Retrieval-Augmented Generation (RAG)

Slido: https://app.sli.do/event/jsrxDoouM68J5dcrHyBMzJ

Yu Meng
University of Virginia
yumeng5@virginia.edu

Oct 29, 2025

Overview of Course Contents

- Week 1: Logistics & Overview
- Week 2: N-gram Language Models
- Week 3: Word Senses, Semantics & Classic Word Representations
- Week 4: Word Embeddings
- Week 5: Sequence Modeling & Recurrent Neural Networks (RNNs)
- Week 6: Language Modeling with Transformers
- Week 8: Transformer and Pretraining
- Week 9: Large Language Models (LLMs) & In-context Learning
- Week 10: Knowledge in LLMs and Retrieval-Augmented Generation (RAG)
- Week 11: LLM Alignment
- Week 12: Reinforcement Learning for LLM Post-Training
- Week 13: LLM Agents + Course Summary
- Week 15 (after Thanksgiving): Project Presentations

Reminders

- Assignment 4 due next Monday 11/03 11:59pm
- First guest lecture next Monday 11/03 (Meet on Zoom; details to be shared later)

(Recap) Scaling Up Pretraining Data

The Pile: 22 sub-datasets (> 800GB), a common choice for pretraining corpus

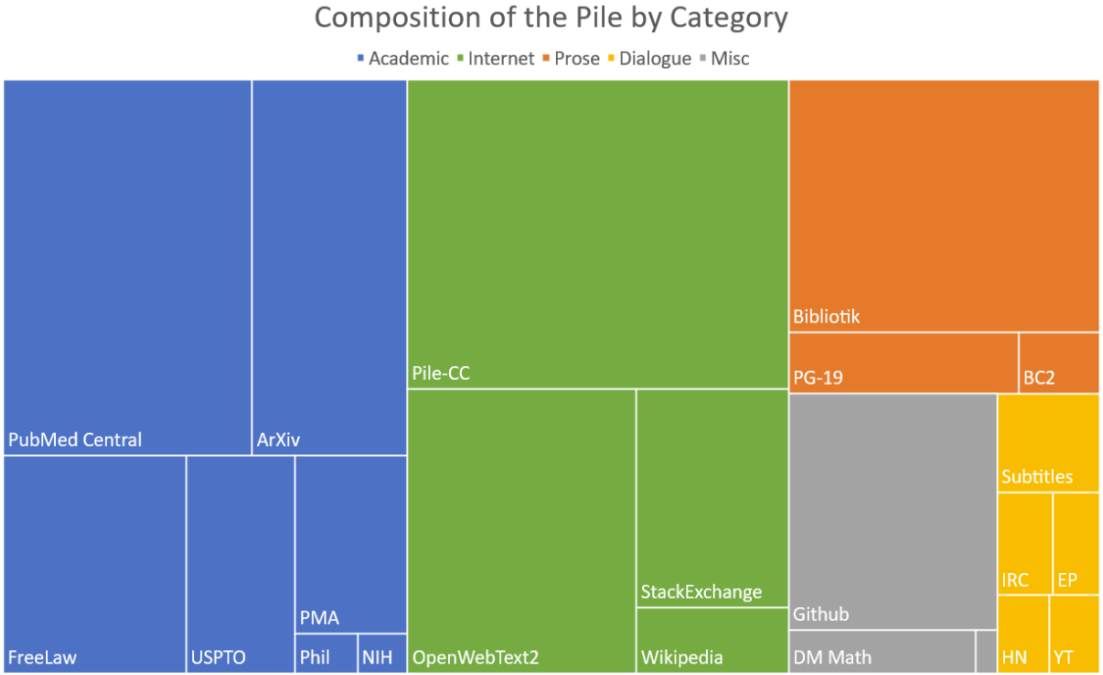
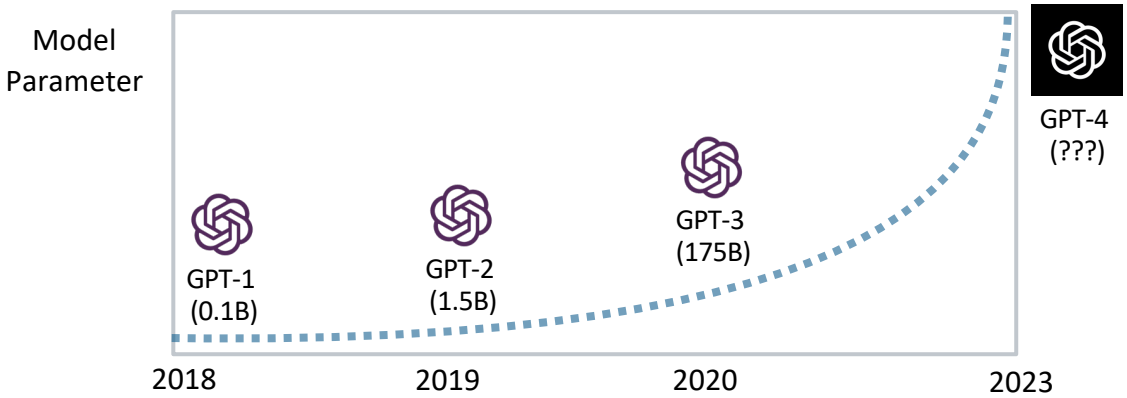


Figure source: https://arxiv.org/pdf/2101.00027

(Recap) Scaling Up Model Sizes

- GPT-1 (2018): 12 layers, 117M parameters, trained in ~1 week
- GPT-2 (2019): 48 layers, 1.5B parameters, trained in ~1 month
- GPT-3 (2020): 96 layers, 175B parameters, trained in several months



Papers: (GPT-1) https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf
(GPT-2) https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
(GPT-3) https://arxiv.org/pdf/2005.14165.pdf

(Recap) Emergent Ability

- Larger models develop **emergent abilities**
  - Skills or capabilities that were not explicitly learned but arise as a result of model capacity
  - Larger models demonstrate surprising abilities in challenging tasks even when they were not explicitly trained for them
- Emergent capabilities typically become noticeable only when the model size reaches a certain threshold (cannot be predicted by small model's performance)

Emergent Abilities of Large Language Models

Jason Wei [1]	jasonwei@google.com
Yi Tay [1]	yitay@google.com
Rishi Bommasani [2]	nlprishi@stanford.edu
Colin Raffel [3]	craffel@gmail.com
Barret Zoph [1]	barretzoph@google.com
Sebastian Borgeaud [4]	sborgeaud@deepmind.com
Dani Yogatama [4]	dyogatama@deepmind.com
Maarten Bosma [1]	bosma@google.com
Denny Zhou [1]	dennyzhou@google.com
Donald Metzler [1]	metzler@google.com
Ed H. Chi [1]	edchi@google.com
Tatsunori Hashimoto [2]	thashim@stanford.edu
Oriol Vinyals [4]	vinyals@deepmind.com
Percy Liang [2]	pliang@stanford.edu
Jeff Dean [1]	jeff@google.com
William Fedus [1]	liamfedus@google.com

