Monsoon 2019

Information Retrieval

by

Dr. Rajendra Prasath

Indian Institute of Information Technology

Sri City – 517 646, Andhra Pradesh, India

Lecture – 08

✧ Topics Covered So Far

- ✧ Bi-Word Index
- ✧ Wild Card Queries
- ✧ Permuterm Index
- ✧ K-gram Index (k = 2 → Bigram Index)
- ✧ Spell Correction

✧ Now: Term Weighting

- ✧ Approaches to terms weighting in IR

Recap: Spelling Tasks

- ✧ Spelling Error Detection
- ✧ Spelling Error Correction:
  - ✧ Autocorrect
    - ✧ hte→the
  - ✧ Suggest a correction
  - ✧ Suggestion lists

Recap: Real word & non-word spelling errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spellings***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel view of spell errors
  - Context-sensitive – so have to consider whether the surrounding words "make sense"
  - *Flying* **form** *Heathrow to LAX* → *Flying* **from** *Heathrow to LAX*

Recap: Language Model

- Take a big supply of words (your document collection with T tokens); let C(w) = # occurrences of w

$$P(w) = \frac{C(w)}{T}$$

- In other applications – you can take the supply to be typed queries (suitably filtered) – when a static dictionary is inadequate

Recap: Channel model

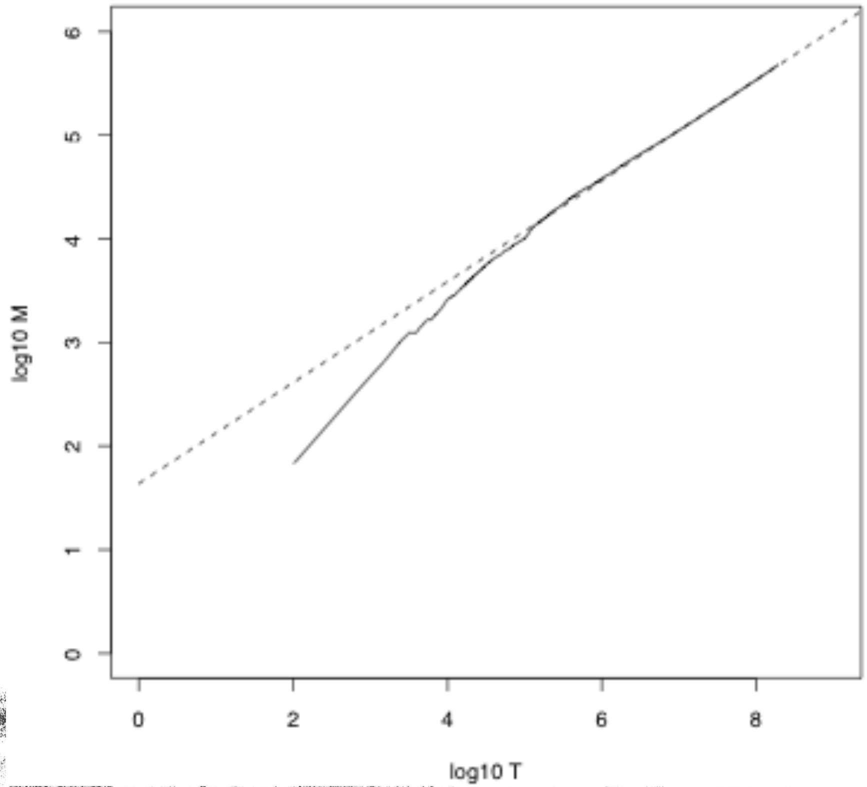
$$P(x|w) = \begin{cases} \dfrac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \dfrac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Kernighan, Church, Gale 1990

