# CS317

Information Retrieval

Week 01

Muhammad Rafi

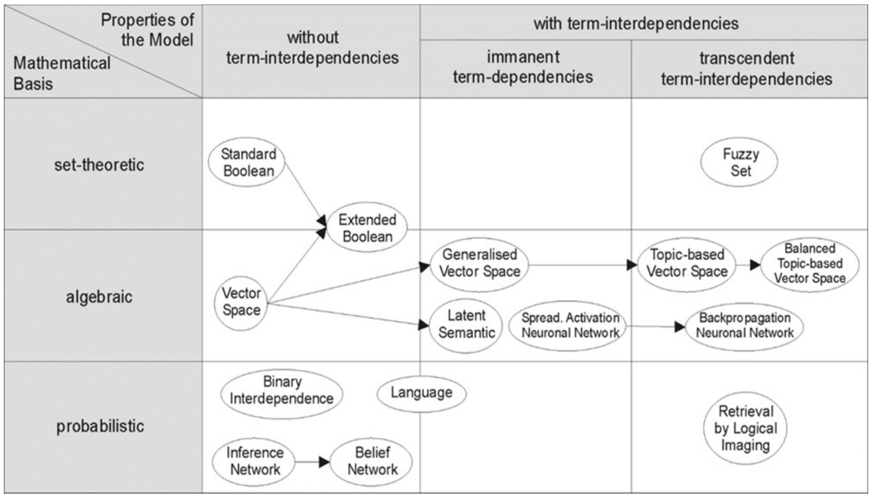
January 23, 2020

# Chapter No. 1
# Boolean Retrieval

# What is a Model?

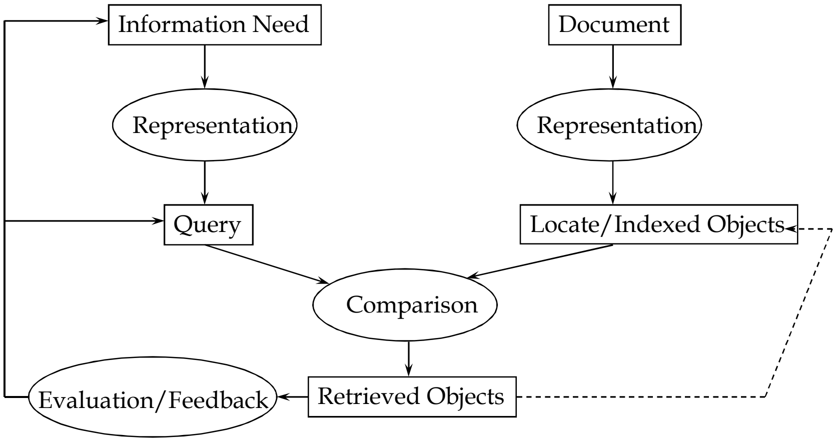
- A model is an abstract representation of a process or object
  - Used to study properties, draw conclusions, make predictions
  - The quality of the conclusions depends upon how closely the model represents the reality.
- A retrieval model describes the human and computational processes involved in ad-hoc retrieval
  - **Example:** A model of human information seeking behavior
  - **Example:** A model of how documents are ranked computationally
  - **Components:** Users, information needs, queries, documents, relevance assessments, ….
  - Retrieval models define relevance, explicitly or implicitly

# IR Models



| Properties of the Model<br>Mathematical Basis | without term-interdependencies | with term-interdependencies<br>immanent term-dependencies | with term-interdependencies<br>transcendent term-interdependencies |
|---|---|---|---|
| set-theoretic | Standard Boolean<br>Extended Boolean | | Fuzzy Set |
| algebraic | Vector Space | Generalised Vector Space<br>Latent Semantic<br>Spread. Activation Neuronal Network | Topic-based Vector Space<br>Balanced Topic-based Vector Space<br>Backpropagation Neuronal Network |
| probabilistic | Binary Interdependence<br>Language<br>Inference Network<br>Belief Network | | Retrieval by Logical Imaging |

# Basic Information Retrieval Process



# Basic Information Retrieval Setting

- Offline / Online
  - Corpus
  - Dynamic /changing collection (www)
- Scale of operation
  - World wide web level
  - Personal / individual level
  - Enterprise / domain specific level
- Type of Content
  - Text/multimedia/ mixed content

Sec. 1.1

# Unstructured data in 1680

- Shakespeare’s work
  - Lets you have all his plays in soft copy with you, in simple text format.
  - 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***?

