I 202: Information Organization & Retrieval Fall 2025

Class 24: Ranking Criteria, Tokenization, LLMs, RAG

Today's Outline

Finish Search Ranking

LLMs

Tokenization

Dense Vectors

RAG

How Search Engines Work

Three main parts:

i. Gather the contents of all web pages (using a program called a **crawler** or **spider**)

ii. Organize the contents of the pages in a way that allows efficient retrieval (**indexing**)

iii. Take in a query, determine which pages match, and show the results (**ranking** and **display** of results)

Last Time

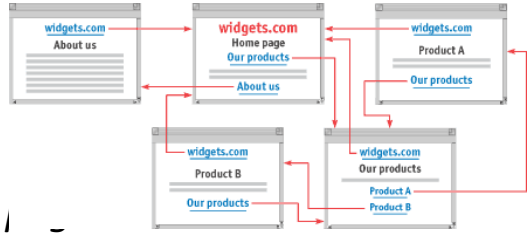
- Search Ranking Evaluation
- Boolean Search
- Statistical Search with Ranking
  - *Zipf's law*
  - *TF-IDF*
  - *BM-25*

Some Additional Ranking criteria

- For a given candidate result page, use:
  - Number of matching query words in the page
  - Proximity of matching words to one another
  - Location of terms within the page
  - Location of terms within tags e.g. <title>, <h1>, link text, body text
  - Anchor text on pages pointing to this one
  - Frequency of terms on the page and in general
  - Link analysis of which pages point to this one
  - (Sometimes) Click-through analysis: how often the page is clicked on
  - How “fresh” is the page
- Complex formulae combine these together.
- Combining these via machine learning started in the 2000’s

Slide adapted from Lew & Davis

Measuring the Importance of Linking



PageRank Algorithm

- *Idea: important pages are pointed to by other important pages*
- *Method:*
  - Each link from one page to another is counted as a "vote" for the destination page
  - But the importance of the starting page also influences the importance of the destination page.
  - And those pages scores, in turn, depend on those linking to them.

Image and explanation from http://www.economist.com/science/tq/displayStory.cfm?story_id=3172188

Measuring the Importance of Linking

PageRank

- Example: each page starts with 100 points.
- Each page's score is recalculated by adding up the score from each incoming link.
  - This is the score of the linking page divided by the number of outgoing links it has.
  - Example: the page in green has 2 outgoing links and so its "points" are shared evenly by the 2 pages it links to.
- Keep repeating the updates until no more changes.

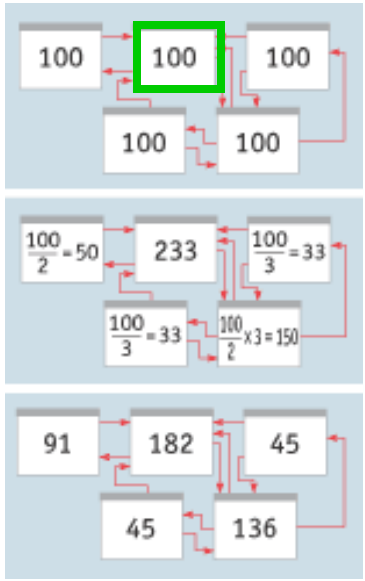


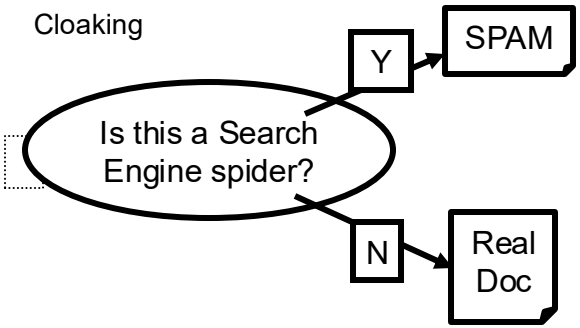
Image and explanation from http://www.economist.com/science/tq/displayStory.cfm?story_id=3172188

Manipulating Ranking

- Motives
  - *Commercial, political*
  - *Promotion funded by advertising budget*
- Operators
  - *Search Engine Optimizers*
  - *Web masters*
  - *Hosting services*
- Forum and Websites
  - *Web master world*
  - *Searchengineland.com*

A FEW SPAM TECHNOLOGIES

- **Cloaking**
  - *Serve fake content to search engine robot*
  - *DNS cloaking: Switch IP address. Impersonate*
- **Doorway pages**
  - *Pages optimized for a single keyword that re-direct to the real target page*
- **Keyword Spam**
  - *Misleading meta-keywords, excessive repetition of a term, fake "anchor text"*
  - *Hidden text with colors, CSS tricks, etc.*
- **Link spamming**
  - *Mutual admiration societies, hidden links, awards*
  - *Domain flooding: numerous domains that point or re-direct to a target page*
- **Robots**
  - *Fake click stream*
  - *Fake query stream*
  - *Millions of submissions via Add-Url*



Meta-Keywords =
"… London hotels, hotel, holiday inn, hilton, discount, booking, reservation, sex, mp3, britney spears, viagra, …"

Slide adapted from Manning, Raghavan, & Schuetze

PAID RANKING

Pay-for-inclusion

- Deeper and more frequent indexing
- Sites are not distinguished in results display

Paid placement

- Keyword bidding for targeted ads

Slide adapted from Lew & Davis

Summary

- Document ranking is complex; we've only scratched the surface and talked about classic approaches
- Machine learning using click data has become dominant on the web (pre-LLM)
- That data isn't available however for other search (on your desktop, in your organization, etc)

Transformer-based Language Models

Language Modeling

- Goal: given a sequence of words, predict the next one
  - *Input: a fragment of a text (a prompt)*
  - *Output: some more text*
- When trained at scale with a special architecture (the transformer model with many layers), becomes very good at generating text.
- In a sense, a large language model is a compression of a massive amount of text input.

Language Modeling Training in Action

TheUpshot

Watch an A.I. Learn to Write by Reading Nothing but <&$"<-]{,^\&

By Aatish Bhatia April 27, 2023

What this neural network actually generates is not letters but probabilities. (These probabilities are why you get a different answer each time you generate a new response.)

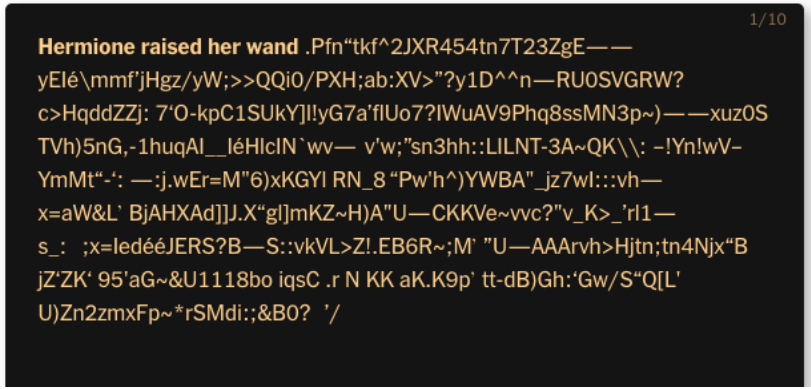
For example, when given the letters stai, it'll predict that the next letter is n, r or maybe d, with probabilities that depend on how often it has encountered each word in its training.

https://www.nytimes.com/interactive/2023/04/26/upshot/gpt-from-scratch.html

Language Modeling Training in Action

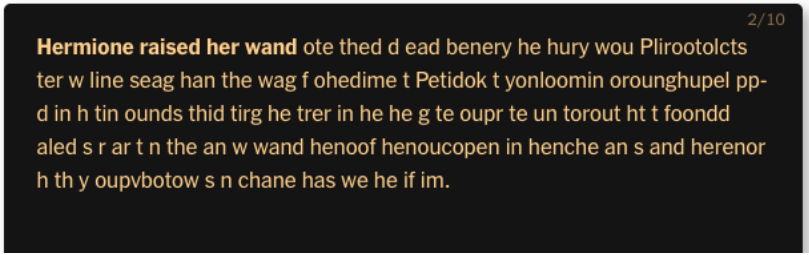
Before training: Gibberish

At the outset, BabyGPT produces text like this:



After 250 rounds: English letters

After **250 rounds of training** — about 30 seconds of processing on a modern laptop — BabyGPT has learned its ABCs and is starting to babble:



(does this remind you of anything from lecture 1?)

https://www.nytimes.com/interactive/2023/04/26/upshot/gpt-from-scratch.html

Language Modeling Training in Action

After 500 rounds: Small words

Each training round, BabyGPT tries to improve its guesses by reducing this loss. After **500 rounds** — or about a minute on a laptop — it can spell a few small words:

2/10

Hermione raised her wand the swored cour buere gof not whe wliknotiling ter wall loweas las the was fachering the thark, saing. He coroune upellemly his the luers thid tha gelu the liknow he got quare thand to thad the food he alker. 'Leeme notera couady exon'.

'On see to thecre an ste dicely ove the would ounto stanly meeas Dum could upreare in inteletinfer."