Overview

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

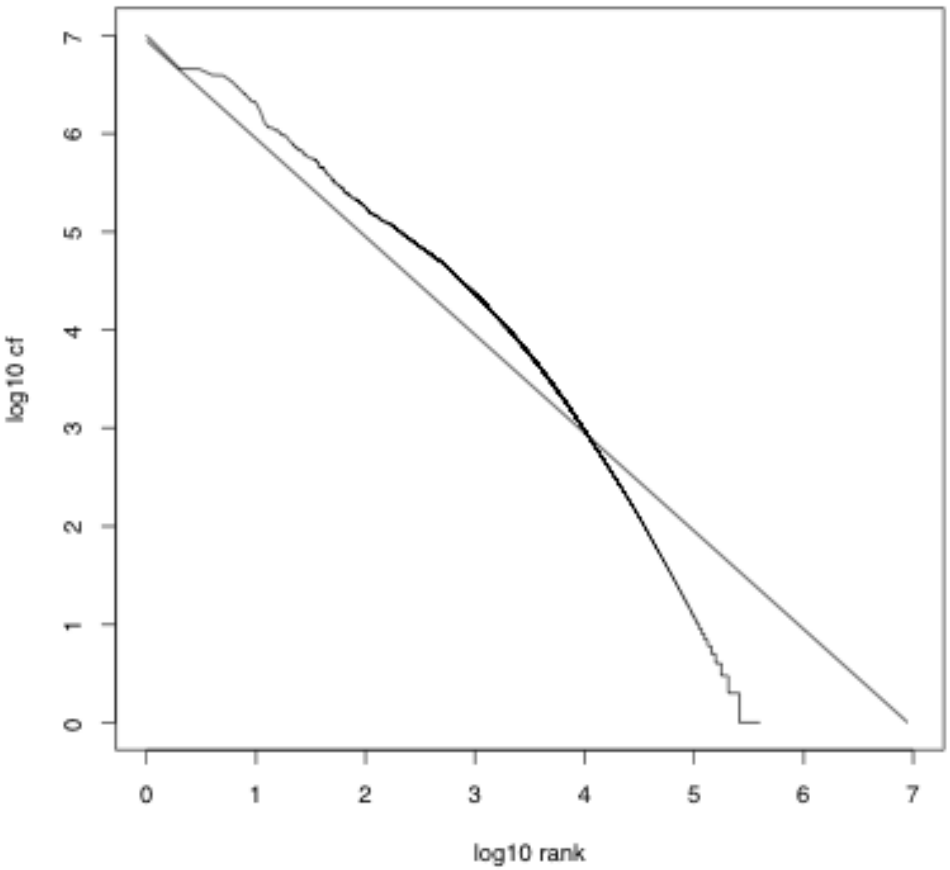
Heaps' law



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 $b = 0.49$.

8

Zipf's law



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

The most frequent term (*the*) occurs cf_1 times, the second most frequent term (*of*) occurs $\mathrm{cf}_2 = \frac{1}{2}\mathrm{cf}_1$ times, the third most frequent term (*and*) occurs $\mathrm{cf}_3 = \frac{1}{3}\mathrm{cf}_1$ times etc.

9

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.

10

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.

11

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.

12

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"

13

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.

14

Jaccard coefficient

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

$$\textsc{Jaccard}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$(A \neq \emptyset$ or $B \neq \emptyset)$

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

15

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

16

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

17

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 $\in \{0, 1\}^{|V|}$.

18

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 $\in \mathbb{N}^{|V|}$.

19

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 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

20

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

21

Log frequency weighting

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

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

- $tf_{t,d} \rightarrow w_{t,d}$:
  $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
- Score for a document-query pair: sum over terms t in both q and d:
  $$\text{tf-matching-score}(q, d) = \sum_{t \in q \cap d} (1 + \log tf_{t,d})$$

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

22

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]

