Natural Language Processing with Deep Learning

Lecture 11 — Retrieval augmented generation

Prof. Dr. Ivan Habernal

January 15, 2026

www.trusthlt.org
Trustworthy Human Language Technologies Group (TrustHLT)
Ruhr University Bochum & Research Center Trustworthy Data Science and Security

Motivation[1]

Important function of large language models is to fill human information needs

For example factoid questions in short texts like the following:

```
Where is the Louvre Museum located?
```

To get an LLM to answer these questions, we can just prompt it!

D. Jurafsky and J. H. Martin (2026). **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models.** 3rd. Online manuscript released January 6, 2026

[1]This lecture is heavily based on the excellent book by Jurafsky and Martin (2026)

Prompting LLM answers factoid questions

For example a pretrained LLM that has been instruction-tuned on answering questions could directly answer the question

```
Where is the Louvre Museum located?
```

by performing conditional generation given this prefix, and take the response as the answer.

This works because

Prompting LLM answers factoid questions

For example a pretrained LLM that has been instruction-tuned on answering questions could directly answer the question

```
Where is the Louvre Museum located?
```

by performing conditional generation given this prefix, and take the response as the answer.

This works because

- LLMs have processed a lot of facts in their pretraining data, including the location of the Louvre, and have encoded this information in their parameters

K. Meng, D. Bau, A. J. Andonian, and Y. Belinkov (2022). **"Locating and editing factual associations in GPT".** In: *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. Curran Associates, Inc.

Prompting LLM answers factoid questions

This works because

- Factual knowledge of this type seems to be stored in the connections in the very large feedforward layers of transformers

But: knowledge stored in the feedforward weights of the LLM leads to a number of problems with prompting as a method for correctly generating factual texts or answers

K. Meng, D. Bau, A. J. Andonian, and Y. Belinkov (2022). **"Locating and editing factual associations in GPT".** In: *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*. Ed. by A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho. Curran Associates, Inc.

Problem #1 of simple prompting: LLMs hallucinate

Hallucination is a response that is not faithful to the facts of the world

LMs sometimes give incorrect factual responses even when the correct facts are stored in the parameters

This seems to be caused by the feedforward layers failing to recall the knowledge stored in their parameters (Jiang, B. Qi, Hong, Fu, Cheng, F. Meng, M. Yu, B. Zhou, and J. Zhou, 2024)

C. Jiang, B. Qi, X. Hong, D. Fu, Y. Cheng, F. Meng, M. Yu, B. Zhou, and J. Zhou (2024). **"On Large Language Models' Hallucination with Regard to Known Facts".** In: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Mexico City, Mexico: Association for Computational Linguistics, pp. 1041–1053

Problem #2 of simple prompting: Proprietary data

Answer from pre-trained parameters does not allow us to ask questions about proprietary data

Problem #2 of simple prompting: Proprietary data

Answer from pre-trained parameters does not allow us to ask questions about proprietary data

- proprietary data like personal email
- healthcare application: LLM to medical records
- company may have internal documents that contain answers for customer service or internal use
- legal firms need to ask questions about legal discovery from proprietary documents

Problem #2 of simple prompting: Proprietary data

Answer from pre-trained parameters does not allow us to ask questions about proprietary data

- proprietary data like personal email
- healthcare application: LLM to medical records
- company may have internal documents that contain answers for customer service or internal use
- legal firms need to ask questions about legal discovery from proprietary documents

None of this data (hopefully) was in the large web-based corpora that large language models are pre-trained on

Problem #3 of simple prompting: Static knowledge

LLMs were pretrained once, at a particular time

LLMs **cannot** talk about about rapidly changing information (like something that happened last week) since they won't have up-to-date information from after their release data

One solution to these problems: RAG

Give a language model external sources of knowledge, use those documents in answering questions

This method is called retrieval-augmented generation (RAG)

1. we use information retrieval (IR) techniques to retrieve documents that are likely to have information that might help answer the question
2. then we use LLM to generate an answer given these documents

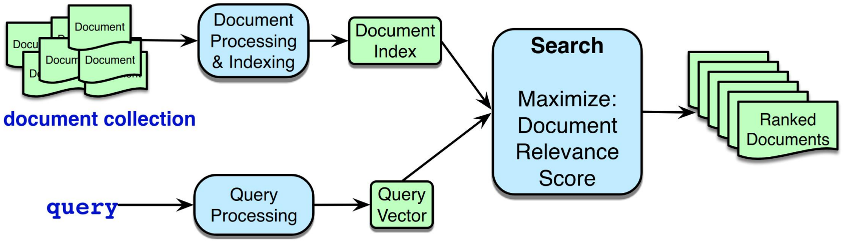
Information retrieval

Information retrieval

- User has an information need
- And has some collection of documents
- User wants to find a relevant document which satisfies their needs

