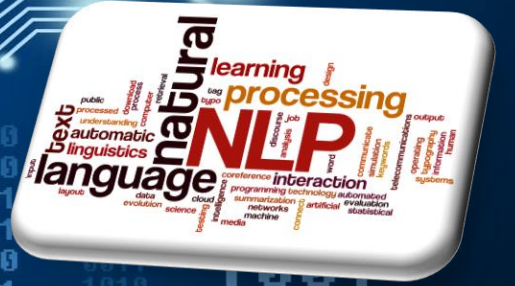
Cairo University
Faculty of Engineering
Computer Engineering Department

Natural Language Processing

Dr. Sandra Wahid



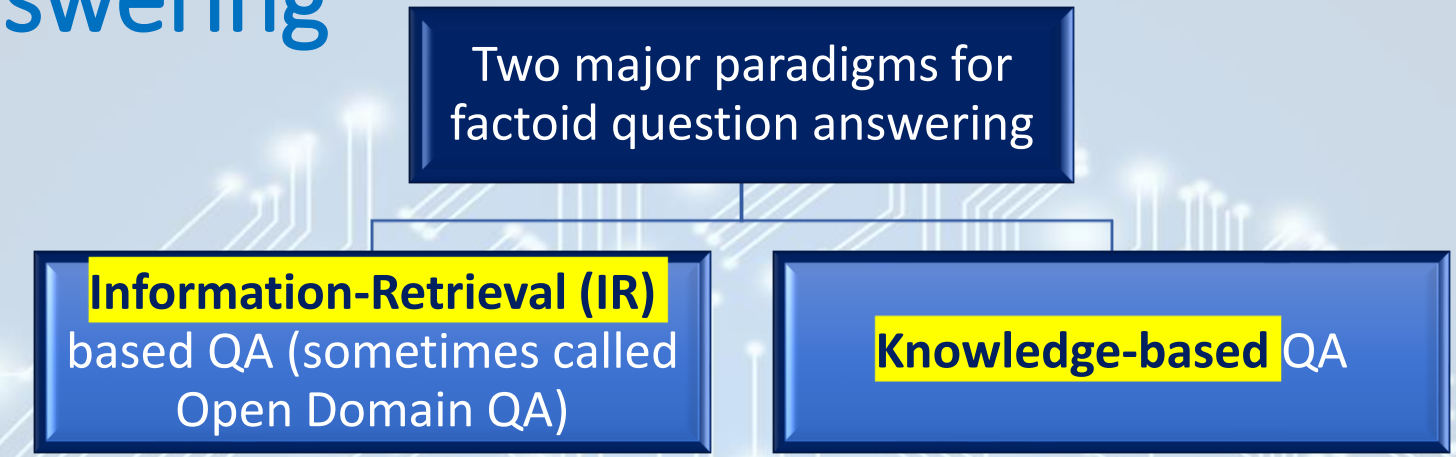
Question Answering

- Question Answering (QA) systems are designed to fill human information needs that might arise in situations like talking to a **virtual assistant**, interacting with a **search engine**, or **querying a database**.

- Most question answering systems focus on a particular subset of these information needs: **factoid questions**, questions that can be answered with simple facts expressed in short texts, like the following:
  - Where is the Louvre Museum located?
  - Who discovered electricity?

- Many other QA tasks include:
  - **long-form question answering** (answering questions like "why" questions that require generating long answers)
  - **community question answering** (using datasets of community-created question-answer pairs like Quora or Stack Overflow)
  - **answering questions on human exams** (like the New York Regents Science Exam as an NLP/AI benchmark to measure progress in the QA field).

Question Answering



Information-Retrieval (IR) based QA:

- Relies on the **vast amount of text** on the web or in collections of scientific papers.
- Given a user question, information retrieval is used to find **relevant** passages. Then neural **reading comprehension** algorithms read these retrieved passages and draw an answer directly from **spans of text**.

