JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Feeding Lots Data to Language Models

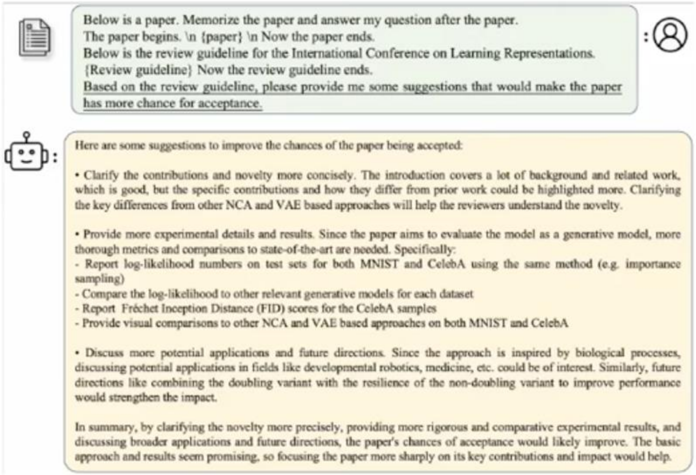
CSCI 601-771 (NLP: Self-Supervised Models)

https://self-supervised.cs.jhu.edu/sp2025/

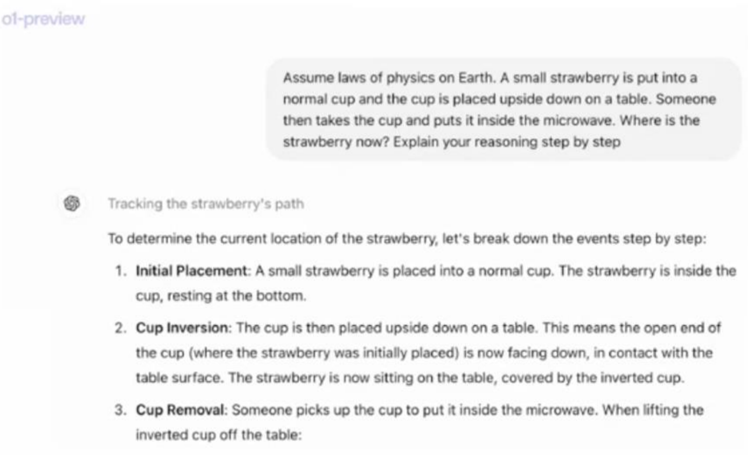
Long inputs/outputs in LM

- Books, scientific articles, government reports, videos, your daily experience, etc. they all are much longer than 10k tokens!!

Long Documents



Inference-time Reasoning



Nominal Context Length

Model	Creators	Year	Model Size	Context Size
GPT-2	OpenAI	2019	1.5 billion	1024
GPT-3	OpenAI	2020	175 billion	2048
PaLM	Google	2022	540 billion	2048
LLaMA	Meta	2023	65 billion	2048
LLaMA-2	Meta	2023	70 billion	4096
Claude-2	Anthropic	2023	? (130 billion)	100k
Claude-2.1	Anthropic	2023	? (130 billion)	200k
GPT-4	OpenAI	2023	? (1.76 trillion)	8192
Mistral	Mistral AI	2023	7 billion	8192 (32k)
Mixtral	Mistral AI	2023	47 billion	8192 (128k)
Gemini (Ultra)	Google	2023	? (1.5 trillion)	32k
LWModel	academia!	2023	7 billion	1 million
Gemini-1.5	Google	2024	? (1.5 trillion)	1 million
GPT-4o	OpenAI	2024	?	128k
LLaMA-3	Meta	2024	405 billion	8192
LLaMA-3.1	Meta	2024	405 billion	128k
Claude-3.5	Anthropic	2024	?	200k

Revisiting

Encoding Positional Information

Recap: Self-Attention

- Given input $\mathbf{x}$:

$$Q = \mathbf{W}^q \mathbf{x}$$
$$K = \mathbf{W}^k \mathbf{x}$$
$$V = \mathbf{W}^v \mathbf{x}$$
$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\alpha}\right)V$$

- Remember that, without positional encoding the input is just bag of words for self-attention.

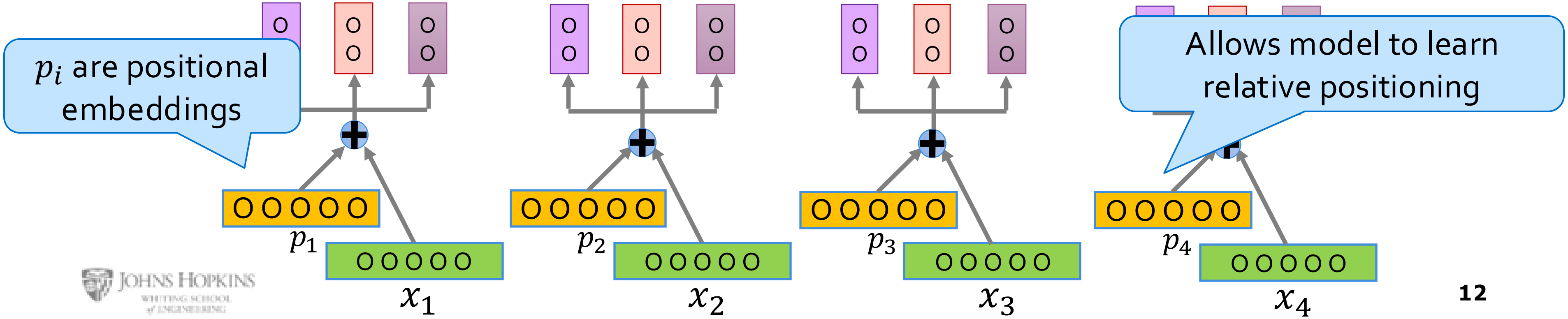
Positional Embeddings: The Flavors

- **Sine embeddings:** add sines and cosines that enable localization

Notable models: Original Transformer

$$Embed(x, i) = v_x + PE_{pos}$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Positional Embeddings: The Flavors

- **Sine embeddings:** add sines and cosines that enable localization

$$Embed(x, i) = v_x + PE_{pos}$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Notable models:
Original Transformer

- **Absolute embeddings:** add a position vector to the embedding

$$Embed(x, i) = v_x + u_i$$

Notable models:
GPT1/2/3 - OPT

- **Limitations:**
  - We can have fixed encoding for each index training position (e.g., 1, 2, 3, … 1000).
  - What happens if we get a sequence with 5000 words at test time?
- We want something that can generalize to <u>arbitrary</u> sequence lengths.

Positional Embeddings: The Flavors

- **Sine embeddings:** add sines and cosines that enable localization

$$Embed(x, i) = v_x + PE_{pos}$$

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Notable models: Original Transformer

- **Absolute embeddings:** add a position vector to the embedding

$$Embed(x, i) = v_x + u_i$$

Notable models: GPT1/2/3 - OPT

- **Relative embeddings:** add a vector to the attention computation

$$QK_{ij} = \boldsymbol{x}_i^T {W_q}^T W_k \boldsymbol{x}_j + \boldsymbol{P}_{ij}$$

Notable models: T5, Gopher, Chinchilla

  - Intuition: encoding the relative positions, for example based on the distance of the tokens in a local window to the current token.

Relative Positional Encoding: An Example

- *"I took the bus and then a cab after work"*
  - position of "bus": 4;
  - position of "cab": 8

- *"During the last two weeks, when my car was in the shop for repair, I took the bus and then a cab after work."*
  - position of "bus": 20;
  - position of "cab": 24

- You want to a way of encoding the sentence so that the attention patterns depend on **the relative distances**.

[Example credit: Christian von der Weth]

Relative Positional Encoding

- There have been various choices:
  - T5 models simplify this into learnable relative embeddings $\boldsymbol{P_{ij}}$ such that:

$$QK_{ij} = \boldsymbol{x_i^T} W_q{}^T W_k \boldsymbol{x_j} + \boldsymbol{P_{ij}}$$

  - DeBERTa learns relative positional embeddings $\tilde{\boldsymbol{p}}_{i-j}$ such that:

$$QK_{ij} = \boldsymbol{x_i^T} W_q{}^T W_k \boldsymbol{x_j} + \boldsymbol{x_i^T} W_q{}^T W_k \, \tilde{\boldsymbol{p}}_{i-j} + \tilde{\boldsymbol{p}}_{i-j}^T W_q{}^T W_k \boldsymbol{x_j}$$

  - Tranformer-XL learns relative positional embeddings $\tilde{\boldsymbol{p}}_{i-j}$ and trainable vectors $\boldsymbol{u}$, $\boldsymbol{v}$ s.t.:

$$QK_{ij} = \boldsymbol{x_i^T} W_q{}^T W_k \boldsymbol{x_j} + \boldsymbol{x_i^T} W_q{}^T W_k \, \tilde{\boldsymbol{p}}_{i-j} + \boldsymbol{u}^T W_q{}^T W_k \boldsymbol{x_j} + \boldsymbol{v}^T W_q{}^T W_k \, \tilde{\boldsymbol{p}}_{i-j}$$

  - ALiBi learns learns a scalar m such that:

$$QK_{ij} = \boldsymbol{x_i^T} W_q{}^T W_k \boldsymbol{x_j} - m \, |i-j|$$

Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, 2020
Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation (2022)

Recap

- **Sine embeddings:** add sines and cosines that enable localization

Notable models:
Original Transformer

- **Absolute embeddings:** add a position vector to the embedding

Notable models:
GPT1/2/3 - OPT

- **Relative embeddings:** add a vector to the attention computation

Notable models:
T5, Gopher,
Chinchilla, Deberta
Tranformer-XL,

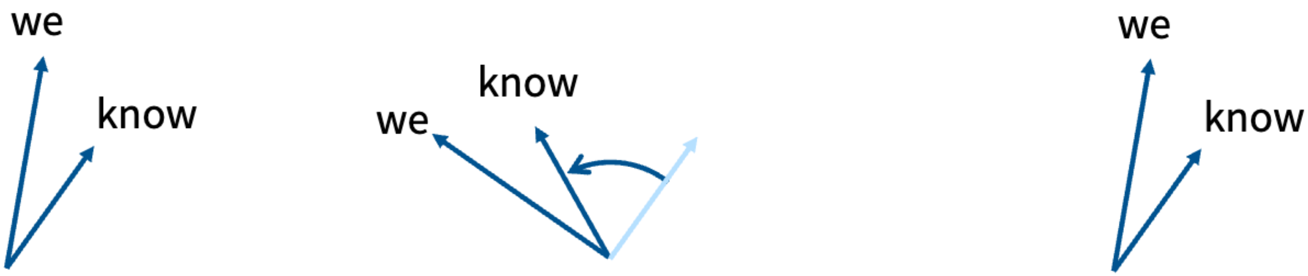
- **RoPE embeddings:** (forthcoming)

Notable models:
GPTJ, PaLM, LLaMA,
Gemma3, Qwen3,
GPL, Apple models.

Bonus

Rotary Positional Encoding (RoPE)

- We want our embeddings to be invariant to absolute position.
- We know that inner products are invariant to arbitrary rotation.



Position independent embedding

Embedding "of course we know"

Rotate by '2 positions'

Embedding "we know that"

Rotate by '0 positions'

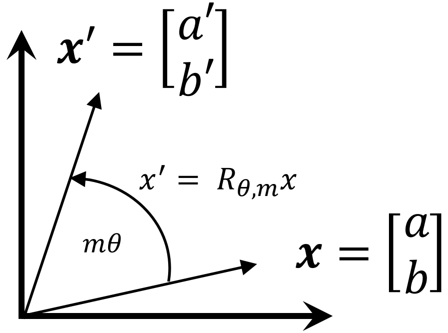
[Slide credit: Tatsu Hashimoto]

Bonus

Thinking About Rotation Matrix

- In 2D, a rotation matrix can be defined in the following form:

$$R_{\theta,m} = \begin{pmatrix} \cos m\theta & -\sin m\theta \\ \sin m\theta & \cos m\theta \end{pmatrix}$$



- The rotation increases with increasing θ and m.

Bonus

Rotary Positional Encoding (RoPE): 2D

- If you have two input representations x_n and x_m at positions n and m:

$$qk_{mn} = \left(R_{\theta,m} W_q \boldsymbol{x}_m\right)^T \left(R_{\theta,n} W_k \boldsymbol{x}_n\right)$$
$$= \boldsymbol{x}_m^T W_q^T R_{\theta,m}^T R_{\theta,n} W_k \boldsymbol{x}_j$$
$$= \boldsymbol{x}_m^T W_q^T R_{\theta,n-m} W_k \boldsymbol{x}_j$$

  - $R_{\theta,m}$: rotation matrix, rotates a vector it gets multiplied to proportional to θ and the position indices m, n.
- Intuition: **nearby** words have **smaller relative rotation**.

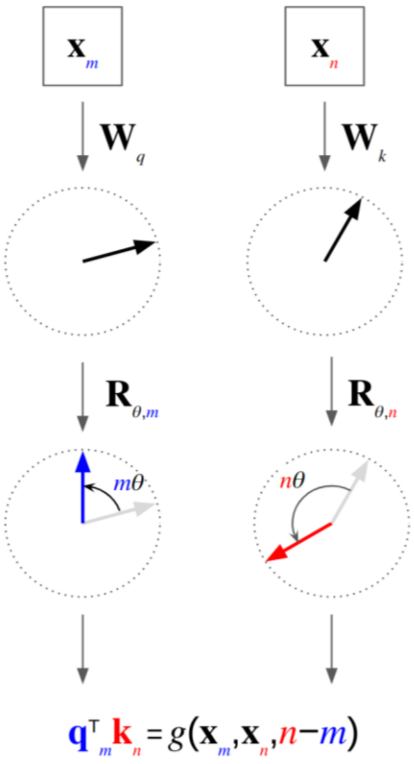


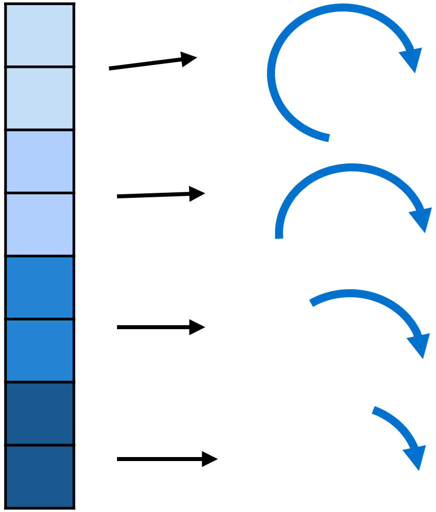
Figure source

RoFormer: Enhanced Transformer with Rotary Position Embedding (2022)

Bonus

Thinking d Dimensions

- Create **pairs of** coordinates that would rotate together
- Each pair can have a different frequency of rotations



Bonus

Thinking d Dimensions

- In practice, we are rotating d dimensional embedding matrices.
- Idea: rotate different dimensions with different angles:
  - $\boldsymbol{\Theta} = [\theta_1, \theta_2, \theta_3, \dots, \theta_{d/2}]$

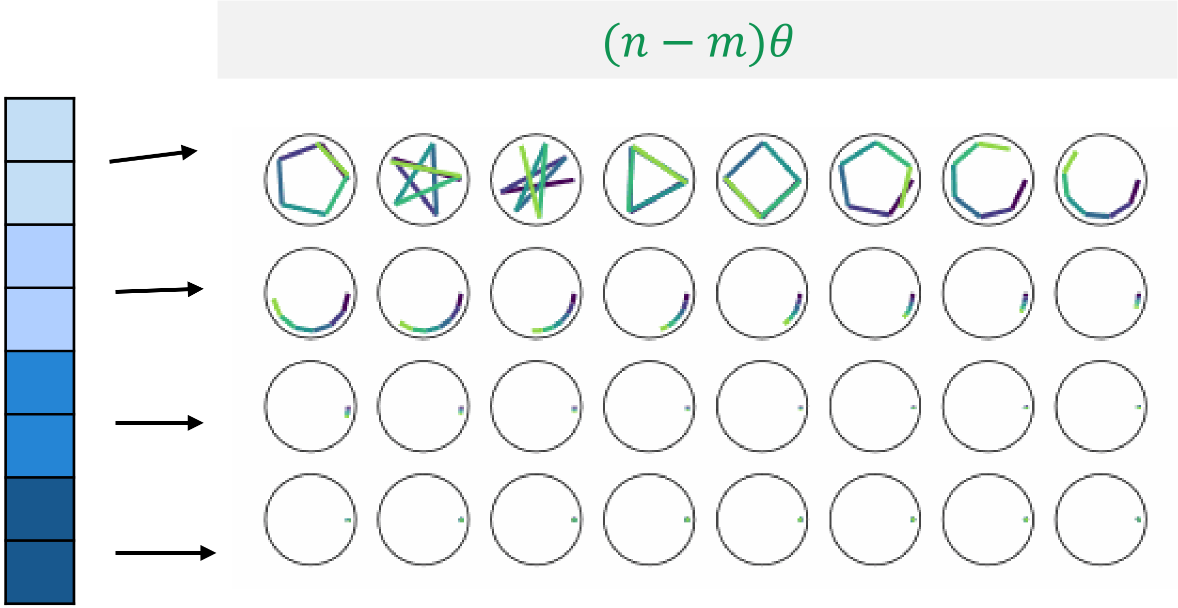
d = the dimension of the query and key space
$m\theta$ = total rotation angle

$$R^d_{\Theta,m} = \begin{bmatrix} \cos m\theta_1 & -\sin m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ \sin m\theta_1 & \cos m\theta_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos m\theta_2 & -\sin m\theta_2 & \cdots & 0 & 0 \\ 0 & 0 & \sin m\theta_2 & \cos m\theta_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos m\theta_{d/2} & -\sin m\theta_{d/2} \\ 0 & 0 & 0 & 0 & \cdots & \sin m\theta_{d/2} & \cos m\theta_{d/2} \end{bmatrix}$$

- By using different frequencies for different dimensions of the positional encoding vector, a unique "fingerprint" can be created for each positional information.

RoPE, visualized.

- Each pair of coordinates are rotated by an angle proportional to the token position $n\theta$ or relative dist of words $(n-m)\theta$
- Top row: high-freq dimensions.
- Bottom row: low-freq dimensions.



