CS11-711 Advanced NLP

Retrieval and Retrieval-Augmented Generation

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/anlp2022/

Standard Prompting

- Combine a prompt template together with an input

Please answer this question:

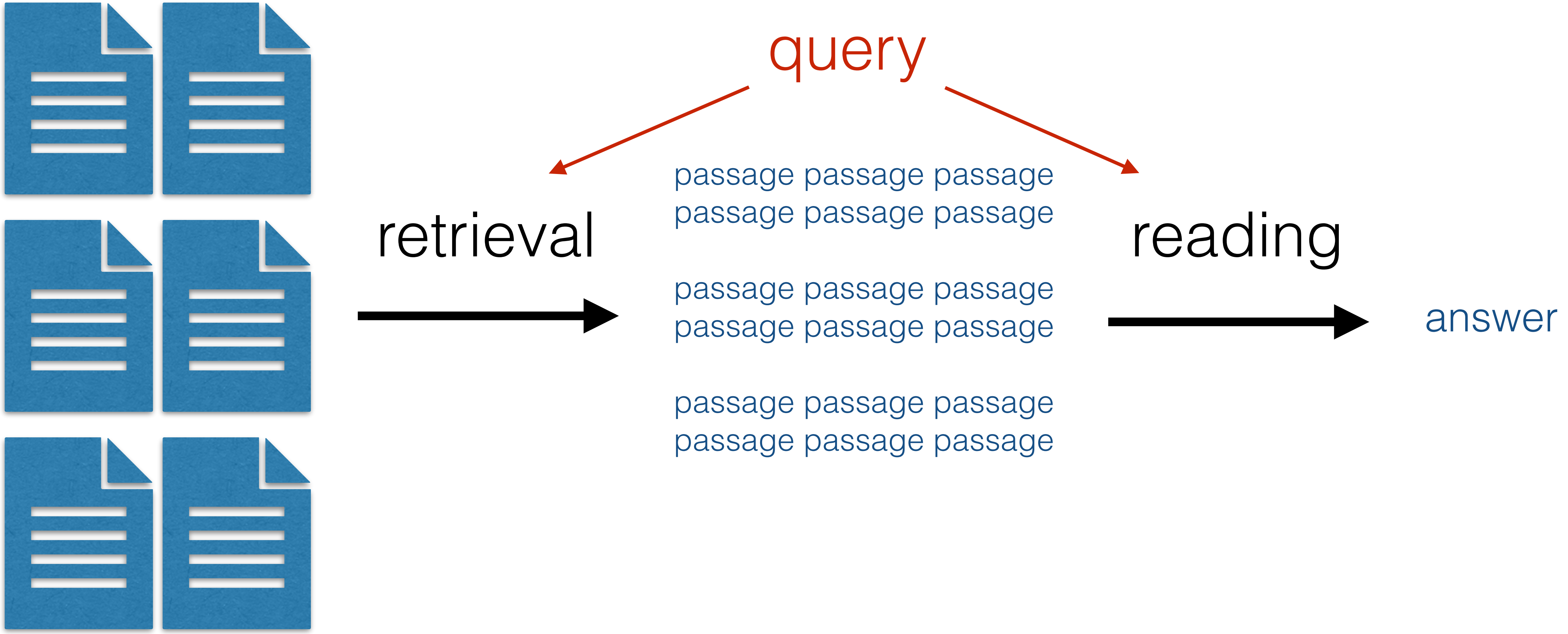
I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are?

Problems

- *Accuracy issues:*
  - **Knowledge cutoffs:** parameters are usually only updated to a particular time
  - **Private data:** data stored in private text or data repositories not suitable for training
  - **Learning failures:** even for data that the model was trained on, it might not be sufficient to get the right answer
- *Verifiability issues:* It is hard to tell if the answer is correct

Retrieval-augmented Generation
(Chen et al. 2017)

- **Retrieve** relevant passages efficiently
- **Read** the passages to answer the query



Example

You

I think Vin Diesel has been a voice actor for several characters in TV series, do you know what their names are? Please list all of them.

ChatGPT

Vin Diesel has lent his voice to several characters across various TV series and shows. Here's a list of the characters he's voiced:

1. **Groot** in "Guardians of the Galaxy Vol. 3" (2023 Movie), "The Guardians of the Galaxy Holiday Special" (2022 TV Show), "I Am Groot" (2022 Short), "Thor: Love and Thunder" (2022 Movie), "Avengers: Endgame" (2019 Movie), "Ralph Breaks the Internet: Wreck-It Ralph 2" (2018 Movie), "Avengers: Infinity War" (2018 Movie), "Guardians of the Galaxy Vol. 2" (2017 Movie), and "Guardians of the Galaxy" (2014 Movie).
2. **Santiago** in "ARK: The Animated Series" (2023 TV Show).
3. **Dominic Toretto** in "Fast & Furious: Spy Racers" (2019 TV Show).

https://www.behindthevoiceactors.com/Vin-Diesel/

Additionally, though the website for "Big Mouth" lists Vin Diesel, it appears to be a misunderstanding or an error, as Nick Kroll is credited as the voice of Vin Diesel in that show.

https://www.behindthevoiceactors.com/tv-shows/Big-Mouth/Vin-Diesel/

These roles showcase Diesel's versatility and contribution to voice acting, particularly in animated series and specials, where he brings iconic characters to life with his distinctive voice.

Retrieval Methods

Retrieval Methods

- **Sparse retrieval**
- **Document-level dense retrieval**
- **Token-level dense retrieval**
- **Cross-encoder reranking**
- **Black-box retrieval** (just ask Google/Bing)

Sparse Retrieval

- Express the query and document as a sparse word frequency vector (usually normalized by length)

q=what is nlp

$d_1 =$ what is life ? candy is life !

$d_2 =$ nlp is an acronym for natural language processing

$d_3 =$ I like to do good research on nlp

$$\begin{array}{c} \text{what} \\ \text{candy} \\ \text{nlp} \\ \text{is} \\ \text{language} \\ \ldots \end{array} \begin{pmatrix} 0.33 \\ 0 \\ 0.33 \\ 0.33 \\ 0 \\ \ldots \end{pmatrix} \quad \begin{pmatrix} 0.25 \\ 0.125 \\ 0 \\ 0.25 \\ 0 \\ \ldots \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0.125 \\ 0.125 \\ 0 \\ \ldots \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0.125 \\ 0 \\ 0 \\ \ldots \end{pmatrix}$$

$q^*d_1 = 0.165$ $\quad q^*d_2 = 0.0825$ $\quad q^*d_3 = 0.0413$

- Find the document with the highest inner-product or cosine similarity in the document collection

Term Weighting
(See Manning et al. 2009)

- Some terms are more important than others; low-frequency words are often more important
- Term frequency - in-document frequency (TF-IDF)

$$\mathrm{TF}(t,d) = \frac{\mathrm{freq}(t,d)}{\sum_{t'} \mathrm{freq}(t',d)} \qquad \mathrm{IDF}(t) = \log\left(\frac{|D|}{\sum_{d' \in D} \delta(\mathrm{freq}(t,d') > 0)}\right)$$

$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

- BM25: TF term similar to smoothed count-based LMS

$$\text{BM-25}(t,d) = \mathrm{IDF}(t) \cdot \frac{\mathrm{freq}(t,d) \cdot (k_1 + 1)}{\mathrm{freq}(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\mathrm{avgdl}}\right)}$$

Inverted Index

- A data structure that allows for efficient sparse lookup of vectors

Sparse Vectors

$$
\begin{array}{r ccc}
 & d_1 & d_2 & d_3 \\
\begin{array}{r} \text{what} \\ \text{candy} \\ \text{nlp} \\ \text{is} \\ \text{language} \\ \end{array} &
\begin{pmatrix} 2 \\ 1 \\ 0 \\ 2 \\ 0 \\ \vdots \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ \vdots \end{pmatrix} &
\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}
\end{array}
\rightarrow
$$