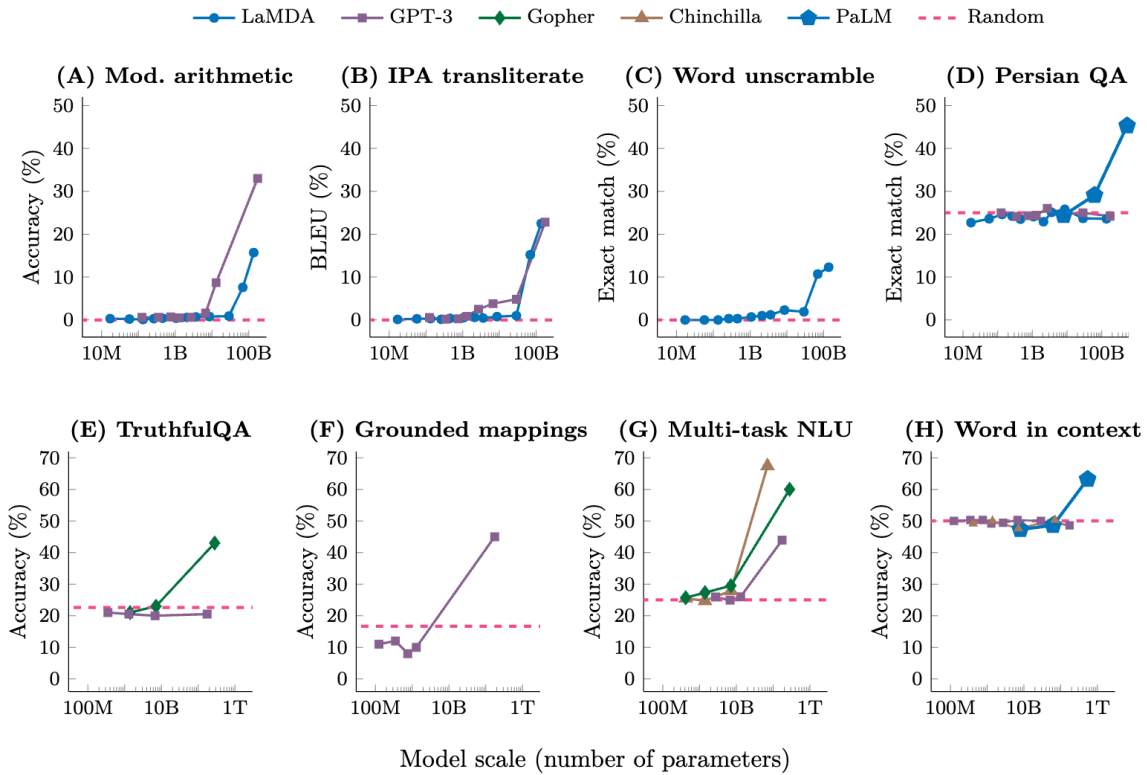
[1] *Google Research* [2] *Stanford University* [3] *UNC Chapel Hill* [4] *DeepMind*

Paper: https://arxiv.org/pdf/2206.07682

(Recap) Experiment Setting

- Consider the **few-shot in-context learning** paradigm
- Consider an ability to be **emergent** when a model has **random** performance until a certain scale, after which performance increases to **well-above random**
- Abilities to test
  - Arithmetic: addition, subtraction, multiplication
  - Transliteration
  - Recover a word from its scrambled letters
  - Persian question answering
  - Question answering (truthfully)
  - Grounded conceptual mappings
  - Multi-task understanding (math, history, law, …)
  - Contextualized semantic understanding

(Recap) Performance vs. Model Scale



Models exhibit random performance until a certain scale, after which performance significantly increases

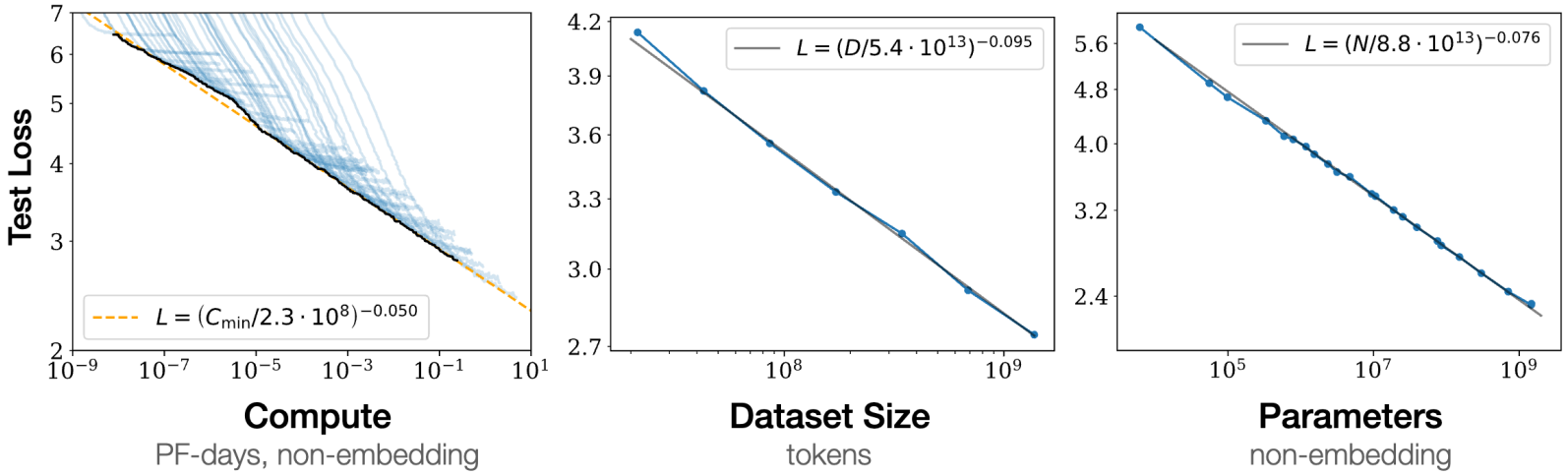
Figure source: https://arxiv.org/pdf/2206.07682

(Recap) Scaling Laws of LLMs

- (Pretrained) LLM performance is mainly determined by 3 factors
  - Model size: the number of parameters
  - Dataset size: the amount of training data
  - Compute: the amount of floating point operations (FLOPs) used for training
- Scaling up LLMs involves scaling up the 3 factors
  - Add more parameters (adding more layers or having more model dimensions or both)
  - Add more data
  - Train for more iterations
- **Scaling laws**: study the correlation between the cross-entropy language modeling loss and the above three factors
- How to optimally allocate a fixed compute budget?

(Recap) Scaling Laws of LLMs

Performance has a power-law relationship with each of the three scale factors (model size, dataset size, compute) when not bottlenecked by the other two



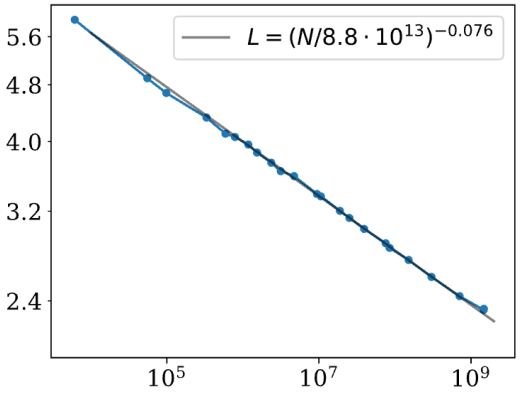
Paper: https://arxiv.org/pdf/2001.08361

(Recap) Scaling Model Parameters

- Language model loss vs. models with a limited number of parameters (N)
  - Only count non-embedding parameters
  - Infinite compute: trained to convergence
  - Infinite dataset: trained with sufficiently large datasets
- Performance depends strongly on scale, weakly on model shape (depth vs. width)

$$\mathcal{L}(N) = \left(\frac{N_c}{N}\right)^{\alpha_N}, \quad \alpha_N \approx 0.076, \quad N_c \approx 8.8 \times 10^{13}$$

Model parameters (non-embedding)

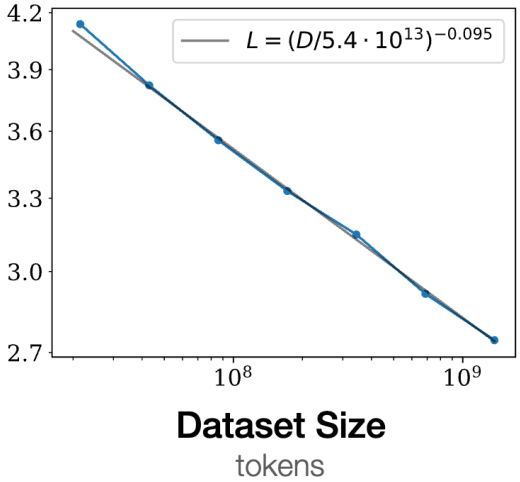


(Recap) Scaling Dataset Size

- Language model loss vs. a limited dataset size (D)
  - Infinite model size: sufficiently large model
  - With appropriate early stopping: avoid overfitting to the training data

$$\mathcal{L}(D) = \left(\frac{D_c}{D}\right)^{\alpha_D}, \quad \alpha_D \approx 0.095, \quad D_c \approx 5.4 \times 10^{13}$$