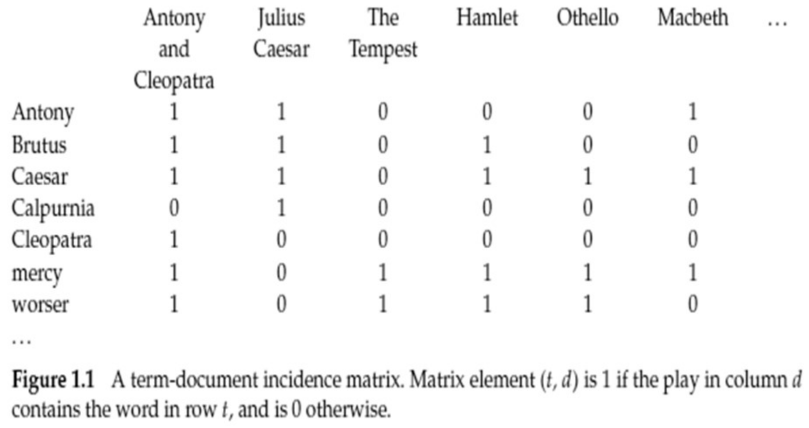
# Example

- Human Method
  - Read the book from start to end, if any play contains Brutus AND Caesar and NOT Calpurnia then the title is returned as result.
- Machine Method
  - Read the each file sequentially for Brutus AND Caesar and NOT Calpurnia, maintain a boolean status for each play. Check for play in which Brutus = 1 AND Caesar=1 and Calpurnia=0
  - Problems:
    - Lets discuss ….

# Boolean Retrieval Model

- The Boolean retrieval model views each document as just a set of words (Multi-set)
- 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.
- One of the classical model of IR.

# Term-Document Matrix



| | 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 | |
| … | | | | | | | |

**Figure 1.1** A term-document incidence matrix. Matrix element (*t*, *d*) is 1 if the play in column *d* contains the word in row *t*, and is 0 otherwise.

## Answer to the Query

To answer the query Brutus AND Caesar AND NOT Calpurnia, we take the vectors for Brutus, Caesar and Calpurnia, complement the last, and then do a bitwise AND:

110100 AND 110111 AND 101111 = 100100

The answers for this query are thus Antony and Cleopatra and Hamlet

## Term-Document Matrix

- Problems
  - Sparse matrix
  - Exact matching
  - Complex query formulation
- Solution: Inverted Index, we will discussed soon

# Adhoc Retrieval Systems

- It is a system that aims to provide documents from within the collection that are relevant to an arbitrary user information need, communicated to the system by means of a one-off, user-initiated query.
- Example:
  - WestLaw
  - Assignment No. 1

# Effectiveness of IR System

- Effectiveness of an IR system is the quality of its search results.
- There are two factors on which a user believe on the IR system.
  - Precision
    - What fraction of the returned results are relevant to the information need?
  - Recall
    - What fraction of the relevant documents in the collection were returned by the system?

# Example

| | Relevant | Irrelevant |
|---|---|---|
| Retrieved | Rel+Ret | Irrel+Ret |
| Non Retrieved | Non-Ret + Rel | Non-Ret + Irrel |

