Question Answering, Information Retrieval and Retrieval-Augmented-Generation

Phạm Quang Nhật Minh

minhpham0902@gmail.com

Introduction to Question Answering (QA)

- Humans ask computers for answers
- QA systems fulfill information needs
- Early QA: databases, simple parsing
- IBM Watson: Jeopardy! champion (2011)
- Modern QA uses large language models
- QA closely linked with search engines

Types of Questions in QA

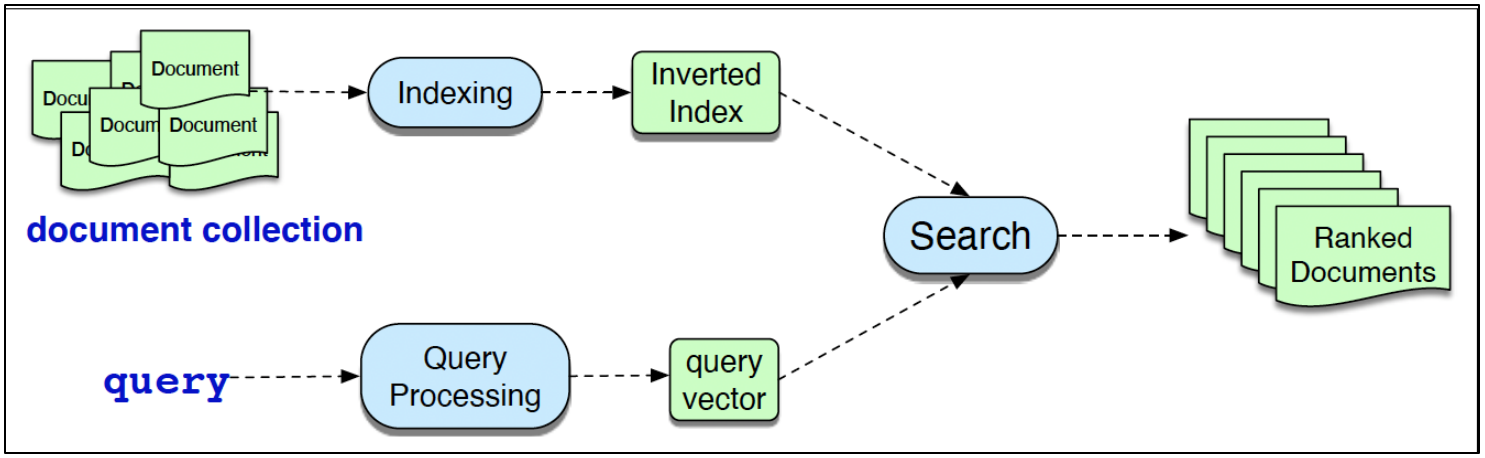
- Factoid questions: short factual answers
  - Examples: "Where is Louvre Museum?”
- Modern systems use large language models
  - Prompting LLMs for fact-based generation
  - LLMs encode facts in their parameters
  - Not all questions suited for prompting

Challenges with Large Language Models

- LLMs can hallucinate incorrect answers
- Hallucinations sound reasonable but false
- LLMs lack calibration for answer confidence
- Proprietary data inaccessible via LLM prompts
- Static models fail with recent information
  - Retrieval-Augmented Generation (RAG) addresses issues

Information Retrieval (IR) Basics

- IR retrieves documents based on queries
- Query: user's information need expressed textually
- Document: unit of indexed retrievable text
- Collection: set of all indexed documents
- High-level architecture includes ranking system
- Uses vector space models for similarity scoring



Term Weighting in IR

- Raw word counts not used directly
- Term Frequency-Inverse Document Frequency (TF-IDF) applied
  - TF measures word frequency in document
  - IDF reduces weight of common words
  - Formula: TF-IDF=TF×IDF

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 and Cosine Similarity

- Documents/queries represented as vectors
- Cosine similarity measures vector alignment:

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

- Higher cosine = better document match
  - Example: Ranking documents by cosine score

Example

				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
$\|q\| = \sqrt{.125^2 + .301^2} = .326$						

			Document 1					Document 2		
word	**cnt**	**tf**	**tf-idf**	**n'lized**	**× q**	**cnt**	**tf**	**tf-idf**	**n'lized**	**×q**
sweet	2	1.301	0.163	0.357	**0.137**	1	1.000	0.125	0.203	**0.0779**
nurse	1	1.000	0.301	0.661	0	0	0	0	0	0
love	1	1.000	0.301	0.661	**0.610**	0	0	0	0	**0**
how	0	0	0	0	0	0	0	0	0	0
sorrow	0	0	0	0	0	1	1.000	0.602	0.979	0
is	0	0	0	0	0	0	0	0	0	0
$\|d_1\| = \sqrt{.163^2 + .301^2 + .301^2} = .456$						$\|d_2\| = \sqrt{.125^2 + .602^2} = .615$				
Cosine: $\sum$ of column: **0.747**						Cosine: $\sum$ of column: **0.0779**				

Figure 14.2 Computation of tf-idf cosine score between the query and nano-documents 1 (0.747) and 2 (0.0779), using Eq. 14.4, Eq. 14.5, Eq. 14.6 and Eq. 14.9.