Dataset size
(# of tokens)

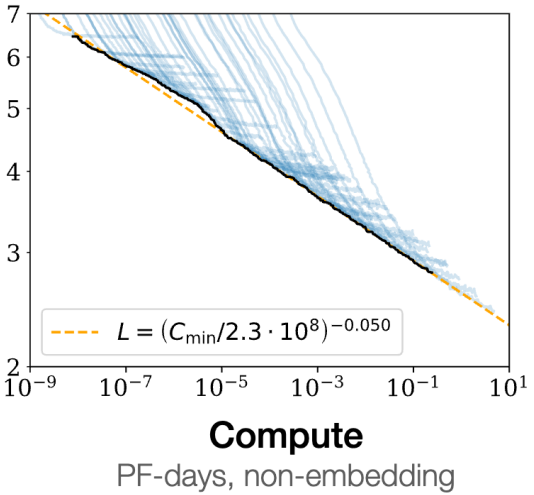


(Recap) Scaling Training Compute

- Language model loss vs. a limited amount of compute (C)
  - Infinite dataset size: sufficiently large training corpus
  - Optimal model size: can effectively learn the data and not excessively compute-consuming

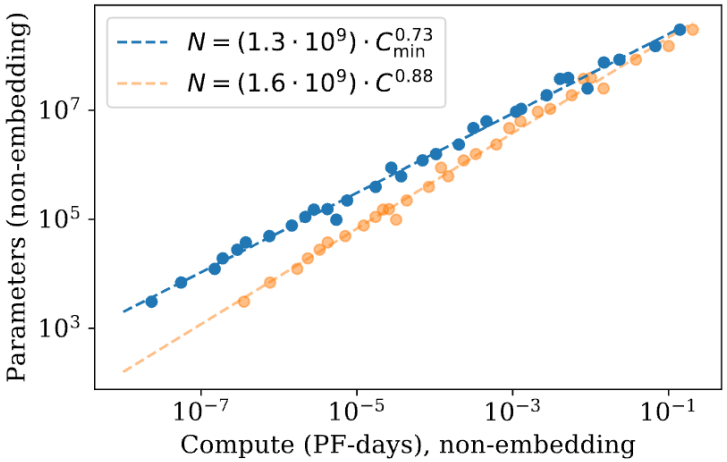
$$\mathcal{L}(C) = \left(\frac{C_c}{C}\right)^{\alpha_C}, \quad \alpha_C \approx 0.050, \quad C_c \approx 3.1 \times 10^8$$

Compute
(# Peta-FLOP days)

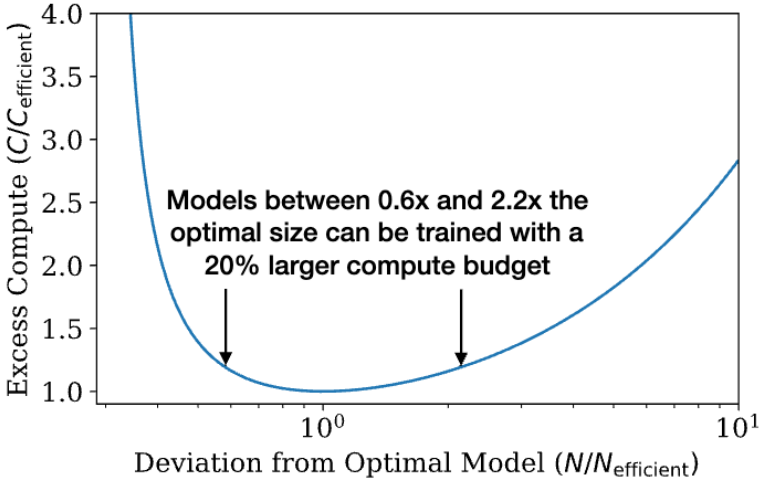


(Recap) Optimal Model Size

- Given a specific amount of training compute C, what's the optimal model size $N(C)$ that leads to minimal language modeling loss?
- $N(C)$ can be fit with a power-law wrt C
- Additional compute needs to be used when model size is suboptimal



$$N(C_{\text{min}}) \propto (C_{\text{min}})^{0.73}$$



(Recap) Introduction to Question Answering

- **Question Answering (QA):** build systems that can automatically answer questions posed by humans in natural language
- Categorization by application domain: closed-domain vs. open-domain QA
- **Closed-domain** QA: answer questions within a specific domain
  - Example: medical, legal, technical fields
  - Models are trained on specialized knowledge to be highly accurate within their domain
- **Open-domain** QA: answer questions from any domain
  - Typically rely on vast (external) knowledge sources like the web or large text corpora
  - Most LLM applications consider open-domain QA settings

(Recap) Introduction to Question Answering

- **Question Answering (QA):** build systems that can automatically answer questions posed by humans in natural language
- Categorization by modeling approach: extractive vs. abstractive QA
- **Extractive** QA: output a span of text extracted directly from a given context
  - A natural language understanding task (reading comprehension)
  - Example: context: "The human brain contains approximately 86 billion neurons" Q: "How many neurons are in the human brain?" A: "86 billion"
  - Can be done with encoder-only LMs (e.g., BERT)
- **Abstractive** QA: synthesize the answer in its own words (rephrasing/summarizing)
  - Example: context: "Albert Einstein published his theory of special relativity which introduced the famous equation $E=mc^2$, which relates energy (E) to mass (m) and the speed of light (c)" Q: "What did Einstein contribute to physics?" A: "Einstein made significant contributions to the theory of special relativity which established the relationship between energy and mass"
  - Need to use a generative LM (e.g., GPT)

(Recap) Introduction to Question Answering

- **Question Answering (QA):** build systems that can automatically answer questions posed by humans in natural language
- Categorization by access to external source: closed-book vs. open-book QA
- **Closed-book** QA: answer questions without access to any external information
  - Accuracy depends heavily on how well the training data covered the relevant information
  - Similar to a human answering a question from memory without looking anything up
- **Open-book** QA: can access external knowledge source to answer the questions
  - Typically using retrieval from reliable external sources that contain
  - Similar to a human answering a question by looking it up in a book or online resource

(Recap) Prompting LMs: Parametric Knowledge

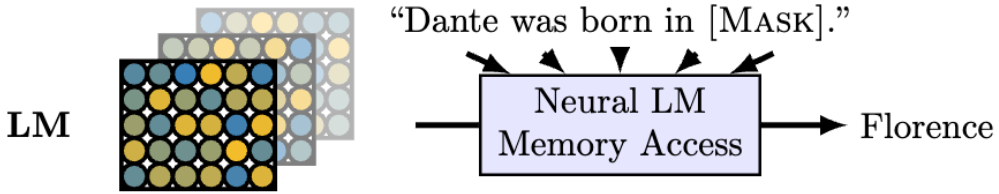
- LMs have learned from a lot of facts in their pretraining data
- LMs can be directly prompted to generate answers to factoid questions (Closed-book QA setting)
- Example:

$P(w|$`Q: Who wrote the book ‘‘The Origin of Species"? A:`$)$ prompt

- Since prompting LLMs only relies on the information stored within the parameters of the model itself, this kind of knowledge is called **parametric knowledge**

(Recap) Language Model as Knowledge Bases

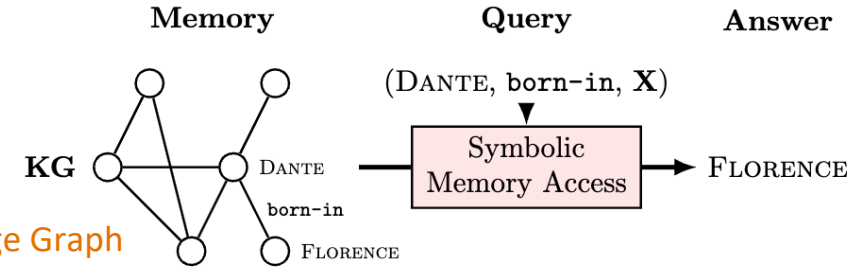
- **Acquisition**: LM's knowledge is derived from the vast amount of pretraining data
- **Access**: information is accessed through natural language prompts
- **Update/maintenance**: re-training/fine-tuning the model with new data
- **Pros**:
  - Handle a wide range of natural language queries with contextual understanding
  - Generalize to unseen queries not seen during training
- **Cons**:
  - May produce incorrect/outdated information
  - Lack interpretability/transparency