Precision = relevant –retrieved / total retrieved

Recall = relevant –retrieved / total Relevant in collection

# Example:

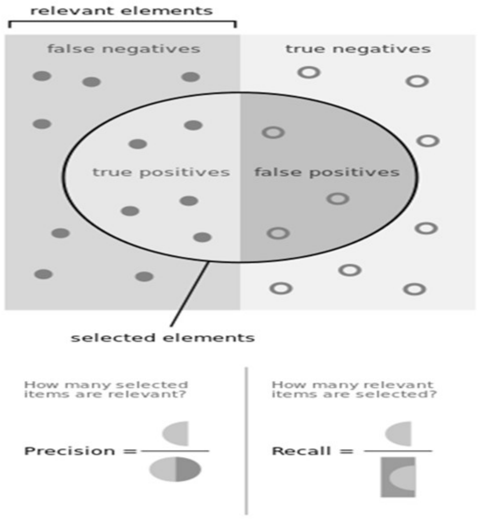


Image source : Wikipedia

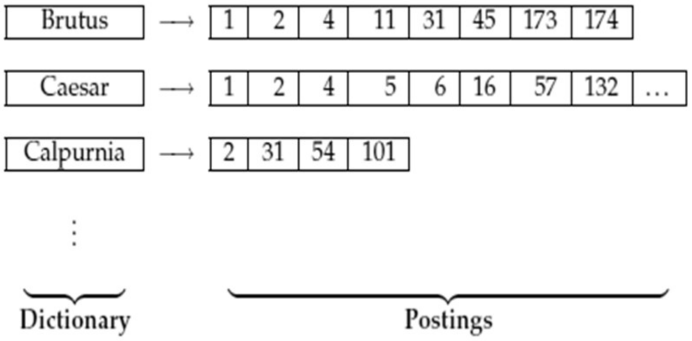
## Example of IR Evaluation

- An IR system returns 8 relevant documents, and 10 irrelevant documents. there are total of 20 relevant documents in the collection. what is the precision of the system, on this search, and what is its recall?
  - precision = 8/18 = 0.44
  - recall 8/20 = 0.4

## Inverted Index

- An inverted index is actual an index of the vocabulary that occurs in a collection.
- There are two part of an inverted index
  - Lexicon/Dictionary/ Vocabulary (terms, words, lexeme, tokens)
  - Posting/Posting-list (document information, where the corresponding term belong)

# Inverted Index



▶ **Figure 1.3** The two parts of an inverted index. The dictionary is commonly kept in memory, with pointers to each postings list, which is stored on disk.

# Example

**Doc 1**
I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

**Doc 2**
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

| term | docID |
|---|---|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 2 |
| brutus | 2 |
| hath | 2 |
| told | 2 |
| you | 2 |
| caesar | 2 |
| was | 2 |
| ambitious | 2 |

⟹

| term | docID |
|---|---|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |

=

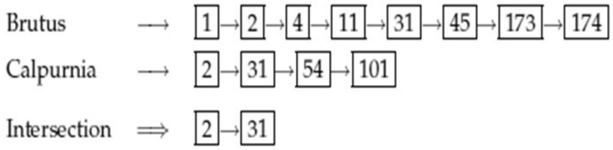
| term | doc. freq. | → | postings lists |
|---|---|---|---|
| ambitious | 1 | → | 2 |
| be | 1 | → | 2 |
| brutus | 2 | → | 1 → 2 |
| capitol | 1 | → | 1 |
| caesar | 2 | → | 1 → 2 |
| did | 1 | → | 1 |
| enact | 1 | → | 1 |
| hath | 1 | → | 2 |
| I | 1 | → | 1 |
| i' | 1 | → | 1 |
| it | 1 | → | 2 |
| julius | 1 | → | 1 |
| killed | 1 | → | 1 |
| let | 1 | → | 2 |
| me | 1 | → | 1 |
| noble | 1 | → | 2 |
| so | 1 | → | 2 |
| the | 2 | → | 1 → 2 |
| told | 1 | → | 2 |
| you | 1 | → | 2 |
| was | 2 | → | 1 → 2 |
| with | 1 | → | 2 |

# Processing Boolean Queries

- Processing Boolean Queries of the form, for example
  - T1 OR T2
  - T1 AND T2
  - T2 AND T3 OR (T4 OR NOT T5)
- Example Query
  - Brutus AND Calpurnia

# Processing Boolean Queries

- Example Query
  - Brutus AND Calpurnia
    - Locate “Brutus” in the Dictionary
    - Retrieve its Postings
    - Locate “Calpurnia” in the Dictionary
    - Retrieve its Postings
    - Get the intersection of the two posting lists

