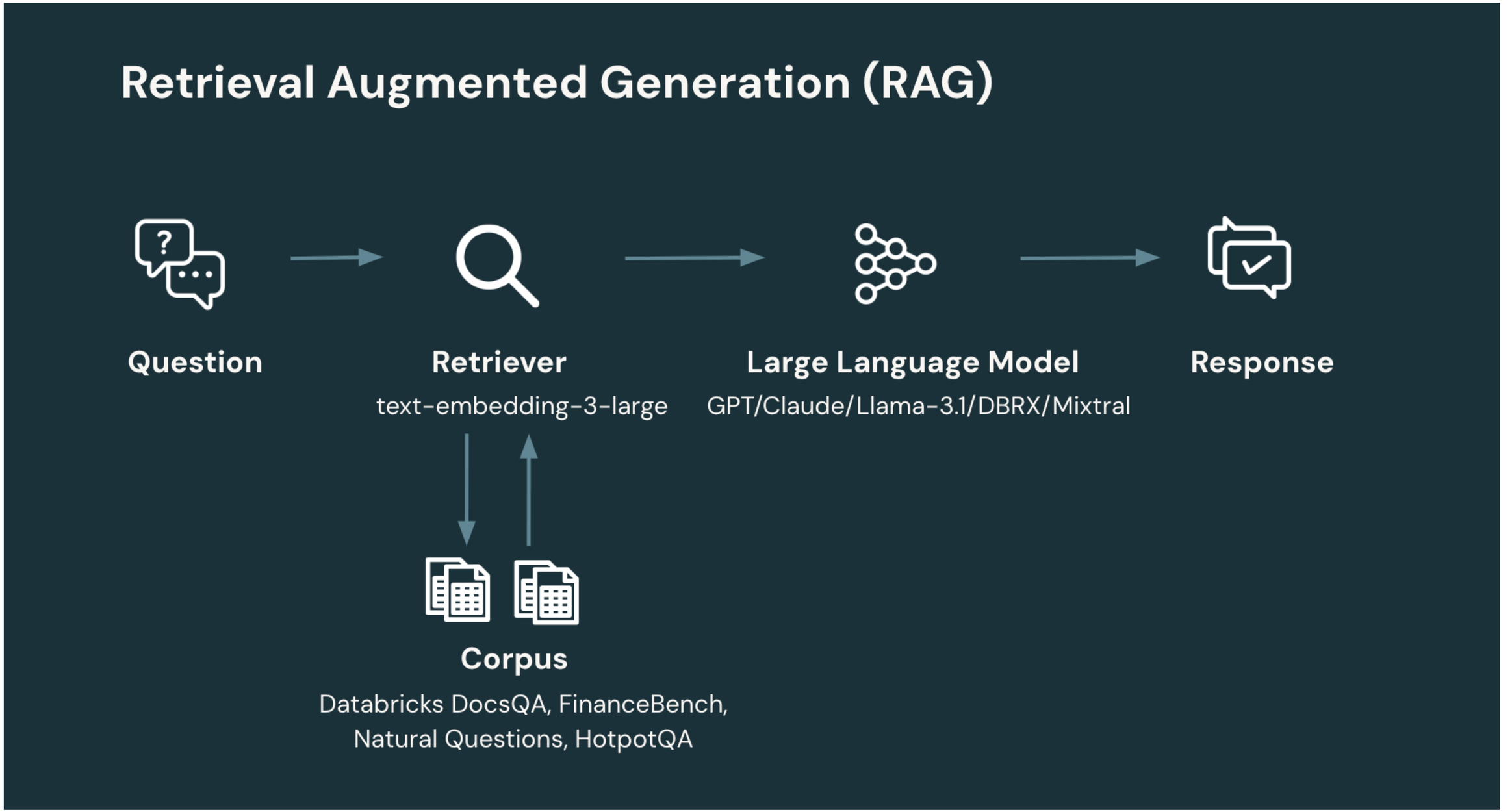
CMPT 413/713: Natural Language Processing

Retrieval and RAG

Spring 2026

2026-03-18

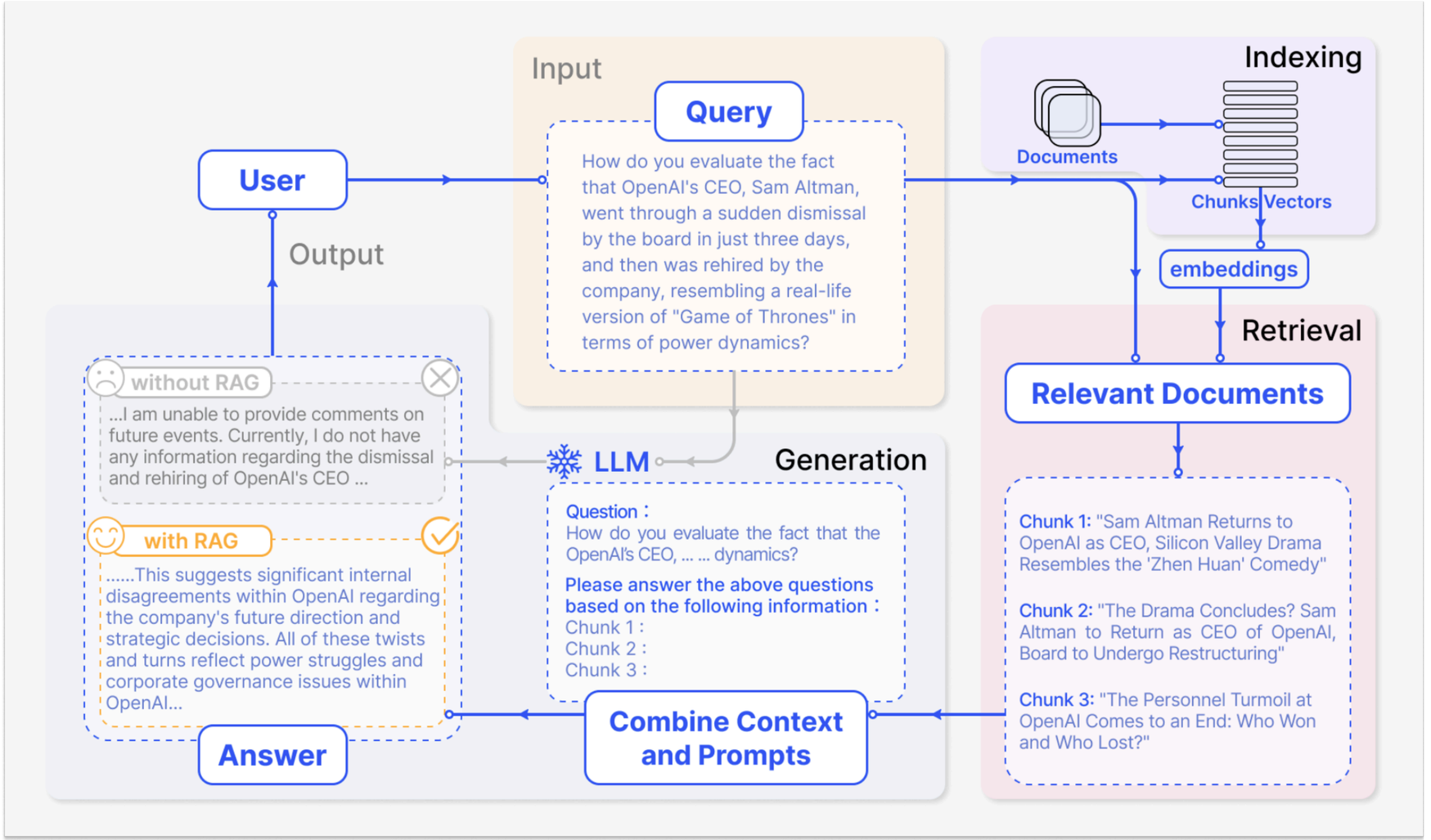
Some slides adapted from Anoop Sarkar, Graham Neubig



https://www.databricks.com/blog/long-context-rag-performance-llms

Retrieval-augmented generation

- RAG for prompting: Retrieved documents / chunks are used to augment prompts



Retrieval-Augmented Generation for Large Language Models: A Survey [Gao et al. 2024]

Retrieval

- Use inverted index for efficient lookups
- Software: Apache Lucene

Traditional retrieval engines

- Retrieval using sparse vectors of terms found in query
- TF-IDF (term-frequency / inverse document frequency)

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

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

- BM25 — TF term with 1) non-linear saturation of counts and 2) accounts for document length (e.g. long documents will have more terms)

Hyperparameters $k_1 \in [1.2,2], b = 0.75$

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

|d} is the document length (in tokens)

Average document length

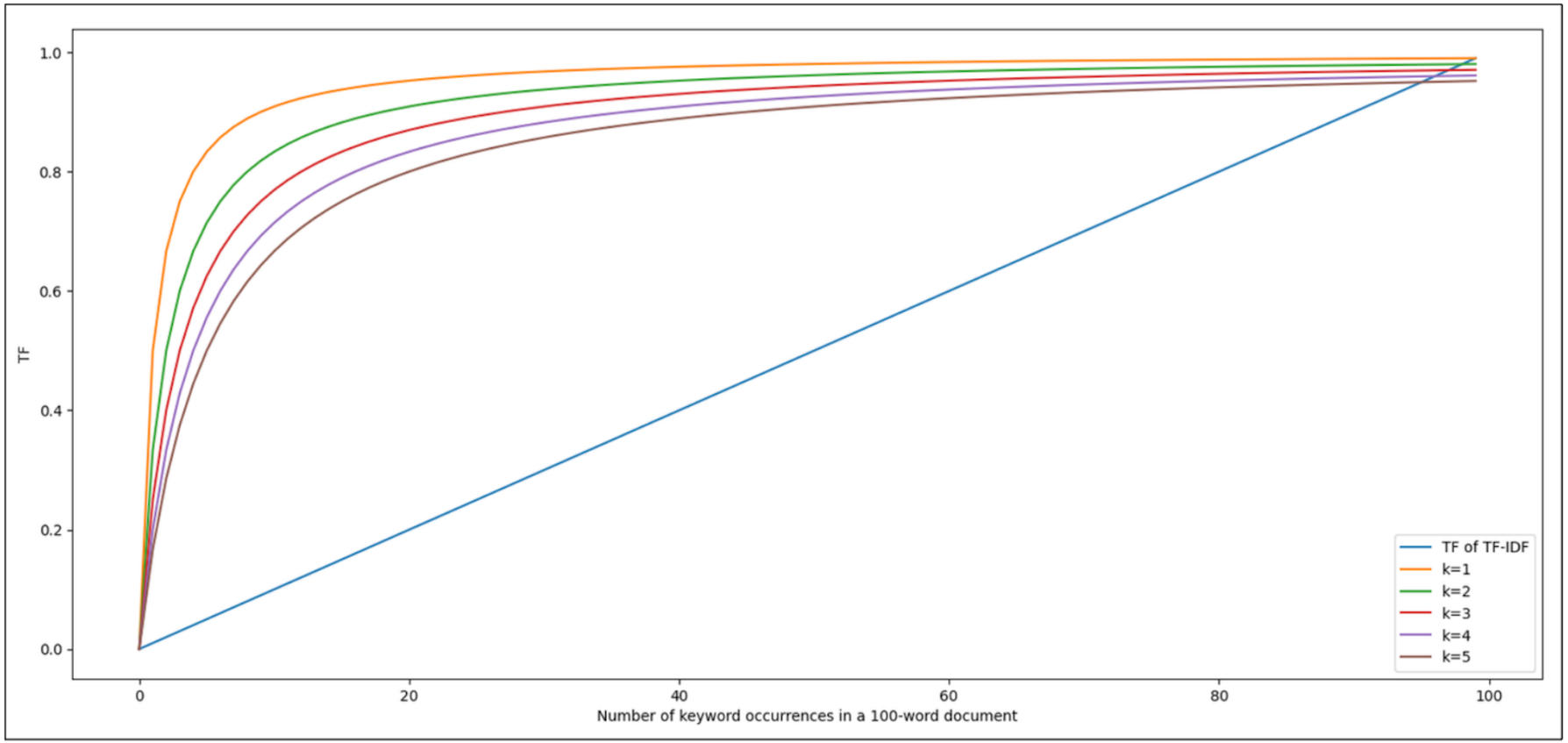
BM-25

$$\text{IDF} = \log \frac{|D| - DF(t) + 0.5}{DF(t) + 0.5}$$

$$\text{TF}(t, d) = \frac{\text{freq}(t, d)}{\text{freq}(t, d) + k}$$

- Few occurrences of a term is more meaningful
- After a point, the impact of increasing counts of a term saturates

Comparison of term frequencies with different values of k

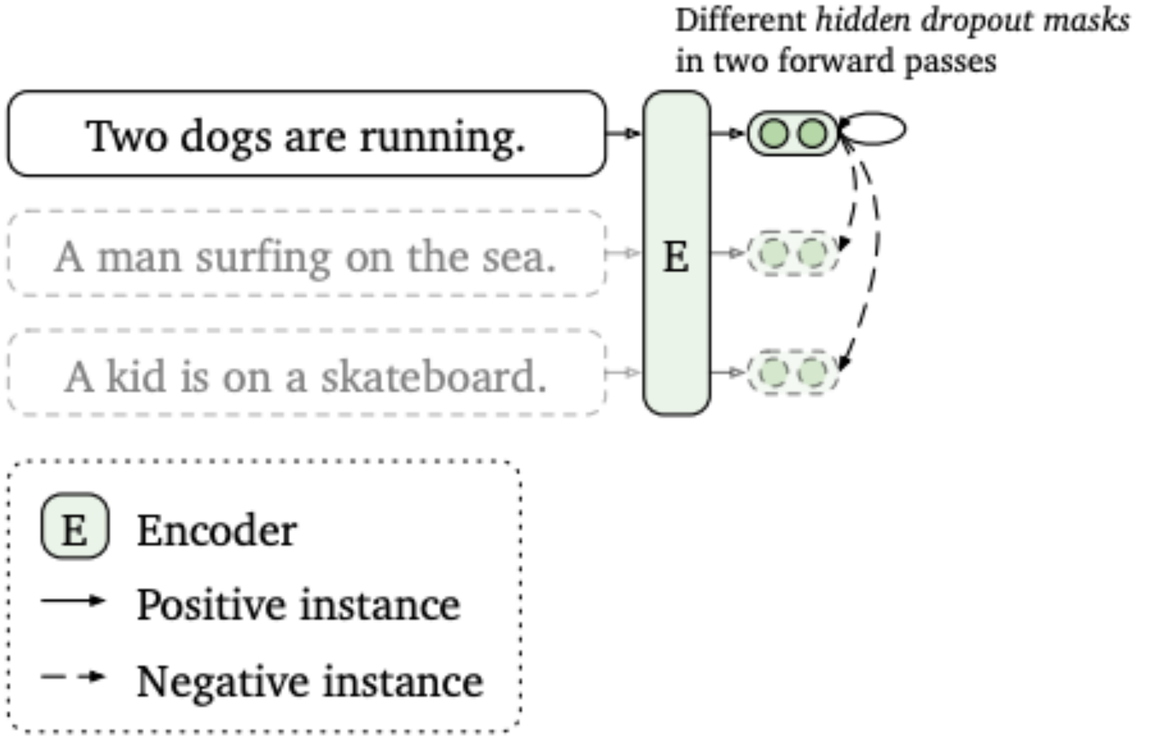


https://zilliz.com/learn/mastering-bm25-a-deep-dive-into-the-algorithm-and-application-in-milvus

Retrieval with dense embeddings

How to get good embeddings?

- Encode document and query with dense embeddings
  - Take embedding for [CLS] / last token
  - Pool all tokens
- What are good encoders to use?
  - Causal LLMs only do unidirectional encoding so is weaker than a bidirectional encoder.
  - Bidirectional encoding with masked language model: BERT, RoBERTa

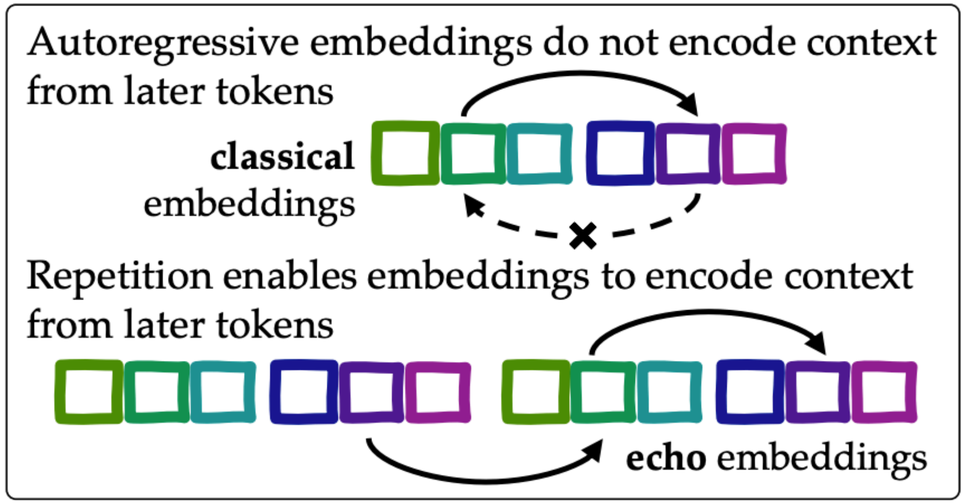


SimCSE [Gao et al 2021]

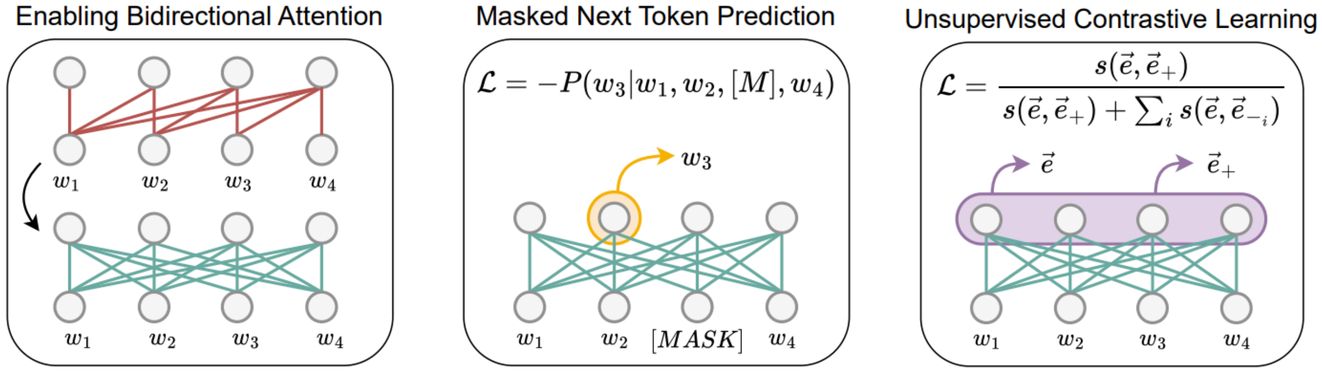
Retrieval with dense embeddings

How to get good embeddings?

- Can we get **bidirectional** information from causal LLMs?
  - Echo embeddings [Springer et al 2024]. Repeat the string multiple times in a unidirectional model
  - LLM2Vec [BehnamGhader et al. 2024], train unidirectional, then remove causal masking and use / train



Echo embeddings [Springer et al 2024]



LLM2Vec [BehnamGhader et al 2024]

Retrieval with dense embeddings

- Evaluation (unsupervised) on massive text embedding benchmark (MTEB)

Categories → **# of datasets →**	**Retr.** 15	**Rerank.** 4	**Clust.** 11	**PairClass.** 3	**Class.** 12	**STS** 10	**Summ.** 1	**Avg** 56
`Encoder-only`								
BERT	10.59	43.44	30.12	56.33	61.66	54.36	29.82	38.33
BERT + SimCSE	20.29	46.47	29.04	70.33	62.50	74.33	31.15	45.45
`S-LLaMA-1.3B`								
Uni + w. Mean	9.47	38.02	28.02	42.19	59.79	49.15	24.98	35.05
LLM2Vec (w/o SimCSE)	15.48	40.99	31.83	50.63	64.54	62.06	26.82	41.43
LLM2Vec	25.93	47.70	37.45	72.21	67.67	71.61	31.23	49.42
Echo	10.36	41.52	30.03	52.08	63.75	59.36	22.79	39.10

LLM2Vec [BehnamGhader et al 2024]

MTEB

https://arxiv.org/abs/2210.07316

MTEB

Massive Text Embedding Benchmark

- MTEB spans 8 embedding tasks covering a total of 58 datasets and 112 languages.
- The benchmark comprises 8 different tasks, with up to 15 datasets each.
- Of the 58 total datasets in MTEB, 10 are multilingual, covering 112 different languages
- Sentence-level and paragraph-level datasets are included to contrast performance on short and long texts. (tasks are split into S2S; P2P; S2P)
- New datasets for existing tasks can be benchmarked in MTEB via a single file that specifies the task and a Hugging Face dataset name

MTEB

Massive Text Embedding Benchmark

- Tasks
  - **Bitext mining**: Inputs are two sets of sentences from two different languages. For each sentence in the first set, the best match in the second set needs to be found.
  - **Classification**: A train and test set are embedded with the provided model. The train set embeddings are used to train a logistic regression classifier
  - **Clustering**: Given a set of sentences or paragraphs, the goal is to group them into meaningful clusters.
  - **Pair Classification**: A pair of text inputs is provided and a label needs to be assigned.
  - **Reranking**: Inputs are a query and a list of relevant and irrelevant reference texts. The aim is to rank the results according to their relevance to the query

MTEB

Massive Text Embedding Benchmark

- Tasks
  - **Retrieval**: Each dataset consists of a corpus, queries and a mapping for each query to relevant documents from the corpus. The aim is to find these relevant documents.
  - **Semantic Textual Similarity** (STS): Given a sentence pair the aim is to determine their similarity. Labels are continuous scores with higher numbers indicating more similar sentences
  - **Summarization**: A set of human-written and machine-generated summaries are provided. The aim is to score the machine summaries.

