Ethics contd.,
Question Answering / Retrieval

CSE 5525: Foundations of Speech and Natural Language Processing

https://shocheen.github.io/courses/cse-5525-spring-2026

THE OHIO STATE UNIVERSITY

Slide Credits: Graham Neubig, Maarten Sap, Ana Marasović

Logistics

- Reminder: Final quiz next Wednesday

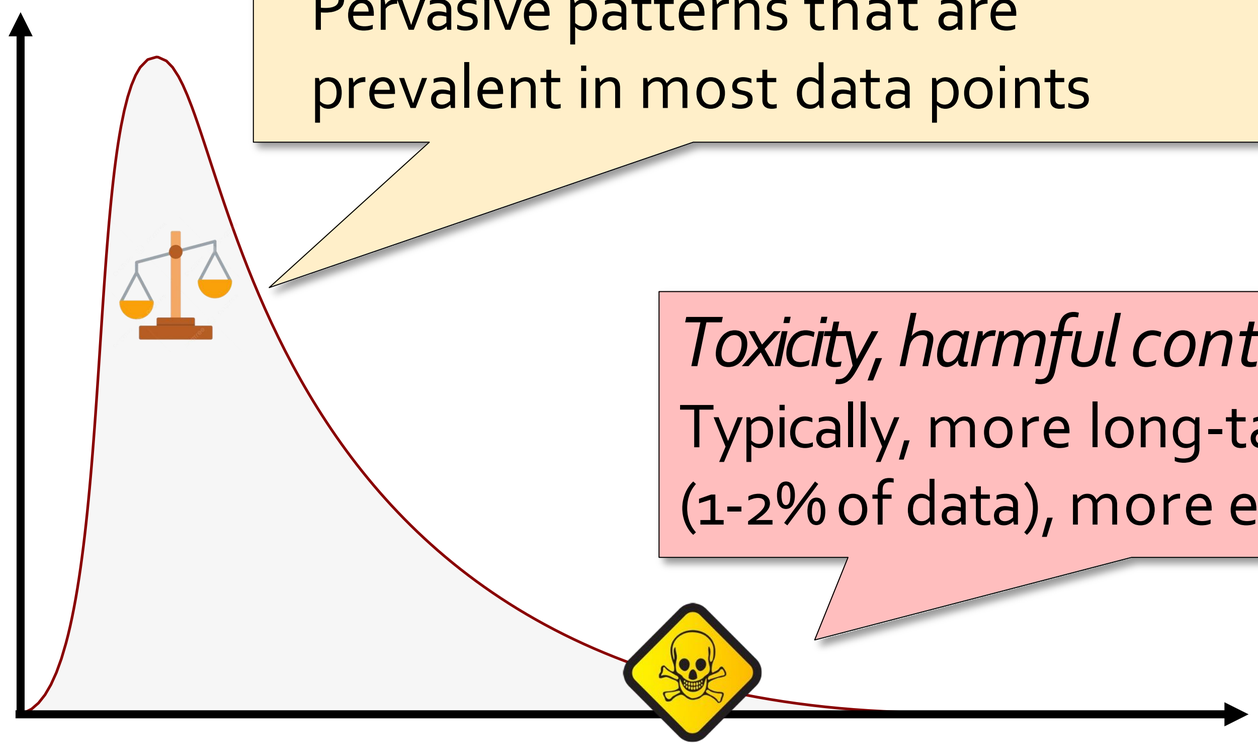
- Project presentations schedule is posted on teams/canvas.

Recap

- Bias in NLP systems:
  - How to measure bias / fairness
  - Where does bias come from
  - Can we mitigate it / how – designing better datasets/modeling/evaluation/systems
- Toxicity / Harmful outputs

Biases vs. toxicity / harmful content

Social biases & stereotypes:
Pervasive patterns that are prevalent in most data points



Toxicity, harmful content:
Typically, more long-tail phenomenon (1-2% of data), more extreme

What counts as harmful content?

- If you were building a language model, what would you not want the models to do?

Why do these models learning undesirable content?

Problems with self-supervised pretraining

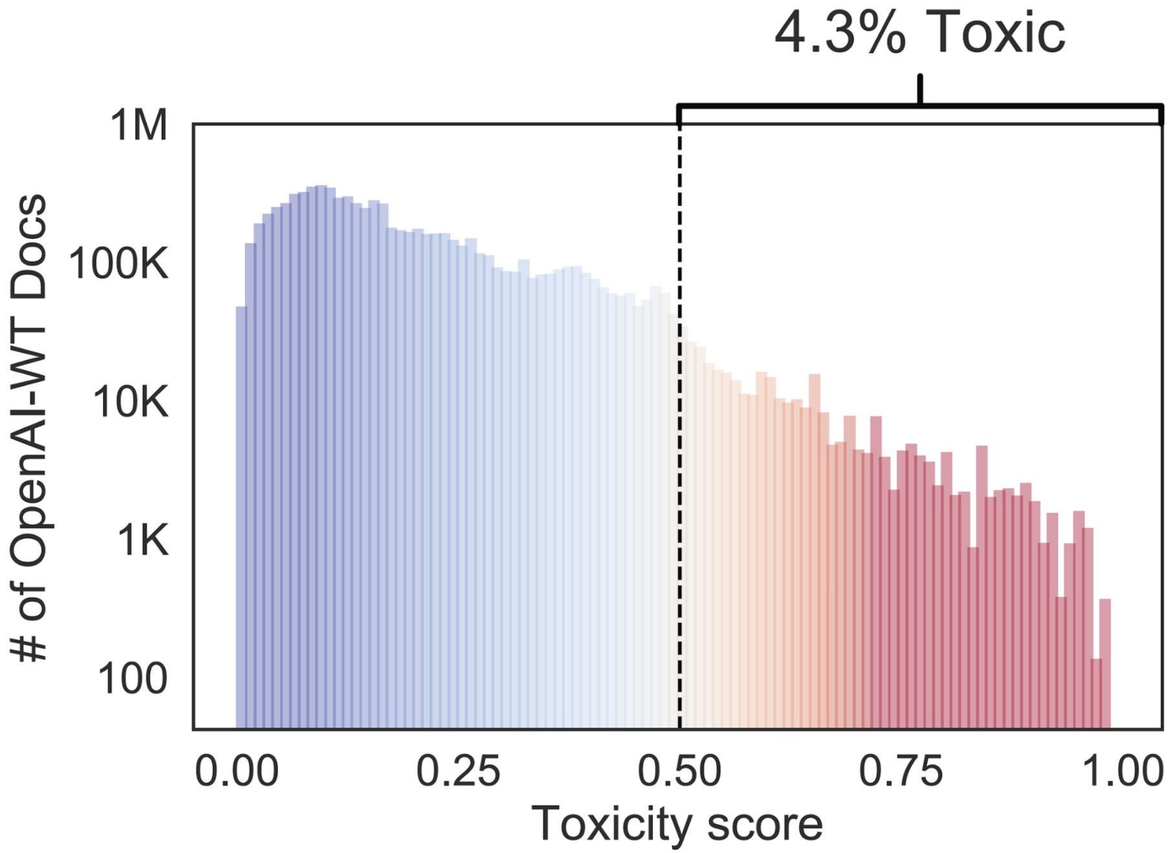
"Feeding AI systems on the world's beauty, ugliness, and cruelty, but expecting it to reflect only the beauty is a fantasy"

Prof. Ruha Benjamin, PhD

- Recipe: scrape as much pretraining data as you can to train your LM
- Consequence: LM ends up learning toxicity, biases, extremism, hate speech…

Toxicity in GPT-2's pretraining data

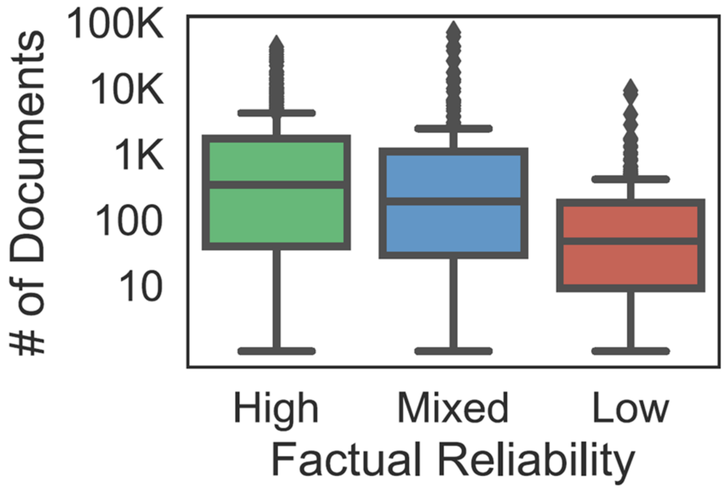
- Gehman et al (2020) accessed the actual GPT-2 training corpus (OpenAI-WT)
- Found >4% of documents (340,000) are toxic

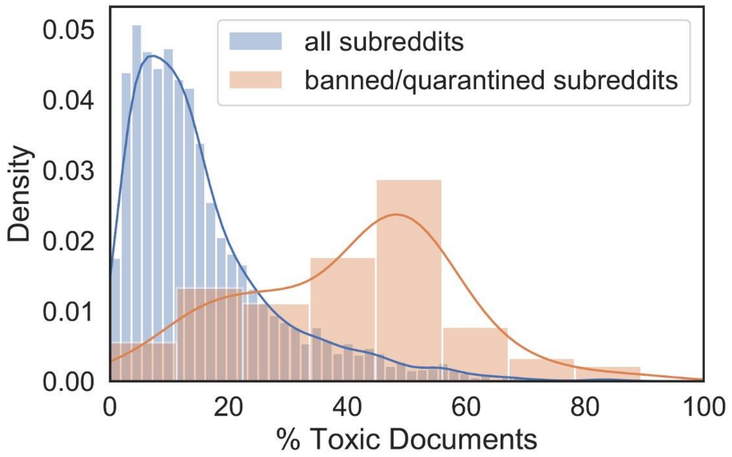


Fake news in GPT-2's pretraining data

- Also looked at sources of documents in training data
- Cross-referencing sources of documents with known factual reliability categorization
  - >272K (3.4%) docs from low/mixed reliability sources
- Examining source where document is shared
  - >200K (3%) docs linked from banned/quarantined subreddits, which typically are more toxic docs
- Important to examine training data
  - Can only do that if publicly released!

- *So… need approaches to safeguard your model against this undesirable content, knowledge, and text.*





How to safeguard your LLMs

aka

How to make sure models do not produce harmful content

Exercise: What would you do?

Overview – LLM safeguarding

- Filtering out toxic training data

- Topic-based filters
- Toxic content detection

- Write demonstrations for refusing to answer
- RLHF models to prefer non-toxic generations

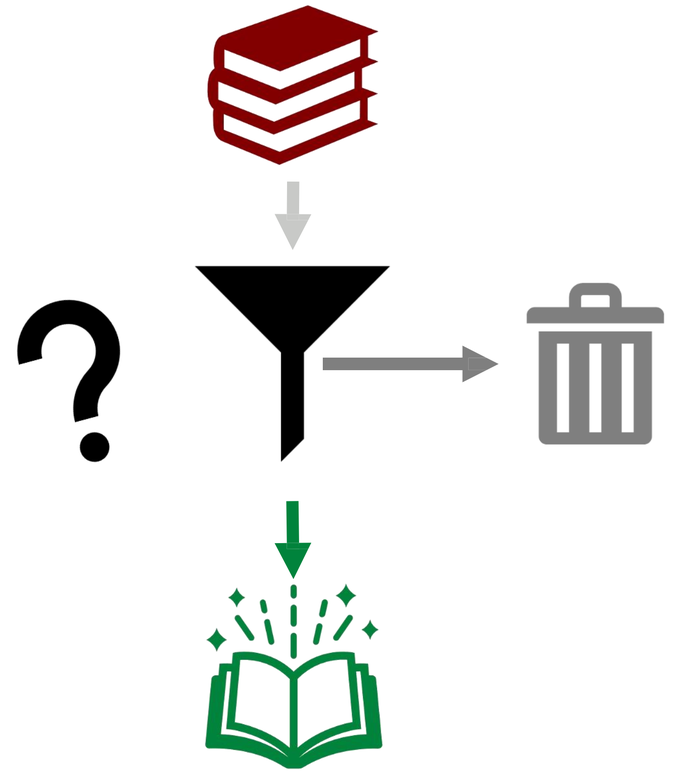
- Generate-then-classify
- Controllable text generation

Dataset filtering

- *Argument*: if you don't want your model to generate toxicity/hate speech, do not train it on such data (garbage in, garbage out)