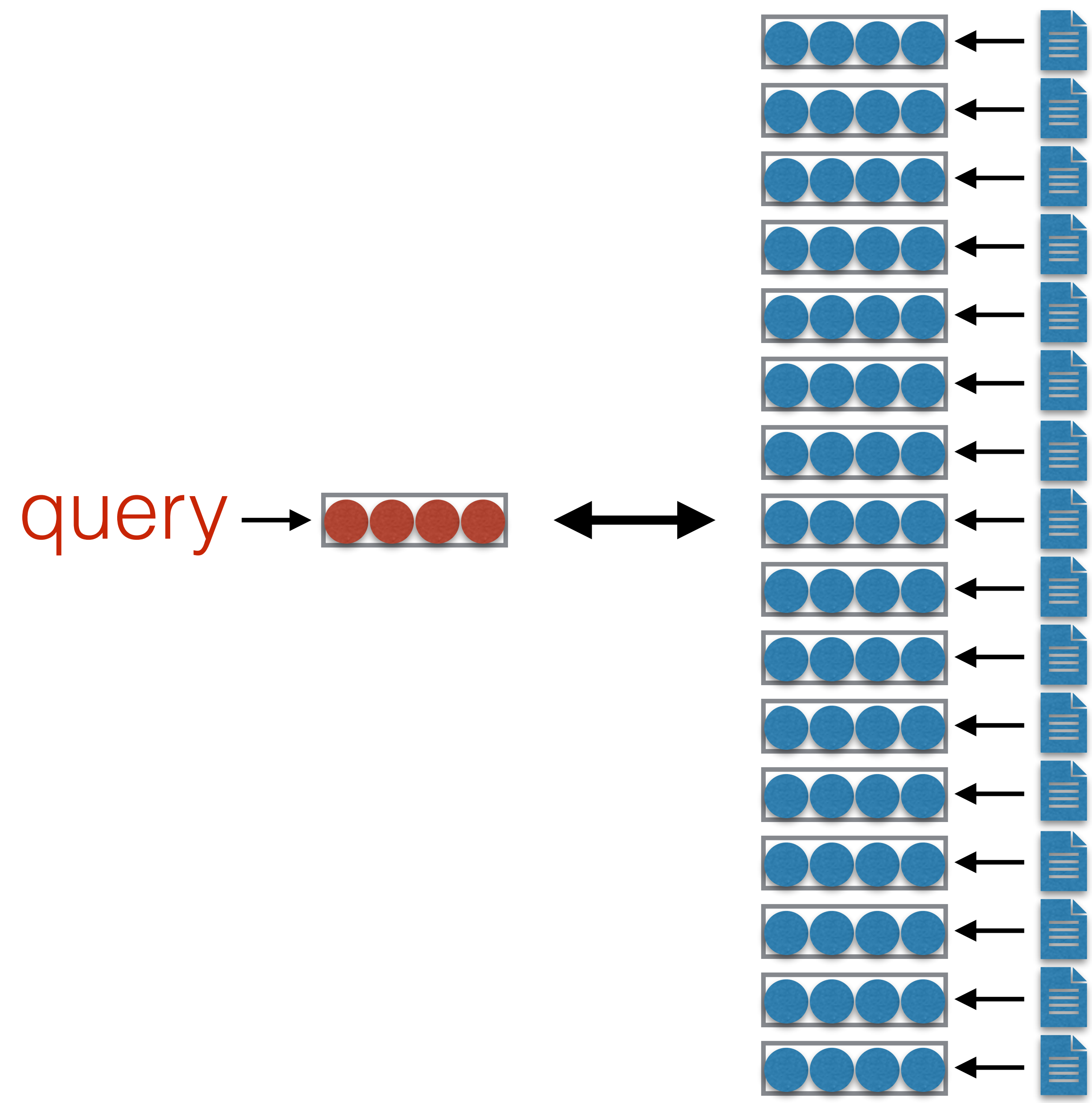
Index

```
{
  "what": [1],
  "candy": [1],
  "nlp": [2,3],
  "is": [1,2],
  "language": [2],
  …
}
```

- Example software: Apache Lucene

Dense Retrieval

- Encode document/query and find nearest neighbor
- Can use:
  - Out-of-the-box embeddings
  - Learned embeddings



Learning Retrieval-oriented Embeddings

- Select positive and negative documents, train using a contrastive loss (e.g. hinge loss)

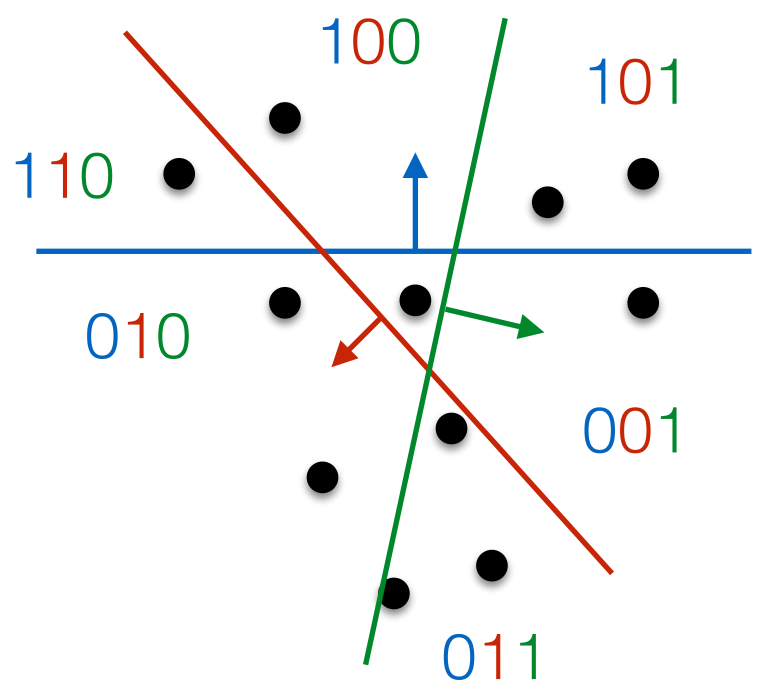
$$\mathcal{L}(\theta, q) = \sum_{d_{\text{pos}} \in D_{\text{pos}}} \sum_{d_{\text{neg}} \in D_{\text{neg}}} \max(0, s(q, d_{\text{neg}}; \theta) - s(q, d_{\text{pos}}; \theta))$$

- **DPR** (Karpukhin et al. 2020): learn encoders based on a BM25 hard negatives and in-batch negatives.
- **Contriever** (Izacard et al. 2022): contrastive learning using two random spans as positive pairs

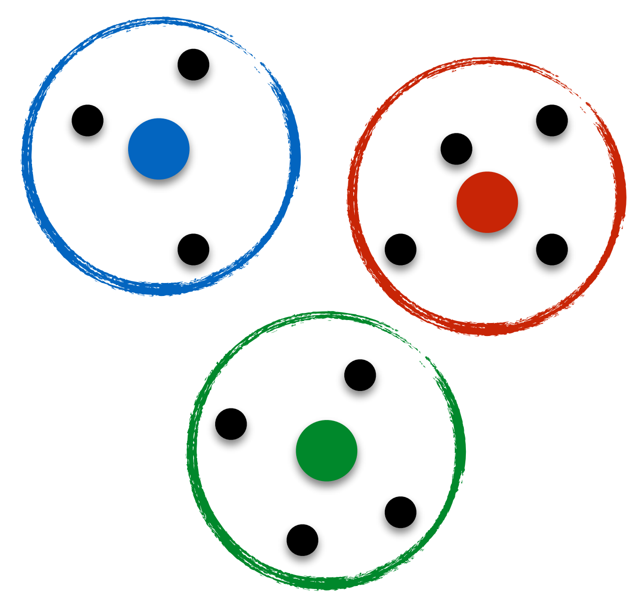
Approximate Nearest Neighbor Search

- Methods to retrieve embeddings in sub-linear time

Locality sensitive hashing: make partitions in continuous space, use like inverted index



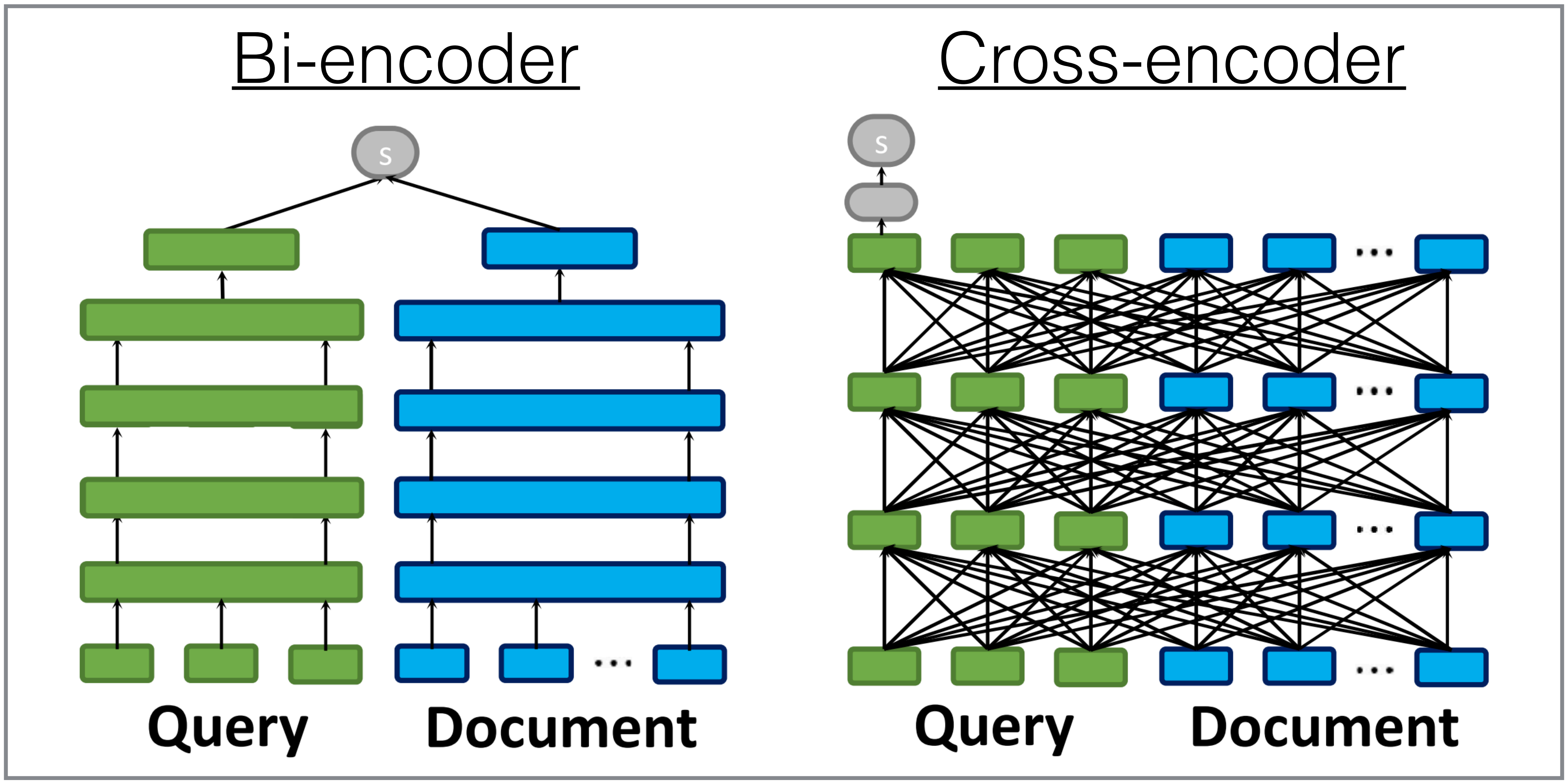
Graph-based search: create "hubs" and search from there