Speed vs performance

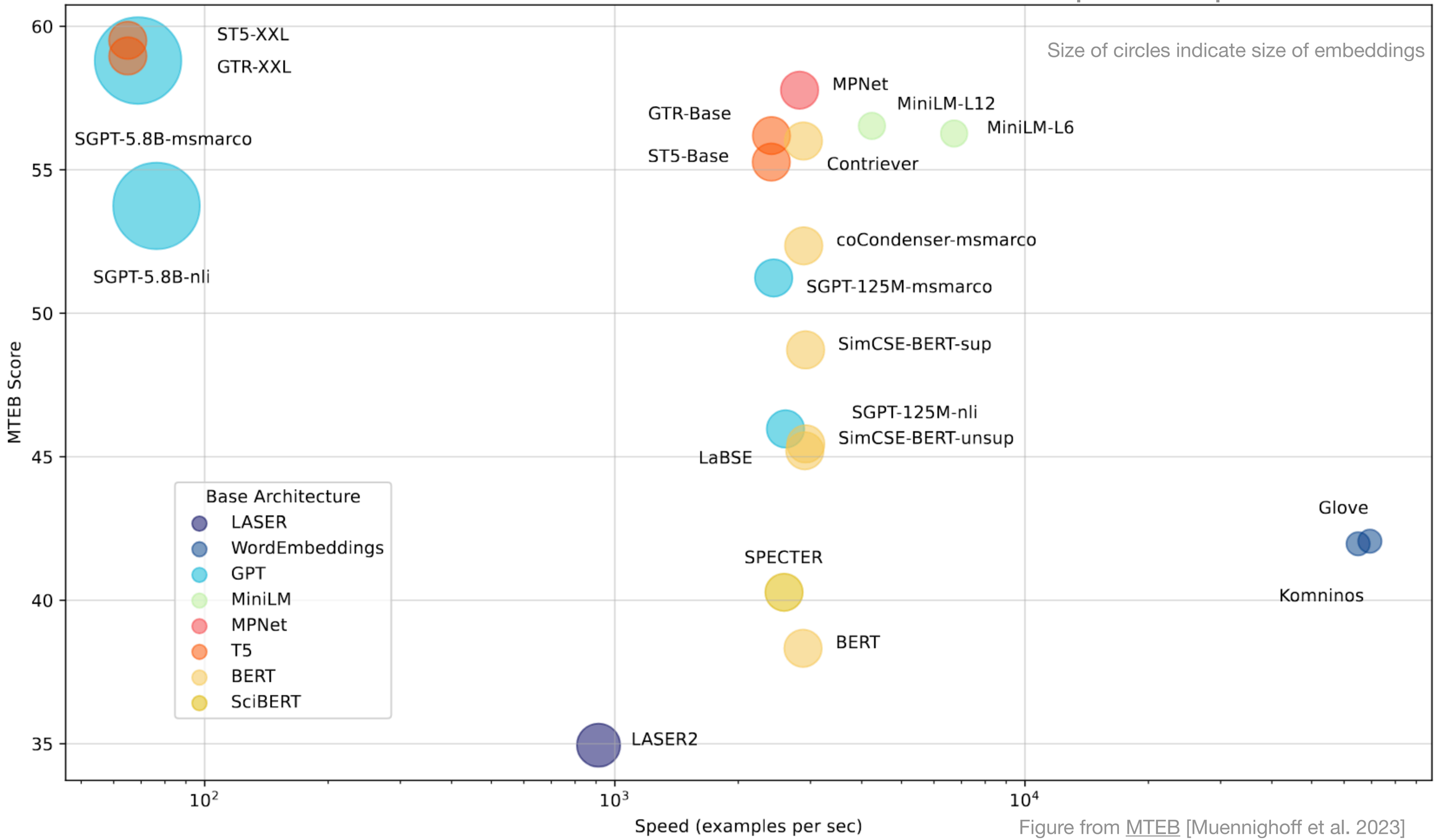


Figure from MTEB [Muennighoff et al. 2023]

Retrieval with dense embeddings

- Evaluation (unsupervised) on massive text embedding benchmark (MTEB)

Categories →	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg
# of datasets →	15	4	11	3	12	10	1	56
Encoder-only								
BERT	10.59	43.44	30.12	56.33	61.66	54.36	29.82	38.33
BERT + SimCSE	20.29	46.47	29.04	70.33	62.50	74.33	31.15	45.45
S-LLaMA-1.3B								
Uni + w. Mean	9.47	38.02	28.02	42.19	59.79	49.15	24.98	35.05
LLM2Vec (w/o SimCSE)	15.48	40.99	31.83	50.63	64.54	62.06	26.82	41.43
LLM2Vec	25.93	47.70	37.45	72.21	67.67	71.61	31.23	49.42
Echo	10.36	41.52	30.03	52.08	63.75	59.36	22.79	39.10

LLM2Vec [BehnamGhader et al 2024]

Retrieval with dense embeddings

- Performance improves with larger and better LLMs

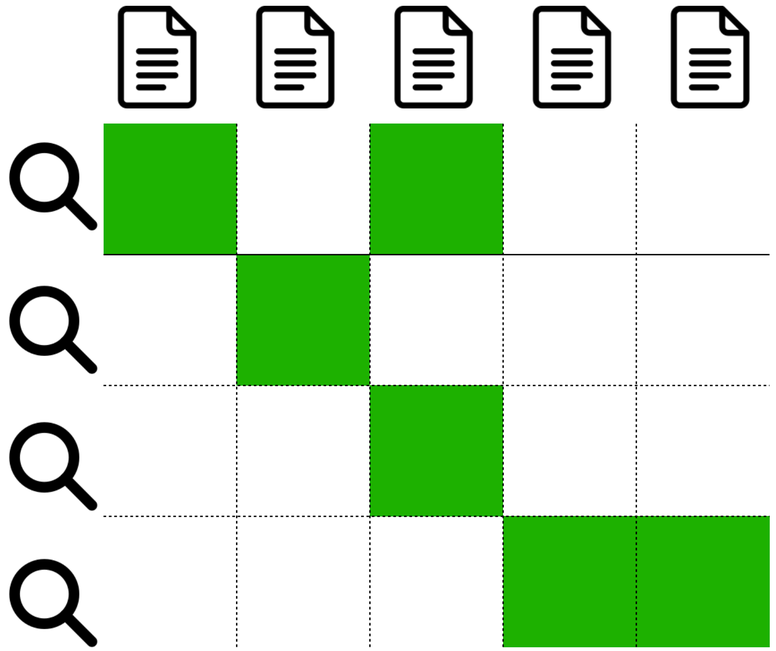
Categories →	Retr.	Rerank.	Clust.	PairClass.	Class.	STS	Summ.	Avg
# of datasets →	15	4	11	3	12	10	1	56
`LLaMA-2-7B`								
Ui + w. Mean	15.16	46.94	36.85	61.41	69.05	63.42	26.64	44.54
LLM2Vec (w/o SimCSE)	19.86	44.74	35.31	61.60	67.94	66.74	26.83	45.70
LLM2Vec	36.75	52.95	40.83	77.89	71.57	76.41	31.38	55.36
Echo	16.16	46.84	34.25	63.54	69.82	67.95	25.57	45.36
`Mistral-7B`								
Uni + w. Mean	10.43	45.11	35.82	60.28	71.14	58.59	26.57	42.46
Bi + Mean	15.84	47.40	35.55	66.53	72.18	71.04	29.93	46.86
LLM2Vec (w/o SimCSE)	19.74	50.43	40.06	70.95	72.51	71.90	27.84	49.43
LLM2Vec	38.05	**53.99**	40.63	**80.94**	**74.07**	**78.50**	30.19	**56.80**
Echo	22.68	51.07	36.78	75.87	72.69	73.60	29.54	50.26
`Meta-LLaMA-3-8B`								
Uni + w. Mean	15.17	46.22	36.84	60.94	67.41	62.80	25.51	43.98
Bi + Mean	3.90	34.56	14.27	42.71	57.89	51.15	23.26	30.56
LLM2Vec (w/o SimCSE)	24.75	49.20	39.74	65.91	69.00	67.85	25.59	48.84
LLM2Vec	**39.19**	53.09	**41.99**	78.01	71.88	75.86	**31.45**	56.23
Echo	12.58	49.79	36.32	68.95	70.22	67.43	26.44	45.32

LLM2Vec [BehnamGhader et al 2024]

Learning retrieval-oriented embeddings

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

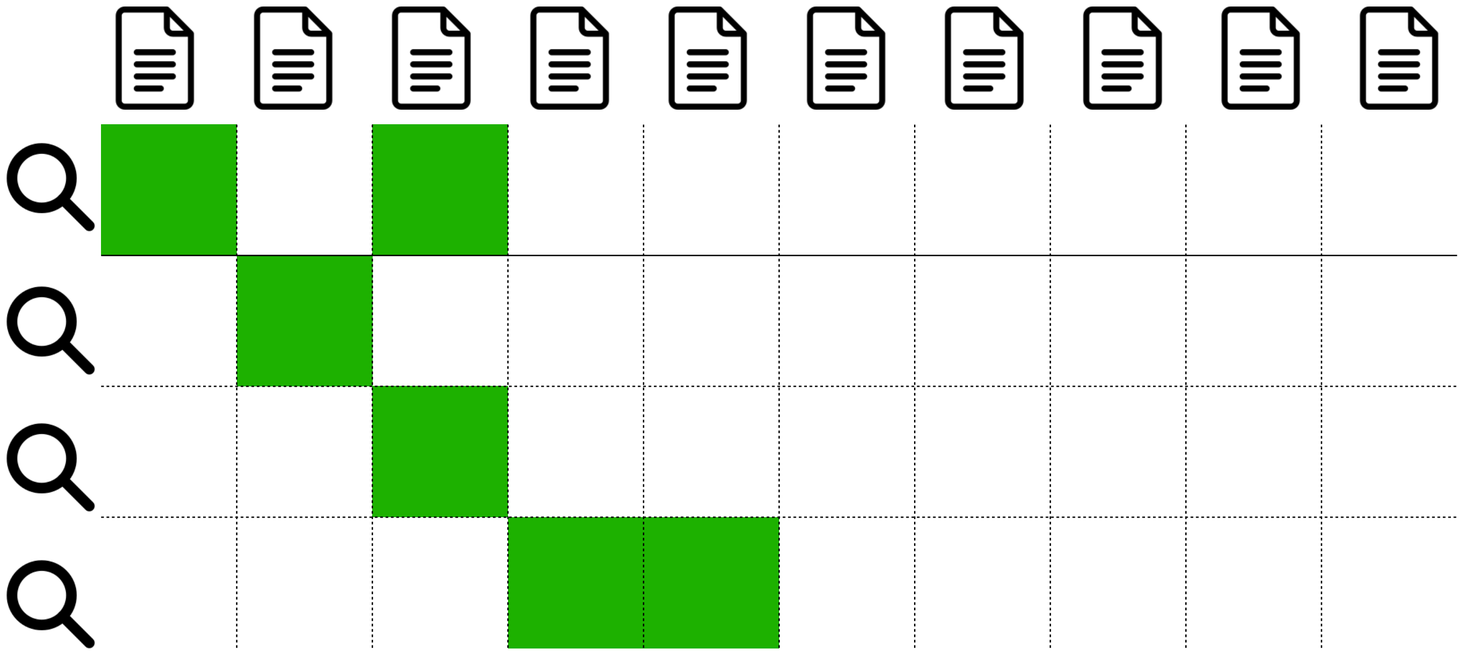
- Use **contrastive loss** to learn retrieval-oriented embeddings
- How to get positive / negative pairs
  - In-batch negative:
    - **Positive pairs**: query + associated documents
    - **Negative pairs**: query + other documents are negative pairs
    - May not get enough hard negatives



Learning retrieval-oriented embeddings

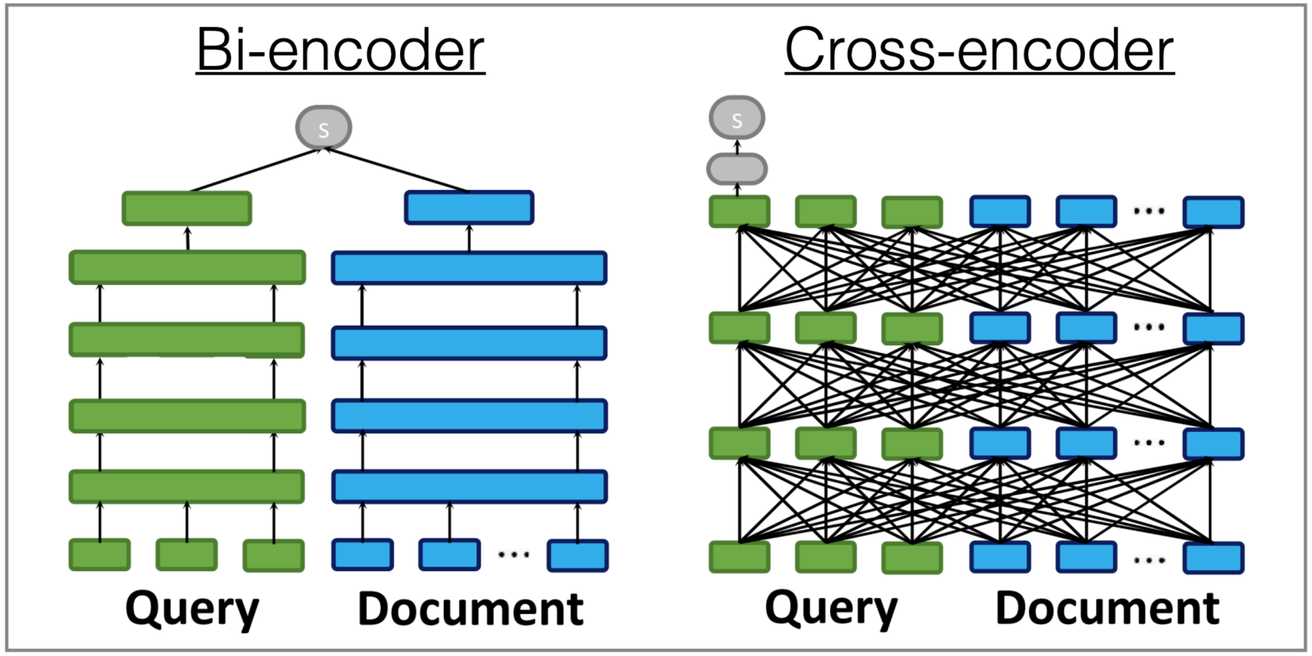
- How to get hard negative samples?
  - Use weaker retriever to retrieve documents and treat them as negatives (but may not be actual negatives!)
- Example systems:
  - DPR [Karpukhin et al. 2020]
    - Use BM25 hard negatives and in-batch negatives
  - Contriever [Izacard et al. 2022]
    - Contrastive learning with two random spans as positives (MoCo with random cropping)

- BM25 - get hard negative



Going beyond query-document embedding similarities

- Train a model to predict score of query/document matching (vs using cosine similarity of embeddings)
- Allows for more fine-grained interaction between tokens
- More expensive than computing similarity, does not allow for the use of ANN



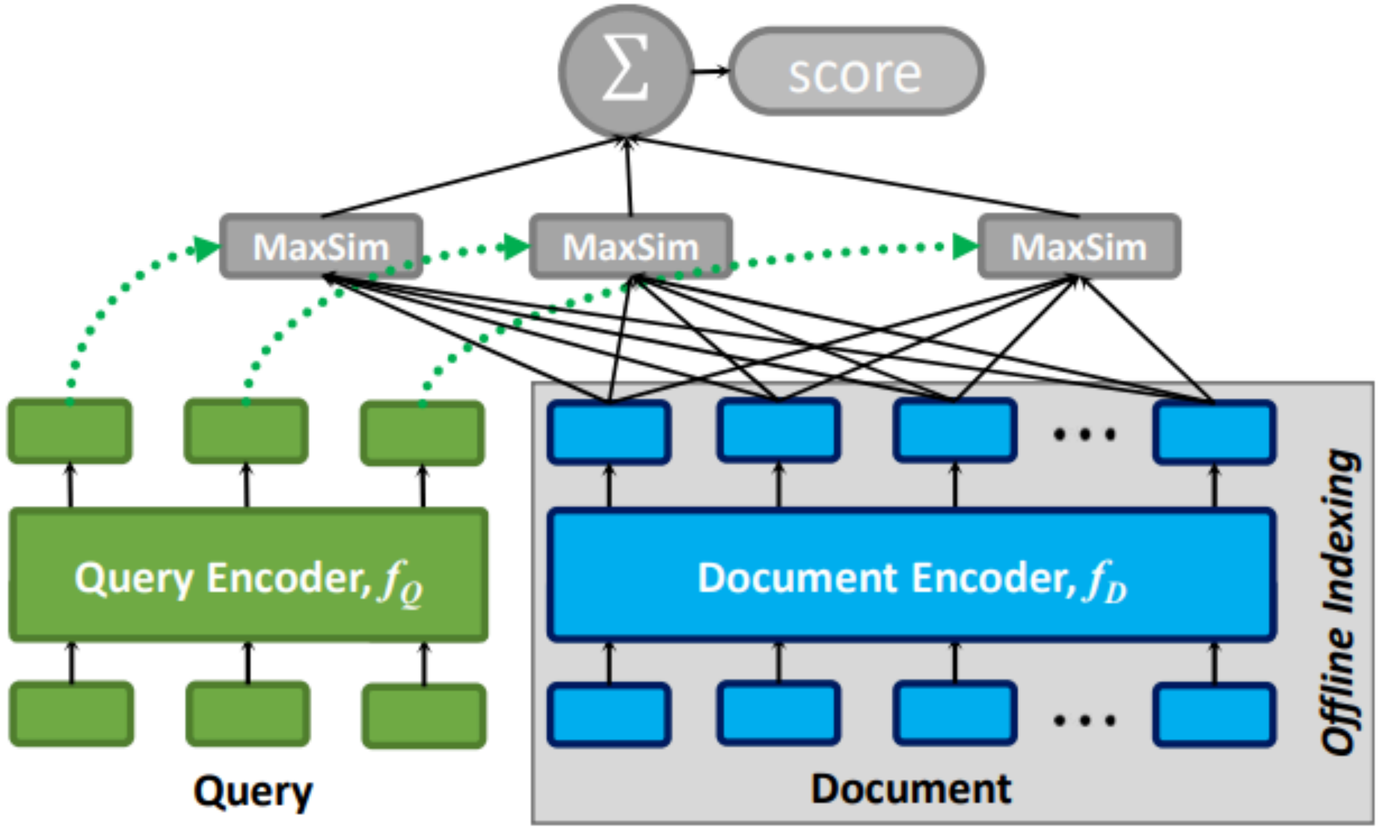
Cross-encoder re-ranking

Figure from ColBERT [Khattab and Zaharia 2020]

Token-level dense retrieval

ColBERT [Khattab and Zaharia 2020]

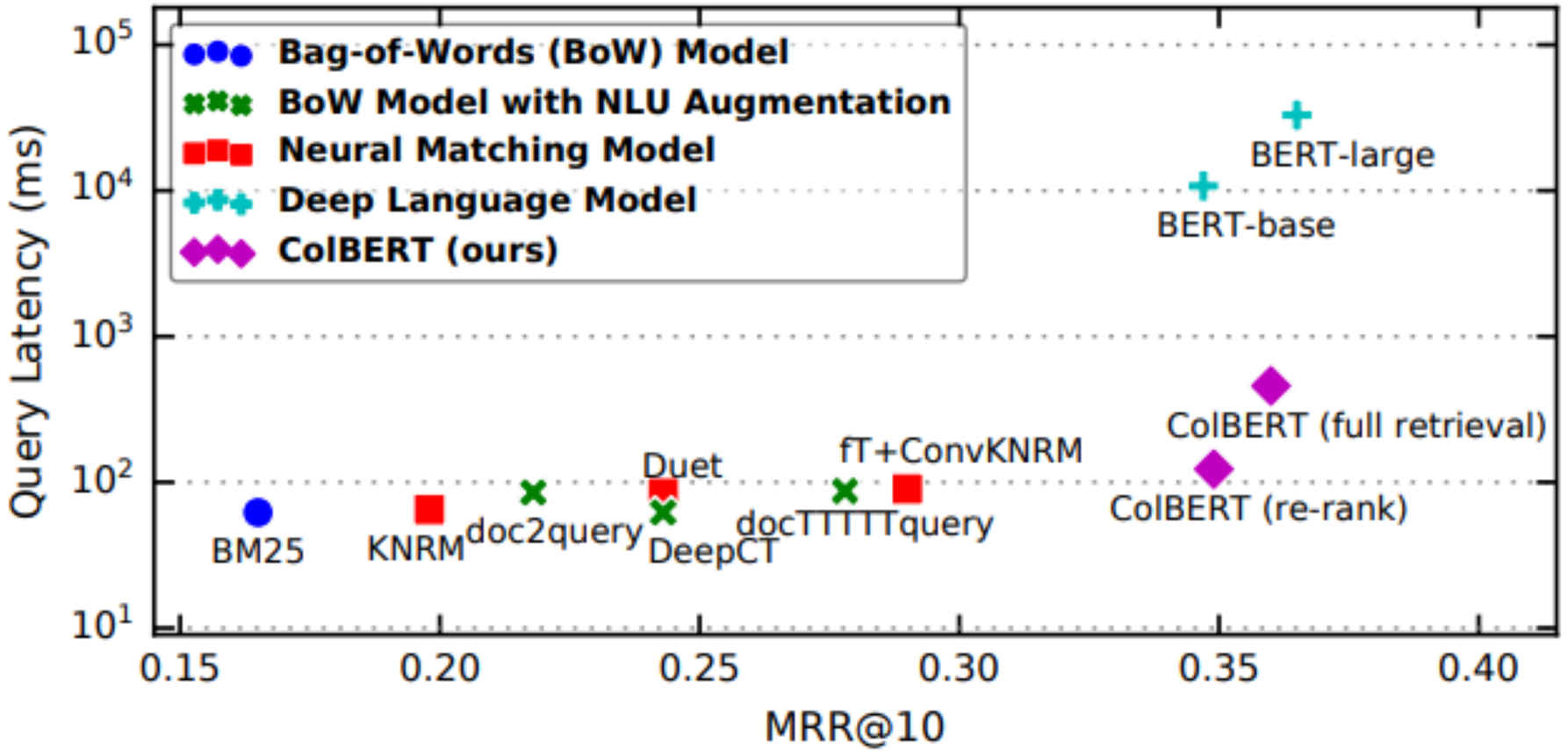
- Contextual representations of all query and document tokens to compute retrieval score
- More accurate but more expensive than single vector retrieval