Typical examples: Web search, RAG

In most cases we do ranked retrieval



The retriever returns top-k documents

These are ranked

We can show the user these, or some subset

Document Relevance Score

Goal is to assign a score to each document for whether it meets the user's information need

Instead, we just approximate this by the textual similarity between the query and the document

Two architectures

Sparse retrieval

- represent query and doc as vectors of word counts
- weighted by tf-idf, BM25

Dense retrieval

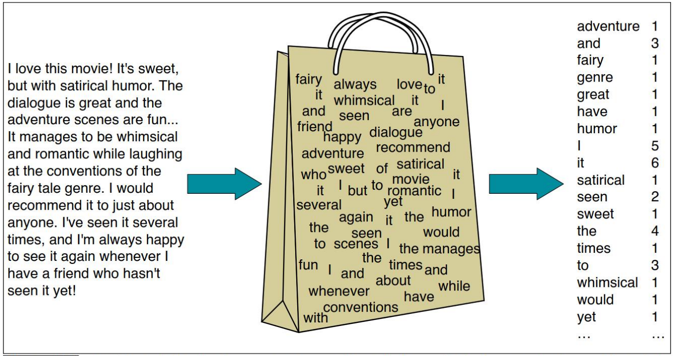
- Use LLM to represent query and doc as embeddings In both cases, similarity is dot product or cosine between query and document representations

Information retrieval

Sparse retrieval: the vector model of IR

The vector space model of IR

Represent a document as a vector of counts of the words it contains: Bag-of-words representation

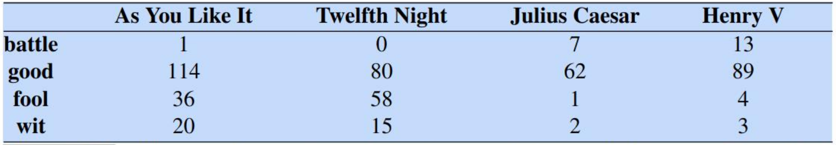


I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

adventure	1
and	3
fairy	1
genre	1
great	1
have	1
humor	1
I	5
it	6
satirical	1
seen	2
sweet	1
the	4
times	1
to	3
whimsical	1
would	1
yet	1
...	...

Figure 11.2 Intuition of the classic vector space model applied to a single document. The position of the words is ignored (the *bag-of-words* assumption) and we make use of the frequency of each word.

Term-document matrix

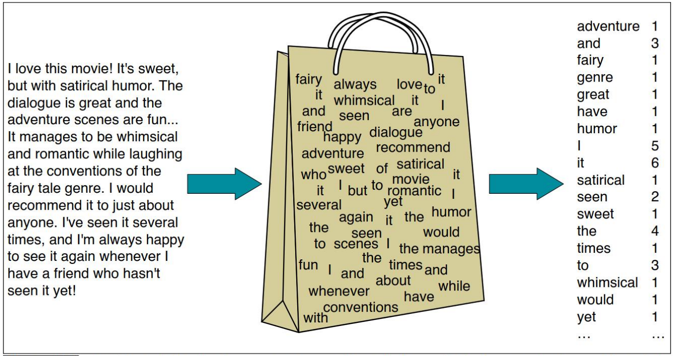


	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Figure 11.3 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

The vector space model of IR

Represent a document as a vector of counts of the words it contains: Bag-of-words representation

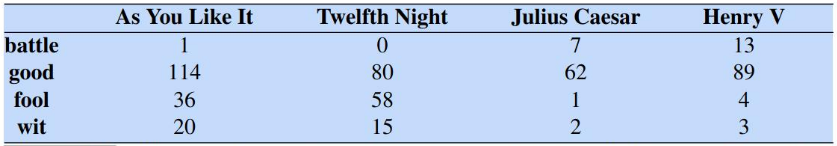


I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

adventure	1
and	3
fairy	1
genre	1
great	1
have	1
humor	1
I	5
it	6
satirical	1
seen	2
sweet	1
the	4
times	1
to	3
whimsical	1
would	1
yet	1
...	...

Figure 11.2 Intuition of the classic vector space model applied to a single document. The position of the words is ignored (the *bag-of-words* assumption) and we make use of the frequency of each word.