- Software: FAISS, ChromaDB

Cross-encoder Reranking

- Jointly encode both queries and documents using neural model (Nogueira et al. 2019)

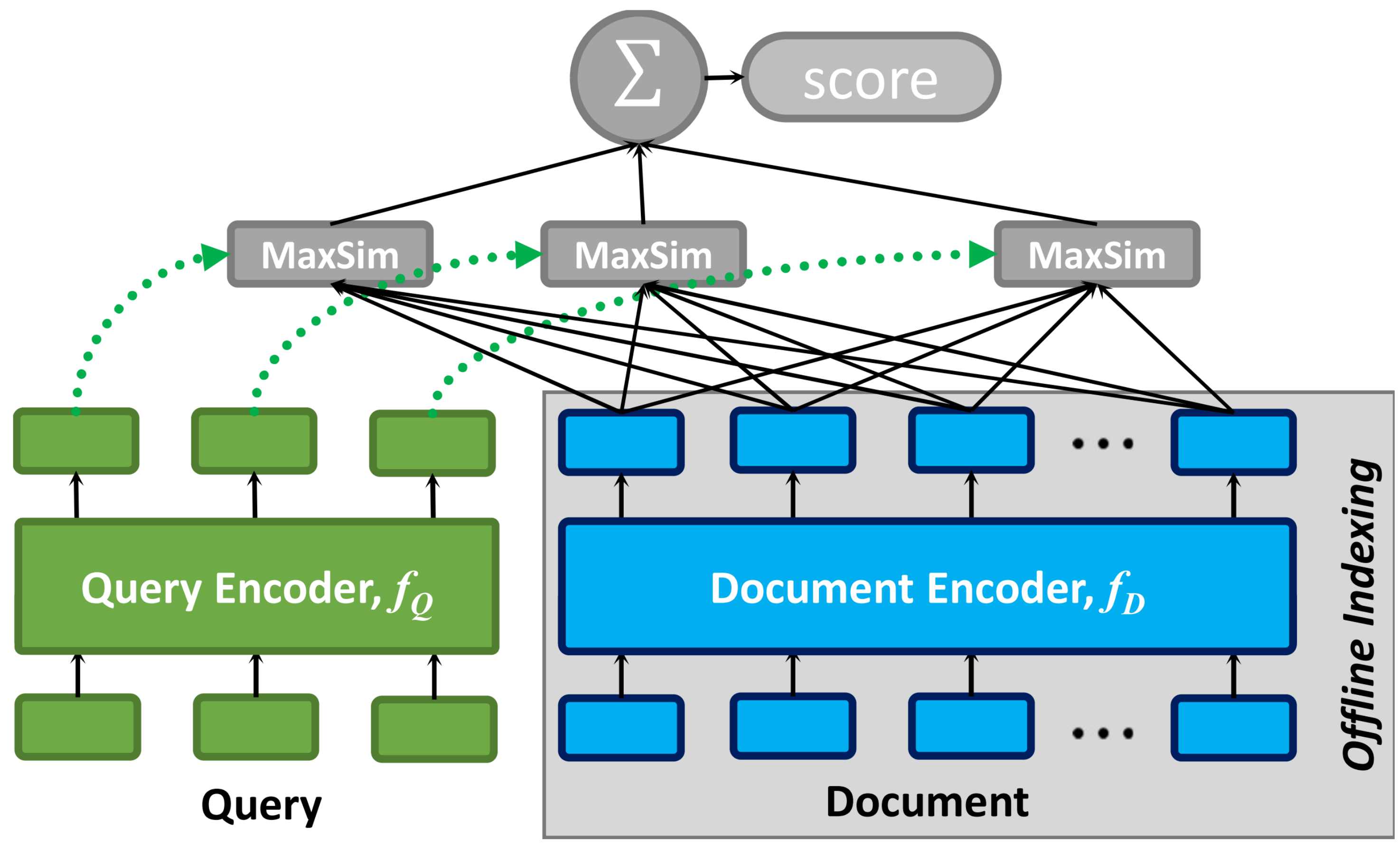


- Precludes approximate nearest neighbor lookup, so can only be used on small number of candidates

Figure from Khattab et al. (2020)

Token-level Dense Retrieval

- ColBERT (Khattab et al. 2020) use contextual representations of all query and document tokens to compute retrieval score.

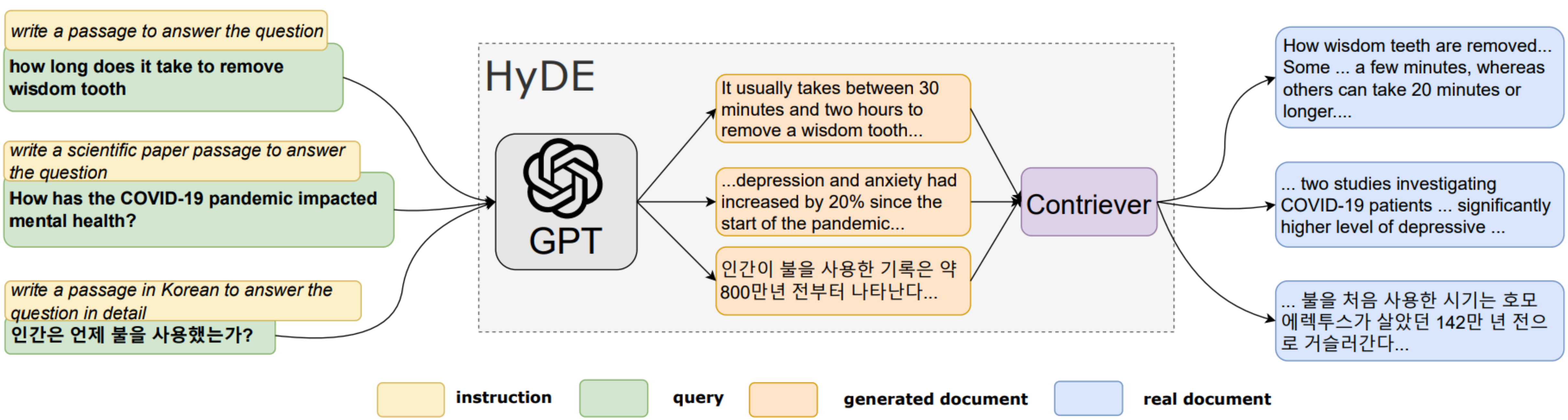


- Significantly more effective (but more costly) than single-vector retrieval

Hypothetical Document Embeddings

(Gao et al. 2022)