Token-level dense retrieval

ColBERT [Khattab and Zaharia 2020]

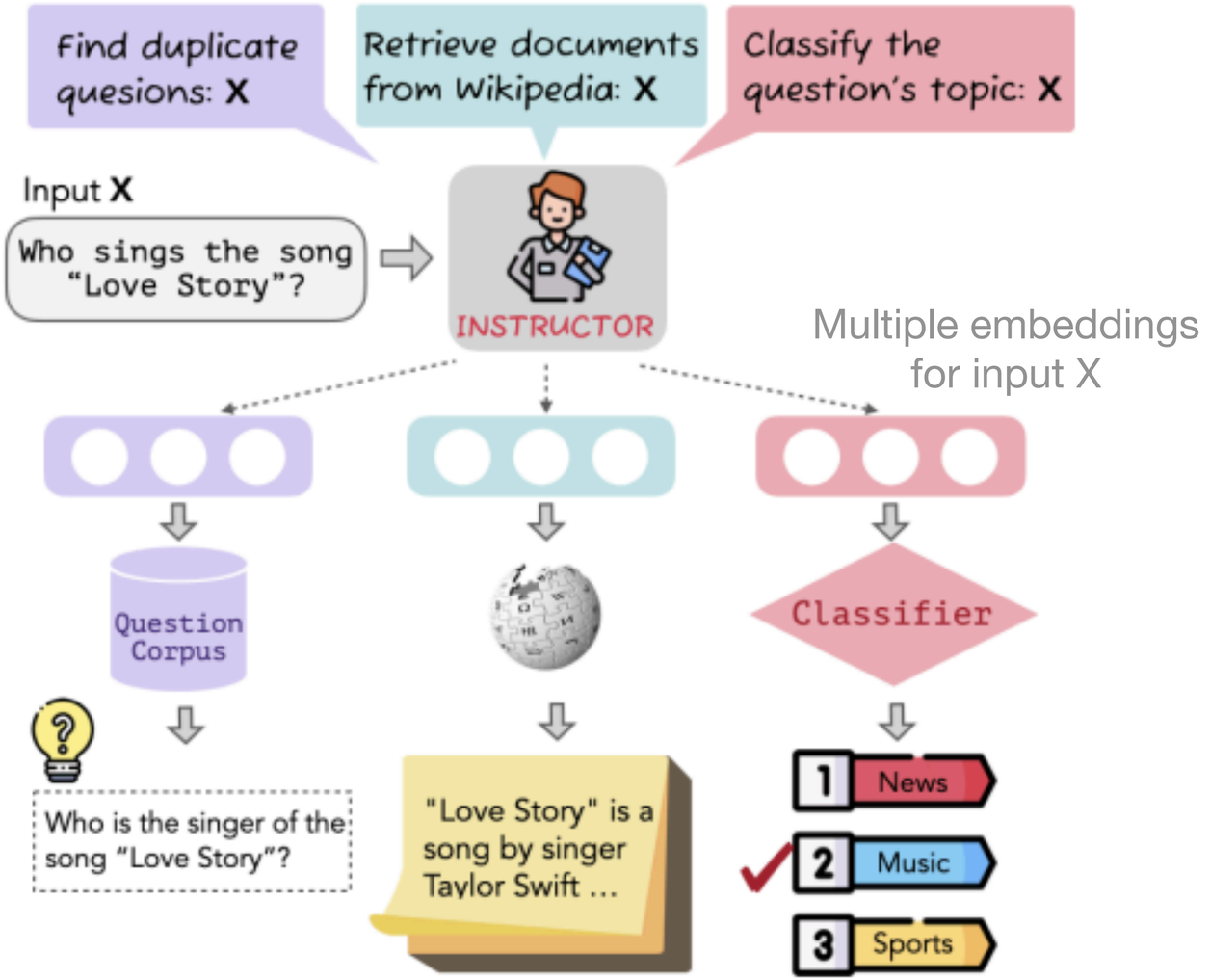
- Late interaction more efficient than all interaction model



Instruction fine-tuned embeddings

One embedder, Any task: Instruction-finetuned text embeddings
[Su et al. 2022]

- Instruction tuning to get task specific embeddings
  - Embedding is function of input X and task
- Use Sentence-T5 embeddings to construct positive / negative pairs based on similarity



Instruction fine-tuned embeddings

One embedder, Any task: Instruction-finetuned text embeddings
[Su et al. 2022]

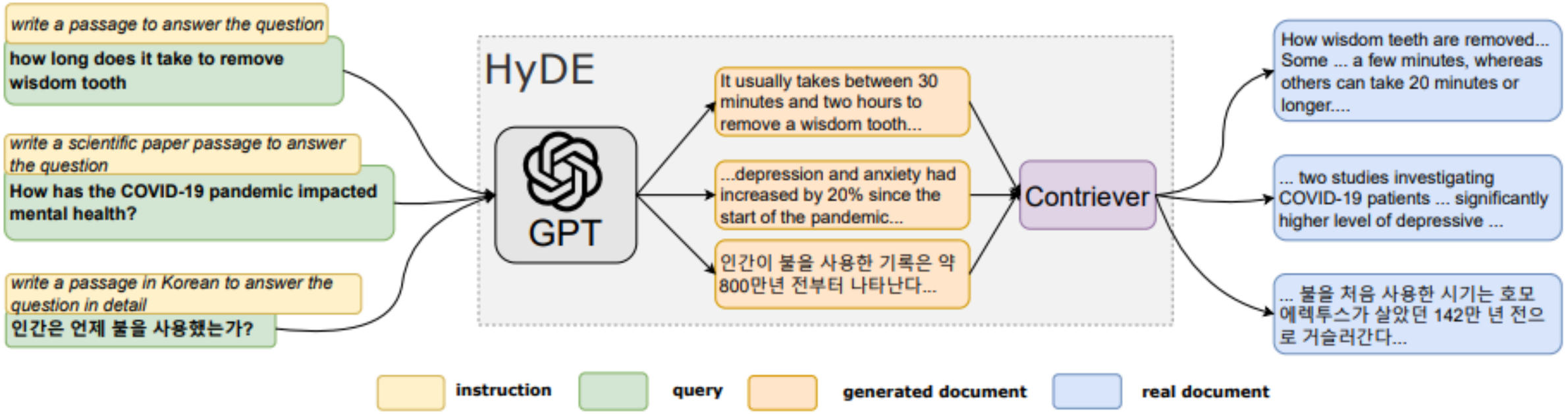
- Evaluation on massive text embedding benchmark (MTEB)

Benchmark	MTEB								Billboard	Prompt	Avg.
Task category	Retri.	Rerank	Cluster	Pair.	Class.	STS	Sum.	Avg.	Text Eval.	Retri.	
# datasets	15	4	11	3	12	10	1	56	3	11	70
Small Models for reference (<500M)											
SimCSE (110M)	21.9	47.5	33.4	73.7	67.3	79.1	23.3	48.7	29.4	58.3	48.2
coCondenser (110M)	33.0	51.8	37.6	81.7	64.7	76.5	29.5	52.4	31.5	59.6	51.8
Contriever (110M)	41.9	53.1	41.1	82.5	66.7	76.5	30.4	56.0	29.0	57.3	53.2
GTR-Large (335M)	47.4	55.4	41.6	85.3	67.1	78.2	29.5	58.3	31.2	59.8	55.1
INSTRUCTOR (335M)	**47.6**	**57.5**	**45.3**	**85.9**	**73.9**	**83.2**	**31.8**	**61.6**	**36.9**	**63.2**	**58.4**
Relative gain (%)	+0.4	+4.5	+8.9	+0.7	+10.1	+6.4	+7.8	+5.7	+18.3	+5.7	+5.9
Large Models for reference(≥500M)											
Sent-T5-XXL (4.8B)	42.2	56.4	43.7	85.1	**73.4**	82.6	30.1	59.5	33.9	61.5	56.5
GTR-XXL (4.8B)	48.1	56.7	42.4	86.1	67.4	78.4	30.6	58.9	32.0	60.8	55.8
SGPT-NLI (5.8B)	32.3	52.3	37.0	77.0	70.1	80.5	30.4	53.7	29.6	57.9	51.9
GTR-XL (1.5B)	48.0	56.0	41.5	86.1	67.1	77.8	30.2	58.4	32.0	60.4	55.5
INSTRUCTOR-XL (1.5B)	**49.3**	**57.3**	**44.7**	**86.6**	73.2	**83.1**	**32.0**	**61.8**	**34.1**	**68.6**	**58.8**
Relative gain (%)	+2.7	+2.3	+7.7	+0.6	+9.1	+6.9	+6.0	+5.8	+6.6	+13.6	+5.9

Hypothetical document embeddings

Precise Zero-Shot Dense Retrieval without Relevance Labels
[Gao et al. 2022]

- No relevance supervision
- 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



Hypothetical document embeddings

Precise Zero-Shot Dense Retrieval without Relevance Labels
[Gao et al. 2022]

- Performance comparable to supervising with relevance judgements

	DL19			DL20		
	map	ndcg@10	recall@1k	map	ndcg@10	recall@1k
w/o relevance judgement						
BM25	30.1	50.6	75.0	28.6	48.0	78.6
Contriever	24.0	44.5	74.6	24.0	42.1	75.4
HyDE	**41.8**	**61.3**	**88.0**	**38.2**	**57.9**	**84.4**
w/ relevance judgement						
DPR	36.5	62.2	76.9	41.8	**65.3**	81.4
ANCE	37.1	**64.5**	75.5	40.8	64.6	77.6
Contriever^{FT}	41.7	62.1	83.6	**43.6**	63.2	**85.8**

Retrieval evaluation

Retrieval and ranking metrics

- Start with gold (ground-truth) relevance judgements
- Mean reciprocal rank (MRR) of first relevant document
  - Okay if there is just one relevant document or only care about the highest ranked one
- Normalized discounted cumulative gain
- Mean average precision
- Recall@N

2: Highly relevant

1: Somewhat relevant

0: Not relevant

Mean reciprocal rank (MRR)

- Average reciprocal rank over different queries

System ranking

RR = 1/2

$$\frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{\text{rank}_q}$$

Normalized discounted cumulative gain

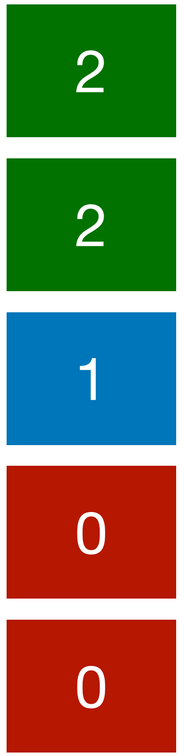
- Compares system ranking to ideal ranking

System ranking

$$\text{nDCG} = \frac{\text{DCG}}{\text{iDCG}}$$

$$\text{DCG}@N = \sum_{i=1}^{N} \frac{\text{relevance}_i}{\log_2(i+1)}$$

Ideal ranking



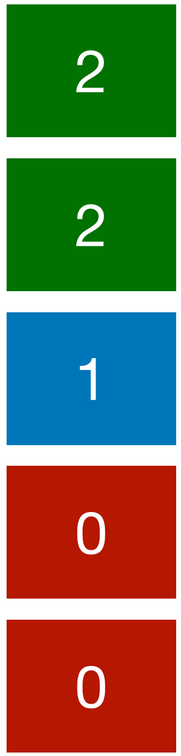
Cumulative gain

- CG@N: Sum of relevance scores for N retrieved document

System ranking CG@5 = 5

sum(0,2,1,2,0) = 5

Ideal ranking CG@5 = 5



sum(2,2,1,0,0) = 5

Discounted cumulative gain

- DCG@N: Add discount of $\frac{1}{\log_2(i+1)}$ for rank i (lower scores for lower ranked items)

System ranking

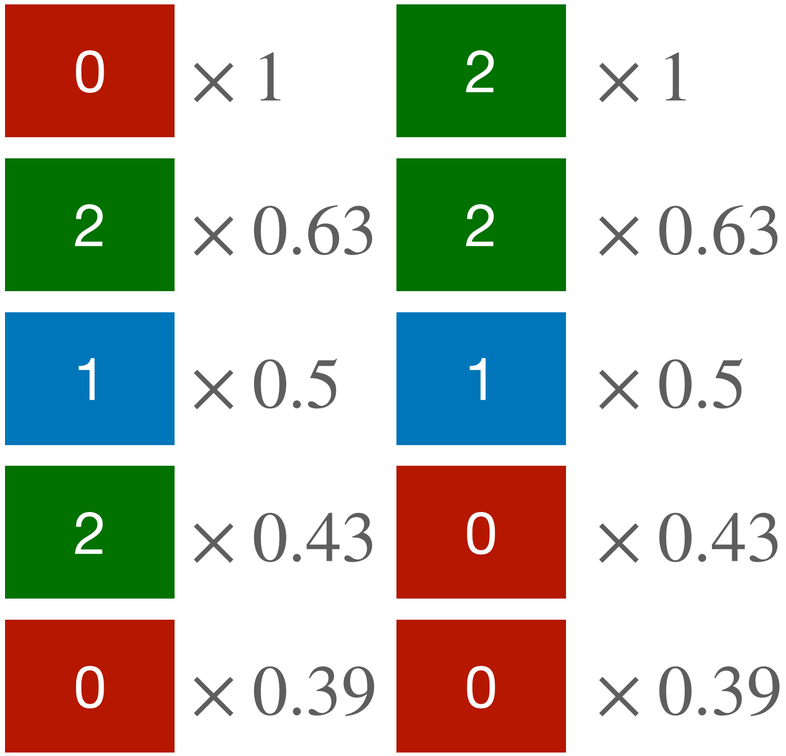
0	$\times 1$	
2	$\times 0.63$	
1	$\times 0.5$	DCG@5 = 2.62
2	$\times 0.43$	
0	$\times 0.39$	

Ideal ranking

2	$\times 1$	
2	$\times 0.63$	
1	$\times 0.5$	iDCG@5 = 3.76
0	$\times 0.43$	
0	$\times 0.39$	

Normalized discounted cumulative gain

- nDCG = DCG/iDCG
- Better score as you pick up more good docs



DCG@2 = 1.26 iDCG@2 = 3.26 **nDCG@2 = 0.386**

DCG@3 = 1.76 iDCG@3 = 3.76 **nDCG@3 = 0.468**

DCG@5 = 2.62 iDCG@5 = 3.76 **nDCG@5 = 0.697**

Other metrics

Mean average precision: average precision at which each relevant document is retrieved

Non-relevant documents are skipped in the average

MAP@5 = 0.638

Recall@N: percentage of time you get a positive document in the top N

R@1 = 0

R@2 = 1

R@3 = 1

R@4 = 1

R@5 = 1

For all these metrics: take average over dataset of queries to get final metric

Retrieval-augmented generation (Retriever + Reader models)

Document Retrieval QA

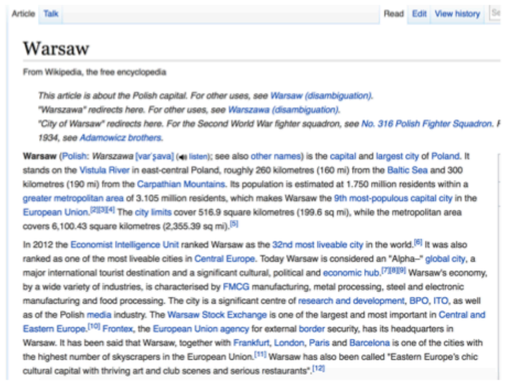
https://aclanthology.org/P17-1171.pdf

Open-domain QA

SQuAD, TREC, WebQuestions, WikiMovies

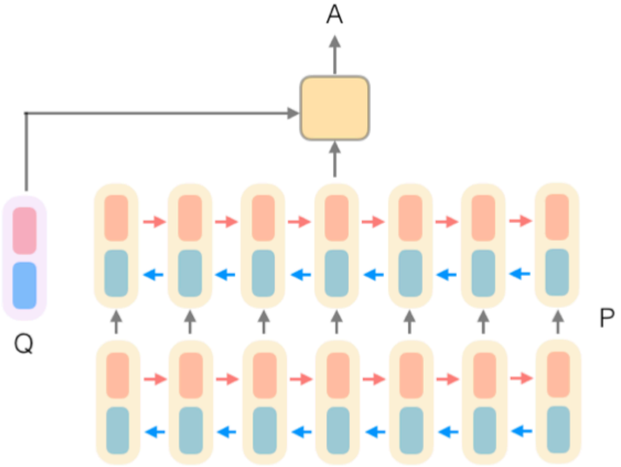
Q: How many of Warsaw's inhabitants spoke Polish in 1933?

Document Retriever



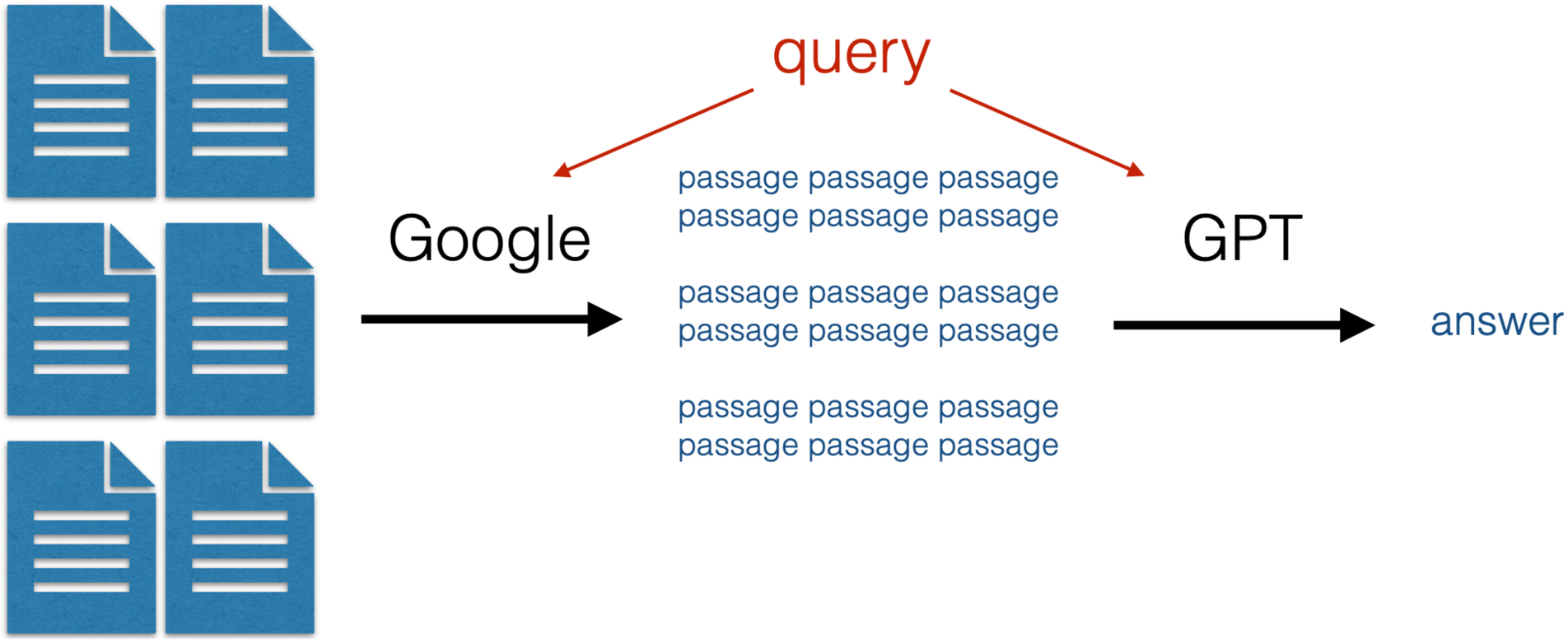
Document Reader

833,500



Retrieval-reader models

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

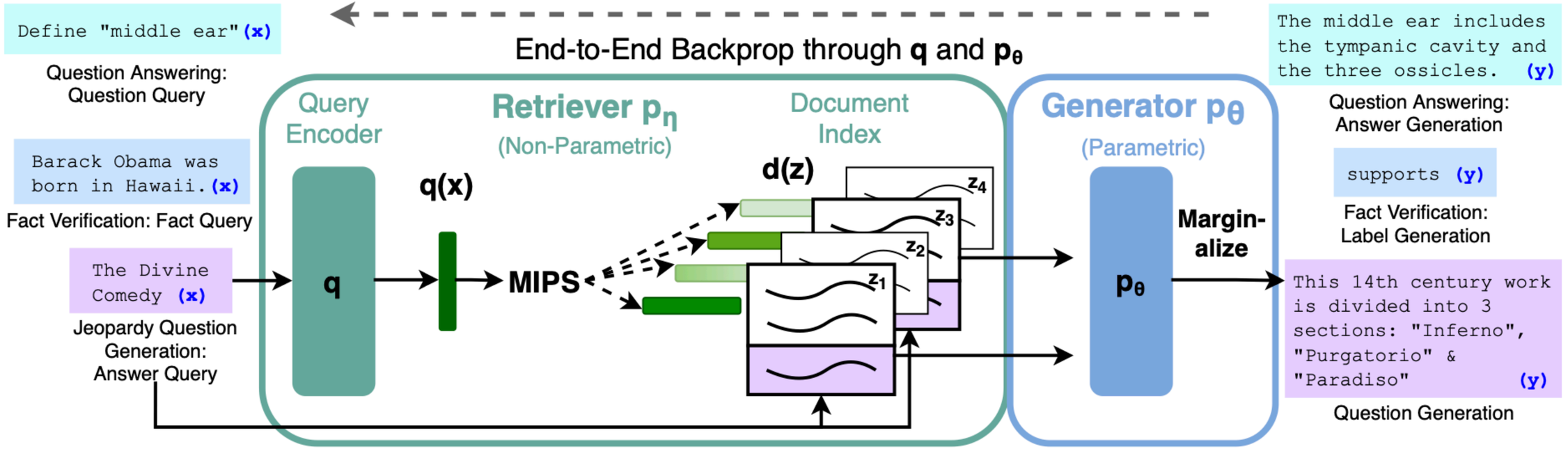


- Passages are concatenated to the context

RAG

paper

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



$$p_\eta(z|x) \propto \exp\left(\mathbf{d}(z)^\top \mathbf{q}(x)\right) \qquad \mathbf{d}(z) = \mathrm{BERT}_d(z), \quad \mathbf{q}(x) = \mathrm{BERT}_q(x)$$

