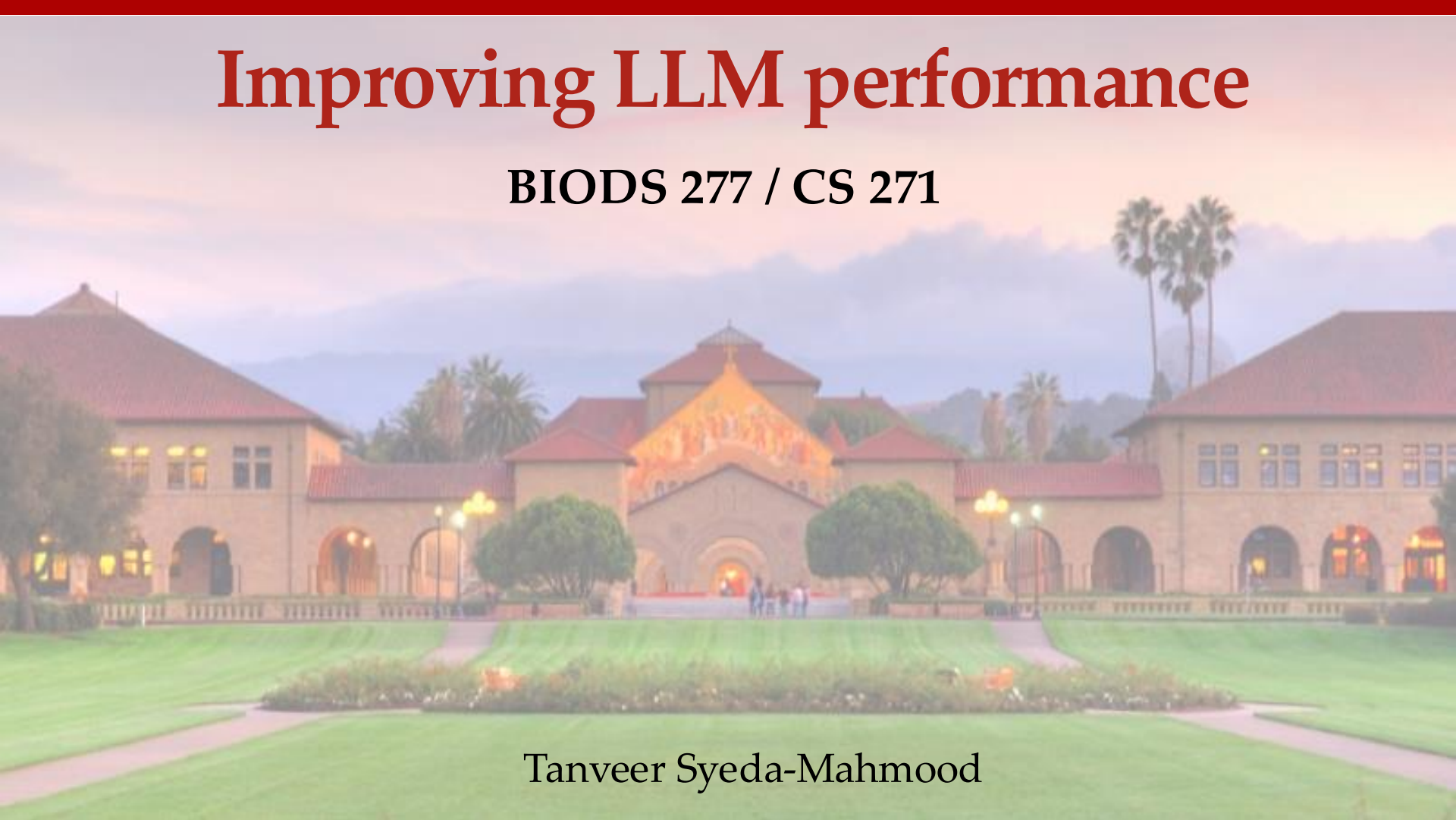
# Improving LLM performance

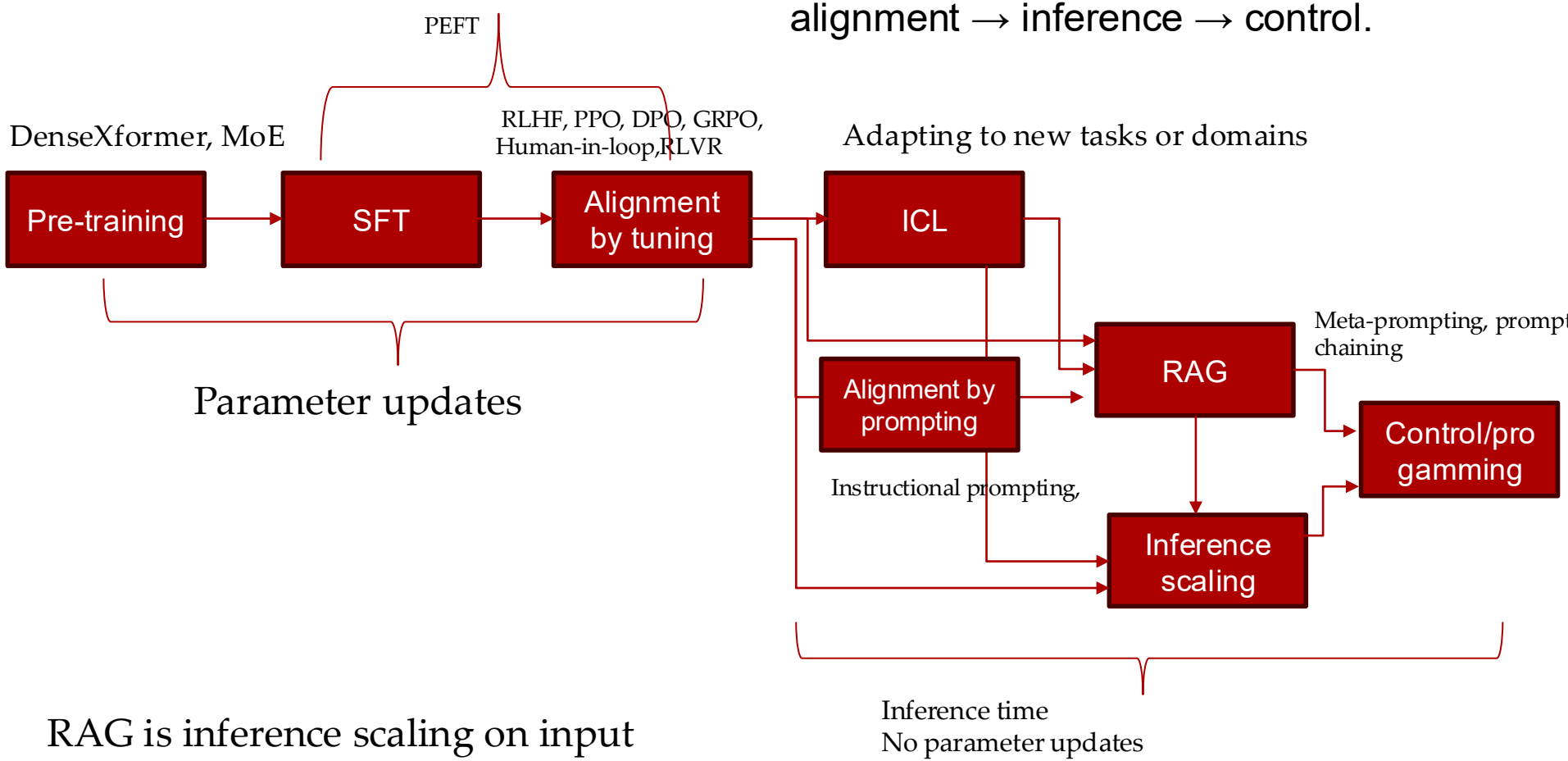
## BIODS 277 / CS 271



Tanveer Syeda-Mahmood

# The LLM Stack

LLM stack — architecture → training → alignment → inference → control.