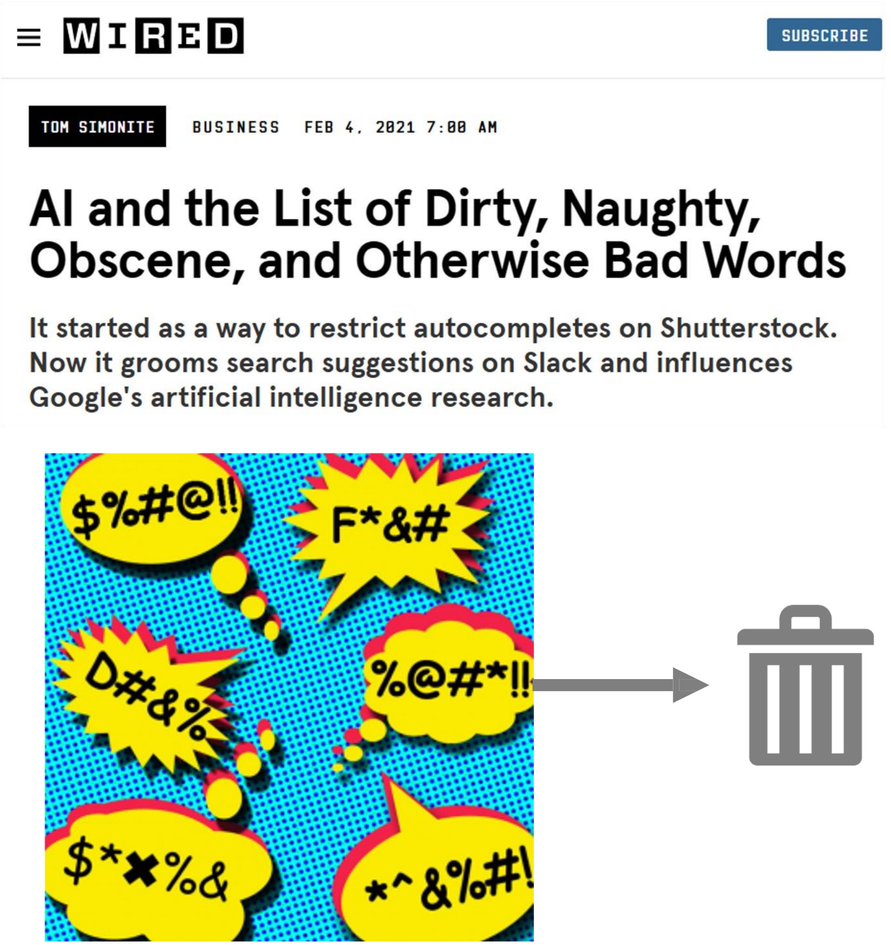
- *Approach*: data filtering to ensure "high quality"

- How do you know what is "high quality" ?
  - GPT-2: Reddit "Karma" score as signal
  - T5, BERT: "blocklist" of "bad words"
  - GPT-3: "quality" classifier
  - Newer works: Combination of all



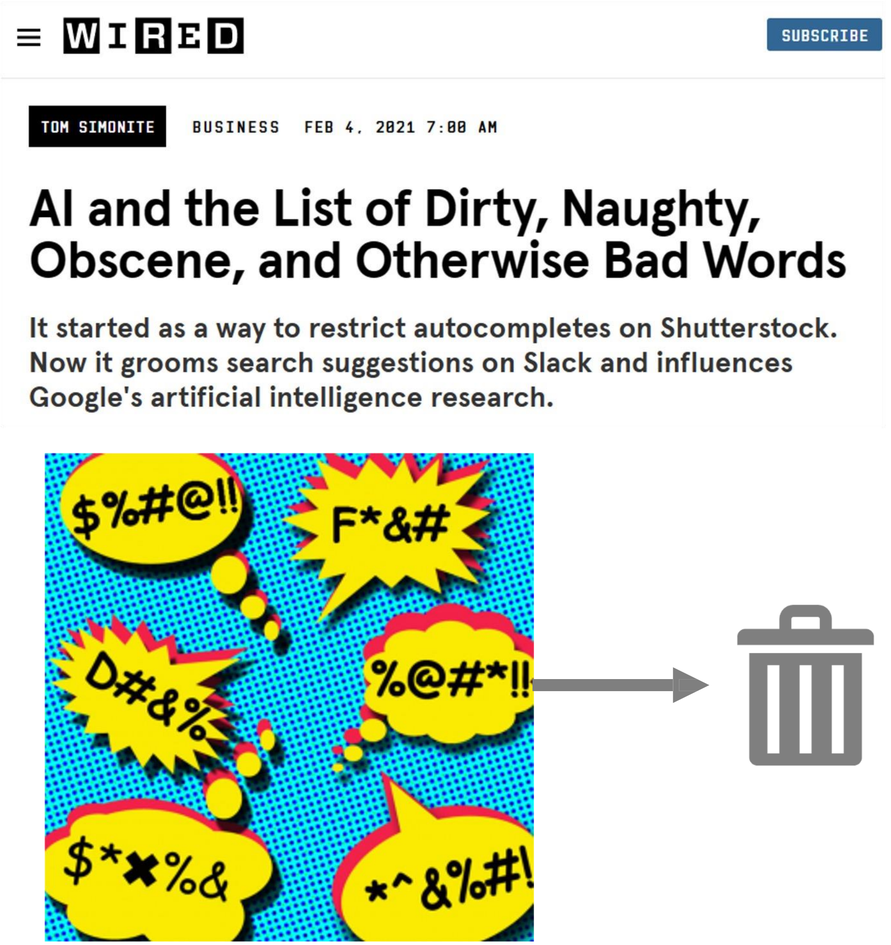
Blocklist of "bad" words

- "List of Dirty, Naughty, Obscene, or Otherwise Bad Words" originally by Shutterstock employees
  - Meant to prevent words in autocomplete settings
- Has been used by most companies creating LLMs
  - BERT, T5, GPT-2, etc.
- If document contains a "bad" word, remove it from training data
  - F*ck, sh*t, sex, vagina, viagra, n**ga, f*g, b*tch, etc.
- *Let's discuss*: what are issues with this?



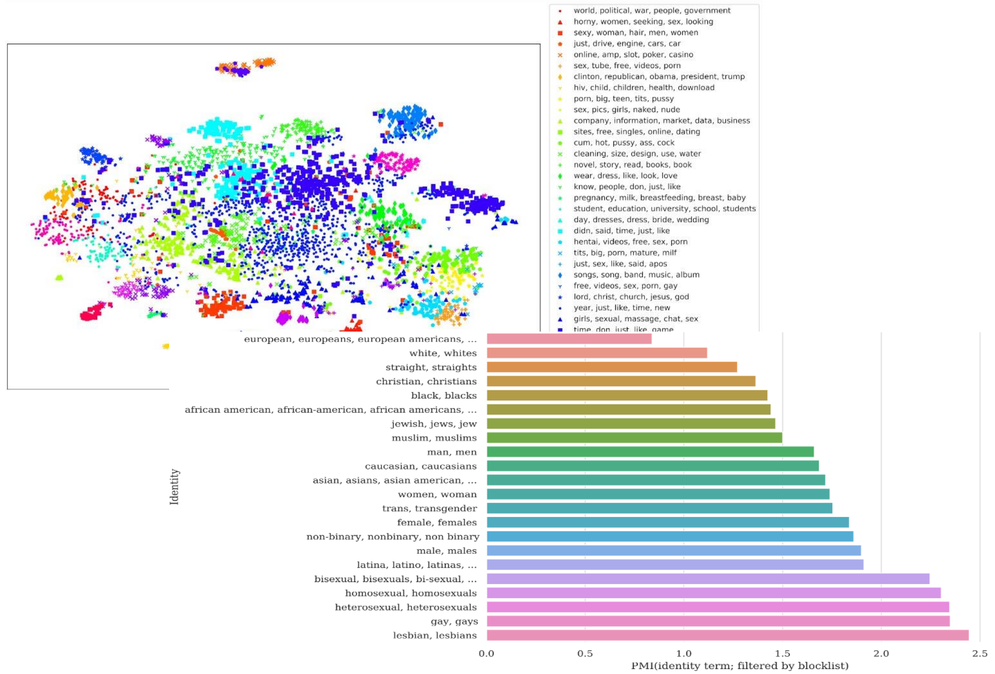
Blocklist of "bad" words

- "List of Dirty, Naughty, Obscene, or Otherwise Bad Words" originally by Shutterstock employees
  - Meant to prevent words in autocomplete settings
- Has been used by most companies creating LLMs
  - BERT, T5, GPT-2, etc.
- If document contains a "bad" word, remove it from training data
  - F*ck, sh*t, sex, vagina, viagra, n**ga, f*g, b*tch, etc.
- *Let's discuss*: what are issues with this?
  - Strong risk of over-deleting bio, legal, minority content



Effect of “bad word” blocklist filtering

- Dodge et al examined the effect of blocklist filtering on the C4 corpus
- When looking at 100k documents that were excluded due to “bad words”
  - Found only 31% related to porn/explicit sex
  - Remaining was biology, medicine, legal
- Also examined the effect on which minority identities were removed
  - Found queer/LGBTQ identity terms removed more
- Examined dialects removed due to “bad words”
  - Found AAE, Hispanic English more likely to be removed



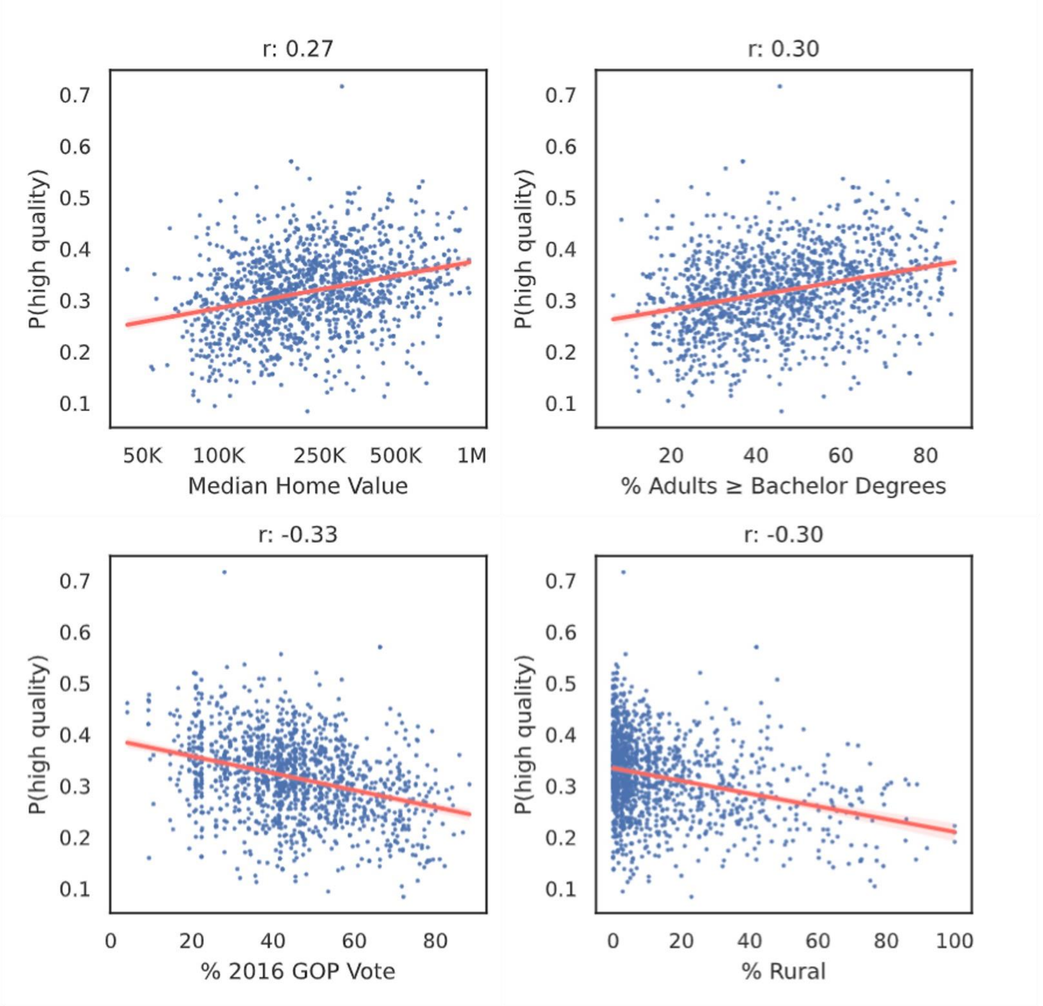
Less likely to be removed
- White-aligned English (6%)
- Other English (7%)

More likely to be removed
- African-American English (42%)
- Hispanic-aligned English (32%)

GPT3 Quality filter backfires

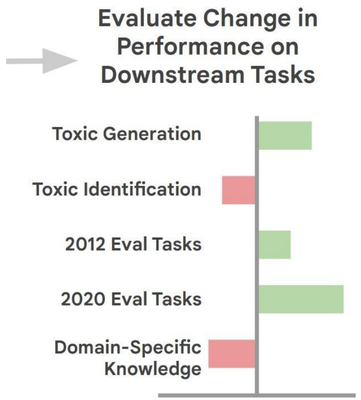
- GPT3 quality filter: similar to GPT2 data
- Gururangan et al. (2022) re-implemented GPT-3 quality filter
- Ran it on articles from school newspapers, which have metadata
- Filter assigns higher quality to articles from
  - Richer counties 💰
  - Counties with more educated adults 👨‍🎓
  - More liberal counties
  - More urban counties
- Raises language ideology question: Whose English is "good English"?

"In order to improve the quality of Common Crawl, we developed an automatic filtering method to remove low quality documents. Using the original WebText as a proxy for high-quality documents, we trained a classifier to distinguish these from raw Common Crawl." – Brown et al. 2020



So… maybe filtering isn’t a good idea since it’ll backfire?

Why LLMs might want to have seen toxic content



- Detecting hate speech [Chiu et al 2022]
  - Longpre et al. (2023) showed that LLMs trained on more toxicity are better toxicity detections
  - Improving hate speech models with data augmentation: ToxiGen [Hartvigsen et al 2022]
- Counter speech generation [Saha et al 2022, Kim et al 2022, Mun et al 2023]

- *If we train on toxicity, something else must be done at a different time!*

Overview – LLM safeguarding

- Filtering out toxic training data

- Topic-based filters
- Toxic content detection

- Write demonstrations for refusing to answer
- RLHF models to prefer non-toxic generations

- Generate-then-classify
- Controllable text generation

Train models to refuse to answer questions

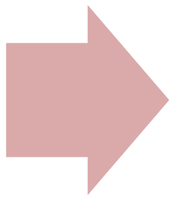
- Include in your training datasets some examples like this:
  - Prompt: Asks for something harmful directly or indirectly. Such as: how do I create an undetectable poison, what are some ways I can humiliate my friend, how can I quickly die…..
  - Responses: some version of [I cannot answer these questions because …]

- Include these both during SFT and RL stages.
  - SFT: prompt and expected response
  - RL: prompt, preferred response (refusal), dispreferred response (compliance).