Term-document matrix



	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

Figure 11.3 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Visualizing document vectors

In two dimensions only: `battle` and `fool`

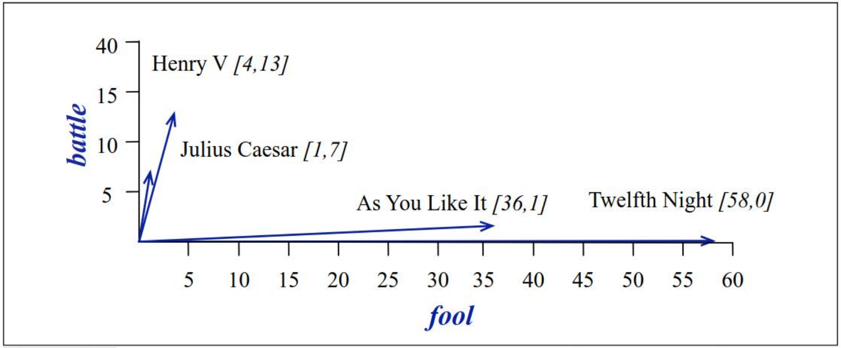


Figure 11.5 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Term weighting: tf-idf and BM25

In IR, we don't use raw word counts like $(1, 114, 36, 20)$ for *As You Like It*

Instead we compute a **term weight** for each document word

Two common term weighting schemes

- tf-idf
- variant of tf-idf called BM25

term frequency **tf** and the inverse document frequency **idf**

Term frequency

The **tf** term tells us how frequent the word is

Term frequency

The **tf** term tells us how frequent the word is

Words that occur more often in a document are likely to be informative about the document's contents

Term frequency

The **tf** term tells us how frequent the word is

Words that occur more often in a document are likely to be informative about the document's contents

We usually use the $\log_{10}$ of the word frequency, rather than the raw count.

- The intuition is that a word appearing 100 times in a document doesn't make that word 100 times more likely to be relevant to the meaning of the document

Term frequency

So if we define $\mathrm{count}(t, d)$ as the raw count of term t in document d, then $\mathrm{tf}_{t,d}$ is

Term frequency

So if we define $\text{count}(t, d)$ as the raw count of term t in document d, then $\text{tf}_{t,d}$ is

$$\text{tf}_{t,d} = \begin{cases} 1 + \log_{10} \text{count}(t, d) & \text{if } \text{count}(t, d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Document frequency

Document frequency df_t of a term t is the number of documents the term t occurs in

Document frequency

Document frequency df_t of a term t is the number of documents the term t occurs in

- Terms that occur in only a few documents are useful for discriminating those documents from the rest of the collection

Document frequency

Document frequency df_t of a term t is the number of documents the term t occurs in

- Terms that occur in only a few documents are useful for discriminating those documents from the rest of the collection
- Terms that occur across the entire collection are not as helpful

Inverse document frequency

Document frequency df_t of a term t is the number of documents the term t occurs in

Inverse document frequency of term t

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

where N is the total number of documents

Inverse document frequency

Document frequency df_t of a term t is the number of documents the term t occurs in

Inverse document frequency of term t

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

where N is the total number of documents

- The fewer documents in which a term occurs, the higher this weight
- The lowest weight of 0 is assigned to terms that occur in every document

tf-idf

The tf-idf value for word t in document d is then the product

$$\text{tf-idf}(t, d) = \text{tf}_{t,d} \cdot \text{idf}_t$$

Document scoring

We have each document d and the query q represented as vectors $\boldsymbol{d} \in \mathbb{R}^n$ and $\boldsymbol{q} \in \mathbb{R}^n$

How do we measure their similarity score?

Document scoring

We have each document d and the query q represented as vectors $\boldsymbol{d} \in \mathbb{R}^n$ and $\boldsymbol{q} \in \mathbb{R}^n$

How do we measure their similarity score?

