# Information Retrieval
# Topic- Scoring, Term Weighting, The Vector Space Model
# (Term frequency and weighting)
# Lecture-23

**Prepared By**

Dr. Rasmita Rautray & Dr. Rasmita Dash
Associate Professor
Dept. of CSE

# Content

- Ranked retrieval
- Jaccard coefficient
- Term frequency

# Ranked retrieval

- Ranking search results: why it is important (as opposed to just presenting a set of unordered Boolean results)
- Thus far, our queries have 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 are not capable of writing Boolean queries . . .
  - . . . or they are, but they think it's too much work.
- 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):

→ 200,000 hits – feast

- Query 2 (boolean conjunction): [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
- The ranking algorithm works: More relevant results are ranked higher than less relevant results.

# Scoring as the basis of ranked retrieval

- How can we accomplish a relevance ranking of the documents 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”.
- Sort documents according to scores

# Query-document matching scores

- How do we compute the score of a query-document pair?
- If no query term occurs in the document: score should be 0.
- The more frequent a query term in the document, the higher the score
- The more query terms occur 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: jaccard(A,B) = $|A \cap B| / |A \cup B|$

($A \neq \emptyset$ 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.

# 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 doesn't 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 for the length of a document.

# Term frequency

- This is a key ingredient for ranking

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

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

# Bag of words model

- The exact ordering of the terms in a document is ignored but the number of occurrences of each term is material.
- Only retain information on the number of occurrences of each term
  - 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.

# Term frequency (tf)

- The term frequency $tf_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.
- We want to use *tf* when computing query-document match scores.
  - But how?
- Raw term frequency is not what we want because:
- 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.

# Instead of raw frequency: Log frequency weighting

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

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

- $tf_{t,d} \rightarrow w_{t,d}$ :
- $0 \rightarrow 0$, $1 \rightarrow 1$, $2 \rightarrow 1.3$, $10 \rightarrow 2$, $1000 \rightarrow 4$, etc.
- Score for a document-query pair: sum over terms t in both q and d:
- 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.

# Example

Query: “best car insurance”

Document: “car insurance auto insurance”

| words | Query | | Document | |
|---|---|---|---|---|
| | tf-raw | tf-wt | tf-raw | tf-wt |
| auto | 0 | 0 | 1 | 1 |
| best | 1 | 1 | 0 | 0 |
| car | 1 | 1 | 1 | 1 |
| insurance | 1 | 1 | 2 | 1.3 |

Thank You