Paper: https://arxiv.org/pdf/1909.01066

(Recap) Real Knowledge Bases

- **Acquisition**: manually constructed by human annotators
- **Access**: information is accessed through queries in specific formats
- **Update/maintenance**: adding/modifying/deleting entries (incrementally) by humans
- **Pros**:
  - Precise & verifiable
- **Cons**:
  - Not able to handle natural language
  - Require massive human efforts to construct & maintain



Knowledge Graph

Paper: https://arxiv.org/pdf/1909.01066

Feedforward Parameters in Transformer

- FFN in Transformer is a 2-layer network (one hidden layer, two weight matrices)

$$\mathrm{FFN}(\boldsymbol{x}_i) = \mathrm{ReLU}(\boldsymbol{x}_i \boldsymbol{W}_1)\boldsymbol{W}_2$$

- FFN constitutes ~2/3 of the total parameters of Transformer

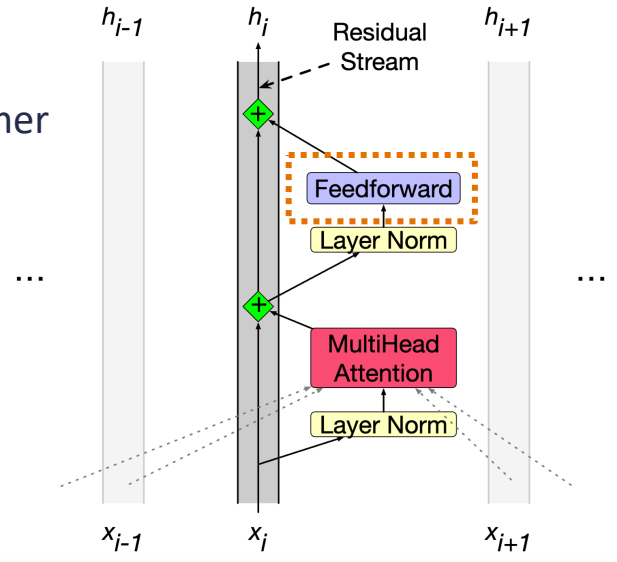


Figure source: https://web.stanford.edu/~jurafsky/slp3/8.pdf

Feedforward Parameters Are Neural Memories

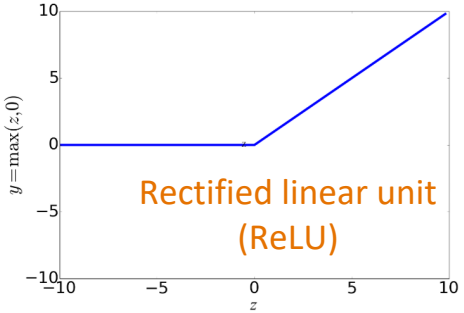
Viewing FFN as key-value memories

$$\mathrm{FFN}(\boldsymbol{x}_i) = \mathrm{ReLU}(\boldsymbol{x}_i \boldsymbol{W}_1)\boldsymbol{W}_2$$

$$\boldsymbol{x}_i \in \mathbb{R}^{d_1}$$

$$\mathrm{FFN}(\boldsymbol{x}_i) = \mathrm{ReLU}(\boldsymbol{x}_i \boldsymbol{K})\boldsymbol{V}$$

$$\boldsymbol{K} \in \mathbb{R}^{d_1 \times d_2} \qquad \boldsymbol{V} \in \mathbb{R}^{d_2 \times d_1}$$



key vectors (column vectors in $\boldsymbol{K}$) act as **pattern detectors** over the input sequence

value vectors (row vectors in $\boldsymbol{V}$) represent **distributions over the output vocabulary**

$$\mathrm{FFN}(\boldsymbol{x}_i) = \sum_{j=1}^{d_2} \mathrm{ReLU}(\boldsymbol{x}_i \cdot \boldsymbol{k}_j)\boldsymbol{v}_j$$

weights of value vectors

Paper: https://arxiv.org/pdf/2012.14913

Memory Keys Correspond to Input Patterns

Each individual key vector corresponds to a specific pattern over the input prefix

Key	Pattern	Example trigger prefixes
$\mathbf{k}^{1}_{449}$	Ends with *"substitutes"* (shallow)	*At the meeting, Elton said that "for artistic reasons there could be no substitutes* *In German service, they were used as substitutes* *Two weeks later, he came off the substitutes*
$\mathbf{k}^{6}_{2546}$	Military, ends with *"base"/"bases"* (shallow + semantic)	*On 1 April the SRSG authorised the SADF to leave their bases* *Aircraft from all four carriers attacked the Australian base* *Bombers flying missions to Rabaul and other Japanese bases*
$\mathbf{k}^{10}_{2997}$	a "part of" relation (semantic)	*In June 2012 she was named as one of the team that competed* *He was also a part of the Indian delegation* *Toy Story is also among the top ten in the BFI list of the 50 films you should*
$\mathbf{k}^{13}_{2989}$	Ends with a time range (semantic)	*Worldwide, most tornadoes occur in the late afternoon, between 3 pm and 7* *Weekend tolls are in effect from 7:00 pm Friday until* *The building is open to the public seven days a week, from 11:00 am to*
$\mathbf{k}^{16}_{1935}$	TV shows (semantic)	*Time shifting viewing added 57 percent to the episode's* *The first season set that the episode was included in was as part of the* *From the original NBC daytime version , archived*

Paper: https://arxiv.org/pdf/2012.14913

Memory Values Correspond to Output Tokens

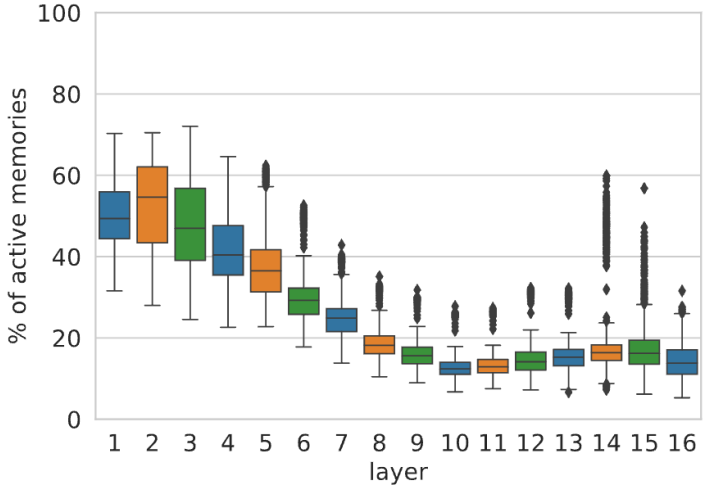
Each value vector (roughly) matches a predicted token distribution

Value	Prediction	Trigger example
$\mathbf{v}_{222}^{15}$	*each*	*But when bees and wasps resemble each*
$\mathbf{v}_{752}^{16}$	*played*	*Her first role was in Vijay Lalwani's psychological thriller Karthik Calling Karthik, where Padukone was cast as the supportive girlfriend of a depressed man (played*
$\mathbf{v}_{2601}^{13}$	*extratropical*	*Most of the winter precipitation is the result of synoptic scale, low pressure weather systems (large scale storms such as extratropical*
$\mathbf{v}_{881}^{15}$	*part*	*Comet served only briefly with the fleet, owing in large part*
$\mathbf{v}_{2070}^{16}$	*line*	*Sailing from Lorient in October 1805 with one ship of the line*
$\mathbf{v}_{3186}^{12}$	*jail*	*On May 11, 2011, four days after scoring 6 touchdowns for the Slaughter, Grady was sentenced to twenty days in jail*