RLHF safeguarding – assumptions

PPO & family:

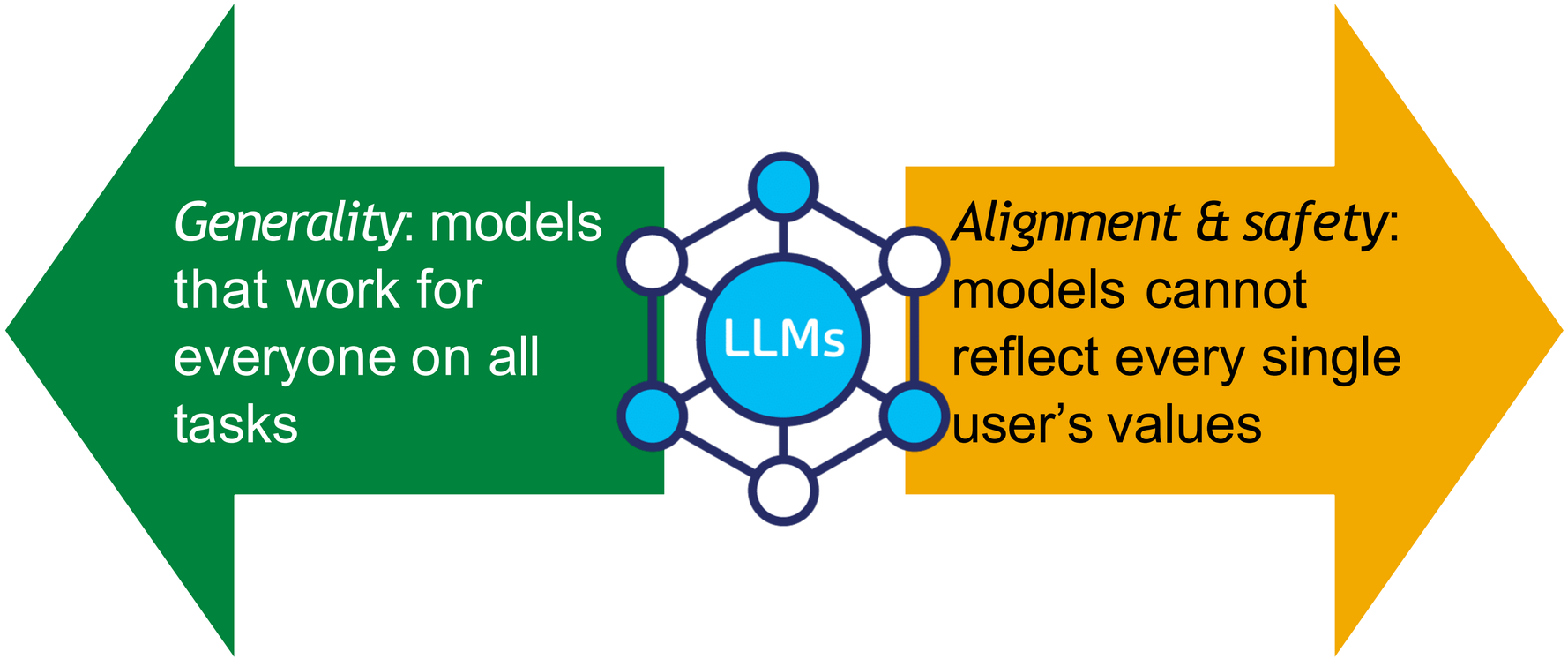
Obtain preference data: which generation is good vs. bad?



RL is done to encourage more like "preferred output"

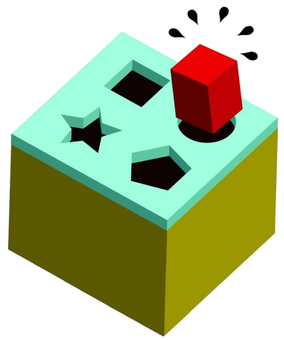
- Big question: what does it mean for a generation to be better/preferred?
  - How to balance harmless and helpful? [Bai et al '23]
    - *E.g., "help me create a poisonous drink."*
  - What if people's preferences are biased or gameable?
    - *E.g., people prefer certainty over uncertainty in answers to questions [Zhou et al. 24]*
  - Fundamental issue: cannot represent all values and cultures into one ranking.
    - *Casper et al. 2023. "Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback." arXiv [cs.AI]. arXiv. http://arxiv.org/abs/2307.15217*

Big unresolved tension



So… what can we do?

- Need to keep studying what models can and can't do, who they work for and don't work for
- Narrow scope of model users
  - Community-specific models (e.g., Masakhane Initiative)
- Specialize models' abilities / away from one-size-fits-all
  - E.g., toxicity explanation generation model needs to generate stereotypes, but story generation models might not
- In line with many legislative efforts:
- legislate the application or task, not the model



Takeaways

- AI systems are biased
- Real world is biased, data is biased
- ML objectives play a role
- Annotation interfaces, context plays a role
- Debiasing is challenging, requires socio-technical lens
- Toxicity and undesirable content
- Longer-tail phenomenon, present in training data
- Filtering data can backfire
- Safeguarding to all people is impossible
- Any questions?

Safeguards from training data	• Filtering out toxic training data
Safeguards from input prompt classification	• Topic-based filters • Toxic content detection
Safeguards from instruction-tuning & RLHF	• Write demonstrations for refusing to answer • RLHF models to prefer non-toxic generations
Safeguards at the output level	• Generate-then-classify • Controllable text generation

Question Answering / Retrieval

Goal: Dive into one NLP application: Question Answering

- QA Landscape
- An Overview of Retriever-Reader & Retriever-Generator Architectures for Open-Ended QA
- Retrieval
- Answer Extractor
- Retrieval Augmented Generation (RAG)
- RAG: Overview of Retriever-Generator training options

Question Answering

We fill our information needs by talking to a virtual assistant or a chatbot, interacting with a search engine, or querying a database

No wonder that question answering has been a major task in NLP

But there isn't one QA task

Intent behind the question:

Was the person seeking information they did not have,

… or trying to test the knowledge of another person or machine?

The task of **Question Answering (QA)** in NLP is more often associated with information-seeking questions:

- The questions are real queries by users of products like Google Search, Reddit, StackOverflow, etc
- Thus, questions are often ill-specified, full of "ambiguity and presupposition"
- Less frequently involve complex reasoning
- Questions often assume no given context and are almost never poised as multiple choice
- Industrial research because research progress directly translates to improved products

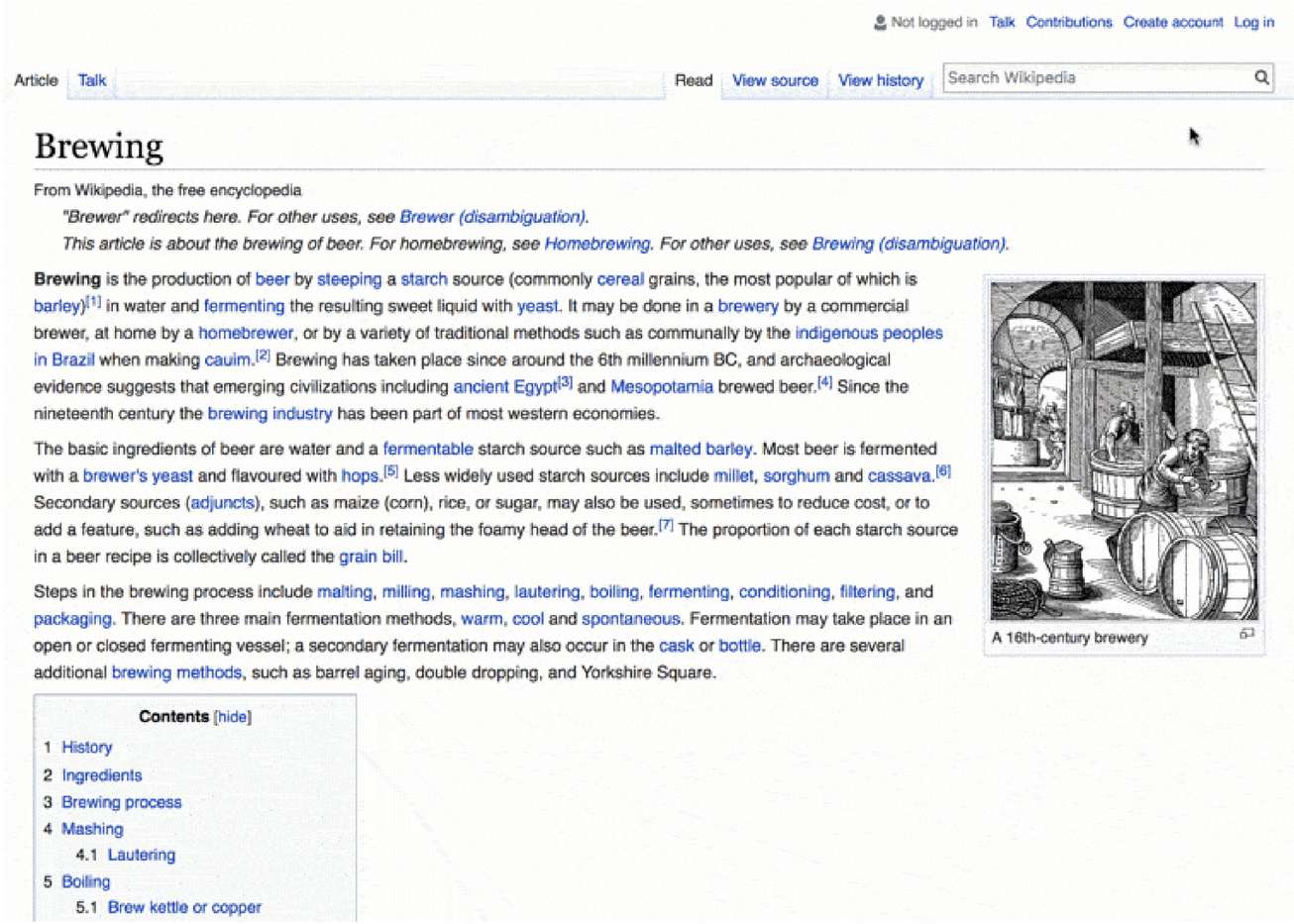
[Rogers et al., 2022]

Example of such QA: NaturalQuestions [Kwiatkowski et al., 2019]

Question: when are hops added to the brewing process?

Short answer: The boiling process

Long answer:
After mashing , the beer wort is boiled with hops (and other flavourings if used) in a large tank known as a " copper " or brew kettle – though historically the mash vessel was used and is still in some small breweries . The boiling process is where chemical reactions take place , including sterilization of the wort to remove unwanted bacteria , releasing of hop flavours , bitterness and aroma compounds through isomerization , stopping of enzymatic processes , [...]



Not logged in Talk Contributions Create account Log in

Article Talk | Read View source View history | Search Wikipedia

Brewing

From Wikipedia, the free encyclopedia

"Brewer" redirects here. For other uses, see Brewer (disambiguation).

This article is about the brewing of beer. For homebrewing, see Homebrewing. For other uses, see Brewing (disambiguation).

Brewing is the production of beer by steeping a starch source (commonly cereal grains, the most popular of which is barley)[1] in water and fermenting the resulting sweet liquid with yeast. It may be done in a brewery by a commercial brewer, at home by a homebrewer, or by a variety of traditional methods such as communally by the indigenous peoples in Brazil when making cauim.[2] Brewing has taken place since around the 6th millennium BC, and archaeological evidence suggests that emerging civilizations including ancient Egypt[3] and Mesopotamia brewed beer.[4] Since the nineteenth century the brewing industry has been part of most western economies.

The basic ingredients of beer are water and a fermentable starch source such as malted barley. Most beer is fermented with a brewer's yeast and flavoured with hops.[5] Less widely used starch sources include millet, sorghum and cassava.[6] Secondary sources (adjuncts), such as maize (corn), rice, or sugar, may also be used, sometimes to reduce cost, or to add a feature, such as adding wheat to aid in retaining the foamy head of the beer.[7] The proportion of each starch source in a beer recipe is collectively called the grain bill.

Steps in the brewing process include malting, milling, mashing, lautering, boiling, fermenting, conditioning, filtering, and packaging. There are three main fermentation methods, warm, cool and spontaneous. Fermentation may take place in an open or closed fermenting vessel; a secondary fermentation may also occur in the cask or bottle. There are several additional brewing methods, such as barrel aging, double dropping, and Yorkshire Square.

A 16th-century brewery

Contents [hide]

1 History
2 Ingredients
3 Brewing process
4 Mashing
4.1 Lautering
5 Boiling
5.1 Brew kettle or copper

Example source: https://ai.google.com/research/NaturalQuestions/visualization

But there isn't one QA task

Intent behind the question:

> Was the person seeking information they did not have,
>
> > … or trying to test the knowledge of another person or machine?

