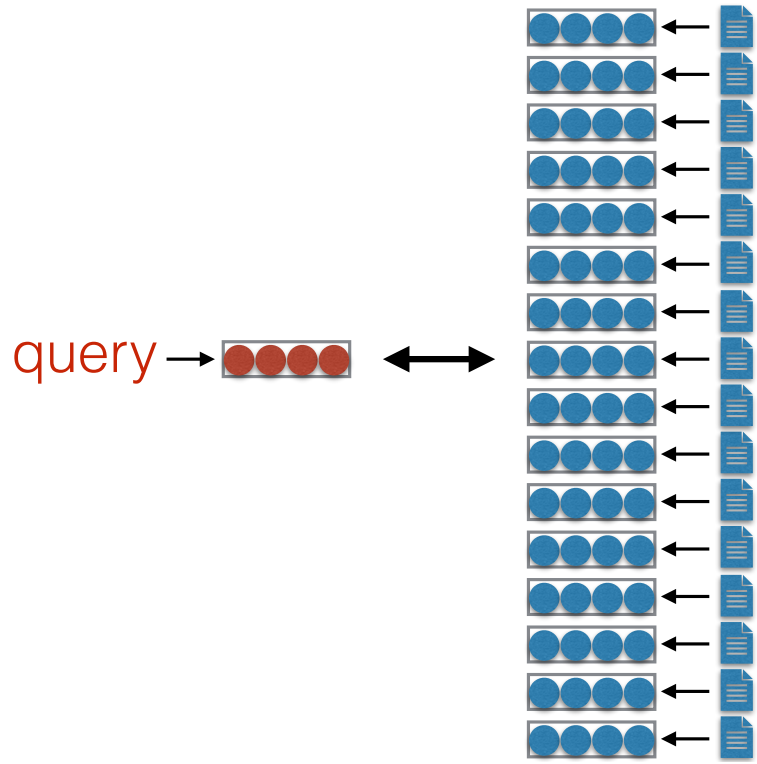
Tools for classical IR:

- ElasticSearch, Lucene, Pyserini, PrimeQA

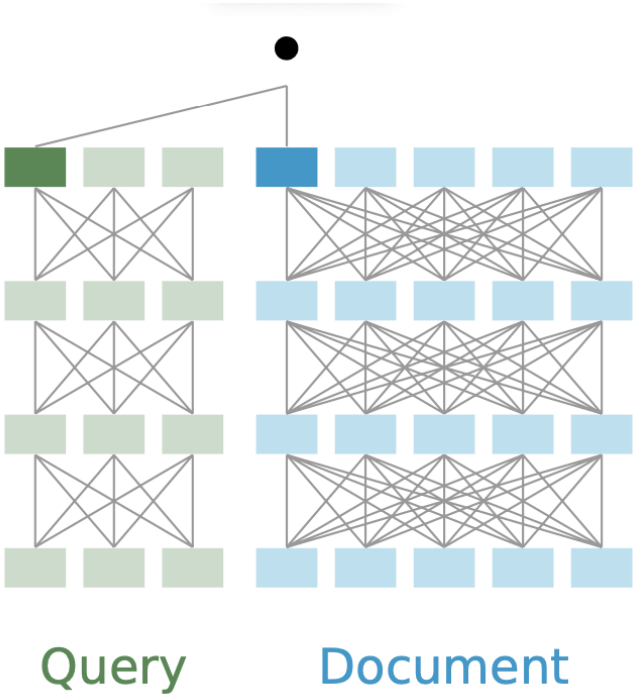
# Neural Information Retrieval

# Dense Retrieval

- **Problem in Classical IR: Vocabulary Mismatch**: Query terms may not match document terms exactly!
- **Dense Retrieval**: Represent queries and documents as dense vectors, and compute similarity between them.
- **Two types**:
  - Out-of-the-box embeddings (e.g., BERT as in PA3)
  - Learned/fine-tuned embeddings



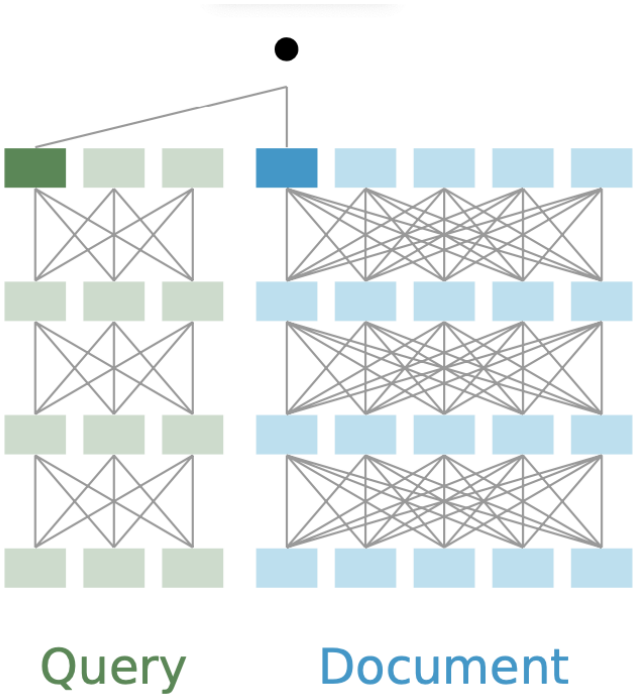
# Bi-Encoder




- Tokenize the query and the document
- **Independently** encode the query and the document
- … into a **single-vector** representation each
- Compare the two vectors using a similarity function (e.g., cosine similarity or dot product)

**Fast**: Document representations can be pre-computed! and Similarity computations are very cheap.

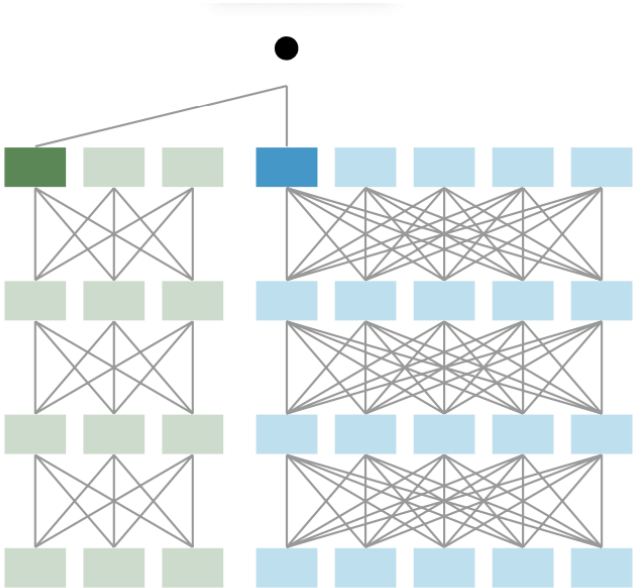
# Bi-Encoder Example: Dense Passage Retrieval (DPR)




- Encodes each query into a 768-dimensional vector
- Trained loss over the similarity scores between the query and:
  - A positive passage
  - A negative passage, sampled from BM25 top-100
  - Many in-batch negative passages: the positive passages for the other queries in the same training batch

[†]Vladimir Karpukhin, et al. "Dense passage retrieval for open-domain question answering." EMNLP 2020

# Bi-Encoder Training




1. Examples: $\left\langle q_i, \text{doc}_i^+, \left\{ \text{doc}_{i,k}^- \right\} \right\rangle$
2. For a BERT-style encoder with $N$ layers:

$$\text{Sim}(q, \text{doc}) = \text{EncQ}(q)_{N,0}{}^\top \, \text{EncD}(\text{ doc })_N$$

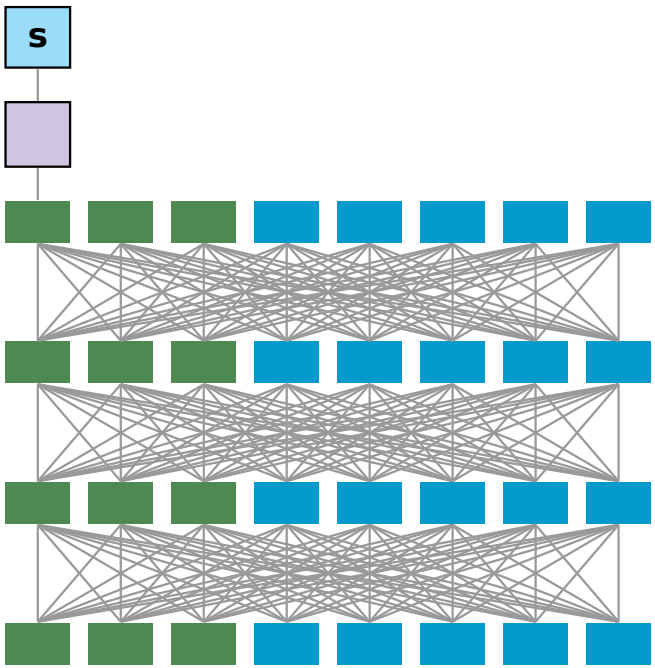
**Loss**: negative log-likelihood of the positive passage

$$-\log \frac{\exp\left(\mathbf{Sim}\left(q_i, \text{doc}_i^+\right)\right)}{\exp\left(\text{Sim}\left(q_i, \text{doc}_i^+\right)\right) + \sum_{j=1}^{n} \exp\left(\text{Sim}\left(q_i, \text{doc}_{i,j}^-\right)\right)}$$

Scalable, but limited query/doc interactions!

