

CS 1671 / CS 2071 / ISSP 2071

Human Language Technologies

Session 25: Information retrieval, RAG

Michael Miller Yoder

April 15, 2026



University of
Pittsburgh

School of Computing and Information

Assessments: homework

- [Homework 3](#) is **due tomorrow, Thu Apr 16**
- Run Jupyter notebooks from templates on the CRCDCD
- Part 1: LLM prompting
 - Use CRCDCD CPUs
- Part 2: Instruction tuning of an LLM
 - Use CRCDCD GPUs

Assessments: project

- [Project final report](#) is **due Tue Apr 28**
- Implement your proposed LLM system, compare with your baseline system
- See [project website](#) for instructions of what to include
- [Project presentation](#) is **in class Fri May 1, 10-11:50am**
 - Ungraded
 - Michael will post a shared PowerPoint

Assessments: extra credit quiz

- Optional, extra credit quiz during class **this Mon Apr 20** covering:
 - Session 21: Prompting, finetuning, and evaluating LLMs
 - J+M 7.6, 8.8.3, 9.4
 - Session 24: Machine translation
 - J+M 12-12.4, 12.6-12.8
 - Session 25 (today): Information retrieval, RAG
 - J+M 11

Learning objectives: information retrieval (IR), RAG

Students will be able to:

- Diagram the process of **classic information retrieval based on sparse embeddings**
- Explain the goal of word frequency weighting schemes like **tf-idf**
- Describe how **retrieval-augmented generation (RAG)** works
- List software that can be used to build classic IR systems and RAG
- Identify and explain a common evaluation IR evaluation metric, **mean reciprocal rank (MRR)**

Information retrieval and question answering

- Information retrieval (IR)
 - Choosing the most relevant document/s from a set of documents given a user's query
 - Search engines
- Closely related to question answering (QA)



Traditional IR with tf-idf and BM25 weighting

Traditional IR: sparse embeddings

Sparse embeddings (bag-of-words) of documents and queries

- Each cell is the count of term t in a document d ($tf_{t,d}$).
- Each document is a **count vector** in \mathbb{N}^V , a column below.



	As You Like It	Twelfth Night	Julius Caesar	Henry V
<i>battle</i>	1	1	8	15
<i>soldier</i>	2	2	12	36
<i>fool</i>	37	58	1	5
<i>clown</i>	6	117	0	0

Raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies
 - Whether in term-document or term-term matrices
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it*, or *they* are not very informative about the context
 - 2 documents that use a lot of *the* are not necessarily similar
- It's a paradox! How can we balance these two conflicting constraints?

Weighting words in term-document and term-term matrices

- **tf-idf** (term frequency-inverse document frequency)
 - For representing documents with their most unique words (for text classification, information retrieval)
 - Term-document matrix

Term frequency (tf)

$$tf_{t,d} = \text{count}(t,d)$$

Instead of using raw count, we squash a bit:

$$tf_{t,d} = \begin{cases} 1 + \log \text{count}(t,d) & \text{if } \text{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Document frequency (df)

df_t is the number of documents term t occurs in.

(note this is not collection frequency, which is the total count across all documents)

"Romeo" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10} \left(\frac{N}{\text{df}_t} \right)$$

- N is the total number of documents in the collection
- Documents can be whatever you want! (Full documents, paragraphs, etc)

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

tf-idf Controls for Frequent but Uninformative Words

Some words are very common in a given document because they are common across all documents (e.g., *the*). They are not discriminative. **tf-idf** (product of term frequency and inverse document frequency) addresses this:

$$tf_{t,d} = \begin{cases} 1 + \log count(t,d) & \text{if } count(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$idf_f = \log_{10} \frac{N}{df_t}$$

$$tf-idf(t, d) = tf_{t,d} \cdot idf_t$$

Final tf-idf weighted values

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

tf-idf:

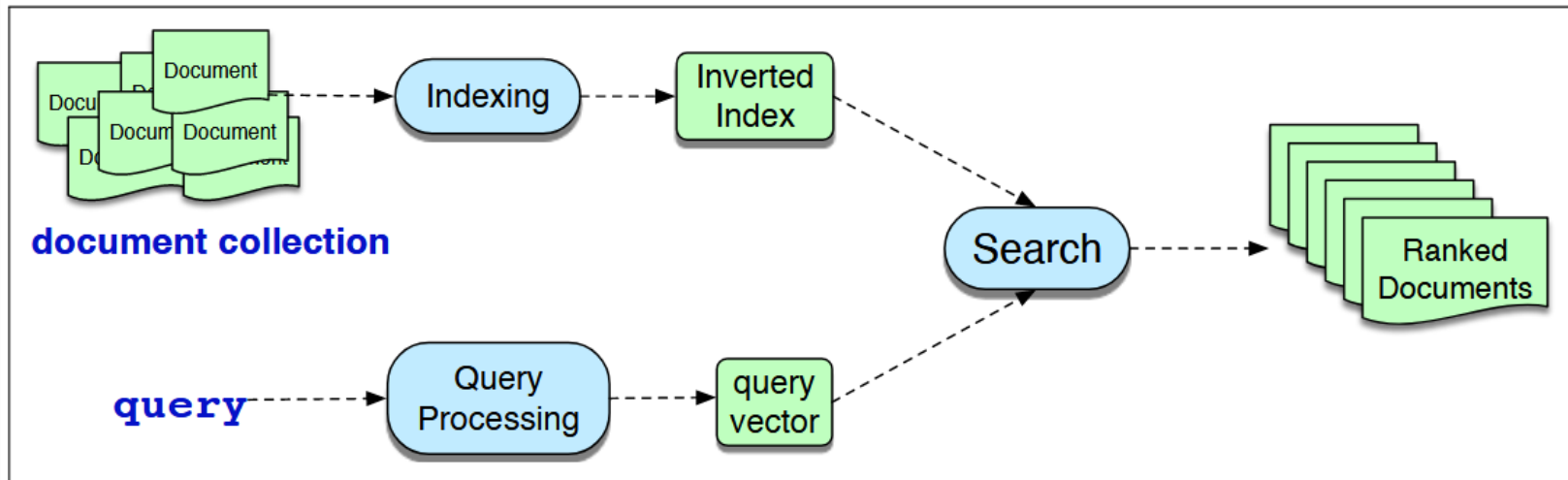
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

BM25 transformations of bag-of-word vectors

- Modification of tf-idf
- Additional parameters:
 - k to control how much we care about word frequency
 - b to control how much we care about document length normalization
- Score of document d given query q :

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

Traditional IR pipeline



- Return documents with most similar vectors to query vector (by cosine similarity)
- Inverted index: $term \{document\ frequency\} \rightarrow document_id1 [term\ frequency] document_id2 [term\ frequency]$
 - E.g. chicken {50} -> 774 [20] 32 [2]

Retrieval-augmented generation (RAG)

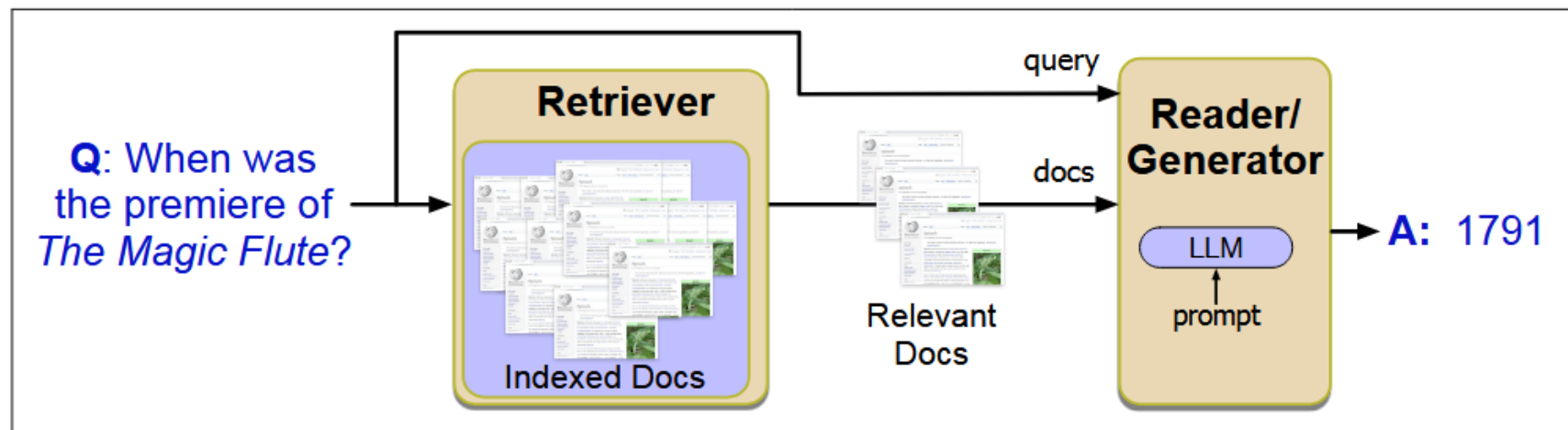
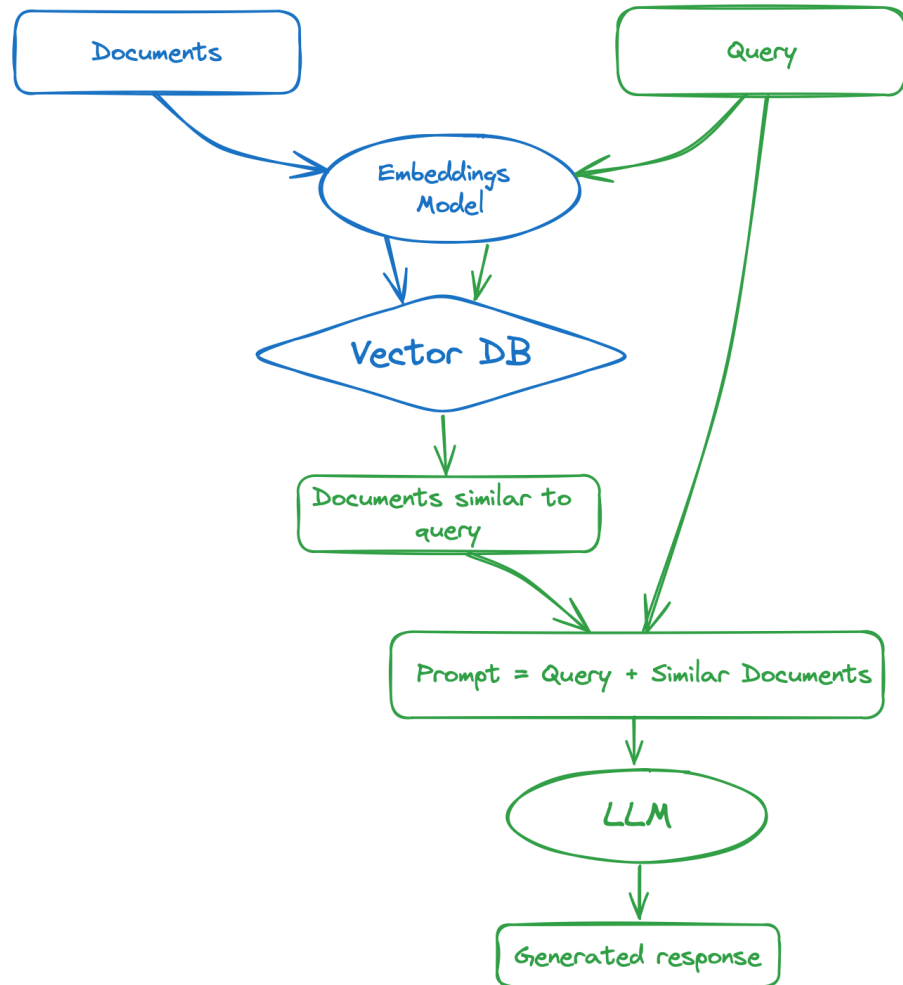