RAG is inference scaling on input

# Prompting methods in alignment

- Zero-shot prompting
- Instructional prompting
- Object-guided prompting
- Rule-based filters
- Self-critiques

Prompts do matter:

- E.g. Different results with slightly different prompt
  - A photo of a plane
  - A photo of a plane parked at the gate
  - A photo of a plane parked at the gate or flying in the sky

The art of prompting

**Be Specific:** Prompt should be specific
**Indicate Intent:** Make sure your intent is clear
**Show Examples:** use them for formatting and inferring commonalities
**Given Clear Instructions:** The instructions shouldn't be ambiguous
**Indicate Desired Output:** Indicate the format of the output

Efficient prompting methods for large language models: A survey

# Elements of a prompt

A prompt contains any of the following elements:

**Instruction** - a specific task or instruction you want the model to perform

**Context** - external information or additional context that can steer the model to better responses

**Input Data** - the input or question that we are interested to find a response for

- **Output Indicator** - the type or format of the output.

# Few-shot prompting

- Is a type of "in-context learning" or "learning by example."
  - Pattern recognition:
    - identifying patterns in how inputs are transformed into outputs.
  - Task inference:
    - The nature of the task being asked to perform
  - Generalization:
    - Generalize from given examples to new inputs
  - Application:
    - Apply the learned pattern to new input
- What are some pitfalls?
  - What if there isn't enough variety?
  - What if there is too much variety?

Few shot example:

- Classify the sentiment of the third movie review. Use the information from the first two examples:
- Review: "This movie was a waste of time.
- "Sentiment: NegativeReview:
- "I couldn't stop laughing throughout the film!
- "Sentiment: PositiveReview:
- "The special effects were amazing, but the plot was confusing
- "Sentiment:"

Provide a possible diagnosis and explain your reasoning:
Symptoms: Fever, cough, fatigue
Diagnosis: Common cold
Explanation: The combination of fever, cough, and fatigue is typical of a common cold. No severe symptoms are present, suggesting a mild viral infection.
Symptoms: Chest pain, shortness of breath, dizziness
Diagnosis: Possible heart attack
Explanation: The combination of chest pain, shortness of breath, and dizziness are warning signs of a possible heart attack. Immediate medical attention is required.
Symptoms: Headache, sensitivity to light, nausea
Diagnosis:
Explanation:

https://www.datacamp.com/tutorial/few-shot-prompting

# Instructional prompting

- Explicitly telling a language model
  - *how it should behave*
  - *what task it should perform*
- using natural language instructions in the prompt

Example

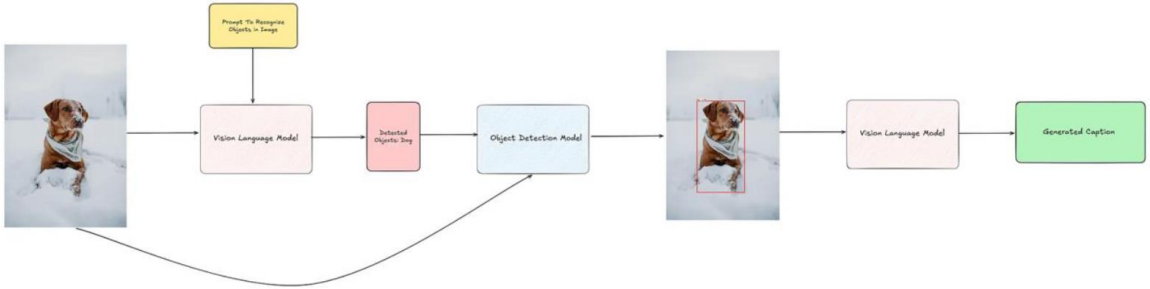
Read the following sales email. Remove any personally identifiable information (PII), and replace it with the appropriate placeholder. For example, replace the name "John Doe" with "[NAME]".
Hi John,
I'm writing to you because I noticed you recently purchased a new car. I'm a salesperson at a local dealership (Cheap Dealz), and I wanted to let you know that we have a great deal on a new car. If you're interested, please let me know.
Thanks,
Jimmy Smith
Phone: 410-805-2345 Email: jimmysmith@cheapdealz.com

Example from learnprompting.org

# Object-guided prompting

- Combined open vocabulary tagging with object detection to generate a more focused caption

system_prompt="""You are a helpful assistant that can analyze images and provide captions. You are provided with images that also contain bounding box annotations of the important objects in them, along with their labels.
Analyze the overall image and the provided bounding box information and provide an appropriate caption for the image.""",




https://medium.com/data-science/prompting-with-vision-language-models-bdabe00452b7

# Exercise: Guess prompting type

```
Classify the text into neutral, negative or positive.
Text: I think the vacation is okay.
Sentiment:
```

```
A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word
whatpu is:
We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word
farduddle is:
```

https://arxiv.org/abs/2005.14165

Here are the rules for question and answer generation. 1) The question should not be a multiple choice question and answer. 2) The answers should be in a single paragraph (no bullet points). 3) The questions should be tagged as Question: and the answers should be tagged as Answer: 4) Do not generate any other text before and after the questions and answers. 5) If you are unable to generate question and answers your response should be - Unable to generate questions and answers. 6) Do not repeat the same question. Using these rules, generate 5 questions and answers based on the following text:

Language models are few shot learners

# Thinking exercise

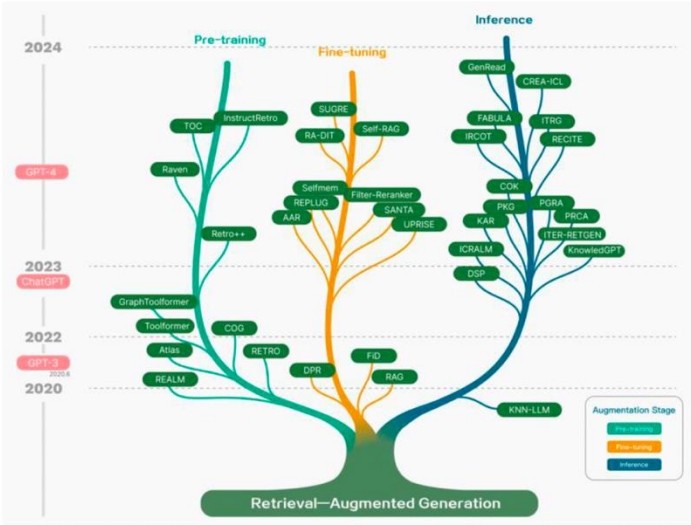
- What is the difference between alignment vs. scaling?
- How is prompting during alignment different from prompting during scaling or ICL?
- How come no parameter updates still improve performance?
- Why not lightweight in-context training at inference time?
- How are mixture of expert models different from model ensembles?

# Why does prompting improves performance?

- Since no LLM parameters are updated, how are we seeing improved performance?
  - By prompting, we are **selecting and configuring behavior** the model already knows how to perform.
  - All of the input is used as the context window and the activations are produced through self-attention
- Do we need sample answers if no loss is being minimized as in training?
  - Sample answers are needed to affect generation because they **shape the hidden representation of the sequence prefix**, which in turn changes the probability distribution for the next token.
- So is there a guarantee that the machine will understand the examples to produce an analogous response?
  - Correctness only exists as *statistical regularity learned during training*. The model is **predicting continuation given evidence.**
  - So there is a *statistical guarantee under the training distribution*, not a logical one.