"I know he ming frefare he fooe eood aras the samer the quied the reacreak the steing unlust leces. Weas come the saium at rall kneat eare the was rias ho lelazed the the God the refer gie mot wire lustere mac mach.

After 30,000 rounds: Full sentences

An hour into its training, BabyGPT is learning to speak in full sentences. That's not so bad, considering that just an hour ago, it didn't even know that words existed!

1/10

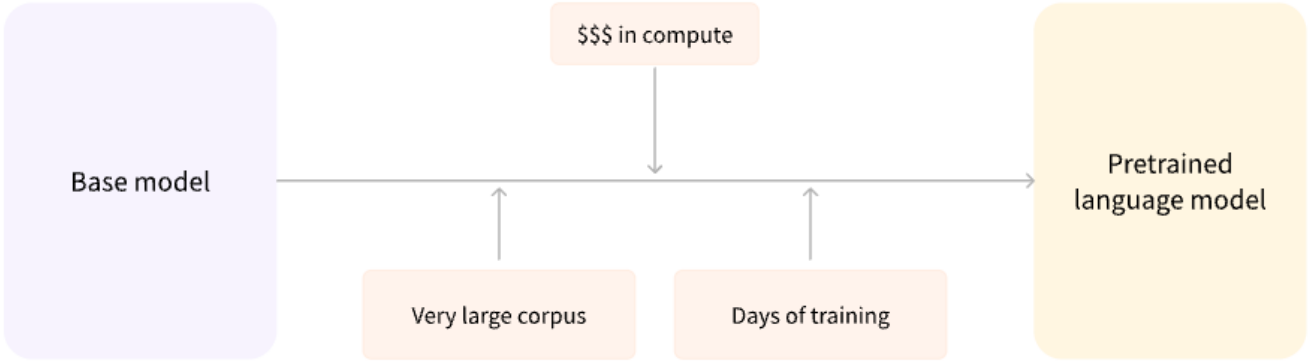
Hermione raised her wand. "Professor Dumbledore never mimmed Harry."

He looked back at the room, but they didn't seem pretend to blame Umbridge in the Ministry. He had taken a human homework, who was glad he had not been in a nightmare bad cloak. Her face looked over her closely past her and saw Harry crossed the grip, looked down at the wall. "Come off!" she said tentatively, with a crumpled note into his own hand on her book. "That's beating attacks how we've got detentions or not to realize how she did the Maps worse doesn't want.

https://www.nytimes.com/interactive/2023/04/26/upshot/gpt-from-scratch.html

LLM Pretraining

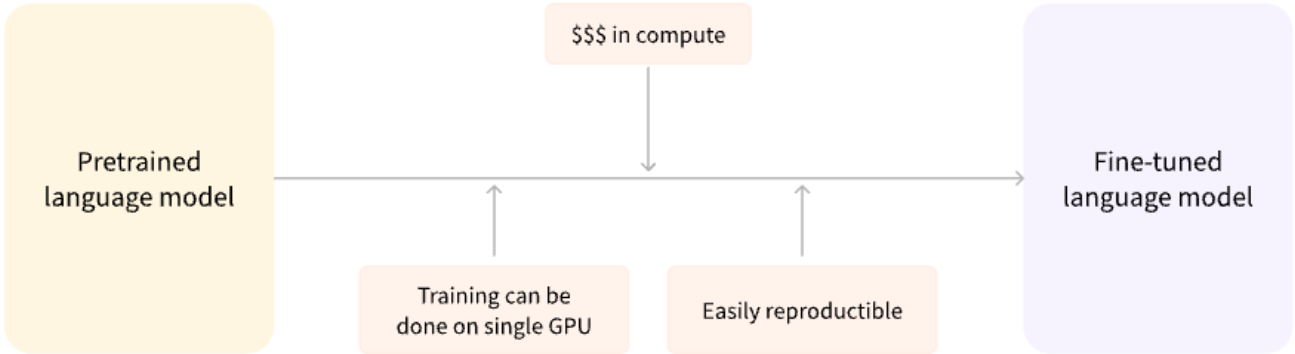
Pretraining is the act of training a model from scratch: the weights are randomly initialized, and the training starts without any prior knowledge.



https://huggingface.co/learn/llm-course/en/chapter1/4

LLM Fine Tuning

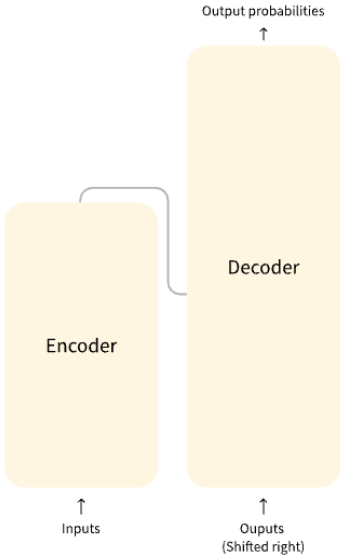
Fine-tuning is the training done after a model has been pretrained. First acquire a pretrained language model, then perform additional training with a dataset specific to a task.



https://huggingface.co/learn/llm-course/en/chapter1/4

Transformer Model

- The transformer model is primarily composed of two blocks:
- **Encoder (left)**: The encoder receives an input and builds a representation of it (its features).
- **Decoder (right)**: The decoder uses the encoder's representation (features) along with other inputs to generate a target sequence.



https://huggingface.co/learn/llm-course/en/chapter1/4

Types of Transformer Models

Language models generally fall into three architectural categories:

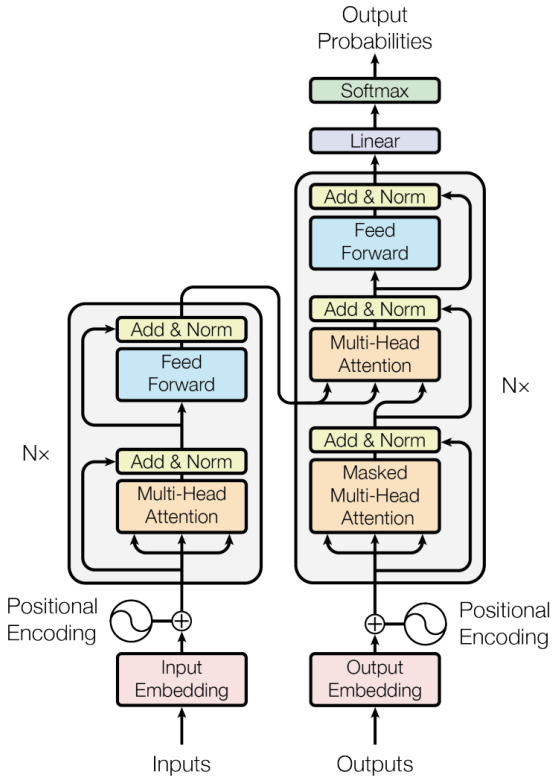
- **Encoder-only models** (like BERT):
  - Use a bidirectional approach to understand context from both directions.
  - Best for NLP tasks like classification, named entity recognition, and similarity comparison.
- **Decoder-only models** (like GPT, Llama):
  - Process text from left to right and are particularly good at text generation tasks.
  - Complete sentences, write essays, or generate code based on a prompt.
- **Encoder-decoder models** (like T5, BART):
  - Combine both approaches, using an encoder to understand the input and a decoder to generate output.
  - For sequence-to-sequence tasks like translation, summarization.

https://huggingface.co/learn/llm-course/en/chapter1/5

Full Transformer Architecture

BERT

Encoder



GPT

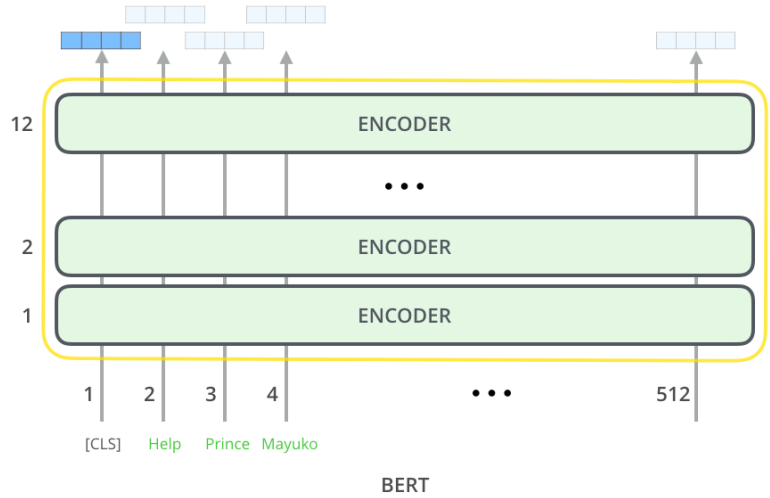
Decoder

https://huggingface.co/learn/llm-course/en/chapter1/5

BERT
(Bi-directional Encoder Representations from Transformers)

BERT takes a sequence of words as input which keep flowing up the stack. Each layer applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder.

