Connecting LLMs to Data with Retrieval

Matei Zaharia

Based on slides by Omar Khattab

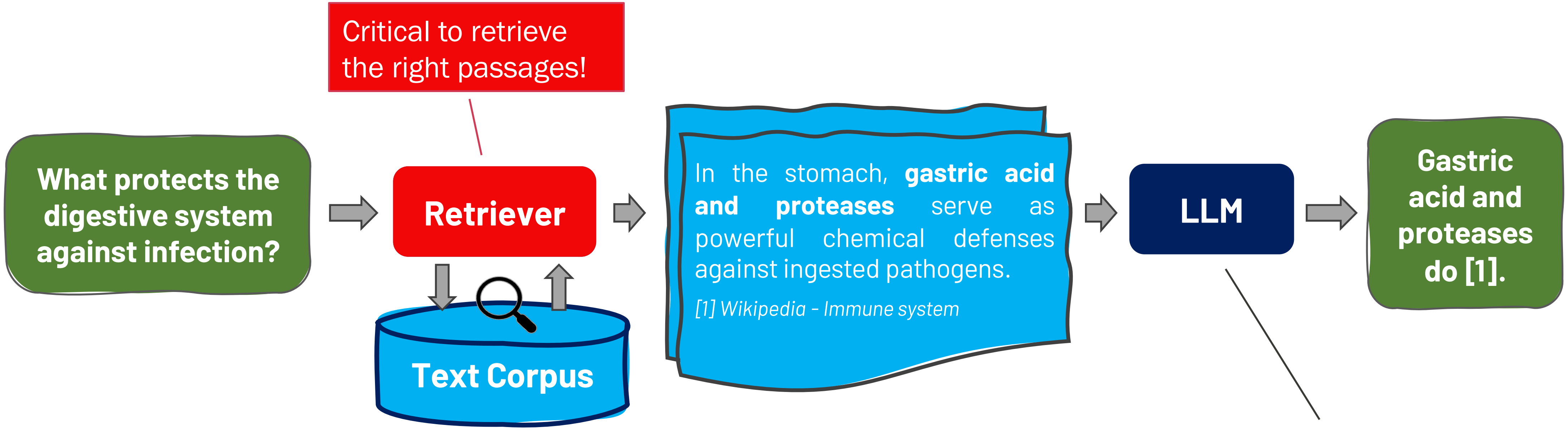
The idea

- LLMs have learned knowledge from their training data, but we often want to use them on data that wasn't in the training set
  - E.g., proprietary data in our app, or new data

- A common solution is to connect LLMs with retrieval (search)

- This is one example of connecting LLMs with *tools* more broadly
  - Can imagine calling into other tools too (e.g., SQL database), and complex pipelines with multiple calls to tools

Retrieval-augmented generation (RAG)



Prompt: Answer this question based on these potentially relevant passages. Provide references in [] format.

Question: ...
Passage [1]: ...

In this lecture

- Introduction to information retrieval (Matei)

- LlamaIndex: an open source toolkit for connecting LLMs to data (Jerry Liu, creator and founder)

What is information retrieval (IR)?

Finding material that fulfills an **information need** from within a **large collection** of **unstructured documents**.

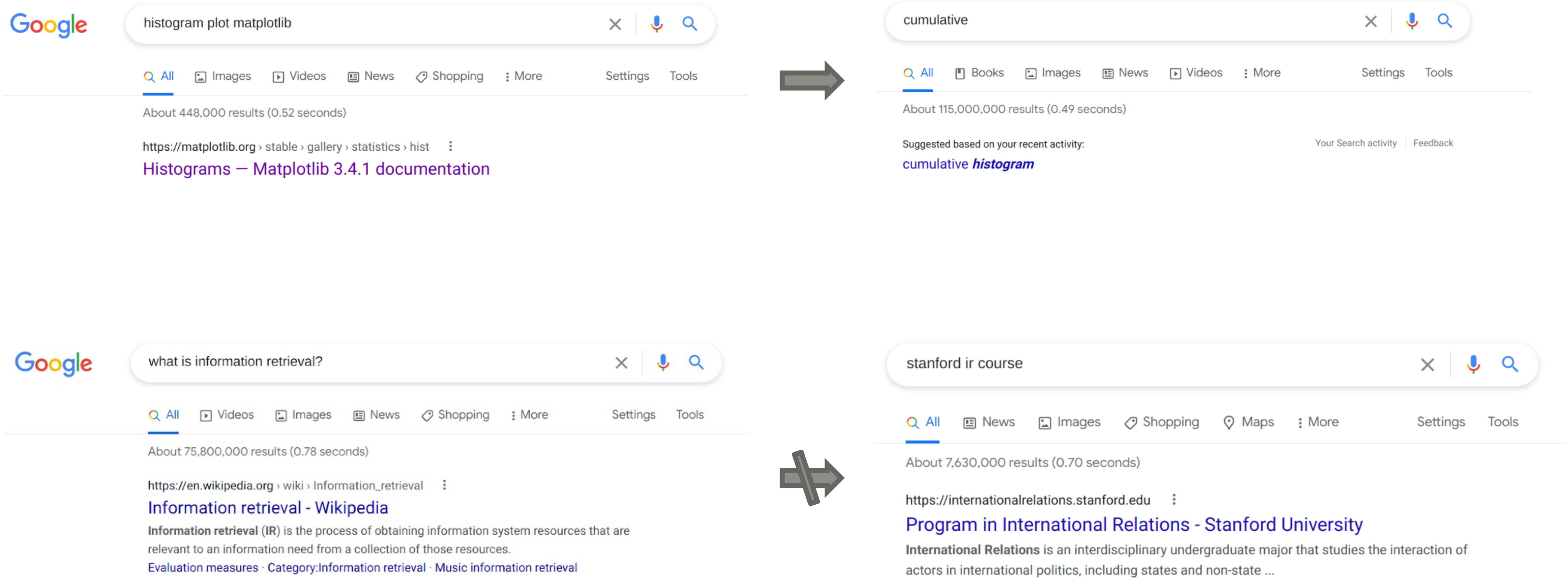
Simplified definition from IIR Book (Manning, Raghavan, and Schütze)

Relevance and the “information need”

- The goal of a search system is to satisfy an **information need**.
  - Material we retrieve is **relevant** only if it advances this goal.

- In most tasks, the user will express a **query**.
  - But queries can be ambiguous, incomplete, or inaccurate.
  - We must rely on our knowledge of the task and the user.