Cosine similarity

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

Evaluation IR

Evaluation

The same **precision** and **recall** metrics we have been using

Evaluation

The same **precision** and **recall** metrics we have been using

We make the assumption that each document returned by the IR system is **either relevant** to our purposes or **not relevant**

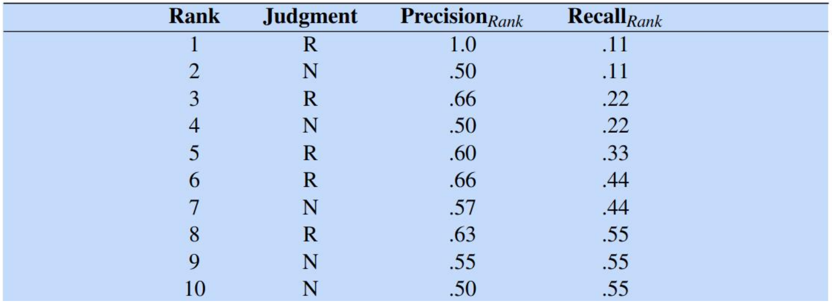
Evaluation

The same **precision** and **recall** metrics we have been using

We make the assumption that each document returned by the IR system is **either relevant** to our purposes or **not relevant**

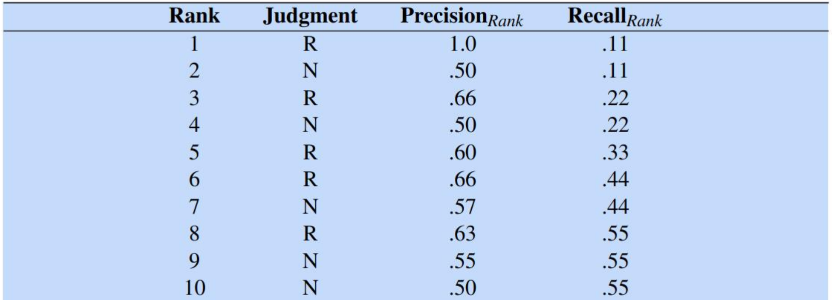
Unfortunately, these metrics do not adequately measure the performance of a system that ranks the documents it returns. For comparing two ranked retrieval systems, we need a metric that prefers the one that ranks the relevant documents higher.

Towards Mean Average Precision (MAP)



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

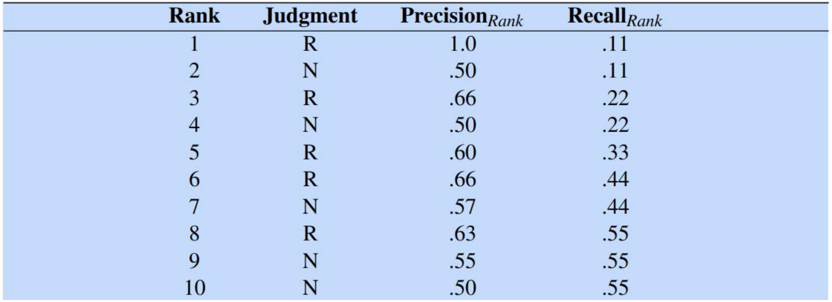
Towards Mean Average Precision (MAP)



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

Let R_r be the set of relevant documents at or above r

Towards Mean Average Precision (MAP)



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

Let R_r be the set of relevant documents at or above r

Let $\text{Precision}_r(d)$ be precision measured at the rank at which document d was found

Mean Average Precision (MAP)

Let R_r be the set of relevant documents at or above r

Let $\text{Precision}_r(d)$ be precision measured at the rank at which document d was found

Mean Average Precision (MAP)

Let R_r be the set of relevant documents at or above r

Let $\text{Precision}_r(d)$ be precision measured at the rank at which document d was found

The average precision AP for a single query is

$$\text{AP} = \frac{1}{|R_r|} \sum_{d \in R_r} \text{Precision}_r(d)$$

Mean Average Precision (MAP)

Let R_r be the set of relevant documents at or above r

Let $\mathrm{Precision}_r(d)$ be precision measured at the rank at which document d was found