In the last layer, each position outputs a vector

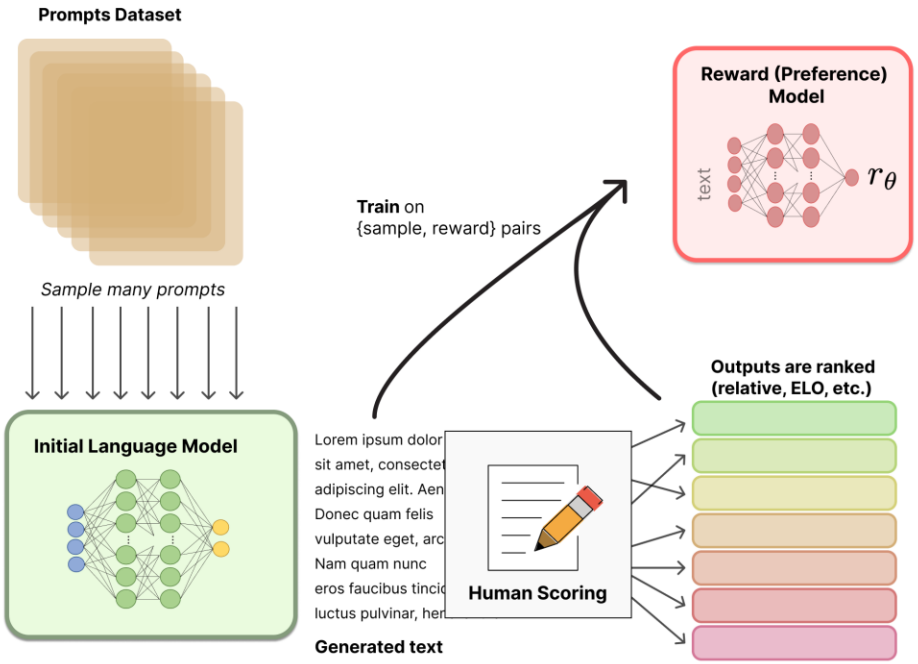


https://jalammar.github.io/illustrated-bert/

Reinforcement Learning with Human Feedback

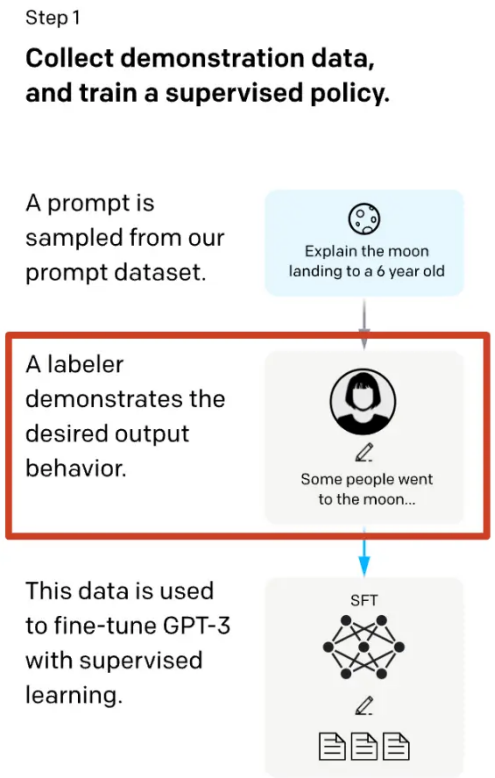
The raw output of an LLM isn't that great as a user interface.

RLHF is used to improve conversation, reduce bias, protect against dangerous queries, etc.



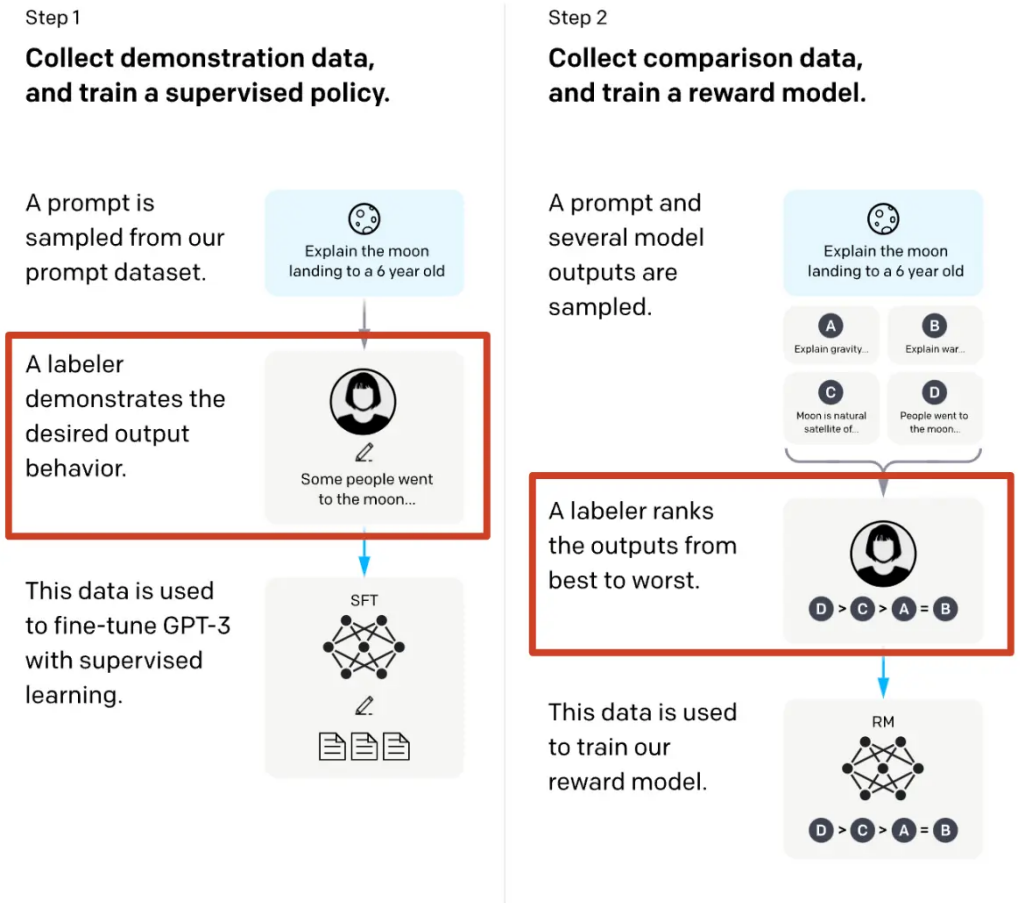
https://huggingface.co/blog/rlhf

Reinforcement Learning with Human Feedback (RLHF)



https://www.labellerr.com/blog/reinforcement-learning-from-human-feedback/

Reinforcement Learning with Human Feedback (RLHF)



https://www.labellerr.com/blog/reinforcement-learning-from-human-feedback/

Reinforcement Learning with Human Feedback (RLHF)

Step 1

Collect demonstration data, and train a supervised policy.

A prompt is sampled from our prompt dataset.

Explain the moon landing to a 6 year old

A labeler demonstrates the desired output behavior.

Some people went to the moon...

This data is used to fine-tune GPT-3 with supervised learning.

SFT

Step 2

Collect comparison data, and train a reward model.

A prompt and several model outputs are sampled.

Explain the moon landing to a 6 year old

A Explain gravity...

B Explain war...

C Moon is natural satellite of...

D People went to the moon...

A labeler ranks the outputs from best to worst.

D > C > A = B

This data is used to train our reward model.

RM

D > C > A = B

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

Write a story about frogs

The policy generates an output.

PPO

Once upon a time...

The reward model calculates a reward for the output.

