# Lect 2: Boolean Retrieval

Dr. Subrat Kumar Nayak

Associate Professor

Department of CSE, ITER, SOADu

# Boolean Retrieval Model

- The ***Boolean retrieval model*** is a model for information retrieval in which we can pose any query which is in the form of a Boolean expression of terms, that is, in which terms are combined with the operators **AND, OR, and NOT**.
- The model views each document as just a set of words.
- Queries are Boolean expressions, e.g., CAESAR AND BRUTUS
- The search engine **returns all documents** that satisfy the Boolean expression.
- A sort of linear scan through documents.

# Boolean Retrieval: Grepping

- The simplest form of document retrieval is for a computer to do linear scan through documents. This process is commonly referred to as *grepping* through text.
- Names after the Unix command grep, which performs this process.
- Grepping through text can be a very effective process, especially given the speed of modern computers
- Often this allows useful possibilities for wild card pattern matching through the use of regular expressions.

Example:

- Suppose you wanted to determine which document contain the words **Information AND Retrieval AND NOT Boolean**
- One way to do that is to start at the beginning and to read through all the text, noting for each document whether it contains Information and Retrieval and excluding it from consideration if it contains Boolean.
- This process is commonly referred to as ***grepping*** through text

# Boolean Retrieval: Grepping

- But for many purposes, you do need more:
- To process large document collections quickly. The amount of online data has grown at least as quickly as the speed of computers, and we would now like to be able to search collections that total in the order of billions to trillions of words.
- To allow more flexible matching operations. For example, it is impractical to perform the query Romans NEAR countrymen with `grep`, where NEAR might be defined as "within 5 words" or "within the same sentence".
- To allow ranked retrieval: in many cases you want the best answer to an information need among many documents that contain certain words.

# Unstructured data in 1620

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out lines containing ***Calpurnia***?
- Why is that not the answer?
  - Slow (for large corpora)
  - *NOT* ***Calpurnia*** is non-trivial
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Ranked retrieval (best documents to return)
    - Later lectures

# Boolean Retrieval: Some terminology

- **Documents:** *documents* means whatever units we have decided to build a retrieval system over. They might be individual memos or chapters of a book.
- **Collection/ Corpus**: We will refer to the group of documents over which we perform retrieval as the (document) ***collection***. It is sometimes also referred to as a ***corpus*** (a *body* of texts).

**Let us consider Shakespeare's Collected Works, and use it to introduce the basics of the Boolean retrieval model.**

# Boolean Retrieval: Term-document incidence matrices

- The way to avoid linearly scanning the texts for each query is to ***index*** the documents in advance.
- The binary **term-document *incidence matrix***, is an outcome of recording each **document** – here a **play** of Shakespeare's – whether it contains each word out of all the words Shakespeare used (Shakespeare used about 32,000 different words)

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 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 |

1 if play contains word, 0 otherwise

***Brutus** AND **Caesar** BUT NOT **Calpurnia***

# Boolean Retrieval: Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) ➔ bitwise *AND*.
  - 110100 *AND*
  - 110111 *AND*
  - 101111 =
  - **100100**

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 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 |

# Answers to query

- Antony and Cleopatra, Act III, Scene ii

*Agrippa* [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus,

When Antony found Julius ***Caesar*** dead,

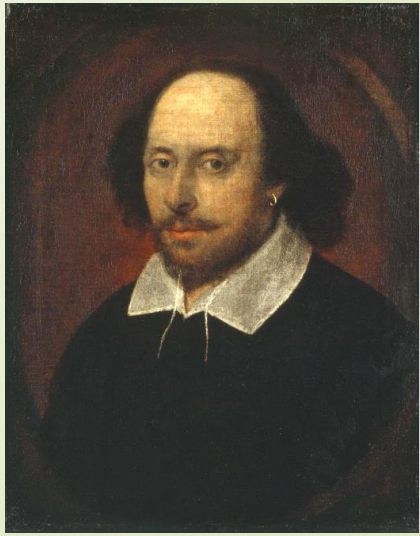
He cried almost to roaring; and he wept

When at Philippi he found ***Brutus*** slain.

- Hamlet, Act III, Scene ii

*Lord Polonius:* I did enact Julius ***Caesar*** I was killed i' the

Capitol; ***Brutus*** killed me.



# Bigger collections

- Consider $N$ = 1 million documents, each with about 1000 words.
- Avg 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents.
- Say there are $M$ = 500K *distinct* terms among these.

# Can’t build the matrix

- 500K x 1M matrix has half-a-trillion 0’s and 1’s.
- But it has no more than one billion 1’s.
  - matrix is extremely sparse.
- What’s a better representation?
  - We only record the 1 positions.