Knowledge-based QA:

- A system instead builds a **semantic representation** of the query.
- For example: mapping "What states border Texas?" to the **logical representation**:

$$\lambda x.state(x) \wedge borders(x, texas)$$

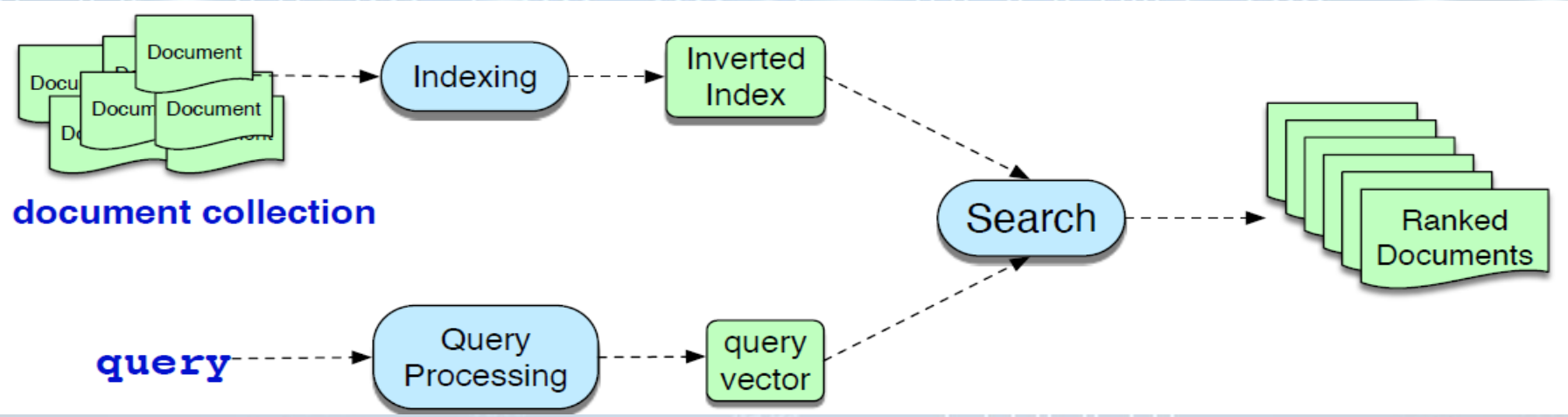
- These meaning representations are then used to query databases of facts.

Question Answering

- Other QA paradigms:
  - How to **query a language model** directly to answer a question, relying on the fact that huge pretrained language models have already encoded a lot of factoids.
  - **Pre-neural hybrid question-answering** algorithms that combine information from IR-based and knowledge-based sources.

Information Retrieval

- IR is the field encompassing the retrieval of all manner of media based on user information needs.
- **Ad hoc retrieval:** in which a user poses a **query** to a retrieval system, which then returns an ordered set of **documents** from some **collection**.
  - A document refers to whatever **unit of text** the system indexes and retrieves (web pages, scientific papers, news articles, or even shorter passages like collection paragraphs).
  - A collection refers to a set of documents being used to satisfy user requests.
  - A term refers to a word in a collection, but it may also include phrases.
  - A query represents a user's information need expressed as a set of terms.



High-level architecture of an ad hoc retrieval engine

Information Retrieval: Term Weighting and Document Scoring

- The basic IR architecture uses the vector space model in which we map queries and document to **vectors** based on unigram word counts, and use the **cosine** similarity between the vectors to rank potential documents→ This is thus an example of the **bag-of-words model** since words are considered independently of their positions.
- Instead of using raw word counts in IR, we use a **term weight** for each document word.
  - Two term weighting schemes are common: the **tf-idf** weighting and **BM25** a slightly more powerful variant.
  - Tf-idf: $\text{tf}_{t,d} = \log_{10}(\text{count}(t,d)+1)$ $\text{idf}_t = \log_{10}\frac{N}{\text{df}_t}$ $\text{tf-idf}(t,d) = \text{tf}_{t,d} \cdot \text{idf}_t$

Remember this is not our default

  - Document Scoring: $\text{score}(q,d) = \cos(\mathbf{q},\mathbf{d}) = \frac{\mathbf{q}\cdot\mathbf{d}}{|\mathbf{q}||\mathbf{d}|}$ **d:** document vector ,**q:** query vector

$$\text{score}(q,d) = \sum_{t\in\mathbf{q}} \frac{\text{tf-idf}(t,q)}{\sqrt{\sum_{q_i\in q}\text{tf-idf}^2(q_i,q)}} \cdot \frac{\text{tf-idf}(t,d)}{\sqrt{\sum_{d_i\in d}\text{tf-idf}^2(d_i,d)}}$$