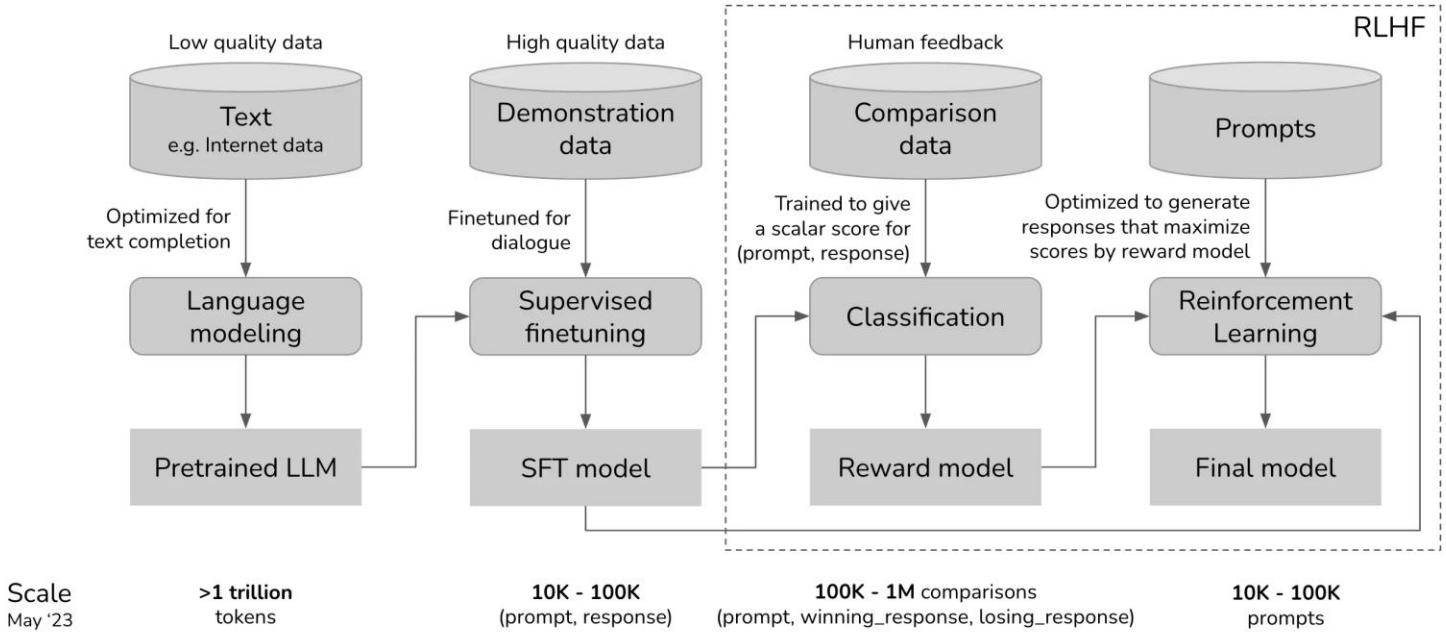
RM

The reward is used to update the policy using PPO.

r_k

https://www.labellerr.com/blog/reinforcement-learning-from-human-feedback/

Putting an LLM Into Production



SFT: supervised fine-tuning

https://www.labellerr.com/blog/reinforcement-learning-from-human-feedback/

TOKENIZATION

Reading from Chapter 2 of Introduction to Information Retrieval goes into detail

How many words in a sentence?

- "I do uh main- mainly business data processing"
  - *Fragments, filled pauses*
- "Seuss's cat in the hat is different from other cats!"
  - ***Lemma**: same stem, part of speech, rough word sense*
    - cat and cats = same lemma
  - ***Wordform**: the full inflected surface form*
    - cat and cats = different wordforms

Issues in Tokenization

- Can't just blindly remove punctuation:
  - m.p.h., Ph.D., AT&T, cap'n
  - prices ($45.55)
  - dates (01/02/06)
  - URLs (http://www.stanford.edu)
  - hashtags (#nlproc)
  - email addresses (someone@cs.colorado.edu)
- Clitic: a word that doesn't stand on its own
  - *"are" in we're, French "je" in j'ai, "le" in l'honneur*
- When should multiword expressions (MWE) be words?

How to do word tokenization in Chinese?

- 姚明进入总决赛 "Yao Ming reaches the finals"

- 3 words?
- 姚明 进入 总决赛
- YaoMing reaches finals

- 5 words?
- 姚 明 进入 总 决赛
- Yao Ming reaches overall finals

How to Handle All this?

- Some embeddings use character representations
- Today, LLMs often use subword encodings
  - *Byte Pair Encoding* *(Sennrich et al. 2016)*
  - *WordPiece* *(Schuster & Nakajima 2012)*

Byte-Pair Encoding

Step	Description
1. Initialization	Start with a base vocabulary containing all unique **characters** (or individual bytes, in Byte-level BPE) in the training corpus. Each word in the corpus is initially split into these individual characters.
2. Count Pairs	Count the frequency of all adjacent pairs of symbols (characters/subwords) across the entire corpus.
3. Merge Most Frequent	Identify the adjacent pair that occurs most frequently and **merge** them into a single, new subword token. This new token is added to the vocabulary.
4. Update and Repeat	Replace all occurrences of the merged pair in the corpus with the new subword token. Re-count the frequency of all new adjacent pairs and repeat the merging process (Steps 2 and 3) until the target vocabulary size is met.

Byte Pair Encoding

First, form the base vocabulary (all characters that occur in the training data)

word	frequency
hug	10
pug	5
pun	12
bun	4
hugs	5

Base vocab: b, g, h, n, p, s, u

Byte Pair Encoding

Next, count the frequency of each character pair in the data, and choose the one that occurs most frequently

character pair	frequency
un	16
h+ug	15
pu	12
p+ug	5
ug+s	5

...

https://www.cs.umd.edu/~miyyer/cs685/slides/tokenization.pdf

BYTE PAIR ENCODING

Next, choose the most common pair (ug) and then merge the characters together into one symbol. Add this new symbol to the vocabulary. Then, retokenize the data, and repeat.

word	frequency
h+*ug*	10
p+*ug*	5
p+u+n	12
b+u+n	4
h+*ug*+s	5

character pair	frequency
un	16
h+ug	15
pu	12
p+ug	5
ug+s	5

...

https://www.cs.umd.edu/~miyyer/cs685/slides/tokenization.pdf

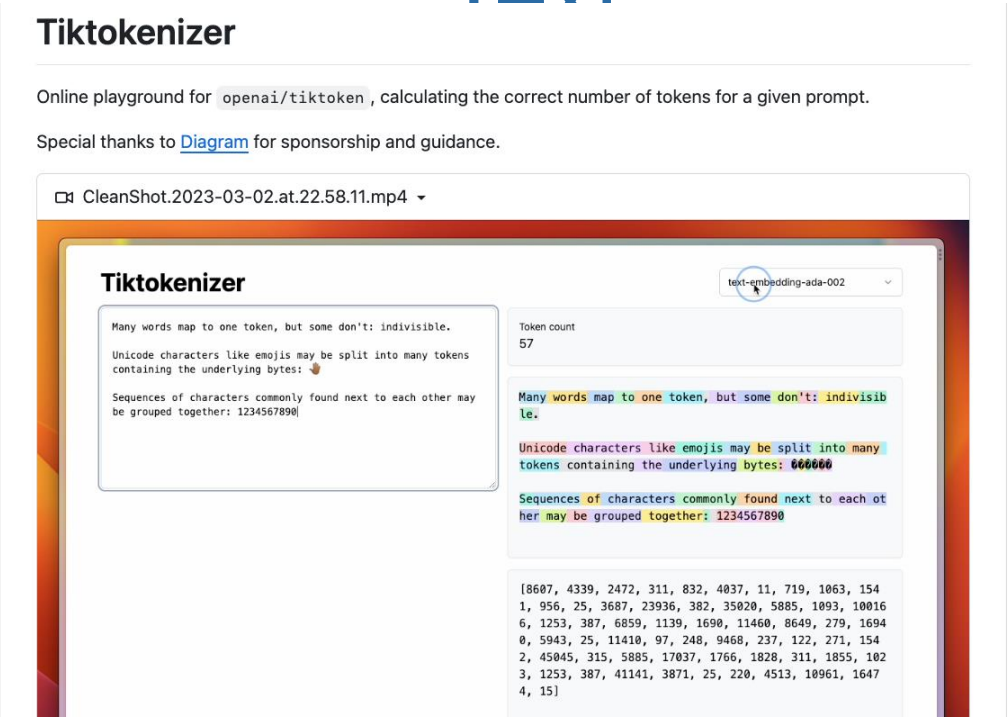
Byte Pair Encoding

Eventually, after a fixed number of merge steps, stop

word	frequency
hug	10
p+*ug*	5
p+*un*	12
b+*un*	4
hug + s	5

new vocab: b, g, h, n, p, s, u, *ug*, *un*, *hug*

Exercise: See How LLMs Tokenize Text

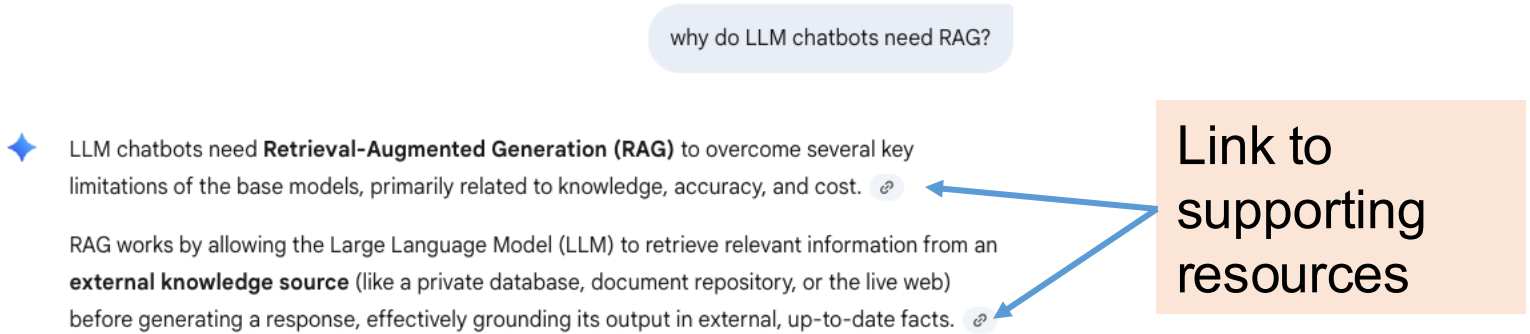


https://tiktokenizer.vercel.app/

Retrieval Augmented Generation

Why Do LLM Search Tools Need RAG?