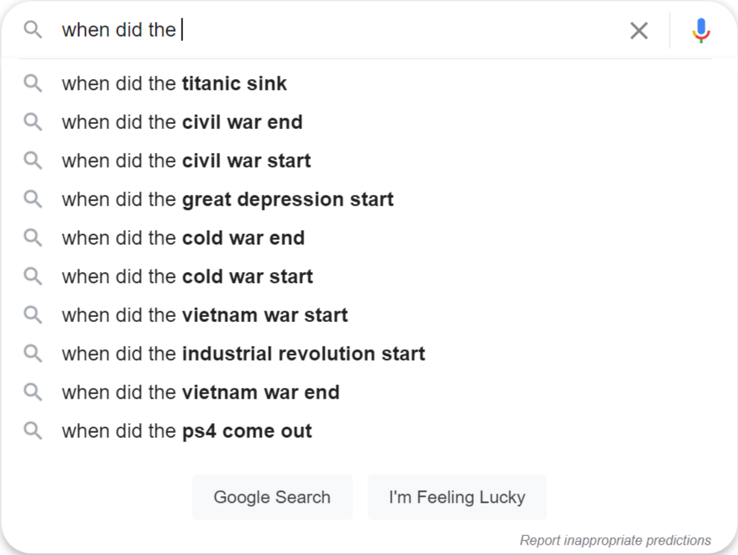
Relevance and the “information need”



Typical information needs vary by task

- Each search task poses unique challenges!
  - Many of them lack key features that make Web search work.

- Unlike, say, Slack search, Web search can often rely on lots of:
  - Popular "head" queries
  - Redundant documents on common topics
  - Explicit (hyper)links between documents



Where does NLP fit in IR?

- Queries and documents are often expressed in natural language.
- Due to **vocabulary mismatch**, lexical matching doesn't suffice!

what compounds protect the digestive system against viruses

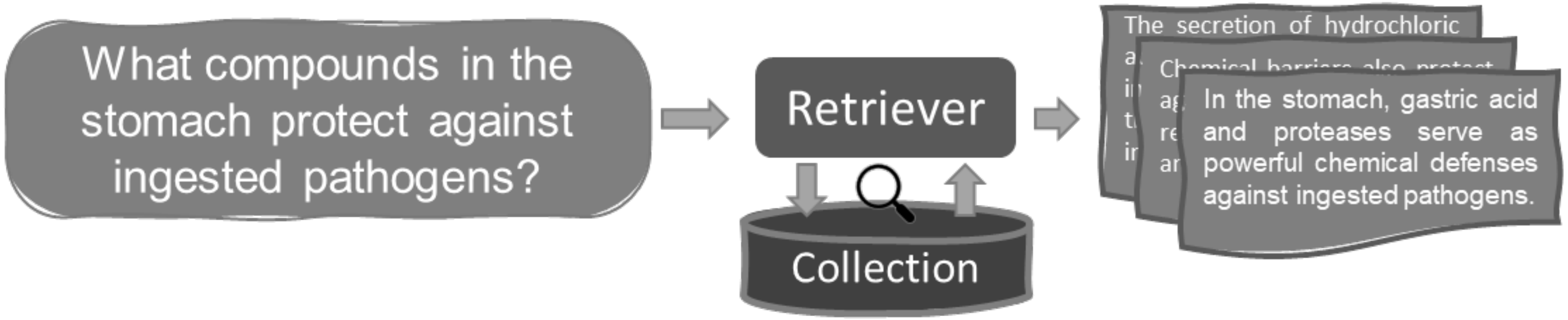
→

In the **stomach**, gastric acid and proteases serve as powerful **chemical defenses** against ingested **pathogens**.

Classical IR

Ranked retrieval

- Scope: A large corpus of text documents (e.g., Wikipedia)
- Input: A textual query (e.g., a natural-language question)
- Output: **Top-K Ranking** of **relevant** documents (e.g., top-100)



How do we conduct ranked retrieval?

- One approach: the **Term-Document Matrix**

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
against	0	0	0	1	0	0	3	2	3	0
age	0	0	0	1	0	3	1	0	4	0
agent	0	0	0	0	0	0	0	0	0	0
ages	0	0	0	0	0	2	0	0	0	0
ago	0	0	0	2	0	0	0	0	3	0
agree	0	1	0	0	0	0	0	0	0	0
ahead	0	0	0	1	0	0	0	0	0	0
ain’t	0	0	0	0	0	0	0	0	0	0
air	0	0	0	0	0	0	0	0	0	0
aka	0	0	0	1	0	0	0	0	0	0

- With the right weights, this lets us answer **single-term** queries!

How do we conduct ranked retrieval?

- For multi-term queries, classical IR models tokenize and then treat the tokens independently.

$$RelevanceScore(query, doc) = \sum_{term \in query} Weight_{doc,term}$$

- This reduces a large fraction of classical IR to:
  - How do we best tokenize (and stem) queries and documents
  - **How do we best weight each term-document pair**

Term–document weighting: intuitions

- **Frequency** of occurrence will be a primary factor
  - If a term t occurs frequently in document d, the document is more likely to be relevant for queries including t

- **Normalization** is also an important component
  - If that term t is common overall, then don't give it high scores
  - If a document d is shorter, this improves its score for each term

- Amplify the important, the trustworthy, the unusual; deemphasize the mundane and the quirky.

Term–document weighting: TF-IDF

Term Frequency, Inverse Document Frequency

- Let $N = |Collection|$ and $df(term) = |\{doc \in Collection : term \in doc\}|$

$$TF(term, doc) = \log(1 + Freq(term, doc))$$

$$IDF(term) = \log \frac{N}{df(term)}$$

TF and IDF grow **sub-linearly** with frequency and with 1/df

$$TF.IDF(term, doc) = TF(term, doc) \times IDF(term)$$

$$TF.IDF(query, doc) = \sum_{term \in query} TF.IDF(term, doc)$$

Term–document weighting: BM25

"Best Match, attempt #25"

$$IDF(term) = \log(1 + \frac{N - df(term) + 0.5}{df(term) + 0.5})$$

$$TF(term, doc) = \frac{Freq(term, doc) \times (k + 1)}{Freq(term, doc) + k \times (1 - b + b \times \frac{|doc|}{avgdoclen})}$$

$$BM25(term, doc) = BM25{:}\,TF(term, doc) \times BM25{:}\,IDF(term)$$

$$BM25\ (query, doc) = \sum_{term \in query} BM25\ (term, doc)$$

k, b are parameters.

Unlike TF-IDF, term frequency in BM25 **saturates** and **penalizes longer documents**!

Robertson, Stephen, and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. Now Publishers Inc, 2009.

Efficient implementation: inverted indexing

- Term-document matrix: Term → Documents
  - But it's extremely sparse and thus wastes space!

- An **inverted index** is just a sparse encoding of this matrix
  - Mapping each unique term t in the collection to a "posting list"
  - The posting list enumerates **non-zero** <Freq, DocID> for t

Beyond term matching in classical IR…

- Query and Document expansion
- Term dependence and phrase search
- Learning to Rank with various features:
  - Different document fields (e.g., title, body, anchor text)
  - Link Analysis (e.g., PageRank)

Lots of IR exploration into these!
However, BM25 was a very strong baseline on the best you can do “ad-hoc”—until 2019 with BERT-based ranking!

IR Evaluation

- A search system must be **efficient** and **effective**

- Efficiency
  - **Latency (milliseconds; for one query)**
  - Throughput (queries/sec)
  - Space (GBs for the index? TBs?)
  - Hardware cost (one CPU core? Many cores? GPUs?)
  - Scaling

IR Effectiveness

- Do our top-k rankings fulfill users' information needs?
  - Often harder to evaluate than classification/regression!