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Lewis et al. NeurIPS 2020]

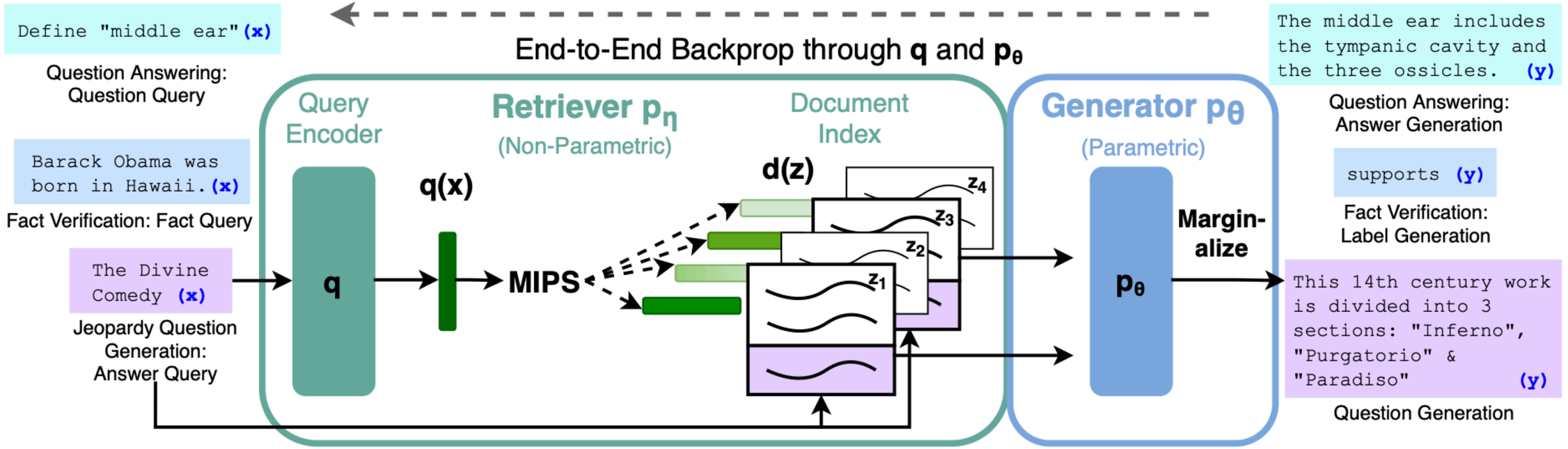
RAG

paper

- Generate using a mixture model
  - Pick a document
  - Generate from document
- Probability of the **retriever** is based on embeddings
- Adjusts retriever to give higher similarities to helpful docs
- Issue: search index becomes stale => can only train q(x)

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

Retriever Generator



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

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Lewis et al. NeurIPS 2020]

RAG paper

RAG-Sequence Model The RAG-Sequence model uses the same retrieved document to generate the complete *sequence*. Technically, it treats the retrieved document as a single latent variable that is marginalized to get the seq2seq probability $p(y|x)$ via a top-K approximation. Concretely, the top K documents are retrieved using the retriever, and the generator produces the output sequence probability for each document, which are then marginalized,

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y|x,z) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) \prod_i^N p_\theta(y_i|x,z,y_{1:i-1})$$

RAG-Token Model In the RAG-Token model we can draw a different latent document for each target *token* and marginalize accordingly. This allows the generator to choose content from several documents when producing an answer. Concretely, the top K documents are retrieved using the retriever, and then the generator produces a distribution for the next output token for each document, before marginalizing, and repeating the process with the following output token, Formally, we define:

$$p_{\text{RAG-Token}}(y|x) \approx \prod_i^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z|x) p_\theta(y_i|x,z_i,y_{1:i-1})$$

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Lewis et al. NeurIPS 2020]

RAG paper

At test time, RAG-Sequence and RAG-Token require different ways to approximate $\arg\max_y p(y|x)$.

RAG-Token The RAG-Token model can be seen as a standard, autoregressive seq2seq generator with transition probability: $p'_\theta(y_i|x, y_{1:i-1}) = \sum_{z \in \text{top-}k(p(\cdot|x))} p_\eta(z_i|x) p_\theta(y_i|x, z_i, y_{1:i-1})$ To decode, we can plug $p'_\theta(y_i|x, y_{1:i-1})$ into a standard beam decoder.

RAG-Sequence For RAG-Sequence, the likelihood $p(y|x)$ does not break into a conventional per-token likelihood, hence we cannot solve it with a single beam search. Instead, we run beam search for each document z, scoring each hypothesis using $p_\theta(y_i|x, z, y_{1:i-1})$. This yields a set of hypotheses Y, some of which may not have appeared in the beams of all documents. To estimate the probability of an hypothesis y we run an additional forward pass for each document z for which y does not appear in the beam, multiply generator probability with $p_\eta(z|x)$ and then sum the probabilities across beams for the marginals. We refer to this decoding procedure as "Thorough Decoding." For longer output sequences, $|Y|$ can become large, requiring many forward passes. For more efficient decoding, we can make a further approximation that $p_\theta(y|x, z_i) \approx 0$ where y was not generated during beam search from x, z_i. This avoids the need to run additional forward passes once the candidate set Y has been generated. We refer to this decoding procedure as "Fast Decoding."

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Lewis et al. NeurIPS 2020]

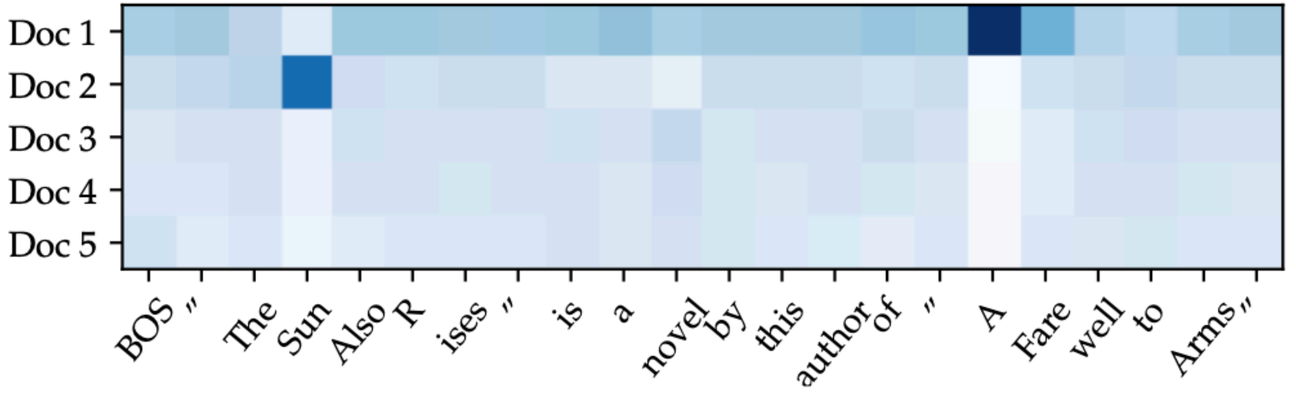
RAG

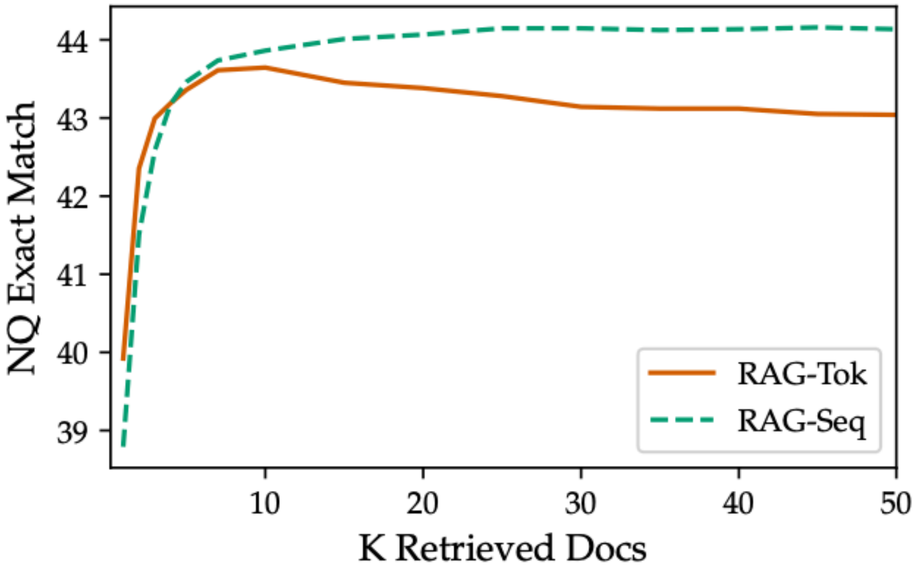
paper

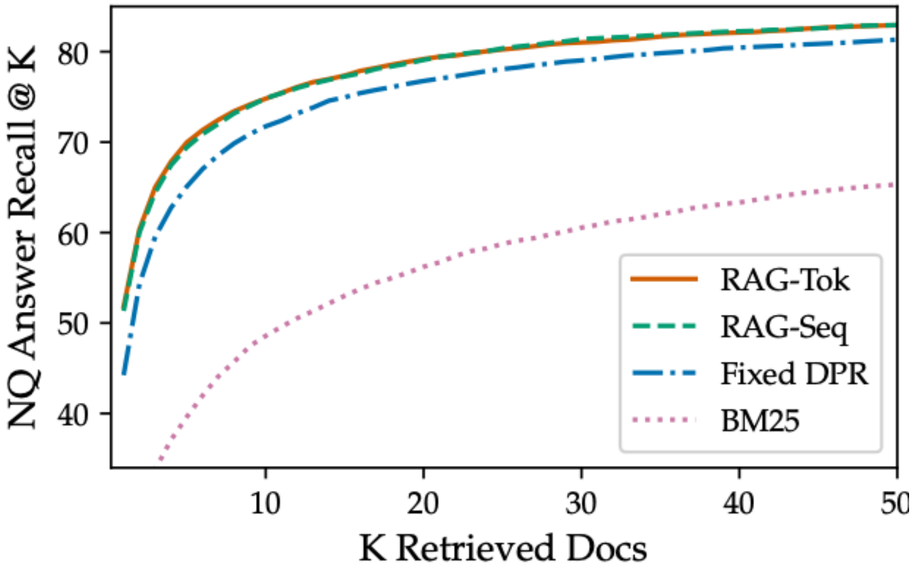
Figure 2: RAG-Token document posterior $p(z_i|x, y_i, y_{-i})$ for each generated token for input "Hemingway" for Jeopardy generation with 5 retrieved documents. The posterior for document 1 is high when generating "A Farewell to Arms" and for document 2 when generating "The Sun Also Rises".

Document 1: his works are considered classics of American literature ... His wartime experiences formed the basis for his novel **"A Farewell to Arms"** (1929) ...

Document 2: ... artists of the 1920s "Lost Generation" expatriate community. His debut novel, **"The Sun Also Rises"**, was published in 1926.







Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Lewis et al. NeurIPS 2020]

Training LMs with retrieved context

- Retrieve context and condition the pre-training of the model on these context
- Examples
  - DrQA [Chen et al. 2017] - retrieve from Wikipedia and train reader Search when the model is uncertain [Jiang et al. 2023]
  - RETRO [Borgeaud et al. 2021] - pre-train language model with retrieved context

When to retrieve?

- Once at beginning of generation
  - Default used by most early systems [Lewis et al. 2020]
- Several times during generation (as needed)
  - Generate a search token [Schick et al. 2023]
  - Search when the model is uncertain [Jiang et al. 2023]
- Every token
  - Fine similar final embeddings [Khadelwal et al. 2019]
  - Approximate attention with nearest neighbours [Bertsch et al. 2023]

When to retrieve?

- Trigger retrieval based on tokens
- Toolformer [Schick et al. 2023] generates tokens that trigger retrieval (or other tools)
- Training 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.

Call QA system

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

Call Calculator

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

Call machine translation system

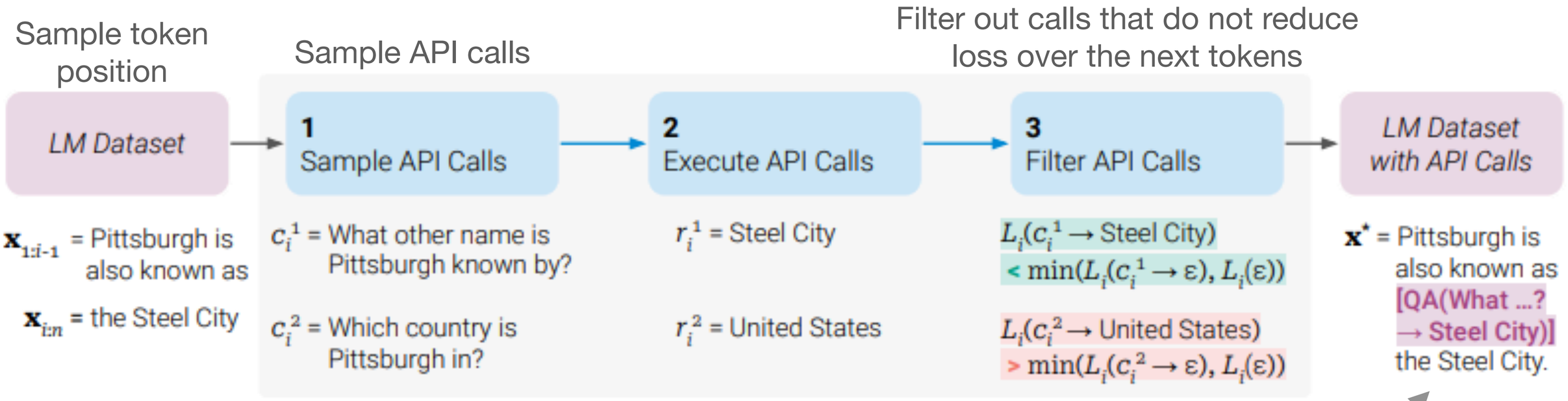
The Brown Act is California's law **[WikiSearch("Brown Act") → The Ralph M. Brown Act is an act of the California State Legislature that guarantees the public's right to attend and participate in meetings of local legislative bodies.]** that requires legislative bodies, like city councils, to hold their meetings open to the public.

Call search

Toolformer: Language Models Can Teach Themselves to Use Tools [Schick et al. NeurIPS 2023]

When to retrieve (or call a tool)?

Convert original dataset into dataset with API calls



Keep calls that make it easier to predict what comes after

$$L_i^- - L_i^+ \geq \tau_f$$

$$L_i^- = \min\left(L_i(\varepsilon), L_i(\mathbf{e}(c_i, \varepsilon))\right)$$

$$L_i^+ = L_i(\mathbf{e}(c_i, r_i))$$

Remaining API calls (with responses) are linearized and interspersed into the original text

Example for question answering tool

Toolformer: Language Models Can Teach Themselves to Use Tools [Schick et al. NeurIPS 2023]

When to retrieve (or call a tool)?

Table 3: Results on subsets of LAMA and various benchmarks requiring mathematical reasoning. For LAMA, Toolformer uses the question answering tool for most examples, clearly outperforming all baselines of the same size and achieving results competitive with GPT-3. For the math benchmarks, Toolformer makes extensive use of the calculator tool, clearly outperforming OPT and GPT-3. Best results with a GPT-J based model are shown in bold, best results overall are underlined.

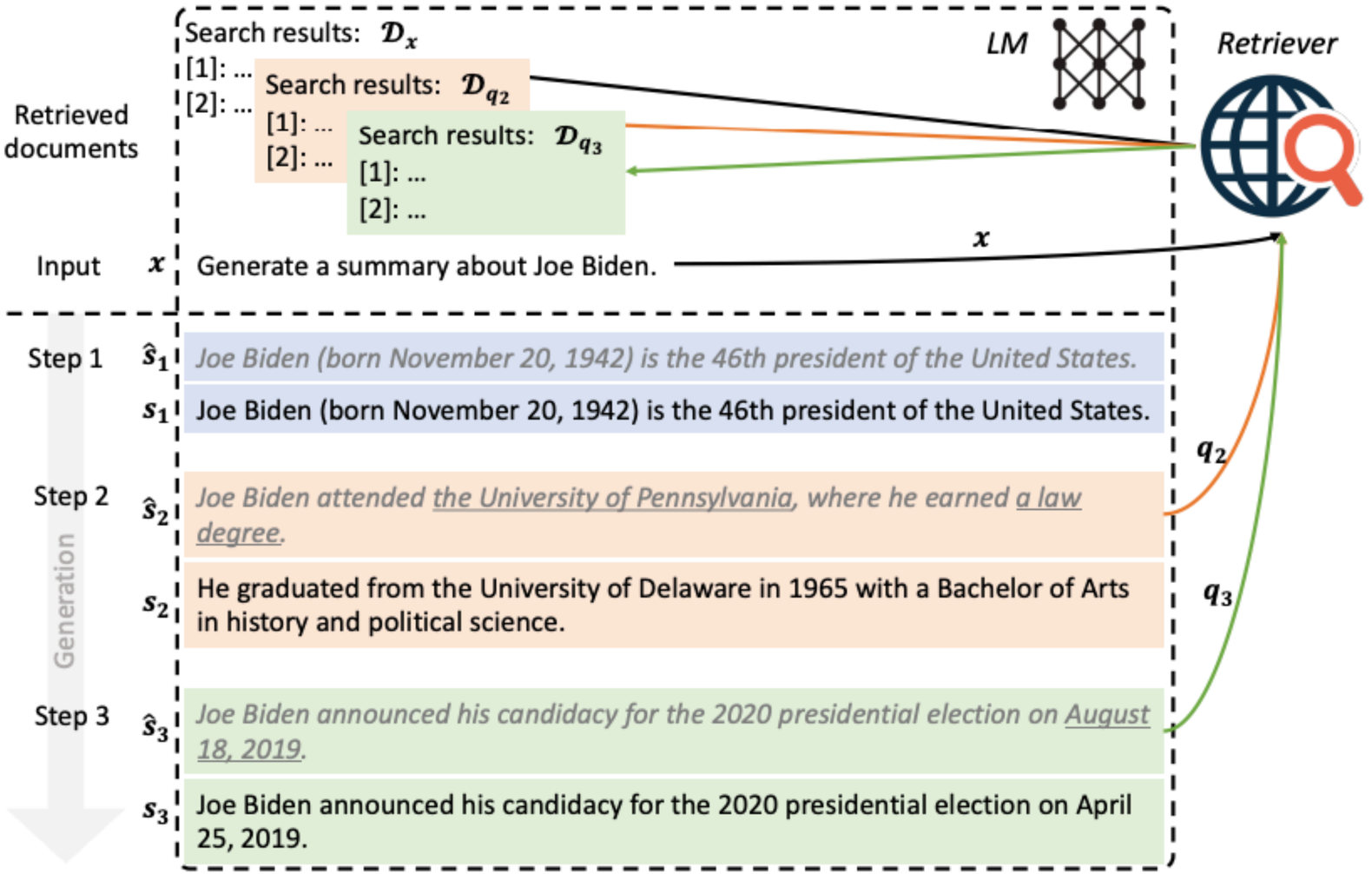
	LAMA			Math Benchmarks		
Model	**SQuAD**	**Google-RE**	**T-REx**	**ASDiv**	**SVAMP**	**MAWPS**
GPT-J	17.8	4.9	31.9	7.5	5.2	9.9
GPT-J + CC	19.2	5.6	33.2	9.6	5.0	9.3
Toolformer (disabled)	22.1	6.3	34.9	14.8	6.3	15.0
Toolformer	**33.8**	**11.5**	**53.5**	**40.4**	**29.4**	**44.0**
OPT (66B)	21.6	2.9	30.1	6.0	4.9	7.9
GPT-3 (175B)	26.8	7.0	39.8	14.0	10.0	19.8

Toolformer: Language Models Can Teach Themselves to Use Tools [Schick et al. NeurIPS 2023]

Retrieve when uncertain

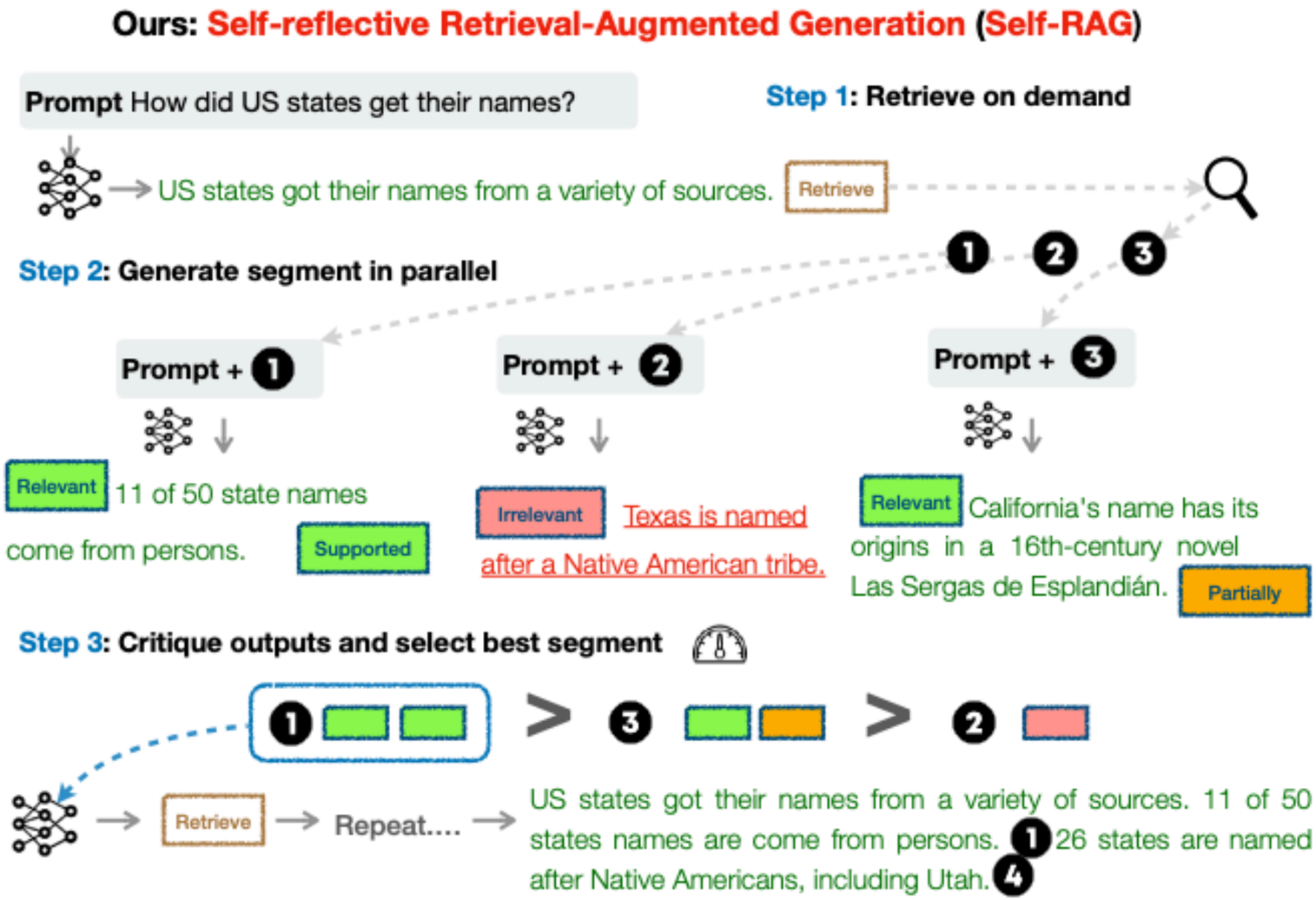
FLARE [Jiang et al. 2023]

