Monsoon 2021

Vector Space Model

- Vector Space Model and Score computation

Dr. Rajendra Prasath

Indian Institute of Information Technology Sri City, Chittoor

3rd September 2021 (rajendra.2power3.com)

> Topics to be covered

Overview

- Recap:
  - Phrase Queries / Proximity Search
  - Spell Correction / Noisy Channel Modelling
  - Index Construction
    - BSBI
    - SPIMI
    - Distributed Indexing
      - An Illustration

- Vector Space Models
- Term Weighting Approaches
  - 5 Different Approaches
    - An Illustration

- More topics to come up … Stay tuned …!!

Recap: Information Retrieval

- **Information Retrieval (IR)** is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- These days we frequently think first of web search, but there are many other cases:
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Legal information retrieval
  - and so on . . .

Blocked Sort-Based Indexing

```
BSBIndexConstruction()
n ← 0
while (all documents have not been processed)
do n ← n + 1
    block ← ParseNextBlock()
    BSBI-Invert(block)
    WriteBlockToDisk(block, f_n)
MergeBlocks(f_1, ..., f_n; f_merged)
```

- Key decision: What is the size of one block?

Single-pass in-memory indexing

- Abbreviation: SPIMI
- Key idea 1: Generate separate dictionaries for each block – no need to maintain term-termID mapping across blocks.
- Key idea 2: Don't sort. Accumulate postings in postings lists as they occur.
- With these two ideas we can generate a complete inverted index for each block.
- These separate indexes can then be merged into one big index.

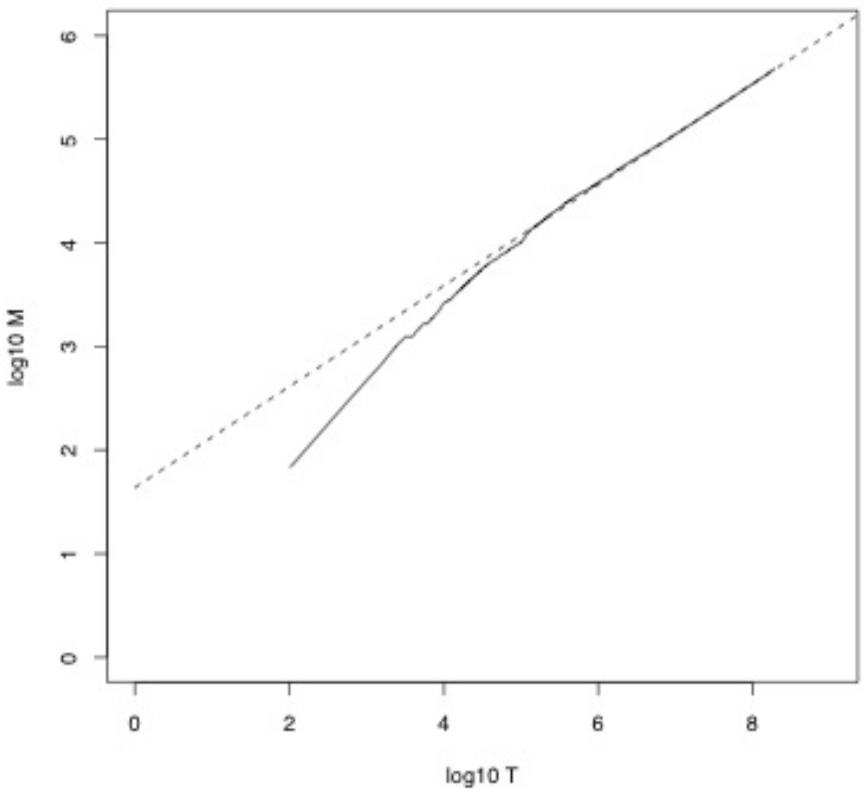
Distributed Indexing

- For web-scale indexing (don't try this at home!): must use a distributed computer cluster
- Individual machines are fault-prone.
  - Can unpredictably slow down or fail.
- How do we exploit such a pool of machines?

