NLU & IR: CLASSICAL IR

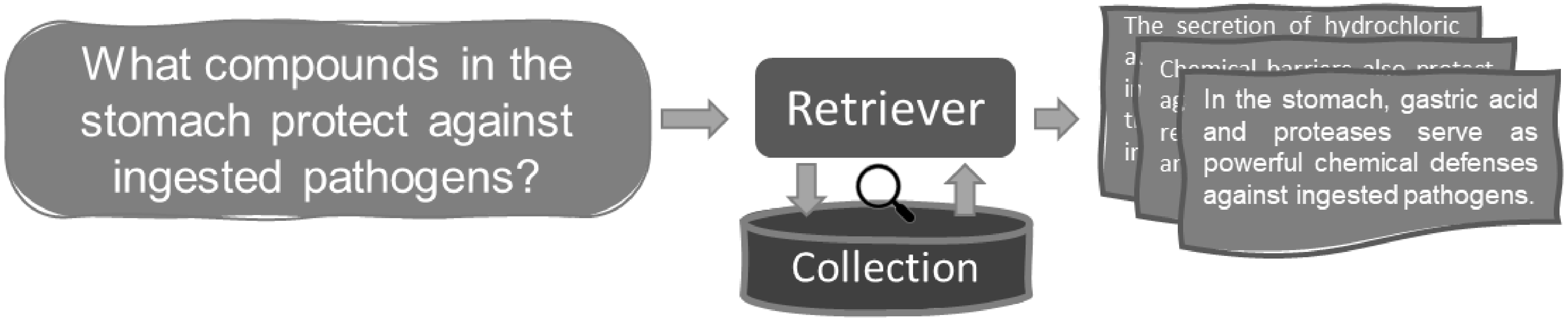
Omar Khattab

CS224U: Natural Language Understanding

Spring 2021

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?

- We’ve touched on one way before: 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 good weights, this allows us to answer **single-term** queries!

How do we conduct ranked retrieval?

- For multi-term queries, classical IR models would 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 remain 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** will remain a primary component too
  - If that term t is rather rare, then document d is even more likely to be relevant for queries including t
  - If that document d is rather short, this also improves its odd

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

Term–Document Weighting: TF-IDF

- 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 both grow
sub-linearly with frequency and 1/df
(in particular, logarithmically).

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

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

Term–Document Weighting: BM25

Or “Finding the best match, *seriously* this time! Attempt #25” :-)

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

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

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

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

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 Retrieval: Inverted Indexing

- Raw Collection: Document → Terms
- Term–document matrix: Term -> Documents
  - But it's extremely sparse and thus wastes space!
- The **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**
  - If we had infinite resources, we'd just hire experts to look through all the documents one by one!

- Efficiency
  - **Latency (milliseconds; for one query)**
  - Throughput (queries/sec)
  - Space (GBs for the index? TBs?)
  - Hardware required (one CPU core? Many cores? GPUs?)
  - Scaling to various collection sizes, under different loads

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 reusable **test collections**

Test Collections

- Document Collection (or “Corpus”)
- Test Queries (or “Topics”)
  - Could also include a train/dev split, if resources allow!
  - Or, in some cases, cross-validation could be used.
- Query–Document Relevance Assessments
  - Is document j relevant to query i?
    - Binary judgments: relevant (0) vs. non-relevant (1)
    - Graded judgments: {-1, 0, 1, 2} (e.g., junk, irrelevant, relevant, key)

We typically have to make the (significant!) assumption that unjudged documents are irrelevant. Some test collections would only label a few positives per query.

Test Collections: TREC

- Text REtrieval Conference (TREC) includes numerous annual tracks for comparing IR systems.

- The 2021 iteration has tracks for Conversational Assistance, Health Misinformation, Fair Ranking, “Deep Learning”.

- TREC tends to emphasize careful evaluation with a <u>very</u> small set of queries (e.g., 50 queries, each with >100 annotated documents)
  - Having only few test queries does <u>not</u> imply few documents!

Test Collections: MS MARCO Ranking Tasks

- MS MARCO Ranking is the largest public IR benchmark
  - adapted from a Question Answering dataset
  - consists of more than 500k **Bing search queries**
    - Sparse labels: approx. <u>one</u> relevance label per query!
    - Fantastic for training IR models!

MS MARCO

- MS MARCO Passage Ranking (9M short passages; sparse labels)
- MS MARCO Document Ranking (3M long documents; sparse labels)
- TREC DL’19 and DL’20 (short&long; dense labels for few queries)

Test Collections: Other Benchmarks

- Lots of small or domain-specific benchmarks!
- BEIR is a recent effort to use those for testing models in “zero-shot” scenarios

We will also see later that OpenQA benchmarks can serve as large IR benchmarks too!