[†] Karpukhin et al. 2020

# Cross-Encoder




Query Document

1. Examples: $\left\langle q_i, \text{doc}_i^+, \left\{ \text{doc}_{i,k}^- \right\} \right\rangle$
2. For a BERT-style encoder with $N$ layers:

$$\mathbf{Rep}(q, \text{doc}) = \boldsymbol{Enc}\left([q; \text{doc}]_{N,0}\right)$$

**Loss**: negative log-likelihood of the positive passage

$$-\log \frac{\exp\left(\mathbf{Rep}\left(q_i, \text{doc}_i^+\right)\right)}{\exp\left(\mathbf{Rep}\left(q_i, \text{doc}_i^+\right)\right) + \sum_{j=1}^{n} \exp\left(\mathbf{Rep}\left(q_i, \text{doc}_{i,j}^-\right)\right)}$$

Rich interactions but doesn't scale!

# Can we keep precomputation and still have fine-grained interactions?

## ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT

Omar Khattab
Stanford University
okhattab@stanford.edu

Matei Zaharia
Stanford University
matei@cs.stanford.edu

**ABSTRACT**

Recent progress in Natural Language Understanding (NLU) is driving fast-paced advances in Information Retrieval (IR), largely owed to fine-tuning deep language models (LMs) for document ranking. While remarkably effective, the ranking models based on these LMs increase computational cost by orders of magnitude over prior approaches, particularly as they must feed each query–document pair through a massive neural network to compute a single relevance score. To tackle this, we present ColBERT, a novel ranking model that adapts deep LMs (in particular, BERT) for efficient retrieval. ColBERT introduces a *late interaction* architecture that independently encodes the query and the document using BERT and then employs a cheap yet powerful interaction step that models their fine-grained similarity. By delaying and yet retaining this fine-granular interaction, ColBERT can leverage the expressiveness of deep LMs while simultaneously gaining the ability to pre-compute document representations offline, considerably speeding up query

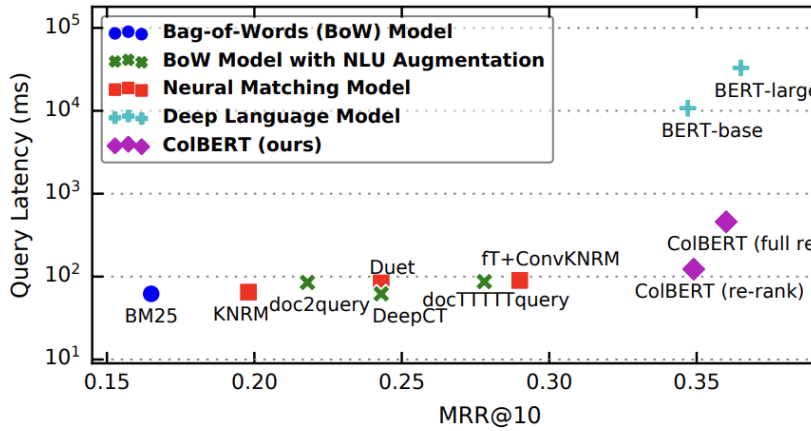
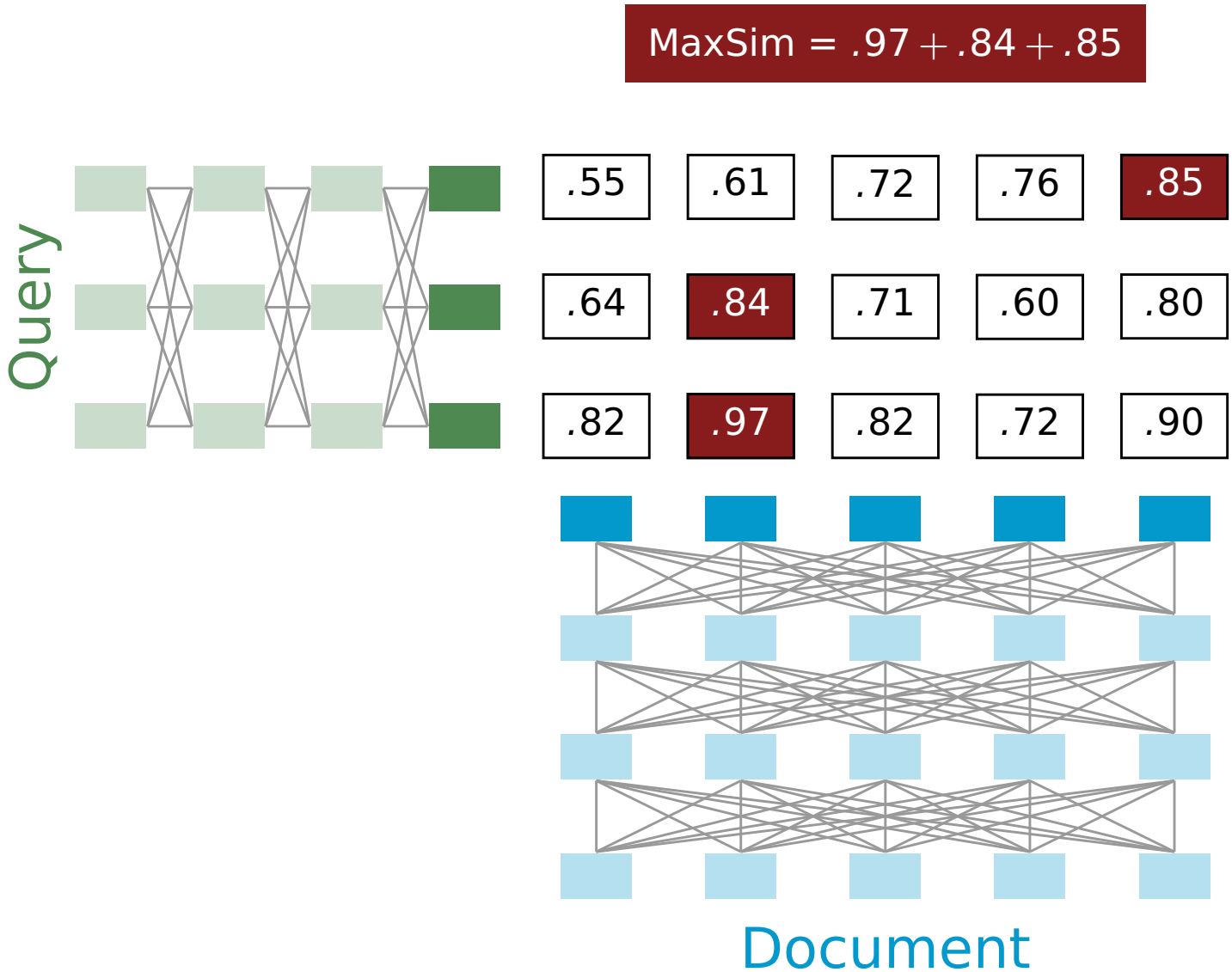
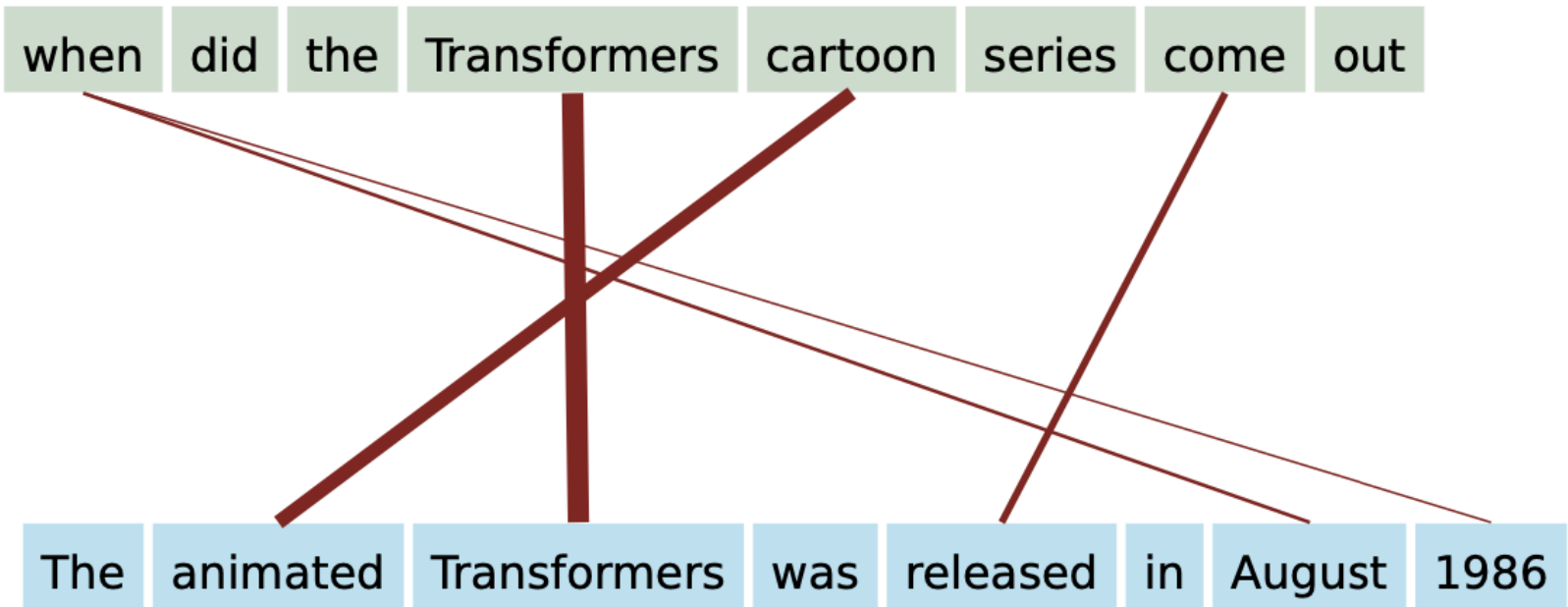



**Figure 1: Effectiveness (MRR@10) versus Mean Que tency (log-scale) for a number of representative ra models on MS MARCO Ranking [24]. The figure also ColBERT. Neural re-rankers run on top of the official top-1000 results and use a Tesla V100 GPU. Methodolo detailed results are in §4.**