Visualization by Sasha Rush: https://colab.research.google.com/drive/1JhVaH1Tr6BMLKX9mgIjN83qFh9DOOyvC?usp=sharing#scrollTo=xeBojFzJRfu1

Bonus

RoPE via Faster (Dense) Operations

- While we can implement this matrix multiplication as is, we can make use of $R^d_{\Theta,m}$'s sparsity for faster multiplication via dense operations :

$$\mathbf{x}' = \mathbf{R}^d_{\Theta,m}\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{d-1} \\ x_d \end{bmatrix} \otimes \begin{bmatrix} \cos m\theta_1 \\ \cos m\theta_1 \\ \cos m\theta_2 \\ \cos m\theta_2 \\ \vdots \\ \cos m\theta_{d/2} \\ \cos m\theta_{d/2} \end{bmatrix} + \begin{bmatrix} -x_2 \\ x_1 \\ -x_4 \\ x_3 \\ \vdots \\ -x_d \\ x_{d-1} \end{bmatrix} \otimes \begin{bmatrix} \sin m\theta_1 \\ \sin m\theta_1 \\ \sin m\theta_2 \\ \sin m\theta_2 \\ \vdots \\ \sin m\theta_{d/2} \\ \sin m\theta_{d/2} \end{bmatrix}$$

Integrating RoPE to Self-Attention

- **Drop the additive** positional encoding and make it **multiplicative to QK** matrices.
- Given input $\mathbf{x}$, the standard SA is computed as:

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathrm{T}}}{\sqrt{d}}\right)\mathbf{V}$$

$$\boldsymbol{Q} = \mathbf{W}^{q}\mathbf{x}, \qquad \boldsymbol{K} = \mathbf{W}^{k}\mathbf{x}, \qquad \boldsymbol{V} = \mathbf{W}^{v}\mathbf{x}$$

- Given input $\mathbf{x}$, SA with RoPE is computed as:

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{Q}_{\text{rope}}\mathbf{K}_{\text{rope}}^{\mathrm{T}}}{\sqrt{d}}\right)\mathbf{V}$$

- Note since each the rotation depends on distance each word pair, it’s not easy to write these in matrix form.

Setting $\Theta = [\theta_1, \theta_2, \theta_3, \dots, \theta_{d/2}]$

- One common way is to set them using a “base” frequency:

$$\mathbf{\Theta} = \left\{\theta_i = B^{-2(i-1)/d},\ i \in [1, 2, \dots, d/2]\right\}$$

- Example:
  - $\theta_1 = 1$, for any choice of B or d
  - $\theta_{d/2} = \frac{1}{B^{1-2/d}}$. If $d = 512$ and $B = 10K$, then $\theta_{256} = 1/10{,}000$
- The exponent $-2(i-1)/d$ scales the input so that:
  - **lower** dimensions correspond to **higher**-frequency sinusoids (rapid changes), and
  - **higher** dimensions correspond to **lower**-frequency sinusoids (slow changes)
- This **spread of frequencies** allows the model to represent both fine-grained (local) and coarse-grained (global) positional relationships.

Q: How should we set the "base" freq?

$$\Theta = \left\{\theta_i = B^{-2(i-1)/d},\ i \in [1, 2, \ldots, d/2]\right\}$$

- You want the largest wavelength (lowest freq) be close to your longest context dist.
- Lowest frequency: $\theta_{d/2} = \frac{1}{B^{1-2/d}}$ ⇒ The largest wavelength: $T_{\max} = \frac{2\pi}{\theta_{d/2}} \approx 2\pi B$
- Examples:
  - If $B = 10{,}000 \Rightarrow\ T_{\max} \approx 62K$ tokens
  - If $B = 100{,}000 \Rightarrow\ T_{\max} \approx 628K$ tokens

- In both original RoPE paper (and the original Transformer paper), $B = 10{,}000$.

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Examples of Base Frequencies

- Llama3 positional encoding: RoPE (θ = 500,000)

We increase the RoPE base frequency hyperparameter to 500,000. This enables us to better support longer contexts; Xiong et al. (2023) showed this value to be effective for context lengths up to 32,768.

- Gemma3 positional encoding: A mix of RoPE (θ = 1,000,000) + RoPE (θ = 10,000)

Long context. Gemma 3 models support context length of 128K tokens, with the exception of the 1B model that has 32K. We increase RoPE base frequency from 10k to 1M on global self-attention layers, and keep the frequency of the local layers at 10k. We follow a process similar to the

Llama3 tech report: https://arxiv.org/pdf/2407.21783
Gemma3 tech report: https://arxiv.org/abs/2503.19786

Summary

- Encoding positional information in language models is a non-trivial problem.
  - We discussed various proposal: learned, absolute, relative encoding, NoPos, etc.

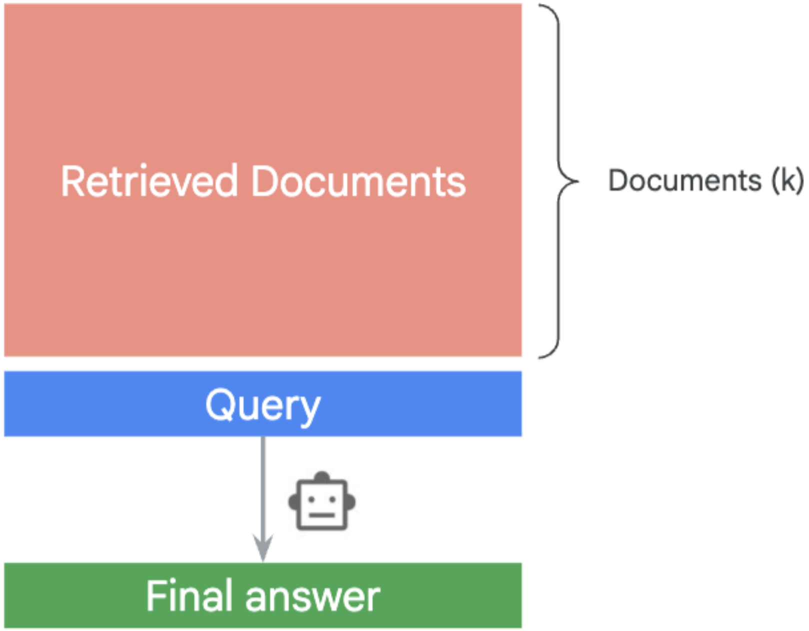
- This is an important literature related to the length generalization of Transformers.

- This is an active research area and likely to change in the coming years.

Long Context Generalization

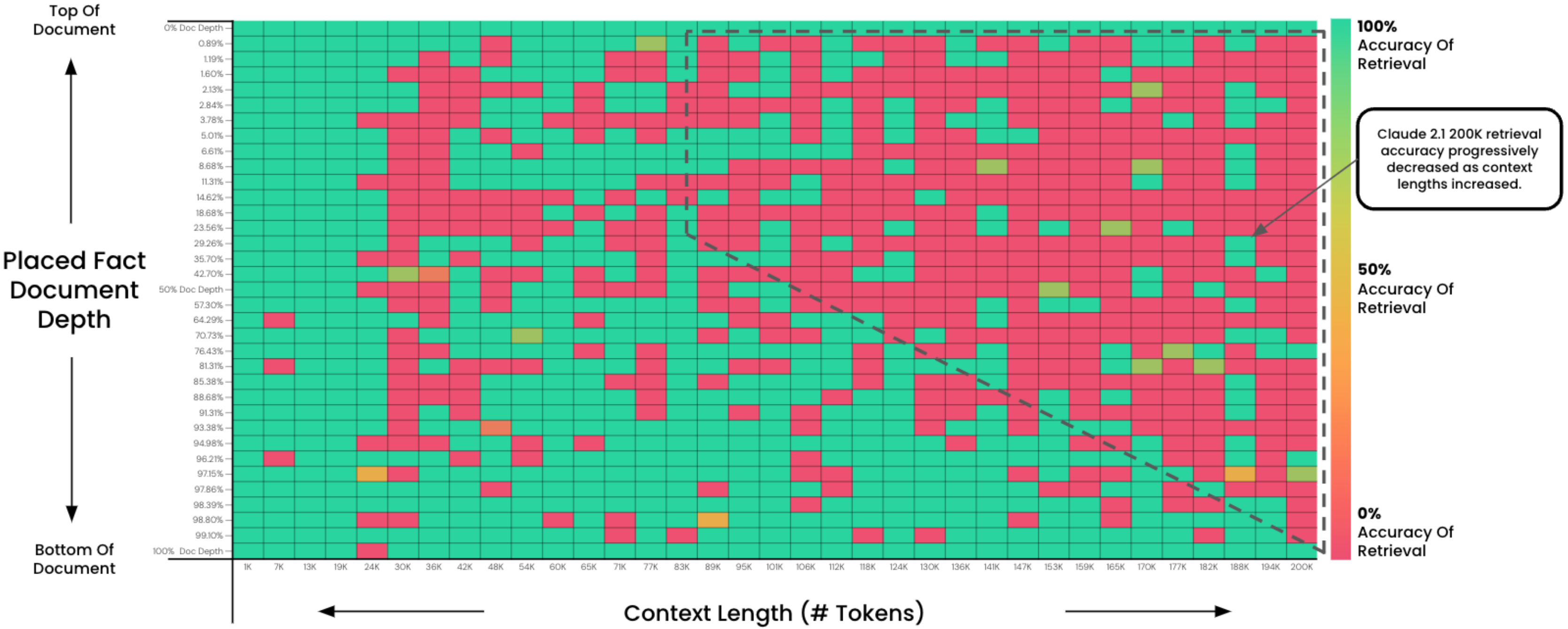
Needle-in-Haystack Problems

- You're given k documents in the input.
  - (e.g., the retrieval results)
- One of these docs may contain the answer to a given query which we want to answer.



Pressure Testing Claude-2.1 200K via "Needle In A HayStack"

Asking Claude 2.1 To Do Fact Retrieval Across Context Lengths & Document Depth

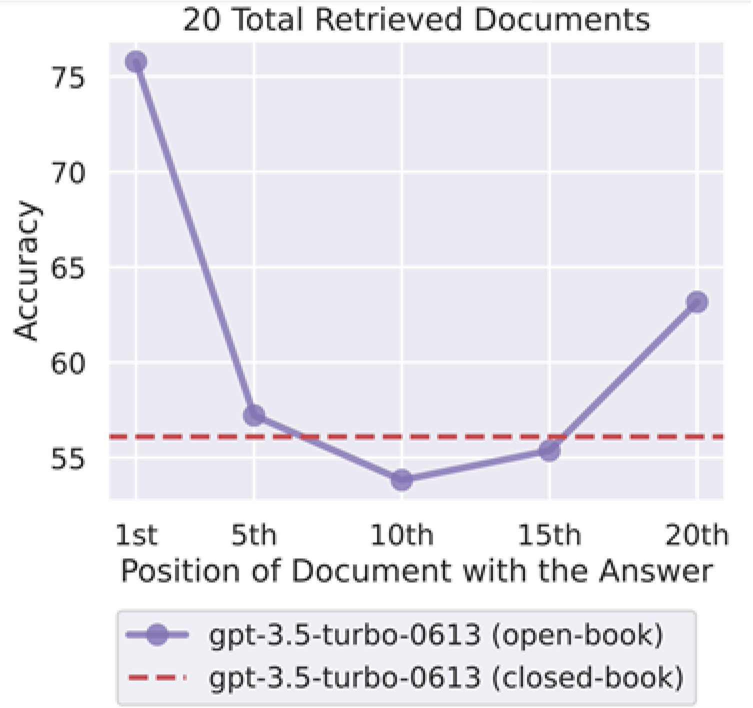


Goal: Test Claude 2.1 Ability To Retrieve Information From Large Context Windows

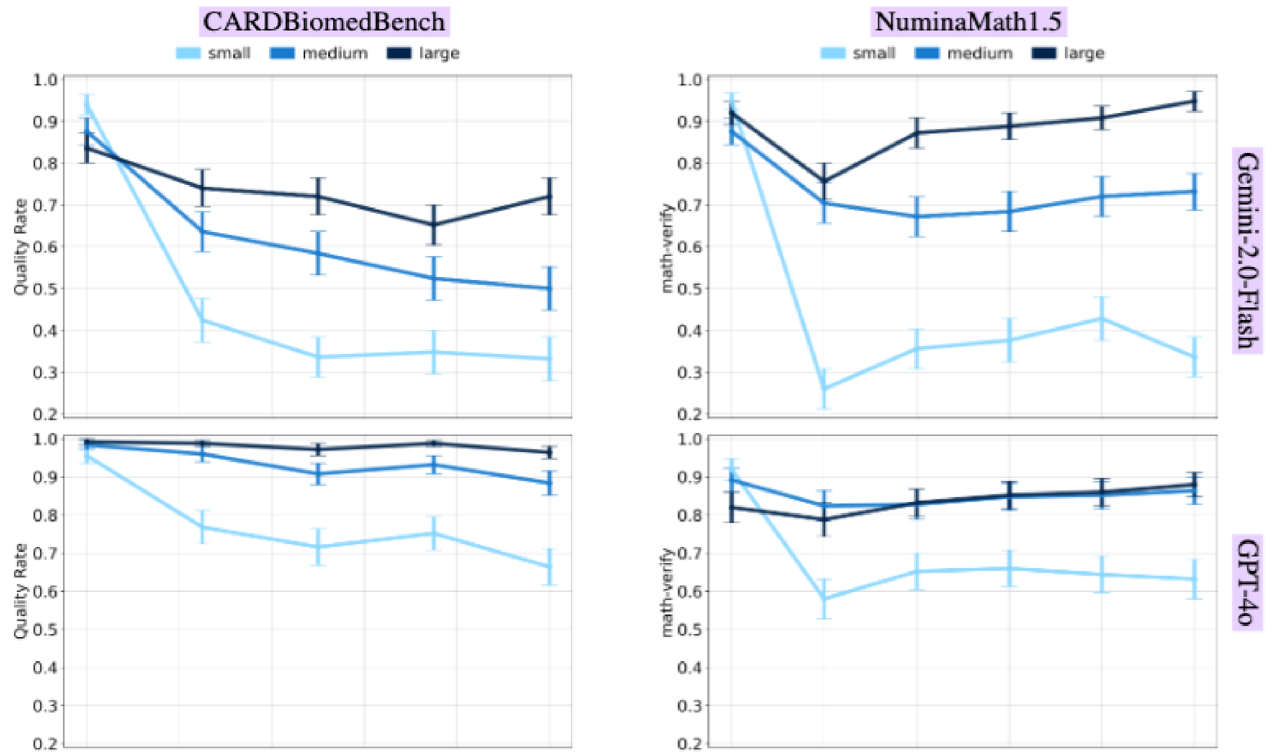
A fact was placed within a document. Claude 2.1 (200K) was then asked to retrieve it. The output was evaluated (with GPT-4) for accuracy.
This test was run at 35 different document depths (top > bottom) and 35 different context lengths (1K >200K tokens).
Document Depths followed a sigmoid distribution

https://github.com/gkamradt/LLMTest_NeedleInAHaystack

Recency vs Primacy bias



Recent vs Primacy Bias: Size of “Gold” doc

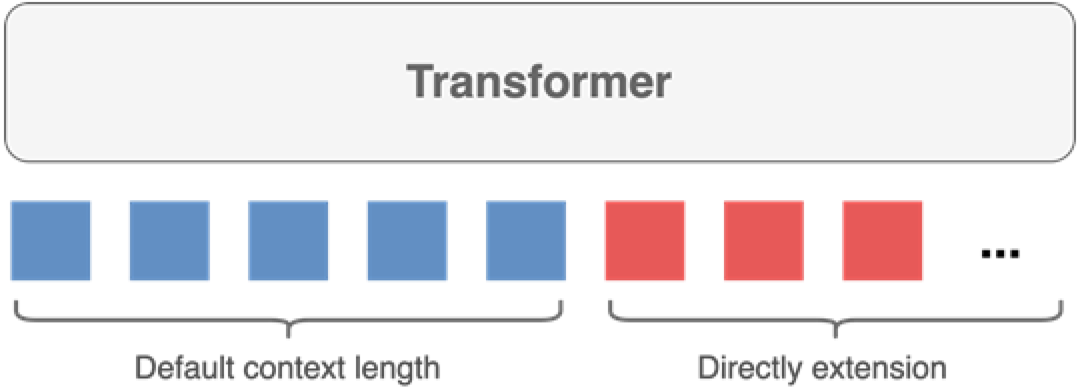


[Hidden in the Haystack: Smaller Needles are More Difficult for LLMs to Find]()

Long Context Extensions

Context Extension: Challenges

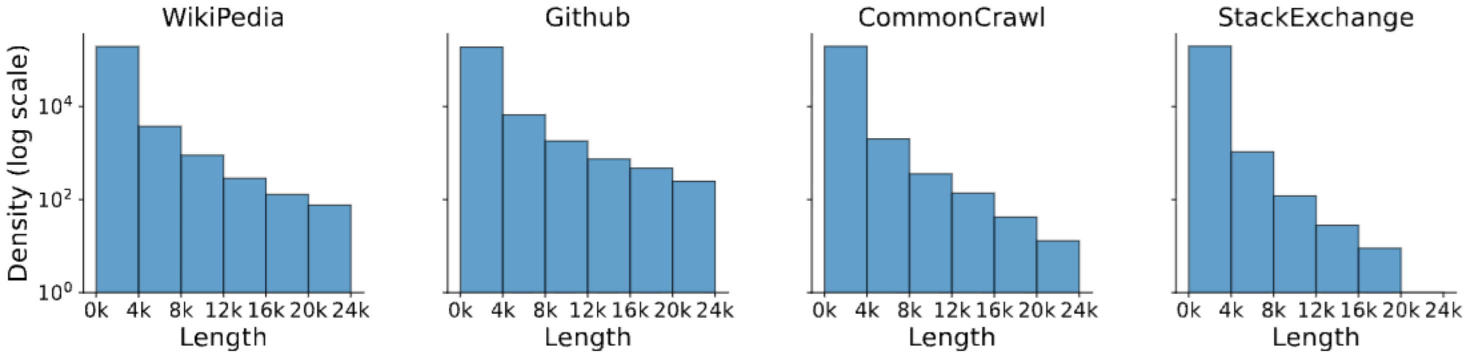
- You're given a transformer trained on a default length.
- Now we want to extend its length.