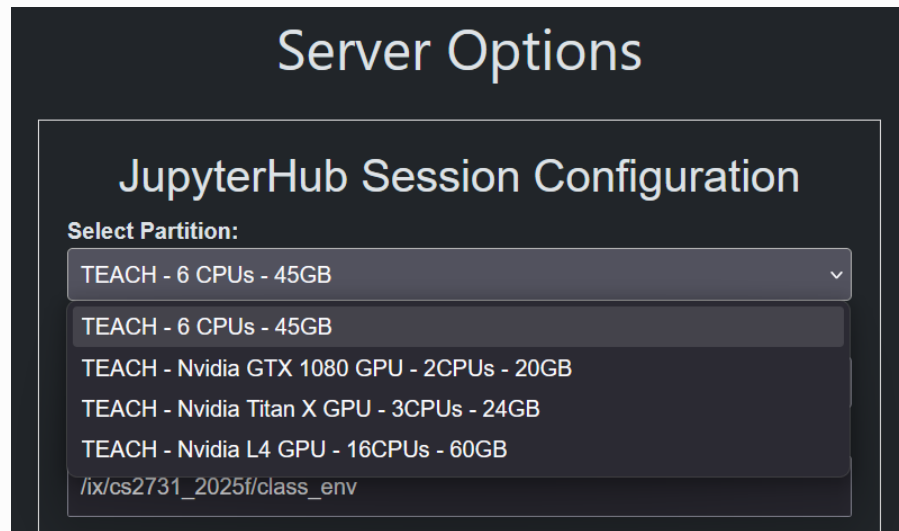
Figure 14.9 Retrieval-based question answering has two stages: **retrieval**, which returns relevant documents from the collection, and **reading**, in which an LLM **generates** answers given the documents as a prompt.



• Coding activity

Notebooks: sparse IR and RAG

1. Go to this [nbgitpuller link](#) (also available on course website)
2. Start a server with **TEACH – 6 CPUs, 48 GB**
 - No need for GPUs
3. Load custom environment at `/ix1/cs1671-2026s/class_env`
4. Open `session25_sparse_ir.ipynb`
5. Open `session25_rag.ipynb`



Notebooks: what to record

- `session25_sparse_ir.ipynb`
 - Record:
 - Observations from trying different queries on MS MARCO
 - Mean reciprocal rank (MRR) on MS MARCO dev subset
- `session25_rag.ipynb`
 - Record:
 - Comparison between directly asking LLM and doing RAG
- If you finish early, try building a classic IR or RAG system on a new corpus of your choosing!

Wrapping up

- Classic information retrieval returns documents based on cosine similarity to the query's sparse embeddings, often transformed with tf-idf or BM25
- Retrieval-augmented generation provides relevant documents as context to an LLM to generate a response to prompts and questions
- Mean reciprocal rank (MRR) can be used for evaluation of information retrieval systems