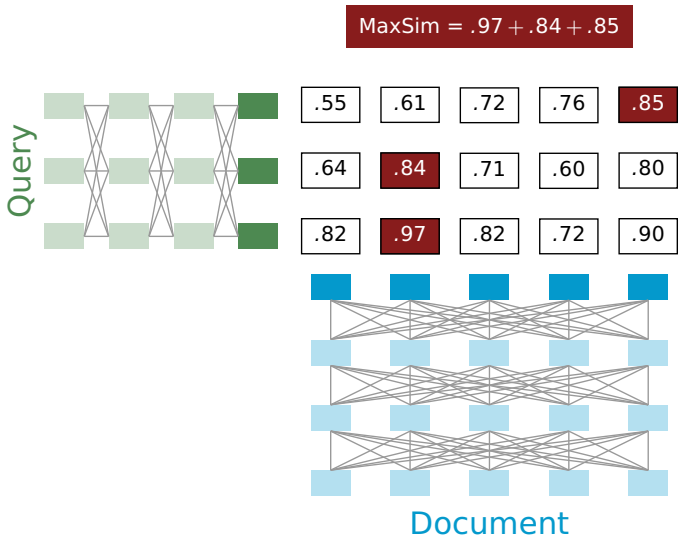



Notice that ColBERT represents the document as a MATRIX, not a vector like the bi-encoder! $\mathbf{d_i} \in \mathbb{R}^{L \times d}$

# Soft Alignment with ColBERT



# ColBERT




1. Examples: $\left\langle q_i, \mathrm{doc}_i^+, \left\{ \mathrm{doc}_{i,k}^- \right\} \right\rangle$
2. Loss: negative log-likelihood of the positive passage, with MaxSim as the basis.

$$\mathrm{MaxSim}(q, \mathrm{doc}) = \sum_i^L \max_j^M \mathrm{Enc}(q)_{N,i}{}^\top \mathrm{Enc}(\mathrm{doc})_{N,j}$$

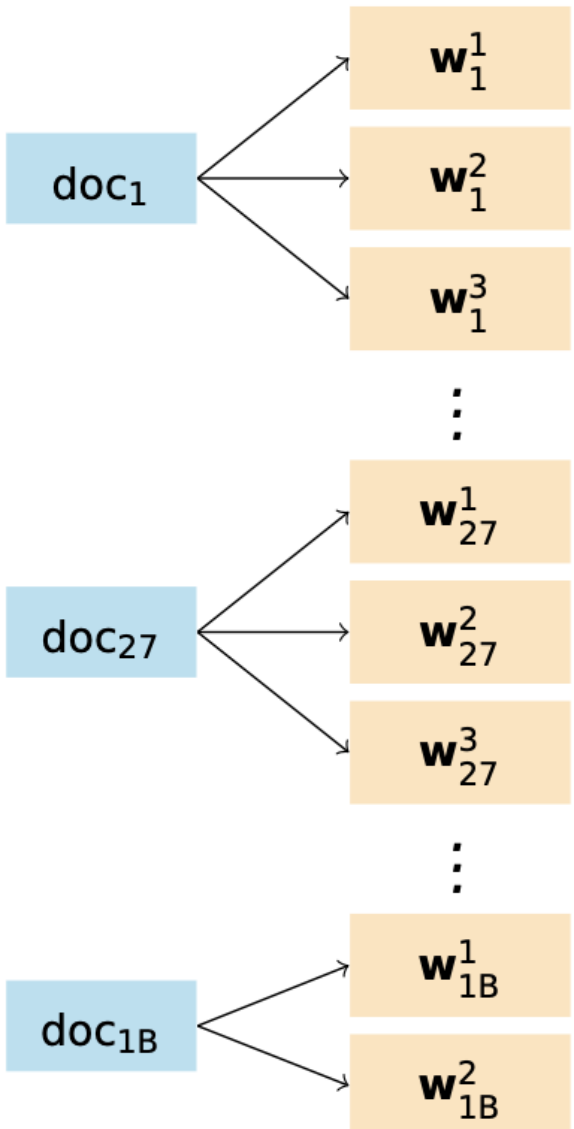
$N$ is the last BERT layer; $L$ is the length of $q$; $M$ the length of doc

Scalable with late, contextual interactions!

† Khattab and Zaharia 2020

# ColBERT as a reranker

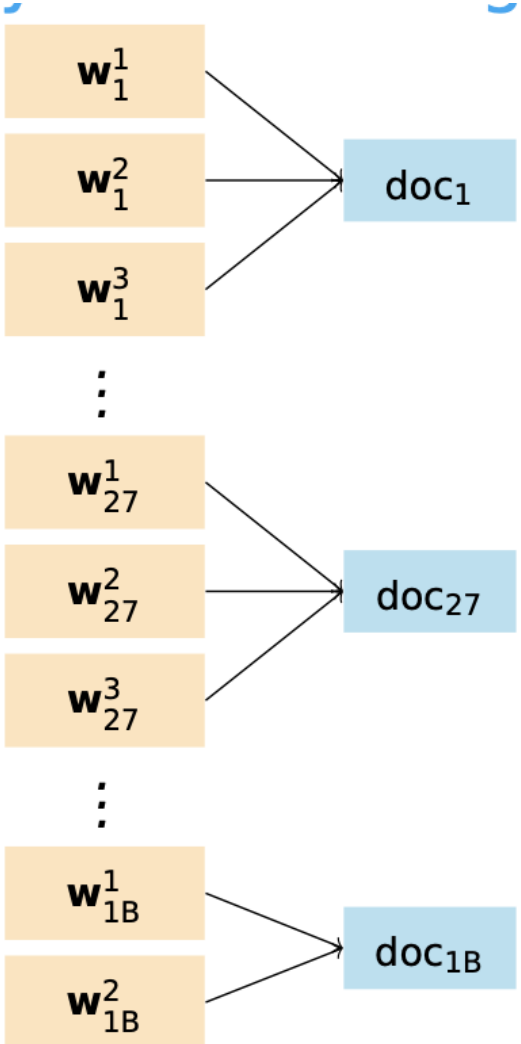
## PA3




Given query $q = [w^1, \ldots, w^M]$ :

1. Get the top $K$ documents for $q$ using a fast, term-based model like BM25 (or a Bi-encoder).
2. Score each of those top $K$ documents using ColBERT.

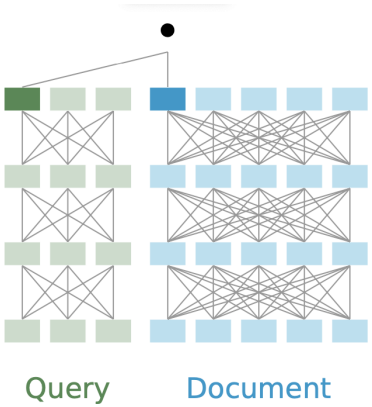
# Beyond reranking for ColBERT

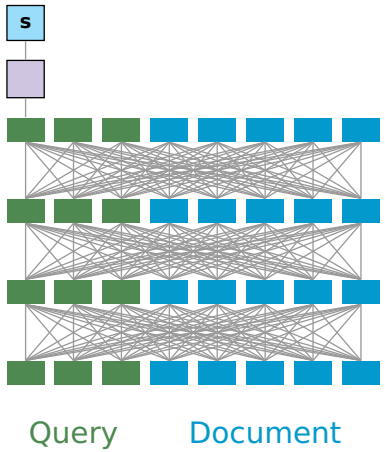


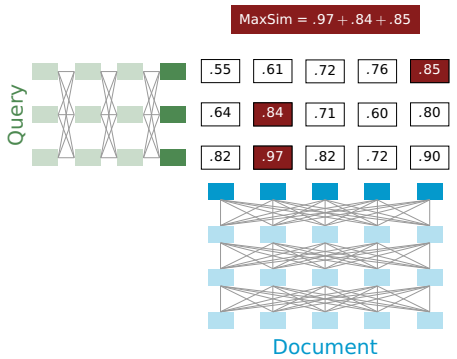

Given query $q$ encoded as vectors $\left[\mathbf{w}^1, \ldots, \mathbf{w}^M\right]$, for each query vector $\mathbf{w}^i$ :

1. Retrieve the $p$ most similar token vectors $\mathbf{w}_j^k$ to $\mathbf{w}^i$.
2. Score each $\text{doc}_j$ using ColBERT.