- If you have lots of users, you can run online experiments…

- But we're typically interested in evaluating on **test collections**
  - These can be very hard to build based on the domain! Need to get people to evaluate many documents for relevance.

IR Effectiveness Metrics

- We'll use "metric"@K, often with K in {5, 10, 100, 1000}.
  - Selection of the metric (and the cutoff K) depends on the task.

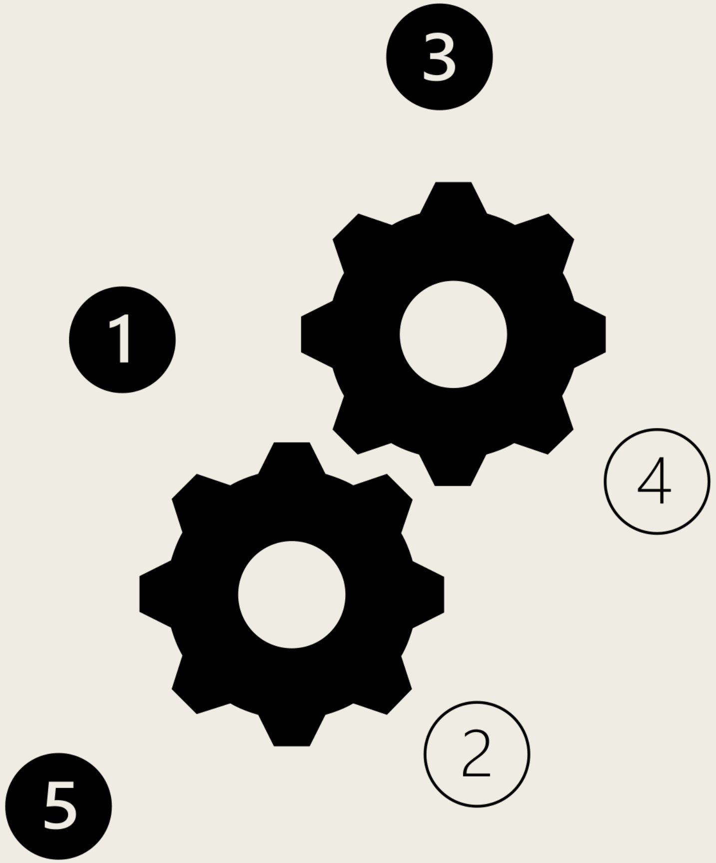
- For all metrics here, we'll average across all test queries.

Common Metrics: Success & MRR

- Let $rank \in \{1, 2, 3, \ldots\}$ be the position of the <u>first</u> relevant document

- Success@K = $\begin{cases} 1 & \text{if } rank \le K \\ 0 & \text{otherwise} \end{cases}$

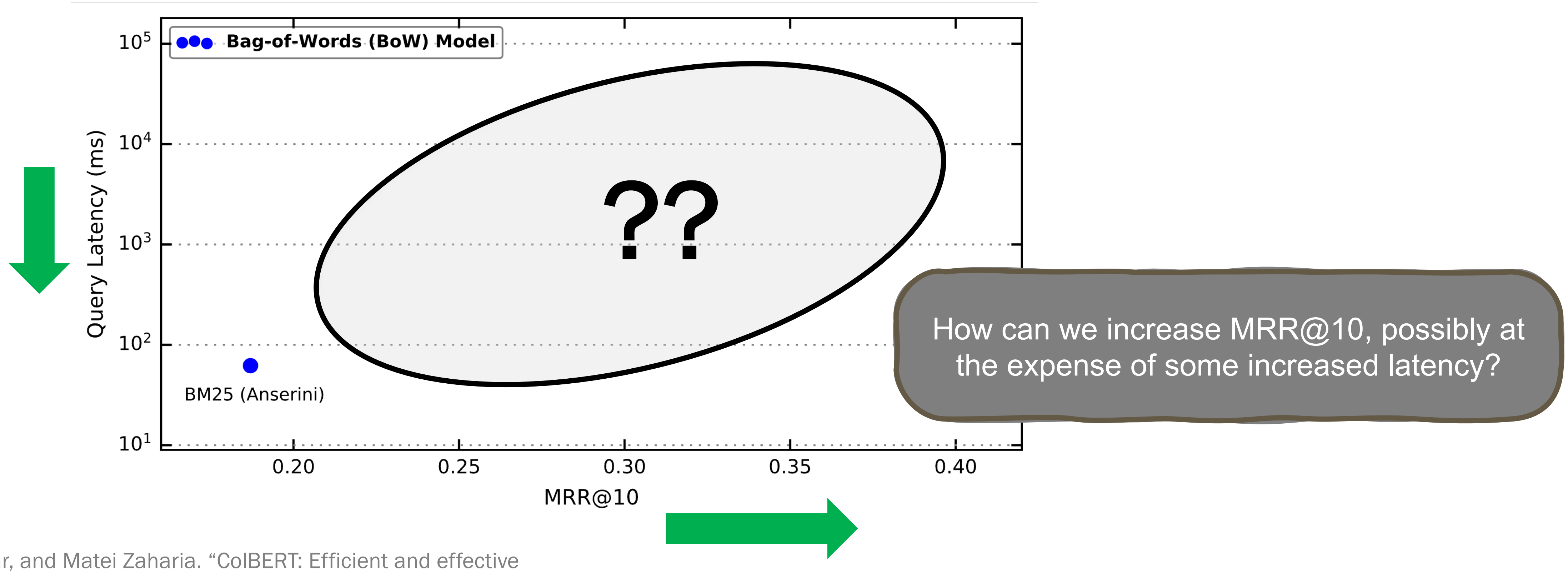
- ReciporcalRank@K = $\begin{cases} 1/rank & \text{if } rank \le K \\ 0 & \text{otherwise} \end{cases}$
  - Mean Reciprocal Rank (MRR) is the average of ReciprocalRank across test set