The task of **Question Answering (QA)** in NLP is more often associated with information-seeking questions:

- The questions are real queries by users of products like Google Search, Reddit, StackOverflow, etc
- Thus, questions are often ill-specified, full of "ambiguity and presupposition"
- Less frequently involve complex reasoning
- Questions often assume no given context and are almost never poised as multiple choice
- Industrial research because research progress directly translates to improved products

Reading Comprehension (RC) is more associated with probing questions

- A block of text that contains information relevant to the questions being asked is provided
- Probe the understanding of the given block of text

[Rogers et al., 2022]

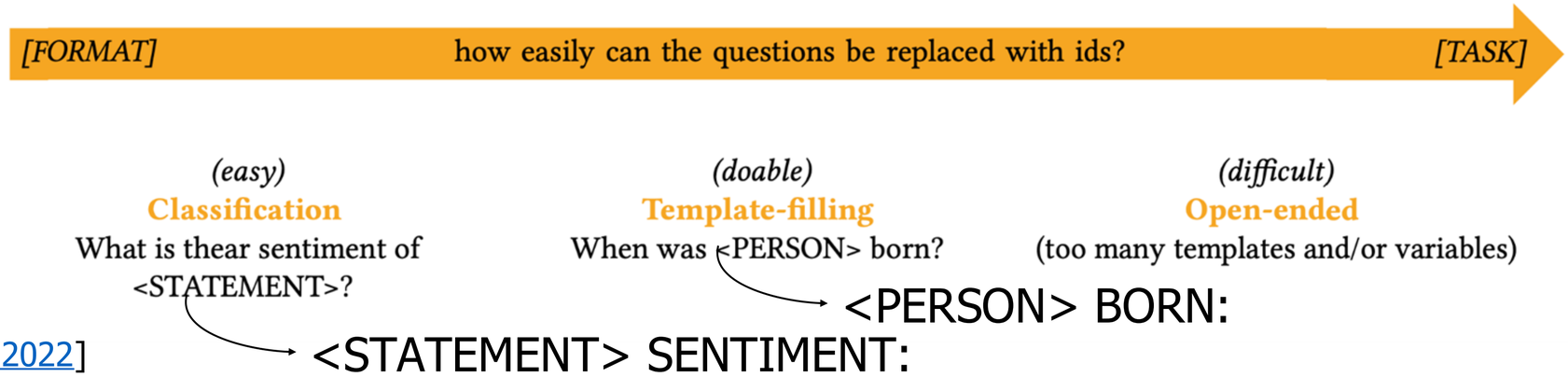
Almost any NLP task can be formulated as question answering

This is leveraged for model reuse, multi-task learning, prompting

- Example: *"Is this movie review sentence negative or positive?"*

In such cases, QA is **not a task but a format:** "a way of posing a particular problem to a machine, just as classification or natural language inference are formats" [Gardner et al., 2019]

The key distinction to keep in mind is "how easy would it be to replace the questions in a dataset with content-free identifiers?" [Gardner et al., 2019]



[Rogers et al., 2022]

Question formats

Evidence	Format	Question	Answer	Example datasets
Einstein was born in 1879.	Questions	When was Einstein born?	1879	SQuAD [235], RACE [156]
	Queries	Which year Einstein born	1879	generated queries in BEIR [282]
	Cloze	Einstein was born in ____.	1979	CNN/Daily Mail [125], CBT [127]
	Completion	Einstein was born ...	in 1879	SWAG [319], RocStories [204]

Natural questions: questions that a human speaker could ask

- Yes/no questions (Did it rain on Monday?)
- Wh-questions (When did it rain?)

Queries: Pieces of information that could be interpreted as a question

Cloze format: Sentences with a masked span that the model needs to predict

Story completion: The choice of the alternative endings for the passage

[Rogers et al., 2022]

Answer formats

Evidence	Format	Question	Answer(s)	Example datasets
Einstein was born in 1879.	Extractive	When was Einstein born?	1879 (token 5)	SQuAD [235], NewsQA [287]
	Multi-choice	When was Einstein born?	(a) 1879, (b) 1880	RACE [156]
	Categorical	Was Einstein born in 1880?	No	BoolQ [65]
	Freeform	When was Einstein born?	1879 (generated)	MS MARCO [19], CoQA [238]

Extractive format / extractive QA

- Either get or find short text segments from the web or some other large collection of documents
- Answer a question by extracting the answer from the evidence text
  - Extracting = find the starting and ending token in the input text
  - If the answer is generated ⇒ **Retrieval-augmented generation (RAG)**
- The limited range of answer options means that it is easier to define what an acceptable correct answer is
- It limits the kinds of questions that can be asked to questions with answers directly contained in the text

[Rogers et al., 2022]

Answer formats (cont.)

Evidence	Format	Question	Answer(s)	Example datasets
Einstein was born in 1879.	Extractive	When was Einstein born?	1879 (token 5)	SQuAD [235], NewsQA [287]
	Multi-choice	When was Einstein born?	(a) 1879, (b) 1880	RACE [156]
	Categorical	Was Einstein born in 1880?	No	BoolQ [65]
	Freeform	When was Einstein born?	1879 (generated)	MS MARCO [19], CoQA [238]

Multiple-choice format

- Questions for which a small number of answer options are given as part of the question text itself
- The answers are longer restricted to something explicitly stated in the text
- The question writer also has full control over the available options, and therefore over the kinds of reasoning that the test subject would need to be capable of
- Writing good multi-choice questions is not easy: if the incorrect options shouldn't be easy to rule out

[Rogers et al., 2022]

Answer formats (cont.)

Evidence	Format	Question	Answer(s)	Example datasets
Einstein was born in 1879.	Extractive	When was Einstein born?	1879 (token 5)	SQuAD [235], NewsQA [287]
	Multi-choice	When was Einstein born?	(a) 1879, (b) 1880	RACE [156]
	Categorical	Was Einstein born in 1880?	No	BoolQ [65]
	Freeform	When was Einstein born?	1879 (generated)	MS MARCO [19], CoQA [238]

Categorical format

- The answers come from a strictly pre-defined set of options (yes/no)

Freeform format

- Generate the answer independently rather than choose from the evidence or available alternatives
- The "gold" answer is probably not the only correct one, which makes evaluation difficult
- Evaluate on at least two axes: linguistic fluency and factual correctness; both hard

[Rogers et al., 2022]

Goal: Dive into one NLP application: Question Answering

- QA Landscape
- An Overview of Retriever-Reader & Retriever-Generator Architectures for Open-Ended QA
- Retrieval
- Answer Extractor
- Retrieval Augmented Generation (RAG)
- RAG: Overview of Retriever-Generator training options

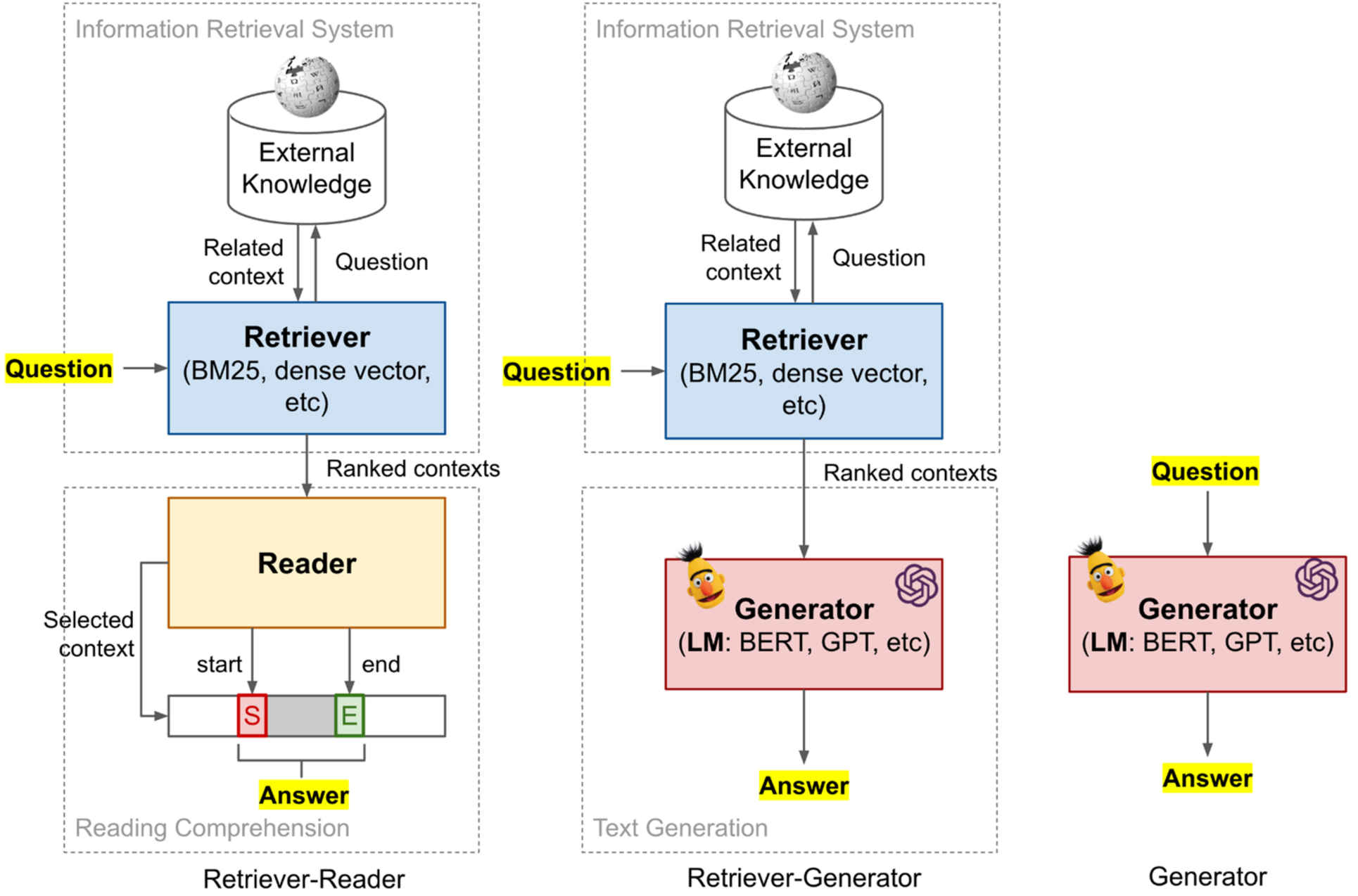
Open-domain Question Answering (ODQA): Asking a model to produce answers to, typically factoid, questions in natural language

The "open-domain" part:

- Relevant context is not provided and needs to be found:
  - When both the question and the context are provided, the task is known as *reading comprehension*
- Answer choices are not provided
  - When the choices are provided, the task is multiple-choice QA

General idea:

- An LLMs' pretraining data potentially didn't contain relevant information, or even if it did, the model is not capable enough to memorize it and refer to it when asked to answer a related question
- Find the relevant context where the answer is contained, then answer the question condition on the found context



Source: https://lilianweng.github.io/posts/2020-10-29-odqa/

Goal: Dive into one NLP application: Question Answering

- QA Landscape
- An Overview of Retriever-Reader & Retriever-Generator Architectures for Open-Ended QA
- Retrieval
- Answer Extractor
- Retrieval Augmented Generation (RAG)
- RAG: Overview of Retriever-Generator training options

Retrieval

- Goal: Given a query, find relevant documents that could answer the query.
  - Ideally: return a ranked list of documents by relevance.

- Input: Query Q, a set of documents {D1, D2, …., D_N }

- How would you design a retriever? Let's discuss.
  - What's the simplest thing you can do?

TF-IDF

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

Return documents that are the most similar to the query

TF-IDF ⇒ BM25 [Robertson et al., 1995]

BM25 is really just a more refined version of TF–IDF with **two additional hyperparameters** [Kamphuis et al. (2020)]

k, a knob that adjust the balance between term frequency and IDF,

- k→0, term frequency has almost no effect, IDF-driven
- E.g., keywords in short texts like tweets/titles, repetition stylistic, special domains

b, which controls the importance of document length normalization

$$\overbrace{\sum_{t \in q} \log\left(\frac{N}{df_t}\right)}^{\text{IDF}} \overbrace{\frac{tf_{t,d}}{k\left(1 - b + b\left(\frac{|d|}{|d_{\text{avg}}|}\right)\right) + tf_{t,d}}}^{\text{weighted tf}}$$

[Equation from: Jurafsky & Martin]

Inverted Index

We need to efficiently find documents that contain tokens in the question/query

The basic search problem in IR is thus to find all documents that contain a term

The data structure for this task is the **inverted index**, which we use for making this search efficient, and also conveniently storing useful information like the document frequency and the count of each term in each document

[Jurafsky & Martin]

Inverted Index (cont.)