Chen et. al, 2023, Extending Context Window of Large Language Models via Positional Interpolation

Context Extension: Challenges

- **Challenge 1:** Adapting the positional encodings
- **Challenge 2:** Efficiency challenge since the FLOP/memory footprint grows quadratically.
- **Challenge 3:** Long data is very limited.



Chen et. al, 2023, Extending Context Window of Large Language Models via Positional Interpolation

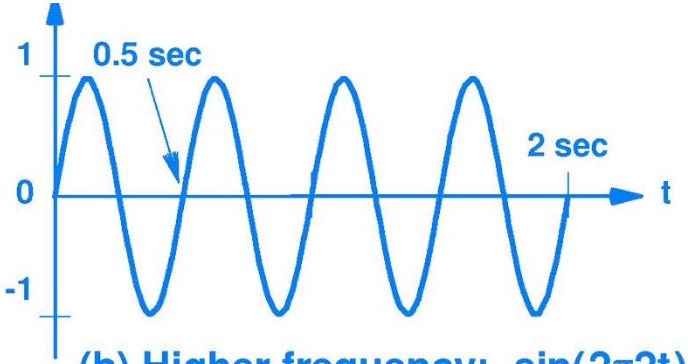
Context Extension during Pre-training

- Two ingredients:
  - Extensible positional encoding
  - Carefully selected long training data
- Recipe:
  - Pre-train a short context model(block size = 4k) on 1 trillion tokens of text
  - Adjust the hyperparameters of the positional embeddings
  - Continue pre-training but increase the block size to support long-contexts (block size = 80k). This may involve training on a smaller amount of data.

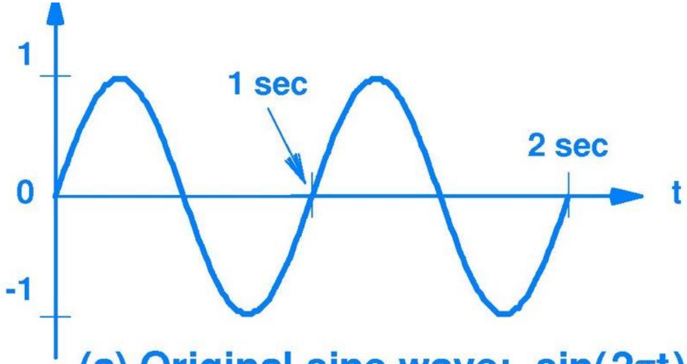
Approach: Changing ROPE Frequencies

- Assuming that we use RoPE encoding, **reduce** the base frequency by α so that more tokens fit within each period.

$$\mathbf{f_W}(\mathbf{x}_i, \boldsymbol{\theta}) = \mathbf{R}(\boldsymbol{\theta}, i)\mathbf{W}^\top \mathbf{x}_i \quad \rightarrow \quad \mathbf{f}(x_i, \boldsymbol{\alpha} \odot \boldsymbol{\theta})$$



(b) Higher frequency: sin(2π2t)



(a) Original sine wave: sin(2πt)

Uniform Scaling

- Linear Position Interpolation (PI):
  - Scale all frequencies by the same scale:

$$\alpha_j^{\mathrm{PI}} = \frac{C}{C'} = \frac{1}{t}$$

  - This lowers every frequency *uniformly* so more tokens fit in each sinusoidal period.

NTK Scaling

$$\alpha_j = \kappa^{-\frac{2j}{d_k}}, \qquad j = 1, \dots, \frac{d_k}{2}$$

  - High frequencies scaled down less (~preserved)
  - Low frequencies are shrunk more (matching PI extension)

- Why is it called "NTK"?
  - Neural Tangent Theory is this idea that, high-frequency information are hard to learn.
  - So, it's better to not mess with them! (i.e., scale high-frequencies minimally)

YaRN Scaling

- Basically, piecewise scaling across dimensions.

$$\alpha_j^{\text{YaRN}} = ((1-\gamma_j)\frac{1}{t} + \gamma_j)/\sqrt{T} \qquad \gamma_j = \begin{cases} 0, & \text{if } \theta_j < p, \\ 1, & \text{if } \theta_j > q, \\ \frac{\theta_j - p}{q - p}, & \text{otherwise.} \end{cases}$$

  - Context length ratio: $t = C'/C$
  - Temperature parameter: T to reduce all freq uniformly, if needed.
  - Interpolation gate for dimension j : γ_j
  - Hyperparameters: p, q, T which can be tuned as needed.
- Modes:
  - Low-frequency dimensions ($\theta_j < p$): fully PI-scaled ($1/t$).
  - High-frequency dimensions ($\theta_j > q$): no scaling (kept as original RoPE) if T=1.
  - Middle band ($q < \theta_j < p$): smoothly interpolated.

Empirical Comparison of RoPE extensions

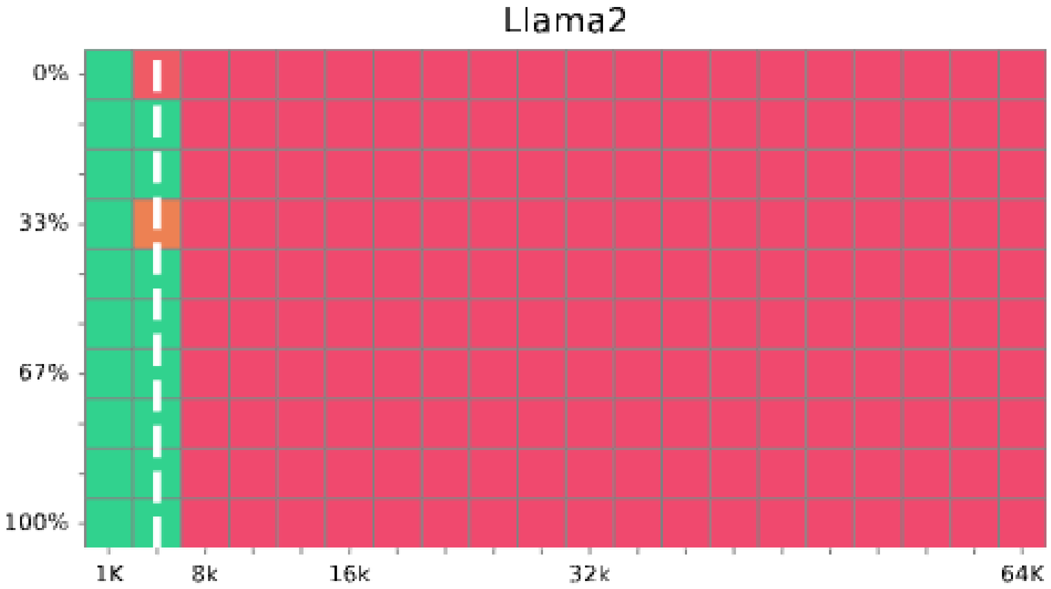
- Base model: llama2-7B which has a context of window of ~4k tokens.
- Goal: extending the context window from 4k to 32k

Table 1: Overview of results across different extension types.

Attention Mechanisms		Model	PPL	Needle	Mshots	LongB	RULER
Exact Attention	Frozen	NTK-F	14.52	18.8	64.5	25.54	0.72
	Fine-Tuned	PI	5.85	42.1	**75.5**	33.48	57.66
		YaRN	5.85	46.7	75.0	33.45	36.95
		CLEX	5.82	71.1	74.0	33.48	52.17
		NTK-32K	**5.79**	**83.7**	71.0	**35.32**	59.42
		NTK-64K	5.93	69.1	73.0	34.30	**60.03**

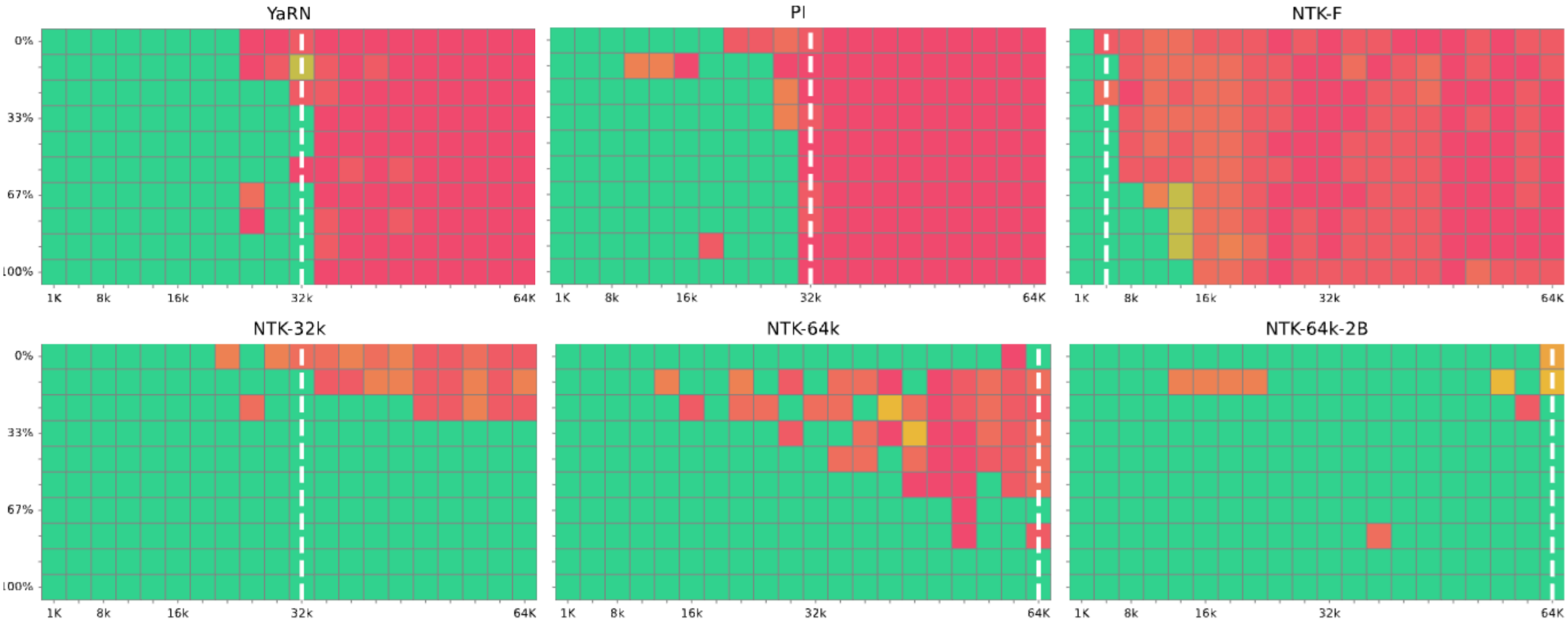
JOHNS HOPKINS WHITING SCHOOL of ENGINEERING

[A Controlled Study on Long Context Extension and Generalization in LLMs, 2025](#)



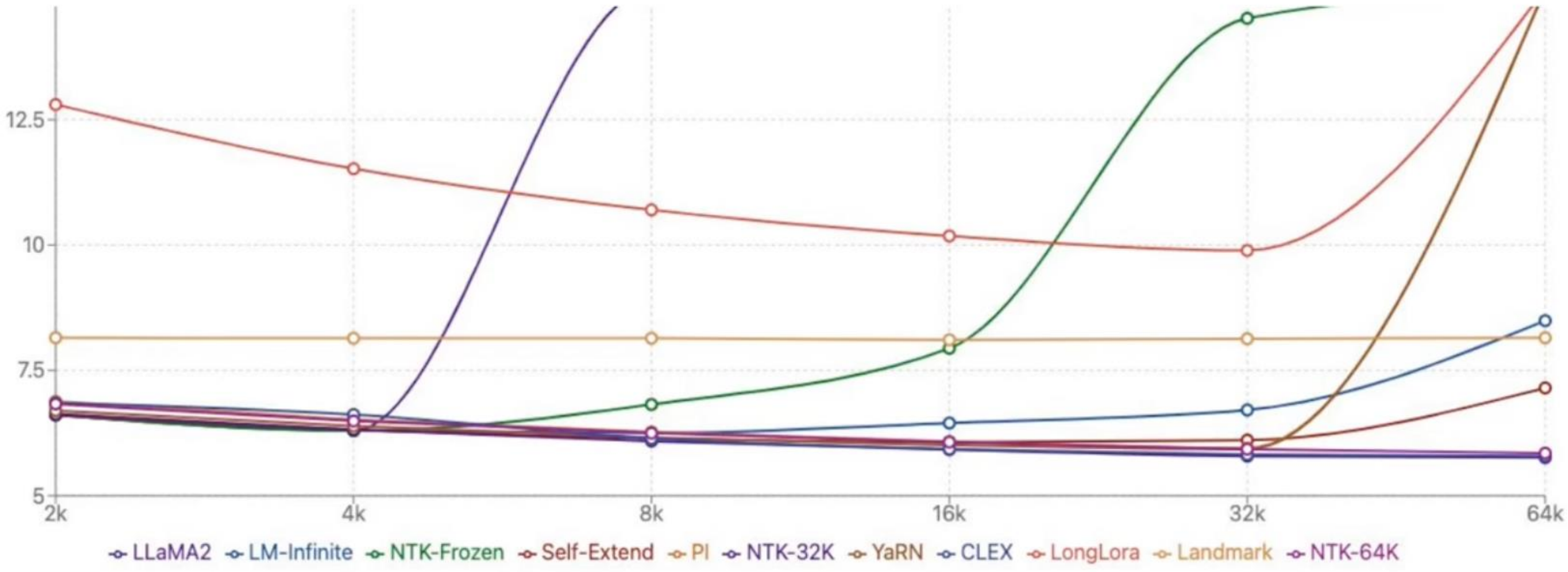
- X-axis shows length of the input.
- Y-axis shows where the “needle” was included in the context (first row: needle in the beginning)
- The white dashed line denotes the longest length examples seen at training or finetuning

[A Controlled Study on Long Context Extension and Generalization in LLMs, 2025]



- X-axis shows length of the input.
- Y-axis shows where the “needle” was included in the context (first row: needle in the beginning)
- The white dashed line denotes the longest length examples seen at training or finetuning

[A Controlled Study on Long Context Extension and Generalization in LLMs, 2025](#)



- Y-axis shows perplexity.
- The base model is trained up to 4k tokens. .
- The extensions are trained on data up to 32k.
- Note some generalize to 64k.

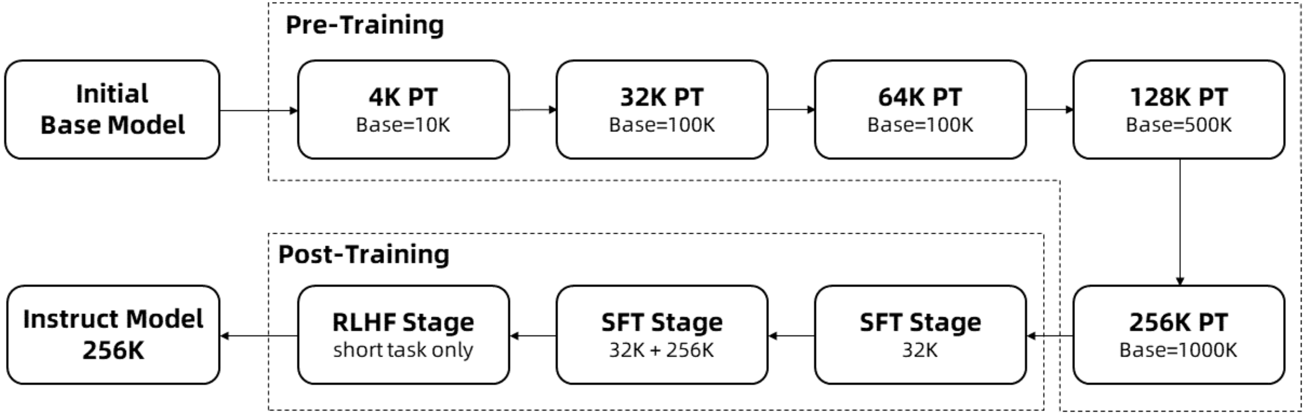
[A Controlled Study on Long Context Extension and Generalization in LLMs, 2025](#)

Model	Positional enc	Pre-training extension	Post-training extension
Qwen 3	RoPE	Increased base freq from 10K to 1M; used YaRN. Also uses Dual Chunk Attention.	Training on a dataset of long Chain-of-Thoughts.
Apple	RoPE (local attention) and no encoding for global attention	Trained on seq up to 65K tokens sampled from "natural" data e.g., licensed books, code repos, synthetic long-form data.	??
GLM	RoPE	Trained on 4K data (base=10K). Then did more training up to 128K (base=1M). Up-sample long documents from the pre-training corpus.	SFT with long reasoning traces (up to 128K)
Gemma3	RoPE	Pre-train on 32K seq (base=10K) and scale to 128K toward the end of training (base=1M).	??
Lamma3	RoPE	Pre-train on shorter sequences (base=500K) and then extend to 128K tokens.	SFT on some (0.1%) long data

Qwen3 Technical Report, Apple Intelligence Foundation Language Models, GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models, Gemma3 tech report

Examples in the wild: Qwen3

- Qwen2.5-7B-Instruct-1M
  - **Changing RoPE base frequency** (to stretch positional embeddings further).
  - **Training recipe** to gradually expose the model to longer contexts.