- **F**orward-**L**ooking **A**ctive **Re**trieval augmented generation (FLARE)
- Tries to generate content
- Retrieve if LM certainty (of generated sentences) is low



Retrieval and critique

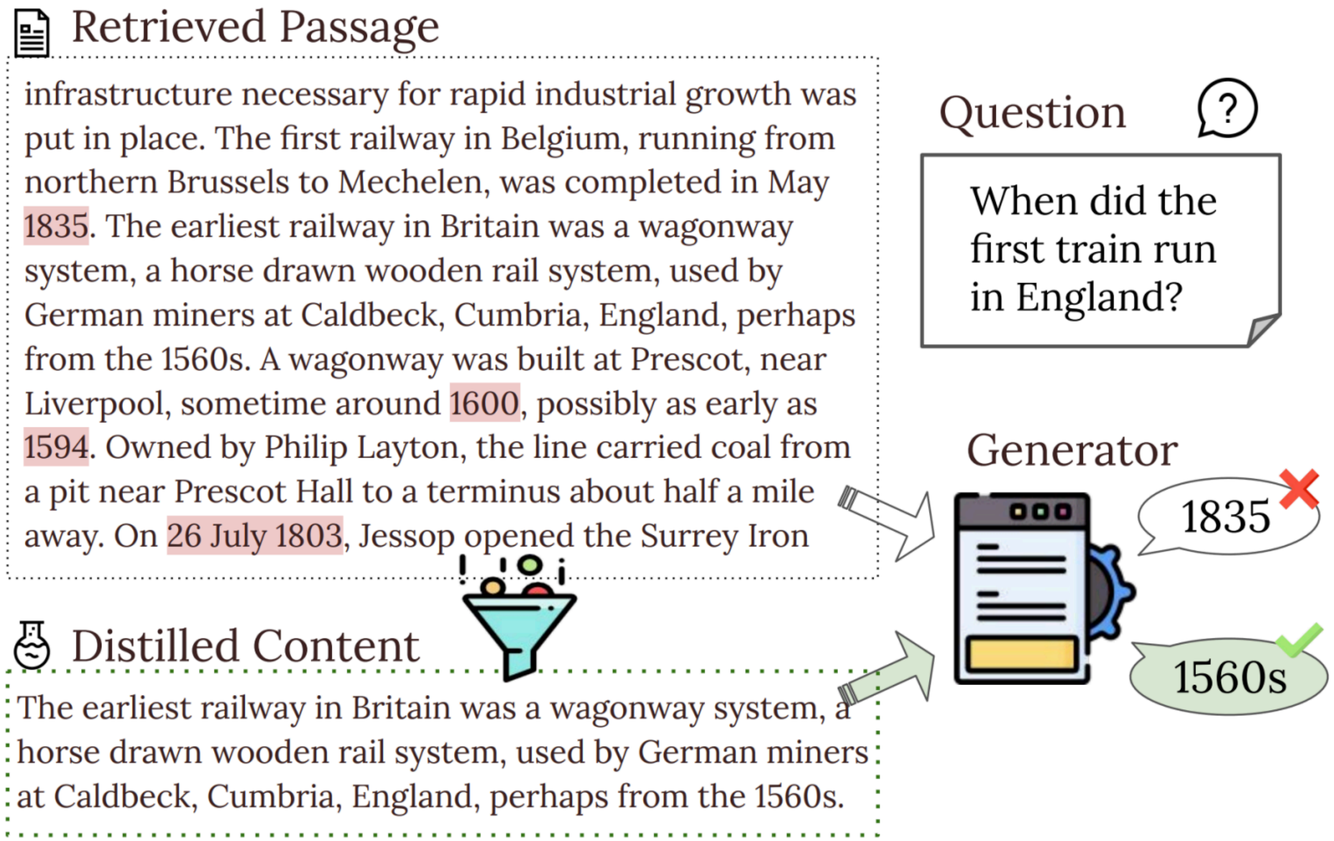
- SELF-RAG [Asai et al. 2023] critiques the retrieved / generated outputs for accuracy
- Generate segments in parallel and select best segments to use



Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection [Asai et al. ICLR 2024]

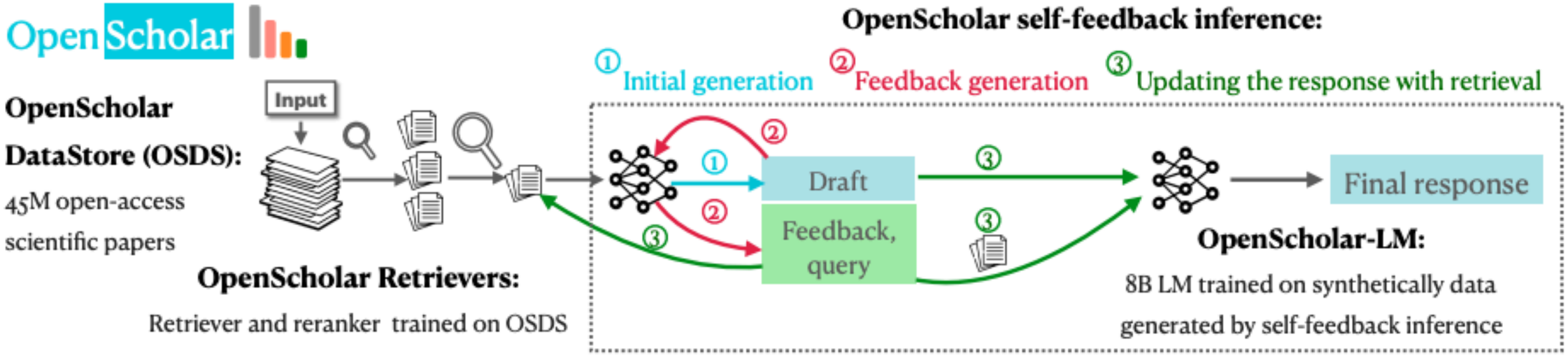
Ensuring use of relevant context

- Better retrievers provides more relevant context
- Decide whether to include passages or not [Asai et al. 2021]



OpenScholar

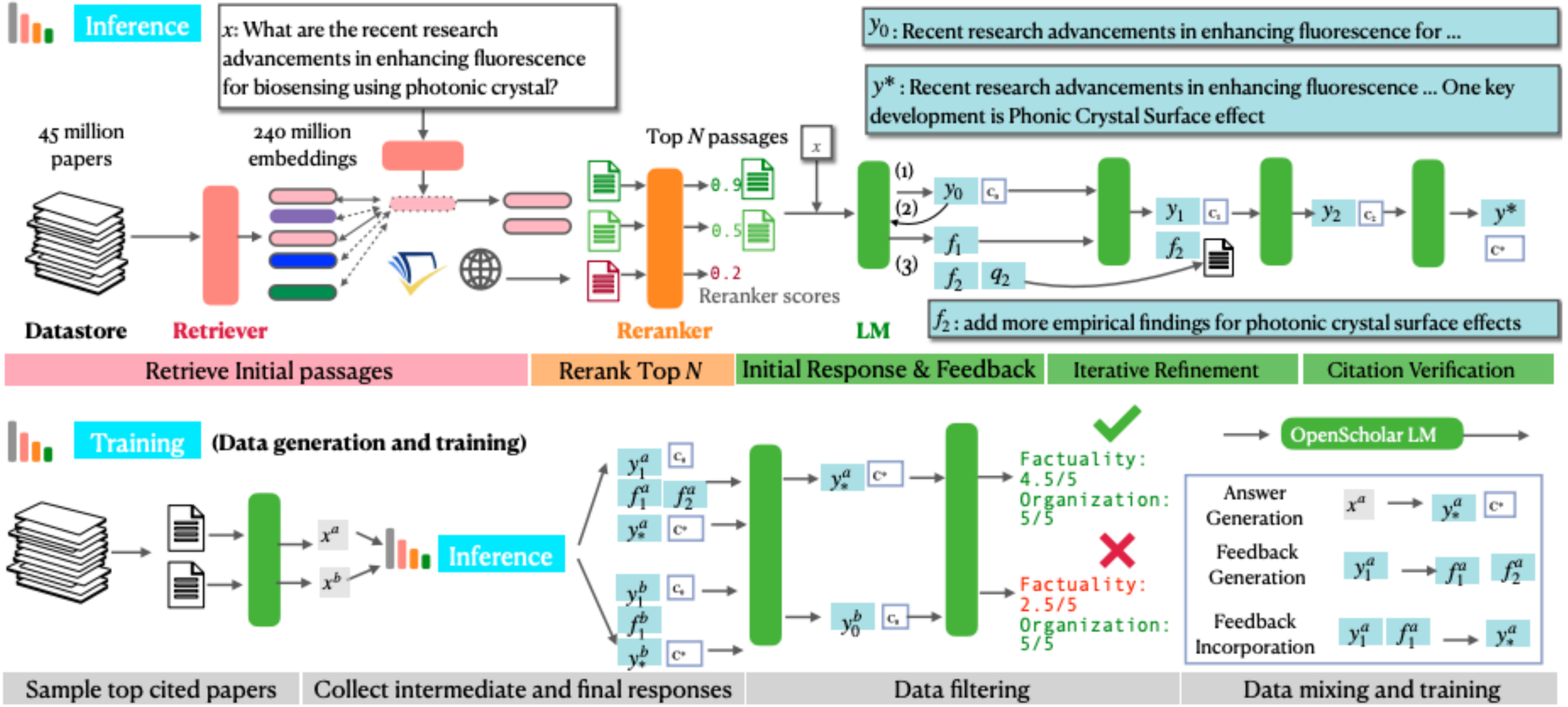
- Improved language model trained on synthetically generated data



OpenScholar: Synthesizing Scientific Literature with Retrieval-augmented LMs [Asai et al. 2024]

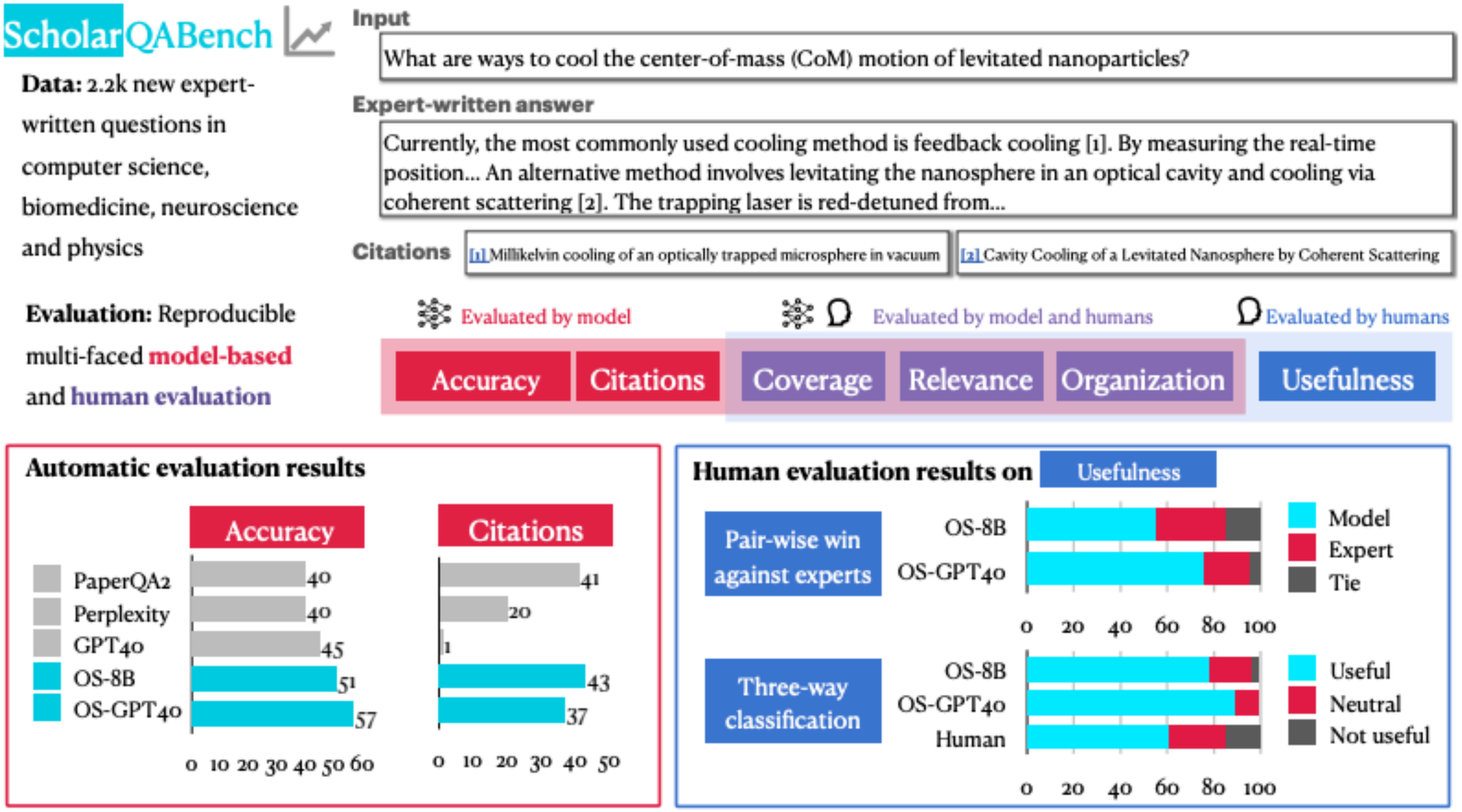
OpenScholar

- Improved language model trained on synthetically generated data



OpenScholar: Synthesizing Scientific Literature with Retrieval-augmented LMs [Asai et al. 2024]

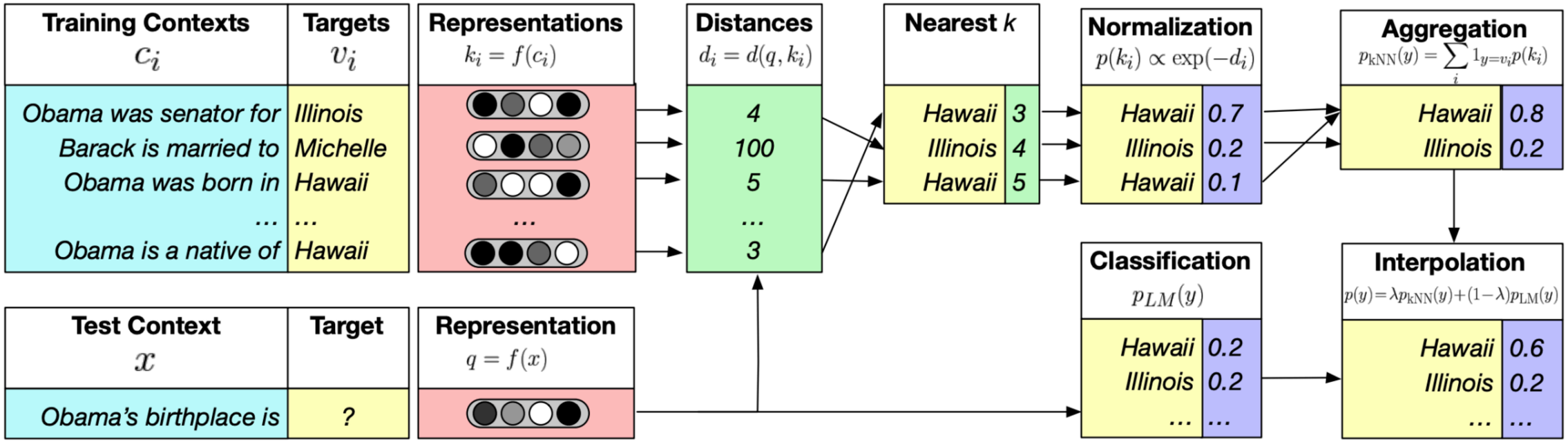
OpenScholar



OpenScholar: Synthesizing Scientific Literature with Retrieval-augmented LMs [Asai et al. 2024]

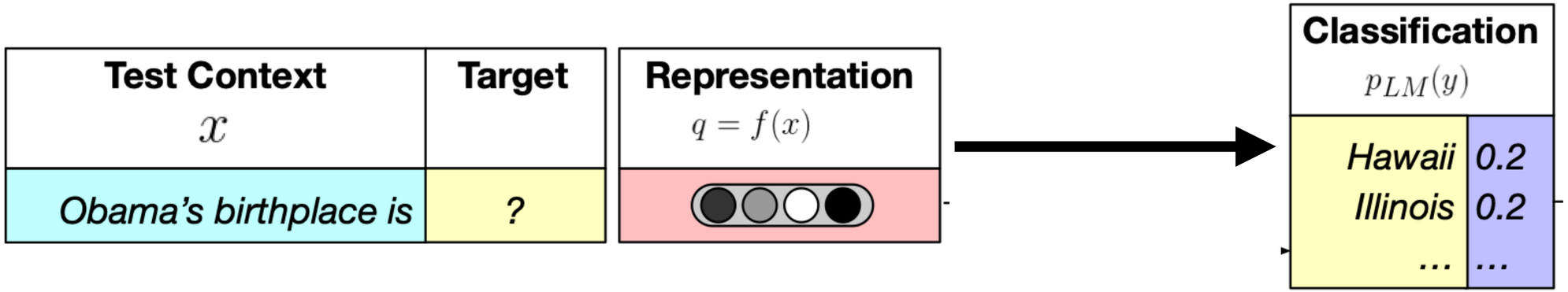
kNN-LM

- Language model similar examples are retrieved and tokens (following token) are used from retrieved examples

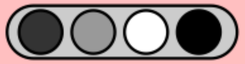


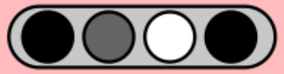
Generalization through Memorization: Nearest Neighbor Language Models [Khandelwal et al. ICLR 2020]

Standard LM prediction

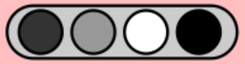


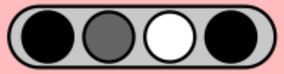
kNN LM prediction (step 1)

Test Context x	Target	Representation $q = f(x)$
Obama's birthplace is	*?*	

Training Contexts c_i	Targets v_i	Representations $k_i = f(c_i)$	Distances $d_i = d(q, k_i)$
Obama was senator for	*Illinois*		*4*
Barack is married to	*Michelle*		*100*
Obama was born in	*Hawaii*		*5*
...	*...*	*...*	*...*
Obama is a native of	*Hawaii*		*3*

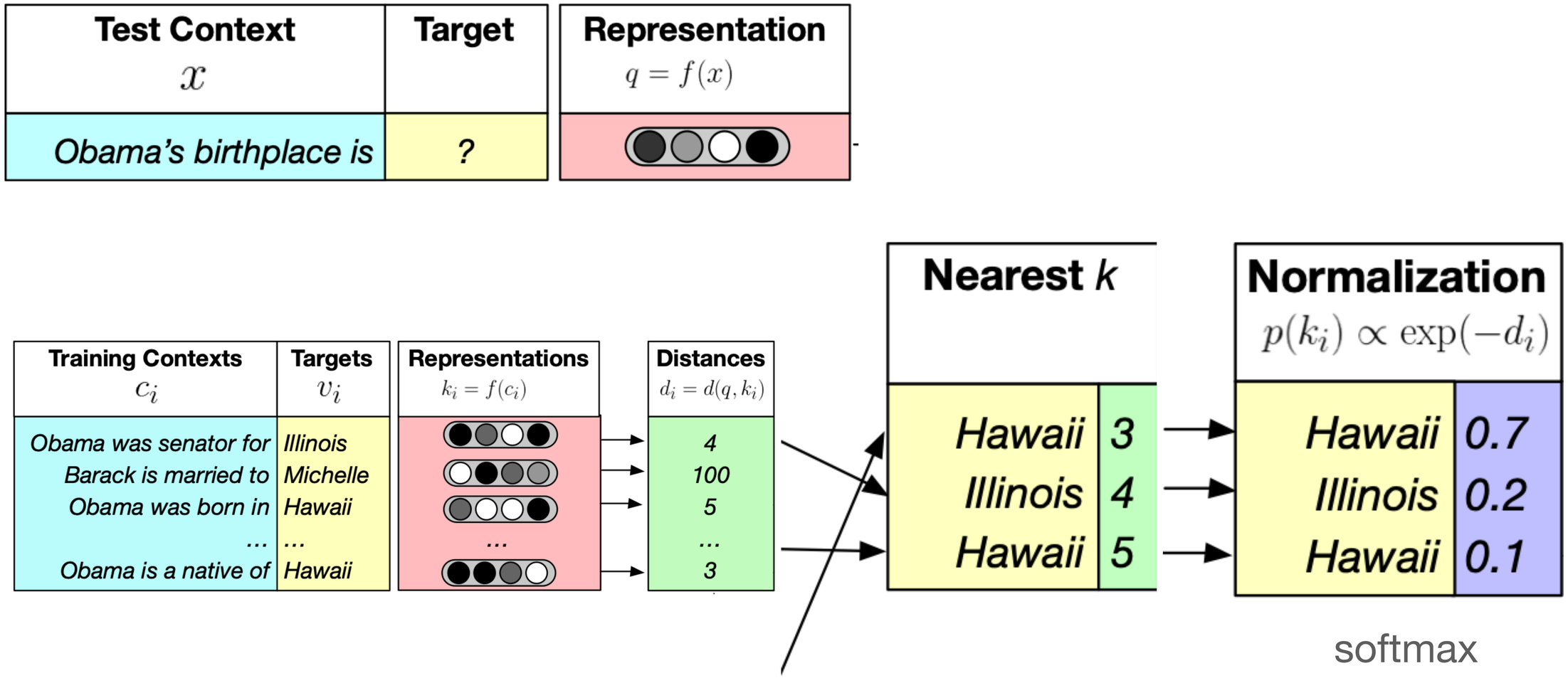
kNN LM prediction (step 2)