# Retrieval augmented generation

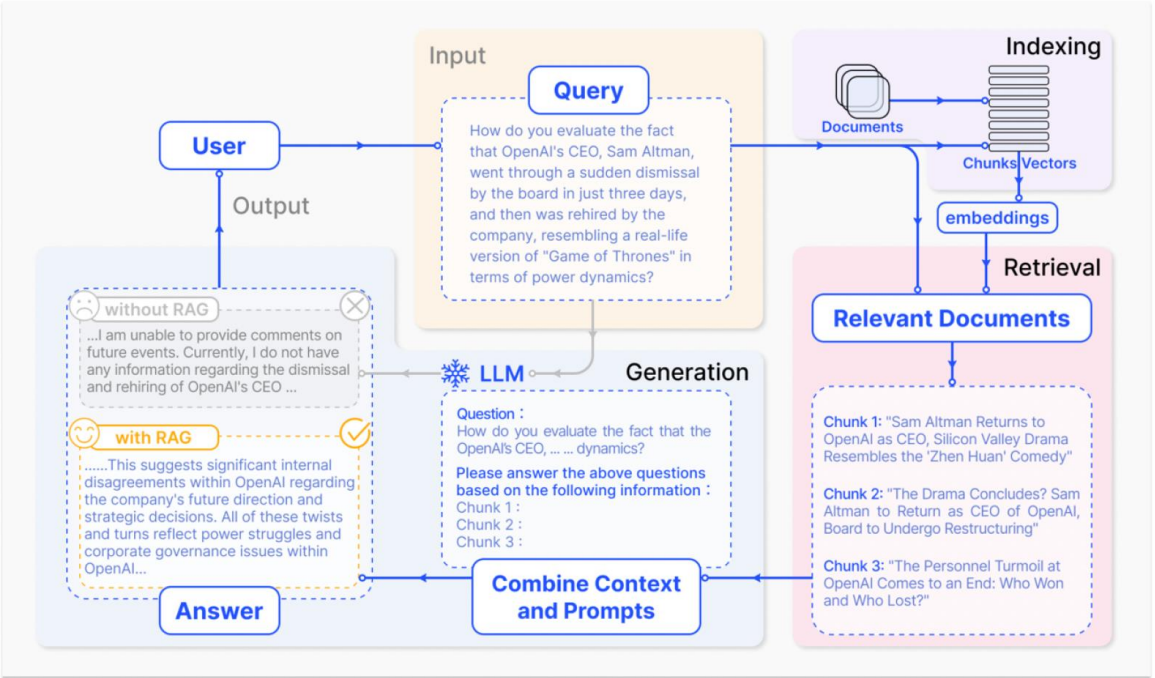
- A way to improve the performance of LLM
  - Could be used during pre-training or fine-tuning but mostly used during inference
- Practical approach to improve the model response with latest onsite knowledge
- Minimizes hallucinations with provided knowledge
- Time-relevant responses
- Increases transparency and gives traceable reasoning
- Cost-effective way to deploy models
- Pivotal technology for chatbots and practical applications.
- Widely implemented in commercial applications and popularized vector databases




Retrieval-Augmented Generation for Large Language Models: A Survey, Jan 2024

# RAG – Key idea

- Given a query to an LLM
  - Search/retrieve relevant documents
  - Construct prompts with the relevant documents
  - LLM to generate revised response
- How to determine relevancy?
- How does the new information integrate into LLM?
- How to evaluate the improvement?



# RAG – the formulation

Given a query $x$, the retriever selects the top-$k$ documents $\{d_1, \ldots, d_k\} \subset \mathcal{D}$ using a similarity function, defined as

$$\text{sim}(x, d_i) = f_q(x)^\top f_d(d_i) \tag{1}$$

where $f_q$ and $f_d$ are neural encoders for the query and documents, respectively. Similarity is computed using the dot product or cosine similarity in a shared embedding space.

The generator produces output $y$ by estimating the conditional probability over possible responses:

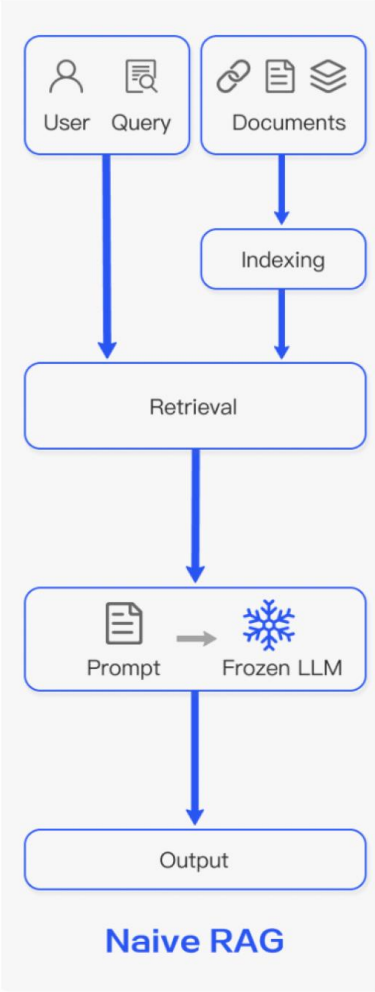
$$P(y \mid x) = \sum_{i=1}^{k} P(d_i \mid x) \cdot P(y \mid x, d_i) \tag{2}$$

Here, $P(d_i \mid x)$ can be derived from normalized similarity scores or treated uniformly in top-$k$ retrieval. The final output can be generated either by marginalizing across retrieved documents or by concatenating them as a single input:

$$y = \arg\max_{y'} P(y' \mid x, \{d_1, \ldots, d_k\}) \tag{3}$$

# Naïve RAG

- Indexing
  - Text conversion
  - Chunking documents
  - Encoding documents
  - Index generation (k-NN), e.g. Faiss index
- Retrieval
  - Cosine similarity retrieval by encoding query
- Generation
  - Responses to the information contained within the provided documents
  - In cases of ongoing dialogues, any existing conversational history can be integrated into the prompt,
  - enabling the model to engage in multi-turn dialog




- Problems:
  - Low precision and recall
  - Irrelevant context in responses
  - Augmentation from the retrieved chunks tricky due to redundancies and repetition
  - Novel content generation may not be possible

# Search/Retrieval methods for RAG

- Unsupervised approaches
  - Sparse retrieval
  - Dense retrieval
  - Sparse vector retrieval
- Supervised approaches
  - Neural IR
  - Re-ranking models

# Lexical or Sparse Retrieval Methods - Unsupervised

- Based on traditional document retrieval approaches
- Breaks up a document and query into tokens
- Match is determined using TF/IDF rules
  - Term Frequency (TF)
    - How often a term occurs in a document
  - Inverse Document Frequency (IDF)
    - How common is such occurrence across documents
- BM25 is a popular sparse retrieval method
  - Uses TF/IDF
  - Document Length Normalization
    - Measure should be robust to document lengths

Given a query $Q$, containing keywords $q_1, \ldots, q_n$, the BM25 score of a document $D$ is:

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where $f(q_i, D)$ is $q_i$'s term frequency in the document $D$, $|D|$ is the length of the document $D$ in words, and avgdl is the average document length in the text collection from which documents are drawn. $k_1$ and $b$ are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2.0]$ and $b = 0.75$.[1] $\text{IDF}(q_i)$ is the IDF (inverse document frequency) weight of the query term $q_i$. It is usually computed as:

$$\text{IDF}(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right)$$

where $N$ is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing $q_i$.