[Qwen2.5-1M Technical Report, 2025](#)

Examples in the wild: Gemma3

- *"Instead of training with 128K sequences from scratch, we pre-train our models with 32K sequences and then scale the 4B, 12B, and 27B models up to 128K tokens at the end of pre-training while rescaling RoPE"*
- *"Our models generalize to 128K, but rapidly degrade as we continue to scale."*

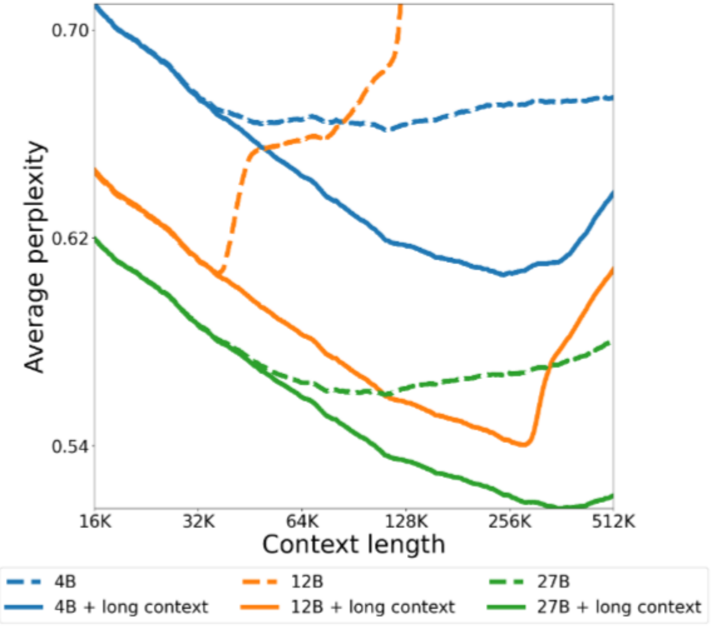


Figure 7 | **Long context** performance of pre-trained models before and after RoPE rescaling.

Gemma3 tech report: https://arxiv.org/abs/2503.19786

Examples in the wild: GLM-4.5

- *We set the maximum sequence length to 4,096 during pre-training, and extended it to 32,768 and 131,072 during the mid-training stage … When extending the sequence length to 32K, we also adjusted RoPE's base frequency from 10,000 to 1,000,000 for better long-context modeling ability."*

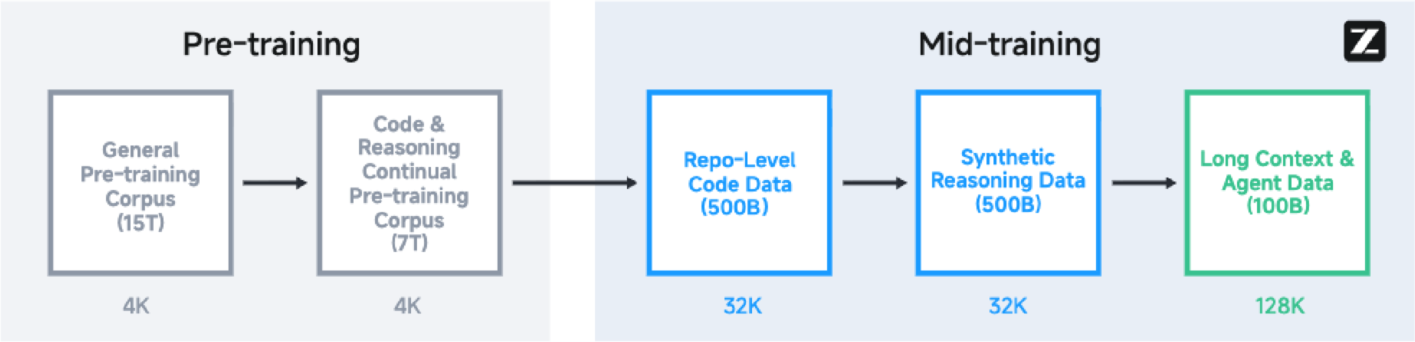


Figure 3: Pre-training and mid-training stages for GLM-4.5. We adapt a multi-stage training recipe and extend the sequence length from 4K to 128K.

Gemma3 tech report: https://arxiv.org/abs/2503.19786

Examples in the wild: Lamma3

3.4.2 Long Context Pre-Training

In the final stages of pre-training, we train on long sequences to support context windows of up to 128K tokens. We do not train on long sequences earlier because the compute in self-attention layers grows quadratically in the sequence length. We increase the supported context length in increments, pre-training until the model has successfully adapted to the increased context length. We assess successful adaptation by measuring whether **(1)** model performance on short-context evaluations has recovered completely and **(2)** the model perfectly solves "needle in a haystack" tasks up to that length. In Llama 3 405B pre-training, we increased context length gradually in six stages, starting from the original 8K context window and ending in the final 128K context window. This long-context pre-training stage was performed using approximately 800B training tokens.

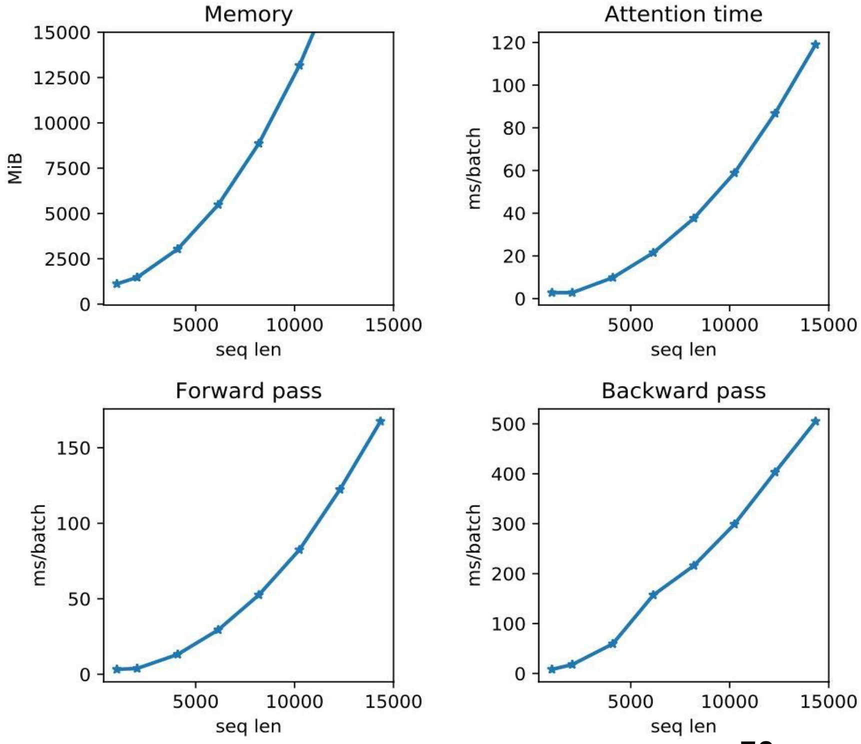
Summary

- RoPE the dominant positional encoding.
- People usually train short and extend longer.

Long Context:
Efficiency Considerations

Transformer LMs and Long Inputs

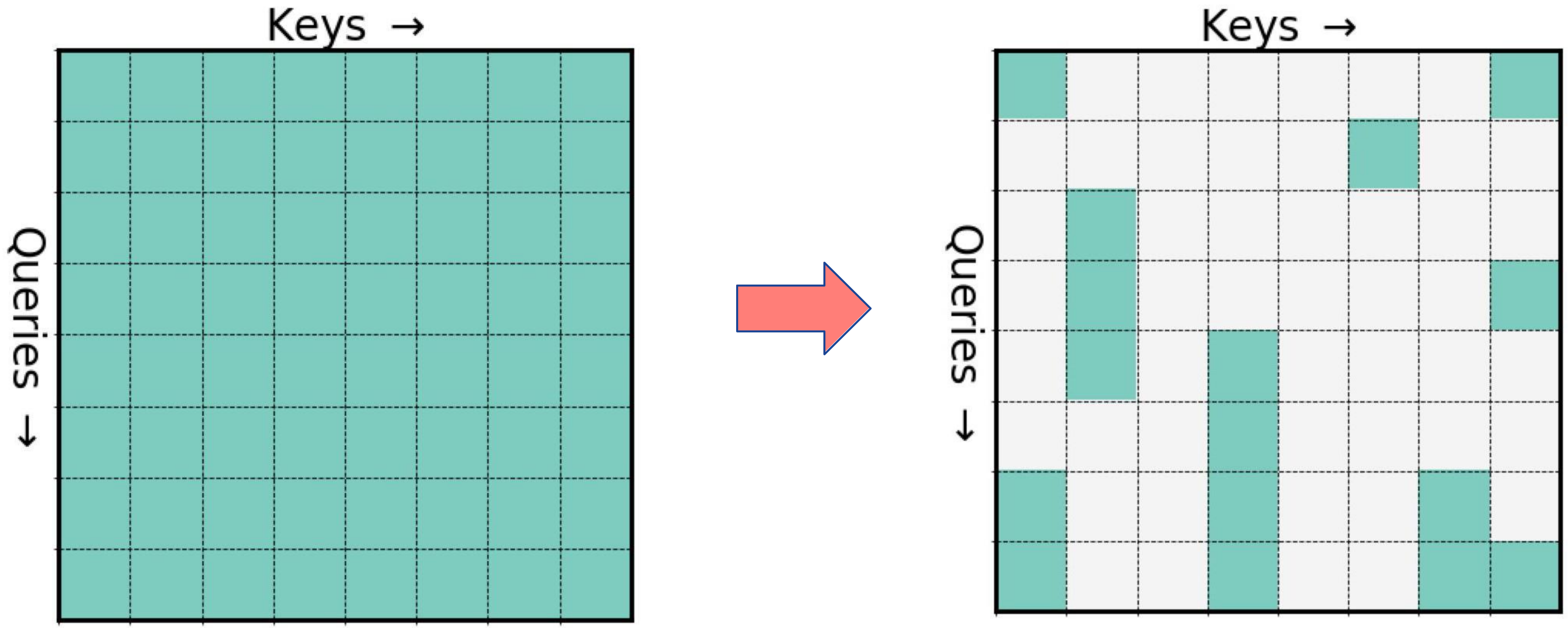
- **Length generalization:** Do Transformers work accurately on long inputs?
- **Efficiency considerations:** How efficient are LMs are long inputs?



JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Sparse Attention Patterns

- The idea is to make the attention operation sparse



Sparse Attention Patterns: Challenge

- Ok sparsity is great, but how to efficiently implement this?

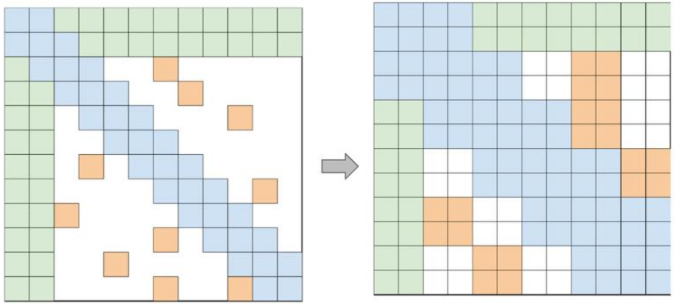
- **Challenge:** Arbitrary sparse matrix multiplication is not supported in DL libraries

- **A solution:** Perform computations in blocks

[NAACL 2021 Tutorial Beltagy, Cohan, Hajishirzi, Min, and Peters]

Pre-specified Sparsity Patterns: Computations

- Efficient blockified implementation

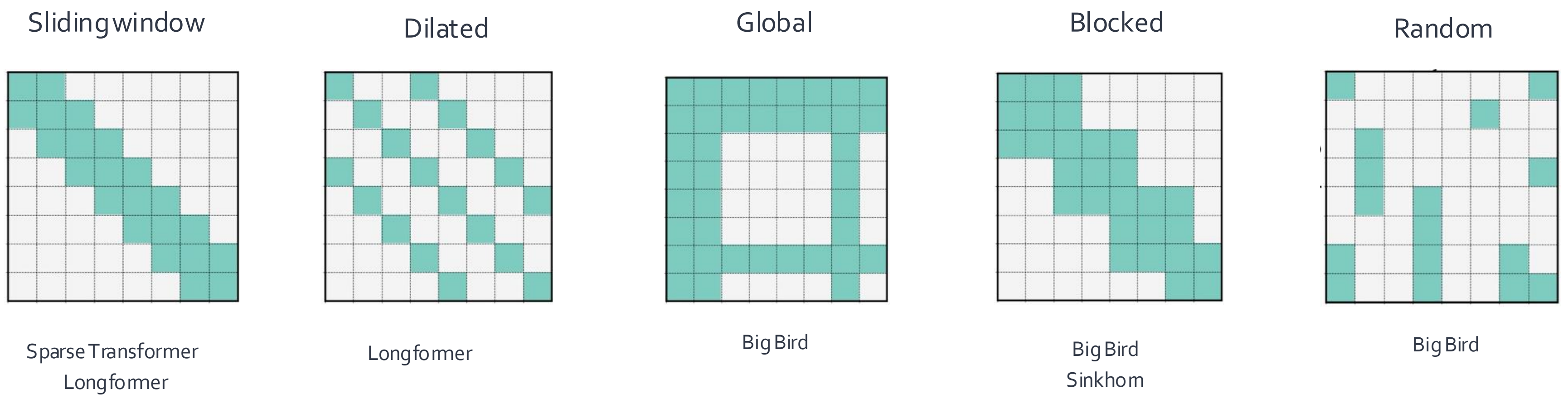


- There are libraries for implementing blockified sparse matrix multiplication.
    - Can be hardware specific
    - Block Sparse ([Gray et al., 2017])
    - TVM toolkit ([Chen et al., 2018])
    - cuSPARSE

[Big Bird ([Zaheer et al., 2020])]

Pre-specified Sparsity Patterns

- A variety of patterns has been explored in the past work
  - Longformer (Beltagy et al., 2020), Sparse Transformer (Child et al., 2019), …

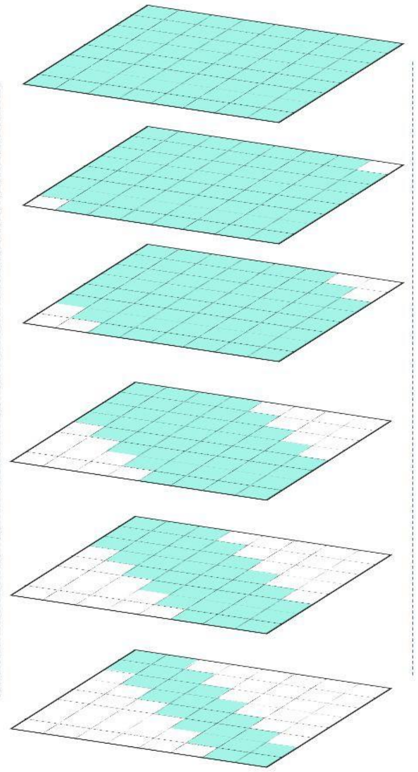


[NAACL 2021 Tutorial Beltagy, Cohan, Hajishirzi, Min, and Peters]

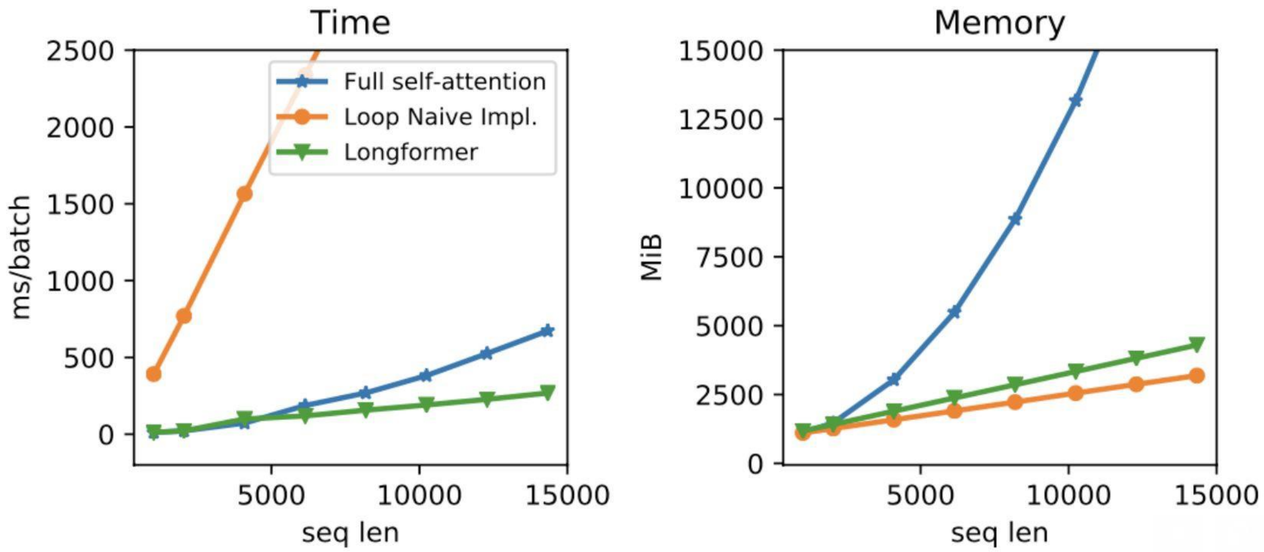
Pre-specified Sparsity Patterns

- Different layers and attention heads can follow different patterns

- A common setup is to have earlier layers with sparser attention pattern.
    - Longformer ([Beltagy et al., 2020](#))



Pre-specified Sparsity Patterns: Computations



JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

[Longformer (Beltagy et al., 2020)]

A Notable Adoption: GPT-3

- Sparse patterns also used in GPT-3 ([Brown et al., 2020](#))

2.1 Model and Architectures

We use the same model and architecture as GPT-2 [RWC+19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, ranging over three orders of magnitude from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3. Previous work [KMH+20]

Summary

- How well do Transformers work on long sequences? Not so well.

- How can we make them more efficient? Induce sparsity.