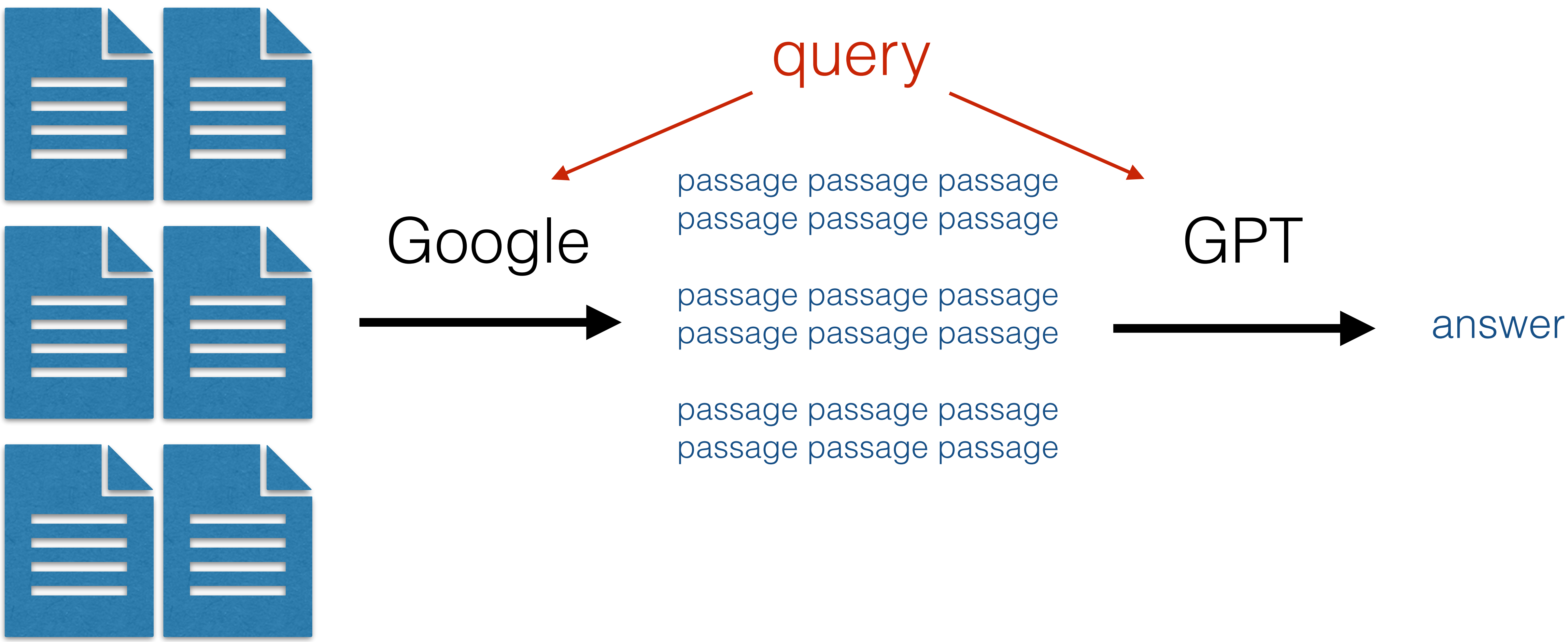
- Generate a “hypothetical document” for the query using an LLM, and try to look it up
- Can be easier than trying to match under-specified query



Retriever-Reader Models

Simple: Just Chain Retrieval+Reading

- Use an out-of-the-box retriever and out-of-the-box reader

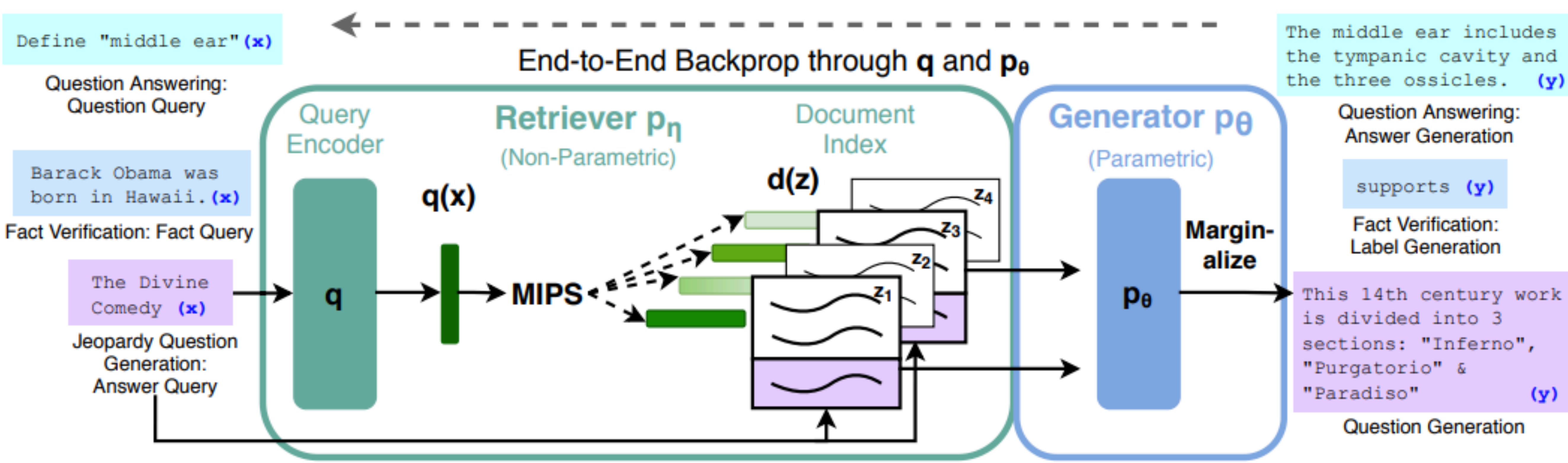


- Passages are concatenated to the context

Retriever + Generator End-to-end Training ("RAG")

(Lewis et al. 2020)

- Train the retriever and reader to improve accuracy
- **Reader:** Maximize generation likelihood given single retrieved document
- **Retriever:** Maximize overall likelihood by optimizing mixture weights over documents



End-to-end Training Equations

(Lewis et al. 2020)

- Generation is a mixture model: pick a document, generate from the document

$$P_{\text{RAG}}(y|x) \approx \prod_i \sum_{z \in \text{top-k}(p(\cdot|x))} \underbrace{p_\eta(z|x)}_{\text{Retriever}} \underbrace{p_\theta(y_i|x, z, y_{1:i-1})}_{\text{Generator}}$$

- Probability of the retriever is based on embeddings

$$p_\eta(z|x) \propto \exp(\mathbf{d}(z)^\top \mathbf{q}(x)) \quad \mathbf{d}(z) = \text{enc}_d(z), \quad \mathbf{q}(x) = \text{enc}_q(x)$$

- Adjusts retriever to give higher similarities helpful docs
- Issue: search index becomes stale → can only train q(x)

When do we Retrieve?

- **Once, at the beginning of generation**
  - Default method used by most systems (Lewis et al. 2020)
- **Several times during generation, as necessary**
  - Generate a search token (Schick et al. 2023)
  - Search when the model is uncertain (Jiang et al. 2023)
- **Every token**
  - Find similar final embeddings (Khandelwal et al. 2019)
  - Approximate attention with nearest neighbors (Bertsch et al. 2023)

Triggering Retrieval w/ Tokens

- Toolformer (Schick et al. 2023) generates tokens that trigger retrieval (or other tools)
- Training is done in an iterative manner - generate and identify successful retrievals

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.

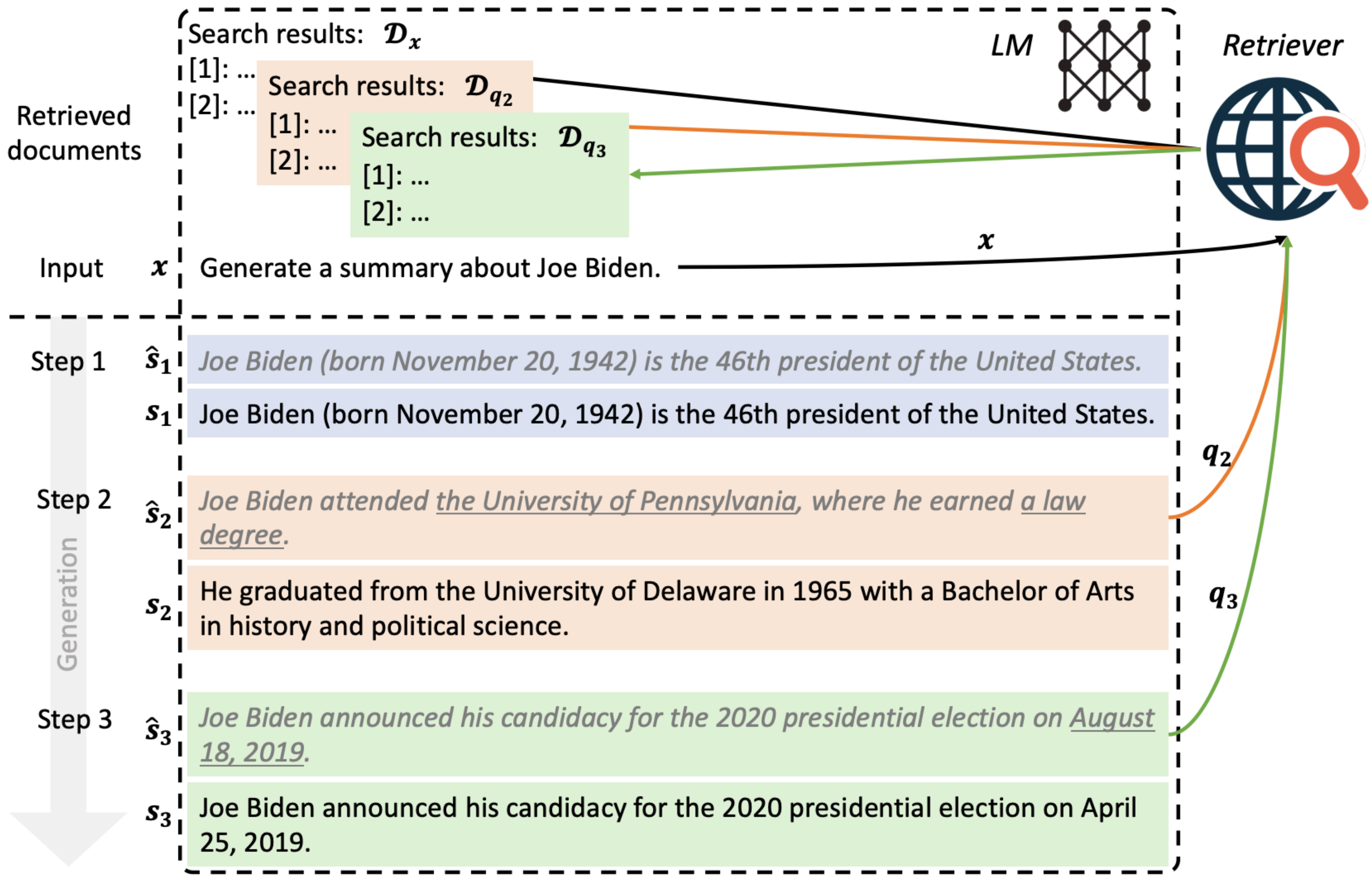
Out of 1400 participants, 400 (or **[Calculator(400 / 1400) → 0.29]** 29%) passed the test.