# Summary: Neural Ranking Paradigms




## Bi-Encoder

- ✓ Independent, Pre-computed Representations
- ✗ Coarse-Grained Representations

## Cross-Encoder

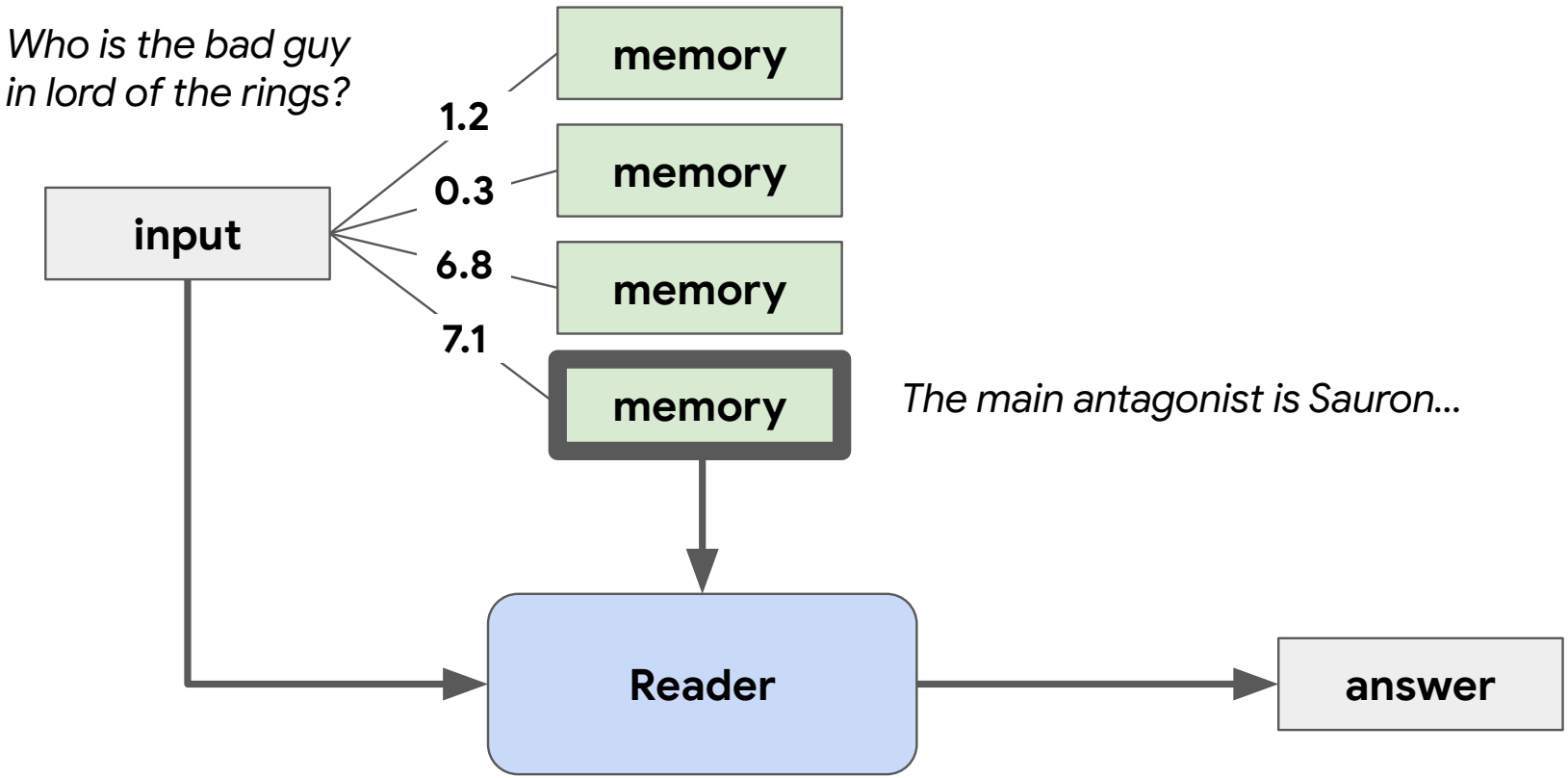
- ✓ Fine-Grained Query-Document Interactions
- ✗ No Pre-computed Representations

## Contextualized Late Interaction over BERT (**ColBERT**)

- ✓ Independent, Pre-computed Representations
- ✓ Fine-Grained Query Document Interactions

# Using Memories in Pretrained Transformer LMs

# How to use memories




- How to use memories. **Simple Text Fusion**: Combine retrieved memories with the input.

# Memory fusion with Cross attention: Retrieval Transformer (RETRO)

ICML 2022, DeepMind

DeepMind

## 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

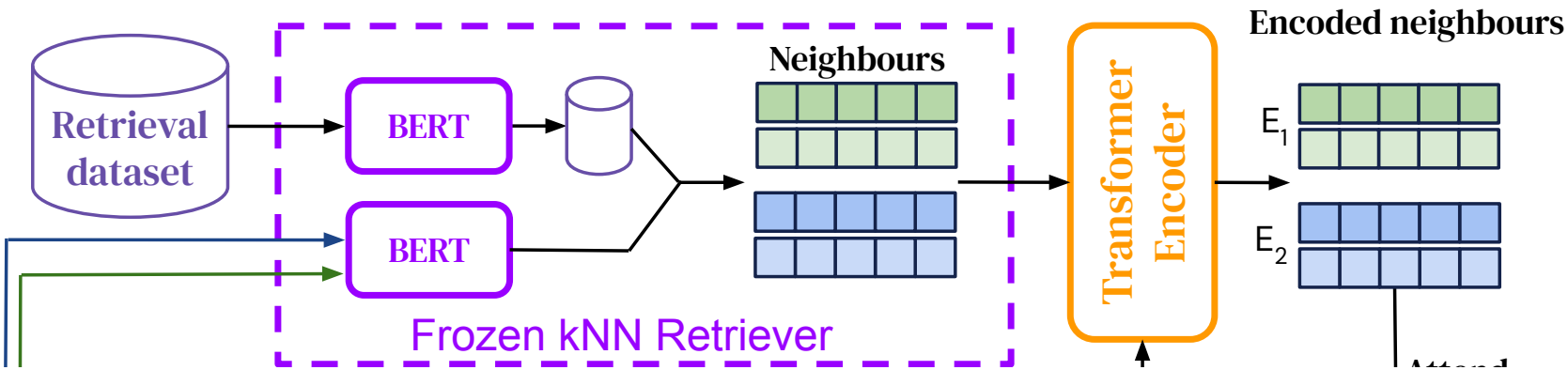
- uses a **2 trillion token database**, obtains comparable performance to GPT-3 and others, but with **25× fewer parameters**

# Retrieval Transformer (RETRO) for Language Modeling

Goal: **retrieve from a database with trillions of tokens** during training and inference

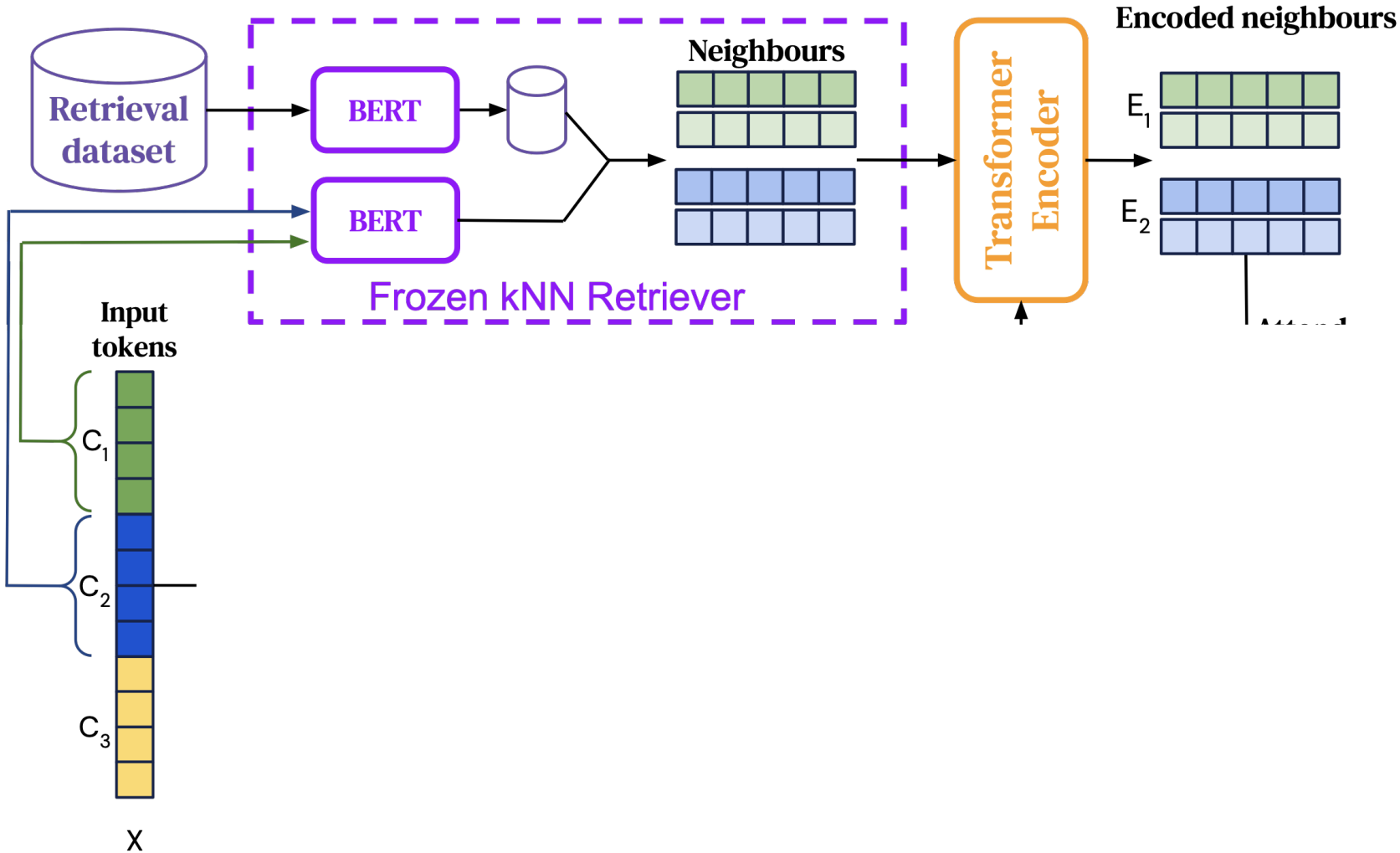
- Operates at a **chunk** level rather than individual tokens, reducing storage and computation requirements
- Training sequences are split into chunks, **augmented with their $k$-nearest neighbours** retrieved from the database
- An encoder-decoder architecture integrates retrieved chunks using **cross-attention**.