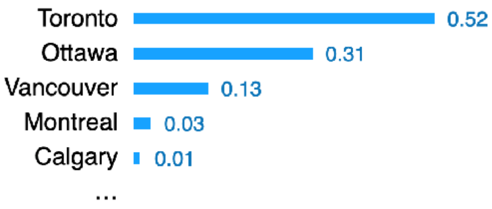
- Many open questions we did not get into:
  - Limitations of sportifying Transformers
  - Alternatives to Transformers (e.g., state-space models)
  - …

Retrieval-Augmented LMs

Retrieval-based Language Models

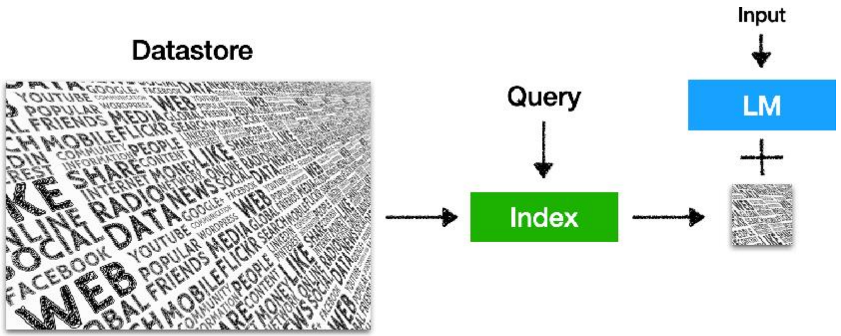
- It is a **language model** $P(x_n \mid x_1, x_2, \cdots, x_{n-1})$

The capital city of Ontario is ___



(can be broadly extended to masked language models or encoder-decoder models)

- It retrieves from an **external datastore** (at least during inference time)



(Also referred to semiparametric and non-parametric models)

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Why retrieval-based LMs?

LLMs can’t memorize all (long-tail) knowledge in their parameters

List 5 important papers authored by Geoffrey Hinton

ChatGPT

Geoffrey Hinton is a renowned computer scientist …
Here are five important papers authored by him:

1. "**Learning Internal Representations by Error Propagation**" (with **D. E. Rumelhart** and **R. J. Williams**) - This paper, published in **1986**, ..

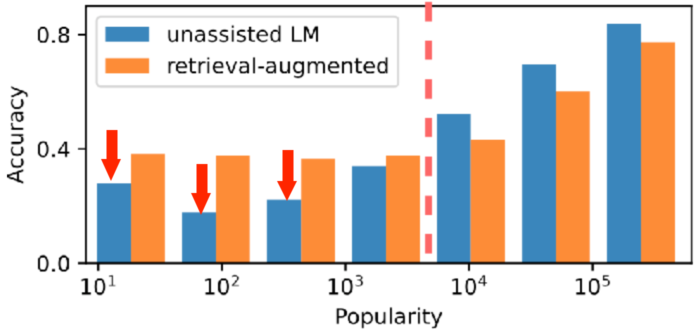
2. "**Deep Boltzmann Machines**" (with **R. Salakhutdinov**) - Published in **2009**, ..

…

4. "**Deep Learning**" (with Y. Bengio and A. Courville) - Published as a book in **2016**,…

5. "**Attention Is All You Need**" (with V. Vaswani, N. Shazeer, et al.) - Published in **2017**, this paper introduced the Transformer model,…

What is Kathy Saltzman’s occupation?



(Mallen et al., 2023)

GPT-3 davinci-003: 20%-30% accuracy

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

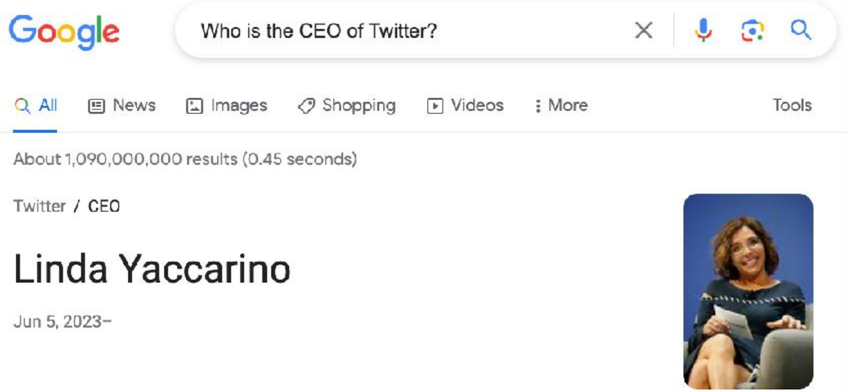
Why retrieval-based LMs?

LLMs’ knowledge is easily outdated and hard to update

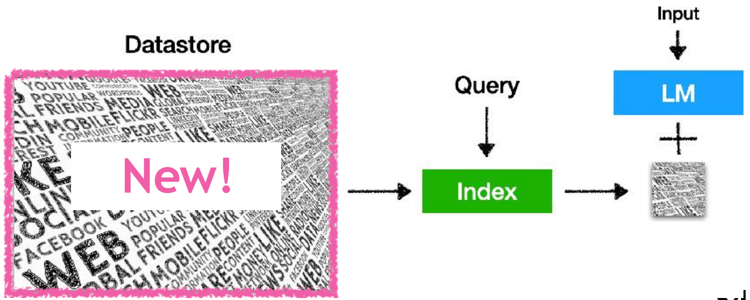
Who is the CEO of Twitter?

ChatGPT

As of my **knowledge cutoff in September 2021**, the CEO of Twitter is **Jack Dorsey**….



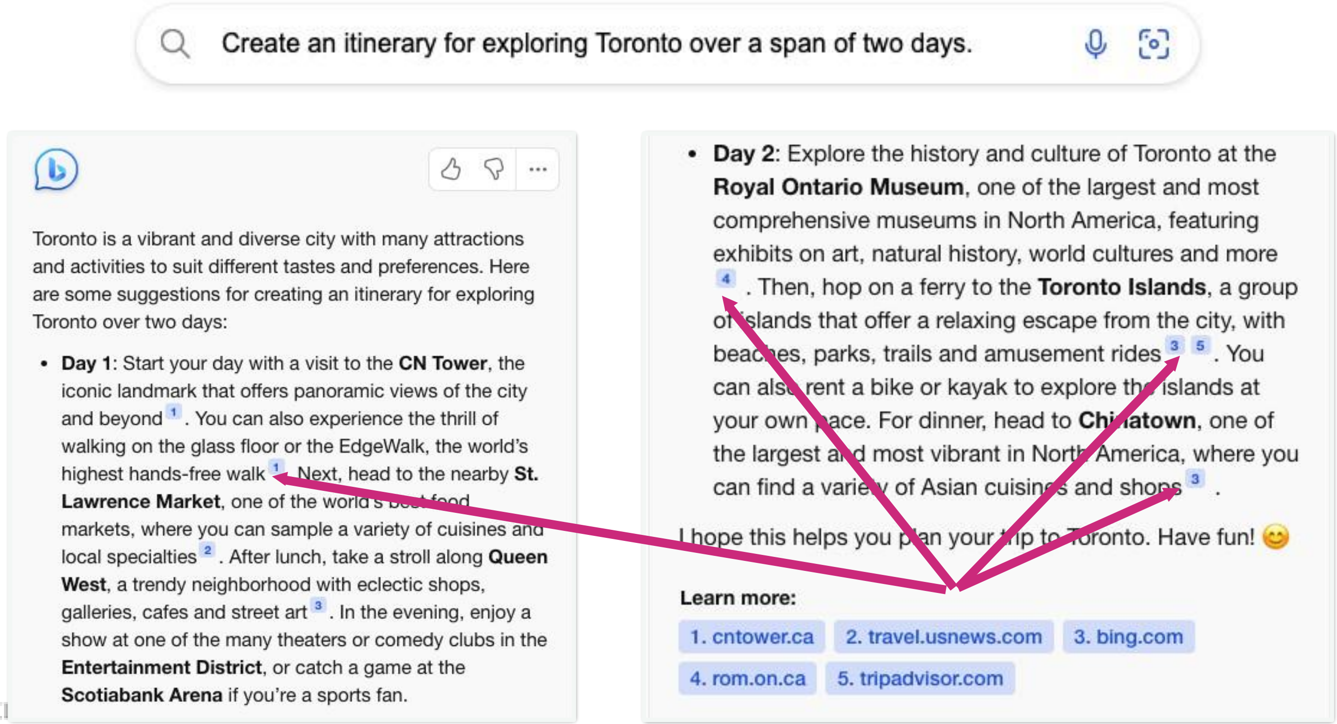
- Existing **knowledge editing** methods are still NOT scalable (active research!)
- The datastore can be easily **updated** and **expanded** - even without retraining!



[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Why retrieval-based LMs?

LLMs' output is challenging to interpret and verify



[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Why retrieval-based LMs?

LLMs are *large* and expensive to train and run

Long-term goal: can we possibly reduce the **training** and **inference costs,** and scale down the size of LLMs?

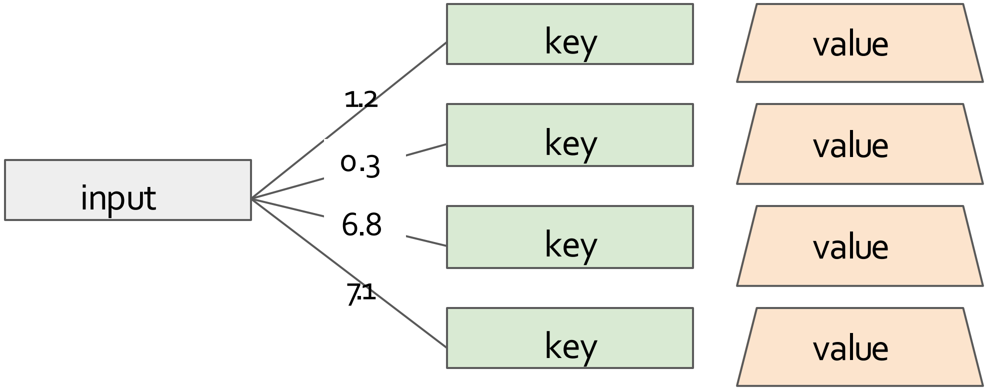
e.g., RETRO (Borgeaud et al., 2021): “obtains comparable performance to GPT-3 on the Pile, despite using **25x fewer parameters**”

What are the Key Design Questions?

- What are your memories?
  - Documents, database records, training examples, etc.
- How to retrieve memories?
  - Use an off-the-shelf search engine (e.g. Google, StackOverflow).
  - How to train your own memory retriever.
- How to use retrieved memories?
  - "Text fusion"
  - Common failure modes:
    - Underutilization: model ignores retrieved memories.
    - Overreliance: model depends too much on memories!

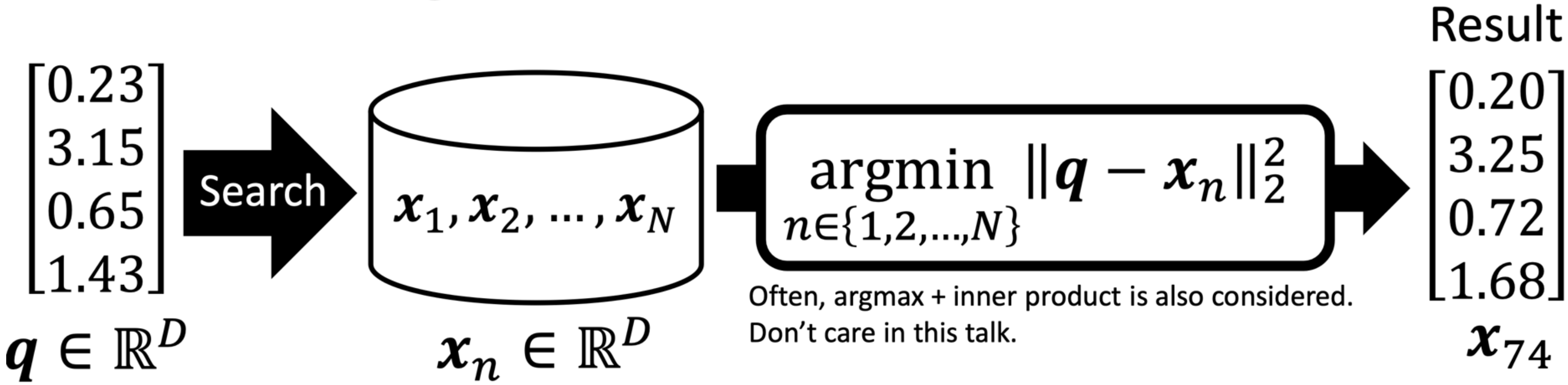
Anatomy of a Neural Retriever

1. Score the input against each key.
2. Return the value for the highest scoring key.



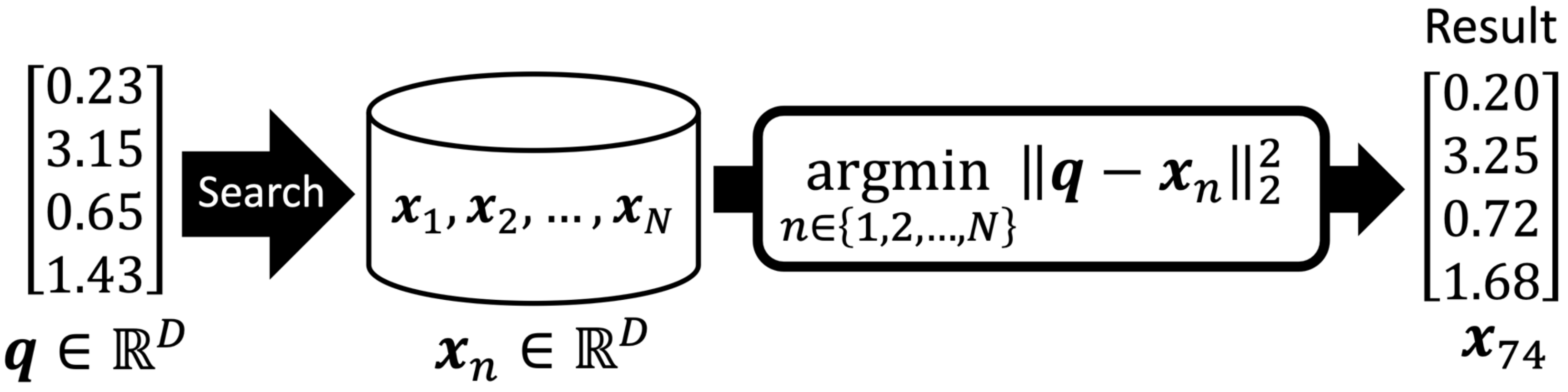
A similarity function: `sim(input, key) → score`

Finding Nearest Neighbors



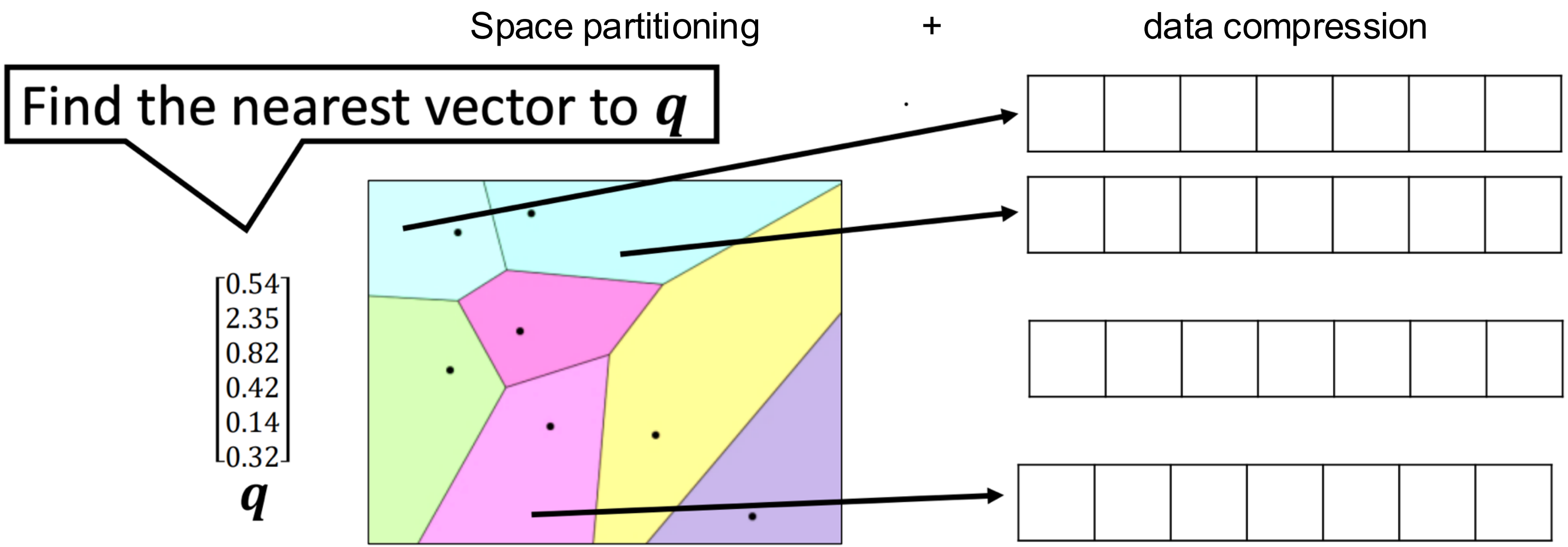
- N D-dim database vectors: $\{x_n\}_{n=1}^{N}$
- Given a query q, find the closest vector from the database
- One of the fundamental problems in computer science
- Solution: linear scan, $O(ND)$, slow ☹

Approximate Finding Nearest Neighbors



- ➢Faster search
- ➢Don't necessarily have to be exact neighbors
- ➢Trade off: runtime, accuracy, and memory-consumption

Approximate NNs: Algorithms, Libraries, Services



- More information: https://github.com/facebookresearch/faiss/wiki

Approximate NNs: Algorithms, Libraries, Services

Algorithm

- Scientific paper
- Math
- Often, by researchers

Product Quantization + Inverted Index (PQ, IVFPQ) [Jégou+, TPAMI 2011]

Hierarchical Navigable Small World (HNSW) [Malkov+, TPAMI 2019]

ScaNN (4-bit PQ) [Guo+, ICML 2020]

Library

- Implementations of algorithms
- Usually, a search function only
- By researchers, developers, etc

faiss

NMSLIB

hnswlib

ScaNN

Service (e.g., vector DB)

- Library + (handling metadata, serving, scaling, IO, CRUD, etc)
- Usually, by companies