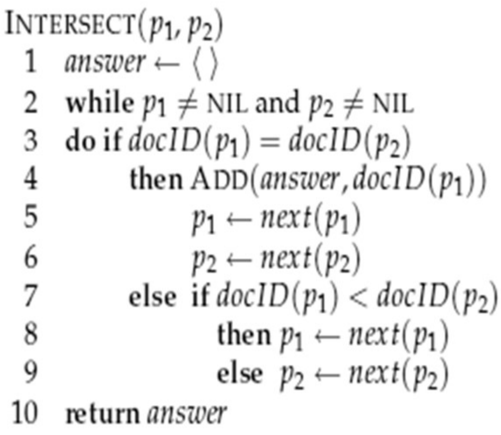


Brutus ⟶ 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

Calpurnia ⟶ 2 → 31 → 54 → 101

Intersection ⟹ 2 → 31

## Intersect (Posting List Algorithm)



```
INTERSECT($p_1, p_2$)
answer ← ⟨ ⟩
while $p_1 \neq$ NIL and $p_2 \neq$ NIL
do if $docID(p_1) = docID(p_2)$
      then ADD(answer, $docID(p_1)$)
           $p_1 \leftarrow next(p_1)$
           $p_2 \leftarrow next(p_2)$
      else if $docID(p_1) < docID(p_2)$
             then $p_1 \leftarrow next(p_1)$
             else $p_2 \leftarrow next(p_2)$
return answer
```

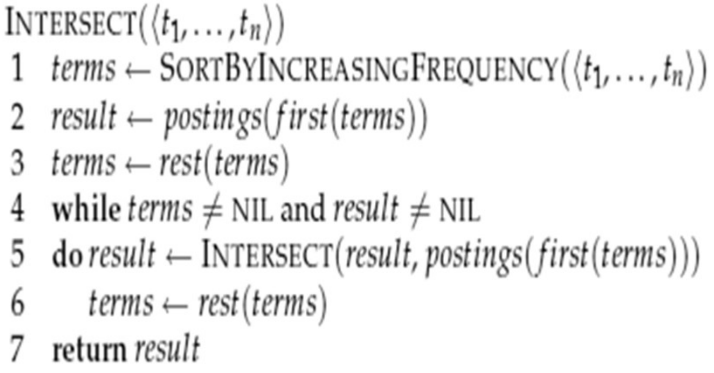
## Query Optimization

- Query optimization is the process of selecting how to organize the work of answering a query so that the least total amount of work needs to be done by the system.
- A major element of this for Boolean queries is the order in which postings lists are accessed.
- What is the best order for query processing?
  - Brutus AND Caesar AND Calpurnia

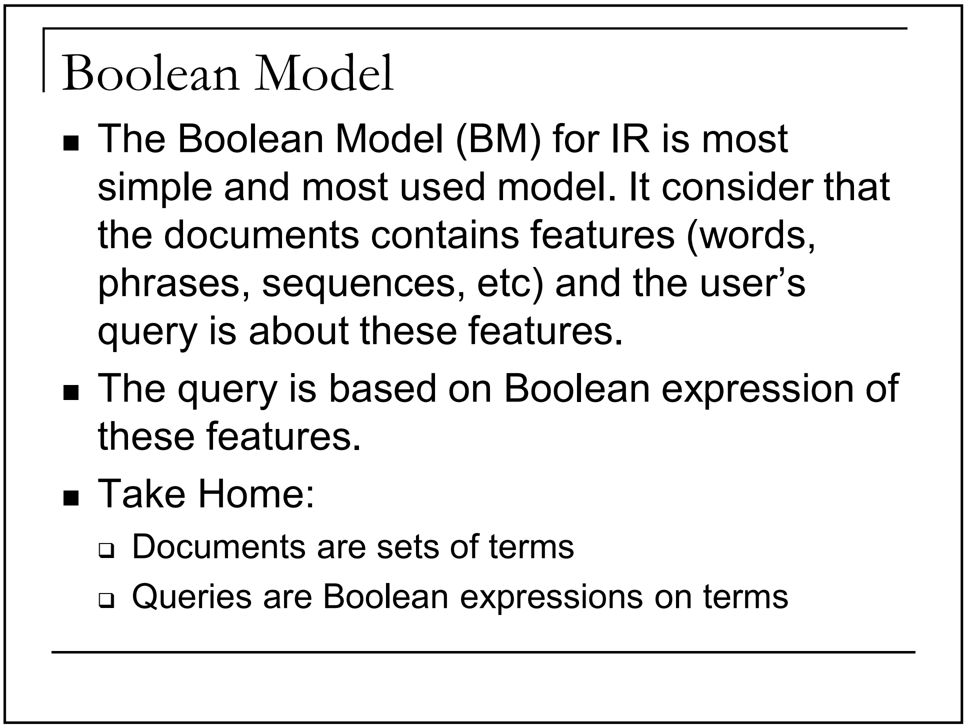
## Query Optimization

- What is the best order for query processing?
  - Brutus AND Caesar AND Calpurnia
    - For each of the t terms, we need to get its postings, then AND them together.
    - The standard heuristic is to process terms in order of increasing document frequency: if we start by intersecting the two smallest postings lists, then all intermediate results must be no bigger than the smallest postings list, and we are therefore likely to do the least amount of total work.

