INFSCI 2140
Information Storage and Retrieval
Lecture 2: Models of Information Retrieval: Boolean model

Peter Brusilovsky
http://www2.sis.pitt.edu/~peterb/2140-051/

Final Group Projects

- Groups of variable size - match the project complexity
- Goal - learn more about IR, do some useful and practical work
- Educational IR system
- Exploring IR aspects
- Analysis

What are your strong sides?

- Are you good in:
  - Reading and analyzing text
  - Programming in C or Java
  - Database programming
  - Web programming
  - Developing educational material
  - Graphic arts and design

Educational IR system

- Small educational programs: demo-simulation / assessment / ITS. Learn nuts and bolts of some complex aspect and help others to learn
- Here are some examples from the past semesters
  - Boolean query
  - Boolean document matching
  - Evaluation measures
  - Vectors, distances, similarities

Exploring IR aspects

- Any part of IR process to practice
  - Document preparations and processing
  - Simple search engine
  - Similarity-based navigation
  - Clustering
  - Adaptive recommendation
- Possible domains
  - C programming
  - Information Retrieval

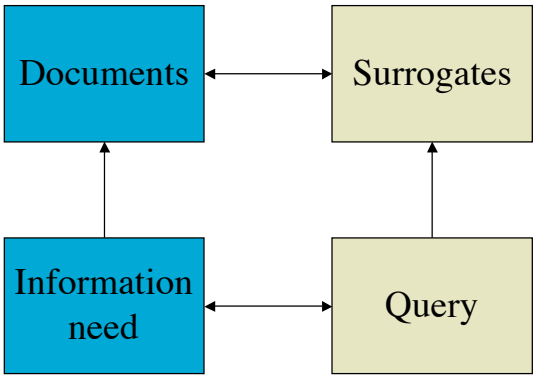
Analysis

- Select a small field of interest on the frontier of IR, analyze a few relevant papers and systems, write an essay (indexed!), prepare a short talk for the last lecture
  - Improving Web search
  - Machine learning for IR
  - Some type of information visualization
  - Clustering
  - Web recommenders...

Overview

- From an information need to documents
- Information Retrieval Models
- Boolean Model
  - Boolean queries
  - DNF and CNF
  - Boolean matching

From information need to documents



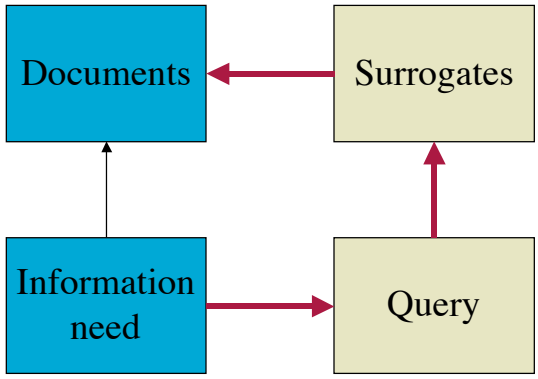
Documents and information need

- IA Goal: user need documents that are relevant to information need
- Ways to find them depend on
  - Is the information need explicit or implicit?
  - Is it a one-time, iterative or continuous process?
  - Do the documents form an unstructured pool or structured space?

Explicit information need

- Explicit need is expressed in a query
- Searching and iterative searching
  - Ad-hoc IR
- Information filtering and SDI
- “Crawling” and infobots
- Long-term information services (FAB)

Searching



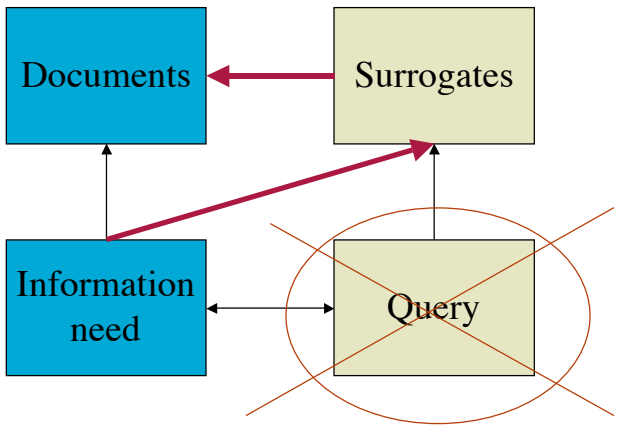
Non-explicit information need

- Incremental search with relevance feedback
- Collaborative filtering
- Browsing
- Browsing / search combination
  - Search/browsing - Golovchinsky
  - Similarity navigation - Tudhope
  - Area search while browsing - Lieberman

When / Why browsing?