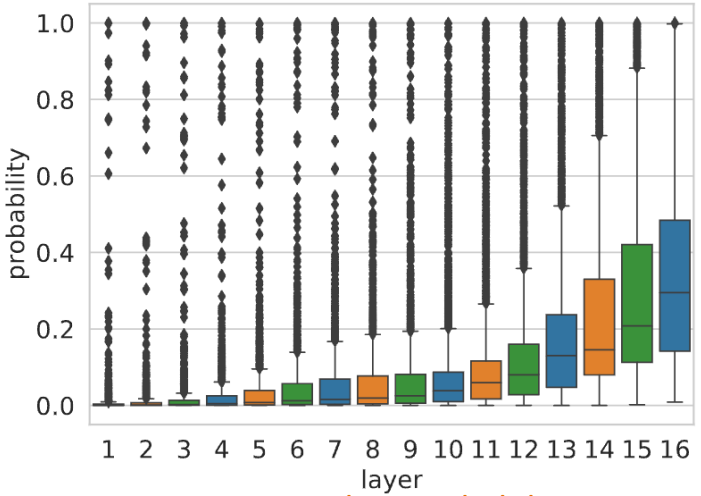
Paper: https://arxiv.org/pdf/2012.14913

Memory Aggregation

- "Active" memories (memory vectors with non-zero coefficients) are typically sparse
- The residual connection sequentially refines token prediction from layer to layer



Fraction of active memory units across layers



Output token probability is gradually refined across layers

Paper: https://arxiv.org/pdf/2012.14913

Further Reading on LLM Parametric Knowledge

- How Much Knowledge Can You Pack Into the Parameters of a Language Model? [Roberts et al., 2020]
- Extracting Training Data from Large Language Models [Carlini et al., 2021]
- Locating and Editing Factual Associations in GPT [Meng et al., 2022]

Agenda

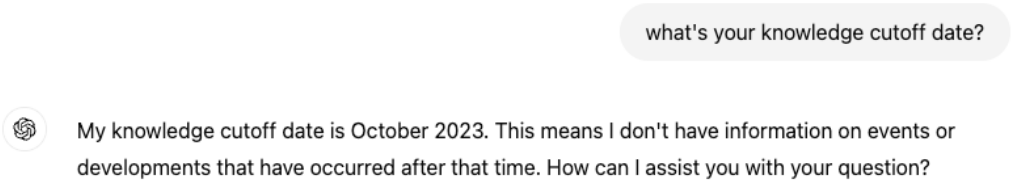
- Hallucination
- Non-parametric Knowledge & Information Retrieval
- Sparse Retrieval (TF-IDF)
- Dense Retrieval
- Evaluation of Retrieval
- Retrieval-Augmented Generation (RAG) for LLMs

Hallucination

- **Hallucination**: LM generates information that is factually incorrect, misleading, or fabricated, even though it may sound plausible or convincing
- Why does hallucination happen?
  - Limited knowledge: LLMs are trained on finite datasets, which don't have access to all possible information; when asked about topics outside their training data, they may generate plausible-sounding but incorrect responses
  - Overgeneralization: LLMs may apply patterns they've learned from one context to another where they don't apply, leading to incorrect conclusions
  - Lack of common sense: While LLMs can process and generate human-like text, they often lack the ability to apply commonsense reasoning to their outputs
  - …

Hallucination Examples

- (**Limited knowledge**) Q: “What were the main features of the iPhone 15 Pro Max?” LLM (trained before 2023): “The iPhone 15 Pro Max features a revolutionary holographic display, quantum computing chip, and telepathic user interface.”
- (**Overgeneralization**) Q: “How do you form the past tense in Japanese?” LLM: “In Japanese, you typically add '-ed' to the end of verbs to form the past tense, just like in English.” (incorrect)
- (**Lack of common sense**) Q: “How many tennis balls can fit in a typical smartphone?” LLM: “Approximately 15-20 tennis balls can fit in a typical smartphone, depending on the model and screen size.”



Knowledge cutoff date of GPT4o

Concerns About Hallucination

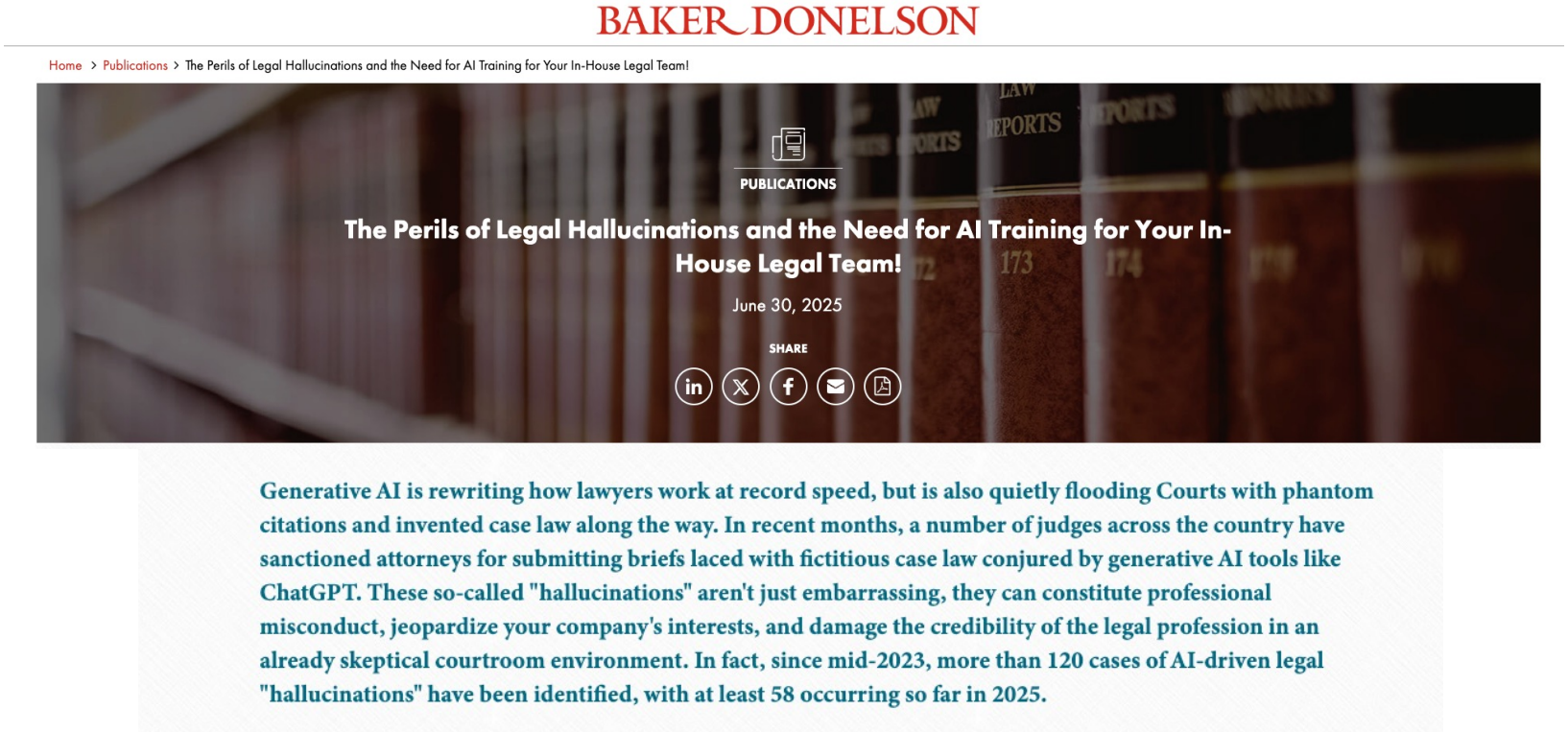


Figure source: https://www.bakerdonelson.com/the-perils-of-legal-hallucinations-and-the-need-for-ai-training-for-your-in-house-legal-team

Further Reading on Hallucination

- LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples [Yao et al., 2023]
- Towards Mitigating Hallucination in Large Language Models via Self-Reflection [Ji et al., 2023]
- Hallucination is Inevitable: An Innate Limitation of Large Language Models [Xu et al., 2024]