Neural IR

Efficiency–Effectiveness Tradeoff

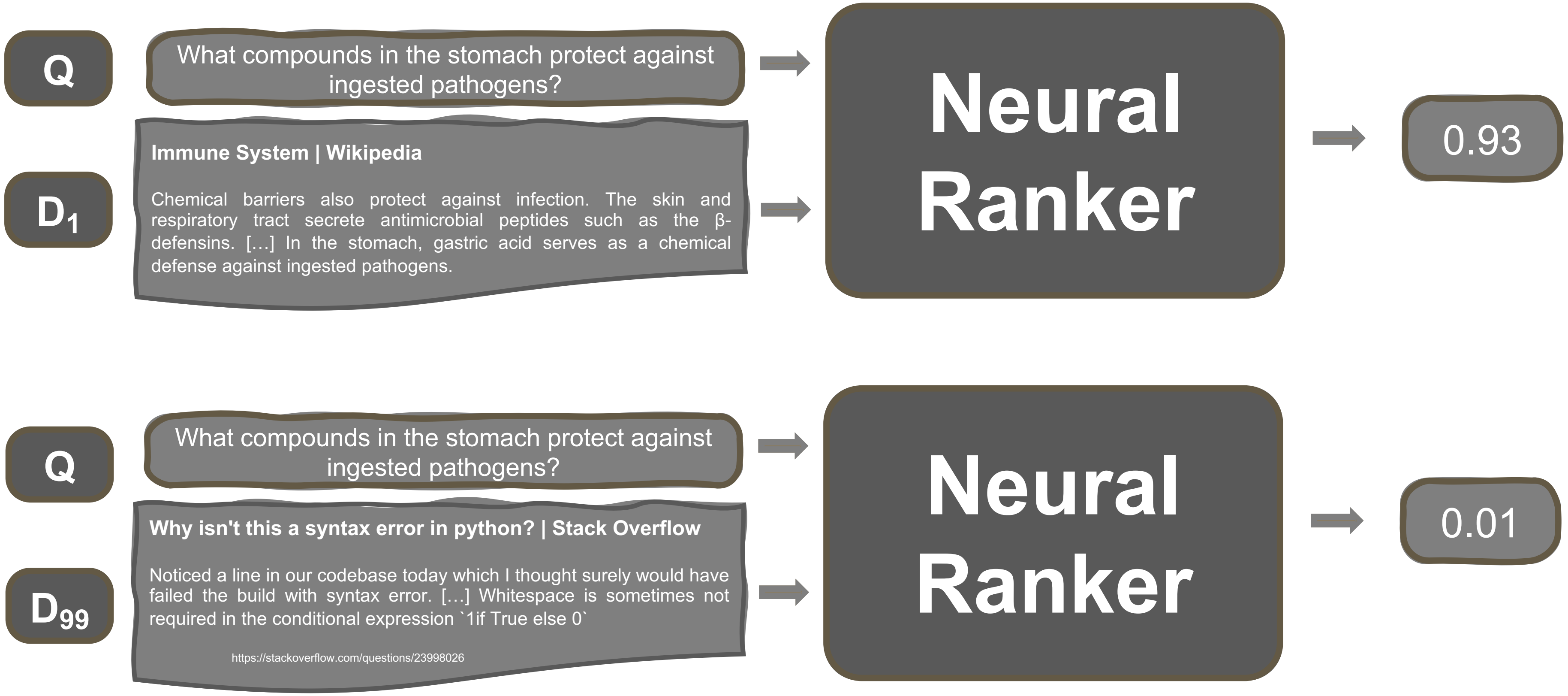
- MS MARCO: Bing Queries, 9M Passages from the Web
  - Effectiveness in **MRR@10** and Efficiency in Latency (**milliseconds**; in log-scale!)



Khattab, Omar, and Matei Zaharia. "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT." SIGIR'20.

Neural Ranking: Functional View

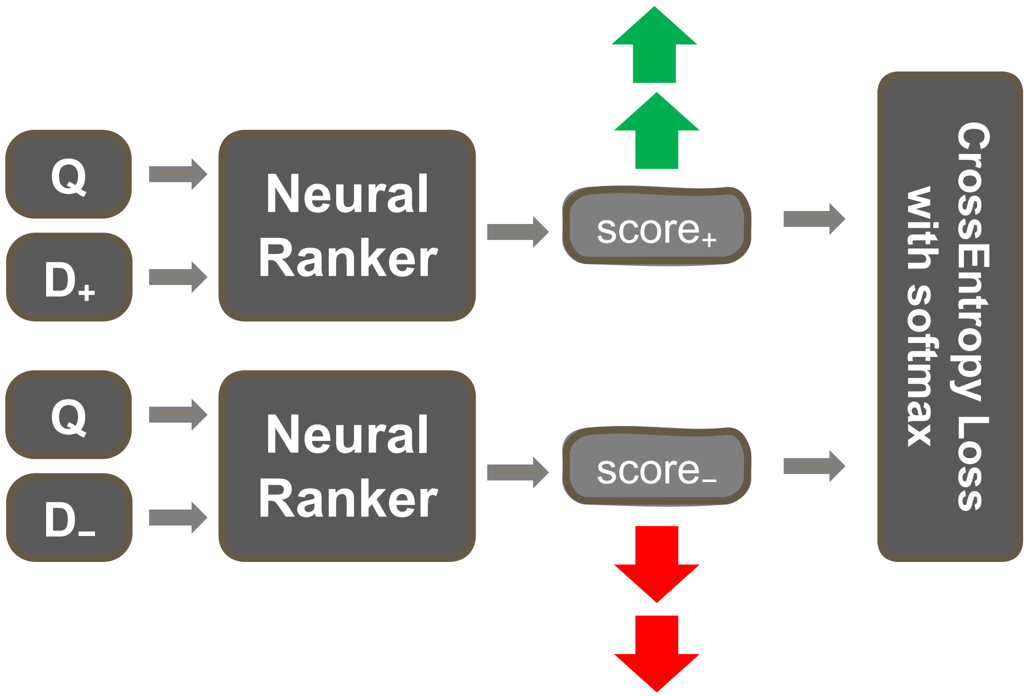
- All we need is a score for every query–document pair
  - We'll sort the results by decreasing score



Neural Ranking: Training

- Many possible choices, but **2-way classification** is often effective!
  - Each training instance is a **triple**

    < query, positive document, negative document >

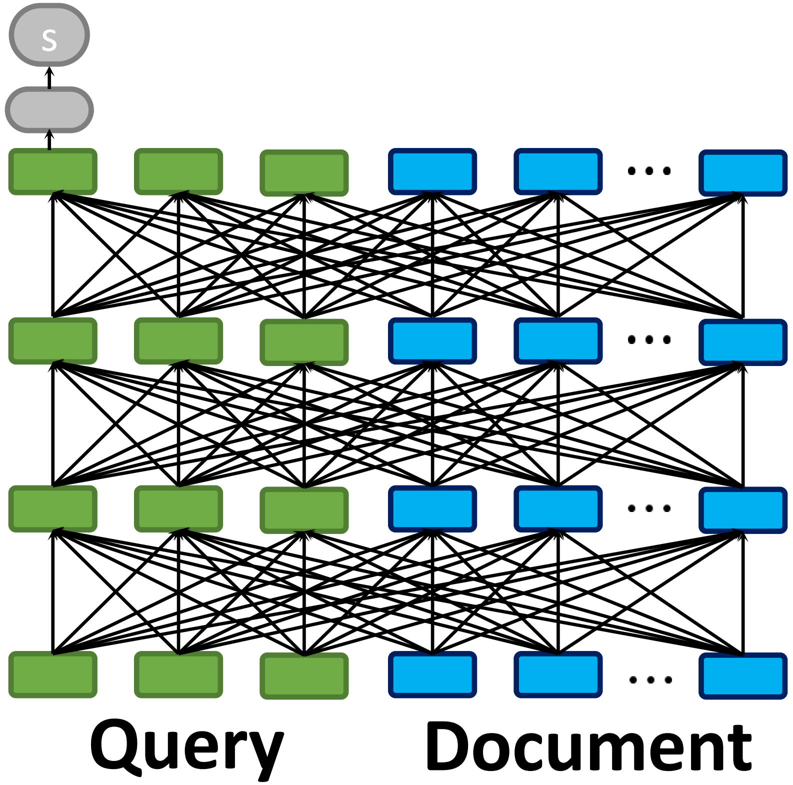


We can get positives for each query from our human relevance assessments.