Sparse retrieval methods:

- Allow exact matching – both a boon and a curse
- Efficient inverted indexing

# Dense or Semantic retrieval - Unsupervised

- Given a set of documents
  - Extract chunks from documents
  - Vectorize the chunks using an embedding
- Given a query:
  - Vectorize the query using the same embedding
- Find the closest match to the query using distance or similarity metrics:
  - Inner product
  - Cosine similarity
  - L2 distance
  - Softmax over relevant scores to convert to probability
- When the documents grow large, the vectors could grow to very large vector stores
  - Compression
  - Approximate nearest neighbor search. Through indexing

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

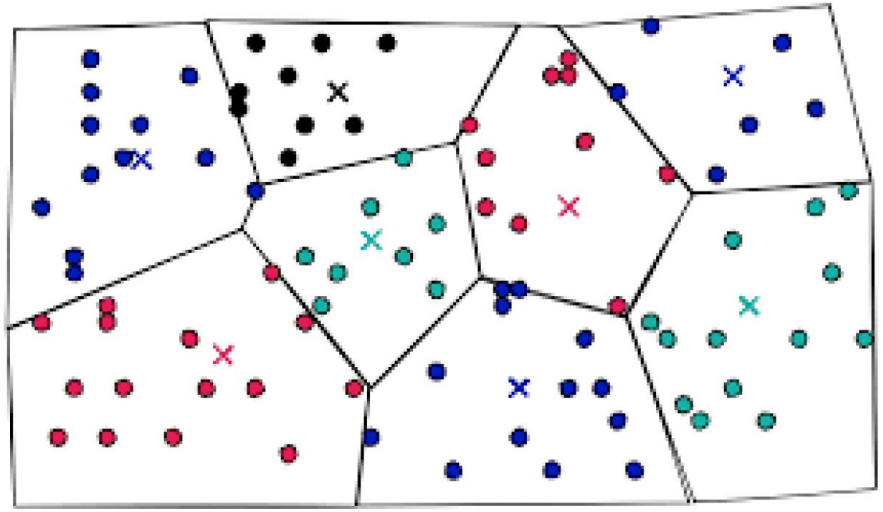
$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

$$p(z \mid x) = \frac{\exp f(x, z)}{\sum_{z'} \exp f(x, z')},$$

$$f(x, z) = \texttt{Embed}_{\texttt{input}}(x)^\top \texttt{Embed}_{\texttt{doc}}(z)$$

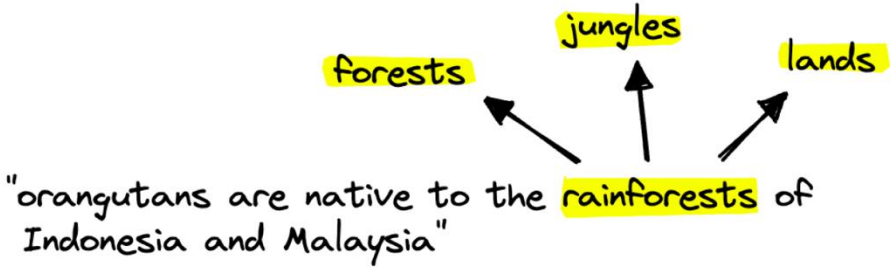
# Approximate nearest neighbor search

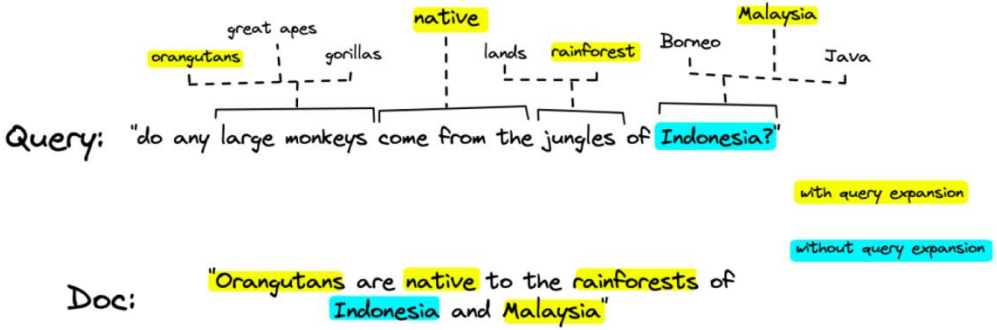
- K-NN algorithms used to reduce search time
- Search accuracy is affected
- Faiss is a popular indexing library supported by many vector db
- Distance metrics include:
  - L2, cosine similarity
- Several indexing algorithms – an active field of research
  - IVF
  - IVF_PQ
  - HNSW
  - Disk ANN



https://www.pinecone.io/learn/series/faiss/product-quantization/

# Retrieval by query expansion



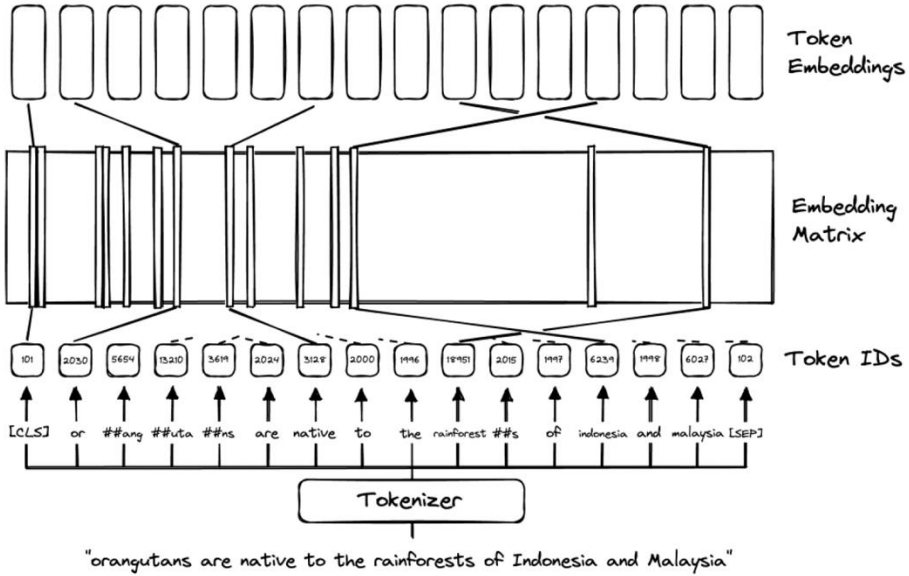

https://www.pinecone.io/learn/splade/

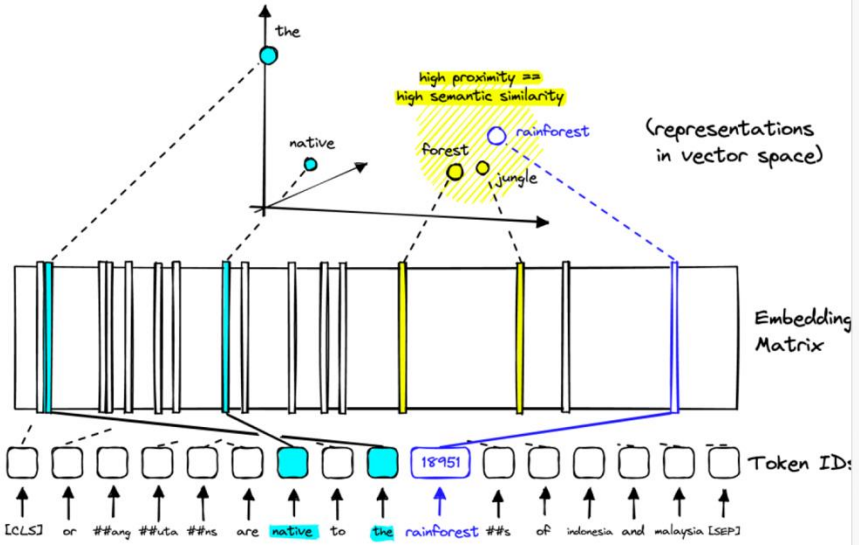
| Document | SPLADE BOW rep | New terms (expansion) | Dropped terms (compression) |
|---|---|---|---|
| Binary (or base-2) a numeric system that only uses two digits—0 and 1. Computers operate in binary, meaning they store data and perform calculations using only zeros and ones. | (binary, 2.96), (base, 1.83), (computer, 1.57), (2, 1.42), (decimal, 1.29), (computers, 1.29), (system, 1.25), (digit, 1.24), (nu, 1.17), (0, 0.76), (zero, 0.74), (., 0.56), (the, 0.53), (two, 0.49), (harmony, 0.48), (operate, 0.4), (computing, 0.31), (digits, 0.24), (neon, 0.19), (pi, 0.17), (##meric, 0.14), (pc, 0.13), (-, 0.06) | harmony, computing, digit, :, the, pc, decimal, computer, pi, neon | —, perform, ##s, or, meaning, data, ), (, store, in, 1, „ 'only', calculations, and, ., that, they, using, a, uses, ones |
| Option Types: Calls & Puts In the special language of options, contracts fall into two categories - Calls and Puts. A Call represents the right of the holder to buy stock. | (option, 2.16), (call, 1.92), (type, 1.75), (contract, 1.58), (put, 1.33), (language, 1.24), (holder, 1.18), (right, 0.95), (putting, 0.89), (deal, 0.83), (puts, 0.71), (options, 0.66), (special, 0.61), (buy, 0.6), (stock, 0.53), (contracts, 0.36), (calls, 0.34), (stuff, 0.25), (called, 0.14), (two, 0.05), (specialized, 0.04), (types, 0.02) | type, contract, put, deal, called, specialized, putting, stuff | into, in, of, :, the, -, and, ., a, categories, &, fall, to, „ represents |

https://europe.naverlabs.com/blog/splade-a-sparse-b
encoder-bert-based-model-achieves-effective-and-
efficient-first-stage-ranking/