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 California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

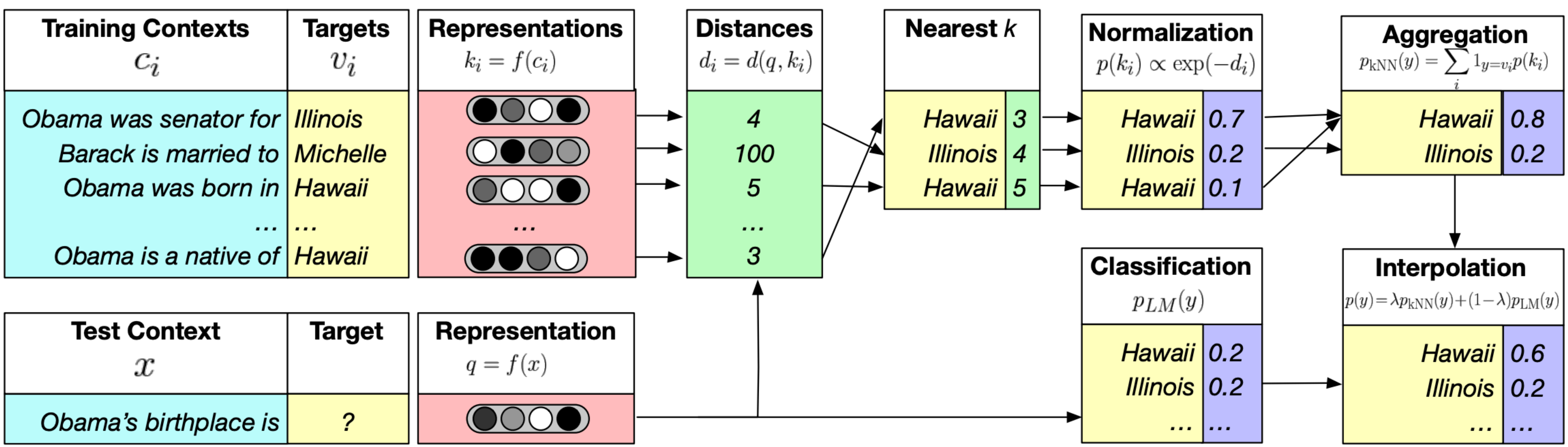
Triggering Retrieval w/ Uncertainty

- FLARE (Jiang et al. 2023) tries to generate content, then does retrieval if LM certainty is low



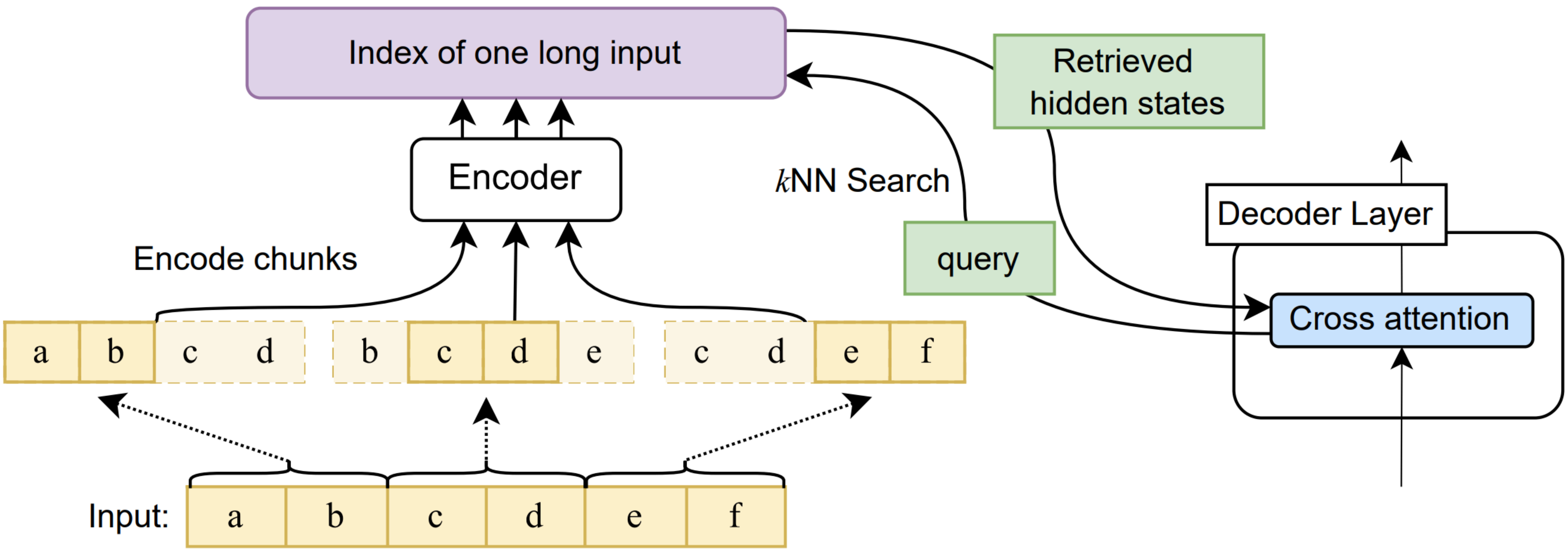
Token-level Softmax Modification

- kNN-LM (Khandelwal et al. 2019) retrieves similar examples, and uses the following token from them



Token-level Approximate Attention

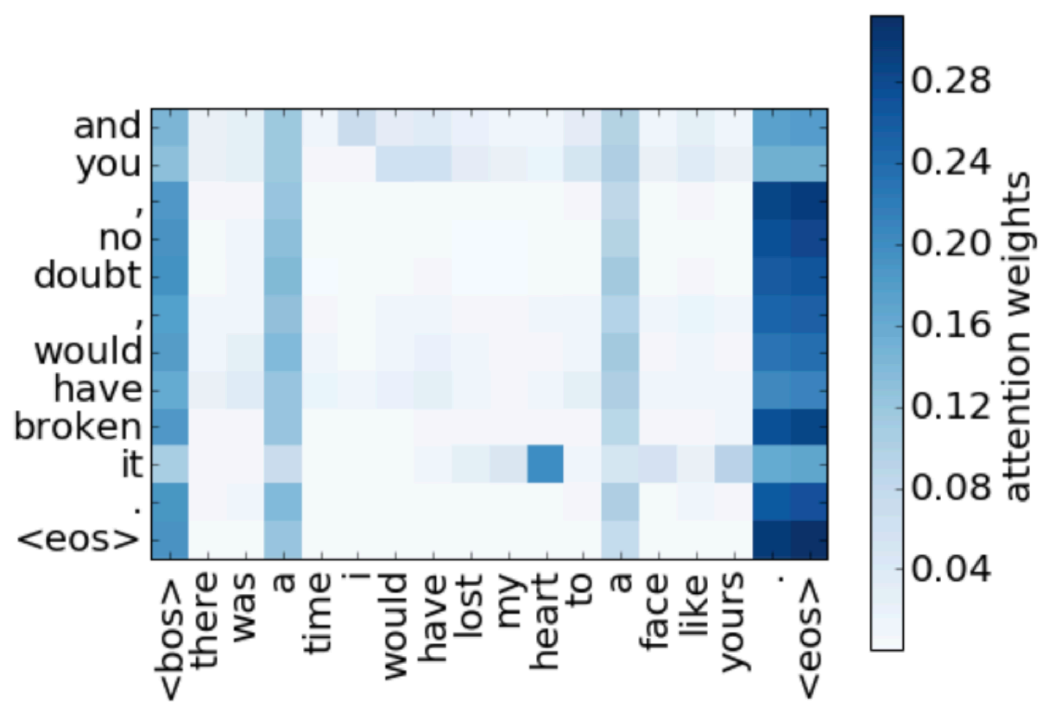
- Unlimiformer (Bertsch et al. 2023) notes that attention is an inner-product and does top-k attention
- First, process input with a sliding window
- Then perform attention using a vector index