Every non-positive can often be treated as an implicit negative.

“All-to-all interaction” ranking with BERT

1. Feed BERT “[CLS] Query [SEP] Document [SEP]”
2. Run this through all the BERT layers
3. Extract the final [CLS] output embedding
   - Reduce to a single score through a linear layer



This is essentially a standard BERT classifier, used for ranking passages.

Of course, we must fine-tune BERT for this task with positives and negatives.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. arXiv:1901.04085 (2019)
Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. SIGIR’19

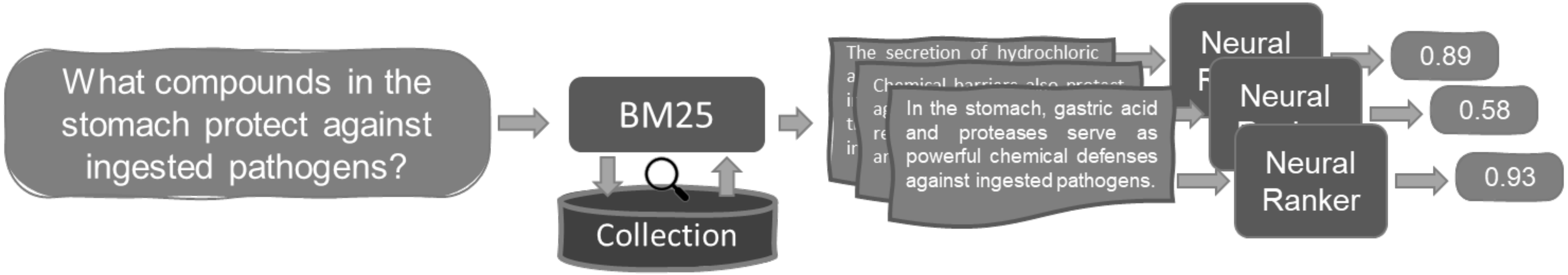
Neural Ranking: Inference

- Given a query Q, pick each document d and pass $<Q, d>$ through the network. Sort all by score, returning the top-k results!

- But collections often have many millions of documents
  - MS MARCO has 9M passages
  - Even if you model runs in 1 millisecond per passage, that’s 9000 seconds per query!

Neural Re-ranking

■ BM25 top-1000 -> Neural IR re-ranker



■ Cuts the work on 10M documents by factor of 10k!
  – But introduces an artificial recall ceiling.

Can we do better?

Yes! Later, we'll discuss **end-to-end retrieval.**

BERT Re-rankers: SOTA in quality (2019)

Rank	Model	Submission Date	MRR@10 On Eval
1	**BERT + Small Training** Rodrigo Nogueira and Kyunghyun Cho - New York University	January 7th, 2019	35.87
2	**IRNet (Deep CNN/IR Hybrid Network)** Dave DeBarr, Navendu Jain, Robert Sim, Justin Wang, Nirupama Chandrasekaran – Microsoft	January 2nd, 2019	28.061

Google The Keyword

SEARCH

Understanding searches better than ever before

Pandu Nayak

Google Fellow and Vice President, Search

Published Oct 25, 2019

If there's one thing I've learned over the 15 years working on Google Search, it's that people's curiosity is endless.

Microsoft Azure

Blog / Virtual Machines

Bing delivers its largest improvement in search experience using Azure GPUs

Posted on November 18, 2019

Jeffrey Zhu

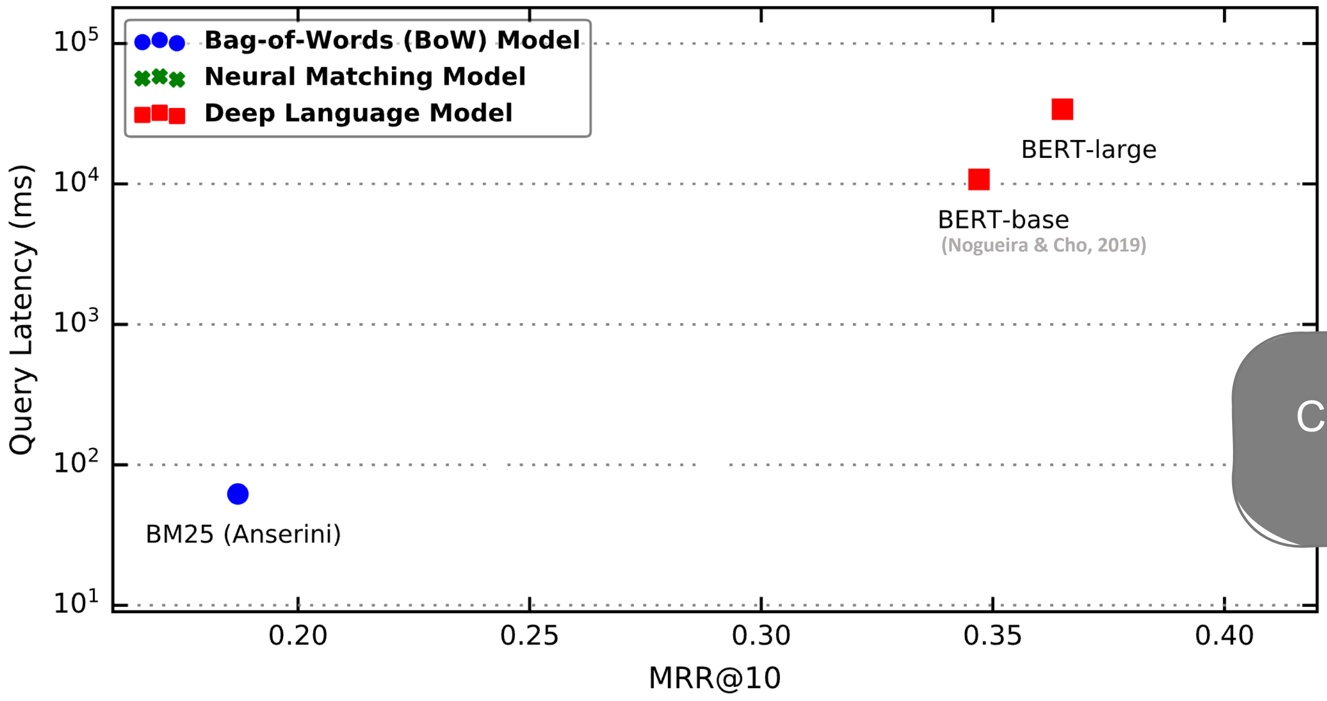
Program Manager, Bing Platform

Over the last couple of years, deep learning has become widely adopted across the Bing search stack and powers a vast number of our intelligent features. We use natural language models to improve our core search

MS MARCO Ranking screenshot as of Jan 2019. From Rodrigo Nogueira's Brief History of DL applied to IR (UoG talk).
https://blog.google/products/search/search-language-understanding-bert/
https://azure.microsoft.com/en-us/blog/bing-delivers-its-largest-improvementin-search-experience-using-azure-gpus/

BERT Re-rankers: efficiency-effectiveness tradeoff

- Dramatic gains in **quality**—but also a dramatic increase in **computational cost**!