The average precision AP for a single query is

$$\mathrm{AP} = \frac{1}{|R_r|} \sum_{d \in R_r} \mathrm{Precision}_r(d)$$

and the Mean Average Precision for a set of queries Q is

$$\mathrm{MAP} = \frac{1}{|Q|} \sum_{q \in Q} \mathrm{AP}(q)$$

Evaluation IR

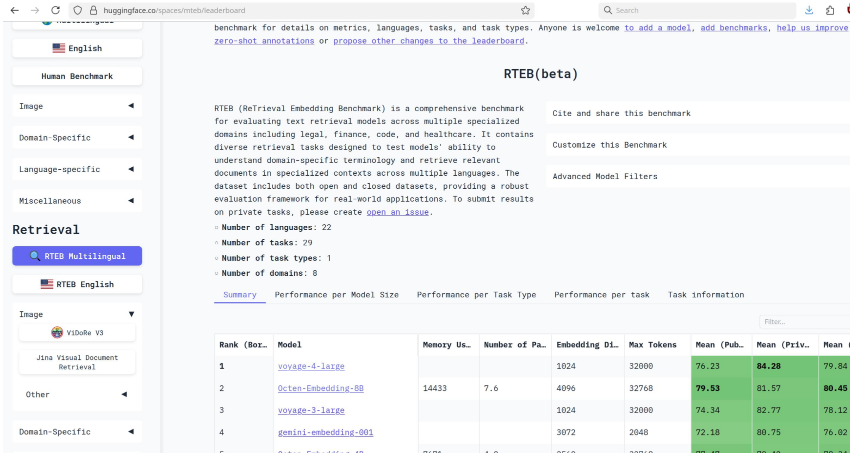
Dense information retrieval

Vector representation

Transformer-based model for representing both the query and the document collection

Recall SentenceBERT, but gazillions of other models...

IR benchmark/leader-board example



benchmark for details on metrics, languages, tasks, and task types. Anyone is welcome to add a model, add benchmarks, help us improve zero-shot annotations or propose other changes to the leaderboard.

RTEB(beta)

RTEB (ReTrieval Embedding Benchmark) is a comprehensive benchmark for evaluating text retrieval models across multiple specialized domains including legal, finance, code, and healthcare. It contains diverse retrieval tasks designed to test models' ability to understand domain-specific terminology and retrieve relevant documents in specialized contexts across multiple languages. The dataset includes both open and closed datasets, providing a robust evaluation framework for real-world applications. To submit results on private tasks, please create open an issue.

- **Number of languages**: 22
- **Number of tasks**: 29
- **Number of task types**: 1
- **Number of domains**: 8

Cite and share this benchmark

Customize this Benchmark

Advanced Model Filters

Summary | Performance per Model Size | Performance per Task Type | Performance per task | Task information

Filter...

Rank (Bor…	Model	Memory Us…	Number of Pa…	Embedding Di…	Max Tokens	Mean (Pub…	Mean (Priv…	Mean (
1	voyage-4-large			1024	32000	76.23	**84.28**	79.84
2	Octen-Embedding-8B	14433	7.6	4096	32768	**79.53**	81.57	**80.45**
3	voyage-3-large			1024	32000	74.34	82.77	78.12
4	gemini-embedding-001			3072	2048	72.18	80.75	76.02
5	Octen-Embedding-4B	7671	4.0	2560	32768	77.47	[illegible]	[illegible]

Octen-Embedding-8B: "Octen-Embedding-8B is a text embedding model designed for semantic search and retrieval tasks. This model is fine-tuned from Qwen/Qwen3-Embedding-8B and supports multiple languages, providing high-quality embeddings for various applications." — "Qwen3 Embedding model series is the latest proprietary model of the Qwen family, specifically designed for text embedding and ranking tasks"

Retrieval Augmented Generation

RAG in a nutshell

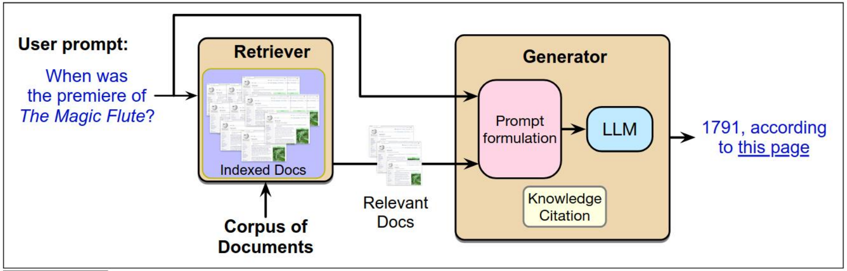


Figure 11.13 Retrieval-augmented generation takes as input a user prompt (which may express an information need like this question example), and a corpus of documents that may be useful in meeting the information need. The method has two stages: **retrieval**, which returns relevant documents from the collection, and **generation**, in which an LLM **generates** text given the documents as a prompt. Some generations include a **knowledge citation** that can help the user decide whether to trust the generation, or follow up if they are interested.