Pinecone

Qdrant

Milvus

jina

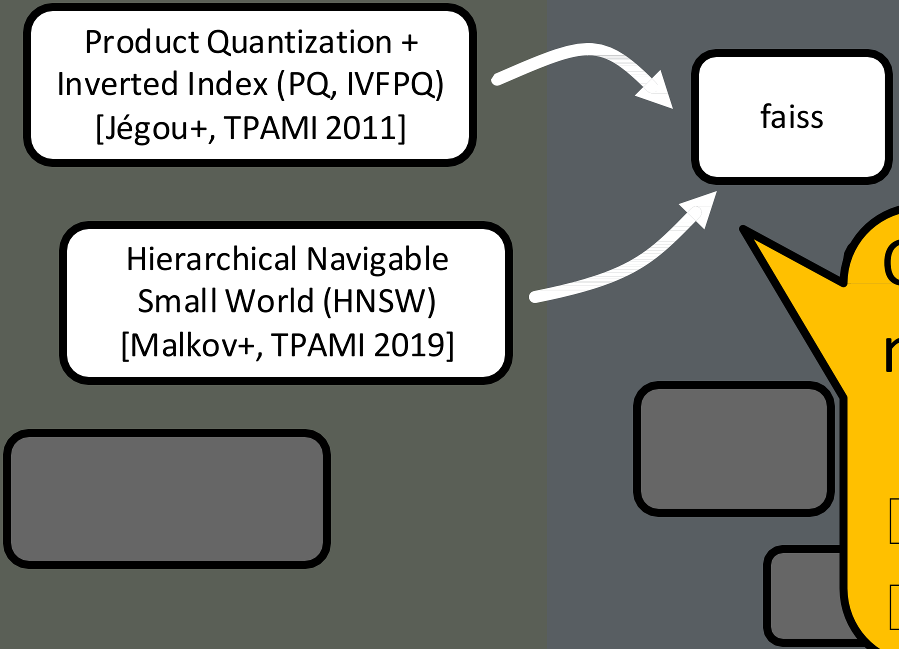
Vald

Vertex AI Matching Engine

Weaviate

[Slides: Yusuke Matsui]

Three levels of technology



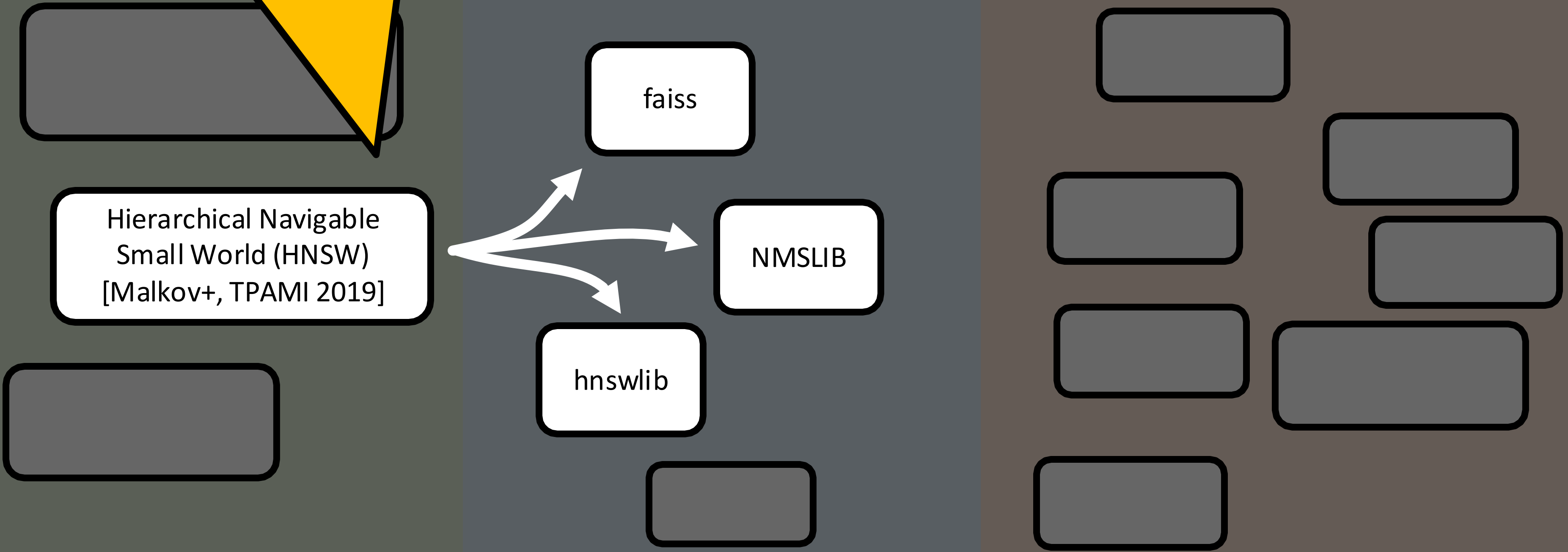
One library may implement multiple algorithms

- ☐ "I benchmarked faiss"
- ☐ "I benchmarked PQ in faiss"

[Slides: Yusuke Matsui]

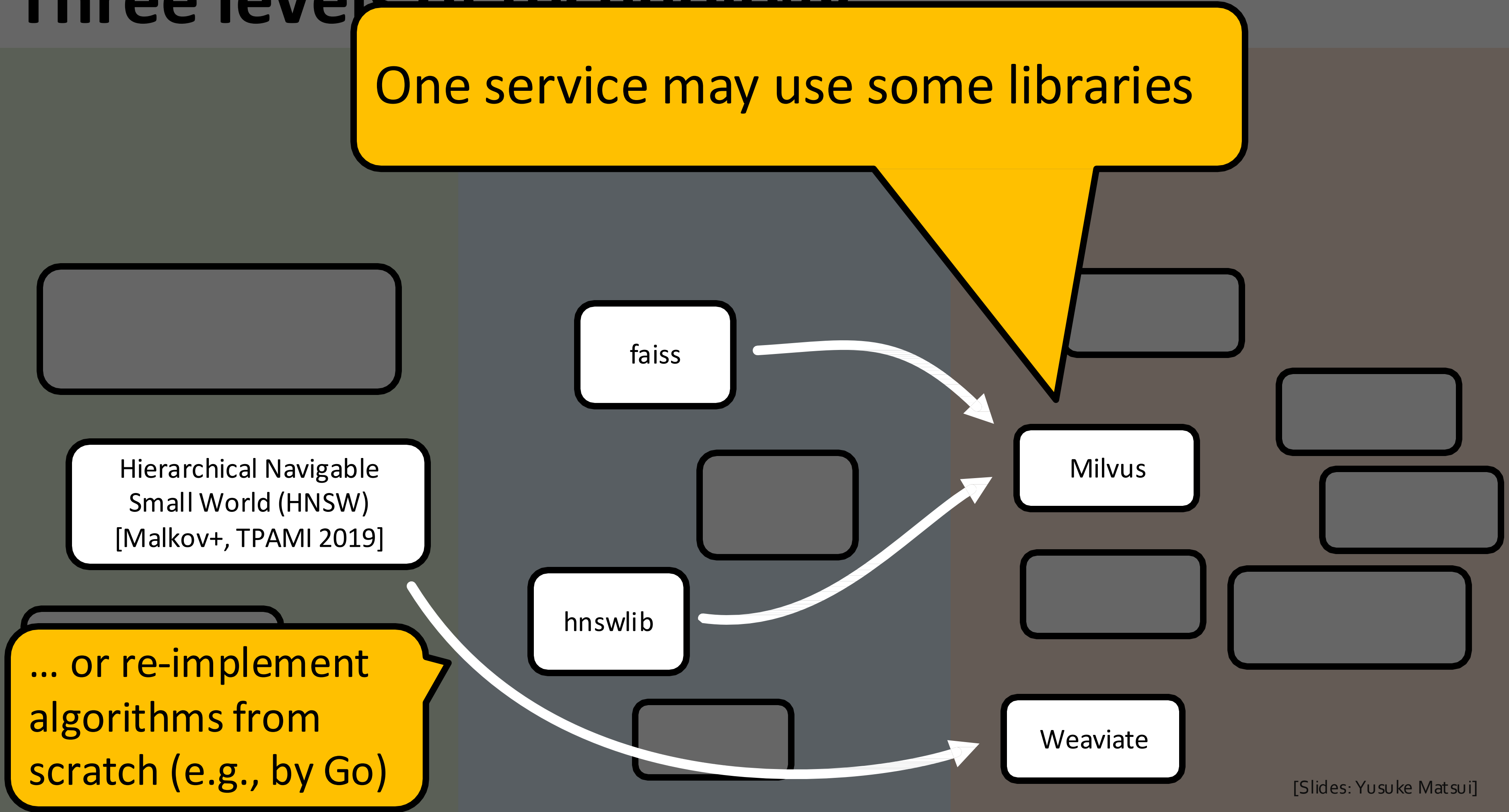
Three levels of technology

One algorithm may be implemented in multiple libraries



[Slides: Yusuke Matsui]

Three levels of technology



[Slides: Yusuke Matsui]

Let’s assume that we have our retrieval engine and data ready

Defining Similarity Metrics: Approach 1

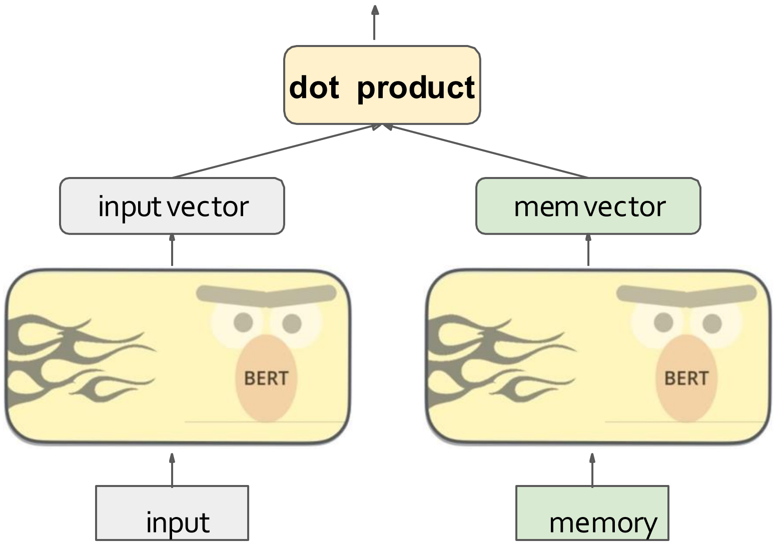
`sim(I,M)= tf(I, M) x log N/d_I`

- `tf(I, M)`: # of occurrences of I in M.
- `N`: # of documents
- `d_I`: # of documents that contain I.

- **Advantages:**
  - Not differentiable; you can't easily optimize it.
- **Disadvantages:**
  - Simple and fast. Often the first idea that you should try!

Defining Similarity Metrics: Approach 2

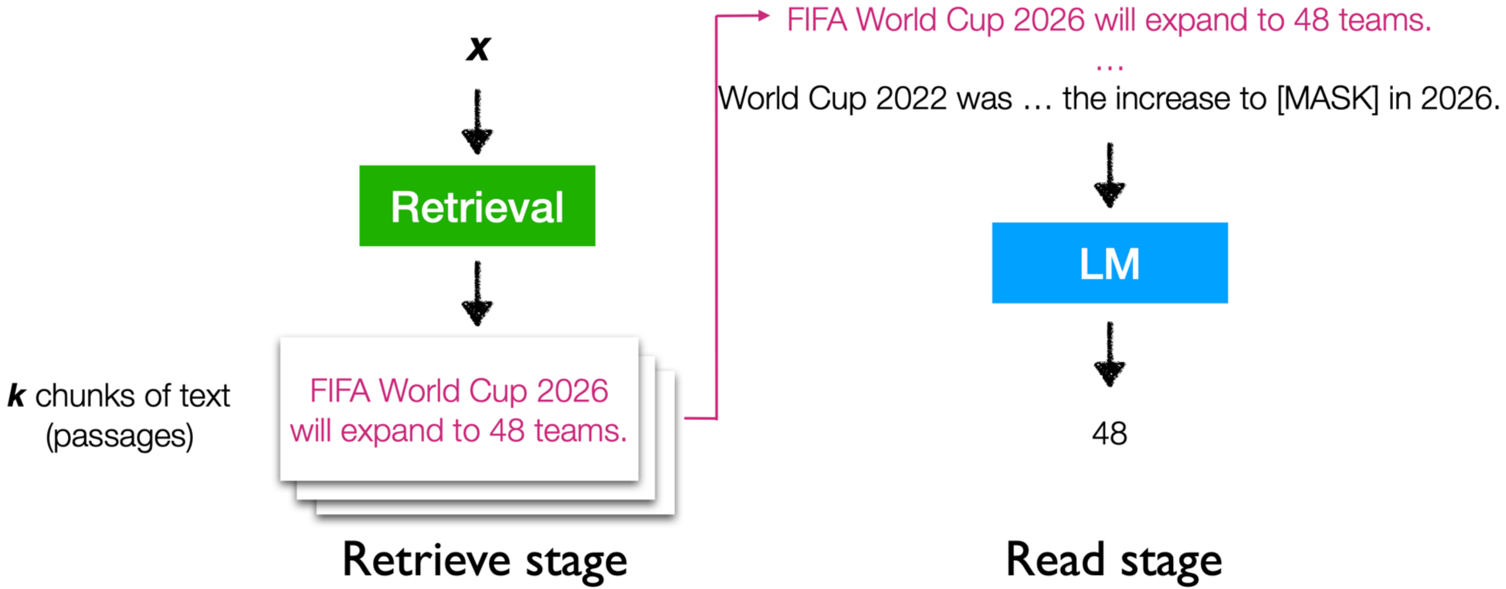
`sim(I,M)= Encoder(I) x Encoder(M)`



- **Advantages:**
  - Differentiable -- can optimize with gradient descent.
- **Disadvantages:**
  - Works well for data on which your LM is pre-trained on.

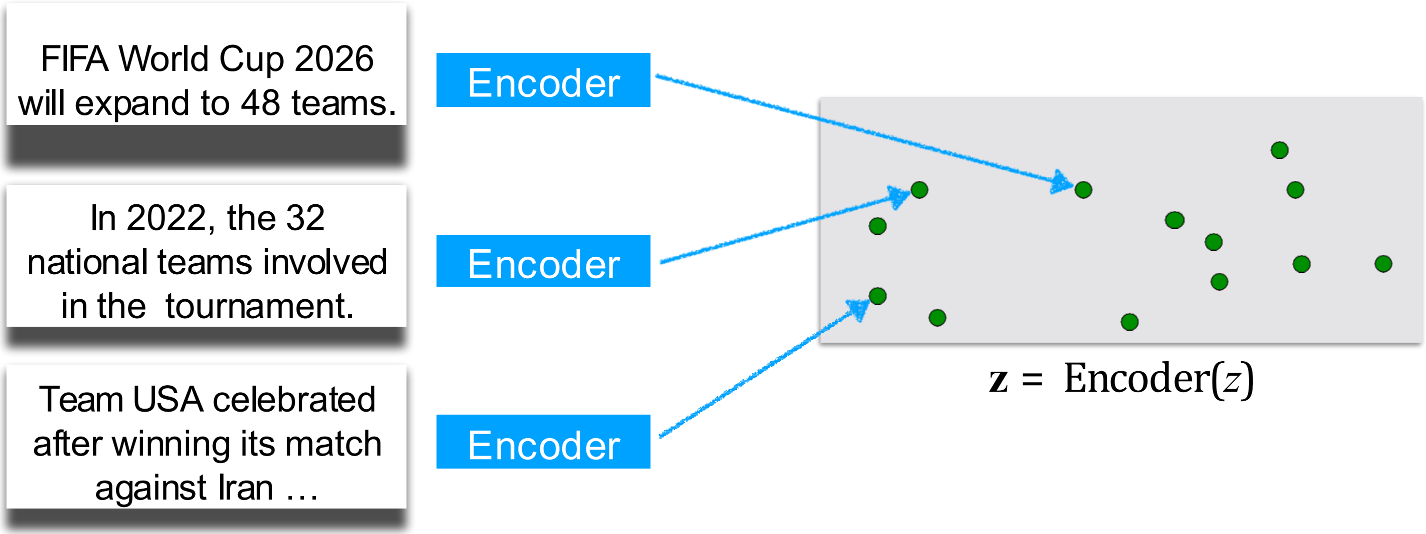
Retrieval-Augmented LM

- ***x*** = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.



[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Retrieval-Augmented LM

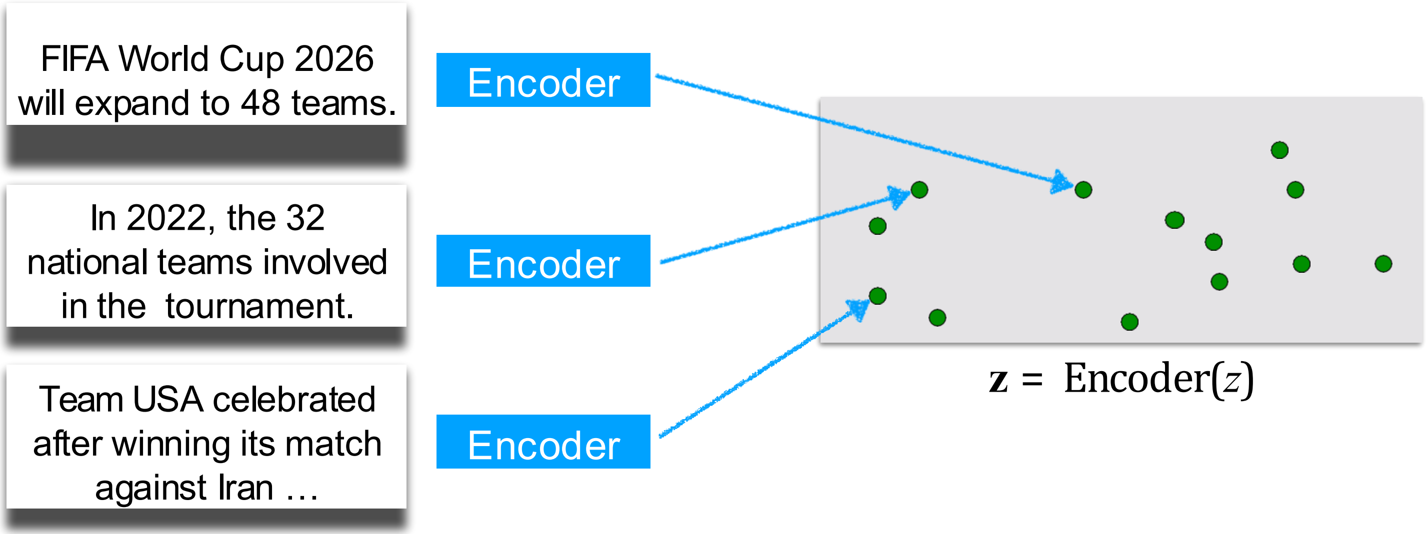


Wikipedia
13M chunks (passages)
(called *documents* in the paper)

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Retrieval-Augmented LM

$\boldsymbol{x}$ = World Cup 2022 was … the increase to [MASK] in 2026.



