CS657A: Information Retrieval
Scored Retrieval Model

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs657/

2nd semester, 2021-22

Tue 1030-1145, Thu 1200-1315

Non-Boolean Match

- A query may be more general than just a set of terms
- It may itself be another document or some free text
- No document can then be expected to match it fully
- Information retrieval task
  - Given a free text query, find the most similar documents
- "Similarity" of a document d with the query q can be measured by a score $s(d, q)$
- Scoring information retrieval task
  - Given a free text query, find the documents that are most similar, i.e., have the largest scores

Zones

- Each document is generally associated with metadata, e.g., title, author, date, etc.
- Sometimes, queries on indexes of these fields, called parametric indexes, are also asked
    - Find documents where "Indian" appears in the title
- Fields are generalized to zones that may contain free text as well
- Separate inverted indexes can be built for each zone

Indian.title → 11 → 34 → 56

Indian.body → 11 → 27 → 42 → 66

- Or, zone may be mentioned *explicitly* in a single inverted index

Indian → 11.title → 11.body → 27.body → 34.title → 42.body → 56.title → 66.body

Weighted Zone Scoring

- Each zone has a weight that adds up to 1
  - For example, "title" has 0.3, and "body" has 0.7
- Given a query term, each zone scores a zone score
- It is 1 if the term is inside the zone; 0 otherwise
- Weighted zone score is the linear sum of the zone scores

$$s(q, d) = \sum_{\forall z \in d} s_z(q)$$

- Given a Boolean combination of query terms, each zone is scored
- These scores are then accumulated and ranked
- Ranked Boolean retrieval
- Weights of zones
  - Can be supplied by the application
  - Machine learned

Term Frequency

- Moving away from the binary model
- If a document contains a query term more number of times, it is more important and should score higher
- Weight of a document d is, therefore, proportionate to the number of times the term t appears in it, called the term frequency

$$tf(t, d) = |t \in d|$$

- This assumes the bag of words model
- *Context* and *sequence* are lost
  - I love butter but I hate cheese
  - I love cheese but I hate butter

Inverse Document Frequency

- Certain terms may appear across all or most documents
  - "bat" in cricket pages
- Consequently, they discriminate little among the documents and are not useful
- Document frequency, df_t, of a term t is the number of documents in the corpus that it appears in
  - Lesser is more discriminative
- Weight of a term is inversely proportionate to the documents it appears in
- If m is the total number of documents in the corpus

$$idf(t) = \log \frac{m}{df_t}$$

- This is called the inverse document frequency
- Logarithmic to make it less drastic
- Theoretical justification from the *log-odds* model
- Does not affect the ranking

Tf-idf

- Combination of term frequency (tf) and inverse document frequency (idf)
- Weight of a term t in a document d is

$$tf\text{-}idf(t, d) = tf(t, d) \times idf(t)$$

- The tf-idf score has following properties
  - *Zero* if t does not appear at all in d
  - *High* when t appears many times in d
  - *Low* when t appears few times in d
  - *High* when t appears in a small number of documents
  - *Low* when t appears in a large number of documents
- Score of a query q for a document d is

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

Exercise

- d_1: Water, water everywhere, not a drop to drink
- d_2: I have filtered water
- d_3: Don't drink and drive, rather drive and drink
- d_4: Water quality is not good here
- d_5: Milk is not good for health
- d_6: Drinking water just after dinner is not healthy
- Query q: drinkable water
- Find tf, idf (with $\log_2$) and tf-idf ($\log_2$) scores, and rank

Terms	idf	tf						tf-idf					
		d_1	d_2	d_3	d_4	d_5	d_6	d_1	d_2	d_3	d_4	d_5	d_6
drink	$\log_2 \frac{6}{3} = 1$	1	0	2	0	0	1	1	0	2	0	0	1
water	$\log_2 \frac{6}{4} = \frac{1}{2}$	2	1	0	1	0	1	1	$\frac{1}{2}$	0	$\frac{1}{2}$	0	$\frac{1}{2}$
Query	-	-	-	-	-	-	-	2	$\frac{1}{2}$	2	$\frac{1}{2}$	0	$\frac{3}{2}$

- Therefore, ranking is
  - (d_1, d_3), (d_6), (d_2, d_4), (d_5)

Variants of Tf

- Tf is too drastic

$$tf(t,d) = \begin{cases} 1 + \log tf_{t,d} & \text{if present} \\ 0 & \text{otherwise} \end{cases}$$

- It still penalizes absent terms heavily

$$tf(t,d) = a + (1-a)\frac{tf_{t,d}}{\max\{tf_{t,d}\}}$$

- a is some default tf
- Very susceptible to outliers and stopwords
- Idf can be also modified as

$$idf(t) = \max\left\{0, \log\frac{m - df_t}{df_t}\right\}$$

Document Vector

- Each document d has a score with each term t in the vocabulary
- Imagine an n-dimensional vector space where n is the total number of terms in the vocabulary
- Each document can be, thus, thought of as a vector (point) in this n-dimensional space
- Its coordinates are the scores corresponding to

$$d[t_i] = \textit{tf-idf}(t_i, d)$$

- This is called the document vector model
- Document d is represented by its corresponding vector $\vec{V}(d)$
- Longer documents have more number of terms and larger tf's
- To balance, document vectors can be *normalized* by their length

$$\vec{v}(d) = \frac{\vec{V}(d)}{|\vec{V}(d)|}$$

Cosine Similarity

- Consider two documents d_1 and d_2 with their corresponding document vectors $\vec{V}(d_1)$ and $\vec{V}(d_2)$
- Cosine similarity measures the *normalized* dot product

$$\text{cosine-sim}(d_1, d_2) = \frac{\vec{V}(d_1).\vec{V}(d_2)}{|\vec{V}(d_1)|.|\vec{V}(d_2)|}$$