Agenda

- Hallucination
- Non-parametric Knowledge & Information Retrieval
- Sparse Retrieval (TF-IDF)
- Dense Retrieval
- Evaluation of Retrieval
- Retrieval-Augmented Generation (RAG) for LLMs

Non-parametric Knowledge

- **Non-parametric knowledge**: (external) information not stored in the model's parameters but can be accessed or retrieved when needed
- Examples:
  - External knowledge bases/graphs
  - Pretraining corpora
  - User-provided documents/passages
- Non-parametric knowledge is typically used to **augment** parametric knowledge (typically via **retrieval**) for more accurate factoid question answering
- Benefits of **non-parametric knowledge**
  - Incorporate more information without increasing model size
  - Easier updates and modifications to the knowledge base
  - Improve model interpretability

Overview: Retrieval-Augmented Generation

- Use a **retriever** to obtain relevant documents to the query from an external text collection
- Use LLMs to generate answers given the documents and a prompt

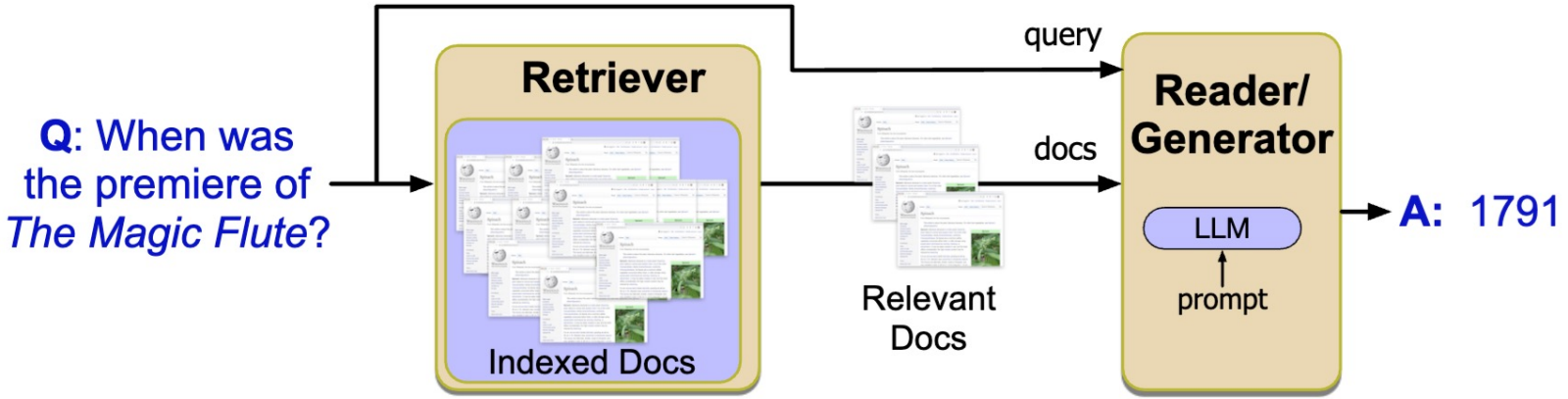


Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

Overview: Information Retrieval (IR)

- **Information retrieval** (**IR**): finding relevant information from a large collection of unstructured data (e.g., documents, web pages) in response to a user query
- **Query**: user-provided input (e.g., keywords or phrases), describing the information they are seeking
- **Documents/corpus**: the data collection that the system searches through
- **Ranking**: sort the search results by relevance based on specific metrics (e.g., keyword matching, semantic similarity)
- Web search engines (e.g., Google, Bing) are IR systems

Sparse vs. Dense Retrieval

- **Sparse** retrieval: based on traditional IR techniques where the representations of documents and queries are sparse (most vector values are zero)
  - Example: TF-IDF
  - Pros: simple and interpretable
  - Cons: lack semantic understanding
- **Dense** retrieval: encode documents and queries into dense vectors (embeddings) using deep neural networks
  - Example: BERT-based encoding methods
  - Pros: semantic & contextualized understanding
  - Cons: computationally more expensive and less interpretable

Agenda

- Hallucination
- Non-parametric Knowledge & Information Retrieval
- Sparse Retrieval (TF-IDF)
- Dense Retrieval
- Evaluation of Retrieval
- Retrieval-Augmented Generation (RAG) for LLMs

TF-IDF Weighting

- Introduced in week 3's lectures $\text{TF-IDF}(w, d) = \text{TF}(w, d) \times \text{IDF}(w)$
- Main idea: represent a document with frequent & distinctive words

TF-IDF weighted

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.246	0	0.454	0.520
good	0	0	0	0
fool	0.030	0.033	0.0012	0.0019
wit	0.085	0.081	0.048	0.054

$\cos(\boldsymbol{v}_{d_2}, \boldsymbol{v}_{d_3}) = 0.10 \quad \cos(\boldsymbol{v}_{d_3}, \boldsymbol{v}_{d_4}) = 0.99$

Raw counts

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

$\cos(\boldsymbol{v}_{d_2}, \boldsymbol{v}_{d_3}) = 0.81 \quad \cos(\boldsymbol{v}_{d_3}, \boldsymbol{v}_{d_4}) = 0.99$

Term Frequency (TF)

- A word appearing 100 times in a document doesn't make it 100 times more likely to be relevant to the meaning of the document
- Instead of using the raw counts, we squash the counts with log scale

$$\mathrm{TF}(w,d) = \begin{cases} 1 + \log_{10} \mathrm{count}(w,d) & \mathrm{count}(w,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Inverse Document Frequency (IDF)

- We want to emphasize discriminative words (with low DF)
- Inverse document frequency (IDF): total number of documents (N) divided by DF, in log scale

$$\text{IDF}(w) = \log_{10}\left(\frac{N}{\text{DF}(w)}\right)$$

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

DF & IDF statistics in the Shakespeare corpus (37 documents)

TF-IDF for Sparse Retrieval

- Score document-query semantic similarity by cosine similarity

$$\cos(\boldsymbol{q}, \boldsymbol{d}) = \frac{\boldsymbol{q} \cdot \boldsymbol{d}}{|\boldsymbol{q}||\boldsymbol{d}|}$$

- Both document and query vectors use TF-IDF weighting
- Can also adopt other weighting schemes (e.g., BM25)

Example: TF-IDF for Sparse Retrieval

- Example query and mini-corpus:

Query:	sweet love
Doc 1:	Sweet sweet nurse! Love?
Doc 2:	Sweet sorrow
Doc 3:	How sweet is love?
Doc 4:	Nurse!

- Query & document vectors:

	Query					
word	**cnt**	**tf**	**df**	**idf**	**tf-idf**	**n'lized** = tf-idf/$\|q\|$
sweet	1	1	3	0.125	0.125	0.383
nurse	0	0	2	0.301	0	0
love	1	1	2	0.301	0.301	0.924
how	0	0	1	0.602	0	0
sorrow	0	0	1	0.602	0	0
is	0	0	1	0.602	0	0

	Document 1			
word	**cnt**	**tf**	**tf-idf**	**n'lized**
sweet	2	1.301	0.163	0.357
nurse	1	1.000	0.301	0.661
love	1	1.000	0.301	0.661
how	0	0	0	0
sorrow	0	0	0	0
is	0	0	0	0

$$\cos(\boldsymbol{q}, \boldsymbol{d}_1) = 0.747$$

	Document 2			
cnt	**tf**	**tf-idf**	**n'lized**
1	1.000	0.125	0.203
0	0	0	0
0	0	0	0
0	0	0	0
1	1.000	0.602	0.979
0	0	0	0

$$\cos(\boldsymbol{q}, \boldsymbol{d}_2) = 0.078$$

Agenda

- Hallucination
- Non-parametric Knowledge & Information Retrieval
- Sparse Retrieval (TF-IDF)
- Dense Retrieval
- Evaluation of Retrieval
- Retrieval-Augmented Generation (RAG) for LLMs

