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
In the stomach, **gastric acid and proteases** serve as powerful chemical defenses against ingested pathogens. [1] Wikipedia - Immune system

**Passage [1]:**

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

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!

![Diagram](image.png)
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**

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
<th>d7</th>
<th>d8</th>
<th>d9</th>
<th>d10</th>
</tr>
</thead>
<tbody>
<tr>
<td>against</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>age</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>agent</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ages</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ago</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>agree</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>ahead</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ain’t</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>air</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>aka</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- 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 $df(\text{term}) = |\{ doc \in Collection : \text{term} \in doc \}|$

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

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

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

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

TF and IDF grow sub-linearly with frequency and with $1/df$
Term-document weighting: BM25

“Best Match, attempt #25”

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

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

\[
BM25(\text{term}, \text{doc}) = BM25: TF(\text{term}, \text{doc}) \times BM25: IDF(\text{term})
\]

\[
BM25(\text{query}, \text{doc}) = \sum_{\text{term} \in \text{query}} BM25(\text{term}, \text{doc})
\]

k, b are parameters.

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

Efficient implementation: inverted indexing

■ Term-document matrix: Term $\rightarrow$ 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 first relevant document

- \[ \text{Success}@K = \begin{cases} 1 & \text{if } rank \leq K \\ 0 & \text{otherwise} \end{cases} \]

- \[ \text{ReciprocalRank}@K = \begin{cases} 1/rank & \text{if } rank \leq K \\ 0 & \text{otherwise} \end{cases} \]
  - 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!)

How can we increase MRR@10, possibly at the expense of some increased latency?

Neural Ranking: Functional View

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

**Q** What compounds in the stomach protect against ingested pathogens?
*Immune System | Wikipedia*
Chemical barriers also protect against infection. The skin and respiratory tract secrete antimicrobial peptides such as the β-defensins. [...] In the stomach, gastric acid serves as a chemical defense against ingested pathogens.

**D1**

**Q** What compounds in the stomach protect against ingested pathogens?
*Why isn’t this a syntax error in python? | Stack Overflow*
Noticed a line in our codebase today which I thought surely would have failed the build with syntax error. [...] Whitespace is sometimes not required in the conditional expression “if True else 0”

**D99**
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

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.

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)

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

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/
BERT Re-rankers: efficiency-effectiveness tradeoff

- Dramatic gains in **quality**—but also a dramatic increase in **computational cost**!
Neural IR paradigms: Representation Similarity

- Tokenize the query and the document
- **Independently** encode the query and the document
- … into a **single-vector** 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

Vladimir Karpukhin, et al. "Dense passage retrieval for open-domain question answering." EMNLP'20
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?

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

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)

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

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 in-domain 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

Robustness: Out-of-domain NDCG@10

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

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

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