An **inverted index**, given a query term, gives a list of documents that contain the term

It consists of two parts:

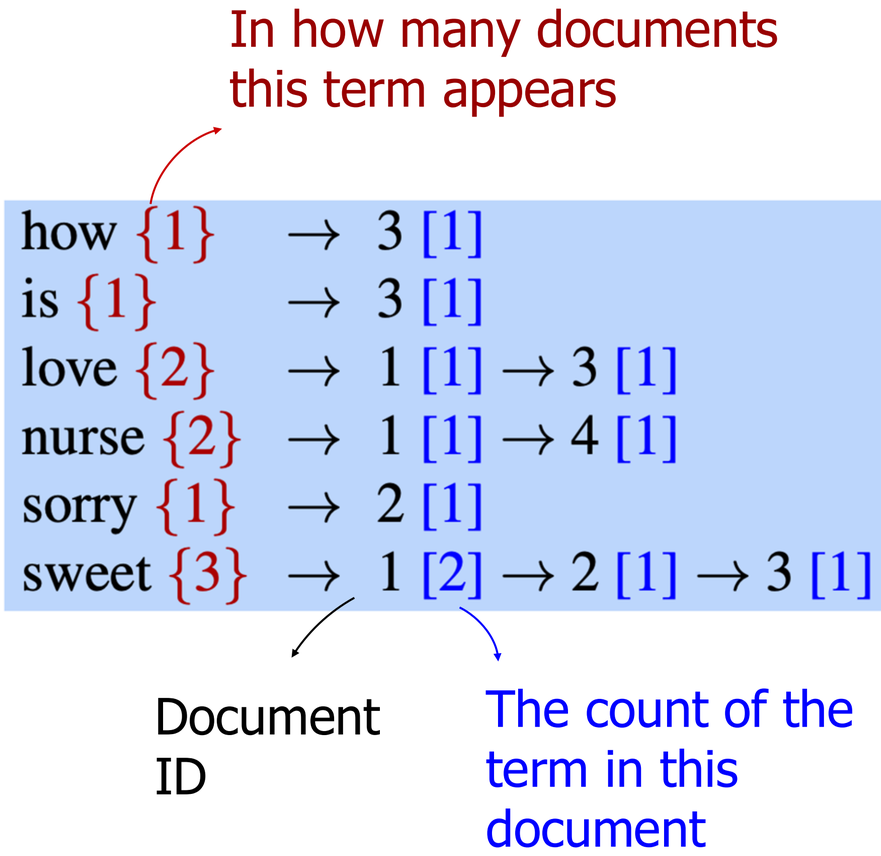
1. Dictionary
2. Postings

Dictionary is a list of terms (designed to be efficiently accessed), each pointing to a postings list for the term

- The dictionary can also store the document frequency for each term

A **postings list** is the list of document IDs associated with each term

- It can also contain information like the term frequency or even the exact positions of terms in the document



[Jurafsky & Martin]

Why could this classic IR approach to retrieval be limited?

tf-idf/BM25 algorithms work only if there is exact overlap of tokens between the question/query and document

What if a question contains a lot of synonyms of tokens in a relevant document?

Vocabulary mismatch problem: The user posing a query (or asking a question) needs

to guess exactly what words the writer of the answer might have used

- Instead of (sparse) word-count vectors, using (dense) embeddings

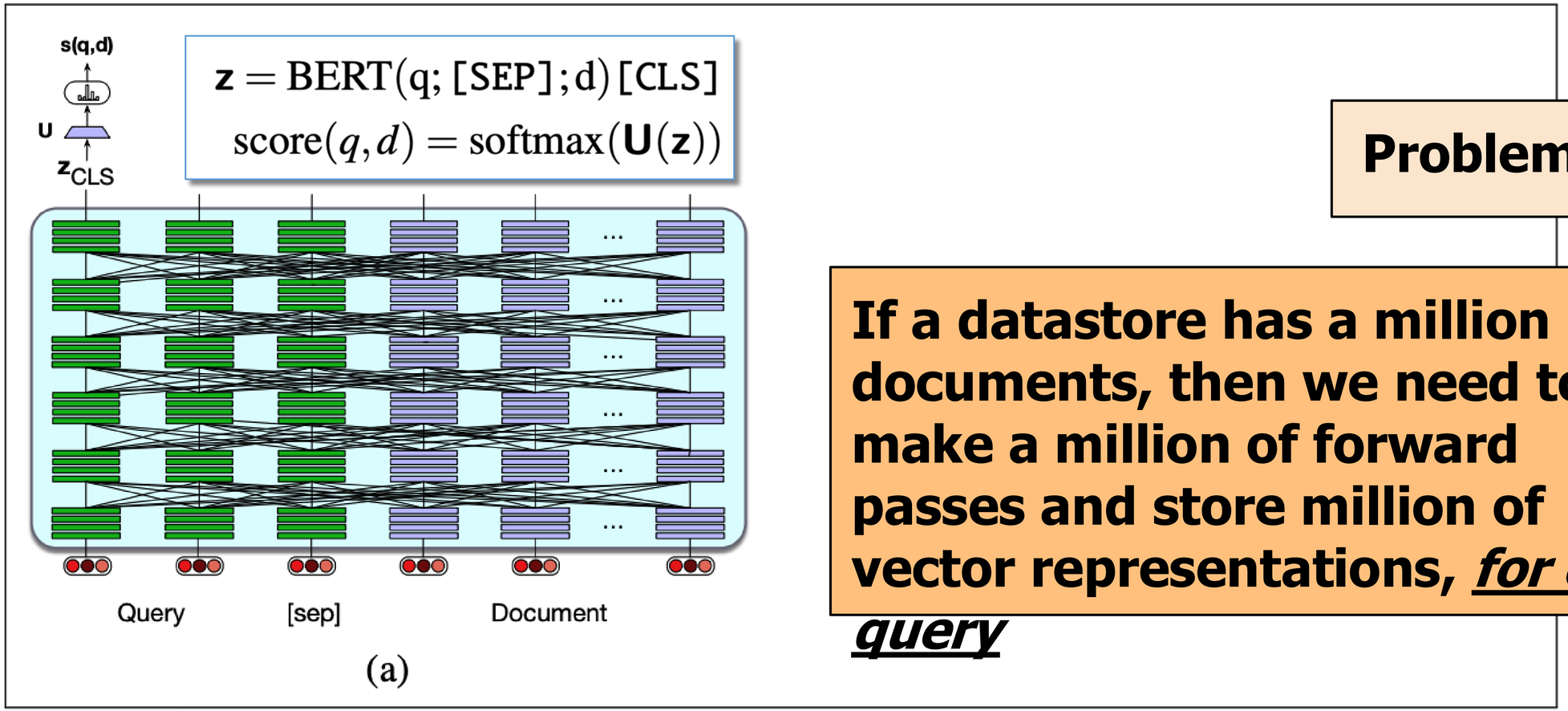
Dense Retrieval

- Learn a similarity function between query and document S(q, D) – parameterized by a neural network

- Given a new query, compute similarity which document in the collection and return a ranked list (or rather top-k documents in the ranked list)

Figure: [Jurafsky and Martin]

Two ways to dense retrieval



Problem?

If a datastore has a million documents, then we need to make a million of forward passes and store million of vector representations, *for each query*

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

Two ways to dense retrieva

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

Figure: [Jurafsky and Martin]

[Dense Passage Retrieval or **DPR;** Karpukhin et al., 2020]

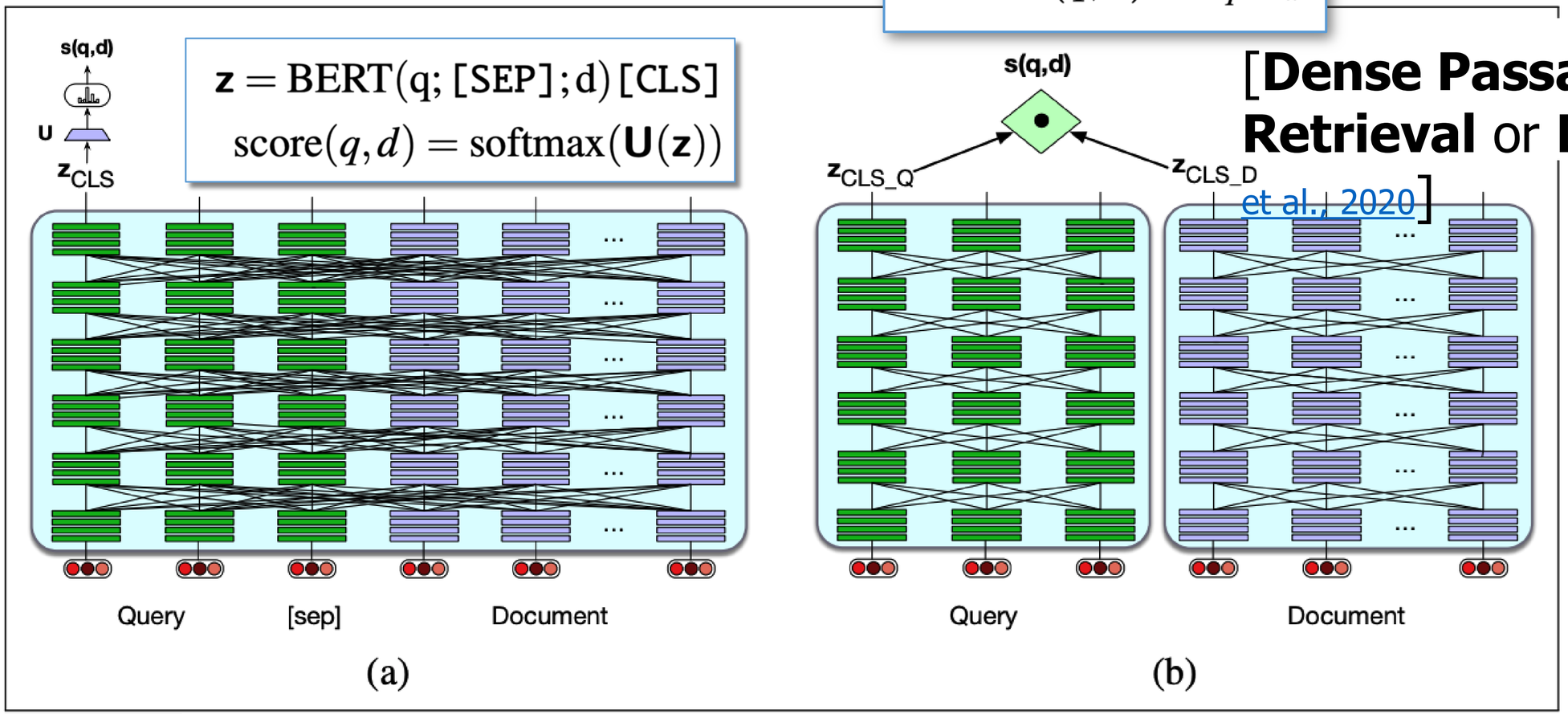


Figure 14.7 Two ways to do dense retrieval, illustrated by using lines between layers to schematically represent self-attention: (a) Use a single encoder to jointly encode query and document and finetune to produce a relevance score with a linear layer over the CLS token. This is too compute-expensive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

Another way to the second approach – ColBERT

[Khattab et al., 2021]

Problem: DPR representations are relatively coarse *They encode each passage into a single high-dimensional vector & estimate relevance via one dot-product*

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

BERT output vectors rescaled to unit length

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

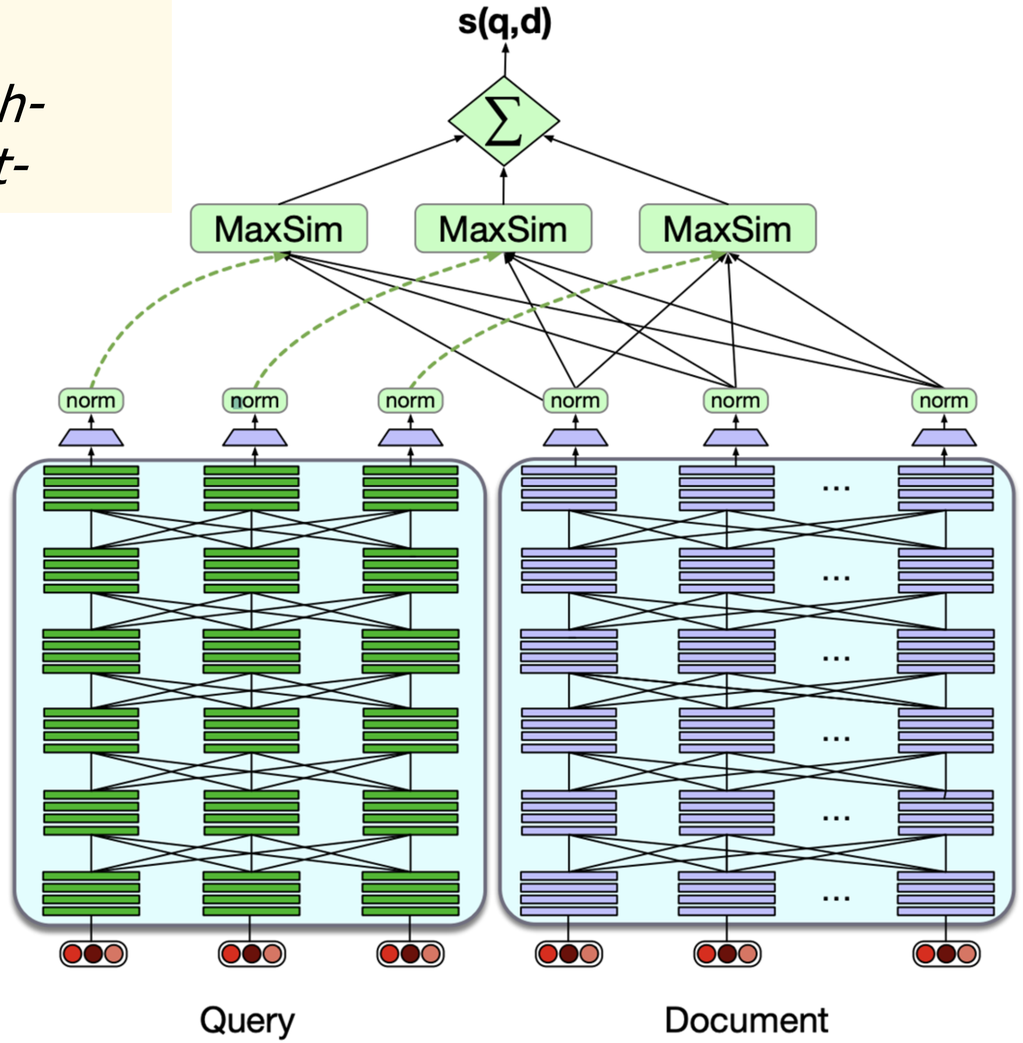


Figure: [Jurafsky and Martin]