Information Retrieval: Term Weighting and Document Scoring

- Queries are usually **very short**, so each query word is likely to have a count of 1. And the cosine normalization for the query (the division by |q|) will be the same for all documents, so won't change the ranking between any two documents → So we generally use the following simple score for a document d given a query q:

$$\text{score}(q,d) = \sum_{t \in q} \frac{\text{tf-idf}(t,d)}{|d|}$$

- Example:

Query: sweet love

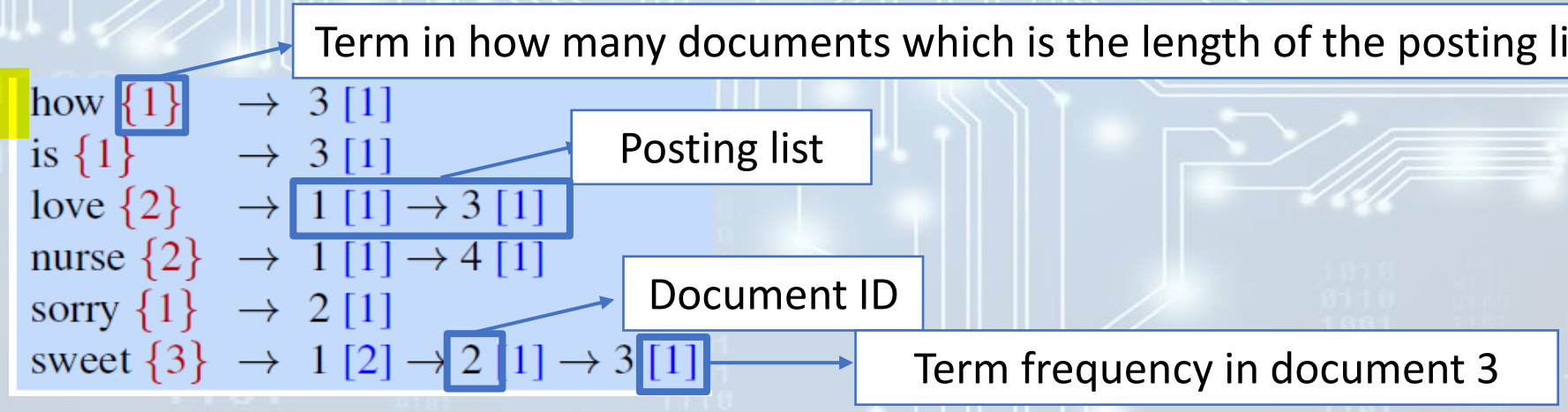
Doc 1: Sweet sweet nurse! Love?
Doc 2: Sweet sorrow
Doc 3: How sweet is love?
Doc 4: Nurse!

		Document 1					Document 2			
word	**count**	**tf**	**df**	**idf**	**tf-idf**	**count**	**tf**	**df**	**idf**	**tf-idf**
love	1	0.301	2	0.301	0.091	0	0	2	0.301	0
sweet	2	0.477	3	0.125	0.060	1	0.301	3	0.125	0.038
sorrow	0	0	1	0.602	0	1	0.301	1	0.602	0.181
how	0	0	1	0.602	0	0	0	1	0.602	0
nurse	1	0.301	2	0.301	0.091	0	0	2	0.301	0
is	0	0	1	0.602	0	0	0	1	0.602	0
$\lvert d_1 \rvert = \sqrt{.091^2 + .060^2 + .091^2} = .141$						$\lvert d_2 \rvert = \sqrt{.038^2 + .181^2} = .185$				

Doc	$\lvert d \rvert$	**tf-idf**(*sweet*)	**tf-idf**(*love*)	**score**
1	.141	.060	.091	1.07
3	.274	.038	.091	0.471
	.185	.038	0	0.205
4	.090	0	0	0