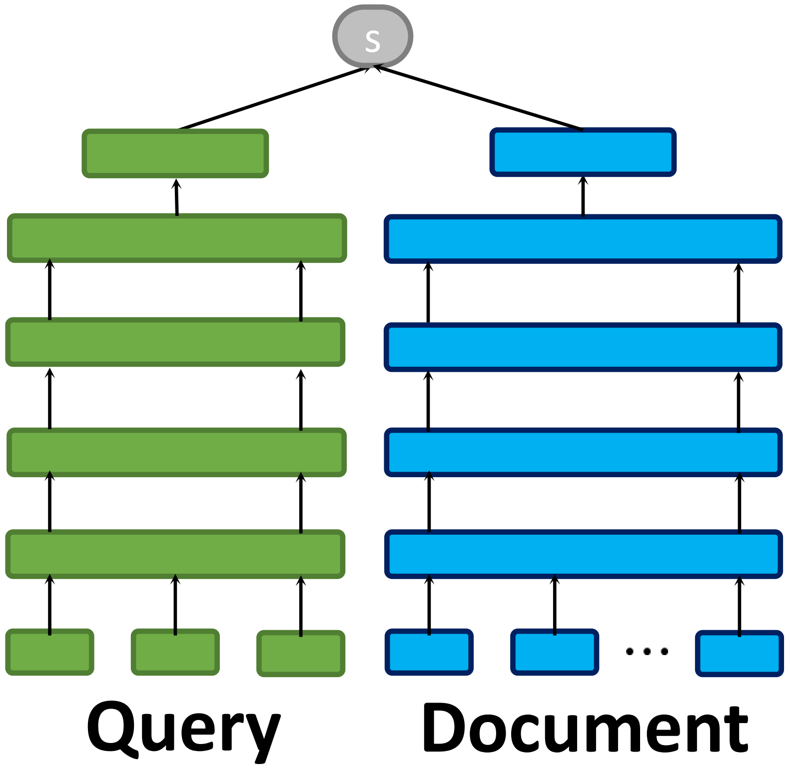


Can we achieve high MRR ***and*** low latency?

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. arXiv:1901.04085 (2019)

Neural IR paradigms: **Representation Similarity**

- Tokenize the query and the document
- **<u>Independently</u>** encode the query and the document
- … into a **<u>single-vector</u>** representation each
- Estimate relevance a dot product
  - Or a cosine similarity



Like learning term weights, this paradigm offers strong **efficiency** advantages:

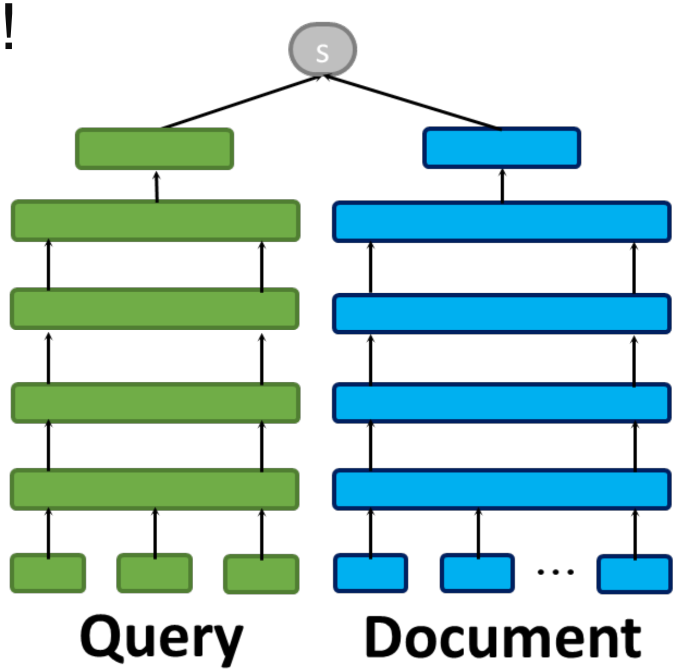
✓ Document representations can be pre-computed!

✓ Query computations can be amortized.

✓ Similarity search is cheap with good data structures.

Representation similarity: Models

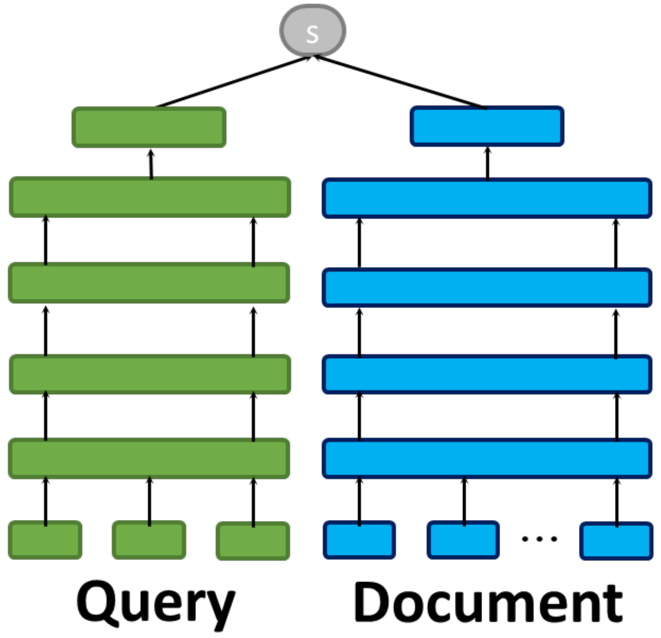
- Many pre-BERT IR models fall under this paradigm!
  - DSSM and SNRM
- Numerous BERT-based models exist
  - SBERT, **DPR**, ORQA, DE-BERT, RepBERT, ANCE
- Approximate Nearest Neighbor data structures (a.k.a. vector DBs) can efficiently do search
  - **HNSW**, LSH, PQ, …



Example: DPR

Dense Passage Retriever (DPR) by *Karpukhin et al.*

- Encodes each passage into a 768-dimensional vector
- Encodes each query into a 768-dimensional vector



- Trained with N-way cross-entropy loss, over the similarity scores between the query and:
  - A positive passage
  - A negative passage, sampled from BM25 top-100
  - Many in-batch negative passages
    - the positive passages for the *other* queries in the same training batch

Representation Similarity: Downsides

❌ **Single-Vector Representations**

- They “cram” queries and documents into a **coarse-grained** representation!