RAG algorithm

The idea of RAG is to condition on the retrieved passages, jointly with some prompt text, for example like `Based on these texts, answer this question: ...`

Given a document collection D and a user query q, the most basic RAG algorithm is

1. Call a retriever to return $R(q) = d_1, \ldots, d_k$, the top-k relevant passages from D
2. Create a prompt that includes q and the retrieved passages
3. Call an LLM with the prompt

LLM prompting with RAG

Schema of a prompt

```
retrieved passage 1
retrieved passage 2
...
retrieved passage k
Based on these texts, answer this question:
What year was the premiere of The Magic
Flute?
```

Extensions of basic RAG

Extensions of basic RAG

In **agent-based RAG**, the system decides when to call a **retrieval agent** and for which collection

Extensions of basic RAG

In **agent-based RAG**, the system decides when to call a **retrieval agent** and for which collection

There may be noise in the retrieved passages; some of them may be irrelevant or wrong, or in an unhelpful order. How can we encourage the LLM to focus on the good passages? Some RAG architectures add a **reranker** that reranks or reorders passages after they are retrieved

Knowledge citation (or Answer Attribution)

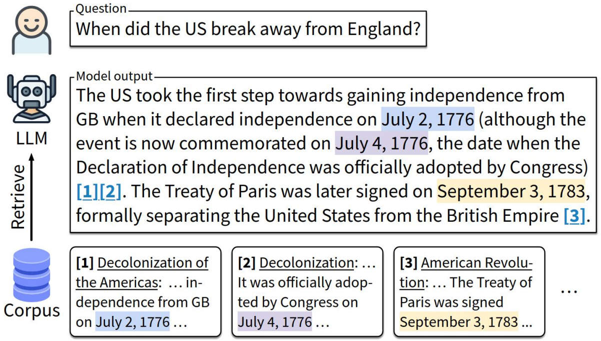

Question
When did the US break away from England?
Model output
The US took the first step towards gaining independence from GB when it declared independence on July 2, 1776 (although the event is now commemorated on July 4, 1776, the date when the Declaration of Independence was officially adopted by Congress) [1][2]. The Treaty of Paris was later signed on September 3, 1783, formally separating the United States from the British Empire [3].
LLM
Retrieve
Corpus
[1] Decolonization of the Americas: … independence from GB on July 2, 1776 …
[2] Decolonization: … It was officially adopted by Congress on July 4, 1776 …
[3] American Revolution: … The Treaty of Paris was signed September 3, 1783 …
…

Figure 1: The task setup of ALCE. Given a question, the system generates text while providing *citing passages* from a large retrieval corpus. Each statement may contain multiple citations (e.g., [1][2]).

T. Gao, H. Yen, J. Yu, and D. Chen (2023). **"Enabling Large Language Models to Generate Text with Citations".** In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, pp. 6465–6488

Knowledge citation (or Answer Attribution)

The simplest way for generating knowledge citations is to specify it as part of the prompt.

Example

```
Write an answer for the given question
using only the provided search results
(some of which might be irrelevant) and
cite them properly... Always cite for any
factual claim
```

T. Gao, H. Yen, J. Yu, and D. Chen (2023). **"Enabling Large Language Models to Generate Text with Citations".** In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Ed. by H. Bouamor, J. Pino, and K. Bali. Singapore: Association for Computational Linguistics, pp. 6465–6488

Answer Attribution can be Unfaithful

"The aforementioned approaches do not account for attributions' faithfulness, i.e. whether the selected documents influence the LLM during the generation."

"Indeed, the presence of an entailment relation or high semantic similarity does not imply that the retrieved document had an influence on the answer generation process."

J. Qi, G. Sarti, R. Fernández, and A. Bisazza (Nov. 2024). **"Model Internals-Based Answer Attribution for Trustworthy Retrieval-Augmented Generation".** In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Ed. by Y. Al-Onaizan, M. Bansal, and Y.-N. Chen. Miami, Florida, USA: Association for Computational Linguistics, pp. 6037–6053

First RAG papers

"We demonstrated a **simple technique** to greatly improve factual unsupervised cloze QA by providing context documents as additional inputs. We used oracle documents to establish an upper bound to this improvement, and found that **using off-the-shelf information retrieval** is sufficient to achieve performance on par with the supervised DrQA system. We also investigated how brittle language models' factual predictions were to **noisy and irrelevant context** documents."

F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel (2020). **"How Context Affects Language Models' Factual Predictions".** In: *Automated Knowledge Base Construction*. Virtual conference

License and credits

Credits

Ivan Habernal

Content from ACL Anthology papers licensed under CC-BY
`https://www.aclweb.org/anthology`

This lecture re-used materials from Jurafsky and Martin (2026). They explicitly state: *“Feel free to use the draft chapters and slides in your classes, print it out, whatever, the resulting feedback we get from you makes the book better!”*, see
`https://web.stanford.edu/~jurafsky/slp3/`