Overview

- Why ranked retrieval?
- Term frequency
- tf-idf weighting
- The vector space model

Heaps' law

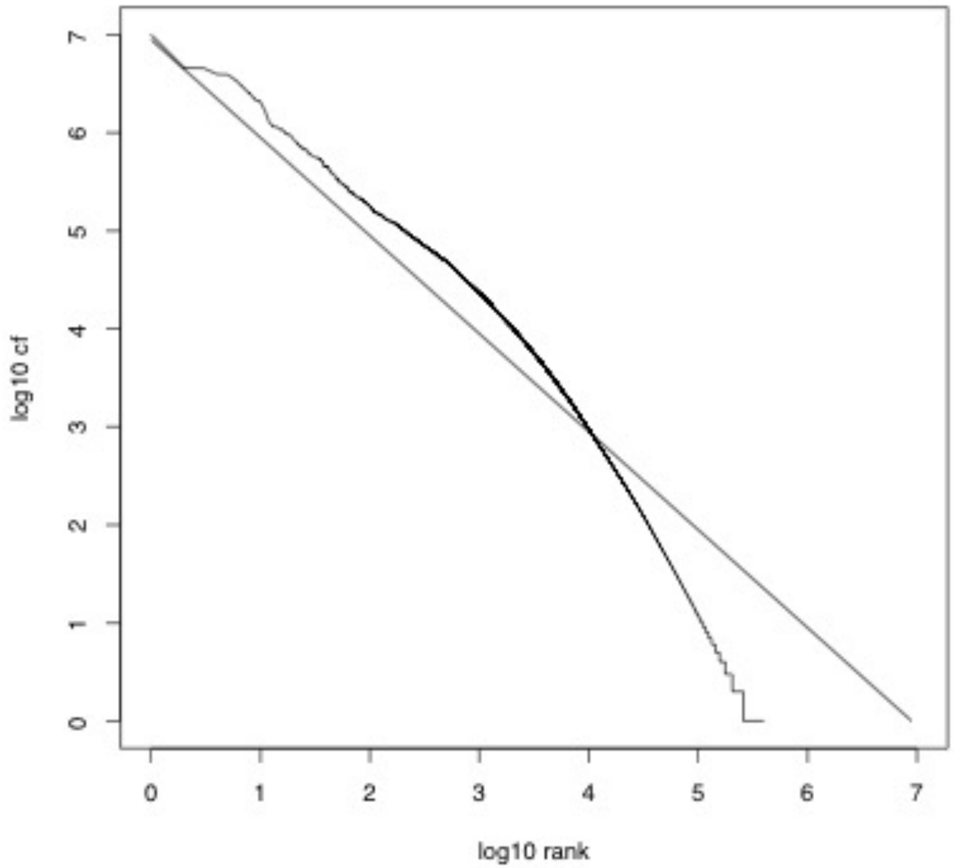


Vocabulary size M as a function of collection size *T* (number of tokens) for Reuters-RCV1.

For these data, the dashed line $\log_{10} M = 0.49 * \log_{10} T + 1.64$ is the best least squares fit.

Thus, $M = 10^{1.64} T^{0.49}$ and $k = 10^{1.64} \approx 44$ and $b = 0.49$.

Zipf's law



$$\mathrm{cf}_i \propto \frac{1}{i}$$

The most frequent term (*the*) occurs cf_1 times, the second most frequent term $\mathrm{cf}_2 = \frac{1}{2}\mathrm{cf}_1$ (*of*) occurs times, the third most

$$\mathrm{cf}_3 = \frac{1}{3}\mathrm{cf}_1$$

frequent term (*and*) occurs times etc.

Ranked Retrieval

- Our Queries have all been Boolean
  - Documents either match or don't

- Good for expert users with precise understanding of their needs and of the collection.
- Also good for applications: Applications can easily consume 1000s of results.

- Not good for the majority of users

- Most users don't want to wade through 1000s of results.
- This is particularly true of web search.

Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650]
  - → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found]
  - → 0 hits – famine
- In Boolean retrieval, it takes a lot of skill to come up with a query that produces a manageable number of hits.

Feast or famine: No problem in ranked retrieval

- With ranking, large result sets are not an issue
- Just show the top 10 results
- Does not overwhelm the user
- Premise: the ranking algorithm works: More relevant results are ranked higher than less relevant results.