Long-context Transformers

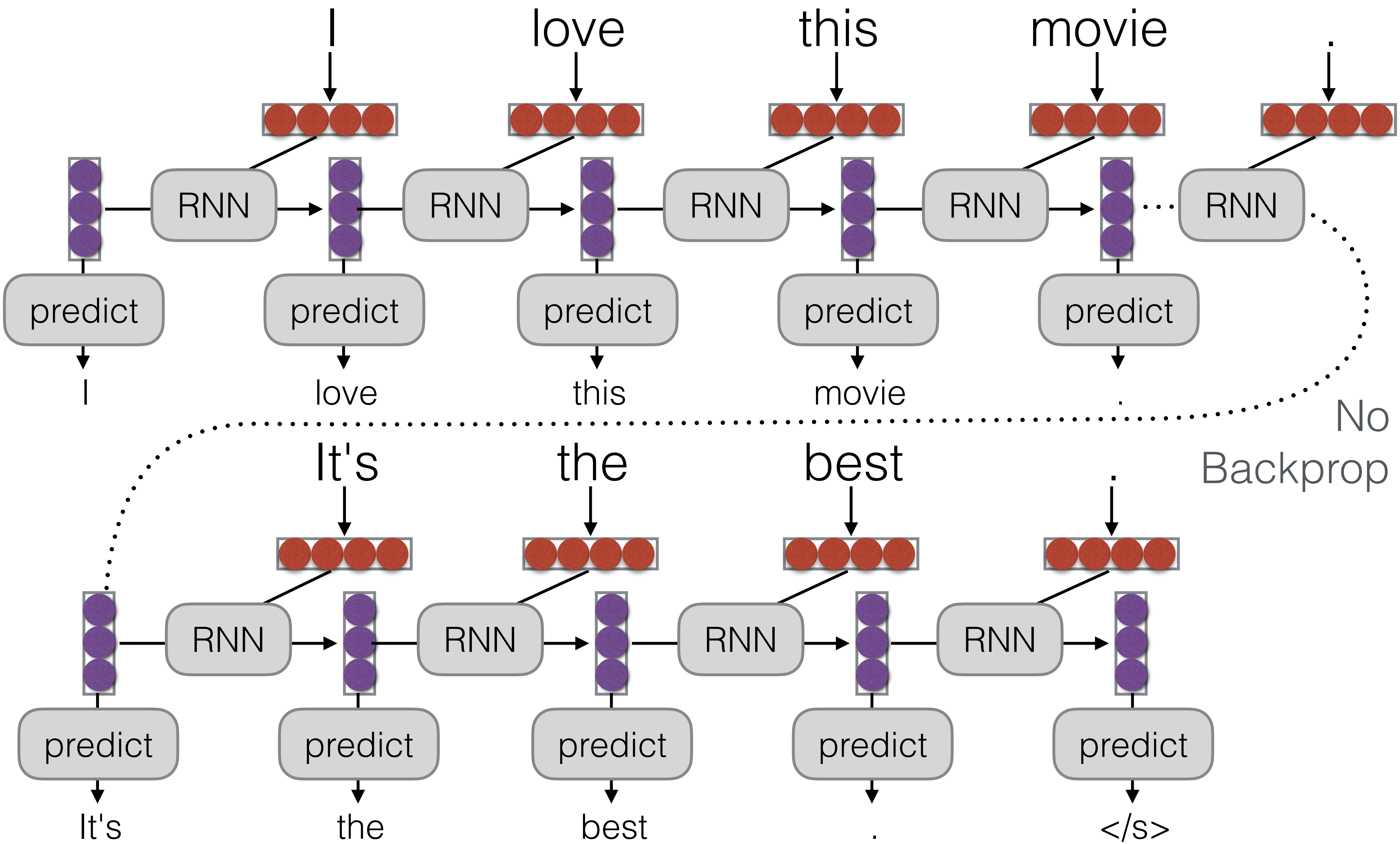
Training Transformers Over Longer Sequences

- Simply attend to all previous words in the document (e.g. Voita et al. 2018)
- + Can relatively simply use document-level context
- + Can learn interesting phenomena (e.g. co-reference)



- - Computation is quadratic in sequence length!

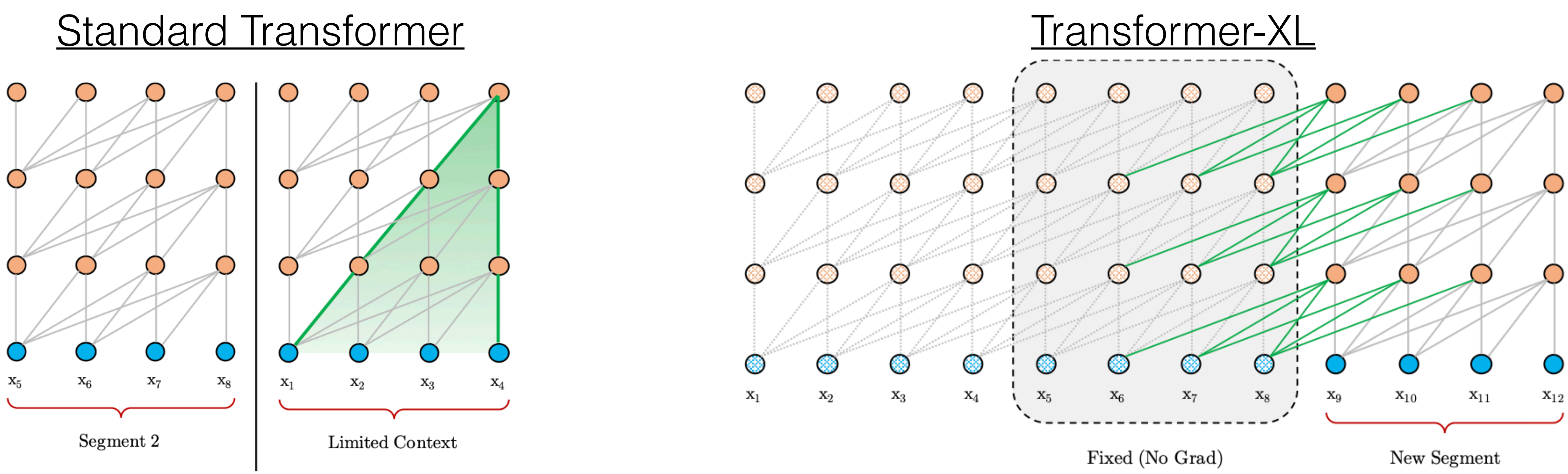
In RNNs: Pass State + Truncated Backprop



Recurrence is slow, improved by Mamba/RWKV (future class)

Truncated BPTT+Transformer

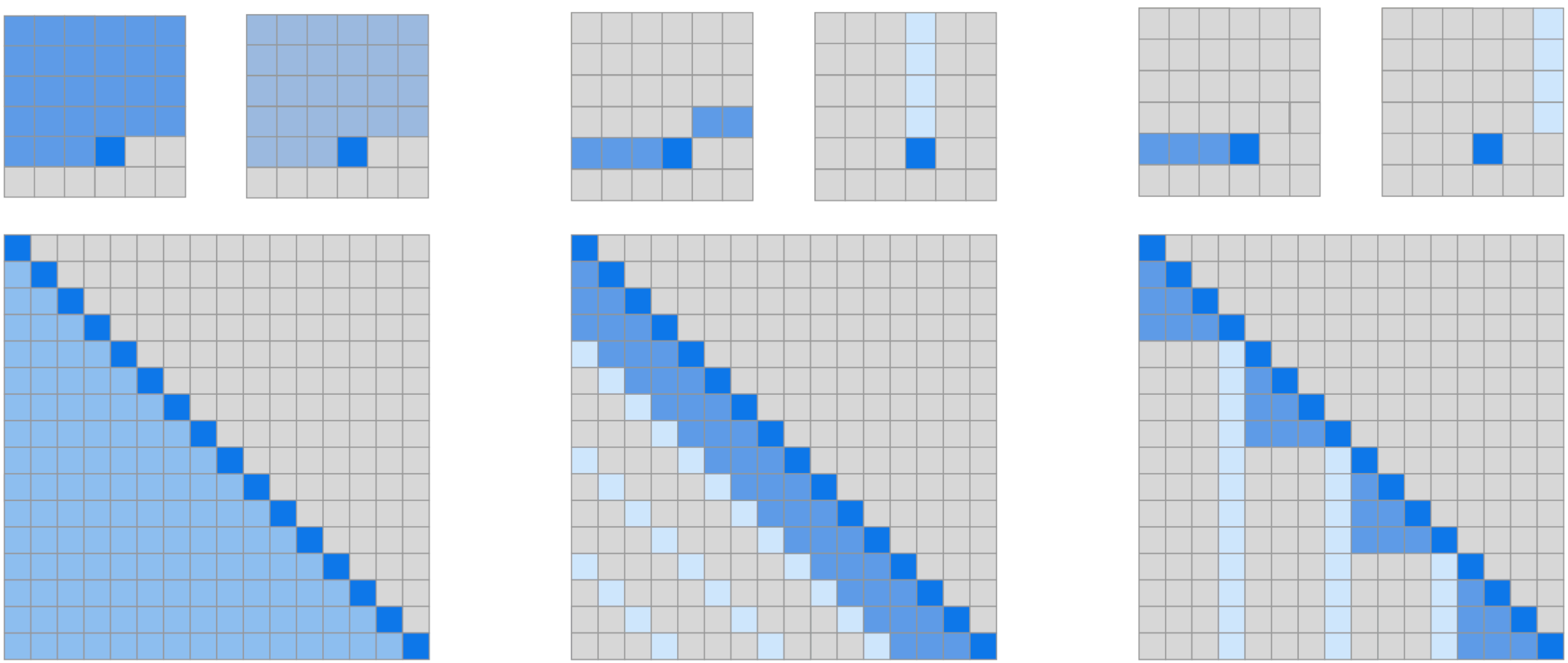
- Transformer-XL (Dai et al. 2019) attends to fixed **vectors** from the previous sentence