## Intersect (Optimized)

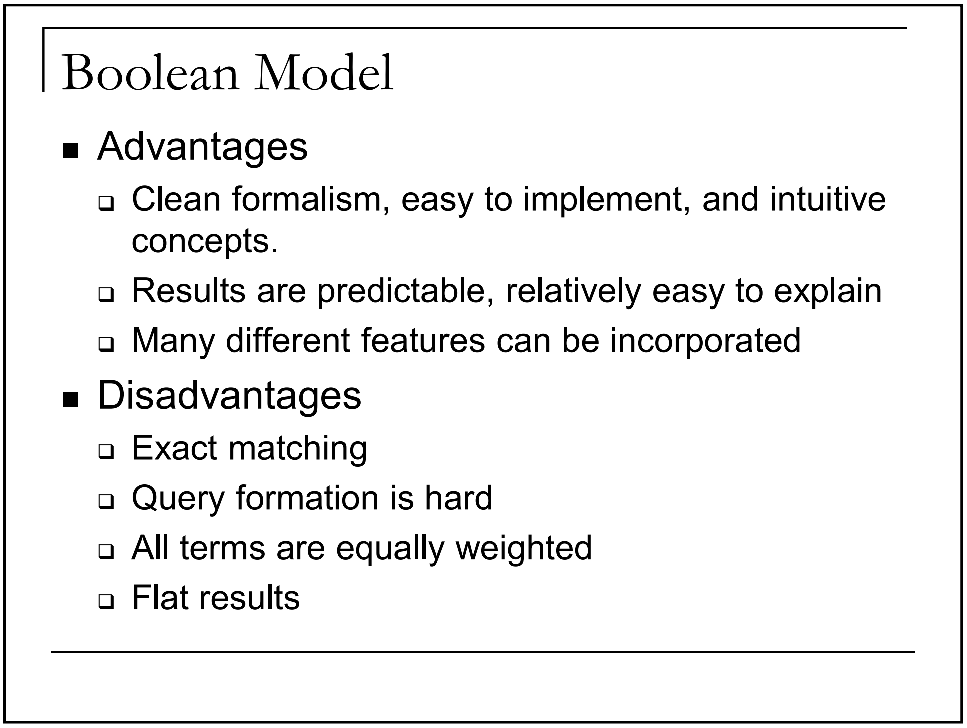


```
INTERSECT($\langle t_1, \ldots, t_n \rangle$)
terms ← SORTBYINCREASINGFREQUENCY($\langle t_1, \ldots, t_n \rangle$)
result ← postings(first(terms))
terms ← rest(terms)
while terms ≠ NIL and result ≠ NIL
do result ← INTERSECT(result, postings(first(terms)))
    terms ← rest(terms)
return result
```



## Boolean Model

- The Boolean Model (BM) for IR is most simple and most used model. It consider that the documents contains features (words, phrases, sequences, etc) and the user's query is about these features.
- The query is based on Boolean expression of these features.
- Take Home:
  - Documents are sets of terms
  - Queries are Boolean expressions on terms