# Retrieval Transformer (RETRO): Database



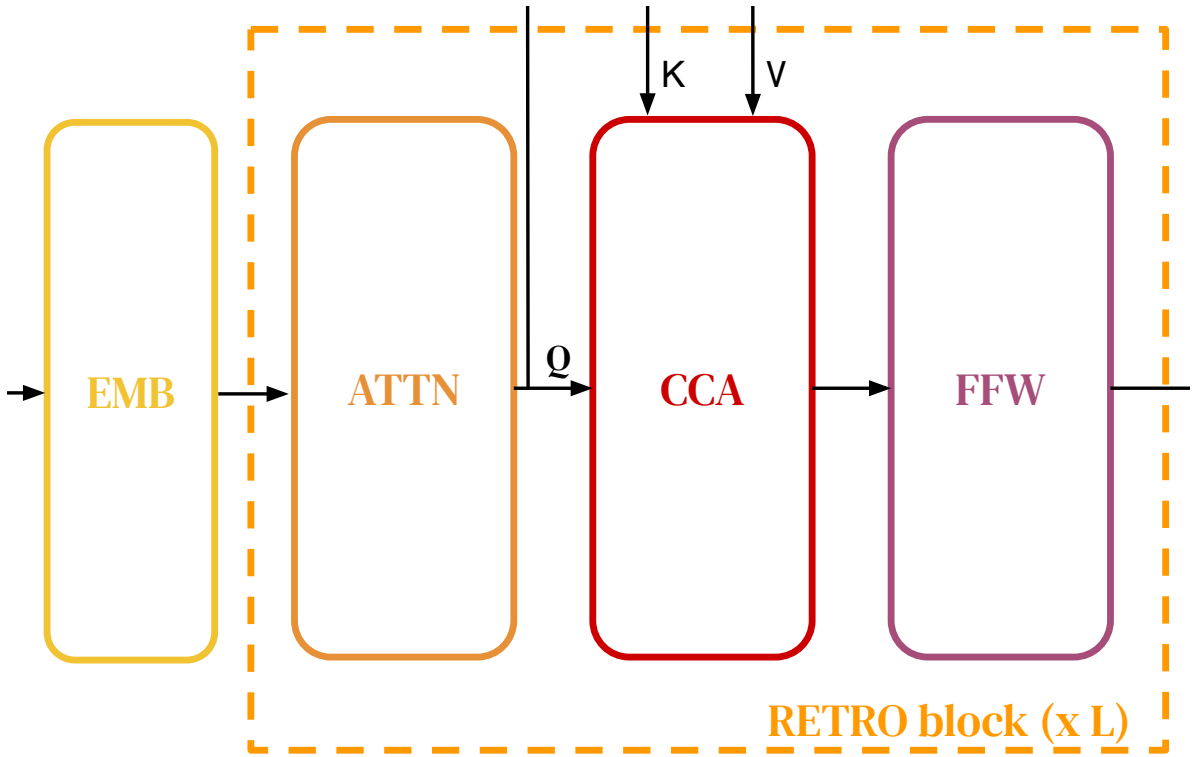

- Database is a **key-value memory**
  - Each **value** consists of two contiguous token chunks, denoted $[N, F]$:
    - $N$: The neighbour chunk used to compute the key.
    - $F$: The continuation chunk in the original document.
  - The **key** is the BERT embedding of $N$, averaged over the chunk.

# Retrieval Transformer (RETRO): Retrieval



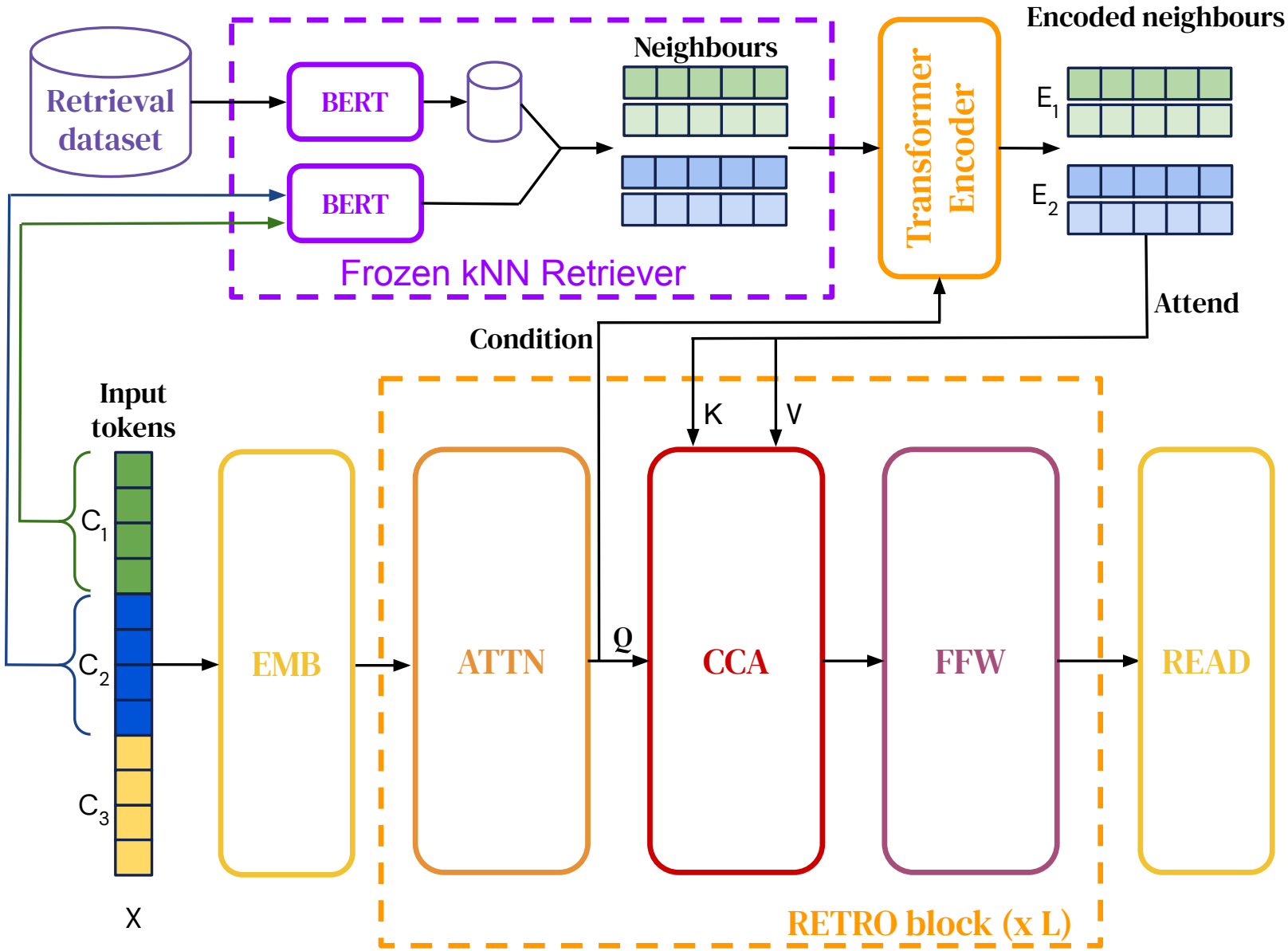

- For each chunk $C$, retrieve its $k$-**nearest neighbours** $\mathrm{RET}(C) \triangleq \left(\left[N^1, F^1\right], \ldots, \left[N^k, F^k\right]\right)$.
- An Encoder processes the retrieved chunks.

# Chunked Cross-Attention (CcA) in RETRO Blocks




- **Cross-Attention with Encoded Retrieval Set**: For each chunk, cross-attention is computed on the retrieved neighbours simultaneously.

# Memory fusion with Cross attention



# Retrival Transformer: Retrieval-Enhanced Language Model

- **RETRO**: a retrieval-enhanced autoregressive language model
- A novel attention block in which: **chunked cross-attention layer** is used to incorporate the retrieved text
- Retrieval is based on a **pre-trained frozen BERT model** (as in PA3) thus, it does not need to train and update a retriever network

# When do we Retrieve?

**RETRO** retrieves at every $k$ tokens (fixed chunks)

# When do we Retrieve?

- **Once, at the beginning of generation**. Default method used by most systems
  - limiting in more general scenarios involving generation of **long texts**, where continually gathering information throughout generation is essential
- **Several times during generation**, as necessary
  - **RETRO** retrieves at every $k$ tokens (fixed chunks)
  - **Toolformer** (Schick et al. 2023): generate a search token

# Toolformer

## Language Models Can Teach Themselves to Use Tools

- **Toolformer** generates tokens that trigger retrieval (or other tools)

> The New England Journal of Medicine is a registered trademark of **[QA("Who is the publisher of The New England Journal of Medicine?") → Massachusetts Medical Society]** the MMS.

> Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

> The name derives from "la tortuga", the Spanish word for **[MT("tortuga") → turtle]** turtle.

> The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the**

# Toolformer Training

Given a dataset set $\mathcal{C} = \left\{\mathbf{x}^1, \ldots, \mathbf{x}^{|\mathcal{C}|}\right\}$ of plain texts. First convert this dataset into a dataset $\mathcal{C}^*$ augmented with API calls.

1. **Few-shot Prompt LM:** Let a LM annotate a huge language modeling dataset with potential API calls
2. **Filtering out unhelpful API calls**: execute these API calls and filter out all calls which do not reduce the loss $L_i$ over the next tokens this creates a new dataset $\mathcal{C}^*$
3. **Finetune** the LM on the API calls that it considers useful, $\mathcal{C}^*$

# Toolformer: use LM to get large number of potential API calls

*Your task is to add calls to a Question Answering API to a piece of text. The questions should help you get information required to complete the text. You can call the API by writing "[QA(question)]" where "question" is the question you want to ask. Here are some examples of API calls:*

**Input:** Joe Biden was born in Scranton, Pennsylvania.

**Output:** Joe Biden was born in **[QA("Where was Joe Biden born?")]** Scranton, **[QA("In which state is Scranton?")]** Pennsylvania.