Inverted Index in IR

- Efficiently finds documents containing query terms
- Dictionary stores terms and metadata
- Postings list maps terms to document IDs
- Includes term frequencies and positions
- Alternatives include bigram indexing, hashing

Evaluating IR Systems

- Metrics: Precision and Recall defined as:
  - Precision = Relevant / Retrieved documents
  - Recall = Relevant / Total relevant documents
- Metrics adapted to ranked retrieval systems
  - Mean Average Precision (MAP) evaluates ranking quality

Dense Vector Representations

- Sparse vectors limited by vocabulary mismatch
- Dense embeddings handle synonyms effectively
- BERT encoders create dense vector representations
- Query and document jointly encoded for relevance scoring

Bi-Encoding vs. Joint Encoding

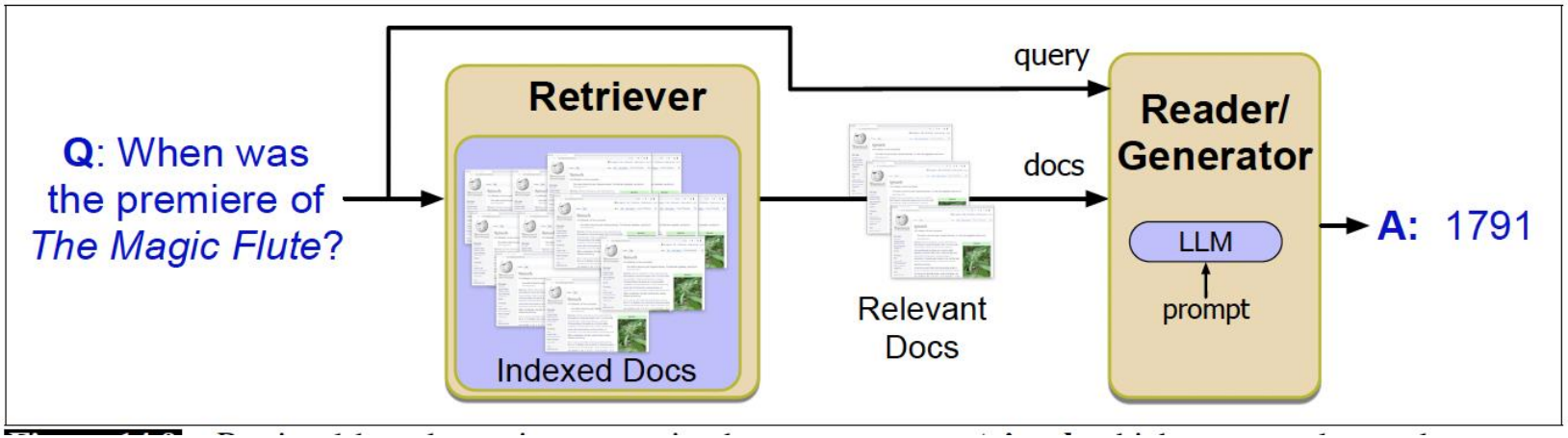
- Joint Encoding
  - Query + document encoded together
  - Accurate but computationally expensive
- Bi-Encoding
  - Separate encoders for query/document
  - Precomputed document embeddings stored

Intermediate Approaches to Dense Retrieval

- BM25 First Pass:
  - Ranks documents using sparse methods
- ColBERT Approach:
  - Encodes tokens individually into vectors
  - Scores based on token-level similarity

Retrieval-Augmented Generation (RAG)

- Two Components:
  - Retriever fetches relevant passages/documents
  - Reader generates answers from retrieved text
- Example Workflow:
  - Query → Retrieve → Generate Answer



Advantages of RAG

- Reduces hallucination by grounding answers
- Combines private/public data effectively
- Handles dynamic or proprietary datasets

Prompt Engineering in RAG

Prompt Engineering in RAG:

- Retrieved passage(s) as input context
- Question appended after passages

Model generates answer token-by-token

Schematic of a RAG Prompt

```
retrieved passage 1

retrieved passage 2

...

retrieved passage n

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

Evaluation Metrics for QA Systems

- Exact Match (% matching gold answer exactly)
- Token F1 Score (partial overlap measure)
- Mean Reciprocal Rank (MRR for ranked answers)

Historical Evolution of QA Systems

- Early systems used structured databases (1960s)
- Parsing + keyword matching approaches emerged later
- Rise of web-driven IR systems in the 1990s

Examples: BASEBALL, LUNAR systems

Modern Neural QA Systems

- Neural reading comprehension introduced in mid-2010s
- Dense retrieval + span-based readers became standard
- End-to-end architectures like RAG dominate today

Summary of Key Concepts

- QA fulfills user information needs effectively
- IR retrieves ranked documents based on queries
- Dense vectors address vocabulary mismatch
- RAG integrates retrieval with generative models