- Reduce hallucinations
- Show proof for what is being claimed
- Give credit for where the info came from



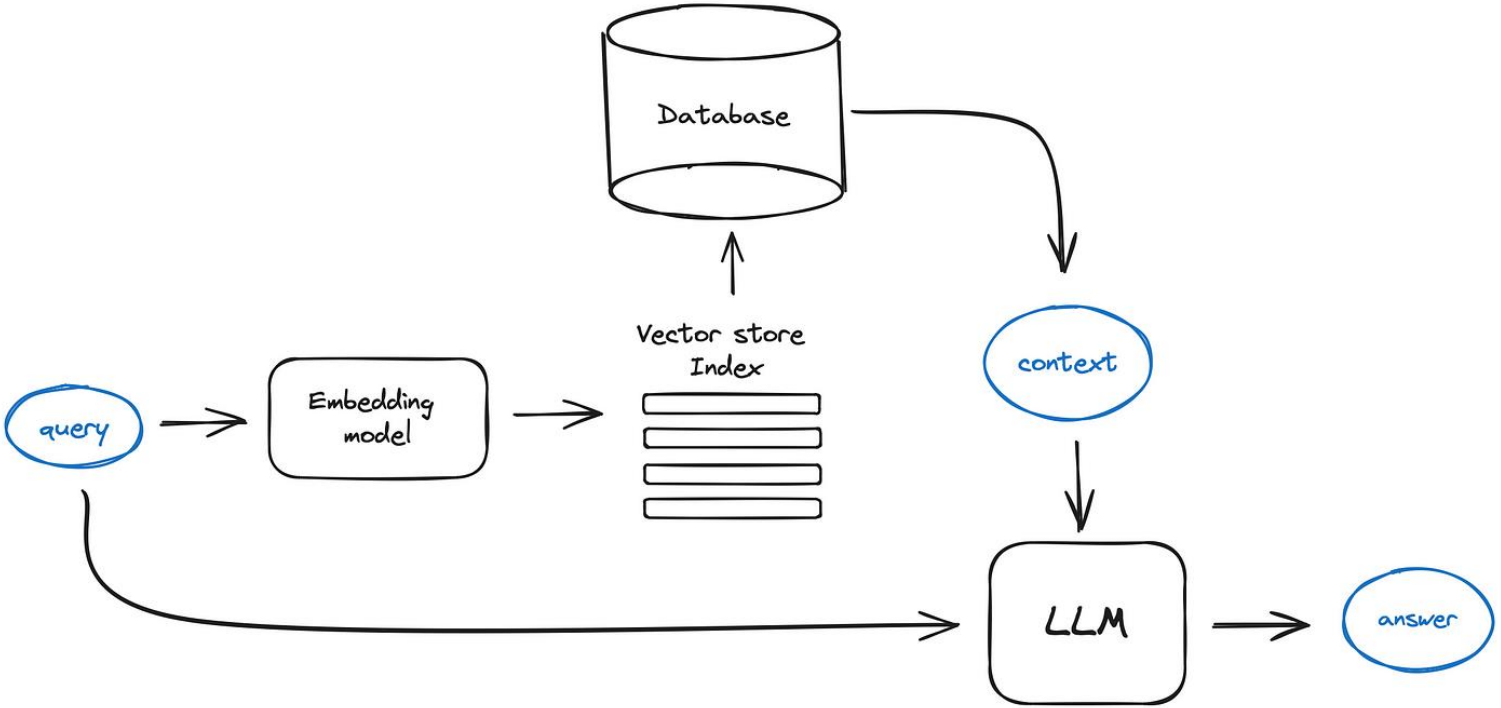
Standard AI Chatbots vs RAG Systems

Feature	Traditional AI Chatbots
Knowledge Source	Static training data only
Knowledge Recency	Degrades over time
Hallucination Risk	High in specialized domains
Context Awareness	Limited
Transparency	Limited explainability

Standard AI Chatbots vs RAG Systems

Feature	Traditional AI Chatbots	RAG Systems
Knowledge Source	Static training data only	Training data + Dynamic retrieval
Knowledge Recency	Degrades over time	Remains current with external sources
Hallucination Risk	High in specialized domains	Reduced with factual grounding
Context Awareness	Limited	Enhanced with domain-specific information
Transparency	Limited explainability	Source attribution capabilities

Basic RAG



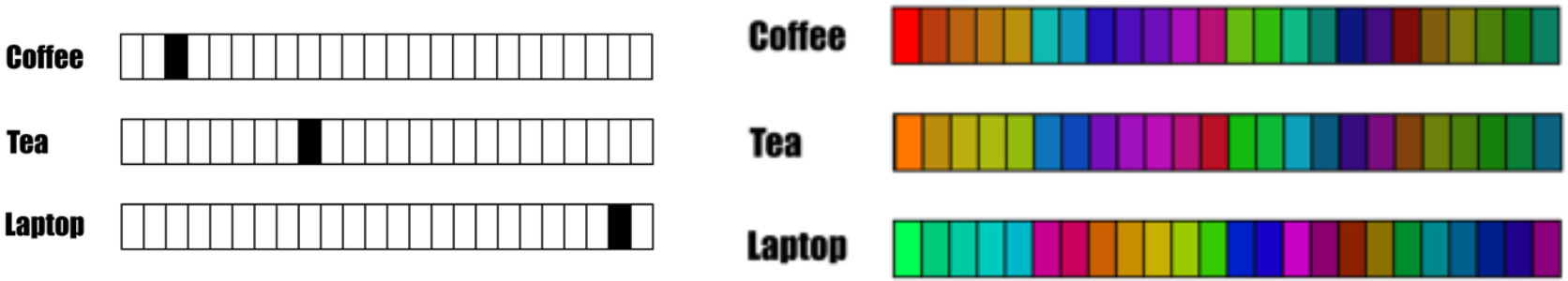
https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6

Let's Revisit Vector Embeddings

Word Vectors and Document Vectors

- In a previous lecture, we saw how **words** can be represented as vectors (embeddings)
  - *Vectors represent the contexts that words appear in*
  - *We computed similarity between words by using cosine similarity*
- **Documents** can also be represented as vectors
  - *One approach: Represent each document as a vector of its tf-idf term weights*
  - *Compare document similarity using cosine similarity*

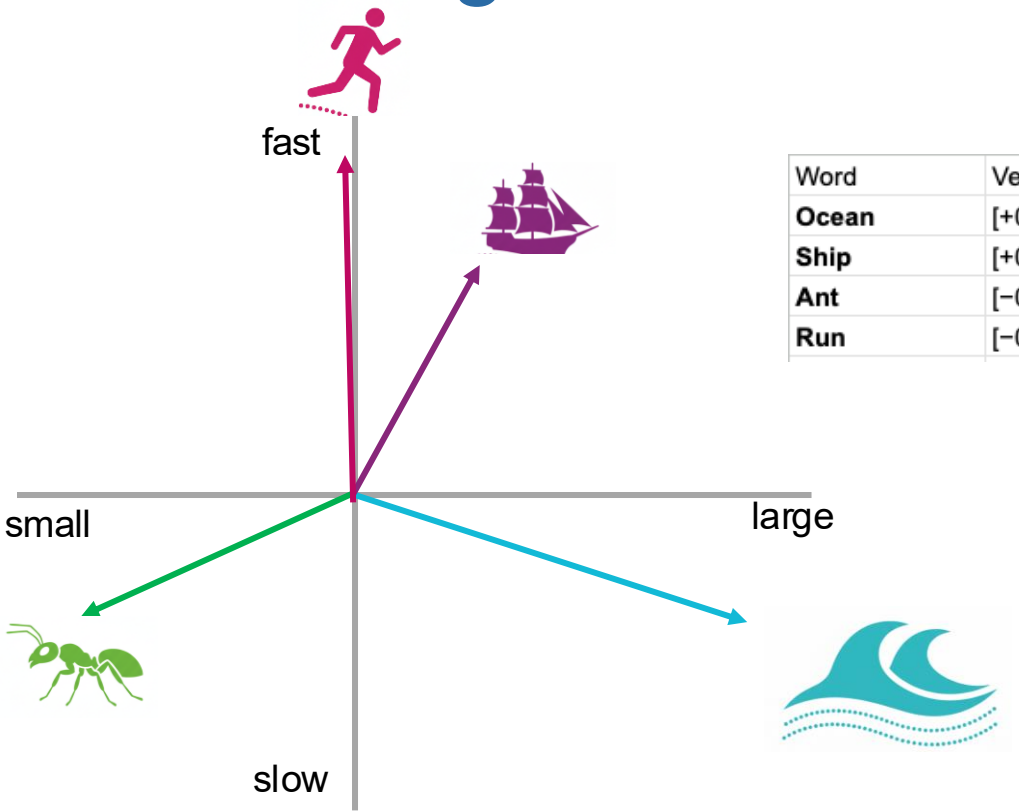
Computing Word Embeddings with Distributions Makes a Richer Representation