Training dense retrievals - Overview

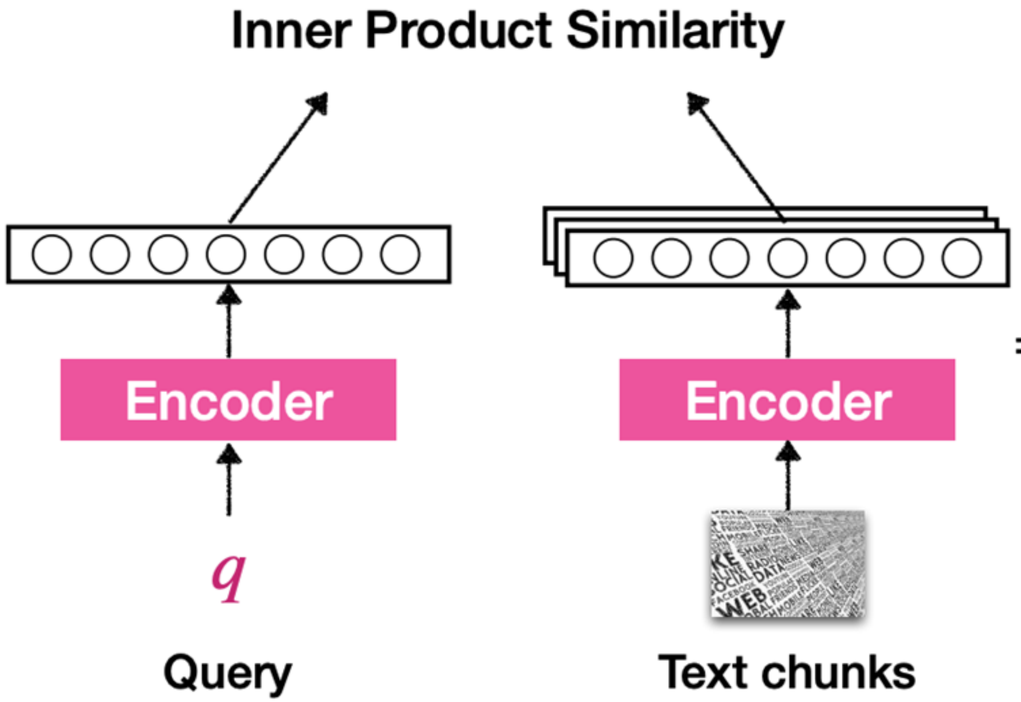
Objective: Train a model to predict the relevance of a document to a given query by optimizing a similarity score (e.g., dot product)

Training Data:

- Requires queries paired with relevant (positive) and irrelevant (negative) documents
- **Positive examples:** From datasets containing human-annotated relevant documents
- **Negative examples:**
  - Typically sampled from the top-1000 results retrieved by an existing retriever (e.g., BM25)
  - This ensures that the negatives are challenging ⇒ Discourage the model from learning only trivial distinctions between relevant and irrelevant documents
- If there are no labeled positive examples, use **relevance-guided supervision** [Khattab et al., 2021]:
  - Extract the top-ranked documents from an existing IR system that contain relevant answer strings (treat these as positive examples)
  - Use documents from the same top-ranked results that lack answer strings as negative examples
  - Train a new retriever and iterate

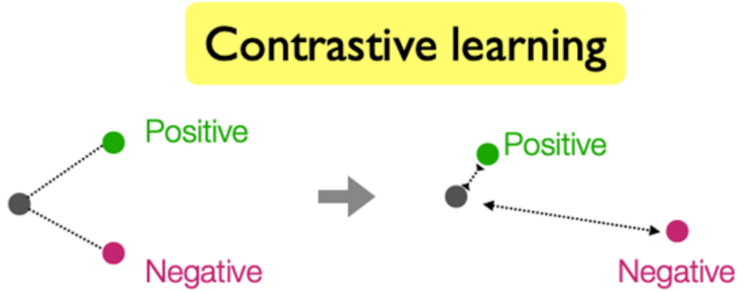
Training dense retrievals (cont.)

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



$$L(q, p^+, p_1^-, p_2^-, \ldots, p_n^-)$$

$$= -\log \frac{\exp(\text{sim}(q, p^+))}{\exp(\text{sim}(q, p^+)) + \sum_{j=1}^{n} \exp(\text{sim}(q, p_j^-))}$$



Source: ACL 2023 Tutorial: Retrieval-based Language Models and Applications; Section 3: Retrieval-based LMs: Architecture

Maximum Inner Product Search (MIPS)

MIPS: The problem of determining a small subset of items in a datastore whose vector representations have the highest dot product with a given query vector

Although there is an **obvious linear-time implementation O(#datastore items ·ᴑ· vector size)**, it is generally **too slow** to be used on practical problems because the datastores are huge (millions or billions of vectors, each hundreds or thousands of dimensions)

To optimize the retrieval speed, the common choice is the approximate nearest neighbors (ANN) algorithm to return approximately top-k nearest neighbors to **trade off a little accuracy lost for a huge speedup**; https://ann-benchmarks.com/

Index

During the **indexing phase**, each document/passage is encoded into a vector

All these document vectors are stored in the **index**, typically in a structure that allows for fast similarity search

FAISS (Facebook AI Similarity Search) https://github.com/facebookresearch/faiss

- A library developed by Meta (Facebook) to perform fast similarity search on large-scale dense vectors
- FAISS builds an index by dividing the dense space into clusters.
- At search time, the query vector is compared only with the most relevant clusters, significantly reducing the number of comparisons

Retrieval Evaluation

Each document returned by the IR system is either relevant to our purposes or not relevant

How to evaluate the retrieval then?

- **Precision:** # relevant documents among all documents retrieved (to be relevant)
- **Recall:** # truly relevant documents that are retrieved
- **F1:** as always, a weighted harmonic mean of the precision and recall

Issue: It doesn't really measure the performance of a system that ranks the documents

	What do retrieve?	How to use retrieval?	When to retrieve?
REALM (Guu et al 2020)	Text chunks	Input layer	Once
Retrieve-in-context LM (Shi et al 2023, Ram et al 2023)	Text chunks	Input layer	Every n tokens
RETRO (Borgeaud et al. 2021)	Text chunks	Intermediate layers	Every n tokens
kNN-LM (Khandelwal et al. 2020)	Tokens	Output layer	Every token
FLARE (Jiang et al. 2023)	Text chunks	Input layer	Every n tokens *(adaptive)*
Adaptive kNN-LM (He et al 2021, Alon et al 2022, etc)	Tokens	Output layer	Every n tokens *(adaptive)*
Entities as Experts (Fevry et al. 2020), Mention Memory (de Jong et al. 2022)	Entities or entity mentions	Intermediate layers	Every entity mentions
Wu et al. 2022, Bertsch et al. 2023, Rubin & Berant. 2023	Text chunks ***from the input***	Intermediate layers	Once or every n tokens

Source: [ACL 2023 Tutorial: Retrieval-based Language Models and Applications]; Section 3

Goal: Dive into one NLP application: Question Answering

- QA Landscape
- An Overview of Retriever-Reader & Retriever-Generator Architectures for Open-Ended QA
- Dense Retrieval
- Answer Extractor
- Retrieval Augmented Generation (RAG)
- RAG: Overview of Retriever-Generator training options

Figure: [Jurafsky and Martin]

Reader algorithms: Answer span extractor

Goal: Given retrieved text (e.g., passage) and the question, predict which token is the start of the answer span and which token is its end

S, E … vectors of the size of the final token representations, randomly initialized & trained

Concatenate question & passage, and delineate them:

- With BERT-like models, use [SEP]
- With today's LM use text like "Passage:"

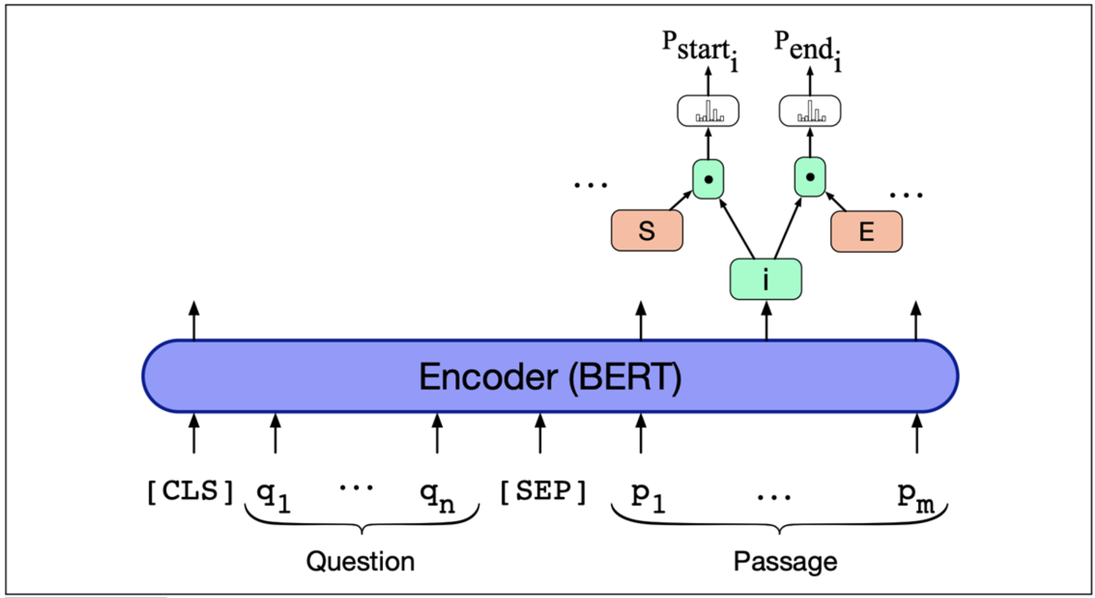


Figure 14.12 An encoder model (using BERT) for span-based question answering from reading-comprehension-based question answering tasks.

A span-start probability for each position i:

$$P_{\text{start}_i} = \frac{\exp(S \cdot p'_i)}{\sum_j \exp(S \cdot p'_j)}$$

The final representation of the j-th token

A span-end probability for each position i:

$$P_{\text{end}_i} = \frac{\exp(E \cdot p'_i)}{\sum_j \exp(E \cdot p'_j)}$$

Figure: [Jurafsky and Martin]

Reader algorithms: Answer span extractor

Minimizing a negative log-likelihood loss encourages the model to maximize the probabilities of the true start (i*) and end (j*) positions

The probability of correctly predicting the answer span involves:

1. Predicting the correct start position
2. Predicting the correct end position

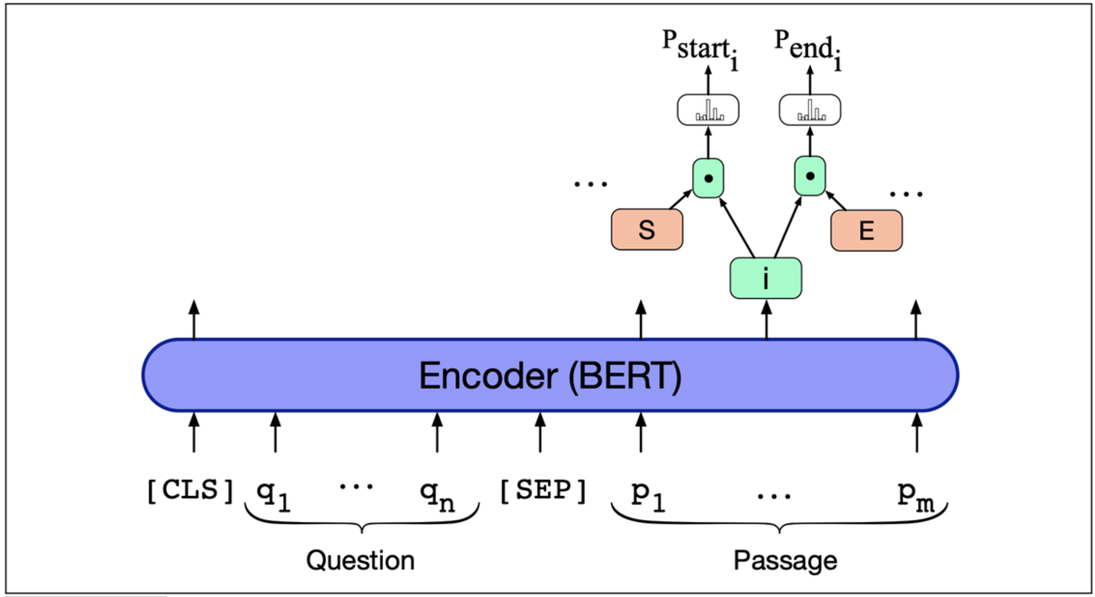


Figure 14.12 An encoder model (using BERT) for span-based question answering from reading-comprehension-based question answering tasks.