Scoring as the basis of ranked retrieval

- We wish to rank documents that are more relevant higher than documents that are less relevant.
- How can we accomplish such a ranking of the documents in the collection with respect to a query?
- Assign a score to each query-document pair, say in [0, 1]
- This score measures how well document and query "match"

Query-document matching scores

- How do we compute the score of a query-document pair?
- Let’s start with a one-term query.
- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score
- We will look at a number of alternatives for doing this.

Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$(A \neq \emptyset \text{ or } B \neq \emptyset)$$

- JACCARD (A, A) = 1
- JACCARD (A, B) = 0 if A ∩ B = 0
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:

- Query: “ides of March”
- Document “Caesar died in March”
- JACCARD(q, d) = 1/6

What's wrong with Jaccard?

- It does not consider term frequency (how many occurrences a term has)
- Rare terms are more informative than frequent terms
  - Jaccard does not consider this information
- We need a more sophisticated way of normalizing the length of a document

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

- Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

- Each document is now represented as a count vector ∈ N|V|.

Bag of words model

- We do not consider the order of words in a document.
- John is quicker than Mary and Mary is quicker than John are represented in the same way.
- This is called a bag of words model.
- In a sense, this is a step back: The positional index was able to distinguish these two documents.
- We will look at "recovering" positional information later in this course.
- For now: bag of words model

Term frequency (tf)

- The term frequency tft,d of term t in document d is defined as the number of times that t occurs in d
- Use tf to compute query-doc. match scores
- Raw term frequency is not what we want
- A document with tf = 10 occurrences of the term is more relevant than a document with tf = 1 occurrence of the term
- But not 10 times more relevant
- Relevance does not increase proportionally with term frequency

Log frequency weighting

- The log frequency weight of term t in d is defined as follows

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$: $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.

- Score for a document-query pair: sum over terms t in both q and d:

- tf-matching-score(q, d) = $\sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$

- The score is 0 if none of the query terms is present in the document

Exercise

- Compute Jaccard matching score & TF matching score for the following query-document pairs

- q: [information on cars]
- d: "all you've ever wanted to know about cars"

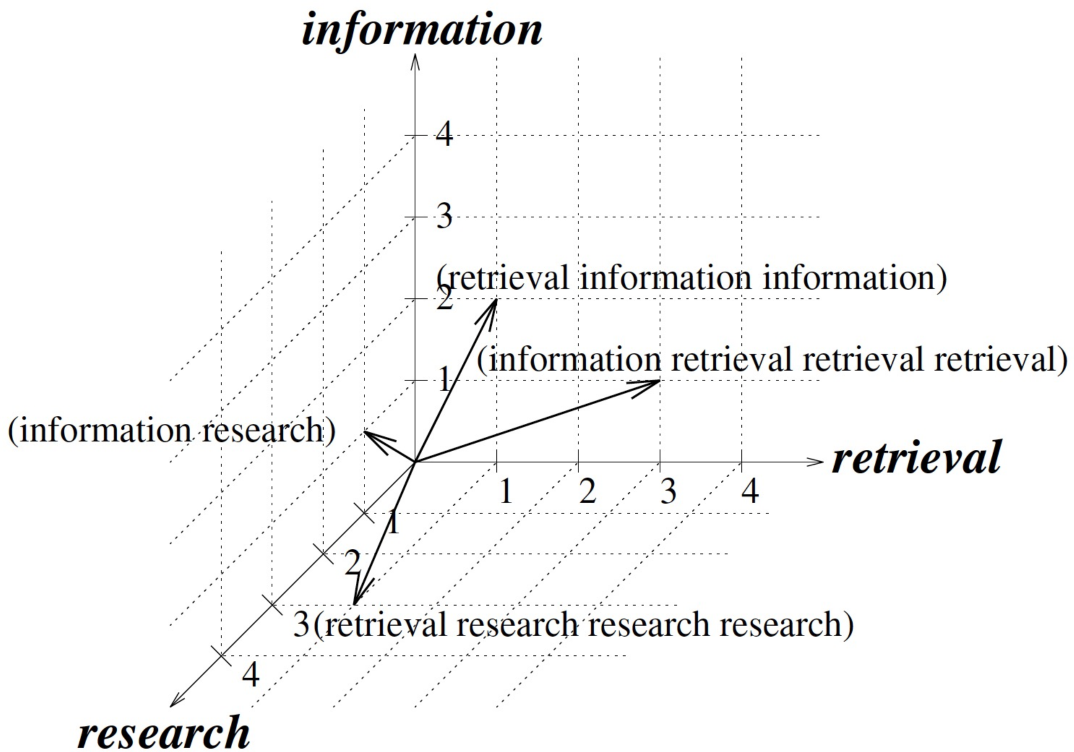
- q: [information on cars]
- d: "information on trucks, information on planes, information on trains"