Wikipedia
13M chunks (passages)
(called *documents* in the paper)

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Retrieval-Augmented LM

$\boldsymbol{x}$ = World Cup 2022 was … the increase to [MASK] in 2026.

Encoder

FIFA World Cup 2026 will expand to 48 teams.

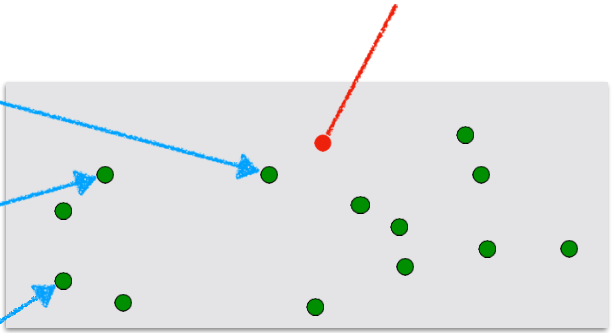
Encoder

In 2022, the 32 national teams involved in the tournament.

Encoder

Team USA celebrated after winning its match against Iran …

Encoder



$$\mathbf{z} = \text{Encoder}(z)$$
$$\mathbf{x} = \text{Encoder}(x)$$

Wikipedia
13M chunks (passages)
(called *documents* in the paper)

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

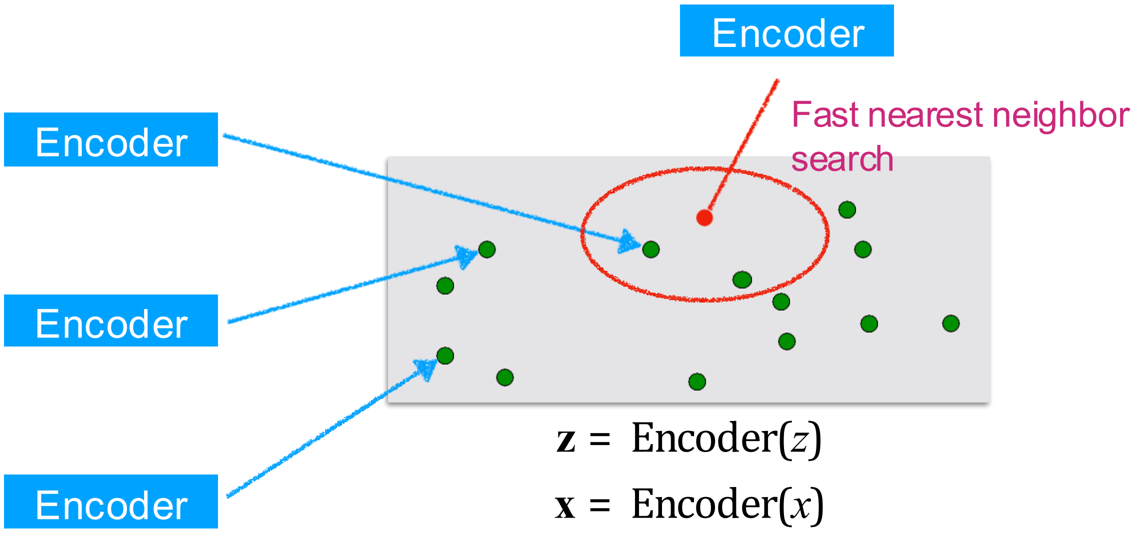
Retrieval-Augmented LM

x = World Cup 2022 was … the increase to [MASK] in 2026.

FIFA World Cup 2026 will expand to 48 teams.

In 2022, the 32 national teams involved in the tournament.

Team USA celebrated after winning its match against Iran …



$$\mathbf{z} = \text{Encoder}(z)$$

$$\mathbf{x} = \text{Encoder}(x)$$

Wikipedia
13M chunks (passages)
(called *documents* in the paper)

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Retrieval-Augmented LM

x = World Cup 2022 was … the increase to [MASK] in 2026.

Encoder

FIFA World Cup 2026 will expand to 48 teams.

Encoder

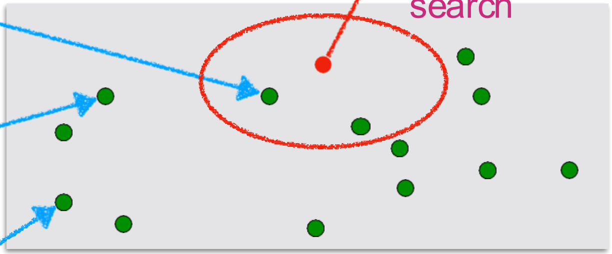
In 2022, the 32 national teams involved in the tournament.

Encoder

Team USA celebrated after winning its match against Iran …

Encoder

Fast nearest neighbor search



$$\mathbf{z} = \text{Encoder}(z)$$

$$\mathbf{x} = \text{Encoder}(x)$$

$$z_1, \ldots, z_k = \text{argTop-}k\,(\mathbf{x} \cdot \mathbf{z})$$

k retrieved chunks

Wikipedia
13M chunks (passages)
(called *documents* in the paper)

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Retrieval-Augmented LM

$x + z_1$ → LM → $P(y \mid x, z_1)$

$x + z_2$ → LM → $P(y \mid x, z_2)$

⋮

$x + z_k$ → LM → $P(y \mid x, z_k)$

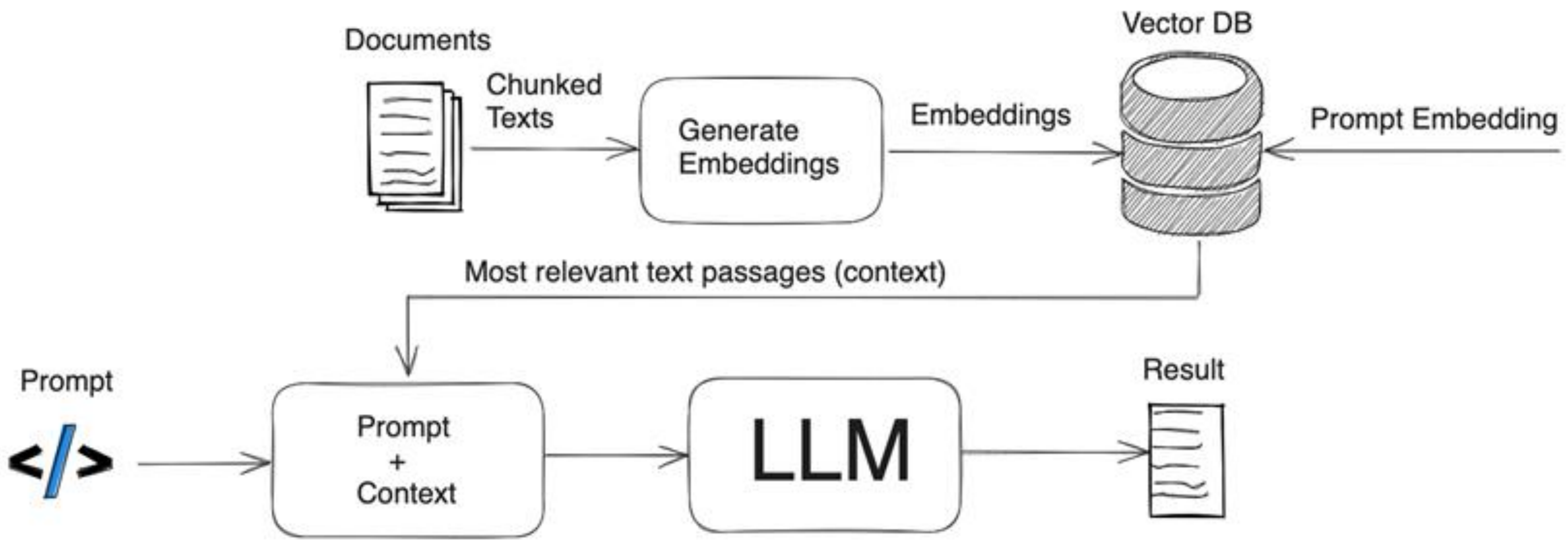
0 if not one of top k

$$\sum_{z \in \mathcal{D}} \underbrace{P(z \mid x)}_{\text{from the retrieve stage}} \underbrace{P(y \mid x, z)}_{\text{from the read stage}}$$

Need to approximate
→ Consider top k chunks only

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

RAG: A more simplified pipeline



Retrieval-Augmented LM: Variants

What to retrieve?

- Chunks
- Tokens
- Others

How to use retrieval?

- Input layer
- Intermediate layers
- Output layer

When to retrieve?

- Once
- Every *n* tokens (*n*>1)
- Every token

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

Retrieval-Augmented LM: Common Variant

What to retrieve?

- **Chunks**
- Tokens
- Others

How to use retrieval?

- **Input layer**
- Intermediate layers
- Output layer

When to retrieve?

- **Once**
- Every *n* tokens (*n*>1)
- Every token

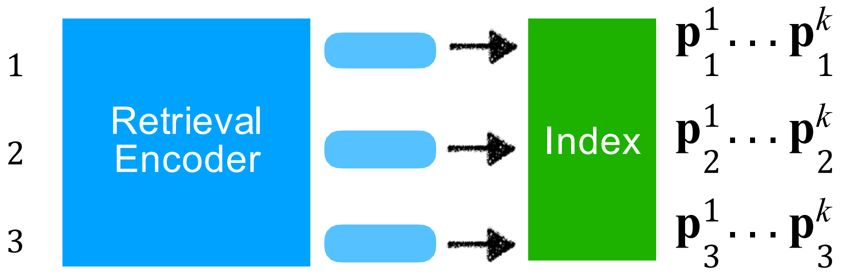
[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

IR in the Middle of LM

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

1 2 3

(k chunks of text per split)



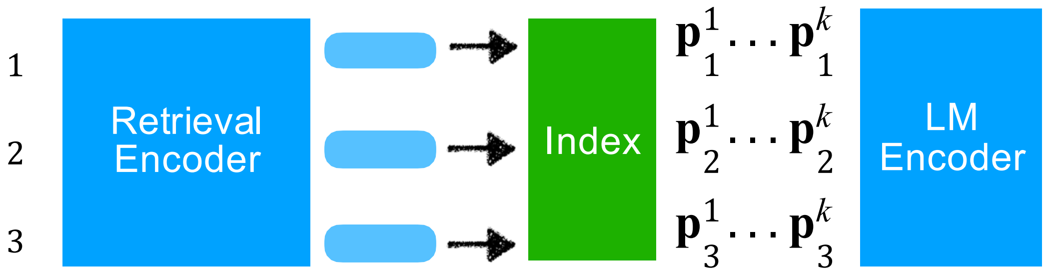
Borgeaud et al. 2021. "Improving language models by retrieving from trillions of tokens"

IR in the Middle of LM

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

1 2 3

(k chunks of text per split)



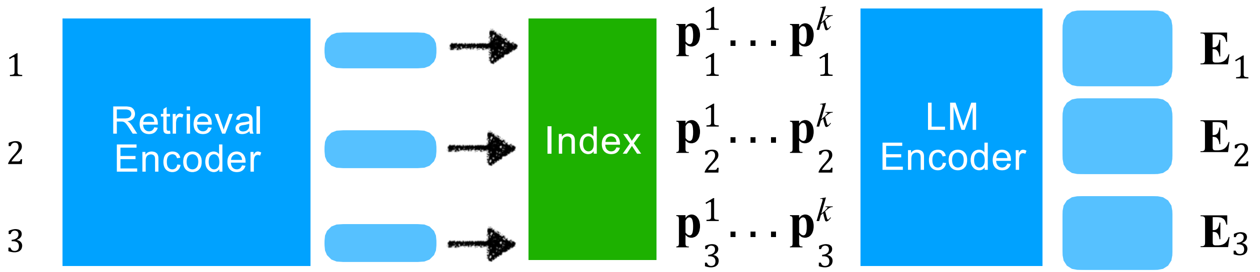
Borgeaud et al. 2021. "Improving language models by retrieving from trillions of tokens"

IR in the Middle of LM

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

1 2 3

(k chunks of text per split)

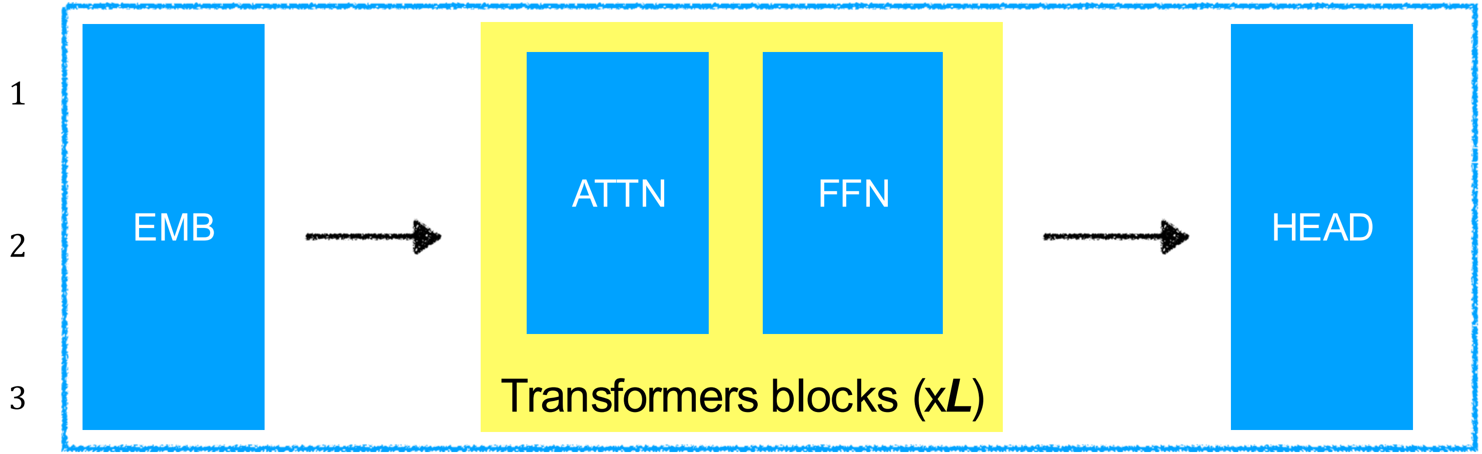


(A $r \times k \times d$ matrix)

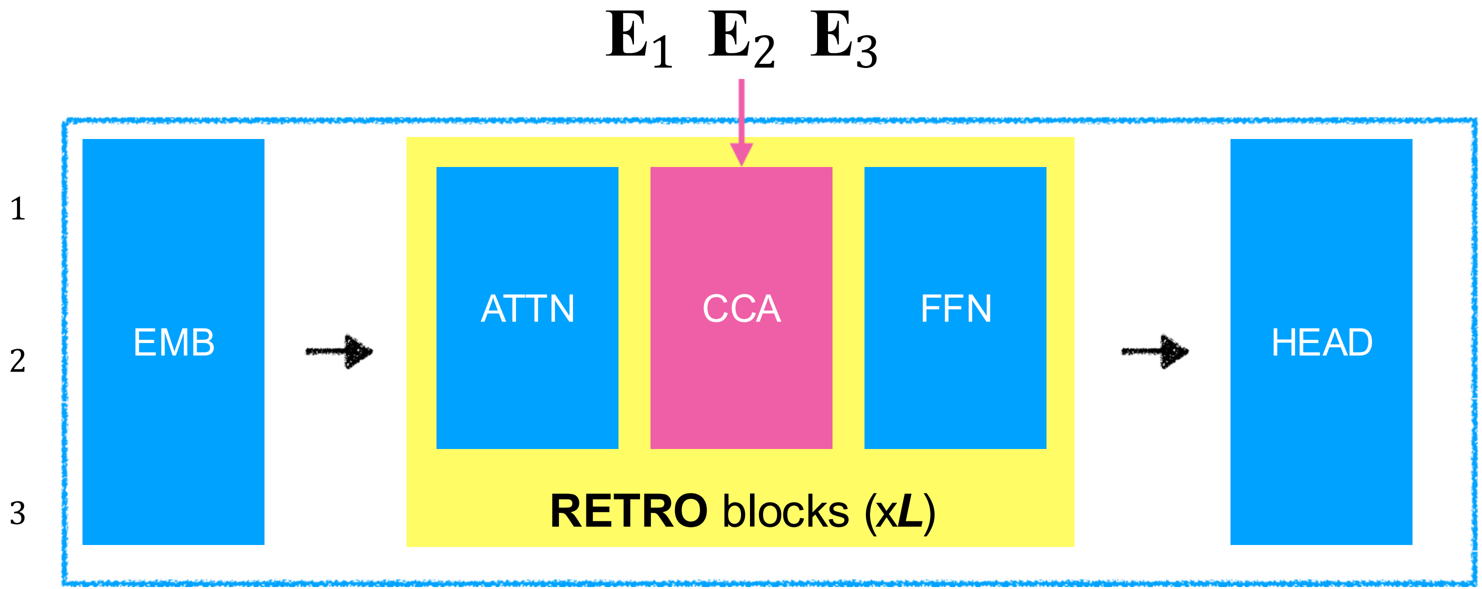
(r = # tokens per text chunk)
(d = hidden dimension)
(k = # retrieved chunks per split)

Borgeaud et al. 2021. "Improving language models by retrieving from trillions of tokens"

Regular Decoder



Regular Decoder with IR Embeddings



Chunked Cross Attention (CCA)

Results

Perplexity: The lower the better

Model	Retrieval Set	#Database tokens	#Database keys	Valid	Test
Adaptive Inputs (Baevski and Auli, 2019)	-	-	-	17.96	18.65
SPALM (Yogatama et al., 2021)	Wikipedia	3B	3B	17.20	17.60
kNN-LM (Khandelwal et al., 2020)	Wikipedia	3B	3B	16.06	16.12
Megatron (Shoeybi et al., 2019)	-	-	-	-	10.81
Baseline transformer (ours)	-	-	-	21.53	22.96
kNN-LM (ours)	Wikipedia	4B	4B	18.52	19.54
RETRO	Wikipedia	4B	0.06B	18.46	18.97
RETRO	C4	174B	2.9B	12.87	10.23
RETRO	MassiveText (1%)	18B	0.8B	18.92	20.33
RETRO	MassiveText (10%)	179B	4B	13.54	14.95
RETRO	MassiveText (100%)	1792B	28B	**3.21**	**3.92**

Significant improvements by retrieving from 1.8 trillion tokens

Results

Perplexity: The lower the better

Model	Retrieval Set	#Database tokens	#Database keys	Valid	Test
Adaptive Inputs (Baevski and Auli, 2019)	-	-	-	17.96	18.65
SPALM (Yogatama et al., 2021)	Wikipedia	3B	3B	17.20	17.60
kNN-LM (Khandelwal et al., 2020)	Wikipedia	3B	3B	16.06	16.12
Megatron (Shoeybi et al., 2019)	-	-	-	-	10.81
Baseline transformer (ours)	-	-	-	21.53	22.96
kNN-LM (ours)	Wikipedia	4B	4B	18.52	19.54
RETRO	Wikipedia	4B	0.06B	18.46	18.97
RETRO	C4	174B	2.9B	12.87	10.23
RETRO	MassiveText (1%)	18B	0.8B	18.92	20.33
RETRO	MassiveText (10%)	179B	4B	13.54	14.95
RETRO	MassiveText (100%)	1792B	28B	**3.21**	**3.92**

Significant improvements by retrieving from 1.8 trillion tokens

JOHNS HOPKINS
WHITING SCHOOL of ENGINEERING

Retrieval-Augmented LM

What to retrieve?