**Input:** Coca-Cola, or Coke, is a carbonated soft drink manufactured by the Coca-Cola Company.

**Output:** Coca-Cola, or **[QA("What other name is Coca-Cola known by?")]** Coke, is a carbonated soft drink manufactured by **[QA("Who manufactures Coca-Cola?")]** the Coca-Cola Company.

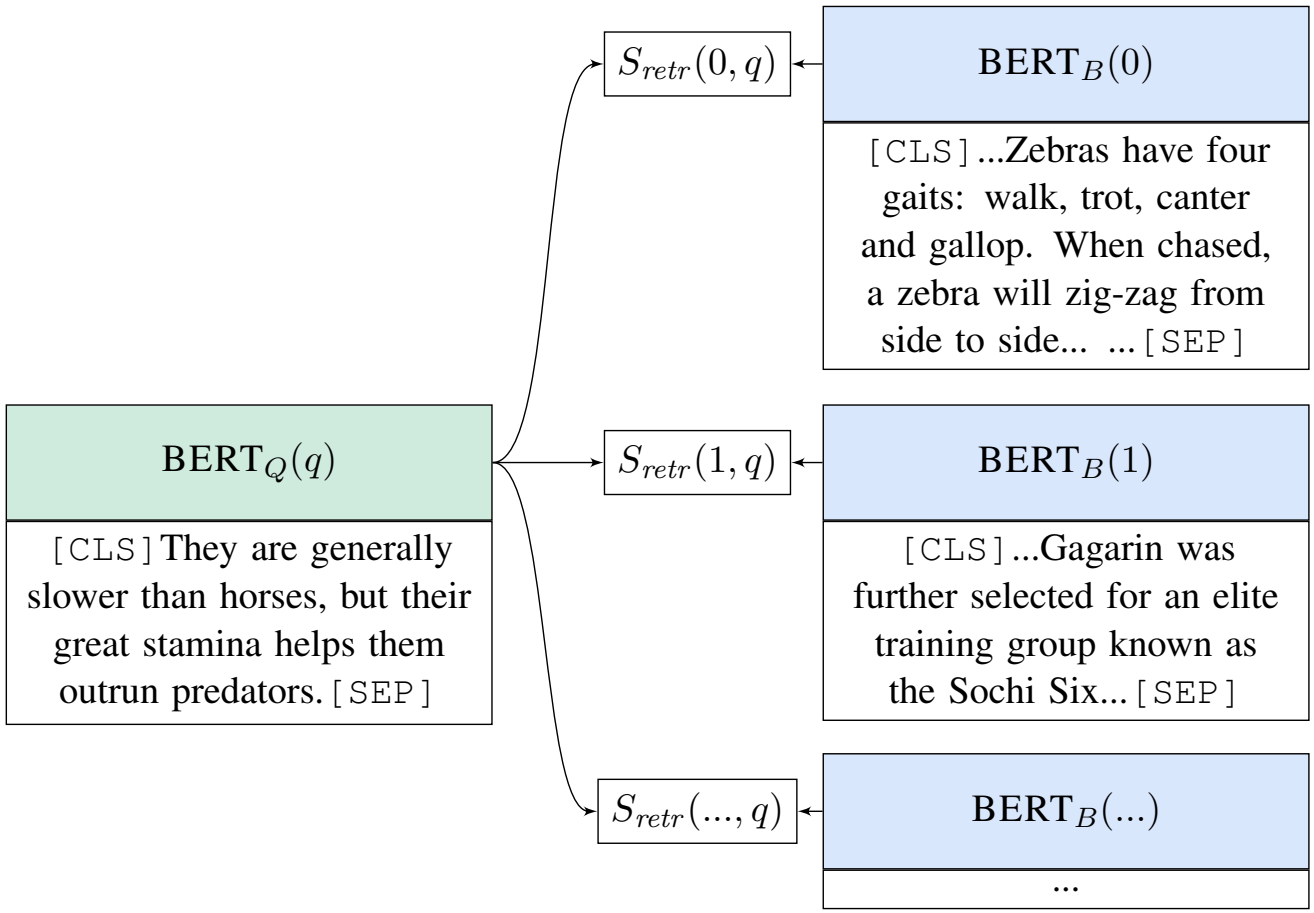
**Input: x**

**Output:**

# Unsupervised Neural Retrieval

# General purpose unsupervised embeddings for dense retrieval

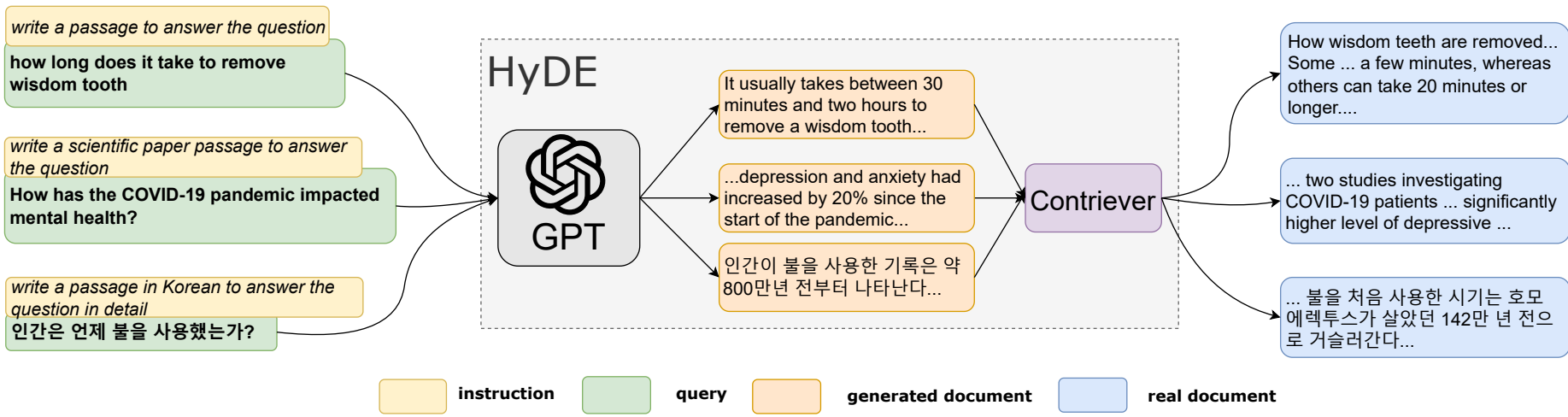
- **The inverse cloze objective** (Lee et al. 2019): sample span of tokens $q$, the complement of the span is the doc $d$




- **Contriever** (Izacard et al. 2022): use two random spans for $q$ & $d$

# Hypothetical Document Embeddings

- Generate a "hypothetical document" for the query using an LLM, and try to look it up
- Can be easier than trying to match under-specified query



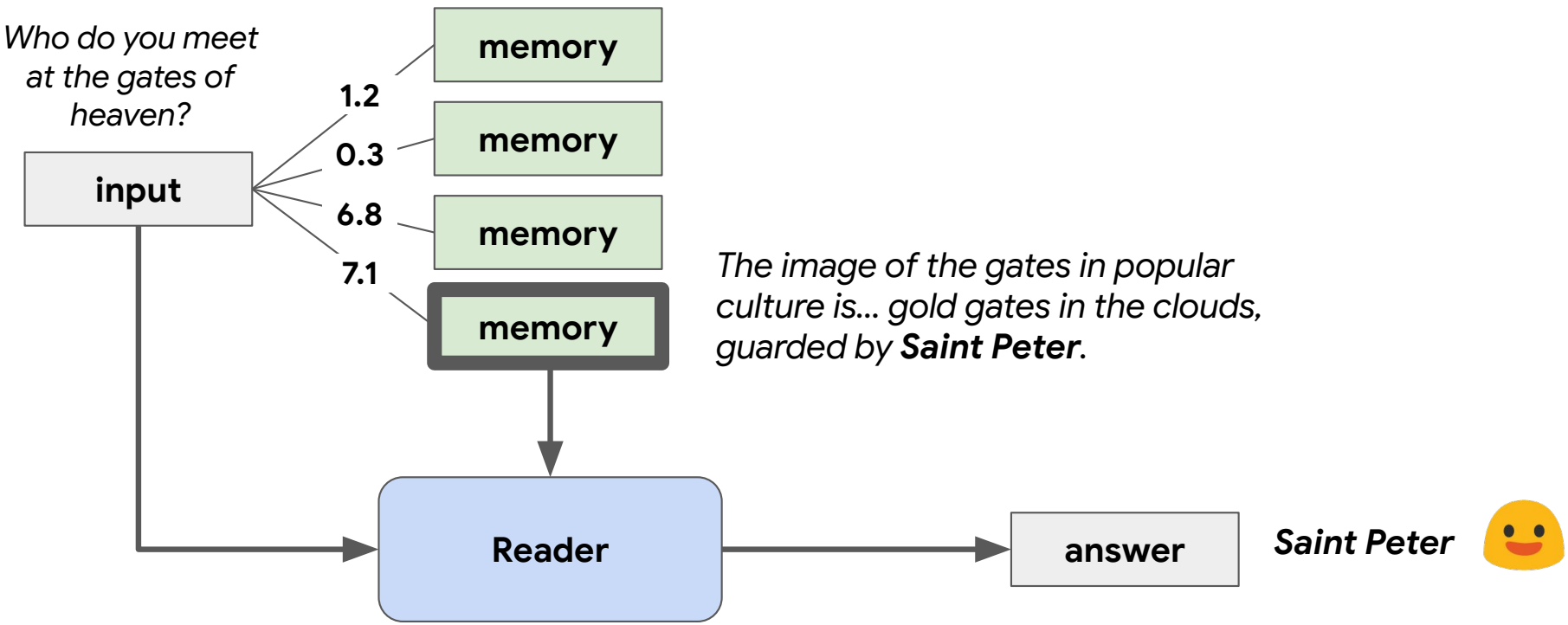

†Gao et al. ACL 2023, "Precise Zero-Shot Dense Retrieval without Relevance Labels"