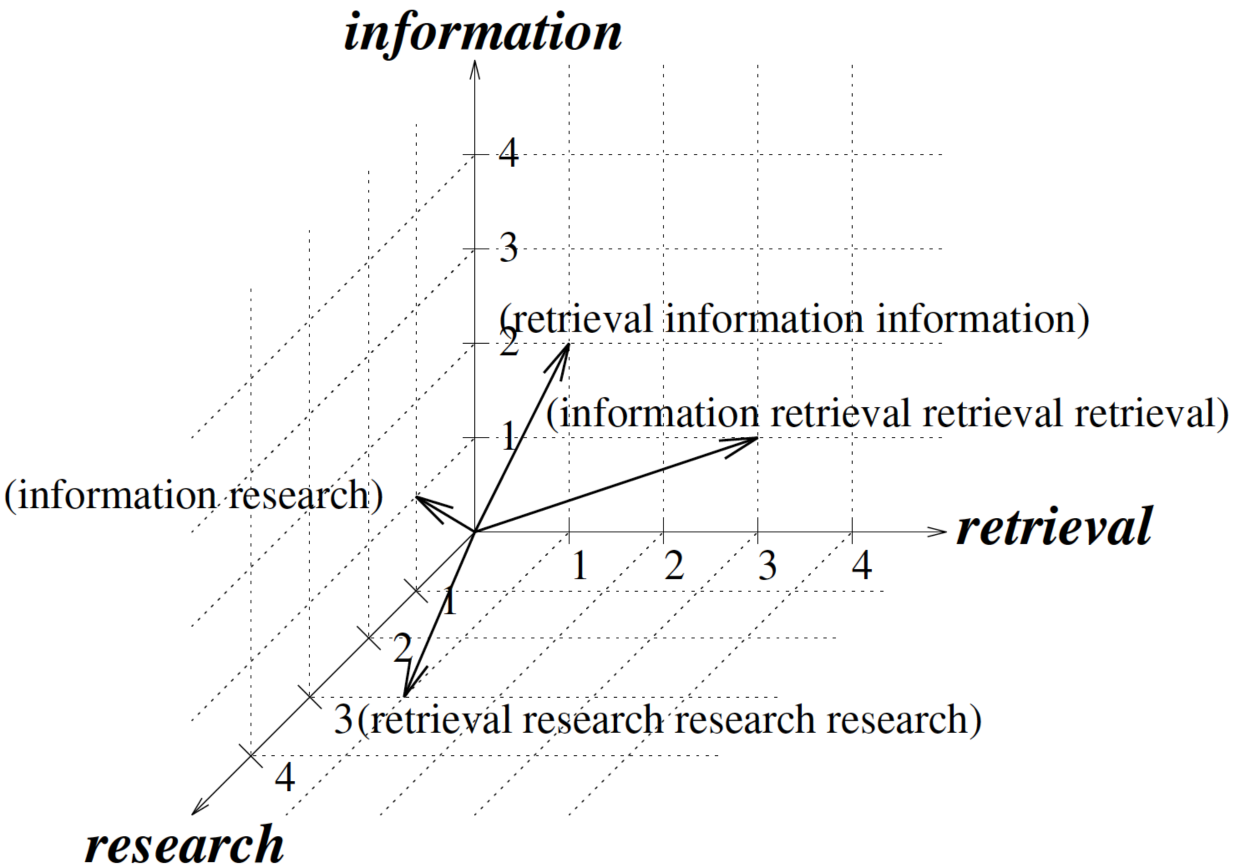
  d: “information on trucks, information on planes, information on trains”
- q: [red cars and red trucks]

  d: “cops stop red cars more often”

23

Vector Space Model

- Consider three word model
  "information retrieval research"



24

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

25

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

26

Finding closeness between texts

- Given two texts in T dimensional vector space:

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

- 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 $u_i \bullet u_j = 0$, if $i \neq j$ - orthogonal)
- Vector Similarity: Closeness between two texts

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

27

Term Weighting

- The Importance of a term increases with the number of occurrences of a term in a text.

- So we can estimate the term weight using some monotonically increasing function of the number of occurrences of a term in a text

- **Term Frequency:**

  The number of occurrences of a term in a text is called **Term Frequency**

28

Term Frequency Factor

- What is Term Frequency Factor?
  - The function of the term frequency used to compute a term's importance
  - Some commonly used factors are:
    - Raw TF factor
    - Logarithmic TF factor
    - Binary TF factor
    - Augmented TF factor
    - Okapi's TF factor

29

The Raw TF factor

- This is the simplest factor
- This counts simply the number of occurrences of a term in a text
- Simply count the number of terms in each document
- More the number, higher the ranking of the document!!

30

The Logarithmic TF factor

- This factor is computed as

  $$1 + \ln(tf)$$

  where tf is the term frequency of a term
- Proposed by Buckley (1993 - 94)
- Motivation:
  - If a document has one query term with a very high term frequency then the document is (often) not better than another document that has two query terms with somewhat lower term frequencies
  - More occurrences of a match should not out-contribute fewer occurrences of several matches

31

Example – log TF factor

- Consider the query: “recycling of tires”
- Two documents:

D1: with 10 occurrences of the word “recycling”

D2: with “recycling” and “tires” 3 times each

- Everything else being equal, if we use raw tf, D1(10) gets higher similarity score than D2 (3+3=6)
- But D2 addresses the needs of the query
- Log TF: reflects usefulness of D2 in similarities
- D1: $1 + \ln(10) = 3.3$ and D2: $2(1+\ln(3)) = 4.1$

32

The Binary TF factor

- The TF factor is completely disregards the number of occurrences of a term.
- It is either one or zero depending upon the presence (one) or the absence (zero) of the term in a text.
- This factor gives a nice baseline to measure the usefulness of the term frequency factors in document ranking

33

The Augmented TF factor

- This TF factor reduces the range of the contributions of a term from the freq. of a term
- How: Compress the range of the possible TF factor values (say between 0.5 & 1.0)
- The augmented TF factor is used with a belief that mere presence of a term in a text should have some default weight (say 0.5)
- Then additional occurrences of a term could increase the weight of the term to some max. value (usually 1.0).