- Like truncated backprop through time for RNNs; can use previous states, but not backprop into them
- See also Mistral's (Jiang et al. 2023) sliding window attention

Sparse Transformers
(Child et al. 2019)

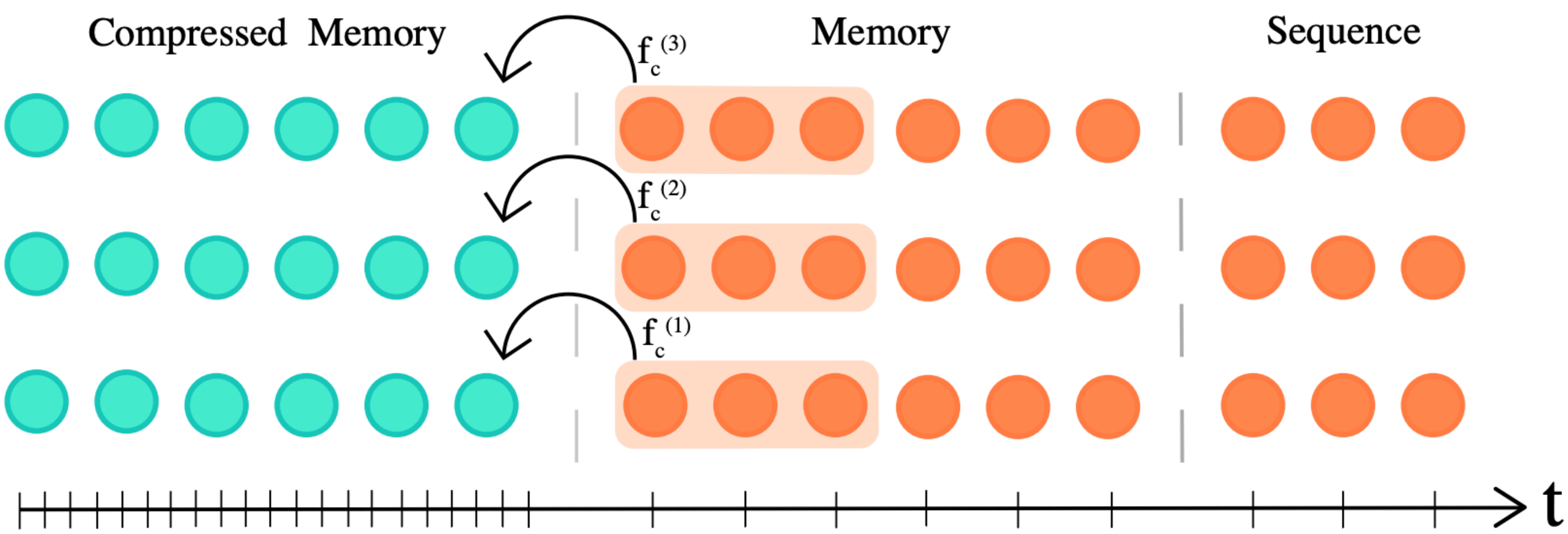
- Add "stride", only attending to every *n* previous states



(a) Transformer (b) Sparse Transformer (strided) (c) Sparse Transformer (fixed)

Compressing Previous States

- Add a "strided" compression step over previous states (Rae et al. 2019)

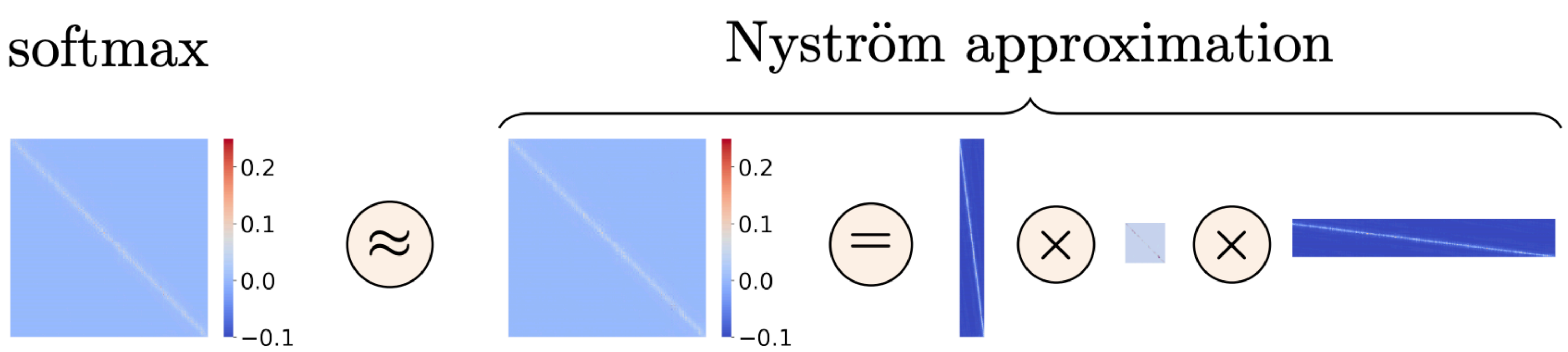


Low-rank Approximation

- Calculating the attention matrix is expensive, can it be predicted with a low-rank matrix?

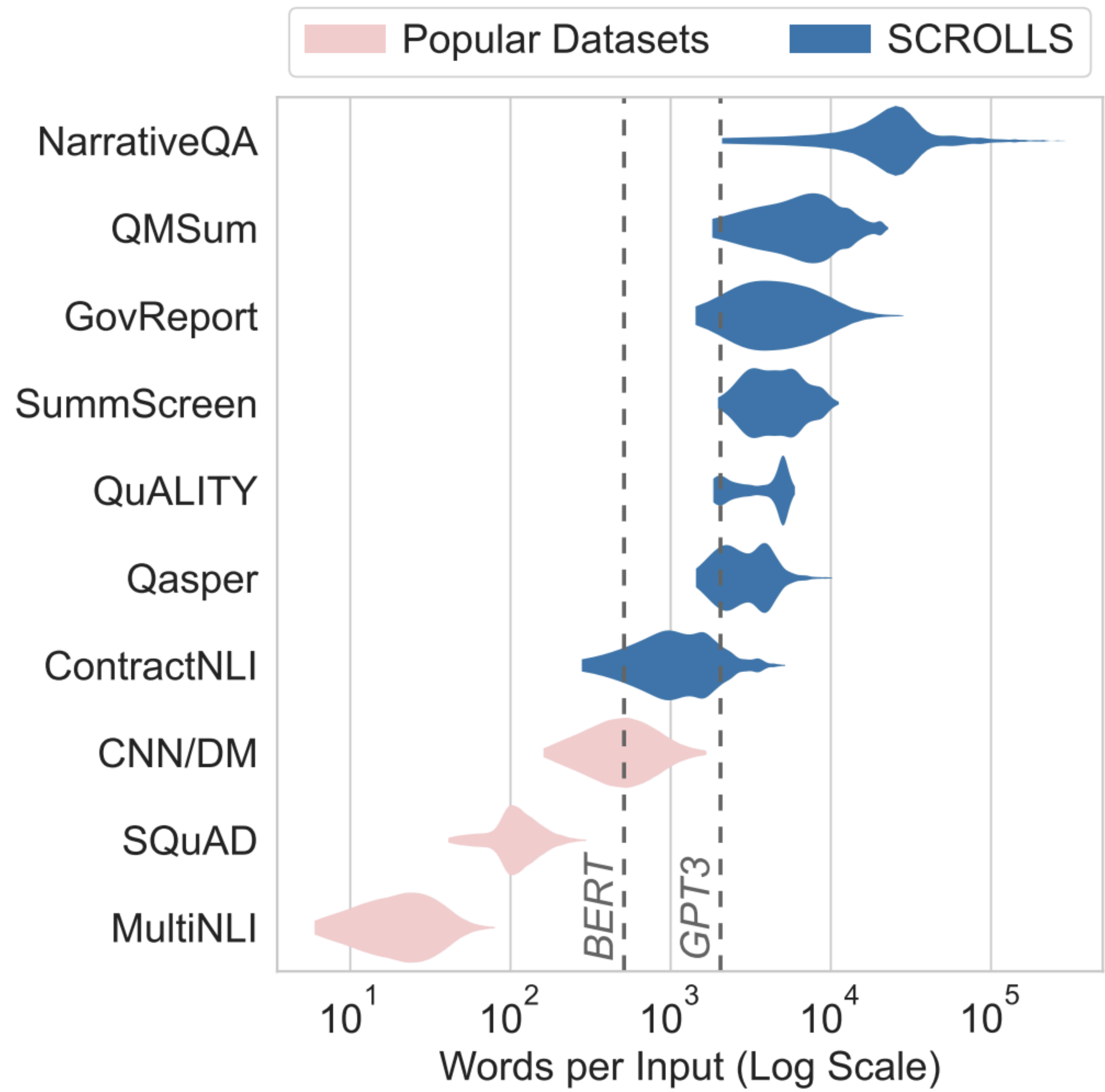
- **Linformer:** Add low-rank linear projections into model (Wang et al. 2020)