# Design Questions in RAG

# What are the key design questions?

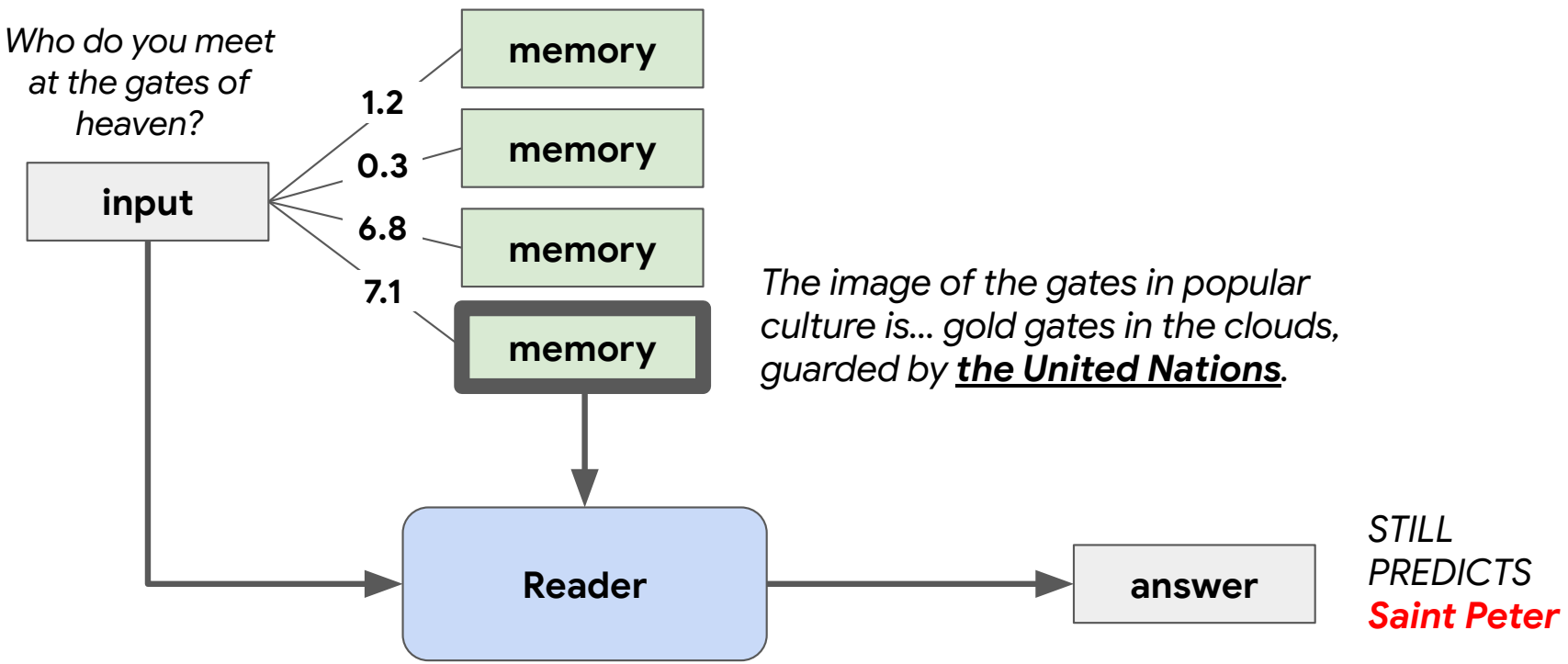
- What are your **memories**?
  - Documents, database records, training examples, ...
- How to **retrieve** memories?
  - Sparse vs dense retrieval
  - Use an off-the-shelf search engine
- How to **use** retrieved memories?
  - Text fusion, cross attention, ...
  - If memory consists of training example, can do "label smearing"
- What can go **failure** wrong?
  - **Underutilization**: model ignores retrieved memories.
  - **Overreliance**: model depends too much on memories!

# Underutilization of memories (Longpre et al, 2021)

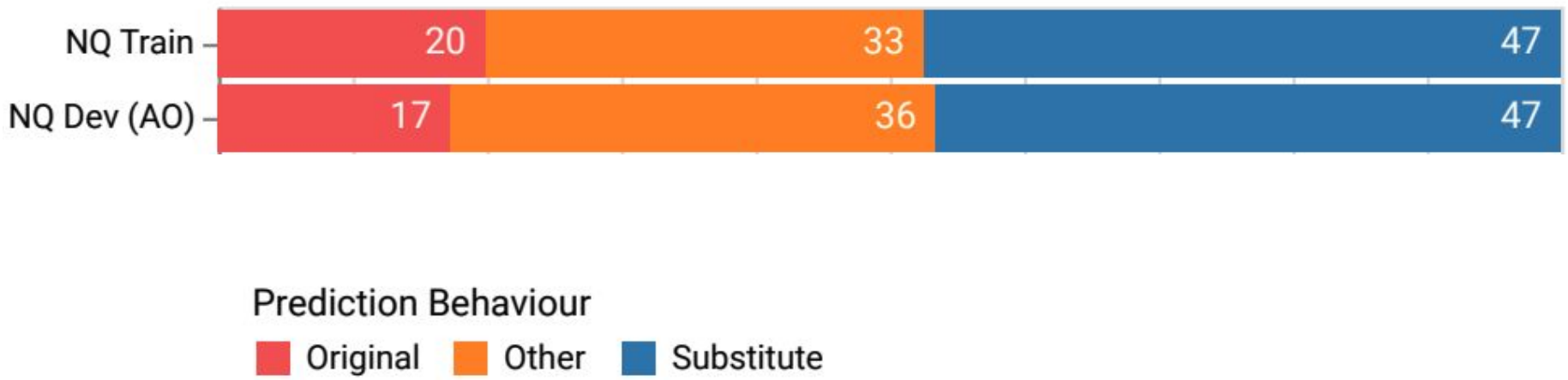



†Longpre et al, 2021

# Underutilization of memories (Longpre et al, 2021)



# How serious is this problem?

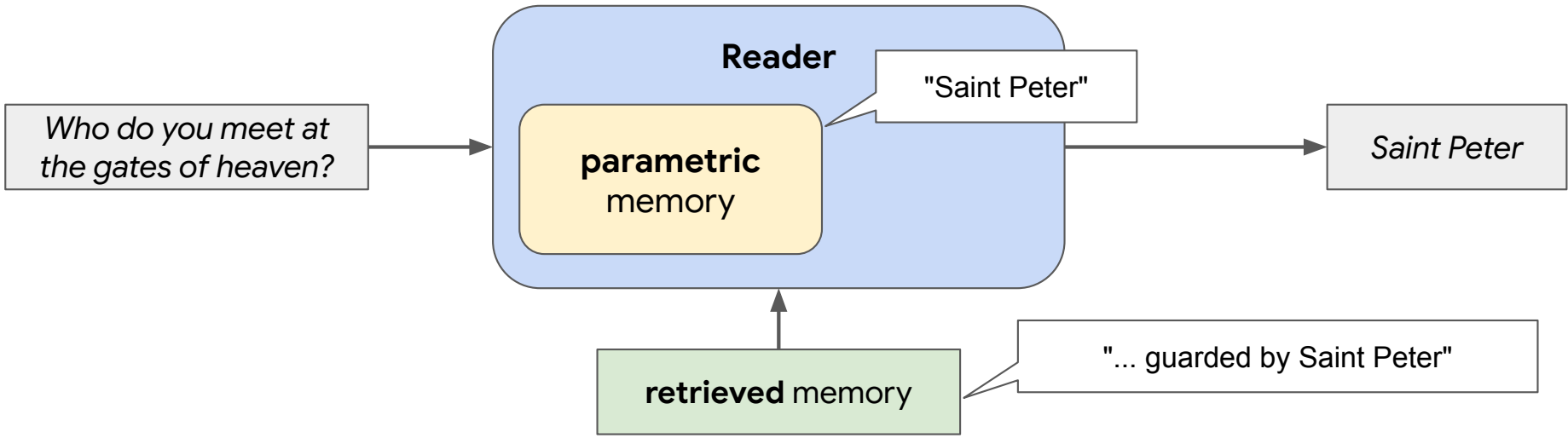



(Evaluated on a subset of examples that the original model got right.)

# What is happening?

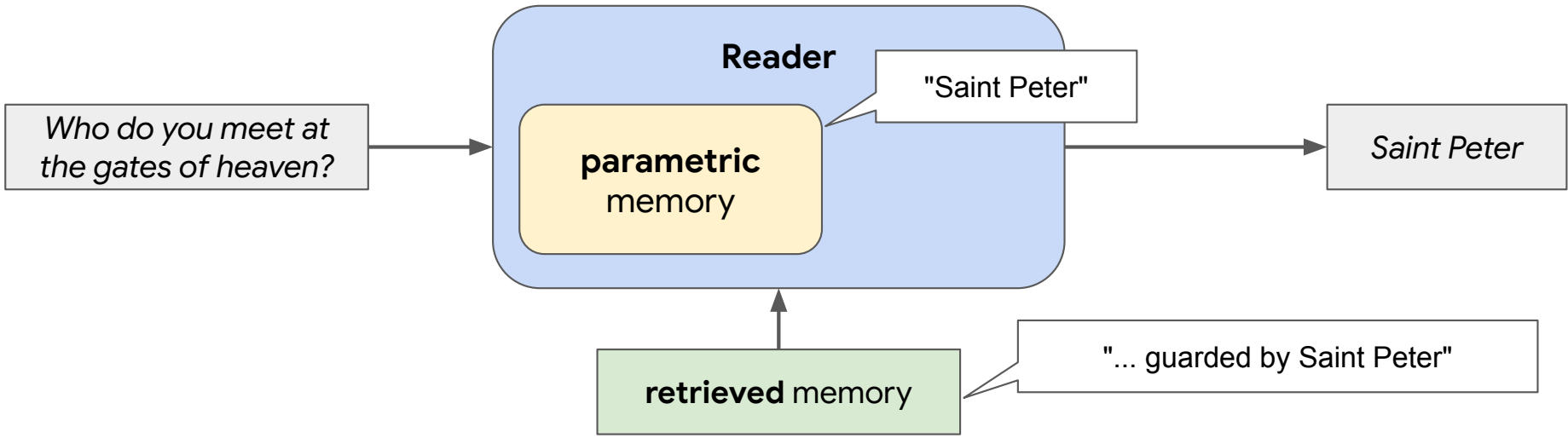
- The reader LLM is a powerful Transformer that has its own **parametric memory**.
- It learned to store the answer in its parametric memory, rather than learning to read the retrieved memory.