  - Measures the cosine of the angle between the vectors
- Consider the length-normalized document vectors
- Then, cosine similarity is their dot product

$$\text{co-sim}(d_1, d_2) = \vec{v}(d_1).\vec{v}(d_2)$$

Example

Term	d_1	d_2	d_3
Indian	115	58	20
ancient	10	7	11
system	2	0	6
length	115.45	58.42	23.60

- Similarities between documents

$$\text{sim}(d_1, d_2) = \frac{115}{115.45} \cdot \frac{58}{58.42} + \frac{10}{115.45} \cdot \frac{7}{58.42} + \frac{2}{115.45} \cdot \frac{0}{58.42} = 0.99$$
$$\text{sim}(d_1, d_3) = \frac{115}{115.45} \cdot \frac{20}{23.60} + \frac{10}{115.45} \cdot \frac{11}{23.60} + \frac{2}{115.45} \cdot \frac{6}{23.60} = 0.88$$
$$\text{sim}(d_2, d_3) = \frac{58}{58.42} \cdot \frac{20}{23.60} + \frac{7}{58.42} \cdot \frac{11}{23.60} + \frac{0}{58.42} \cdot \frac{6}{23.60} = 0.89$$

- d_1, d_2 is the closest pair
- d_3 is closer to d_2

Query Vector

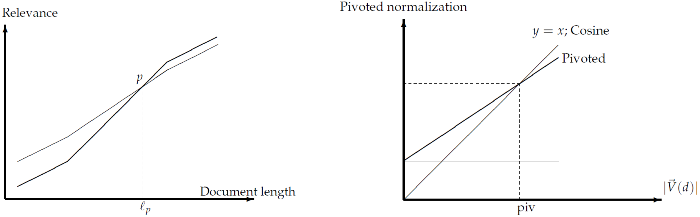
- Similar to a document, the query can also be viewed as a vector
- It is again just a *bag of words*
- Consider query q with the keywords "ancient" and "system"
- $\vec{V}(q) = (0, 1, 1)$ with $\vec{v}(q) = (0.00, 0.71, 0.71)$
- Most similar document is the one with the *highest* cosine similarity

$$\arg\max_{d_i}\{\text{co-sim}(q, d_i)\} = \arg\max_{d_i}\{\vec{v}(q).\vec{v}(d_i)\}$$

- Document vector retrieval task
- If $\vec{v}(q).\vec{v}(d_i)$ is the highest, then $\vec{V}(q).\vec{v}(d_i)$ is the highest as well
  - Thus, normalization of query is not required

Document Length Normalization

- Longer documents have more terms and, therefore, larger tf's
- Verbose documents may simply repeat terms to improve scores
- Documents on multiple topics need not be compensated
- Length normalization improves scores of shorter documents and deteriorates those of longer ones
- Ideally, with *known* queries and relevant documents



- Rotate about pivot to make cosine length similar

$$a.|\vec{V}(d)| + (1 - a).l_p \approx a.u_d + (1 - a).l_p$$

  - u_d is the number of unique terms in the document
- This is called pivot length normalization

Improving Efficiency

- Brute-force method of computing scores with all the documents and ranking them is *not* scalable
- Only terms that appear in query need to be examined
  - Rest of the scores are 0
- Filter query terms whose *idf* is too low
  - Similar to stopwords
- *Pre-compute* champion lists for each term
  - Documents ranked for only that term
  - Offline process
  - Take union of top-r of every query term to get top-K
- Build tiered index
  - Each level (tier) lists only those documents whose *tf* for the term is greater than a threshold
  - Continue with tiers till top-K results are obtained

Relevance Feedback

- Update query vector $\vec{q}$ that maximizes similarity with relevant documents and minimizes that with irrelevant ones
- D_r and D_n are sets of *actual* relevant and irrelevant documents

$$\vec{q}_{opt} = \arg\max_{\forall \vec{q}} [sim(\vec{q}, D_r) - sim(\vec{q}, D_n)]$$

- Using cosine similarity

$$\vec{q}_{opt} = \frac{1}{|D_r|} \sum_{\forall \vec{d}_r \in D_r} \vec{d}_r - \frac{1}{|D_n|} \sum_{\forall \vec{d}_n \in D_n} \vec{d}_n$$

- However, D_r and D_n are not known fully
- Only a partial subset U_r and U_n from user is known
- Rocchio algorithm

$$\vec{q}_m = \alpha.\vec{q}_0 + \beta.\frac{1}{|U_r|} \sum_{\forall \vec{d}_r \in U_r} \vec{d}_r - \gamma.\frac{1}{|U_n|} \sum_{\forall \vec{d}_n \in U_n} \vec{d}_n$$

  - Positive feedback is more important: $\alpha = 1.00, \beta = 0.75, \gamma = 0.15$