SPLADE v2: Sparse Lexical and Expansion Model for Information Retrieval, SIGIR 2021

# SPLADE model

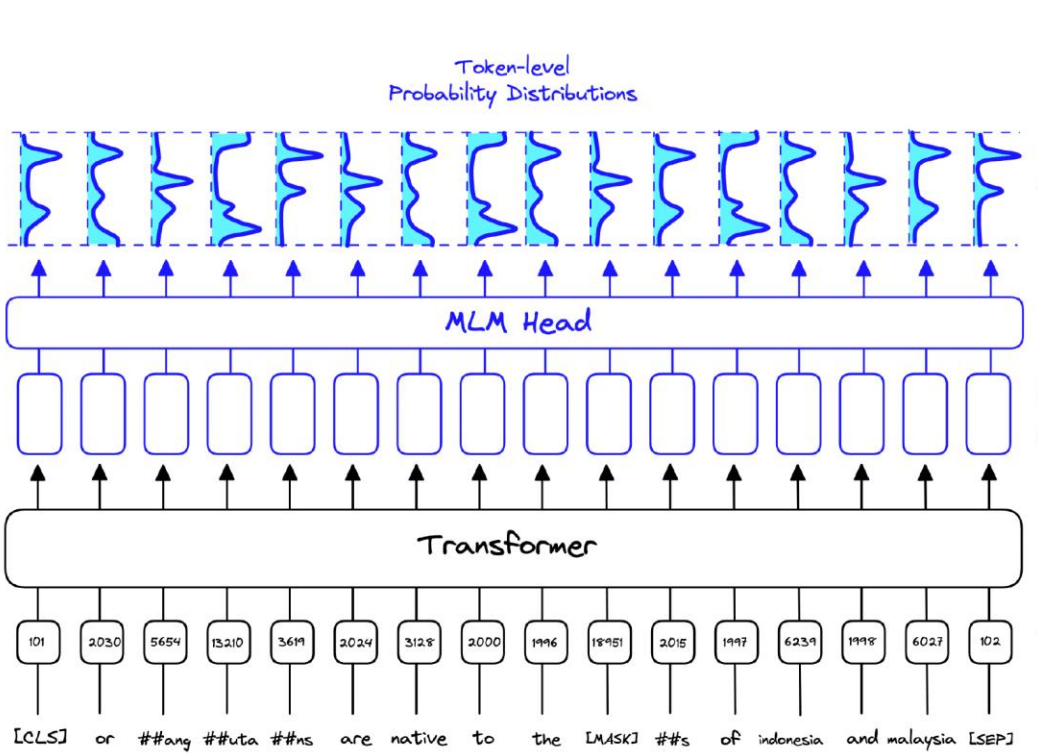


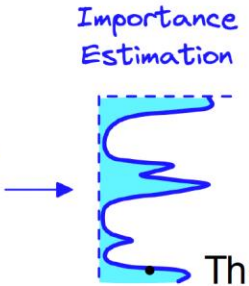




Uses the MLM head of transformer to predict the in-content term expansions

https://www.pinecone.io/learn/splade/

# SPLADE model




$$w_j = \sum_{i \in t} log(1 + ReLU(w_{ij}))$$

i is the input token, j is the vocabulary token

The number of non-zero values in SPLADE query and document vectors is typically greater than in traditional sparse vectors

- Sparse retrieval systems are not optimized for this.
- The distribution of non-zero values deviates from the traditional distribution expected by the sparse retrieval systems, again causing slowdowns.

The MLM head gives us a probability distribution for each token, whether or not they have been masked. These distributions are aggregated to give the importance estimation.

# SPLADE model

- Training data:
  - (query q, document+, document-)
- **Architecture:**
  - **Encoder**
    - BERT (or a BERT-like transformer)
    - Usually fine-tuned (not frozen)
  - **Projection / "MLM-style" head**
    - A linear layer with output size = vocabulary size
    - Same shape as an MLM head, but **not used for mask prediction**

For a text $x$ (query or document):

1. BERT outputs contextual embeddings:

$$h_1, h_2, \ldots, h_n$$

2. Each token projects to vocabulary logits:

$$z_{i,v} = h_i W_v$$

3. SPLADE aggregation:

$$w_v = \max_i \log(1 + \mathrm{ReLU}(z_{i,v}))$$

$$\mathrm{score}(q, d) = \sum_v w_v^{(q)} \cdot w_v^{(d)}$$

Contrastive loss

For a query $q$, positive doc $d^+$, negative doc $d^-$:

$$\mathcal{L}_{rank} = -\log \frac{\exp(\mathrm{score}(q, d^+))}{\exp(\mathrm{score}(q, d^+)) + \exp(\mathrm{score}(q, d^-))}$$

$$\mathcal{L}_{sparse} = \lambda \sum_v |w_v|$$

Training loss function

$$\mathcal{L} = \mathcal{L}_{rank} + \lambda \mathcal{L}_{sparse}$$

# How SPLADE differs from Pure Lexical Search

"neural retrieval models"

```
neural    → doc42 (tf=1)
retrieval→ doc42 (tf=1)
models    → doc42 (tf=1)
```

```
neural      → 1.3
retrieval   → 1.1
models      → 0.9
search      → 0.6
ranking     → 0.4
bert        → 0.2
...
```

Can query expansion be done without SPLADE?

# Hybrid search methods

- Combine lexical and semantic searches
- Reciprocal rank fusion a popular fusion approach

$$RRF(d) = \Sigma(r \in R)\ 1 / (k + r(d))$$

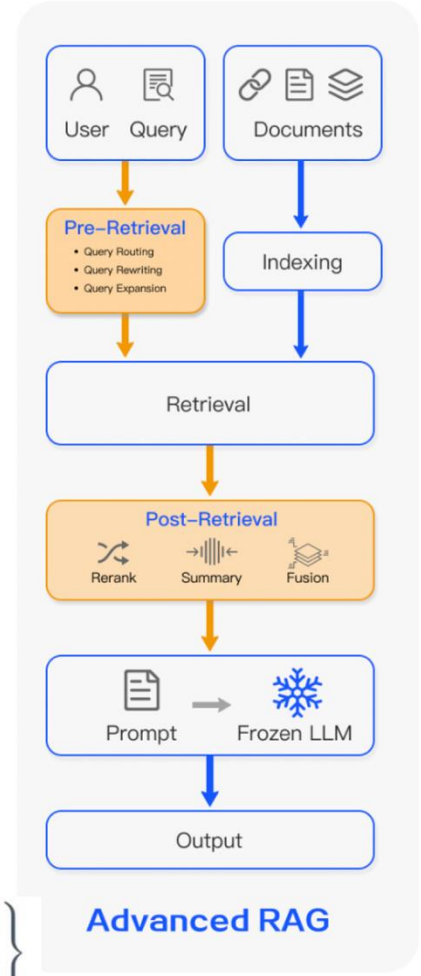
  - Where:

    - d is a document
    - R is the set of rankers (retrievers)
    - k is a constant (typically 60)
    - r(d) is the rank of document d in ranker r