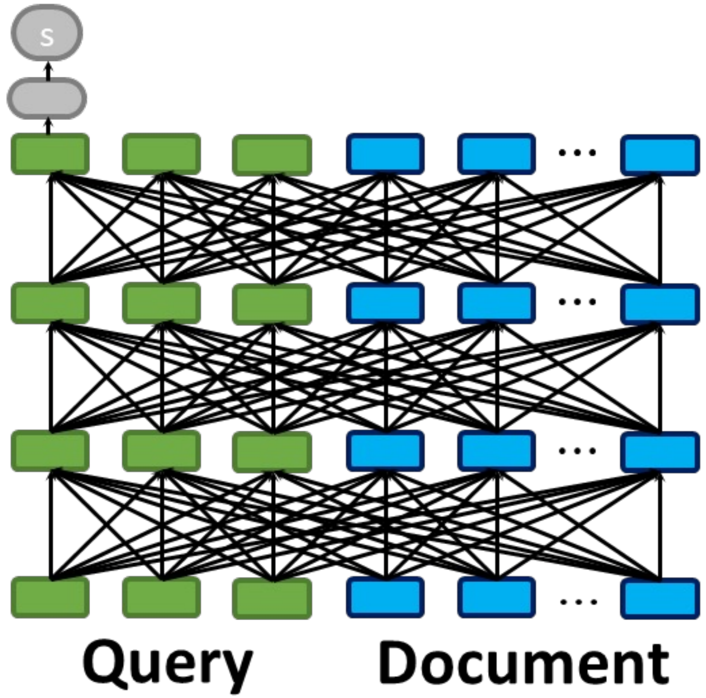
❌ **No Fine-Grained Interactions**

- They estimate relevance as **single dot product**!
- We lose **term-level interactions**, which we had in:
  - Query-Document interaction models (e.g., BERT or Duet)
  - And even term-weighting models (e.g., BM25)

Can we keep precomputation and still have fine-grained interactions?

Omar Khattab and Matei Zaharia. “ColBERT: Efficient and effective passage search via contextualized late interaction over BERT." SIGIR’20.

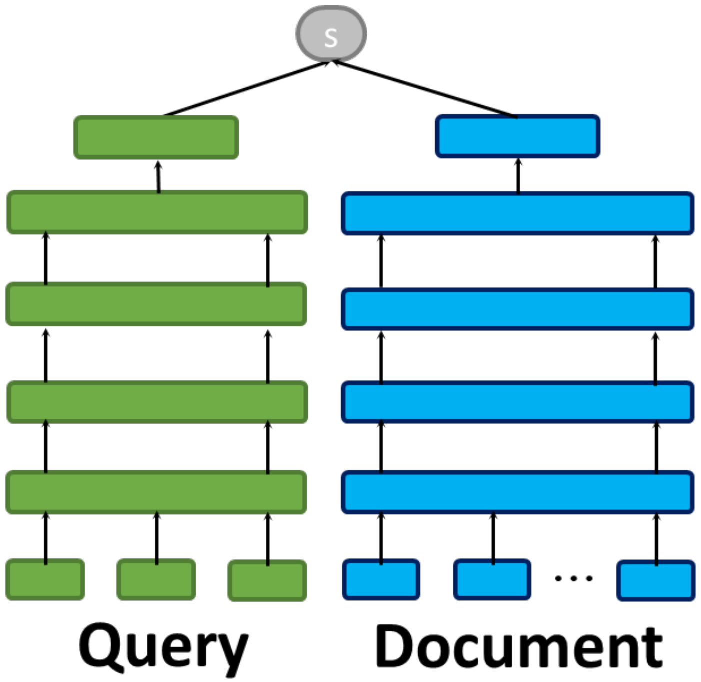
Neural IR paradigms so far



(a) Query–Document Interaction

✓ Fine-Grained Interactions

✗ Expensive Joint Conditioning

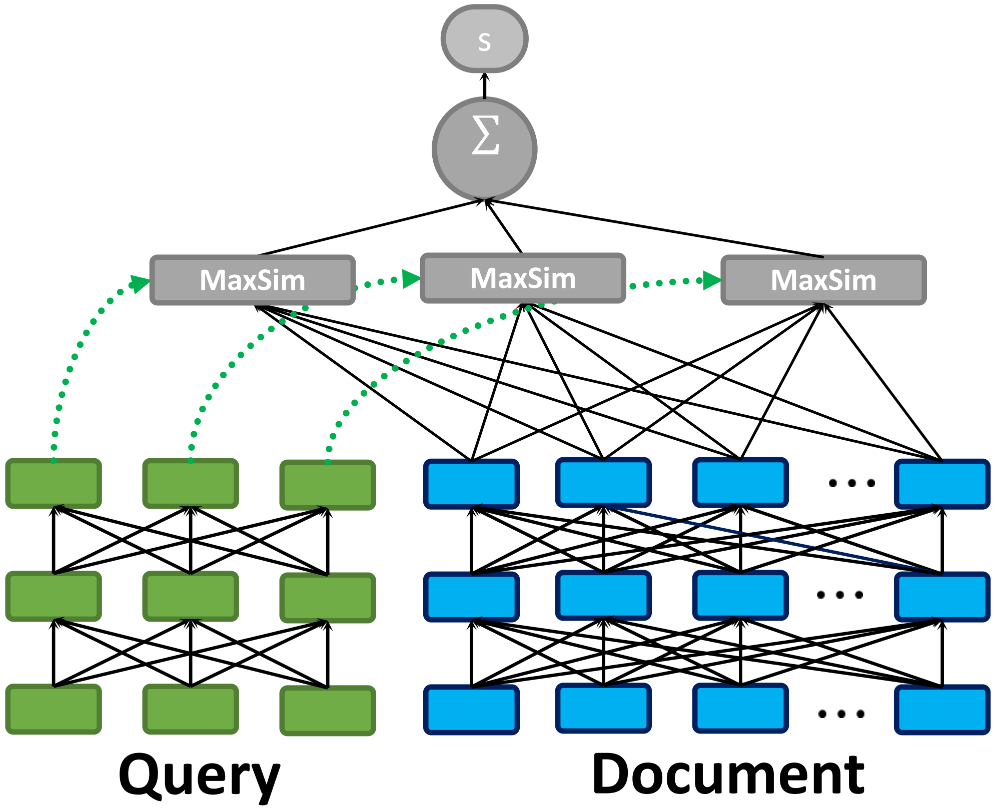


(b) Representation Similarity

✓ Independent, Dense Encoding

✗ Coarse-Grained Representation

Neural IR paradigms: **Late interaction**



ColBERT

Can we keep precomputation and still have fine-grained interactions?

✓ Independent Encoding

✓ Fine-Grained Representations

✓ Can do end-to-end retrieval on a full collection (ANN + pruning)

Omar Khattab and Matei Zaharia. "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT." SIGIR'20.

Late interaction: real example of matching

when did the transformers cartoon series come out?