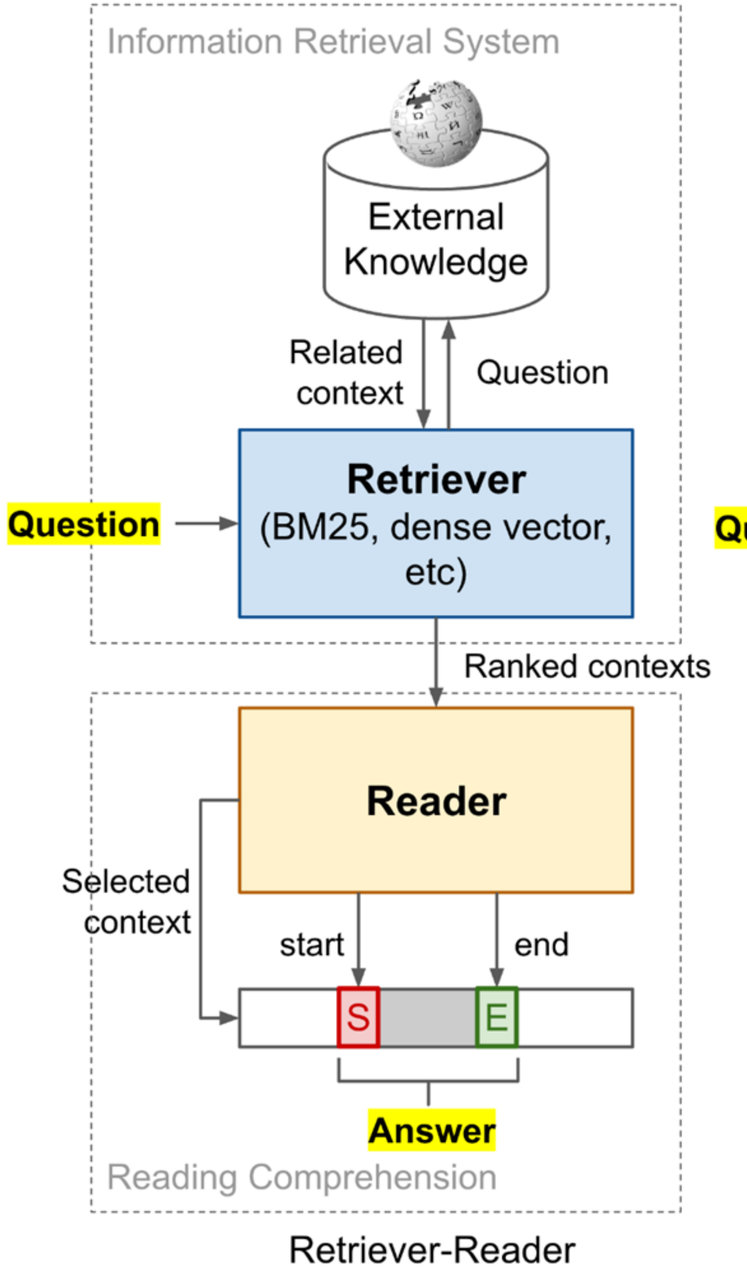
These two events are treated as independent events, the joint probability of both predictions being correct is the product of the two probabilities

Therefore the loss is:

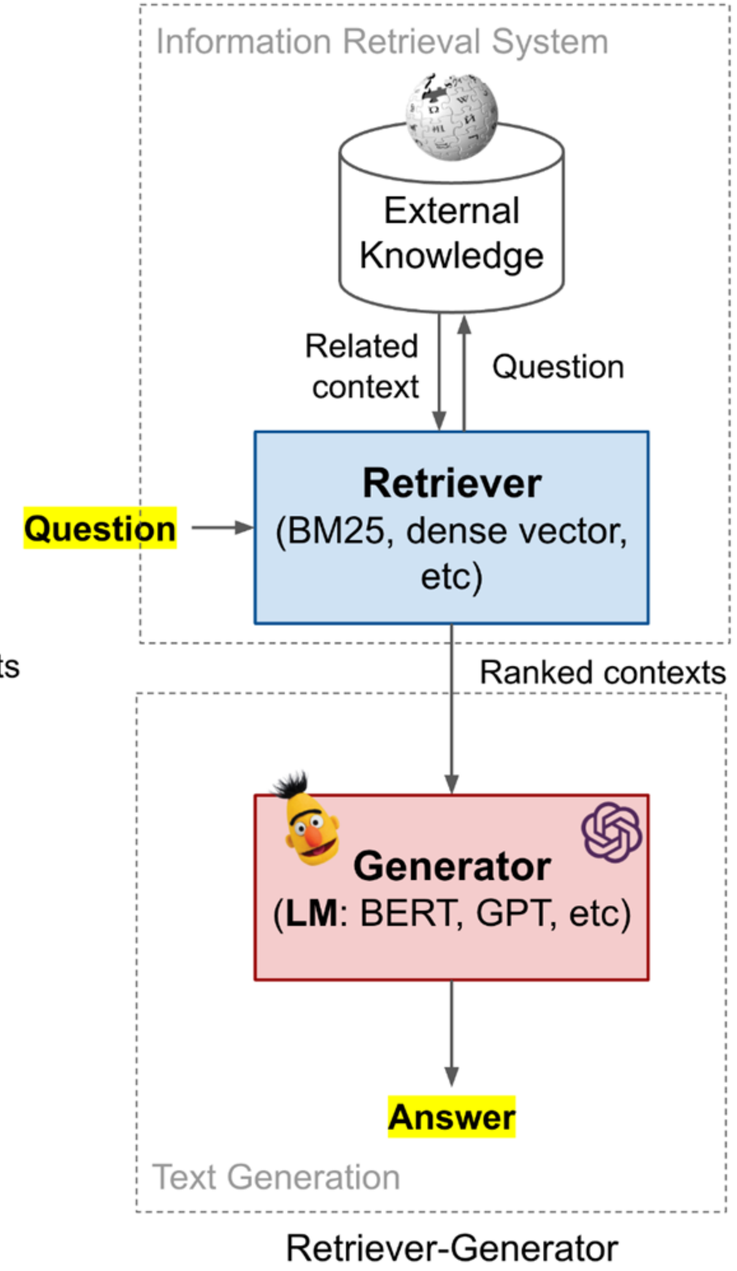
$$L = -\log(P_{\text{start}_{i*}} \cdot P_{\text{end}_{j*}}) = -\log P_{\text{start}_{i*}} - \log P_{\text{end}_{j*}}$$

Goal: Dive into one NLP application: Question Answering

- QA Landscape
- An Overview of Retriever-Reader & Retriever-Generator Architectures for Open-Ended QA
- Dense Retrieval
- Answer Extractor
- Retrieval Augmented Generation (RAG)
- RAG: Overview of Retriever-Generator training options



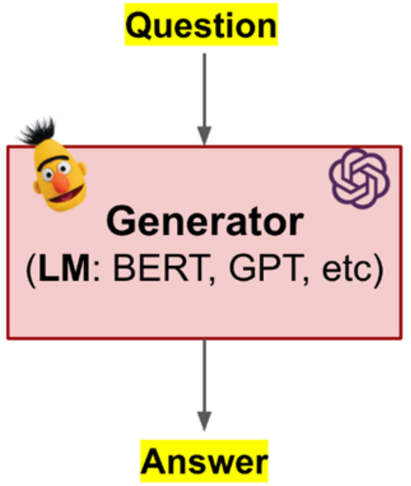
Retriever-Reader



Retriever-Generator

A LM's ***parametric knowledge***: The information that the model has encoded within its parameters/weights during training that it can then use to do tasks for which that knowledge is required

So, why LLMs need retrieval?



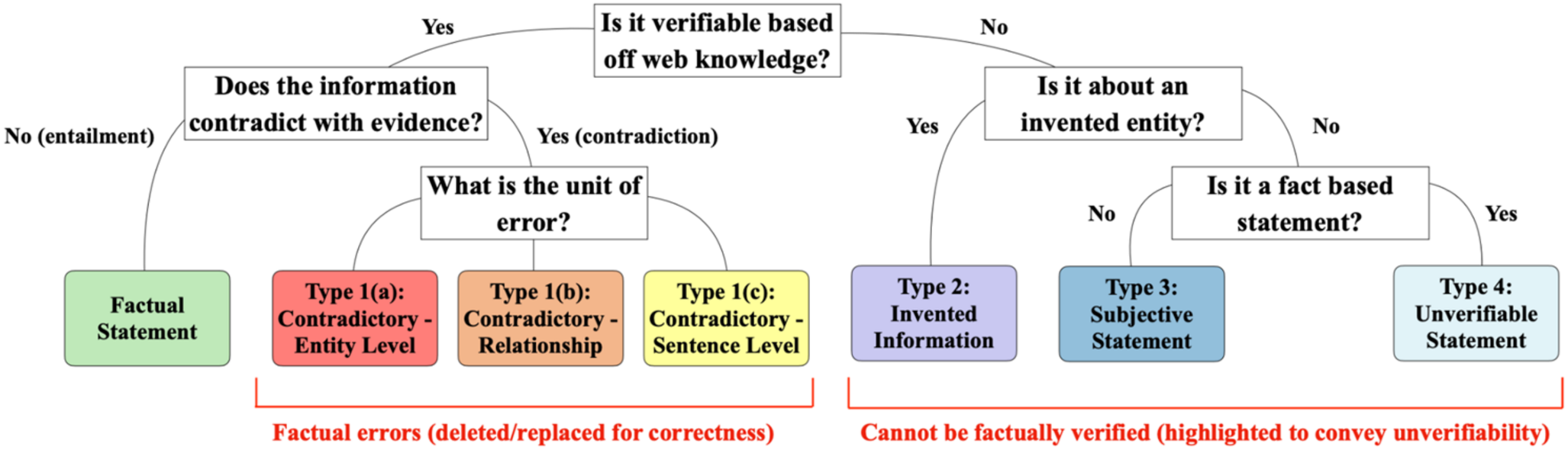
Generator

Source: https://lilianweng.github.io/posts/2020-10-29-odqa/

Hallucination & Factuality

A hallucination is a response that is not faithful to the facts of the world

Fine-grained hallucination taxonomy [Mishra et al., 2024]:



Retrieval Augmented Generation (**RAG**)

1. Use a retriever to identify top-k most relevant documents for the user's prompt
2. Add the retrieved documents to the input
3. Ask the LLM to generate the answer based on this rich context
4. Finetune the answer-generator LLM and/or retrieval

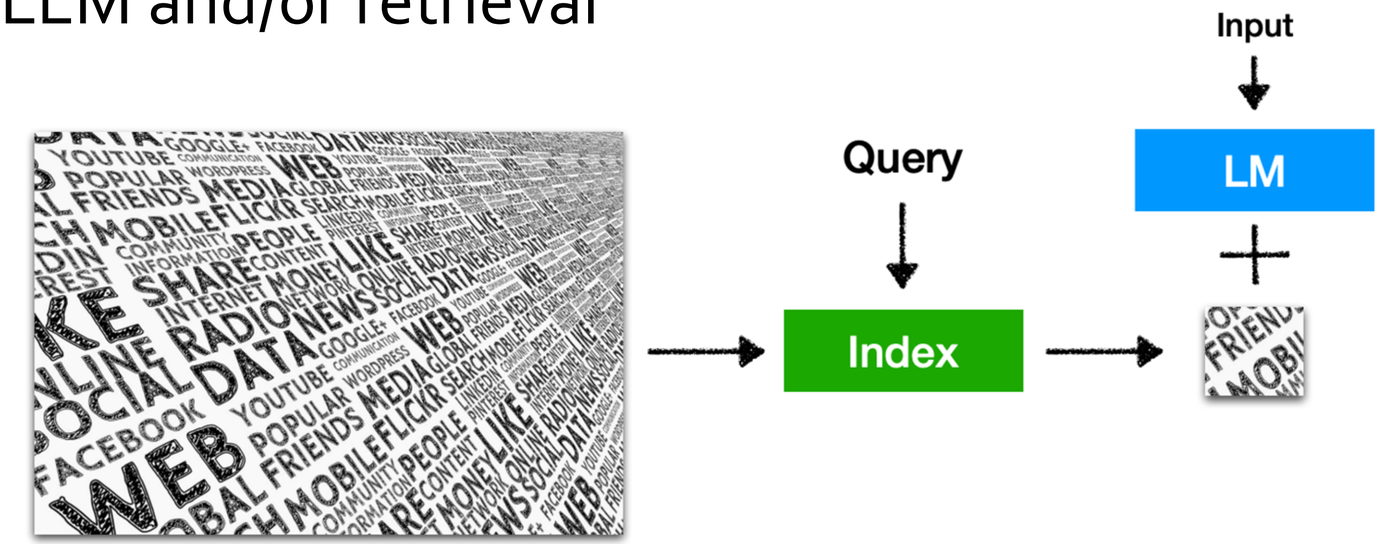


Figure Source: ACL 2023 Tutorial: Retrieval-based Language Models and Applications; Section 2