# Advanced RAG Methods

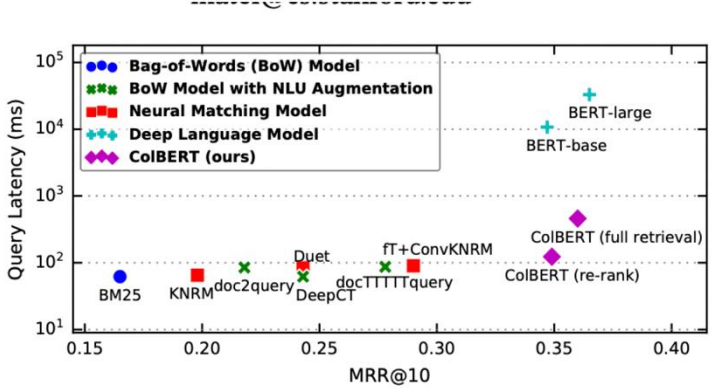
- Indexing
  - Prefiltering text, Decide what to tokenize
  - Adding metadata, Better chunking
- Retrieval
  - Fine tuning embedding (pairs of questions and answers are used to teach the retrieval engine).
    - E.g. BGE model on enterprise knowledge
  - Sparse, Dense, and Multivector models
    - E.g. COLBERT, SpladeV2
  - Dynamic embedding
    - E.g. Open AI's embeddings-ada-02, adapts to the context in which words are used
- Post-retrieval augmentation
  - Re-ranking
    - E.g. Langchain (document diversity, selection of documents)
  - Perplexity-based filtering (mutual information)

$$\mathrm{PPL}(X) = \exp\left\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\right\}$$



# Neural IR approaches

- Supervised approaches
  - Training data has pairs of documents and queries and ranked by relevance reflected in the loss function
  - Index time, project the documents using the learned representation
  - At search time, project query and find nearest matches
- Learn a representation of query and document such that the bes matching documents have a higher similarity
- Adapt transformer models like BERT to develop the embeddings
- Much more compute-intensive than unsupervised methods
- Many neural IR approaches can be used for re-ranking and search as well.



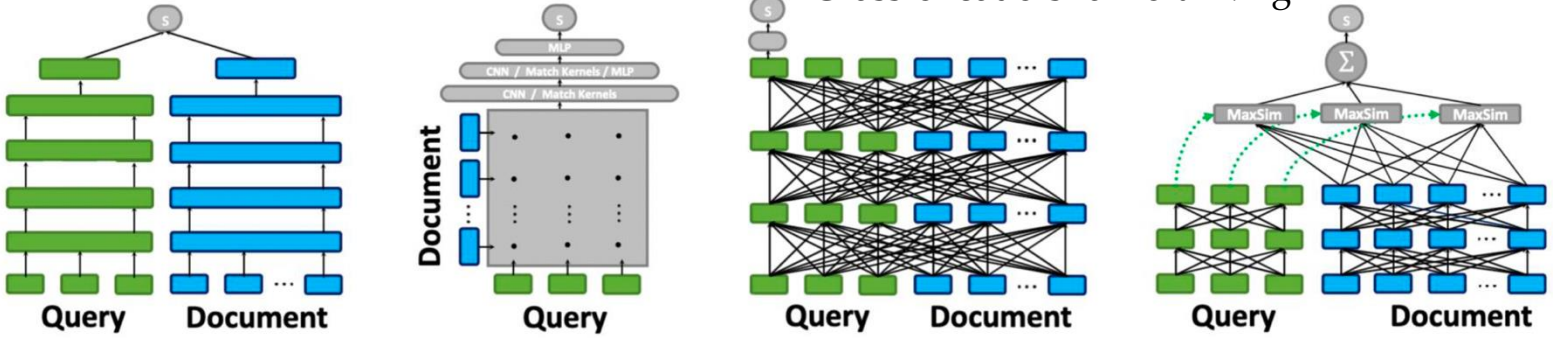

Contextualized Late Interaction over BERT, SIGIR 2020

# Neural IR approaches

The supervised approach uses IR data such as labeled query-document pairs, to learn a representation that is optimized for ranking and retrieval

Cross-encoders for reranking

Transformer layers




Query and document embeddings separately computed and compared at search time using cosine similarity

An interaction matrix that reflects the similarity between every pair of words across q and d fed to a classifier for relevance. The similarity matrix is fed to classifiers

Use a transformer to match and decode to a similarity value

COLBERT

Contextualized Late Interaction over BERT, SIGIR 2020

# COLBERT

- **ColBERT = late interaction dense retrieval**
- Instead of:
  - collapsing a document into **one vector** (dense retrievers), or
  - collapsing it into **one sparse bag of terms** (BM25 / SPLADE),
- ColBERT keeps **a vector per token** and lets queries and documents interact *at retrieval time*.
  - *Each query token gets to look for its best matching token in the document."*

Vector per token from BERT contextual embedding

For a query $q$ and document $d$:

1. For each query token $q_i$,
2. Compute similarity with **all** document tokens:

$$\text{sim}(q_i, d_j) = q_i \cdot d_j$$

3. Take the **maximum** similarity:

$$s_i = \max_j \text{sim}(q_i, d_j)$$

4. Sum over query tokens:

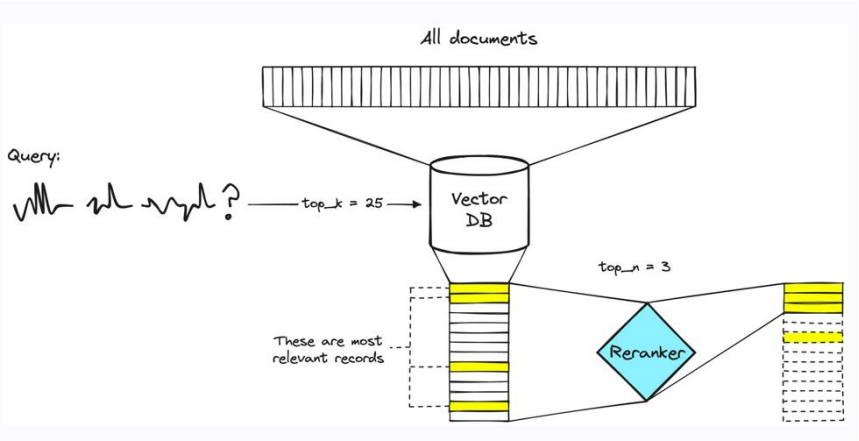
$$\text{score}(q, d) = \sum_i s_i$$

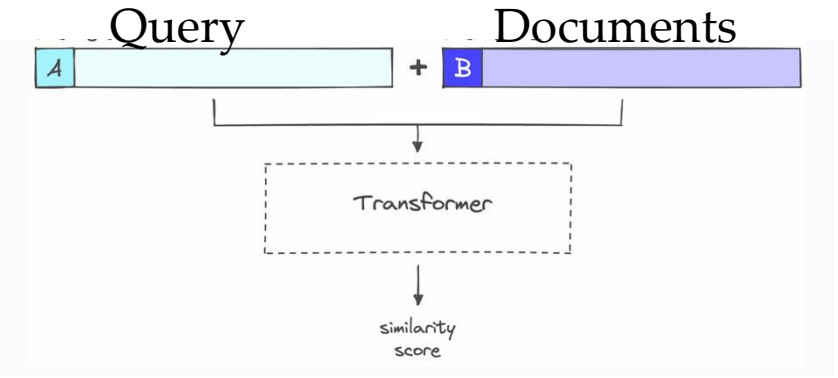
This is called **MaxSim**.