- **Chunks**
- Tokens
- Others

How to use retrieval?

- Input layer
- **Intermediate layers**
- Output layer

When to retrieve?

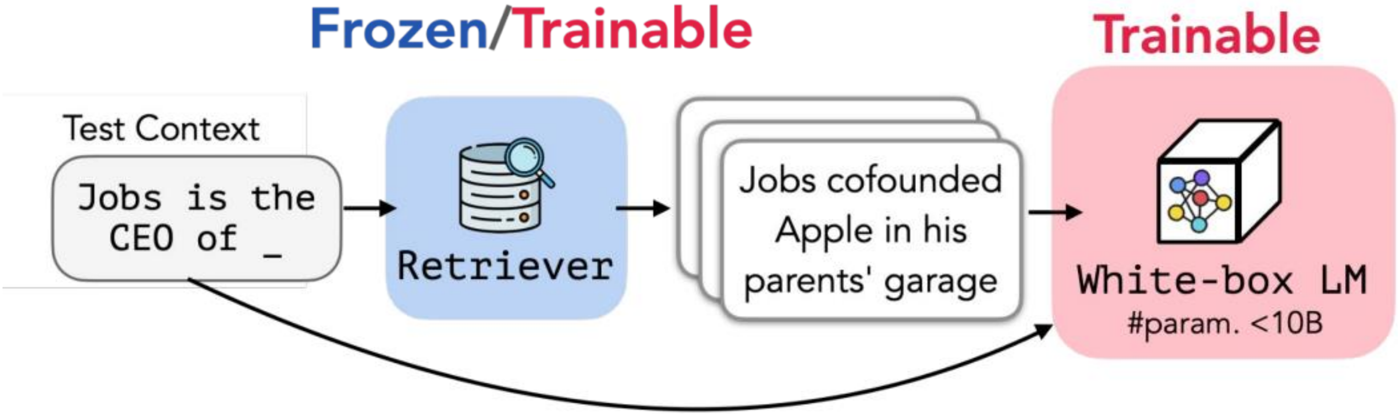
- Once
- **Every *n* tokens (*n*>1)**
- Every token

[Slides: Akari Asai, Sewon Min, Zexuan Zhong, Danqi Chen]

How do you train these models?

End-to-end Training

- There are various ideas in the literature for how to train these models efficiently and in an end-to-end fashion.



Main takeaways

- How do we enable LMs to utilize external knowledge?
  - Retrieval-augmented language models
- A retriever is a function, f(input, memory) → score
- What we did not discuss:
  - Attribution: Tracing decisions to the source knowledge
  - How to modify the knowledge
  - Conflicting knowledge
  - Editing knowledge
  - More efficient scaling
  - ....

Practical tips for building IR systems

- Try the easiest approach.
- For example, you can fire up an Elastic Search IR with a single line of command:
- …
- And here is how you query.

Practical tips for building IR systems

- Try the easiest approach.
- For example, you can fire up an Elastic Search IR with a single line of command:
- …
- And here is how you query.

Always try the easiest test

- Sample sentences from indexed corpus and see if the IR engine is able to retrieve them.
    - (may be useful to define a few retrieval metrics here)
- If it’s robust, try queries that are perturbations of the indexed content.

Retrieval-Augmented Generation Limitations

- **However, RAG has several limitations:**
- **Extra complexity**
  - It needs to define what to store in the database.
- **Lacks contextual understanding.**
  - Summarization / Complex Reading Comprehension
- **Stateless.**
  - It does not hold memory of what is previously retrieved.

How were retrieval engines designed before NNs?

Classic Information Retrieval

```
t = token
d = document (movie review)
term_frequency(t,d) = number of times t occurs in d
document_frequency(t) = # documents t occurs in
N = number of documents
inverse_document_frequency(t) = N / document_frequency(t)
tf-idf(t,d) = tf(t,d) x inverse_document_frequency(t)
```

- A vector for d is produced by stacking tf-idf(t,d) in an order determined by the toke vocab IDs
- **BM25** adds two parameters: k, a knob that adjust the balance between term frequency and IDF, and b, which controls the importance of document length normalization [Kamphuis et al. (2020)]
- We can represent a question and each document in the external databases like this, and use vector similarity measurements to find the most similar documents for a given question

Inverted Index

```
t = token
d = document (movie review)
term_frequency(t,d) = number of times t occurs in d
document_frequency(t) = # documents t occurs in
N = number of documents
inverse_document_frequency(t) = N / document_frequency(t)
tf-idf(t,d) = tf(t,d) x inverse_document_frequency(t)
```

- We need to efficiently find documents that contain words in the question
- The basic search problem in IR is thus to find all documents that contain a term
- The data structure for this task is the **inverted index**, which we use for making this search efficient, and also conveniently storing useful information like the document frequency and the count of each term in each document

[Slide credit: Ana Marasovic]

Inverted Index (cont.)

- An **inverted index**, given a query term, gives a list of documents that contain the term
- It consists of two parts:
  - Dictionary
  - Postings
- **Dictionary** is a list of terms (designed to be efficiently accessed), each pointing to a postings list for the term.
  - The dictionary can also store the document frequency for each term
- A **postings list** is the list of document IDs associated with each term
  - It can also contain information like the term frequency or even the exact positions of terms in the document

In how many documents this term appears

how $\{1\} \rightarrow 3\ [1]$
is $\{1\} \rightarrow 3\ [1]$
love $\{2\} \rightarrow 1\ [1] \rightarrow 3\ [1]$
nurse $\{2\} \rightarrow 1\ [1] \rightarrow 4\ [1]$
sorry $\{1\} \rightarrow 2\ [1]$
sweet $\{3\} \rightarrow 1\ [2] \rightarrow 2\ [1] \rightarrow 3\ [1]$

Document ID

The count of the term in this document

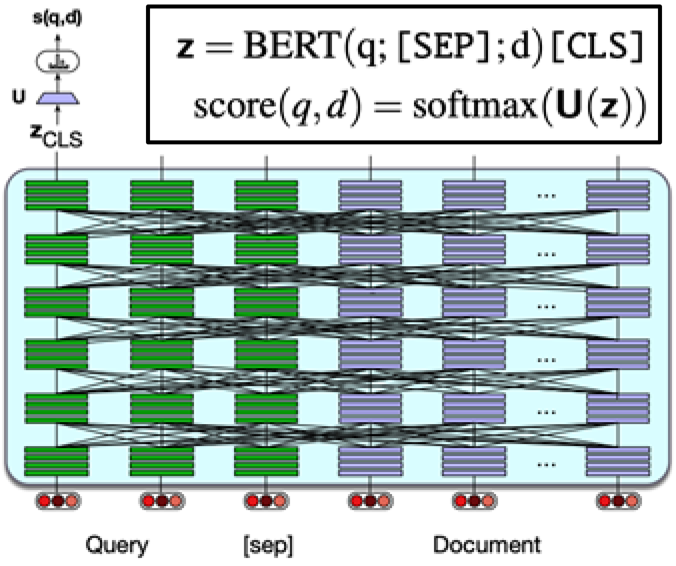
Figure: [Jurafsky and Martin]

Why could this classic IR be limited?

- tf-idf/BM25 algorithms work only if there is exact overlap of words between the question and document
- What if a question contains a lot of synonyms of tokens in a relevant document?
- **Vocabulary mismatch problem:** The user posing a query (or asking a question) needs to guess exactly what words the writer of the answer might have used
  - Instead of (sparse) word-count vectors, using (dense) embeddings

Figure: [Jurafsky and Martin]

Two ways to dense retrieval



(a)

Problem?

Figure 14.7 Two ways to do dense retrieval, illustrated by using lines between layers to schematically represent self-attention: (a) Use a single encoder to jointly encode query and document and finetune to produce a relevance score with a linear layer over the CLS token. This is too compute-expensive to use except in rescoring (b) Use separate encoders for query and document, and use the dot product between CLS token outputs for the query and document as the score. This is less compute-expensive, but not as accurate.

Two ways to dense ret

$$\mathbf{z}_q = \mathrm{BERT}_Q(\mathrm{q})[\mathrm{CLS}]$$
$$\mathbf{z}_d = \mathrm{BERT}_D(\mathrm{d})[\mathrm{CLS}]$$
$$\mathrm{score}(q,d) = \mathbf{z}_q \cdot \mathbf{z}_d$$

Figure: [Jurafsky and Martin]

$$\mathbf{z} = \mathrm{BERT}(\mathrm{q};[\mathrm{SEP}];\mathrm{d})[\mathrm{CLS}]$$
$$\mathrm{score}(q,d) = \mathrm{softmax}(\mathbf{U}(\mathbf{z}))$$

[DPR; Karpukhin et al., 2020]

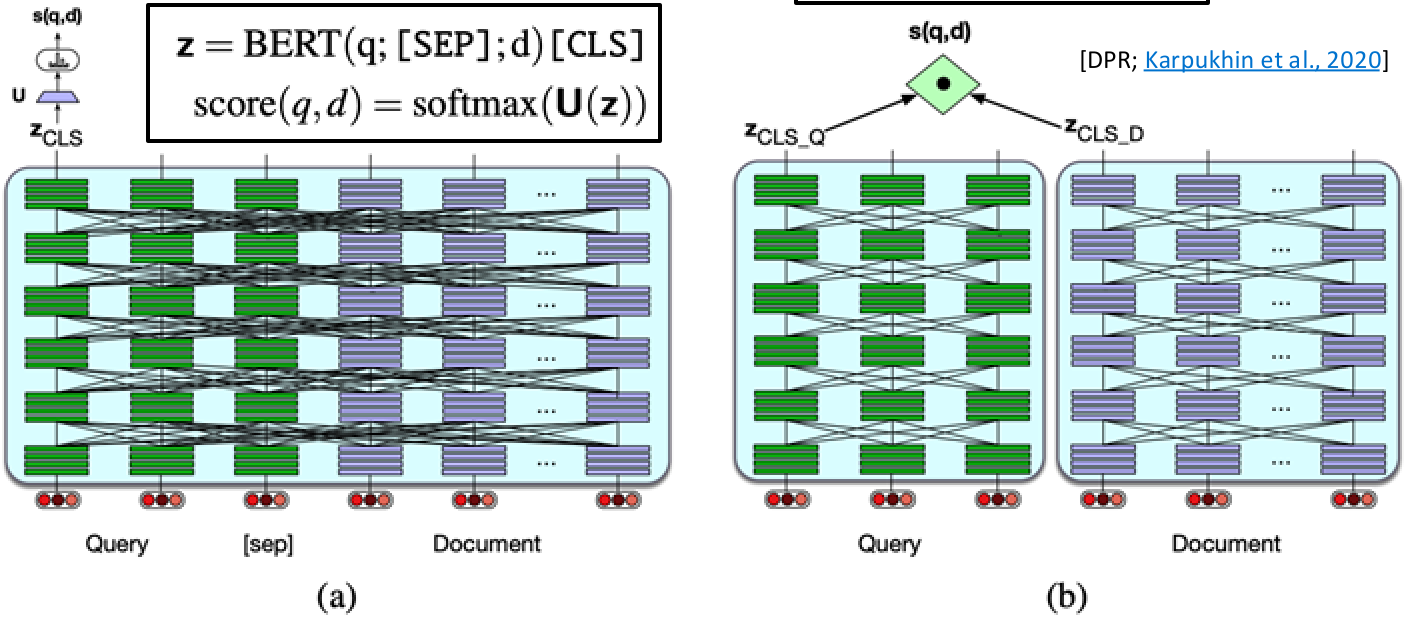


Figure 14.7 Two ways to do dense retrieval, illustrated by using lines between layers to schematically represent self-attention: (a) Use a single encoder to jointly encode query and document and finetune to produce a relevance score with a linear layer over the CLS token. This is too compute-expensive to use except in rescoring (b) Use separate encoders for query and document, and use the dot product between CLS token outputs for the query and document as the score. This is less compute-expensive, but not as accurate.

Another way to the second approach – ColBERT

[Khattab et al., 2021]

$$\text{score}(q,d) = \sum_{i=1}^{N} \max_{j=1}^{m} \mathbf{E}_{q_i} \cdot \mathbf{E}_{d_j}$$

BERT output vectors rescaled to unit length

Essentially, for each token in q, ColBERT finds the most contextually similar token in d, & then sums up these similarities

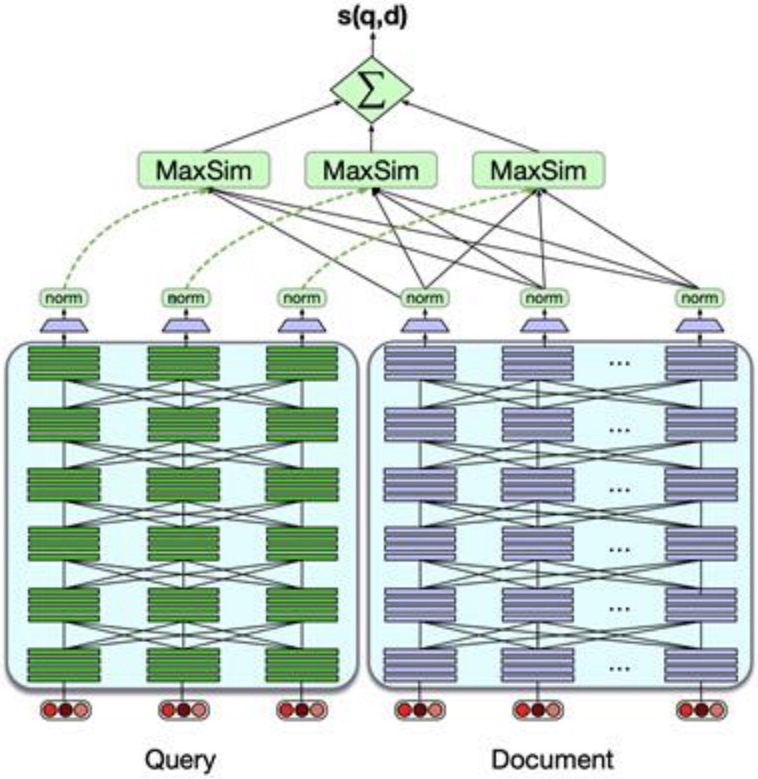
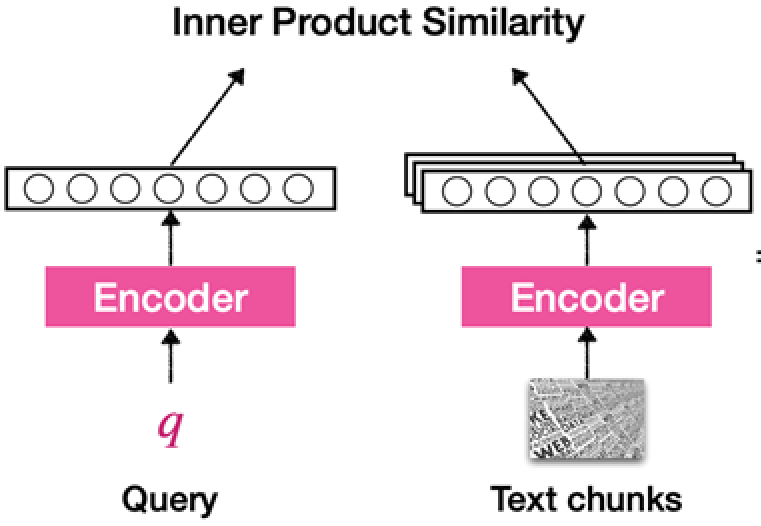


Figure: [Jurafsky and Martin]

132

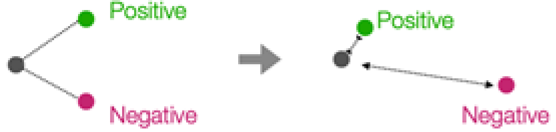
Training dense retrievals

- Train the model to maximize the score for positive documents and minimize the score for negative ones:



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

Contrastive learning



No "Free Lunch" in Long Context

- No single approach is universally optimal for long context learn
- There is always a tradeoff between performance and cost.
- It makes long context learning a challenging problem.