Reminder: Input Context Length

The attention matrix is quadratic in the maximum sequence length

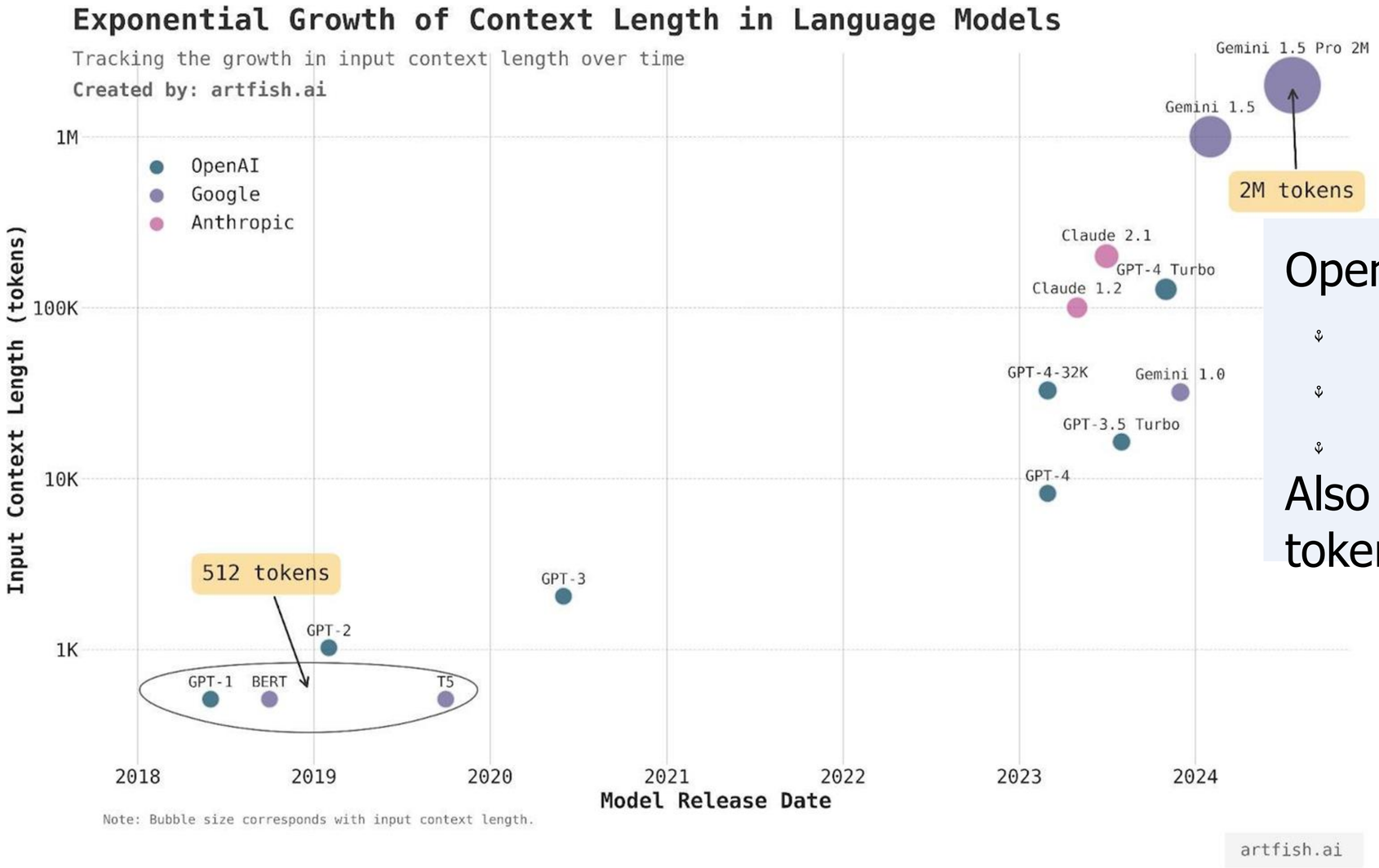
- If the length of an input sequence doubles, the amount of memory required quadruples
- Training an LLM on sequence lengths of 128k will require ~1024x the memory compared to training on sequence lengths of 4k
- GPU memory

Poor generalization due to position encoding:

- RoPE is the current choice (aims to preserve the relative distance between tokens)
- Performance quickly breaks down for sequence lengths significantly longer than the model has seen before [Press et al., 2022]

You can process sequences of arbitrary lengths, but you shouldn't expect a good performance for sequences longer than what's used for pretraining/post-training because of RoPE, & creators of LLMs are prevented from increasing the sequence length drastically due to the GPU memory limits

Input Context Length



Open-weight LLMs:

- LLaMA 3.1
- Qwen 2.5
- Mistral-Large

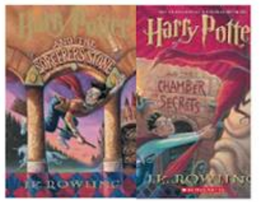
Also fit 128K tokens!

Source: https://www.artfish.ai/p/long-context-llms

Google's Gemini 1.5 can (almost) fit the entire Harry Potter + Lord of the Ring series in its 2 million context window

2M — Gemini 1.5 2M (June 2024)

200K — Claude 2.1 (July 2023)



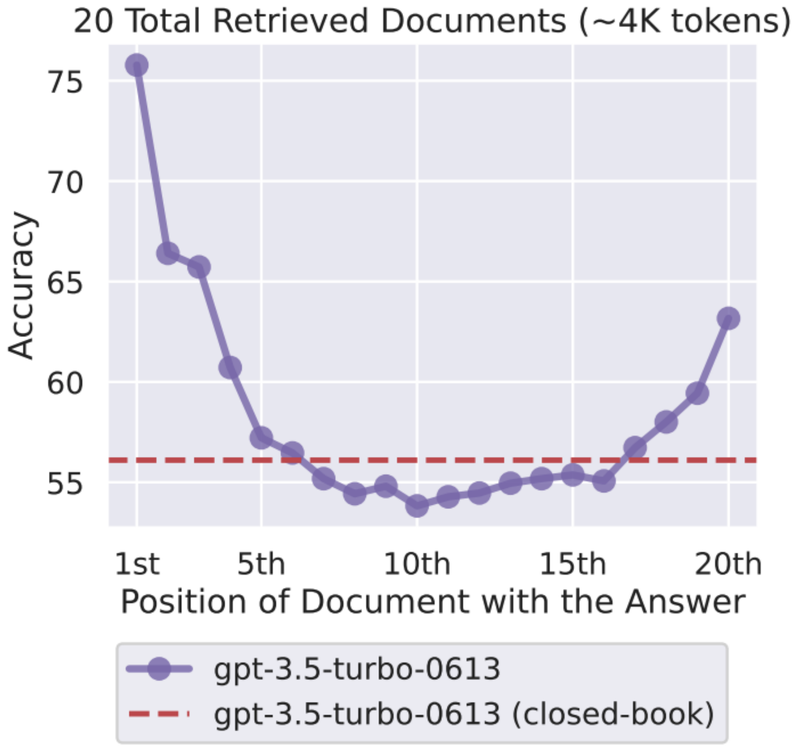
128K — GPT-4 Turbo (March 2023)

GPT-3.5 Turbo (March 2022)

Source: https://www.artfish.ai/p/long-context-llms

Practical Considerations

1. Longer the input context length, the more potentially relevant documents we can squeeze, but:
   - [Liu et al., 2023]: "*models are **better at using relevant information** that occurs **at the very beginning** (primacy bias) **or end of its input context** (recency bias), and performance degrades significantly when models must access and use information located in the middle of its input context*"
1. [Ying et al., 2024]: Conflicts between internal/parametric knowledge and knowledge in a given context
   - Curation of the context is thus important



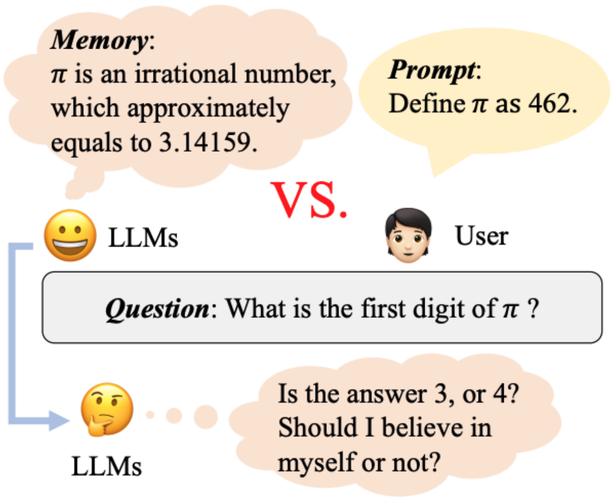


Figure 1: In conflict situation , LLMs may depend on the prompt or intuitively answer based on memory

Goal: Dive into one NLP application: Question Answering

- QA Landscape
- An Overview of Retriever-Reader & Retriever-Generator Architectures for Open-Ended QA
- Dense Retrieval
- Answer Extractor
- Retrieval Augmented Generation (RAG)
- RAG: Overview of Retriever-Generator training options

Retrieval-based LMs: Training

Training challenges:

- External datastore is huge
  ⇒ Expensive to update the index = recompute dense vectors for all documents
- LLMs are large
  ⇒ Expensive to finetune an LLM to generate answers

Option 1 – Independent Training:
Retrieval models and language models are trained independently

Source: ACL 2023 Tutorial: Retrieval-based Language Models and Applications; Section 4

Problem with this training option

We ignore that a retriever and a generator work together

We want to improve the RAG system such that:

- The retriever learns to select passages that make the generator's job easier
- The generator learns to adapt to whatever the retriever gives

$$L = -\log \sum_{d \in \text{Docs}} p_{\text{retriever}}(d|q;\theta_r) p_{\text{generator}}(y|q,d;\theta_g)$$

Millions or billions ⇒ Too slow

Retrieval-based LMs: Training

Option 2 – Sequential Training:
One component is first trained independently & then fixed, the other component is trained with an objective that depends on the first one

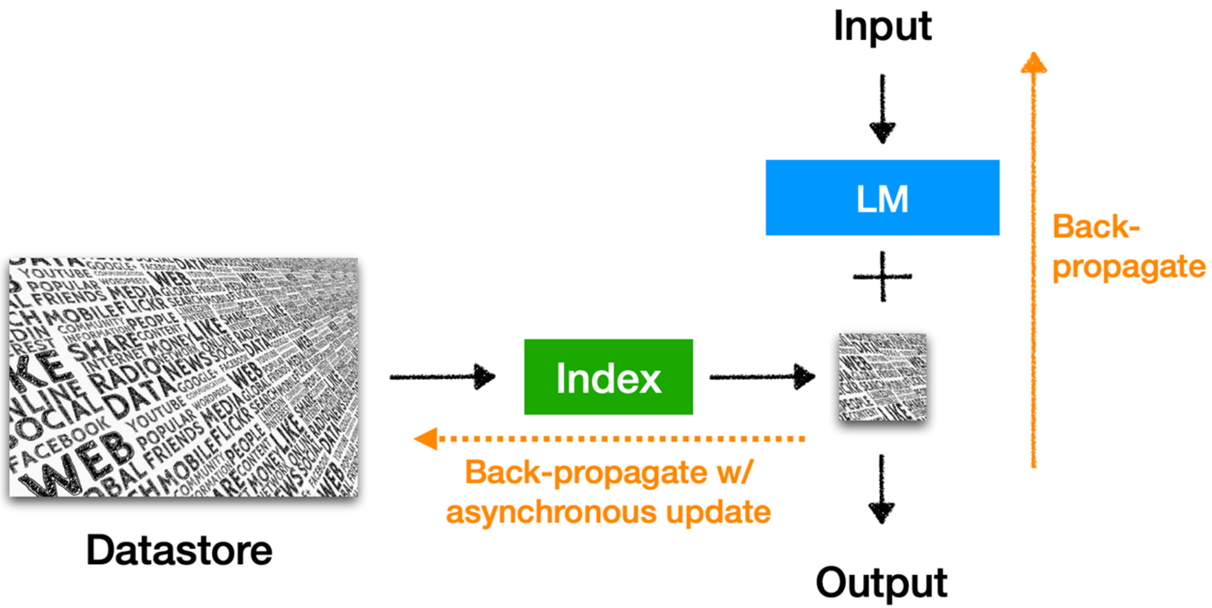
- Retrievers are trained to provide text that helps LMs the most

Source: [ACL 2023 Tutorial: Retrieval-based Language Models and Applications]; Section 4

Retrieval-based LMs: Training

Option 3 – Joint training w/ asynchronous index update:

- Retrieval models and language models are trained jointly
- Allow the index to be "stale"; rebuild the retrieval index every T steps



Source: ACL 2023 Tutorial: Retrieval-based Language Models and Applications; Section 4

Retrieval-based LMs: Training

Option 4 – Joint training w/ in-batch approximation:

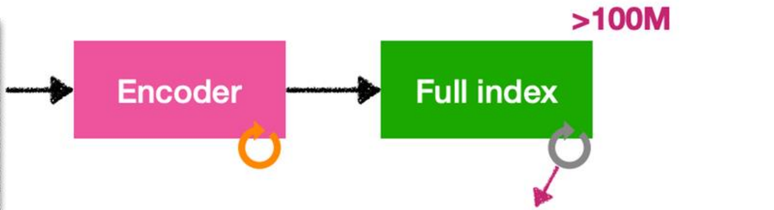
- Retrieval models and language models are trained jointly
- Use "in-batch index" instead of full index
- Treat each other document in the batch as a negative example for a given query

$$L_{\text{retriever}} = -\log p_{\text{retriever}}(d^+|q;\theta_r) = -\log \frac{e^{\text{sim}(q,d^+)}}{\sum_{d' \in \text{Batch}} e^{\text{sim}(q,d')}}$$

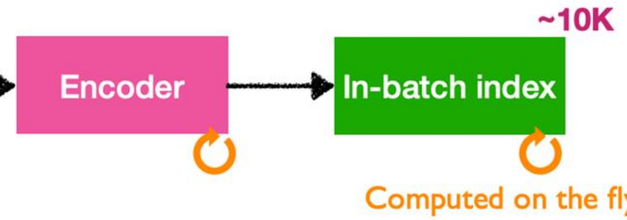
$$L_{\text{generator}} = -\log p_{\text{generator}}(y_{\text{correct}}|q,d^*), \quad \text{where } d^* \sim p_{\text{retriever}}(\cdot|q)$$

$$L = L_{\text{retriever}} + L_{\text{generator}}$$

Full corpus



Training batch



Source: [ACL 2023 Tutorial: Retrieval-based Language Models and Applications]; Section 4

Training method	👍	👎
Independent training (Ram et al 2023; Khandelwal et al 2020) Sequential training (Borgeaud et al 2021; Shi et al 2023)	✳ Easy to implement: off-the-shelf models ✳ Easy to improve: sub-module can be separately improved	✳ Models are not end-to-end trained — suboptimal performance
Joint training: async update (Guu et al 2020; Izacard et al 2022) Joint training: in-batch approx (Zhong et al 2022; Min et al 2023; Rubin and Berant 2023)	✳ End-to-end trained — very good performance!	✳ Training may be complicated (overhead, batching methods, etc) ✳ Train-test discrepancy still remains

How do retrieval-based language models perform on downstream tasks? → **Section 5**!

Source: [ACL 2023 Tutorial: Retrieval-based Language Models and Applications]; Section 4

ODQA “Developer Guide”

https://drive.google.com/file/d/1vh8S13V-LvgdhTlBrcvoXUiPYuzLCSTA/view?usp=sharing