[…] the animated […] The Transformers […] […] It was released […] on August 8, 1986

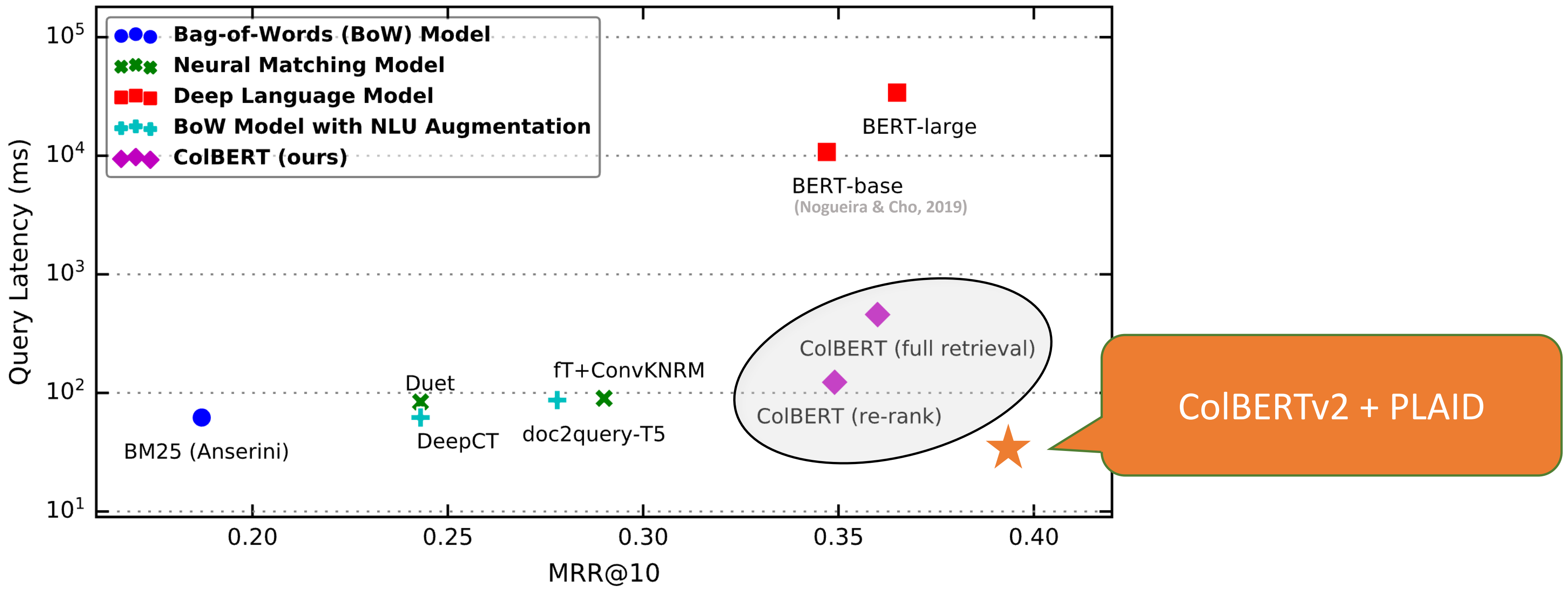
when did the transformers cartoon series come out?

[…] the animated […] The Transformers […] […] It was released […] on August 8, 1986

when did the transformers cartoon series come out?

[…] the animated […] The Transformers […] […] It was released […] on August 8, 1986

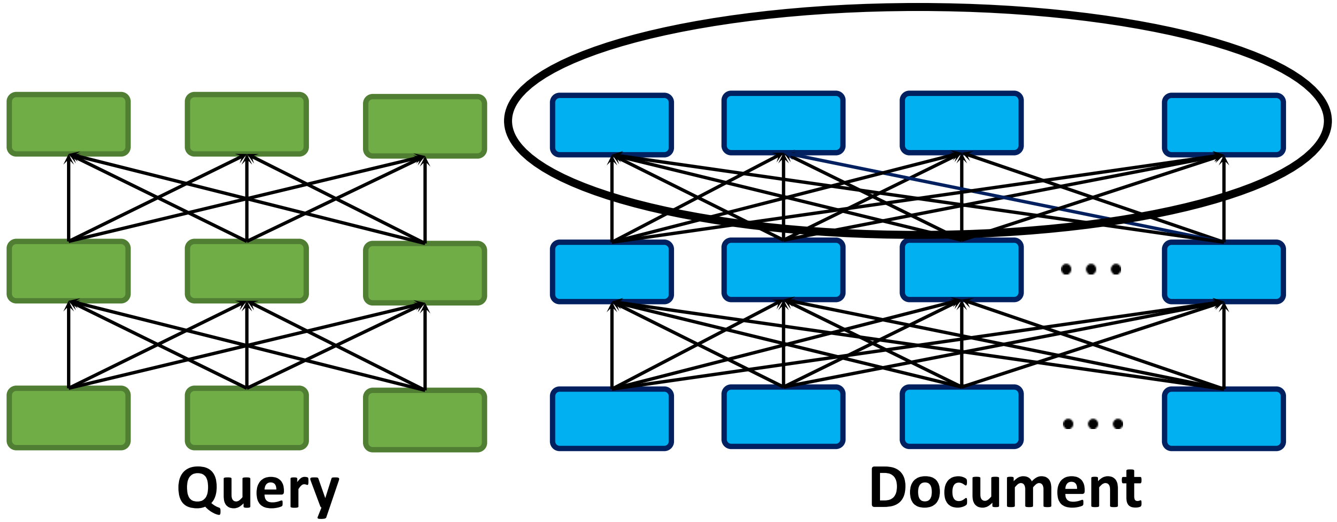
Late interaction: ColBERT Results



Omar Khattab and Matei Zaharia. "ColBERT: Efficient and effective passage search via contextualized late interaction over BERT." SIGIR'20.

Relation to term-document matrices

- ColBERT represents each doc as a matrix of term **embeddings**, instead of a vector of term **weights**.



Late interaction (ColBERT)

Robustness: Out-of-domain quality

- So far, we've looked at <u>in-domain</u> effectiveness evaluations.
  - We had training and evaluation data for MS MARCO.

- We often want to use retrieval in new, out-of-domain settings.
  - … with NO training data and NO validation data.
  - This is sometimes called a "zero-shot" setting; it emphasizes transfer.

- BEIR is a recent benchmark for IR models in "zero-shot" scenarios

Thakur, Nandan, et al. "BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models." *arXiv:2104.08663* (2021)

Robustness: Out-of-domain NDCG@10

- **Fine-grained interaction** is key to robustly high precision

IR Task	Classical IR **BM25**	Interaction Models **ELECTRA re-ranker**	Representation Similarity **DPR**	Representation Similarity **SBERT**	Late Interaction **ColBERT**
BioMed	48	**49**	22	34	**49**
QA	38	**51**	33	41	48
Tweet	**39**	31	16	26	27
News	37	**43**	16	37	39
Arguments	**52**	35	15	34	25
Duplicates	53	56	20	58	**60**
Entity	29	38	26	34	**39**
Citation	**16**	15	8	13	15
Fact-Check	48	52	34	47	**54**
Overall Avg	42	**45**	23	39	44

Table aggregated from the BEIR results (Table 2) by Thakur, Nandan, et al. "BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models." *arXiv:2104.08663* (2021)

Final Thoughts on IR

- IR quality is essential to making retrieval-based LLM apps work
  - Biggest problem for quality in production RAG apps is often retrieval

- Tuning on each domain improves quality, but there is research on retrievers that work well out-of-domain

- Speed matters! Achieved via inductive biases in model design