Test Context x	Target	Representation $q = f(x)$
Obama's birthplace is	*?*	

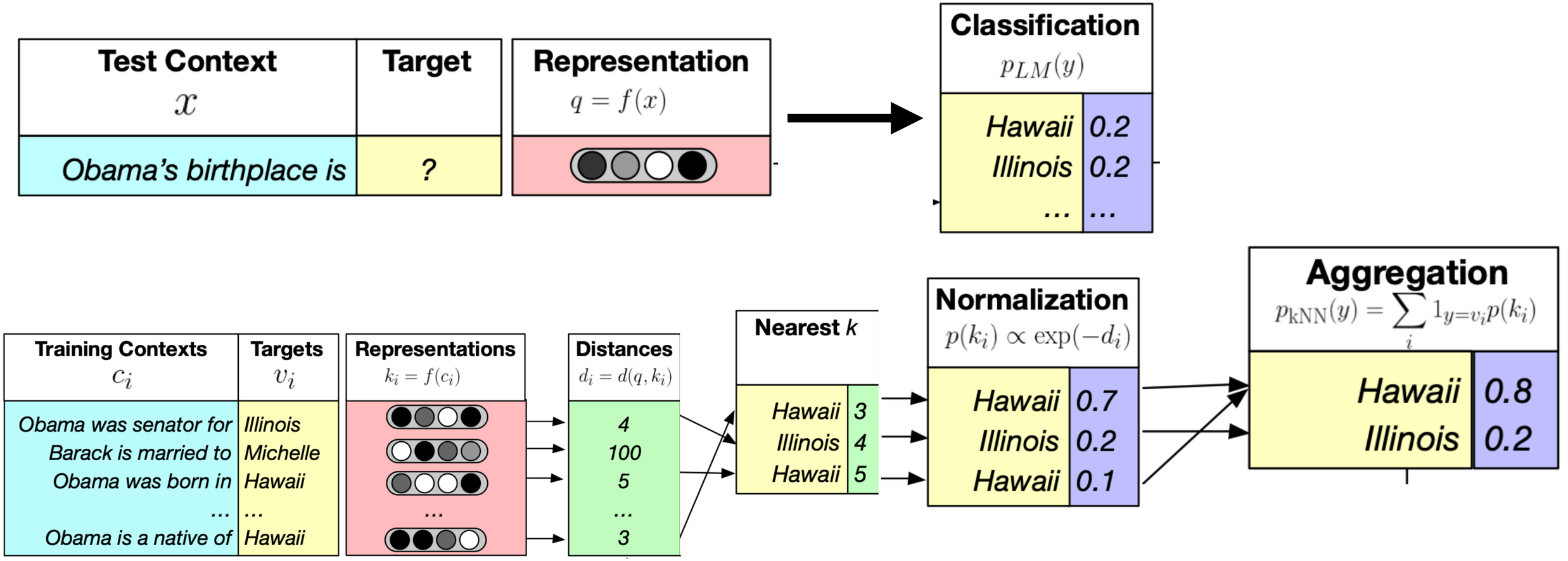
Training Contexts c_i	Targets v_i	Representations $k_i = f(c_i)$	Distances $d_i = d(q, k_i)$
Obama was senator for	*Illinois*		*4*
Barack is married to	*Michelle*		*100*
Obama was born in	*Hawaii*		*5*
...	*...*	*...*	*...*
Obama is a native of	*Hawaii*		*3*

Nearest *k*	
Hawaii	*3*
Illinois	*4*
Hawaii	*5*

kNN LM prediction (step 3)



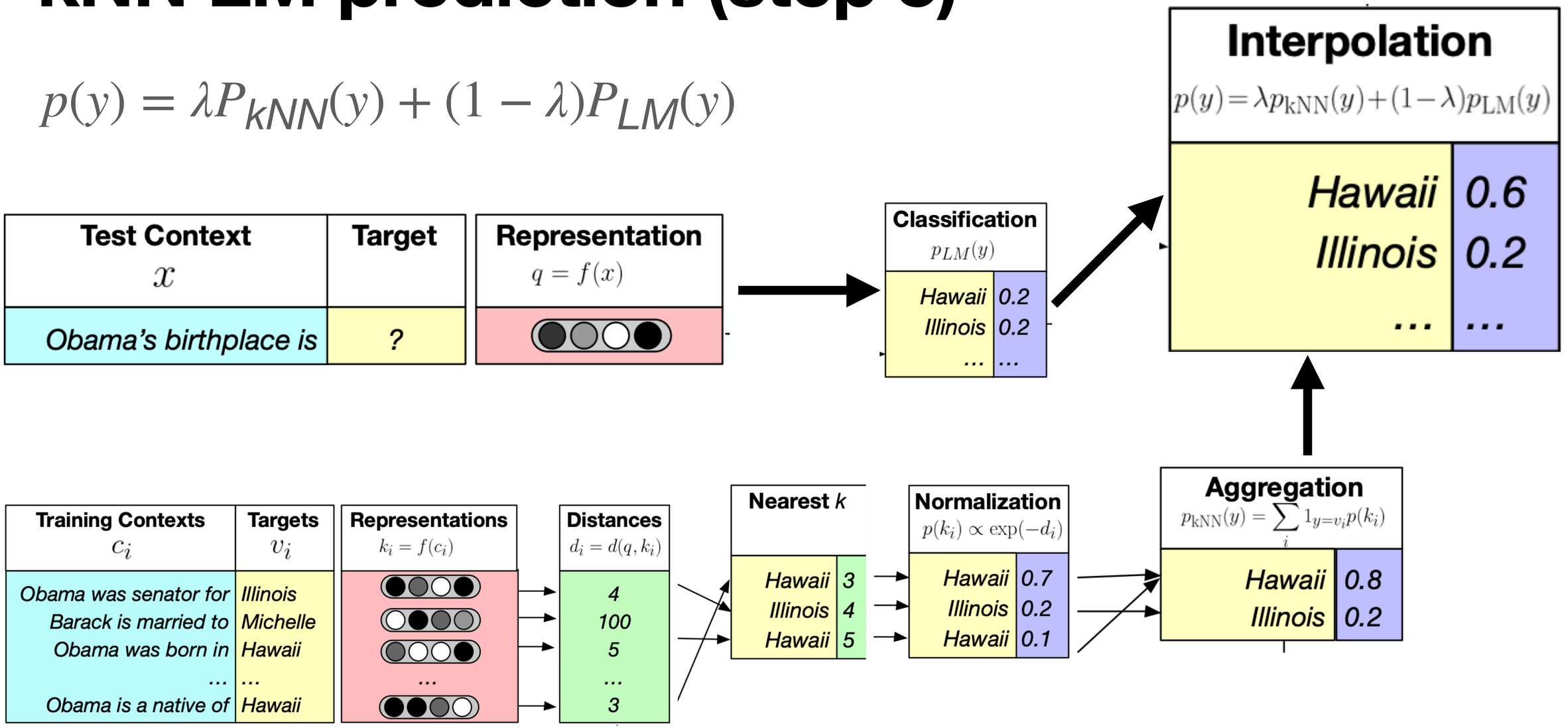
kNN LM prediction (step 4)



$$P_{kNN}(y) \approx \sum_{k_i, v_i \in \mathcal{N}} 1_{y=v_i} \exp(-d(k_i, f(x)))$$

kNN LM prediction (step 5)

$$p(y) = \lambda P_{kNN}(y) + (1 - \lambda) P_{LM}(y)$$



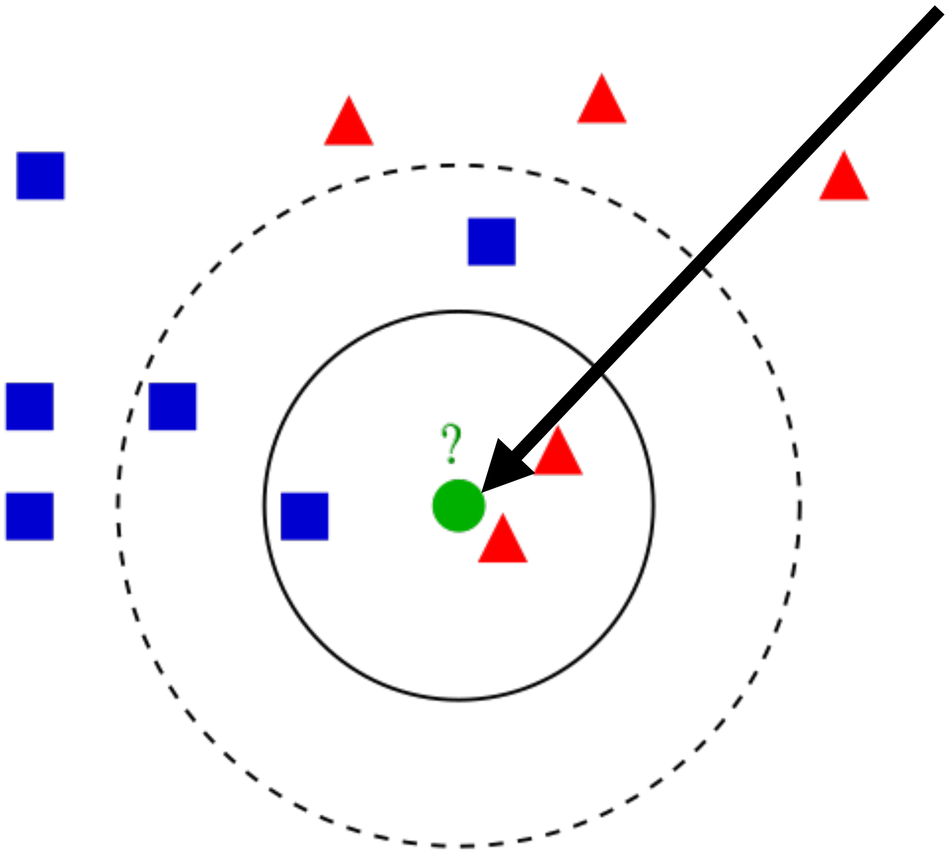
Model	Perplexity (↓) Dev	Test	# Trainable Params
Baevski & Auli (2019)	17.96	18.65	247M
+Transformer-XL (Dai et al., 2019)	-	18.30	257M
+Phrase Induction (Luo et al., 2019)	-	17.40	257M
Base LM (Baevski & Auli, 2019)	17.96	18.65	247M
+kNN-LM	**16.06**	**16.12**	247M
+Continuous Cache (Grave et al., 2017c)	17.67	18.27	247M
+kNN-LM + Continuous Cache	**15.81**	**15.79**	247M

Table 1: Performance on WIKITEXT-103. The kNN-LM substantially outperforms existing work. Gains are additive with the related but orthogonal continuous cache, allowing us to improve the base model by almost 3 perplexity points with no additional training. We report the median of three random seeds.

Nearest neighbour search

Nearest Neighbour

k neighbours in vector space



- The query vector is compared to other data vectors in the same vector space.
- Choose the class that has the most representatives inside the search perimeter.
- The search perimeter is determined by the cosine similarity of the query vector to the vectors stored in a kNN storage
- Efficient disk based kNN retrieval for very large sets of vectors is available: SCaNN, FAISS, annoy, etc.

Different types of ANNs

Vector transformation and encoding (focus on reduced memory)

- Quantization (FAISS, ScaNN)

Structures for searching (focus on fast search)

- Hashes (LSH)
- Trees (ANNOY)
- Graphs (HNSW)

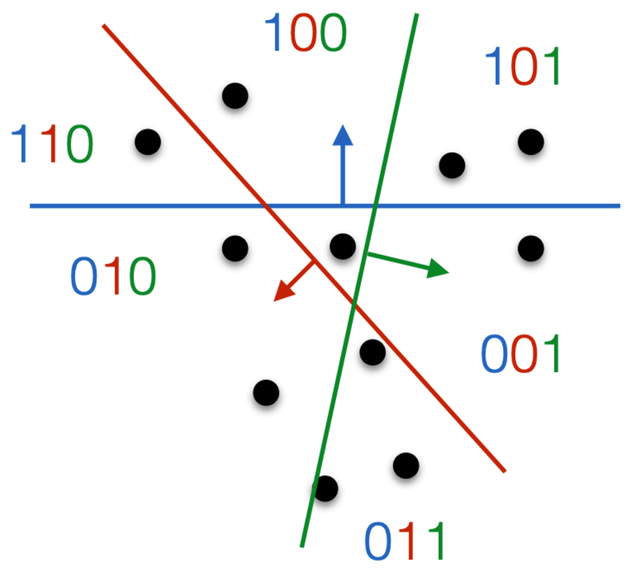
Methods can be combined

FAISS library: https://github.com/facebookresearch/faiss

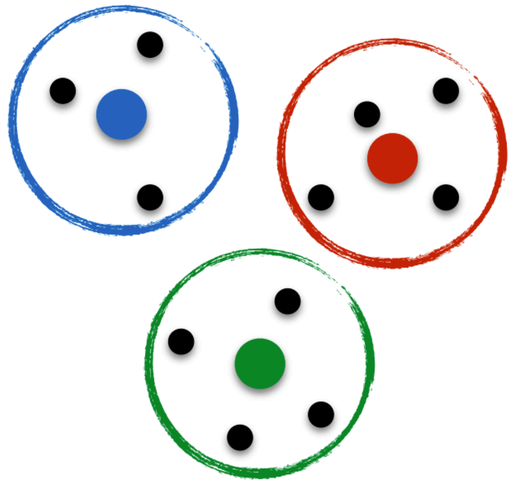
Approximately nearest neighbours

- 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

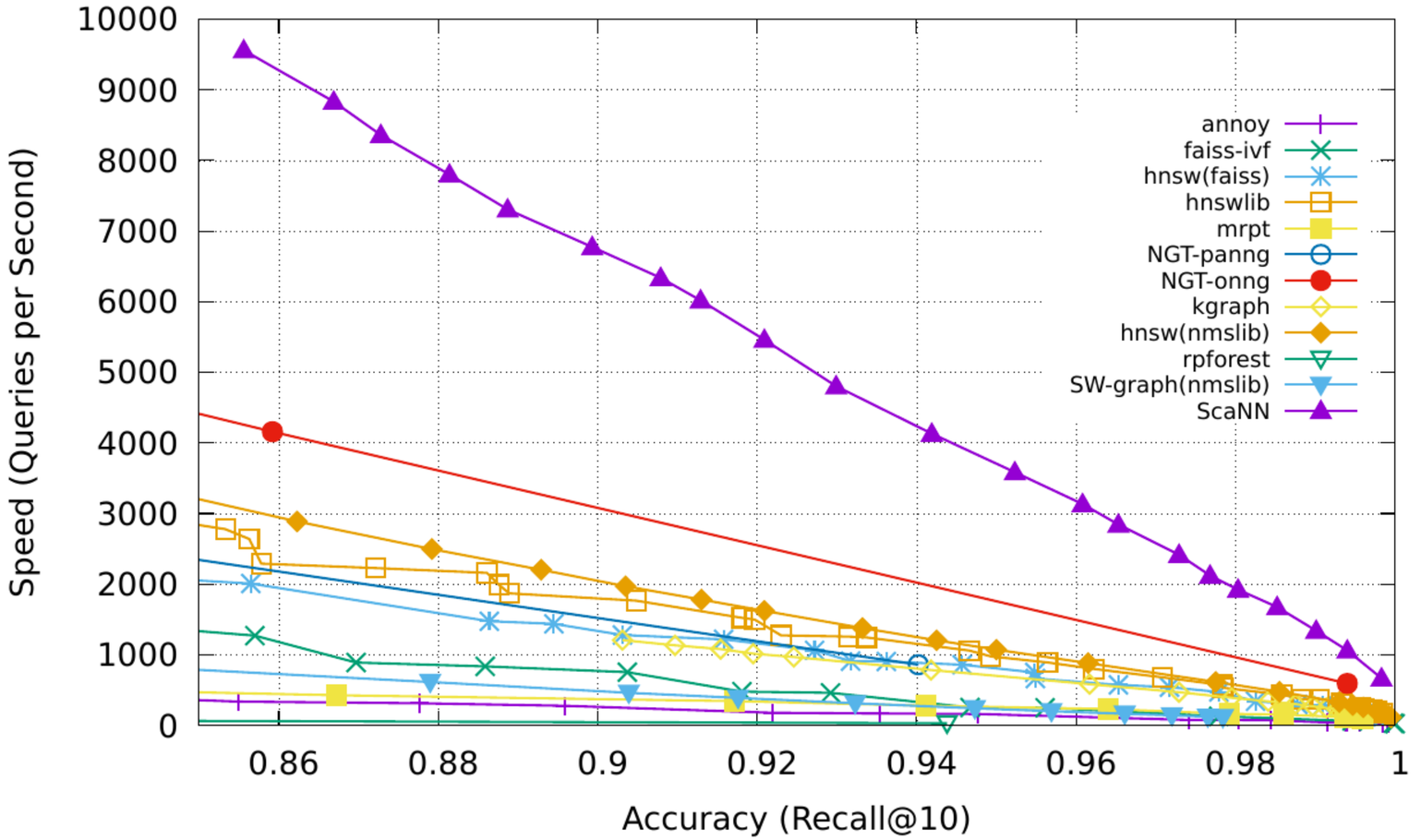


Efficient retrieval from memory

Approximate nearest neighbours (ANN) algorithms to return top k nearest neighbours using maximum inner product search (MIPS)

- LSH (Locality Sensitive Hashing) - hashing function so that similar inputs are mapped to same buckets with high probability
- ANNOY (ANN Oh Yeah) - Random projection trees where nodes splits input space into half
- HNSW (Hierarchical Navigable Small World)
  - Hierarchical layers of small-world graphs (points in the bottom layers)
  - Can be used with FAISS
- FAISS (Facebook AI Similarity Search) - vector quantization - partition vector space into clusters
- ScaNN (Scalable Nearest Neighbours) - Anisotropic vector quantization (quantize points while maintaining distances)

https://lilianweng.github.io/posts/2023-06-23-agent/

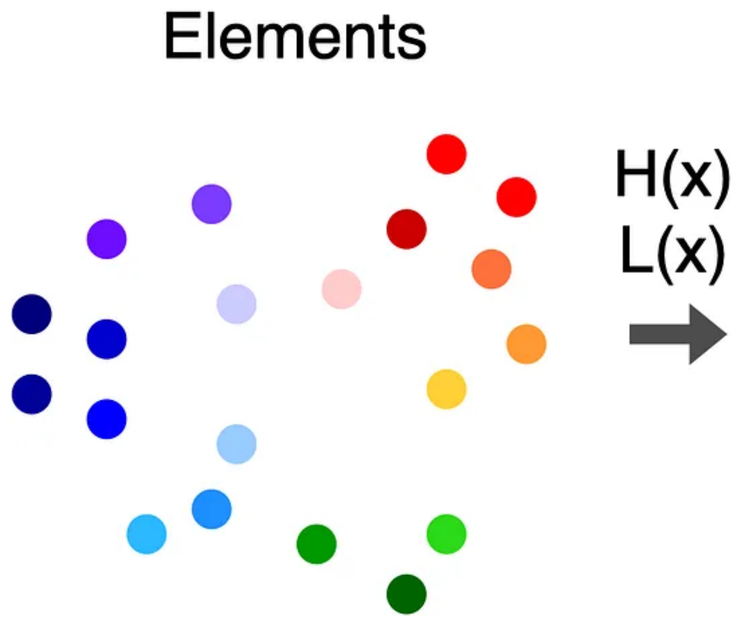


https://blog.research.google/2020/07/announcing-scann-efficient-vector.html

Note: nmslib focus is on non-metric spaces

Locality Sensitive Hashing (LSH)

- Hash-based ANN
- Construction
  - Map data points into buckets so points near each other are in the same buckets with high probability
- Search