## Boolean Model

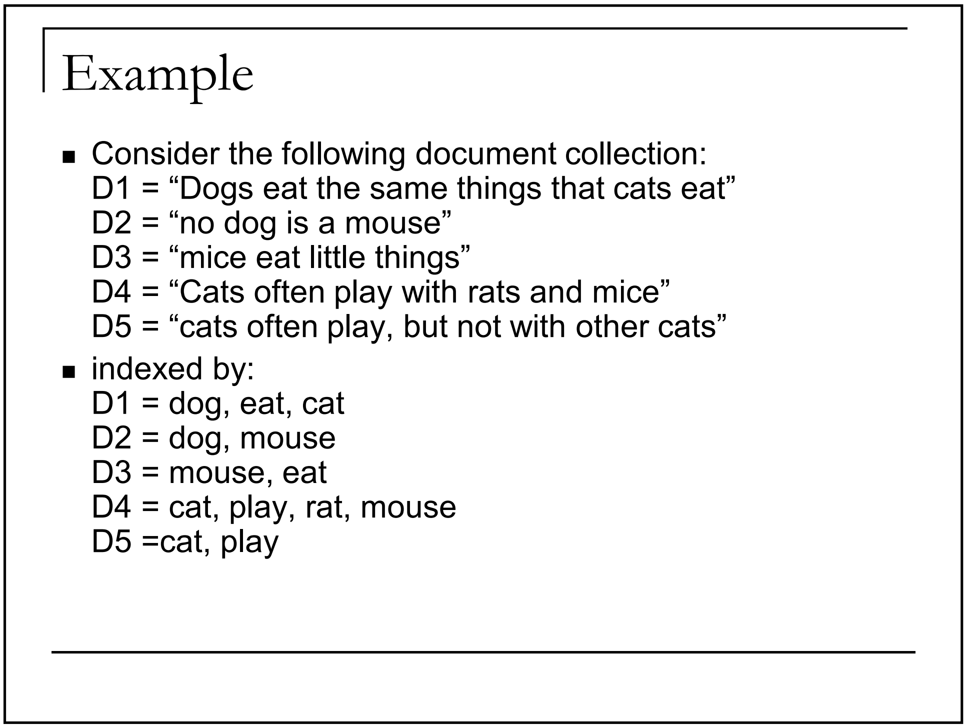
- Advantages
  - Clean formalism, easy to implement, and intuitive concepts.
  - Results are predictable, relatively easy to explain
  - Many different features can be incorporated
- Disadvantages
  - Exact matching
  - Query formation is hard
  - All terms are equally weighted
  - Flat results

# Observations on Boolean Model

- Retrieval based on binary decision criteria with no notion of partial matching
- No ranking of the documents is provided (absence of a grading scale)
- Information need has to be translated into a Boolean expression which most users find awkward
- The Boolean queries formulated by the users are most often too simplistic
- As a consequence, the Boolean model frequently returns either too few or too many documents in response to a user query

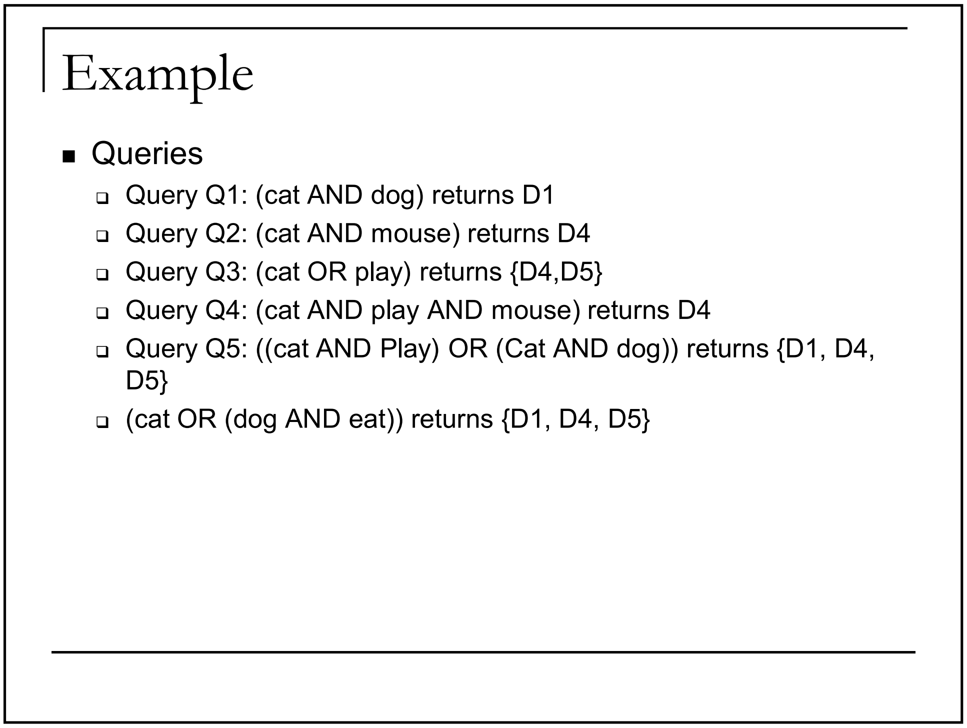
# Observations on Boolean Model

- The Boolean model imposes a binary criterion for deciding relevance
- The question of how to extend the Boolean model to accomodate partial matching and a ranking has attracted considerable attention in the past
- Two extensions of boolean model:
  - Extended Boolean Model
  - Fuzzy Set Model