Trained similarly to SPLADE for retrieval-optimized embeddings

# Scalable retrieval pipeline

- **Retriever**
  - BM25 / SPLADE / DPR
  - Fast, coarse recall
  - Top 1k–10k docs
- **Refiner**
  - ColBERT or similar
  - Better semantic matching
  - Top 100–500 docs
- **Cross-encoder (reranker)**
  - Highest quality
  - Top 10–50 docs



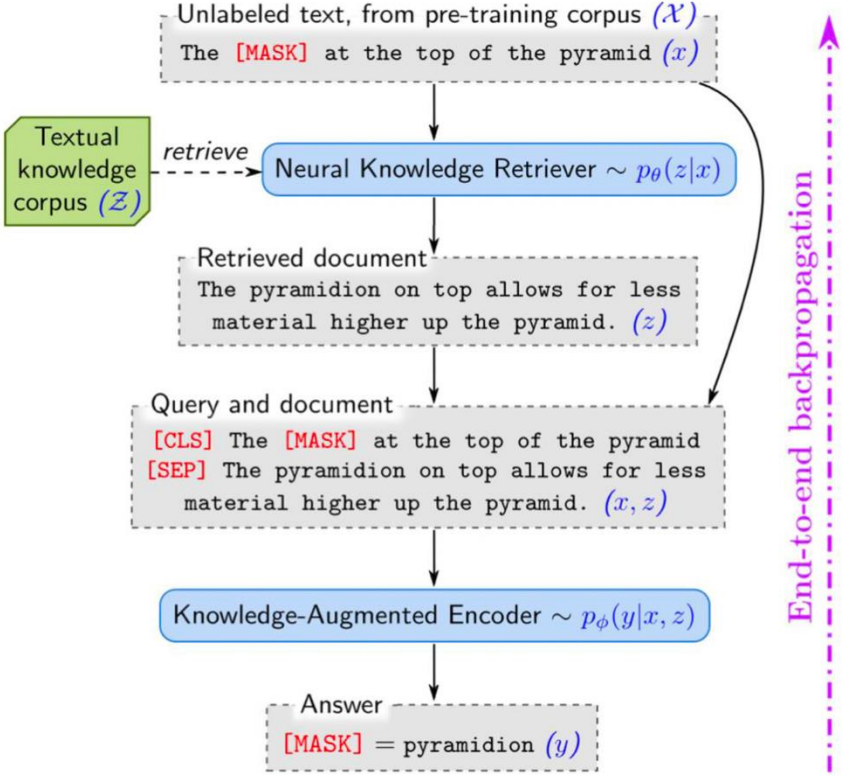



# RAG for pre-training and fine-tuning

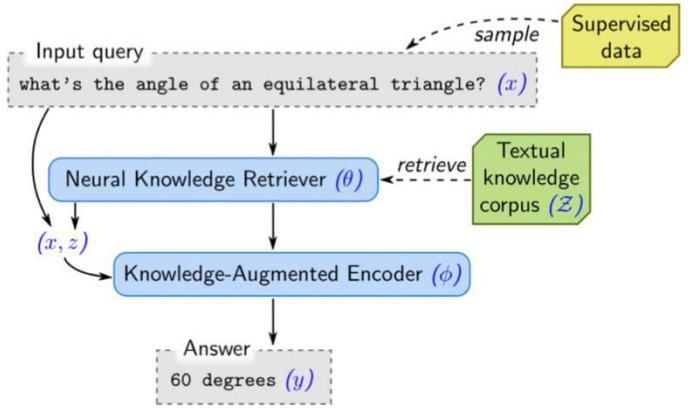
- RAG in pre-training or fine-tuning:
  - Use external knowledge as a non-parametric memory in addition to parametric seq2seq models.
- Benefits:
  - Results in smaller model sizes
  - Better explainability for model predictions
  - Better adaptability to new information without re-training

# REALM

## REALM: Retrieval-augmented language model pre-training




End-to-end backpropagation




Supervised fine-tuning.

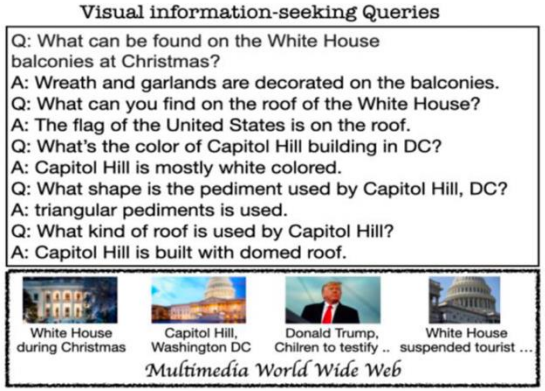
REALM: Retrieval-augmented language model pre-training, SIGIR 2020

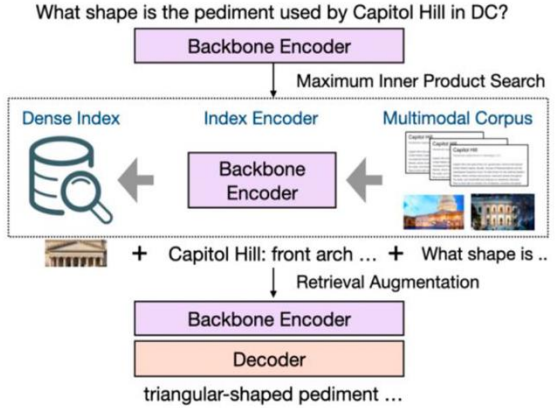
Language model pre-training algorithm
**Learned neural document retriever** using an unsupervised **fill-in-the-blank** training objective.

Perplexity score can be used as a loss function for training

# Multimodal RAG for pre-training or fine-tuning

- Uses image and text information to retrieve relevant documents
  - The knowledge to answer questions may lie within images
- Pre-training or fine-tuning with non-parametric multimodal memories
  - Generates more visually grounded output