LSH Table

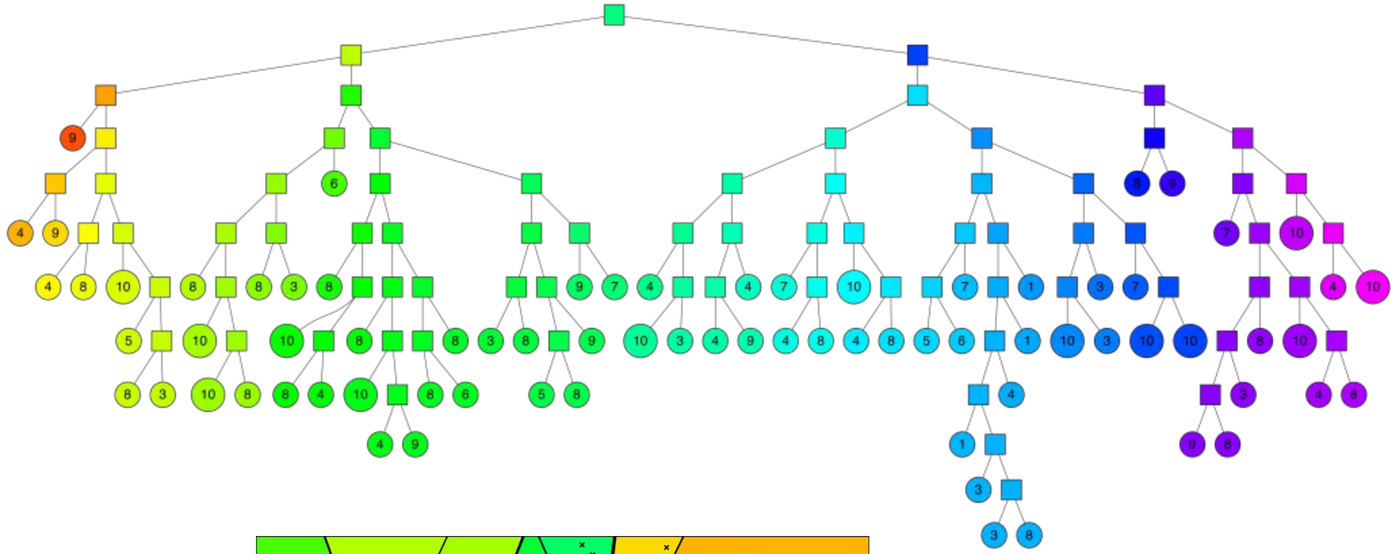
L(x)	Collisions
1	
2	
3	
4	
5	

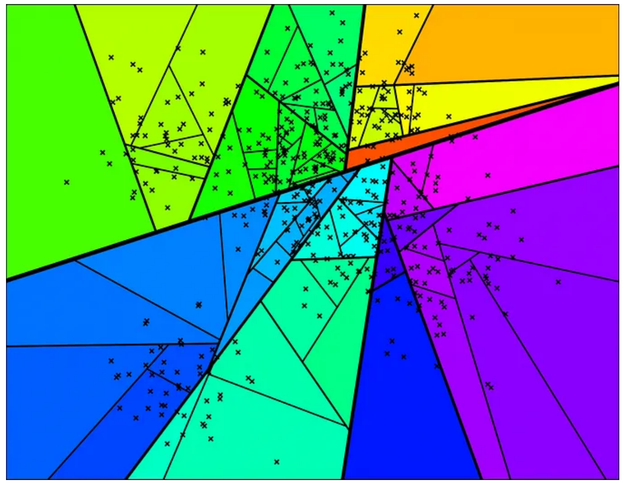
Hash Table

H(x)	Collisions
1	
2	
3	
4	
5	

ANNOY

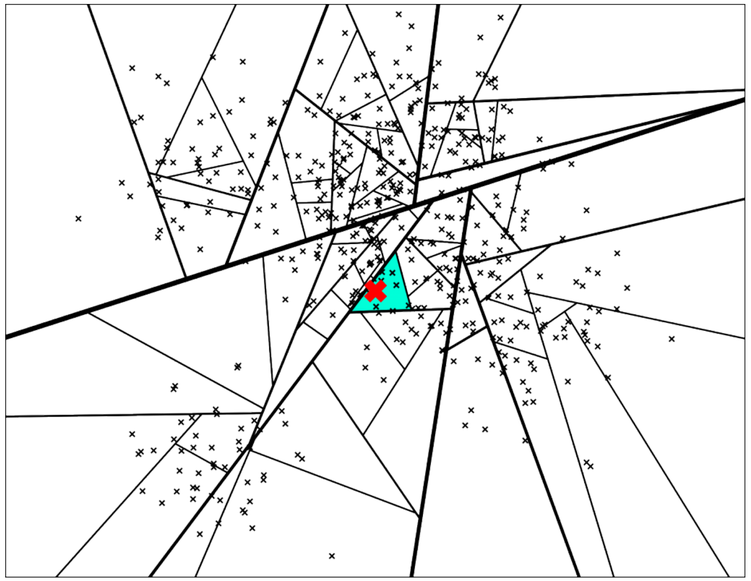
- Tree-based ANN
- Construction
  - Pick two points at random and split the space in two by the hyperplane between the two points
  - Repeat recursively until number of points associated with a node is of size < threshold
  - Parameters: number of trees
- Search
  - Traverse down the tree from the root



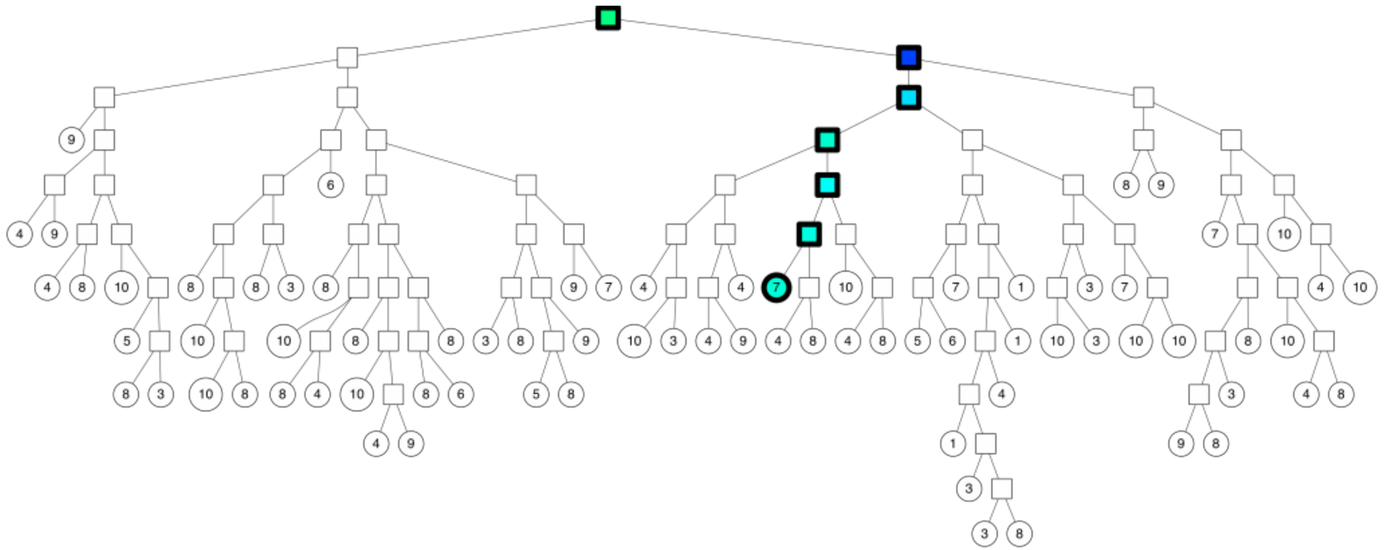


https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

ANNOY



- Tree-based ANN
- Construction
  - Pick two points at random and split the space in two by the hyperplane between the two points
  - Repeat recursively until number of points associated with a node is of size < threshold
  - Parameters: number of trees
- Search
  - Traverse down the tree from the root



https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html

Hierarchical navigable small world (HNSW)

- Graph based ANN
- Hierarchical layers of Navigable Small World Graphs

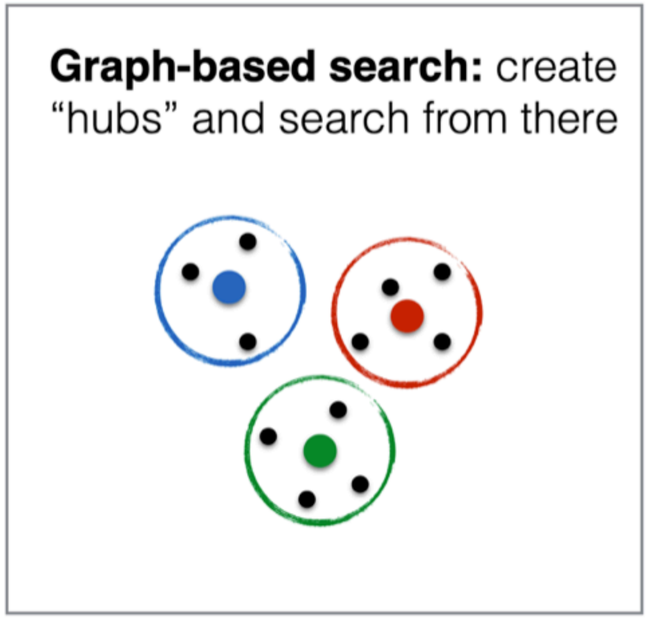
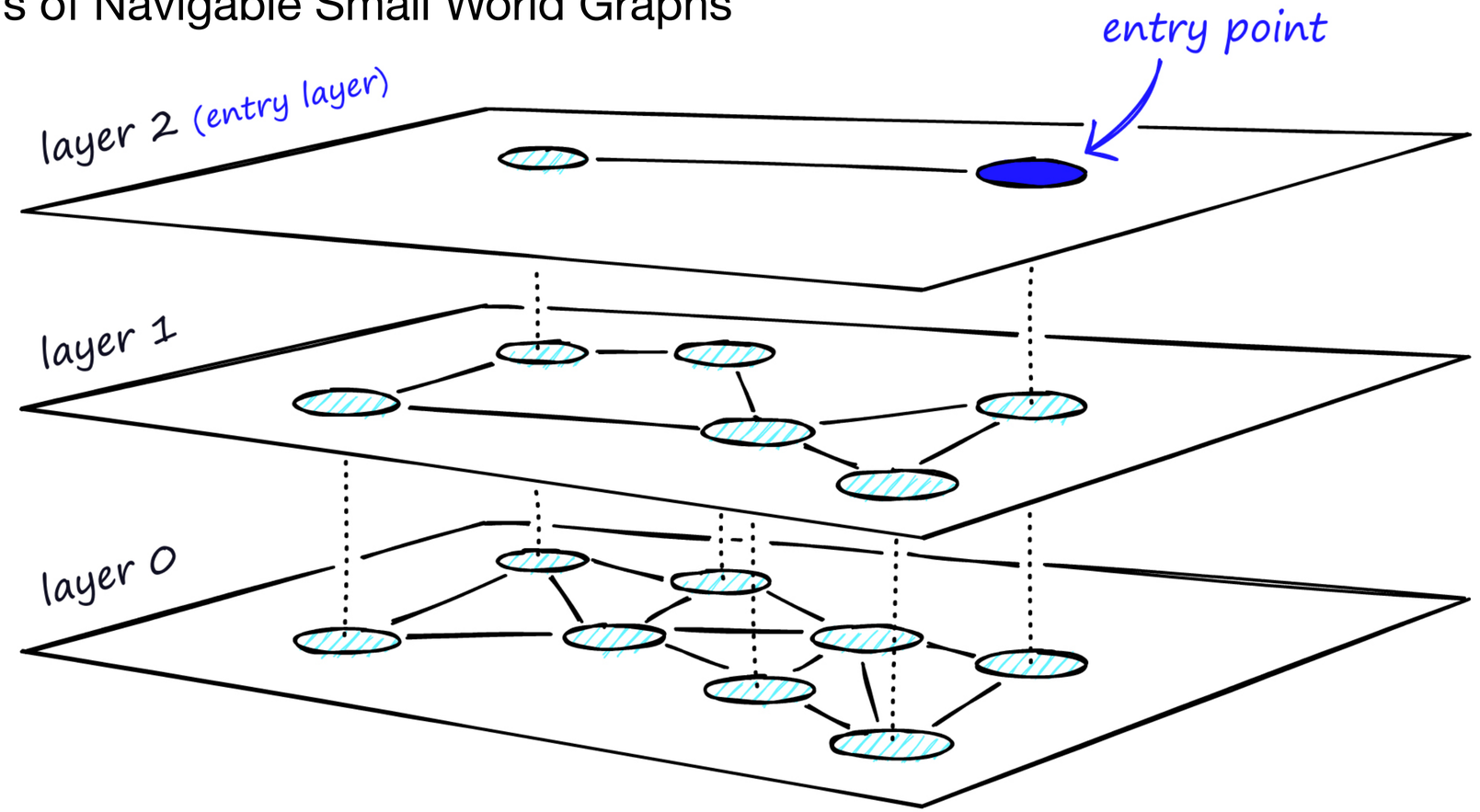


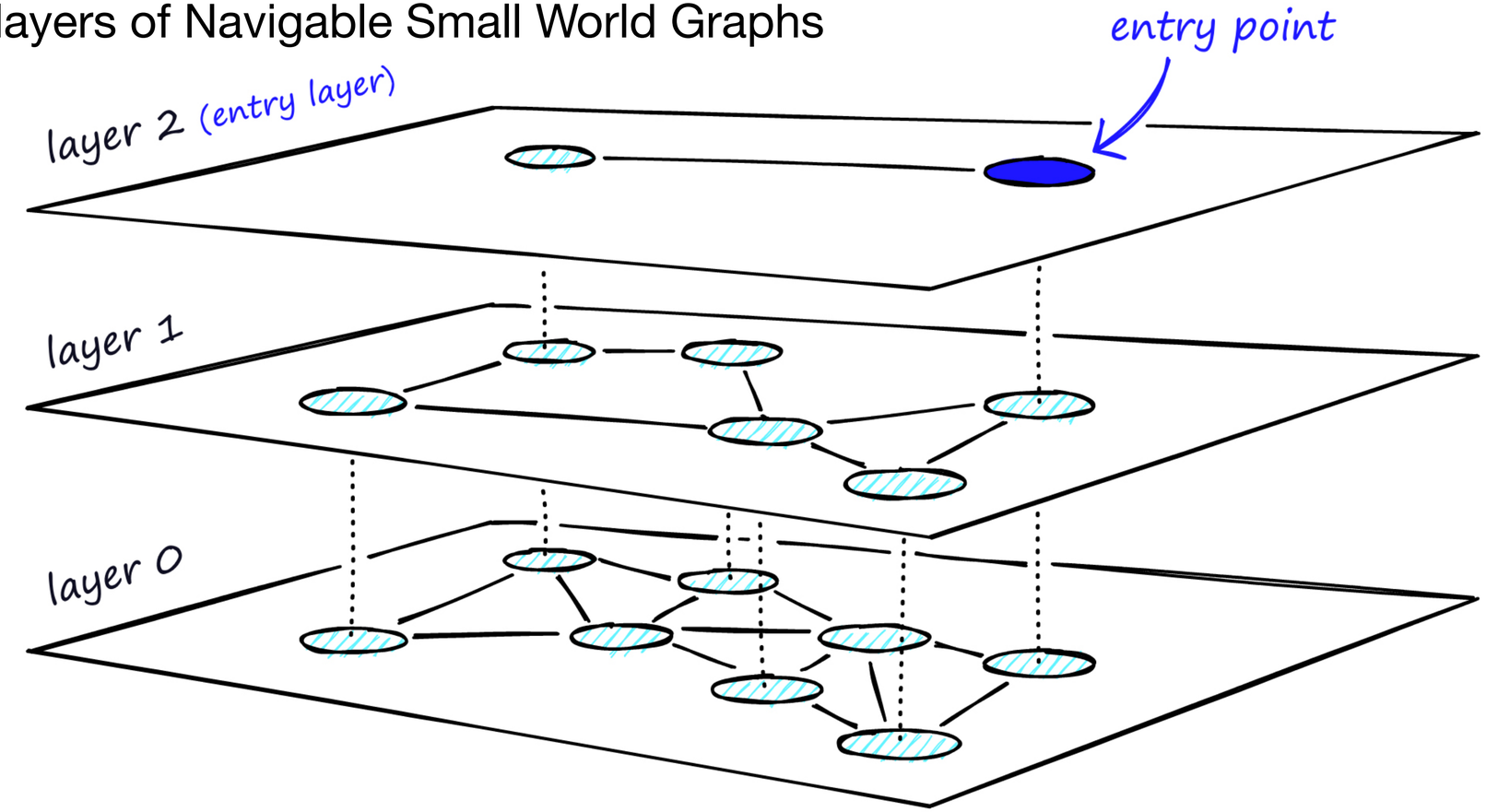
Figure credit: Graham Neubig



https://www.pinecone.io/learn/series/faiss/hnsw/

Hierarchical navigable small world (HNSW)

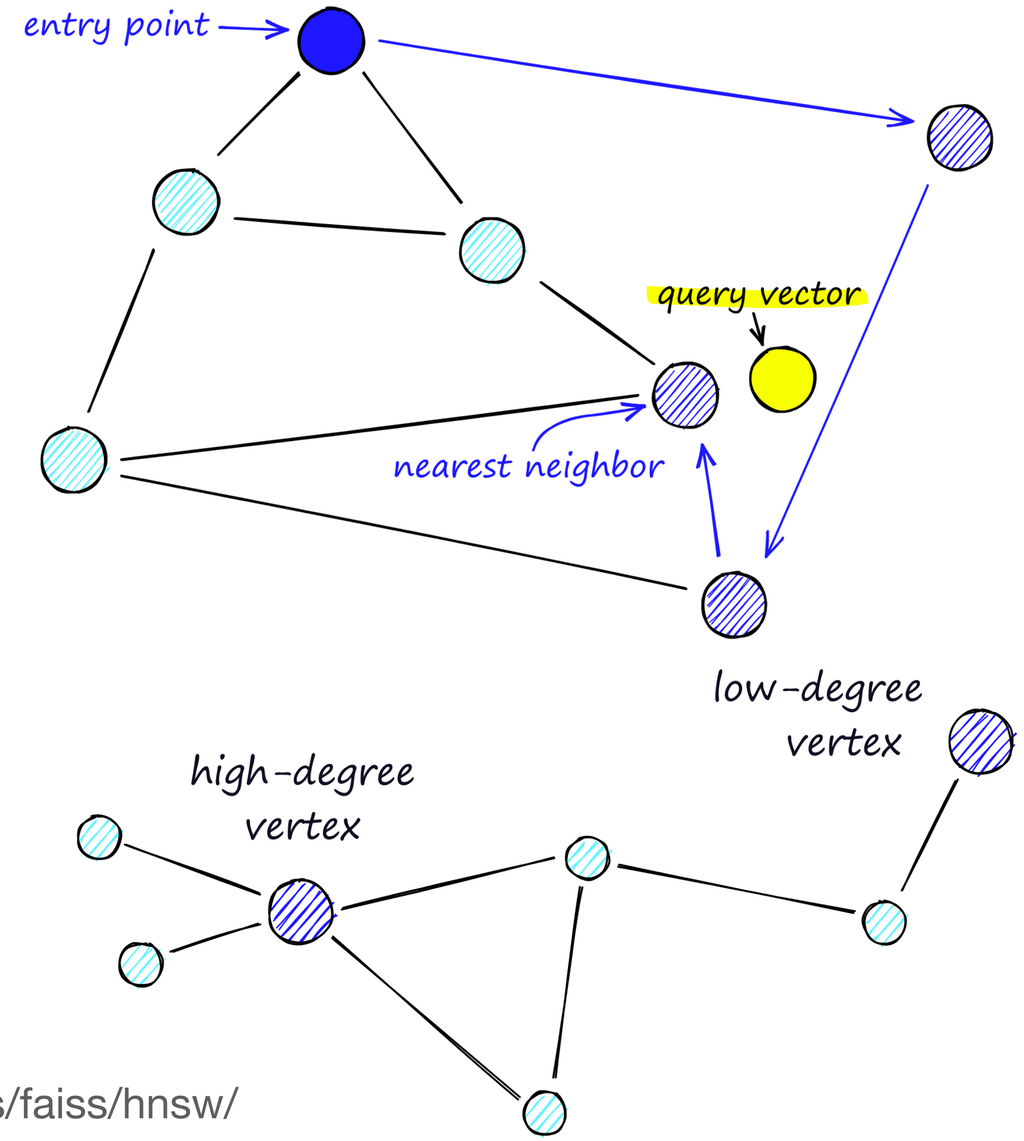
- Graph based ANN
- Hierarchical layers of Navigable Small World Graphs



https://www.pinecone.io/learn/series/faiss/hnsw/

HNSW: Navigable Small World Graphs

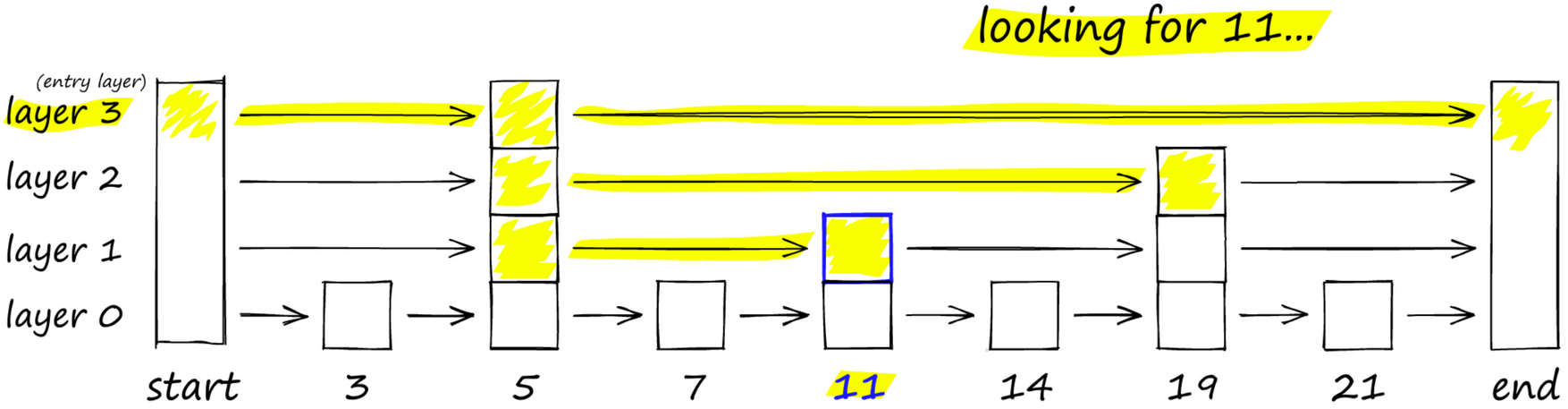
- Navigable Small World Graphs
  - Built on the idea that there real-world networks exhibit the small-world phenomena: there are high-degree nodes that link together the graph in small number of steps
  - Proximity graph: two vertices are linked if they are close together
  - Proximity graph with both long and short range links
  - Search
    - Start at entry point, greedily select neighbours that are closer to our query vector
    - Search undergoes two phase types (is it just two phases?):
      - “zoom in” pass through high-degree nodes
      - “zoom out” pass through low-degree nodes



https://www.pinecone.io/learn/series/faiss/hnsw/

HNSW: Probability skip list

- Probability skip list [Pugh 1990]
  - Layers of sorted linked lists
    - as we go down, the number of entries that are ‘skipped’ decreases and the links get denser
  - Start at the highest layer with the longest skips
    - If the current node key is larger than the search target key, then move down to the next level, go back to the previous node