Dense Retrieval

- Motivation: sparse retrieval (e.g., TF-IDF) relies on the exact overlap of words between the query and document without considering semantic similarity
- Solution: use a language model to obtain (dense) distributed representations of query and document
- The retriever language model is typically a small text encoder model (e.g., BERT)
  - Retrieval is a natural language understanding task
  - Encoder-only models are more efficient than LLMs for this purpose
- Both query and document representations are computed by text encoders

Dense Retrieval: Cross-encoder

- Process query-document pairs together
- Relevance score produced directly by the model output
- (+) Capture intricate interactions between the query and the document
- (-) Not scalable to large retrieval corpus
- Good for small document sets

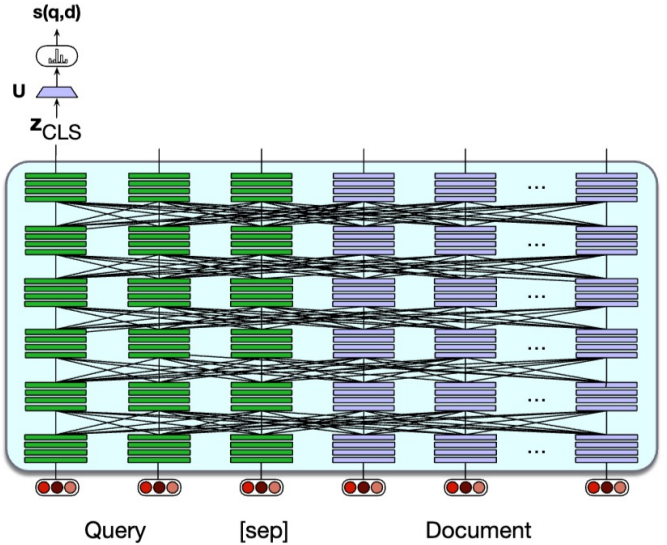


Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

Dense Retrieval: Bi-encoder

- Independently encode the query and the document using two separate (but often identical) encoder models
- Use cosine similarity between the query and document vectors as relevance score
- (+) Document vectors can be precomputed
- (-) Cannot capture query-document interactions
- Common choice for large-scale retrieval

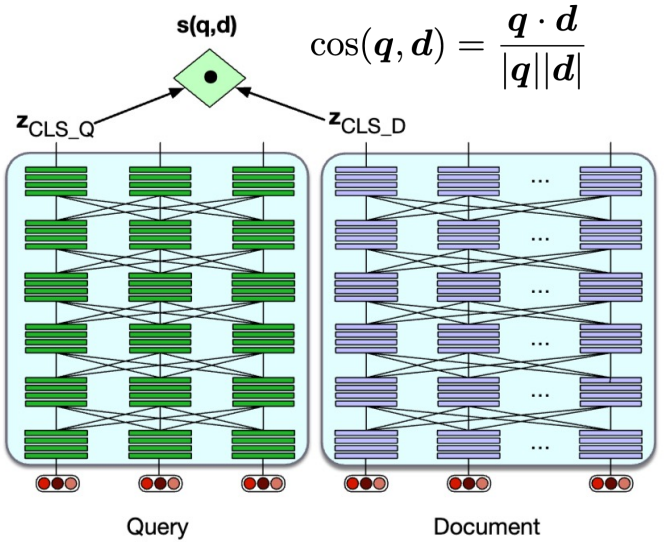


Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

Agenda

- Hallucination
- Non-parametric Knowledge & Information Retrieval
- Sparse Retrieval (TF-IDF)
- Dense Retrieval
- Evaluation of Retrieval
- Retrieval-Augmented Generation (RAG) for LLMs

Evaluation of IR Systems

- Assume that each document returned by the IR system is either **relevant** to our purposes or **not relevant**
- Given a query, assume the system returns a set of ranked documents $\boldsymbol{T}$
  - A subset $\boldsymbol{R}$ of these are relevant (The remaining $\boldsymbol{N} = \boldsymbol{T} - \boldsymbol{R}$ is irrelevant)
  - There are $\boldsymbol{U}$ documents in the entire retrieval collection that are relevant to this query
- **Precision:** the fraction of the returned documents that are relevant

$$\text{Precision} = \frac{|R|}{|T|}$$

- **Recall:** the fraction of all relevant documents that are returned

$$\text{Recall} = \frac{|R|}{|U|}$$

Precision & Recall @ k

- We hope to build a retrieval system that ranks the relevant documents higher
- Use precision & recall @ k (among the top-k items in the ranked list) to reflect this

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

Assume there are 9 total relevant documents in the retrieval corpus

Figure source: https://web.stanford.edu/~jurafsky/slp3/14.pdf

Average Precision

Average precision (AP): mean of the precision values at the points in the ranked list where a relevant document is retrieved

Indicator function of whether the document is relevant

$$\text{AP} = \frac{1}{|R|} \sum_{k=1}^{|T|} (\text{Precision@}k \times \mathbb{1}(d_k \text{ is relevant}))$$

Rank	Judgment	Precision_{Rank}	Recall_{Rank}
1	R	1.0	.11
2	N	.50	.11
3	R	.66	.22
4	N	.50	.22
5	R	.60	.33
6	R	.66	.44
7	N	.57	.44
8	R	.63	.55
9	N	.55	.55
10	N	.50	.55

Agenda

- Hallucination
- Non-parametric Knowledge & Information Retrieval
- Sparse Retrieval (TF-IDF)
- Dense Retrieval
- Evaluation of Retrieval
- Retrieval-Augmented Generation (RAG) for LLMs

RAG for LLMs

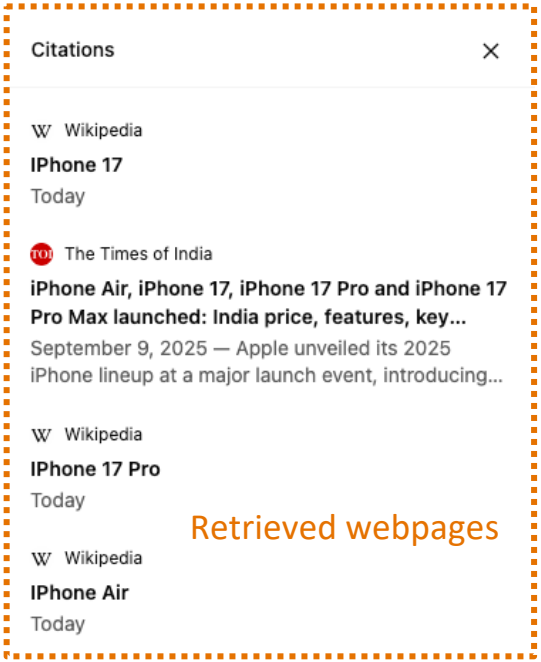
RAG is a common technique to enhance factuality of LLM generation & mitigate hallucination

Time-sensitive query triggers RAG for ChatGPT

What is the latest iPhone model?

Searching for latest iPhone model Apple

Retriever performs web search



Retrieved webpages

Screenshot taken: https://chatgpt.com/

RAG vs. Direct Prompting

- Prompting relies on LM's parametric knowledge to directly answer the question:

$P(w|$ `Q: Who wrote the book ‘‘The Origin of Species"? A:` $)$ prompt

- RAG prepends the set of retrieved passages to the question

Schematic of a RAG Prompt

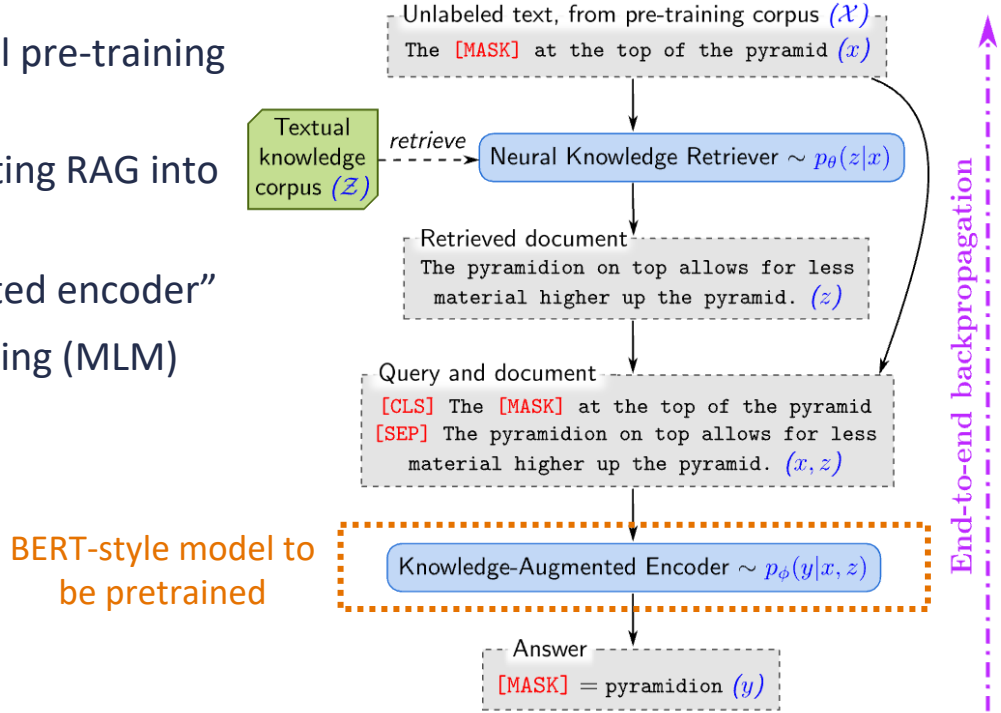
```
retrieved passage 1

retrieved passage 2

...

retrieved passage n
```

Returned by the retriever

```
Based on these texts, answer this question: Q: Who wrote
the book ‘‘The Origin of Species"? A:
```

RAG in Pretraining

- Retrieval-Augmented Language Model pre-training (REALM)
- The first paper that studies incorporating RAG into encoder pretraining (BERT style)
- Main model is a "knowledge-augmented encoder"
- Pretrain with masked language modeling (MLM) loss conditioned on retrieved content



Paper: https://arxiv.org/pdf/2002.08909

RAG for Text Generation

- The first paper that studies RAG for text generation
- Both the retriever (an encoder-only LM) and the generator (an LLM) are trained

Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Patrick Lewis[†‡], Ethan Perez[⋆],

Aleksandra Piktus[†], Fabio Petroni[†], Vladimir Karpukhin[†], Naman Goyal[†], Heinrich Küttler[†],

Mike Lewis[†], Wen-tau Yih[†], Tim Rocktäschel[†‡], Sebastian Riedel[†‡], Douwe Kiela[†]

Paper: https://arxiv.org/pdf/2005.11401

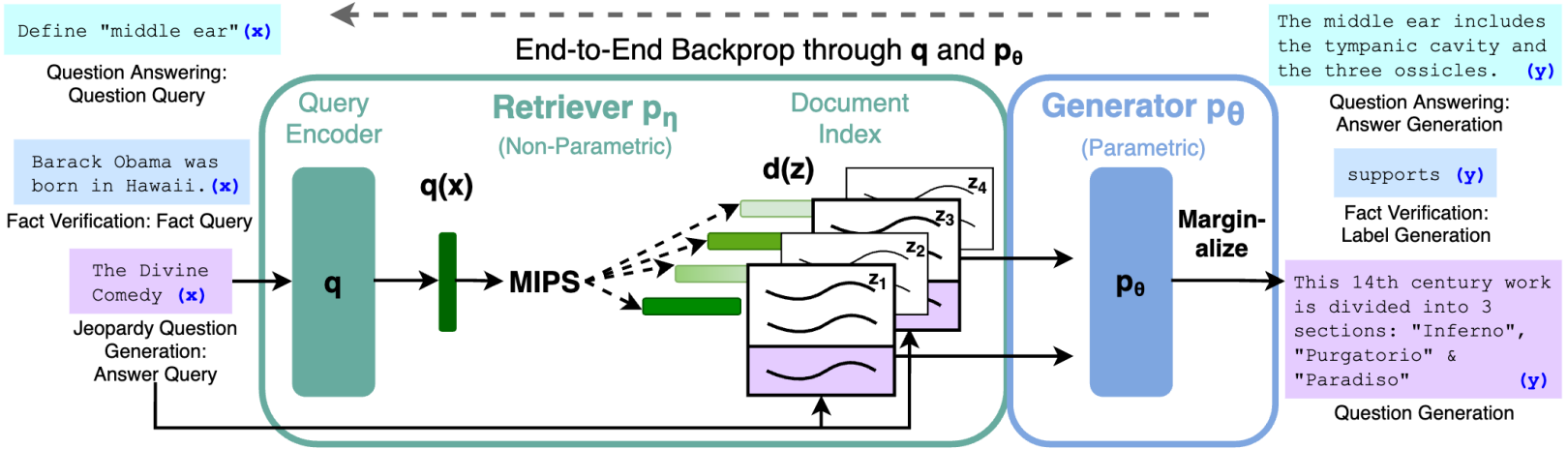
A Latent Variable Model

The retrieved documents are treated as latent variables (z) for generation

$$p(\boldsymbol{y}|\boldsymbol{x}) = \sum_{\boldsymbol{z}\in\mathcal{D}} p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{z})$$

Retrieve document (z) based on query (x)

Generate answer (y) based on retrieved docs (z) and query (x)



RAG-Sequence Model

- Use the same retrieved document to generate the complete sequence
- Treat the retrieved document as a single latent variable
- Marginalize to get the generation probability $p(\boldsymbol{y}|\boldsymbol{x})$ via a top-K approximation

$$p_{\text{RAG-sequence}}(\boldsymbol{y}|\boldsymbol{x}) \approx \sum_{\boldsymbol{z} \in \text{top-K}(p(\cdot|\boldsymbol{x}))} p_\eta(\boldsymbol{z}|\boldsymbol{x}) p_\theta(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{z}) = \sum_{\boldsymbol{z} \in \text{top-K}(p(\cdot|\boldsymbol{x}))} p_\eta(\boldsymbol{z}|\boldsymbol{x}) \prod_{i=1}^{N} p_\theta(y_i|\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{<i})$$

Top-K approximation
(only consider the top-K retrieved docs)

The same retrieved doc (z) is used to generate all tokens in the sequence

RAG-Token Model

- Can use different retrieved documents to generate different tokens in a sequence
- Marginalization is performed for each generated token (rather than at sequence level)

$$p_{\text{RAG-token}}(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i=1}^{N} p_\theta(y_i|\boldsymbol{x}, \boldsymbol{y}_{<i}) \approx \prod_{i=1}^{N} \sum_{\boldsymbol{z} \in \text{top-K}(p(\cdot|\boldsymbol{x}, \boldsymbol{y}_{<i}))} p_\eta(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y}_{<i}) p_\theta(y_i|\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{y}_{<i})$$

Different retrieved doc (z) can be used to generate different tokens in the sequence

RAG-Sequence & RAG-Token Results

Evaluation results on open-domain QA tasks:

- Natural Questions (NQ)
- TriviaQA (TQA)
- WebQuestions (WQ)
- CuratedTrec (CT)

	Model	NQ	TQA	WQ	CT
Closed Book	T5-11B [52]	34.5	- /50.1	37.4	-
	T5-11B+SSM[52]	36.6	- /60.5	44.7	-
Open Book	REALM [20]	40.4	- / -	40.7	46.8
	DPR [26]	41.5	**57.9**/ -	41.1	50.6
	RAG-Token	44.1	55.2/66.1	**45.5**	50.0
	RAG-Seq.	**44.5**	56.8/**68.0**	45.2	**52.2**

Further Reading on RAG

- Generalization through Memorization: Nearest Neighbor Language Models [Khandelwal et al., 2019]
- Active Retrieval Augmented Generation [Jiang et al., 2023]
- Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection [Asai et al., 2023]
- InstructRAG: Instructing Retrieval-Augmented Generation via Self-Synthesized Rationales [Wei et al., 2024]

Thank You!

Yu Meng
University of Virginia
yumeng5@virginia.edu