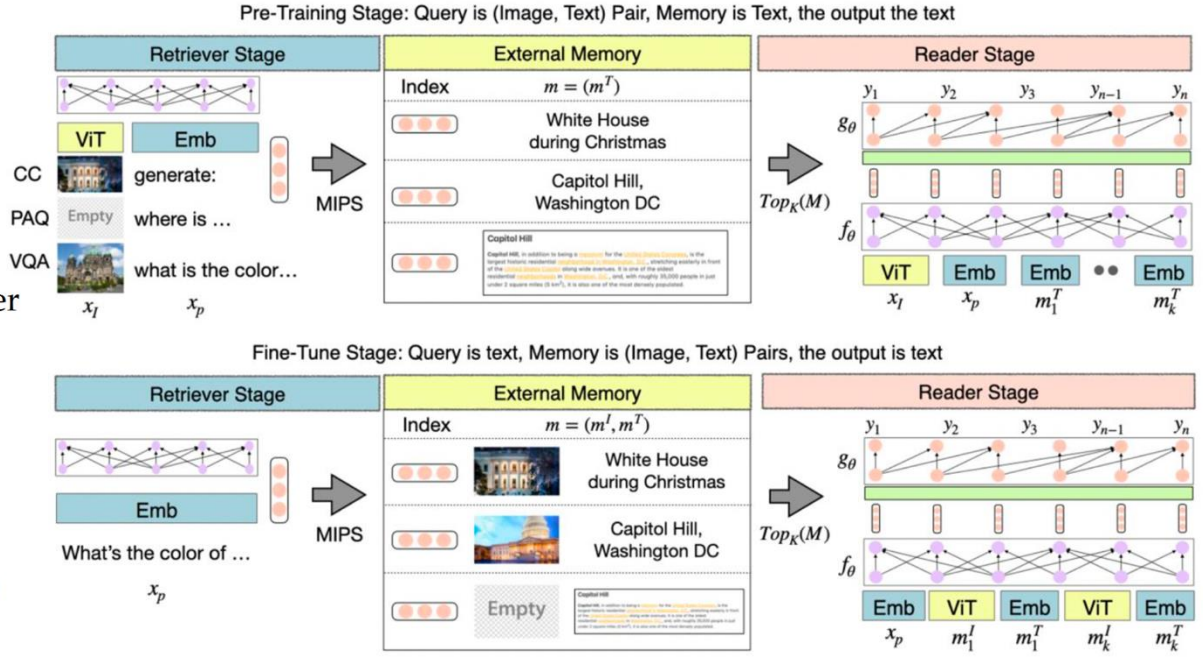




MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text, EMNLP 2022

# MURAG – Multimodal RAG

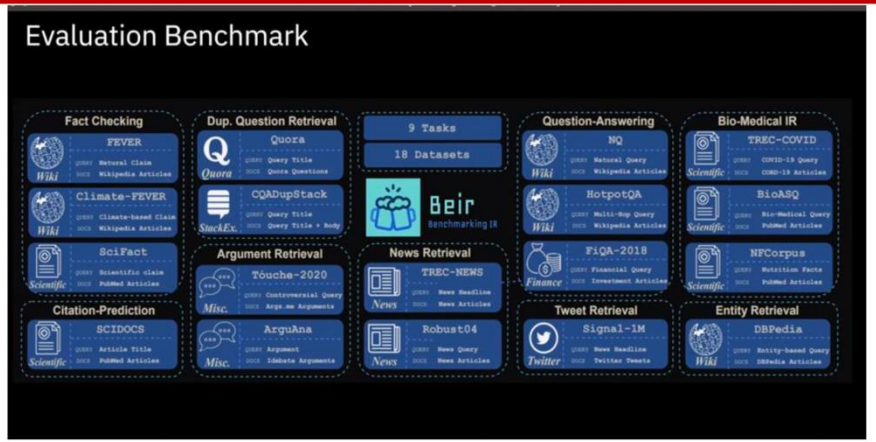
https://arxiv.org/abs/2210.02928

- During pre-training: Train an encoder to produce relevant captions from a textual memory bank.
- During RAG: Use the same encoder for encoding image-text pairs in a multimodal memory bank.
- Input is a textual prompt, the retrieved documents are multimodal, which are then fed to the generator to generate text




MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text, EMNLP 2022

# Evaluation Benchmarks




Bier Benchmark

- 18 datasets, 9 tasks

The MTEB benchmark (Multi-Task Evaluation Benchmark)

- 56 datasets distributed among 8 distinct tasks.
- 2000+ outcomes on the leaderboard.

Evaluation metrics

- NDCG (normalized discounted cumulative gain),
- MRR(mean reciprocal rank)
  - It focuses on the position of the first relevant item in the ranked list
  - Weighting is stricter
- MAP
  - Works for binary relevance
- Precision, Recall, Accuracy
  - Recall@K is another popular metric
- NDCG allows for fair comparisons between lists of varying lengths and relevance distribution
- NDCG considers the entire ranking order and assigns higher weights to relevant items at higher positions
- Both binary and numerical scores are handled

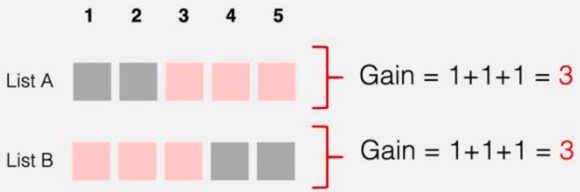
$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}.$$

where $\mathrm{rank}_i$ refers to the rank position of the *first* relevant document for the *i*-th query.

## NDCG

$$\mathrm{DCG@K} = \sum_{k=1}^{K}\frac{rel_i}{\log_2(i+1)}$$

$$\mathrm{NDCG@K} = \frac{DCG@K}{IDCG@K}$$



# RAG – newer work

- Self-Reflection Loops:
  - Introduce feedback loops where the LLM evaluates its own responses and uses the feedback to refine future retrievals and generations.
  - This process, also known as Self-RAG, helps to improve the accuracy and coherence of the generated text. iterative prompting and correction
- Modular RAG:
  - Aims to build a highly customizable and versatile RAG system.
  - It incorporates various modules for different tasks, such as search, retrieval, and generation, allowing for flexibility and adaptation to different use cases.
- RAG Fusion:
  - Combines the power of RAG with Reciprocal Rank Fusion (RRF).
  - It generates multiple queries, re-ranks the retrieved documents using RRF, and fuses them to create a more comprehensive and relevant knowledge base for the LLM.
- Agentic RAG:
  - Leverages the capabilities of LLMs as agents, granting them access to various tools and information sources. LLM-based query routing to relevant knowledge sources
  - The retrieval component becomes agentic, enabling the LLM to actively query, gather, and process information to generate more accurate and comprehensive responses.

GraphRAG also emerging

# Factors affecting RAG performance

- Extraction accuracy of the ingestion pipeline (structural shredders, chunkers, summarizers)
- How it got represented (a function of embedding choices and semantic expansions)
  - What representation capture semantics of documents?
  - What embedding spaces bring queries and documents closer?
- How it was searched (lexical, semantic, fusion, re-ranking)
- How it was aggregated and rolled up for the specific use case (chunk, page, document level roll-up)
- How the query was analyzed (semantic enrichment of the query)
  - Query rewriting
- Prompt engineering to prime the LLM
- Whether RAG is used only during inference or at pre-training or fine-tuning as well

# Using RAG for healthcare applications

- *Diagnostic Assistance*
  - Similar patient cases
  - Retrieve relevant clinical guidelines
- Summarization
  - EMR, discharge notes
- Medical QA
- Patient education chatbots
- Clinical trial matching
  - *Aligning patient profiles with the eligibility criteria extracted from trial registries*
  - *Retriever identifies protocols, generator generates inclusion and exclusion criteria.*
- Biomedical literature synthesis
  - For research purposes

# RAG applications in healthcare

**Table 1.** Taxonomy of RAG applications in healthcare.

| No. | Category | Description |
|---|---|---|
| 1 | General Clinical Applications of RAG | Broad applications of RAG for tasks like clinical summarization, decision support, and guidelines. |
| 2 | RAG Chatbots for Patient Interaction | Conversational agents enhanced by retrieval for providing personalized medical advice. |
| 3 | Specialty-Focused RAG Models | RAG frameworks tailored for domains such as cardiology, nephrology, or oncology using specialty-specific knowledge bases. |
| 4 | RAG for Signal and Time-Series Tasks | Integration of RAG with biosignals like ECG, EEG, or wearable data for diagnostic interpretation. |
| 5 | Graph-Based and Ontology-Aware RAG Frameworks | Use of structured clinical ontologies or knowledge graphs for enhanced retrieval and explainability. |
| 6 | RAG with Blockchain and Secure Architectures | Incorporation of privacy-preserving, decentralized data retrieval using blockchain-enhanced architectures. |
| 7 | Radiology-Specific Retrieval-Augmented QA | RAG systems designed for image-report alignment, report generation, and visual question answering in radiology. |

*medical reasoning, discharge planning, and infectious disease support*

*patient self-management, pre-consult triage, mental health screening, telemedicine*

*specialty decision support, patient-specific education, and knowledge reinforcement*

*using time-series data enhance diagnostic accuracy by integrating structured physiological signals into prompts*

*keyword, vector, and graph retrieval with a validated knowledge graph using topic-based chunking and semantic matching*

*Secure and auditable retrieval in sensitive domains, e,g. pediatrics*

*Second opinions, structured reporting, and continuous medical education in complex cases*

Retrieval-Augmented Generation (RAG) in Healthcare: A Comprehensive Review

# RAG Benchmarks in healthcare

- *MedQA (USMLE):*
  - *multiple-choice question dataset derived from medical licensing exams, focused on clinical knowledge assessment*
- *PubMedQA:*
  - *biomedical abstracts paired with yes/no/maybe questions, requiring grounded reasoning and evidence-based answers*
- *MIMIC-IV:*
  - *de-identified EHR dataset supporting tasks such as summarization, question answering, and document retrieval*
- *MedDialog:*
  - *multilingual dataset of doctor–patient conversations, suitable for training and evaluating medical dialogue systems*

# RAG challenges in healthcare

- Effective use of clinical knowledge graphs in RAG
- Static versus dynamic RAG
  - updating retrieval indexes for evolving knowledge
- Multimodal integration still not prevalent
- Handling privacy issues
- Domain-specific evaluation metrics
- Clinical workflow integration
- Bias and fairness issues