Think of the colors as showing complex nuance about which words have appeared in the same context

These are real numbers instead of frequency

Plotting Vectors on a 2D Graph

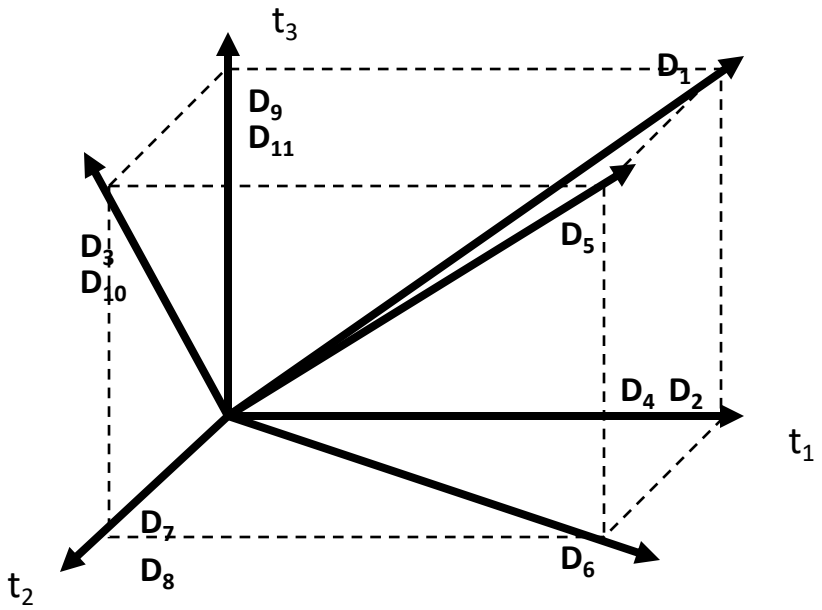


Word	Vector [X, Y]	Intuition
Ocean	[+0.9,−0.2]	Very Larget(high X), Very slow (low/negative Y).
Ship	[+0.5,+0.6]	Medium-large (medium X), Fast (high Y).
Ant	[−0.8,−0.4]	Very small (negative X), Slow (negative Y).
Run	[−0.1,+0.9]	Not big or small (near 0 X), Very fast (high Y).

A vector is not just a point on a graph; it's the **path** from the center to that point, defined by its components.

X-axis: a scale from Large (+X) to Slow (-X)
Y-axis: a scale from Fast (+Y) to Slow (-Y)
(0,0) is the “average” word

Documents in Vector Space

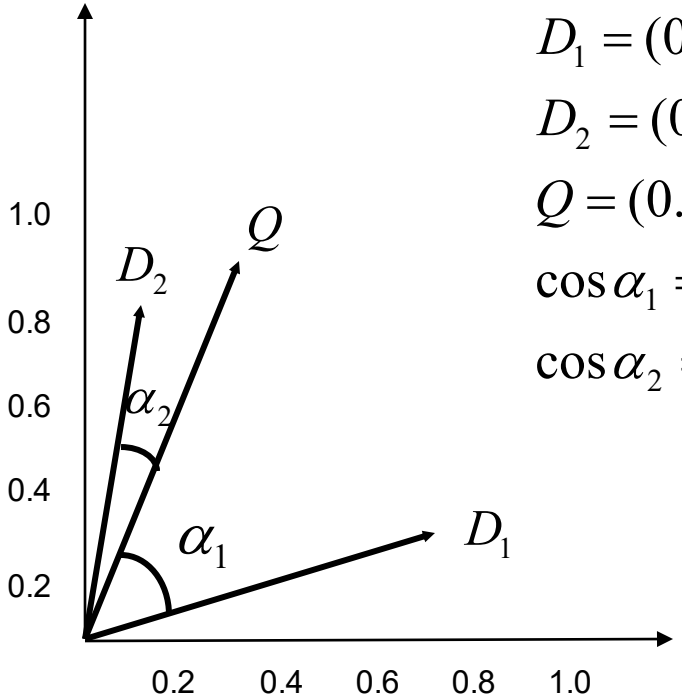


Assumption: Documents that are “close together” in space are similar in meaning.

Vector Space Model

- Documents are represented as vectors in term space
- Queries represented the same as documents
- Query and Document weights are based on length and direction of their vector
- A vector distance measure (similarity measure) between the query and documents is used to rank retrieved documents

Computing Similarity Scores to Determine Which Documents are Close to the Query



$$D_1 = (0.8, 0.3)$$

$$D_2 = (0.2, 0.7)$$

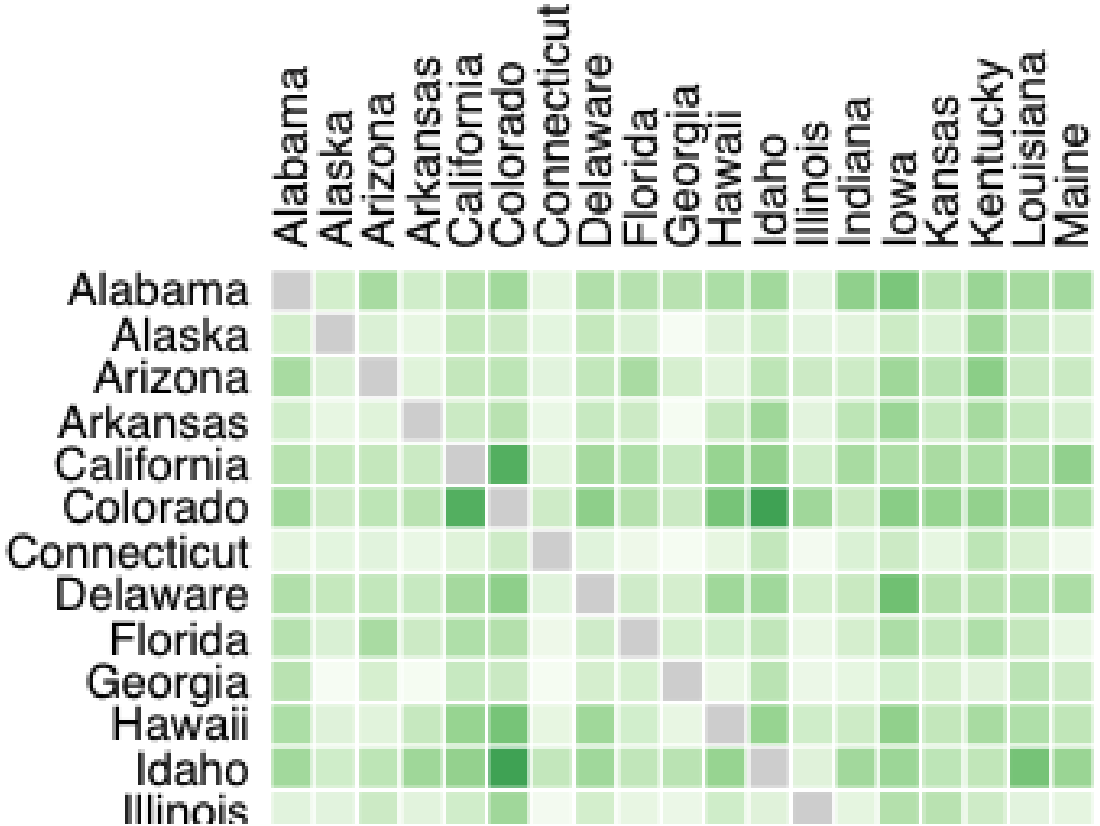
$$Q = (0.4, 0.8)$$

$$\cos\alpha_1 = 0.74$$

$$\cos\alpha_2 = 0.98$$

Can also use document vectors to computer Similarity

Here, for State-of-the-State Speeches
Via BM-25 vectors



https://observablehq.com/@kerryrodden/introduction-to-text-analysis-with-tf-idf

TF-IDF (SPARSE) VS DENSE VECTORS

- Tf-idf vectors: fast to compute, storage is smaller because vectors are sparse
- BUT they require exact matches between query terms and documents
- LLMs / word embeddings allow matching of concepts even if the words used differ between query and document
- These “dense vectors” are much more powerful for similarity search
- BUT this is much more complex to compute, store, and retrieve