- **Nystromformer:** Approximate using the Nystrom method, sampling "landmark" points (Xiong et al. 2021)



Benchmarks for Long-context Models

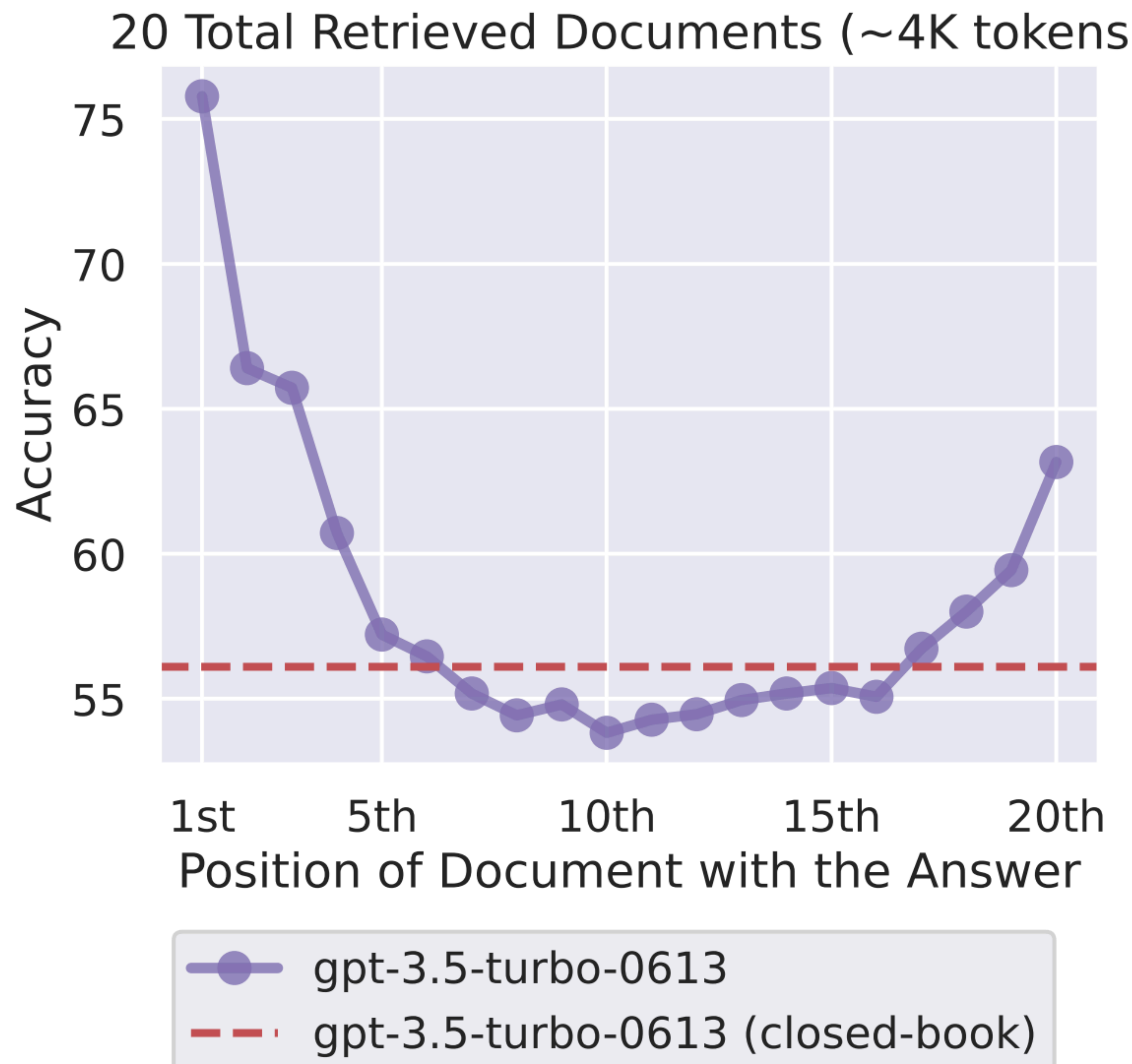
- **Long Range Arena:** Composite benchmark containing mostly non-NLP tasks (Tay et al. 2020)
- **SCROLLS:** Benchmark containing long-context summarization, QA, etc. (Shaham et al. 2022)



Effectively Using Long Contexts

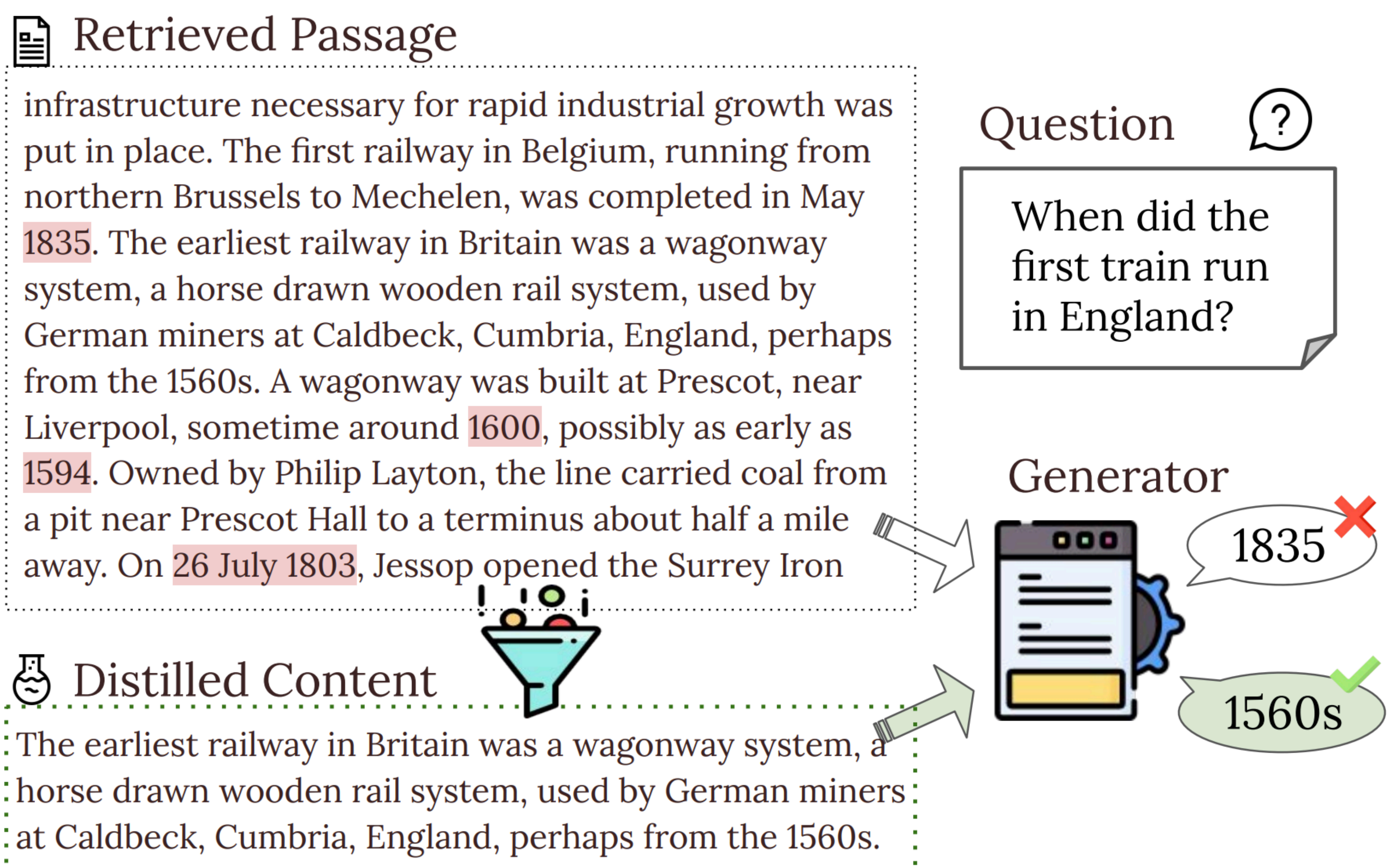
As Context Increases, Models Miss Relevant Info

- e.g. “lost-in-the-middle” (Liu et al. 2023) demonstrates that models pay less attention to things in the middle of context windows



Ensuring Use of Relevant Context

- Better retrievers make more relevant context
- Decide whether to include passages (Asai et al. 2021)
- Filter down to parts of retrieved passages (Wang et al. 2023)



Questions?