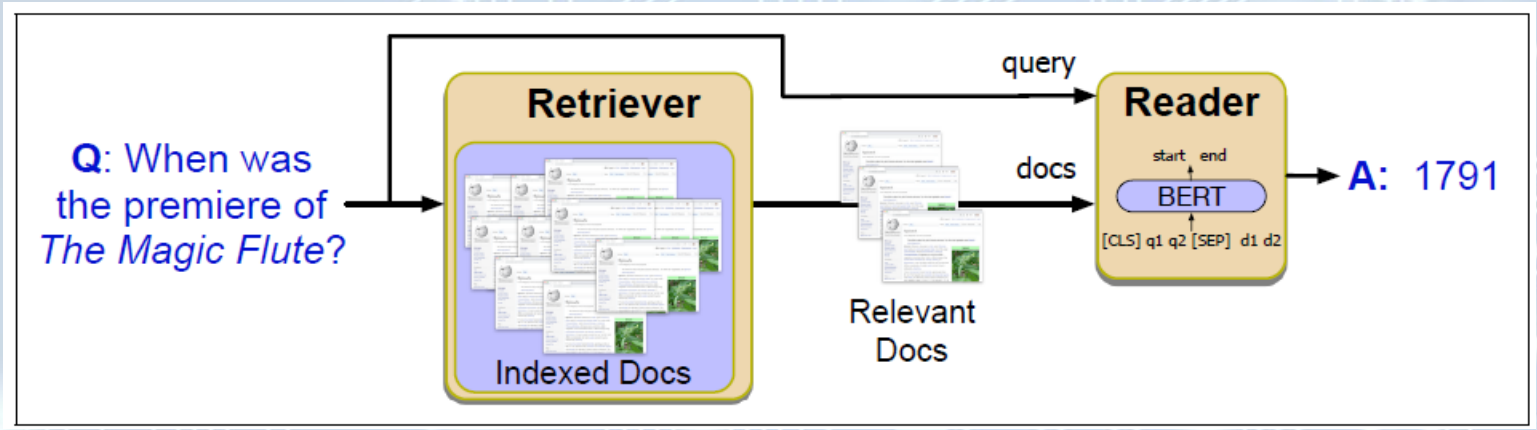
Information Retrieval: Inverted Index

- In order to compute scores, we need to efficiently find documents that contain words in the query → the basic search problem in IR is thus to find all documents d ϵ C that contain a term q ϵ Q.
- The data structure for this task is the **inverted index**:
  - Given a query term, gives a list of documents that contain the term.
  - It consists of two parts, a **dictionary** and the **postings**.
  - The dictionary is a list of terms (designed to be efficiently accessed), each pointing to a postings list for the term.
  - A postings list is the list of document IDs associated with each term, which can also contain information like the term frequency or even the exact positions of terms in the document.
  - Example:



IR-based Factoid Question Answering

- This is a **two stage** model: **1-Retriever 2-Reader**



1-Retriever: returns the **relevant documents** (discussed in the previous slides)

2-Reader: takes a passage as input and **produce the answer**

- In the **extractive QA** we discuss here, the answer is a span of text in the passage.
- More difficult task of **abstractive QA**, in which the system can write an answer which is not drawn exactly from the passage.

IR-based Factoid Question Answering: Reader

- Example (consider extractive QA): given a question like "How tall is Mt. Everest?" and a passage that contains the clause *"Reaching 29,029 feet at its summit"*, a reader will output *29,029 feet*.

- The answer extraction task is commonly modeled by **span labeling**: identifying in the passage a span (a continuous string of text) that constitutes an answer.

- Neural algorithms for reading comprehension are given a question *q* of *n* tokens *q1,…, qn* and a passage *p* of *m* tokens *p1,…,pm* → *t*heir goal is to compute the probability *P(a|q,p)* that each possible span *a* is the answer.

- If each span *a* starts at position a_s and ends at position a_e: $P(a|q,p) = P_{\text{start}}(a_s|q,p)P_{\text{end}}(a_e|q,p)$

- Thus, for each token *pi* in the passage we'll compute two probabilities: $p_{start}(i)$ that *pi* is the start of the answer span and $p_{end}(i)$ that *pi* is the end of the answer span.

- A standard baseline algorithm for reading comprehension is to pass the question and passage to any encoder like BERT.

Evaluation of Factoid Answers

- Using **Mean Reciprocal Rank (MRR).**
- MRR is designed for systems that return a **short ranked list of answers** or passages for each test set question, which we can compare against the (human-labeled) correct answer.
- First, each test set question is scored with the reciprocal of the rank of the first correct answer.
  - For example, if the system returned five answers to a question but the first three are wrong (so the highest-ranked correct answer is ranked fourth), the reciprocal rank for that question is ¼.
  - The score for questions that return no correct answer is 0.
- The MRR of a system is the average of the scores for each question in the test set.
- In some versions of MRR, questions with a score of zero are ignored in this calculation.
- Formally, for a system returning ranked answers to each question in a test set Q, (or in the alternate version, let Q be the subset of test set questions that have non-zero scores):

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{rank_i}$$


Thank You