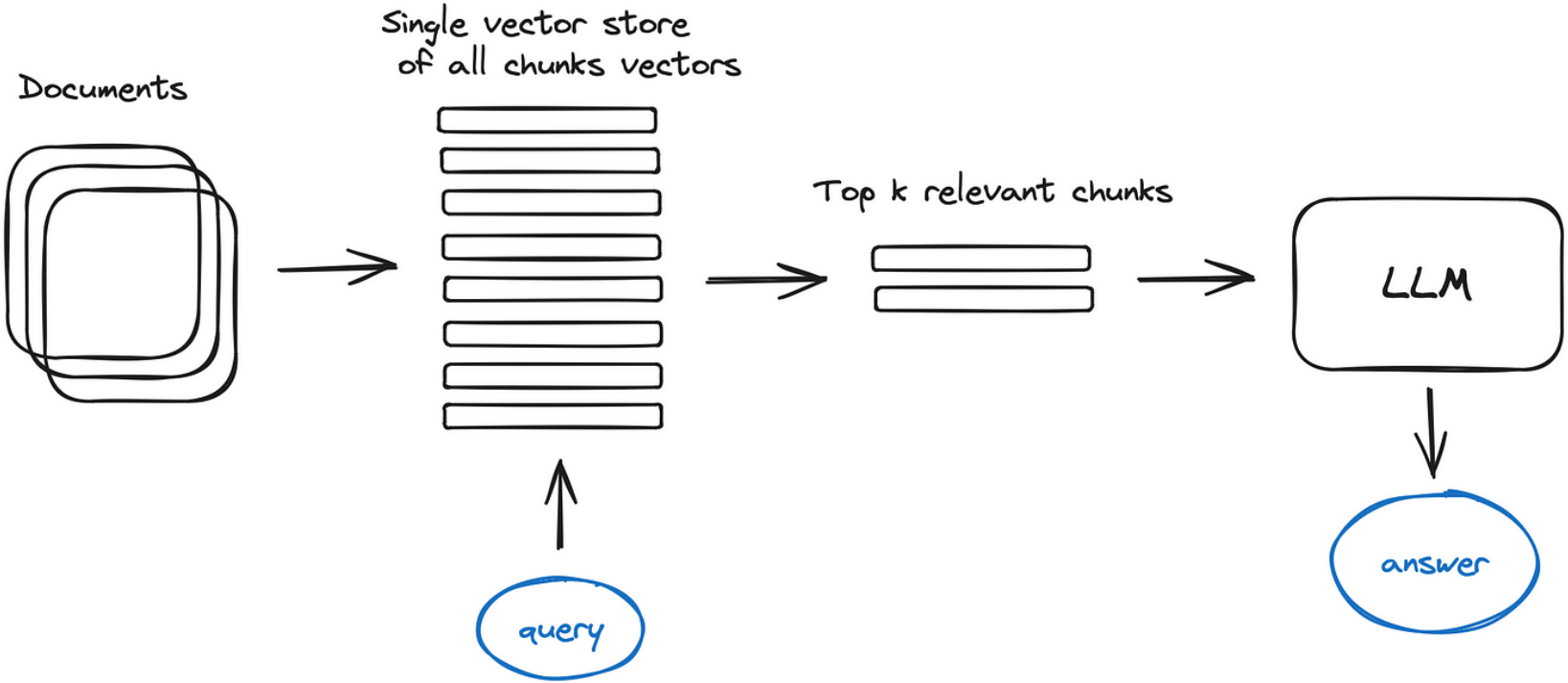
Dense Vectors for Representing Paragraphs

- Start with a word embedding model like BERT
- Then train new vector embeddings:
  - *Get pairs of paragraphs known to be similar/different*
  - *Use contrastive learning to train an encoder model so that:*
    - Positive pairs have high cosine similarity
    - Negative pairs have low cosine similarity
- Get positive/negative pairs from click logs, QA datasets, etc

Retrieving with Dense Vectors

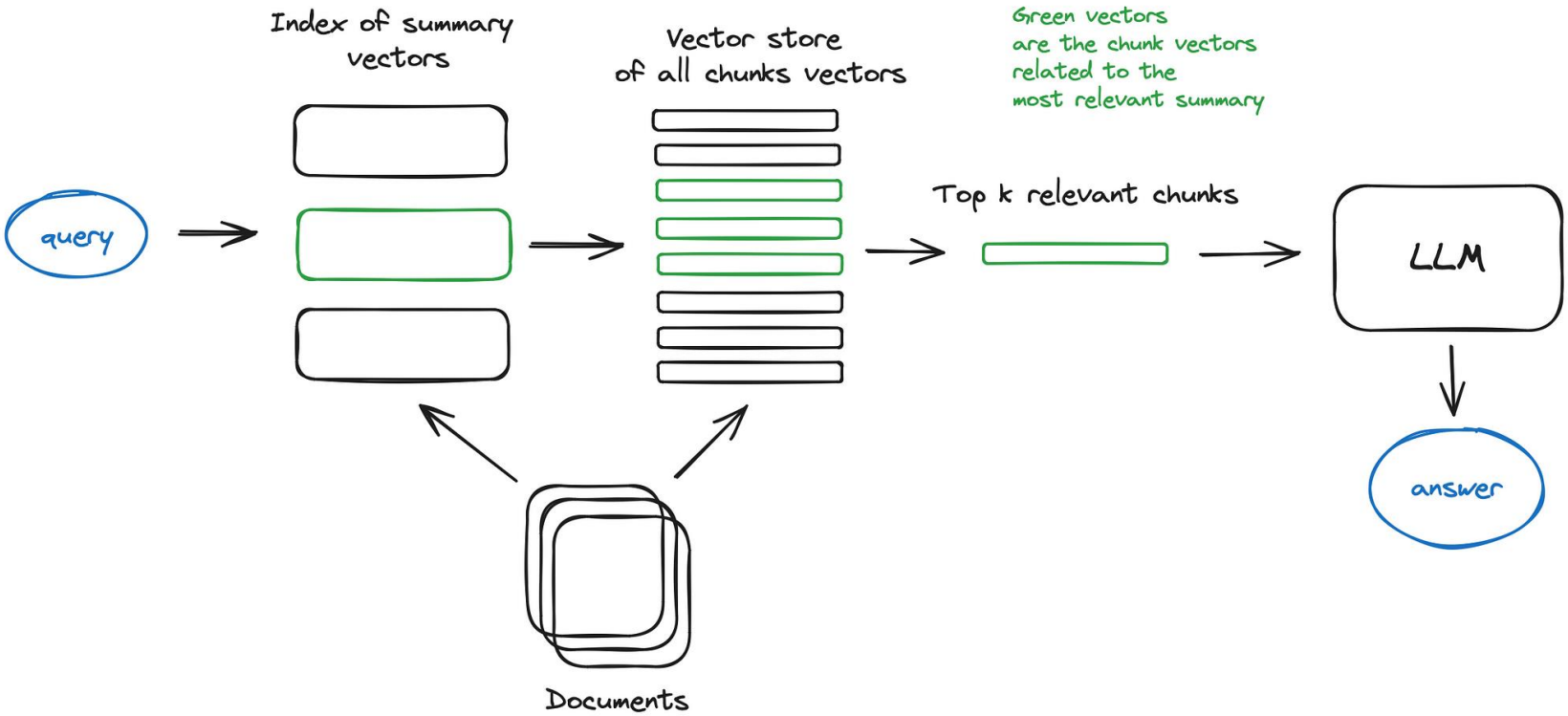
- Have to do a nearest-neighbor search to find relevant vectors for the query
- Many specialized indexes and vector stores have been developed to speed this up

Ranking Text “Chunks”