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

Split (→)					Train	Dev	Test			(Train + Dev + Test)	
Task (↓)	**Domain (↓)**	**Dataset (↓)**	**Title**	**Relevancy**	**#Pairs**	**#Query**	**#Query**	**#Corpus**	**Avg. Docs / Q**	**Avg. Q Len**	**Avg. Doc Len**
Passage-Retrieval	Misc.	MSMARCO	✗	Binary	532,761	—	6,980	8,841,823	1.1	5.96	55.98
Bio-Medical Information Retrieval (IR)	Bio-Medical	(1) TREC-COVID	✓	3-level	—	—	50	171,332	493.5	10.60	160.77
	Bio-Medical	(2) NFCorpus	✓	3-level	110,575	324	323	3,633	38.2	3.30	232.26
	Bio-Medical	(3) BioASQ	✓	Binary	32,916	—	500	14,914,602	4.7	8.05	202.61
Question Answering (QA)	Wikipedia	(4) NQ	✓	Binary	132,803	—	3,452	2,681,468	1.2	9.16	78.88
	Wikipedia	(5) HotpotQA	✓	Binary	170,000	5,447	7,405	5,233,329	2.0	17.61	46.30
	Finance	(6) FiQA-2018	✗	Binary	14,166	500	648	57,638	2.6	10.77	132.32
Tweet-Retrieval	Twitter	(7) Signal-1M (RT)	✗	3-level	—	—	97	2,866,316	19.6	9.30	13.93
News-Retrieval	News	(8) TREC-NEWS	✓	5-level	—	—	57	594,977	19.6	11.14	634.79
Argument Retrieval	Misc.	(9) ArguAna	✓	Binary	—	—	1,406	8,674	1.0	192.98	166.80
	Misc.	(10) Tóuche-2020	✓	6-level	—	—	49	382,545	49.2	6.55	292.37
Duplicate-Question Retrieval	StackEx.	(11) CQADupStack	✓	Binary	—	—	13,145	457,199	1.4	8.59	129.09
	Quora	(12) Quora	✗	Binary	—	5,000	10,000	522,931	1.6	9.53	11.44
Entity-Retrieval	Wikipedia	(13) DBPedia	✓	3-level	—	67	400	4,635,922	38.2	5.39	49.68
Citation-Prediction	Scientific	(14) SCIDOCS	✓	Binary	—	—	1,000	25,657	4.9	9.38	176.19
Fact Checking	Wikipedia	(15) FEVER	✓	Binary	140,085	6,666	6,666	5,416,568	1.2	8.13	84.76
	Wikipedia	(16) Climate-FEVER	✓	Binary	—	—	1,535	5,416,593	3.0	20.13	84.76
	Scientific	(17) SciFact	✓	Binary	920	—	300	5,183	1.1	12.37	213.63

Table 1: Statistics of all the tasks, domains and datasets included in **BEIR**. Few datasets contain documents without titles. Relevancy column indicates the relation between the query and document: binary (relevant, irrelevant) or further graded into sub-levels. Avg. Docs/Query column indicates the average relevant documents per question.

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 [macro-]average across all queries.
  - All queries will be assigned equal weight, for our purposes.

IR Effectiveness Metrics: Success & MRR

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

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

- ReciporcalRank@K = $\begin{cases} 1/rank & \text{if } rank \leq K \\ 0 & \text{otherwise} \end{cases}$

  – This is MRR (M for “mean”), but dropped the M as we’re looking at only one query

IR Effectiveness Metrics: Precision & Recall

- Let $Ret(K)$ be the top-K retrieved documents
- Let Rel be the set of all documents judged as relevant

- Precision@K = $\frac{|Ret(K) \cap Rel|}{K}$

- Recall@K = $\frac{|Ret(K) \cap Rel|}{|Rel|}$

IR Effectiveness Metrics: MAP

- (M)AP = (Mean) Average Precision

- Let $rank_1, rank_2, \dots, rank_{|Rel|}$ be the positions of all relevant documents
  - Compute precision@i at each of those positions—and average!

- Equivalently, AveragePrecision@K =

$$\frac{\sum_{i=1}^{K} \begin{cases} Precision@i & \text{if } relevant?(i^{th}\ document) \\ 0 & \text{otherwise} \end{cases}}{|Rel|}$$

IR Effectiveness Metrics: DCG

- Discounted Cumulative Gain
  - Not inherently normalized, so we also consider Normalized DCG

$$DCG@K = \sum_{i=1}^{K} \frac{graded\_relevance(i^{th}\ document)}{\log_2(i+1)}$$

$$NDCG@K = \frac{DCG@K}{ideal\ DCG@K}$$

Next...

- Neural IR.

References

Manning, Christopher, Prabhakar Raghavan and Schutze, H. “Introduction to Information Retrieval.” (2008).

Manning, Christopher, and Pandu Nayak (2019). CS276 Information Retrieval and Web Search: Evaluation [Class handout]. Retrieved from http://web.stanford.edu/class/cs276/19handouts/lecture8-evaluation-6per.pdf

Hofstätter, Sebastian. Advanced Information Retrieval: {IR Fundamentals, Evaluatin, Test Collections} [Class handout]. Retrieved from https://github.com/sebastian-hofstaetter/teaching

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

Nguyen, Tri, et al. "MS MARCO: A human generated machine reading comprehension dataset." CoCo@ NIPS. 2016.

Craswell, Nick, et al. "TREC Deep Learning Track: Reusable Test Collections in the Large Data Regime." arXiv preprint arXiv:2104.09399 (2021).

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