34

Augmented TF factor - Scoring

- This TF factor is computed as follows:

$$0.5 + 0.5 \times \frac{tf}{\text{maximum tf in text}}$$

- The augmented TF factor emphasizes that more matches are more importance than fewer matches (like log TF factor)
- A single match contributes at least 0.5 and high TFs can only contribute at most another 0.5
- This was motivated by document length considerations and does not work as well as log TF factor in practice.

Okapi's TF factor

- Robertson et. Al (1994) developed Okapi Information Retrieval System and proposed another TF factor
- This TF factor is based on Approximations to the 2-Poossion Model:
- This factor $$\frac{tf}{2 + tf}$$

  is quite close to the log TF factor in its behavior
- In practice, log TF factor is effective for good document ranking

36

Exercise – E08

- Consider a collection of n documents
- Let n be sufficiently large (at least 100 docs)
- 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 Similarity between each query and and documents

37

Summary

In this class, we focused on:

(a) Words / Terms / Lexical Units

(b) Term Weighting approaches

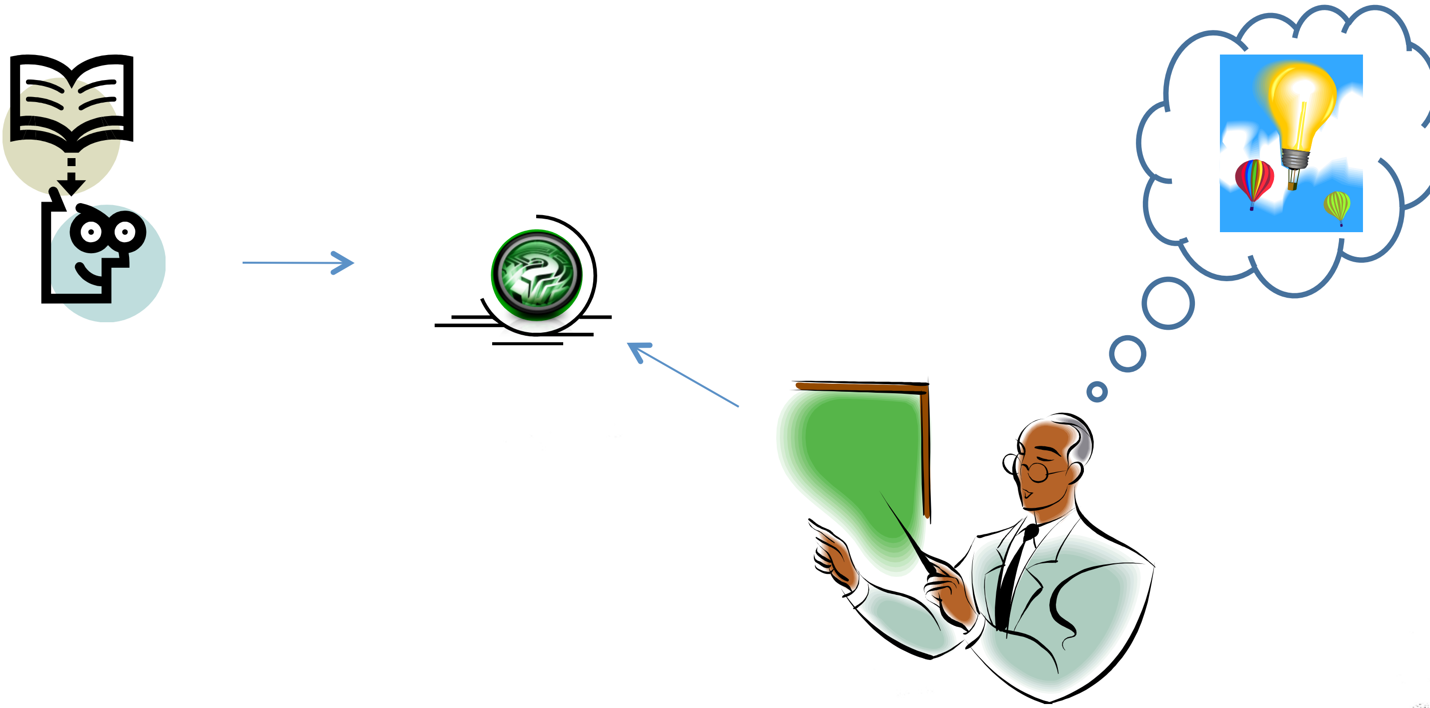
(c) Evaluating the best term weighting approach

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/)

Thanks …



… Questions ???