https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6

Index into Vector Store with Summary Vectors



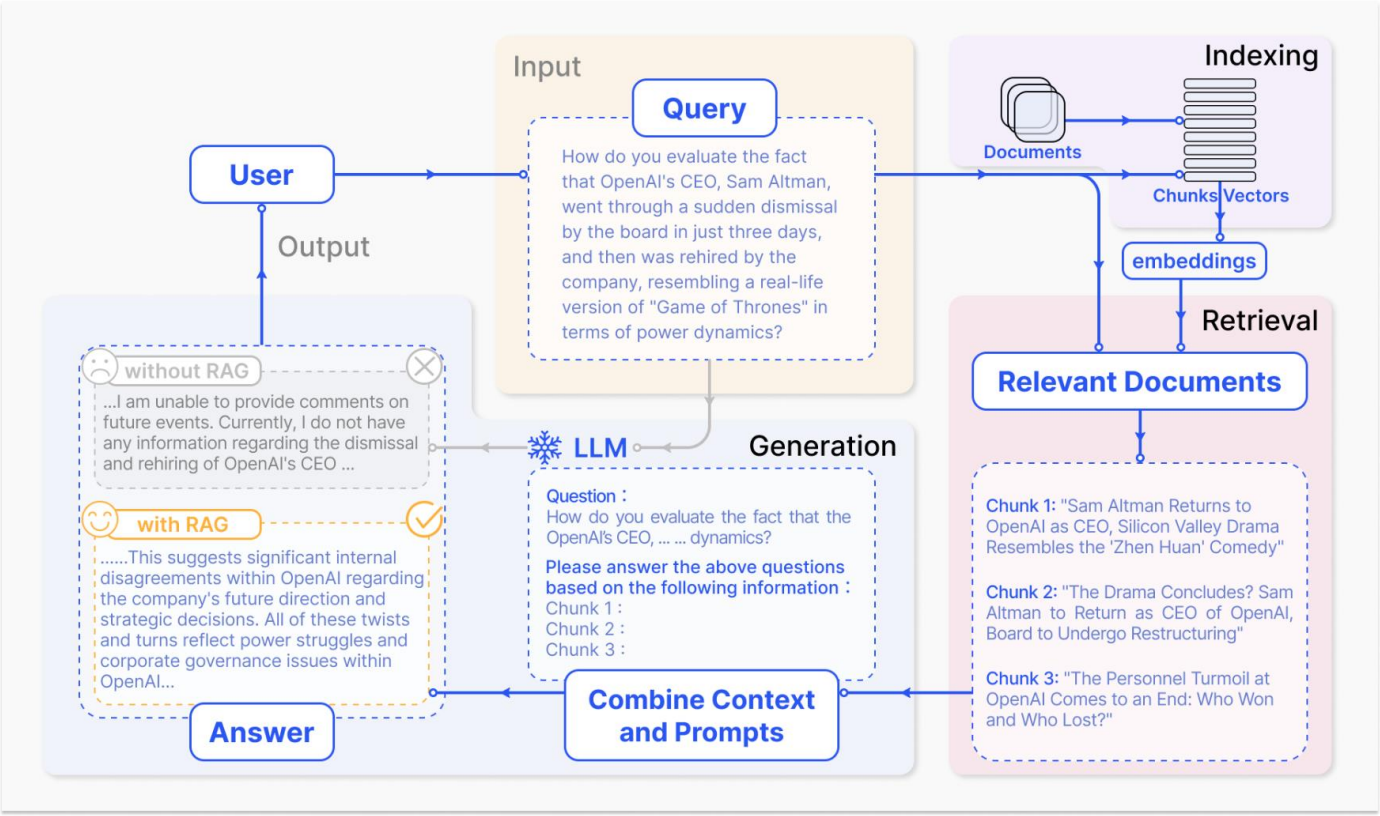


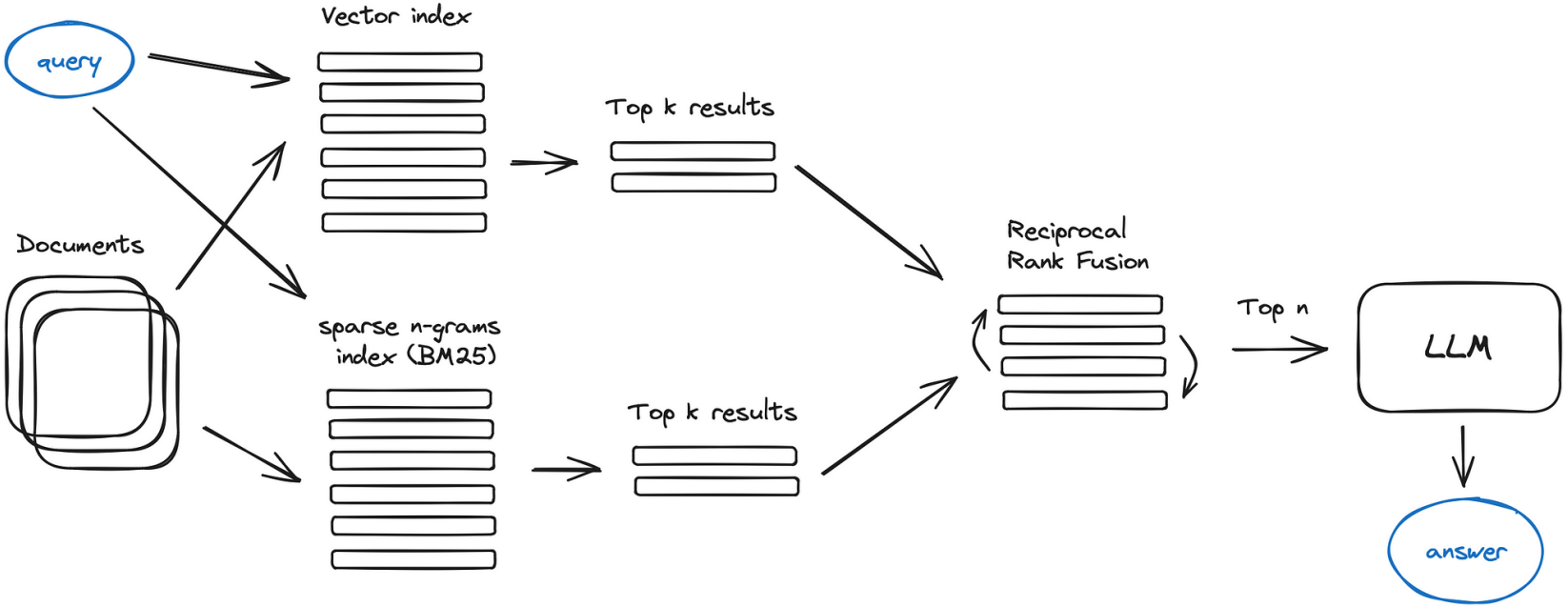
Fig. 2. A representative instance of the RAG process applied to question answering. It mainly consists of 3 steps. 1) Indexing. Documents are split into chunks, encoded into vectors, and stored in a vector database. 2) Retrieval. Retrieve the Top k chunks most relevant to the question based on semantic similarity. 3) Generation. Input the original question and the retrieved chunks together into LLM to generate the final answer.

Dense Plus Sparse Vectors

- Because dense (semantic) and sparse (lexical) vectors have complementary strengths, RAG systems often use **Hybrid Search**.
- This involves:
  1. *Generating **both** a dense vector (for meaning) and a sparse vector (like BM25, for exact keywords) for the query.*
  2. *Retrieving documents based on **both** the vector similarity search (FAISS) and the BM25 score.*
  3. *Combining the results using a weighted formula to get the best of both worlds.*

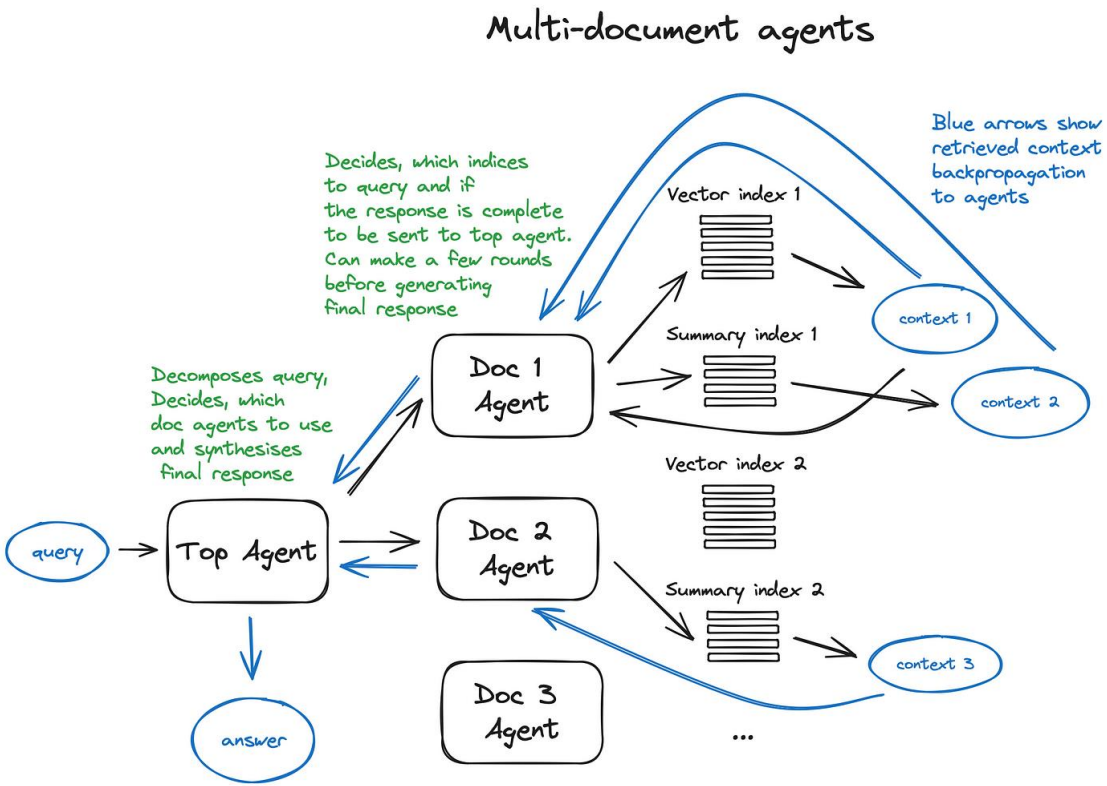
From Gemini

Hybrid Search Uses both Dense and Sparse (BM-25) Vectors



https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6

Agent-based Models and RAG



https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6

Implementation Considerations

Factor	Key Consideration	Performance Impact
Vector Database	HNSW vs. flat indexes	Retrieval speed and accuracy
Embedding Models	Domain-specific adaptation	Retrieval precision
Chunking Strategy	Semantic vs. fixed-length	Information relevance
Retrieval Algorithm	Multi-stage approaches	Precision-latency balance
Prompt Engineering	Integration techniques	Factual accuracy

HNSW: Hierarchical Navigable Small World graph-based index

SUMMARY: LLMS AND RAG

- LLMs generate text to follow a prompt
- They are a super-summary of the text they've seen
- They can make up information
- RAG is a method for anchoring what they generate to original sources
- However, it can be difficult to get RAG to rank passages well