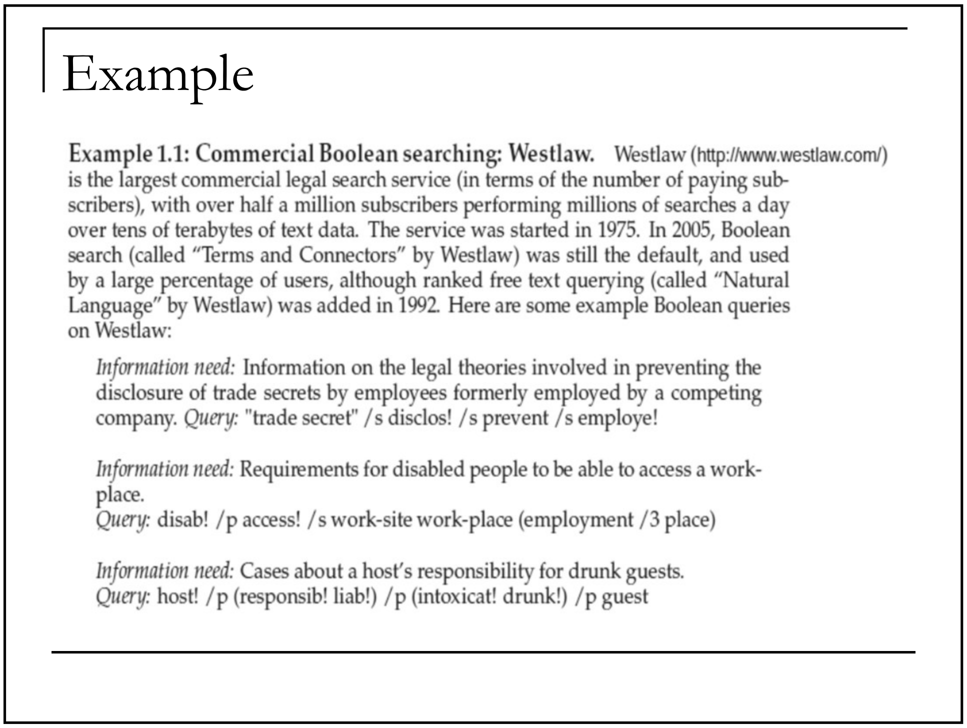
# Example

- Consider the following document collection:
  D1 = “Dogs eat the same things that cats eat”
  D2 = “no dog is a mouse”
  D3 = “mice eat little things”
  D4 = “Cats often play with rats and mice”
  D5 = “cats often play, but not with other cats”
- indexed by:
  D1 = dog, eat, cat
  D2 = dog, mouse
  D3 = mouse, eat
  D4 = cat, play, rat, mouse
  D5 =cat, play



# Example

- Queries
  - Query Q1: (cat AND dog) returns D1
  - Query Q2: (cat AND mouse) returns D4
  - Query Q3: (cat OR play) returns {D4,D5}
  - Query Q4: (cat AND play AND mouse) returns D4
  - Query Q5: ((cat AND Play) OR (Cat AND dog)) returns {D1, D4, D5}
  - (cat OR (dog AND eat)) returns {D1, D4, D5}



# Example

**Example 1.1: Commercial Boolean searching: Westlaw.** Westlaw (http://www.westlaw.com/) is the largest commercial legal search service (in terms of the number of paying subscribers), with over half a million subscribers performing millions of searches a day over tens of terabytes of text data. The service was started in 1975. In 2005, Boolean search (called "Terms and Connectors" by Westlaw) was still the default, and used by a large percentage of users, although ranked free text querying (called "Natural Language" by Westlaw) was added in 1992. Here are some example Boolean queries on Westlaw:

*Information need:* Information on the legal theories involved in preventing the disclosure of trade secrets by employees formerly employed by a competing company. *Query:* "trade secret" /s disclos! /s prevent /s employe!

*Information need:* Requirements for disabled people to be able to access a workplace.
*Query:* disab! /p access! /s work-site work-place (employment /3 place)

*Information need:* Cases about a host's responsibility for drunk guests.
*Query:* host! /p (responsib! liab!) /p (intoxicat! drunk!) /p guest