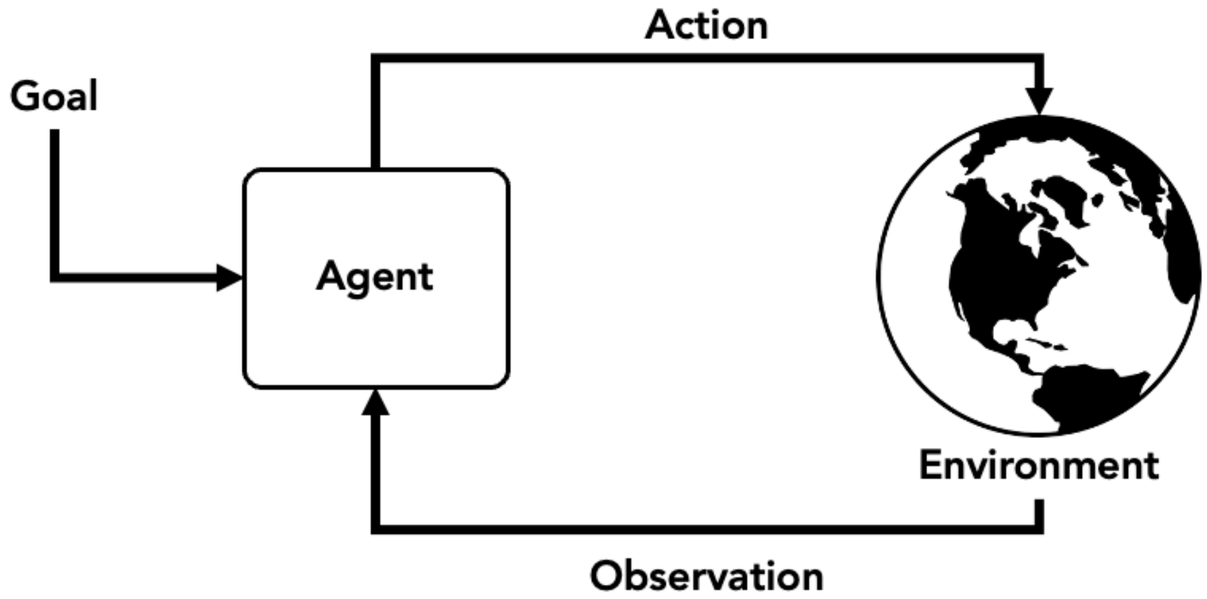


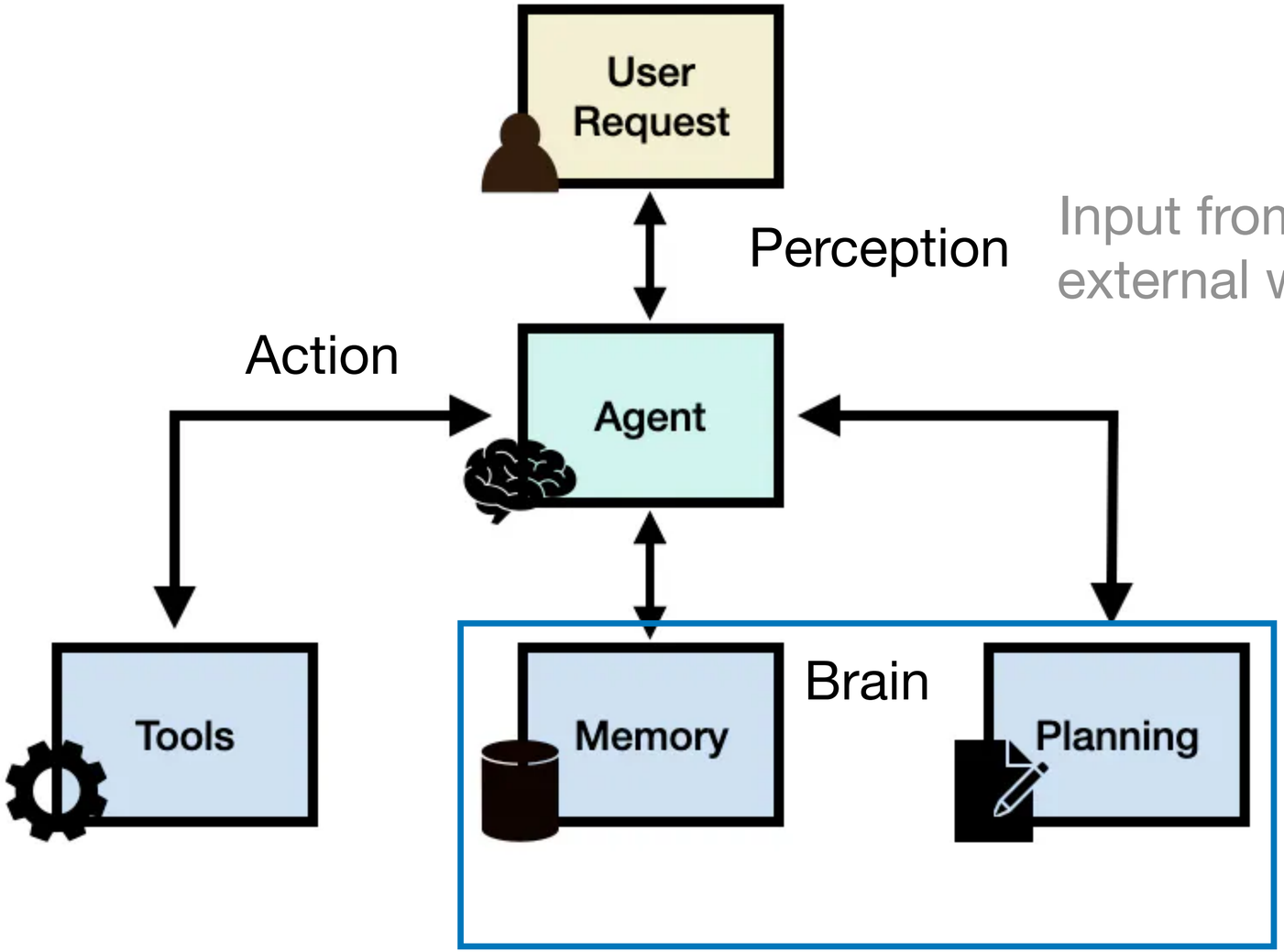
https://www.pinecone.io/learn/series/faiss/hnsw/

Beyond retrieval

LLMs as agents

- There is a lot of knowledge in LLMs
- But they can't "act"
- Can we leverage LLMs to build a "smart" agent that can interact with the environment to achieve given goals?





API calls to external services, expert modules, etc

Calendar()
Calculator()
ExecuteCode()
Search()

Reflection
Self-critics
Chain-of-thoughts
Subgoal decomposition

https://www.promptingguide.ai/research/llm-agents

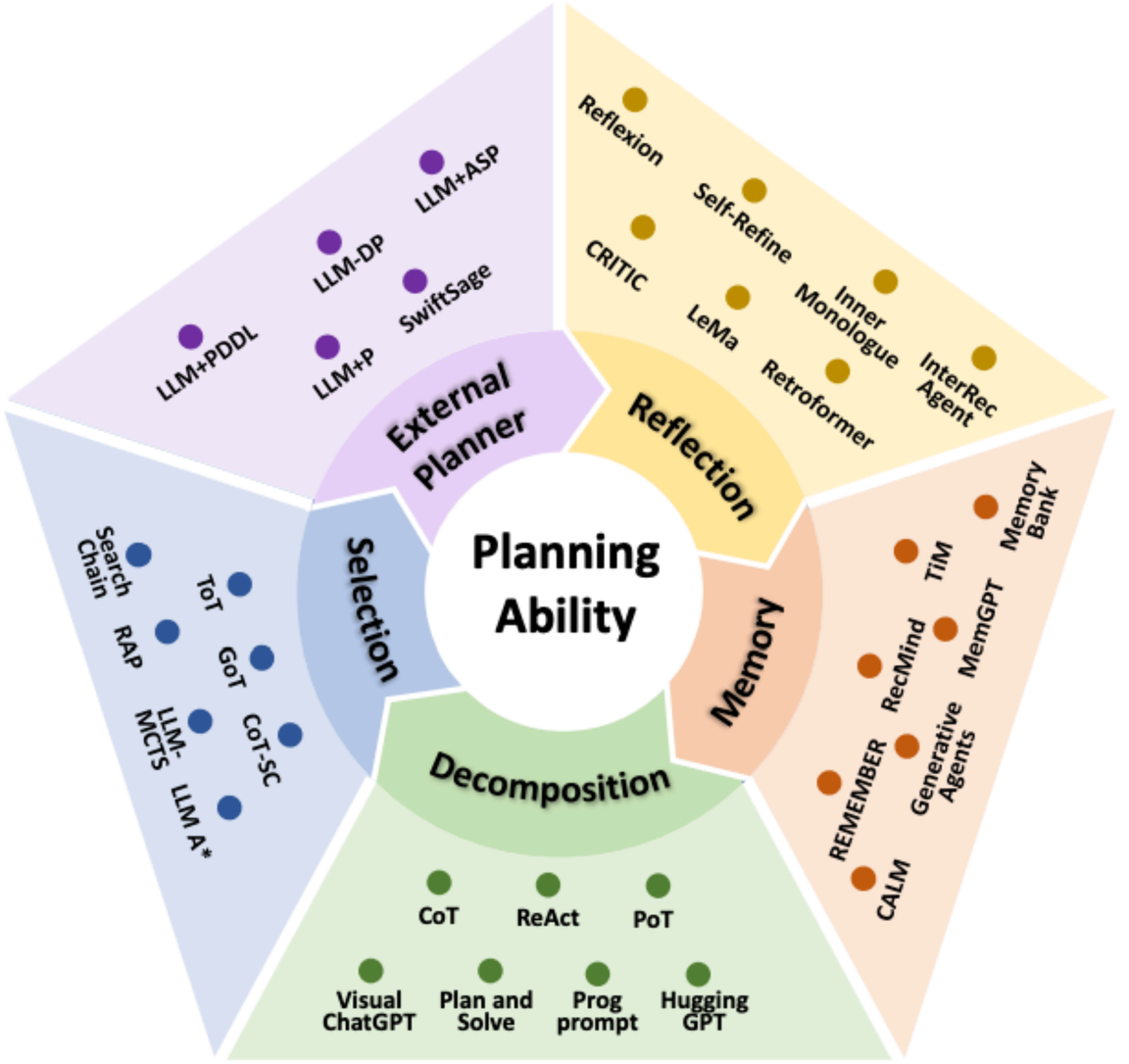
Planning

- What does the agent need to do to accomplish a specified goal?
- Low-level vs high-level planning
  - High-level plan: Identify subgoals for a long-horizon task
  - Low-level plan (sometime low-level control): Identify sequence of actions
- Traditionally use symbolic reasoning
  - Hard to recover from errors
  - Difficult to convert expert knowledge into planning languages such as PDDL (Planning Domain Definition Language)
- Use of LLM for planning
  - Lot of expert domain knowledge already encoded in language
  - Can we use LLMs to help us plan?

Taxonomy for planning with LLMs

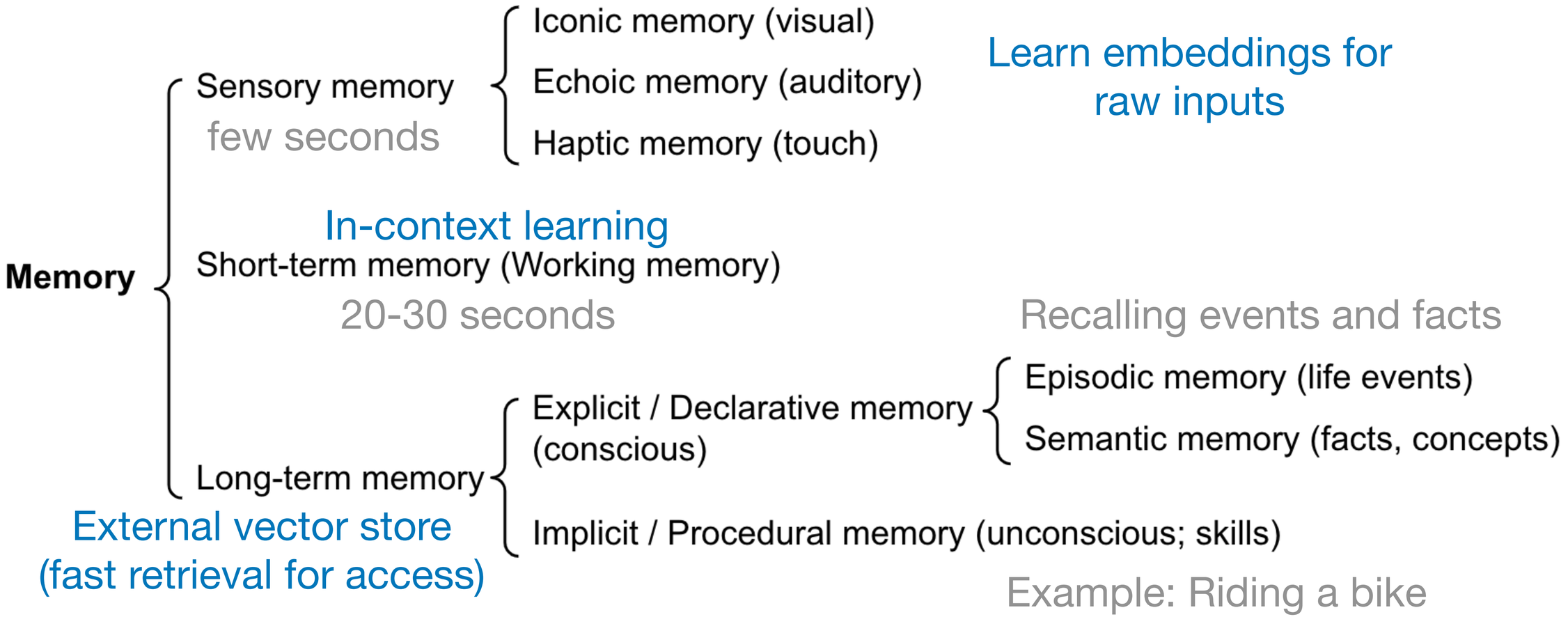
Given environment and sequence of actions, and an overall task goal, identify subgoals and actions

- **Task decomposition** - figure out subgoals, do planning for subgoals if needed
- **Multi-plan selection** - generate multiple plans and then select one
- **External planner** - LLM used to formalized the problem which is passed to an external planner
- **Reflection and refinement** - After obtaining a plan, the LLM future reflects on the plan and refine it to fix any issues with the original plan
- **Memory-augmented planning** - Uses external memory to retrieve information (common sense knowledge, domain-specific knowledge, etc) and then determines plan based on that



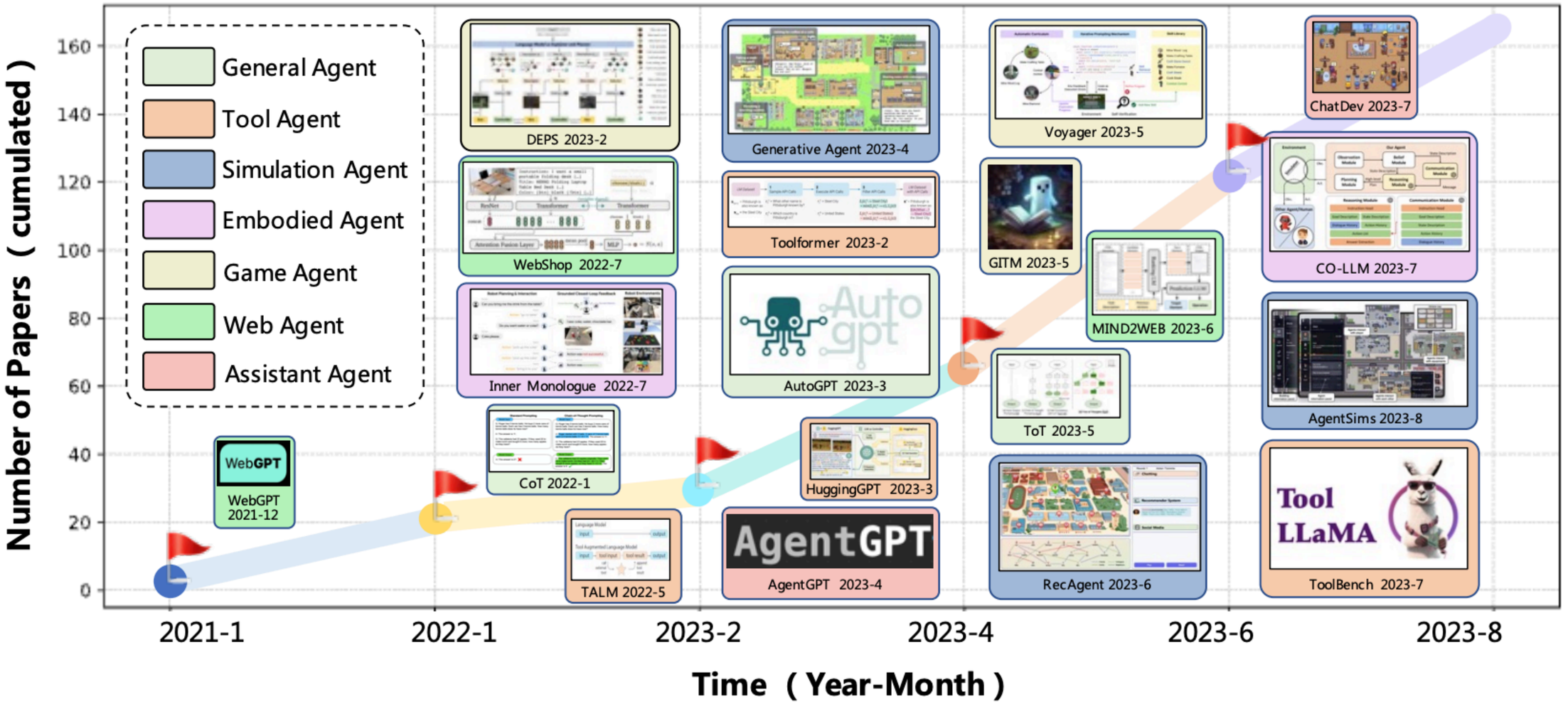
Understanding the planning of LLM agents: A survey [Huang et al, 2024]

Types of memory



https://lilianweng.github.io/posts/2023-06-23-agent/

Lots of work!



A Survey on Large Language Model based Autonomous Agents [Wang et al, 2023]

Learn about LLM agents from Berkeley!

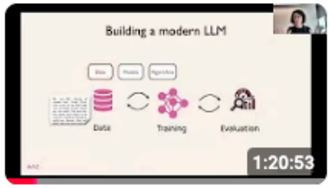
Advanced LLM Agents MOOC ▶ Play all

https://rdi.berkeley.edu/adv-llm-agents/sp25

CS 194/294-280 (Advanced LLM Agents) - Lecture 6,...

Berkeley RDI Center on Decentrali...
1 waiting
• Scheduled for 3/10/25, 5:00 PM

Notify me



CS 194/294-280 (Advanced LLM Agents) - Lecture 4,...

Berkeley RDI Center on Decentrali...
3.1K views
• Streamed 13 days ago

CC

CS 194/294-280 (Advanced LLM Agents) - Lecture 3, Yu...

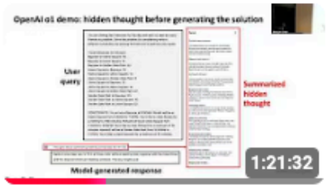
Berkeley RDI Center on Decentrali...
4.8K views
• Streamed 3 weeks ago

CC

CS 194/294-280 (Advanced LLM Agents) - Lecture 2,...

Berkeley RDI Center on Decentrali...
5.9K views
• Streamed 1 month ago

CC



CS 194/294-280 (Advanced LLM Agents) - Lecture 1,...

Berkeley RDI Center on Decentrali...
13K views
• Streamed 1 month ago

CC

LLM Agents MOOC ▶ Play all

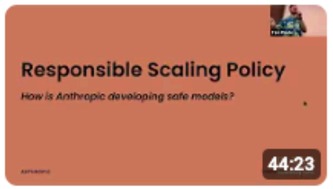
Large Language Model Agents MOOC F24

https://llmagents-learning.org/f24

CS 194/294-196 (LLM Agents) - Lecture 12, Dawn...

Berkeley RDI Center on Decentrali...
9.4K views
• Streamed 3 months ago

CC



CS 194/294-196 (LLM Agents) - Lecture 11, Ben...

Berkeley RDI Center on Decentrali...
4.3K views
• Streamed 3 months ago

CC

CS 194/294-196 (LLM Agents) - Lecture 10, Percy...

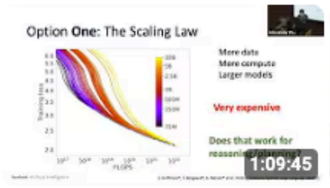
Berkeley RDI Center on Decentrali...
6.4K views
• Streamed 3 months ago

CC

CS 194/294-196 (LLM Agents) - Lecture 9, Jim Fan

Berkeley RDI Center on Decentrali...
7.1K views
• Streamed 4 months ago

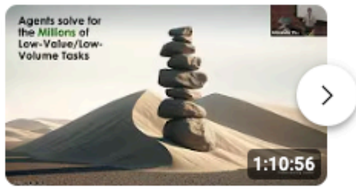
CC



CS 194/294-196 (LLM Agents) - Lecture 8,...

Berkeley RDI Center on Decentrali...
7.7K views
• Streamed 4 months ago

CC



CS 194/294-196 (LLM Agents) - Lecture 7, Nicolas...

Berkeley RDI Center on Decentrali...
8.5K views
• Streamed 4 months ago

CC