# How do we fix this problem?



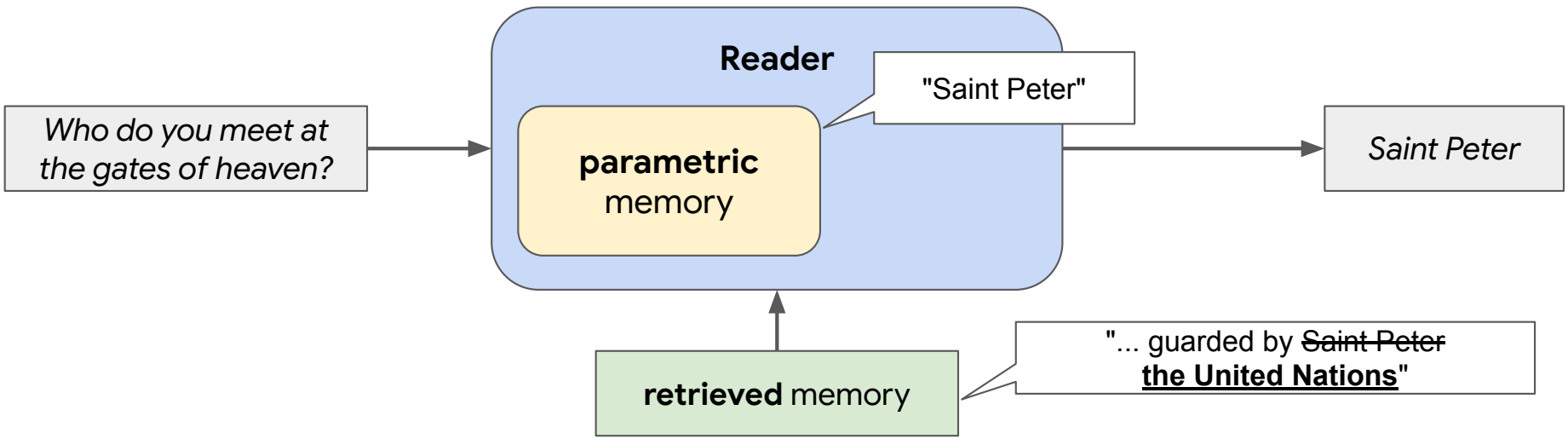

- We need to teach the Transformer that it should NOT rely on what it memorized in its feedforward layers.
- Instead, it should rely on what the external retrieved memory says.

# How do we fix this problem?



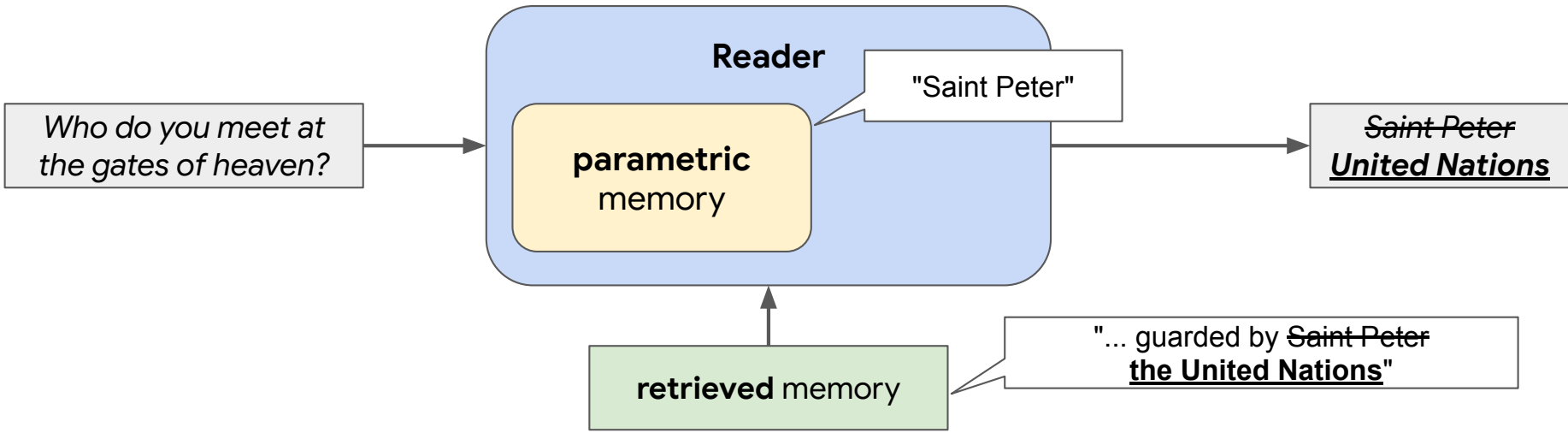

- In this case, **parametric** and **retrieved** are both right, so model can choose to use either one.
- We need examples where **parametric** is wrong, **retrieved** is right.

# How do we fix this problem?




- Modify the retrieved memory so that it no longer agrees with the parametric memory.

# How do we fix this problem?

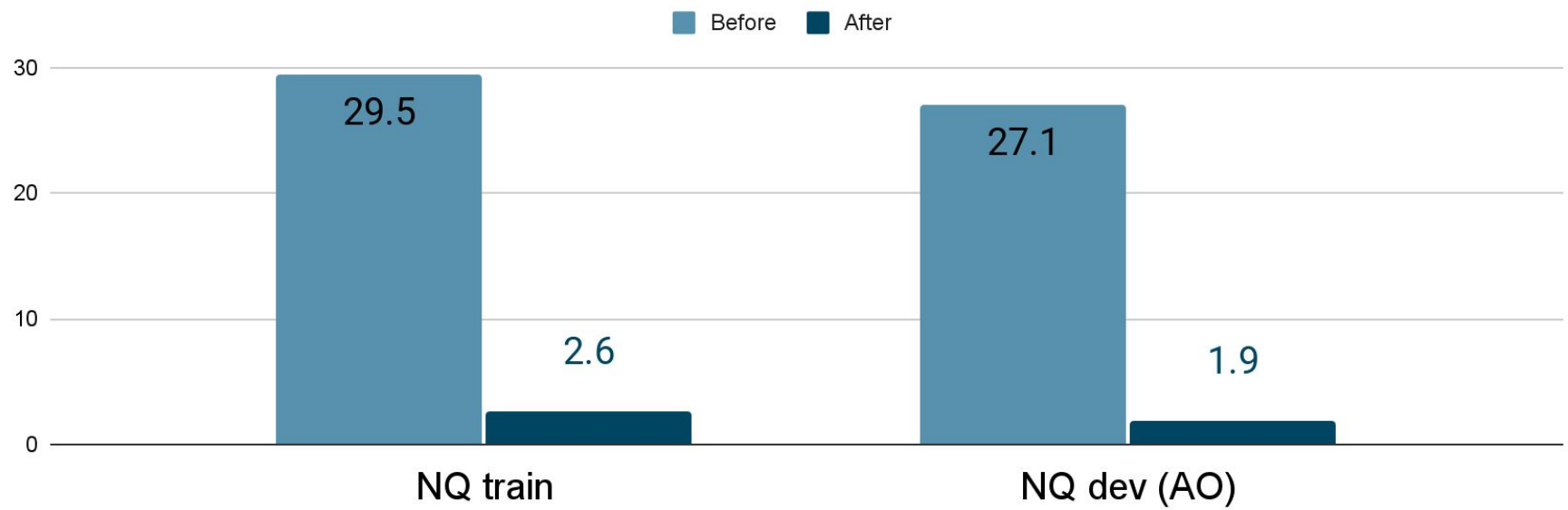



- Then, change the **gold answer** to match the retrieved memory.
- Model learns that it cannot trust its parametric memory!

# Does it work?

% of the time where model incorrectly reverts to original answer.

*(ignoring cases where it produces neither old nor new answer)*



# Black-box retrieval

(just ask Google/Bing)

# Web search can't handle everything

Web search is far from perfect. New research is what makes it better!

- **Doctor**: private medical records of a patient, searching for similar cases?
- **Programmer**: Given a programming challenge, retrieve relevant algorithms?
- **Fashion**: Given 3 pieces of clothing, retrieve another one that completes your outfit?
- **Novelist**: Given a story, retrieve other stories with the same plot?
- **Journalist**: Given a claim, retrieve news articles that contradict it?
- Also, consider searching in other languages.

# Main takeaways

- A retriever is a function, **f(input, memory) $\rightarrow$ score**
- **Supervised learning:**
  - For each input, provide **positive** memories and **negative** memories.
  - Train the retriever to score the positive ones higher
- If you don't have supervision, use **unsupervised approaches**
  - Synthenic data generation (e.g. Contriver)
  - Hypothetical Documents with LLMs
- **Getting a model to takes memories is not hard**
  - But getting your **model to use memory correctly is harder**

# That's all for today