- q: [red cars and red trucks]
- d: "cops stop red cars more often"

Vector Space Model

- Consider three word model

  "information retrieval research"



Measure of Closeness of Vectors

- How to measure the closeness between two vectors (texts)?

- Two texts are semantically related if they share some vocabulary
  - More Vocabulary they share, the stronger is the relationship
  - This implies that the measure of closeness increases with the number of words matches between two texts

- If matching terms are important then the vectors should be considered closer to each other

Modern Vector Space Models

- The length of the sub-vector in dimension-i is used to represent the importance or the weigh of word-i in a text
- Words that are absent in a text get a weight – 0 (zero)
- Apply vector inner product measure between two vectors:
- This vector inner product increases:
  - # words match between two texts
  - Importance of the matching terms

Finding closeness between texts

- Given two texts in T dimensional vector space:

  $\vec{P} = (p_1, p_2, \ldots, p_T)$ and $\vec{Q} = (q_1, q_2, \ldots, q_T)$

- The inner product between these two vectors:

$$\vec{P} \cdot \vec{Q} = \sum_{i=1}^{T} \sum_{j=1}^{T} p_i \times \vec{u_i} \cdot q_j \times \vec{u_j}$$

- Vectors u_i and u_j are unit vectors in dimensions *i and j* (Here $u_i \cdot u_j = 0$, if $i \neq j$ - orthogonal)
- Vector Similarity: Closeness between two texts

$$similarity(\vec{P}, \vec{Q}) = \sum_{i=1}^{T} p_i \times q_i$$

Exercise – Try Yourself

- Consider a collection of n documents
- Let n be sufficiently large (at least 100 docs)
  - You can take our dataset
  - Sports News Dataset (ths-181-dataset.zip)

- **Find two lists:**
  - The most frequency words and
  - The least frequent words
  - Form k (=10) queries each with exactly 3-words taken from above lists (at least one from each)
  - Compute Cosine Similarity between each query and and documents

Summary

In this class, we focused on:

(a) Recap: Positional Indexes

i. Wild card Queries
ii. Spelling Correction
iii. Noisy Channel modelling for Spell Correction

(b) Various Indexing Approaches

i. Block Sort based Indexing Approach
ii. Single Pass In Memory Indexing Approach
iii. Distributed Indexing using Map Reduce
iv. Examples

Acknowledgements

Thanks to ALL RESEARCHERS:

1. Introduction to Information Retrieval Manning, Raghavan and Schutze, Cambridge University Press, 2008.
2. Search Engines Information Retrieval in Practice W. Bruce Croft, D. Metzler, T. Strohman, Pearson, 2009.
3. Information Retrieval Implementing and Evaluating Search Engines Stefan Büttcher, Charles L. A. Clarke and Gordon V. Cormack, MIT Press, 2010.
4. Modern Information Retrieval Baeza-Yates and Ribeiro-Neto, Addison Wesley, 1999.
5. Many Authors who contributed to SIGIR / WWW / KDD / ECIR / CIKM / WSDM and other top tier conferences
6. Prof. Mandar Mitra, Indian Statistical Institute, Kolkatata (https://www.isical.ac.in/~mandar/)

Questions
It’s Your Time

Contact Information:

Dr. Rajendra Prasath
IIIT Sri City, Chittoor