- Hard to express an information need in a formal query
- Documents are connected to each other
- Humans aren't good in formulating requests, but good in recognizing
- Opportunity to *navigate* to similar documents

Browsing



Classic ad-hoc IR

- Documents form an unstructured *document space*
- Information need made explicit in a form of a *query*
- User searches once - one query against the document space (*ad-hoc IR*)
- The IR system returns an *ordered* subset of documents that are *relevant* to the query

User's prospect

- Query
  - a question in natural language
  - a list of terms
- Matching
  - The obvious exact match
  - The range match
  - Approximate match ? *("tell me something about…", "tell me something that I don't know?")*

Problems with matching

- The fact that a document contains a term requested in a query doesn't mean that the document should be retrieved:
  - we have to consider all the term and the condition expressed in the query
  - it doesn't mean that the document is strongly related to the term
  - a document can't be suitable for other reasons (age for example)

Exact Match

- Examples
  - all the documents authored by …
  - all the documents that have in the title the word …
- It is possible with well structured documents and databases

Range Match

- Example
  - all the documents authored from “date1” to “date2” .
- Range match works with numerical databases

Approximate match

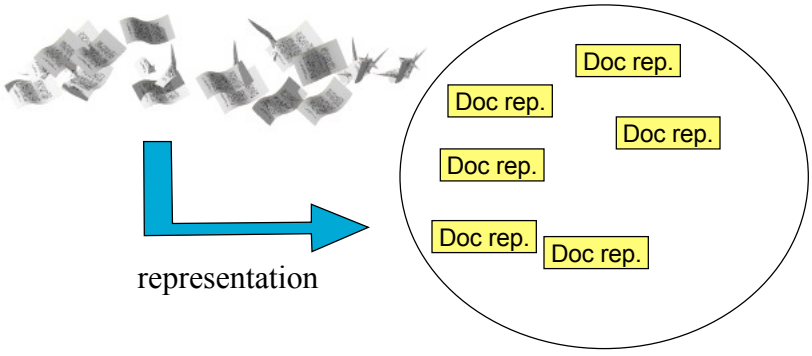
- Necessary if the information need is not precise and if the database is not well organized.
- Sometimes it is necessary to combine the different kind or queries.

Designer's prospects for Q & M

- Can we consider query as a document?
- No
  - let's consider a query as a characteristic function defined for document space
- Yes
  - let's put a query into a space and calculate "closeness" or similarity
- In any case we need a *measure* for relevance ("closeness", similarity…)

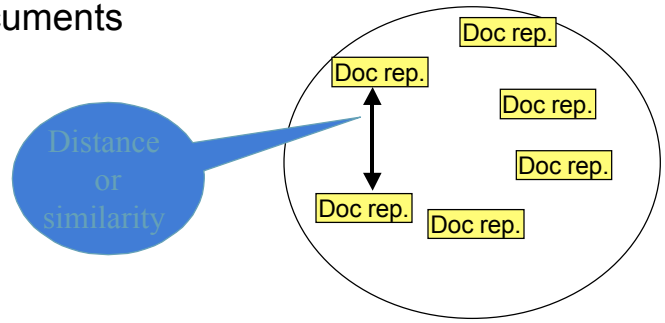
Document Space

- For our purposes we call document space the set of document



Document Space

- Document space is organized in some way. For example it could be possible to calculate a distance or a similarity between different documents



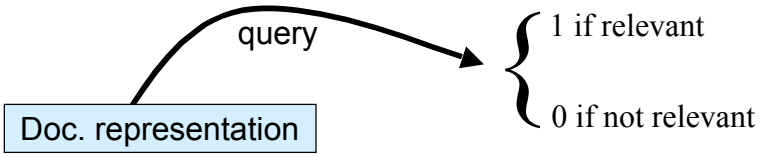
Models: Classic and New

- Boolean Model
  - Classic
  - Extended
  - Fuzzy
- Vector Model
  - Classic
  - Others (generalized, LSI, Neural Networks)
- Probabilistic Model

We will work by models, discussing Q&M for each model separately

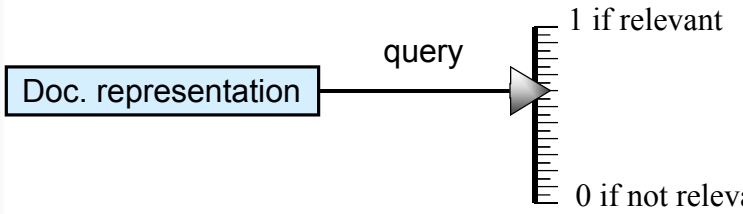
The query is NOT a part of the document space

- The query defines an evaluation function: a function that have a value 0 or 1 depending if the document is relevant or is not relevant



The query is NOT a part of the document space

- In more sophisticated systems the evaluation function can have values **in** the interval [0,1], allowing to rank documents.



Boolean Model

- Set Theoretic Approach
- Documents form a large set
- A query defines a subset (I.e., 0 or 1)
- Elementary query has a clearly defined subset
- To make a complex query one can use Boolean functions for set operation

Boolean queries

- Boolean queries are based on Boolean Algebra
- Terms are join using connectives as AND, OR, NOT
- The search can be expanded using stemming, thesaurus or list of related terms
- Example:

restaurants AND (mideastern OR vegetarian) AND inexpensive

Elementary Query

- Formatted fields
  - Exact match (Year = 1999)
  - Range match (Price < $50)
- Text fields
  - Word inclusion (Programming in Title)
  - Stemming/wildcards (Program$ in Title)
  - Phrase inclusion (“Programming language” in Title)
  - Approximate match (sounds like “John”)

Boolean Operators

- Q1 AND Q2
  - Documents that are in BOTH sets: Q1 and Q2
- Q1 OR Q2
  - Documents that are in at least in one set: Q1 or Q2
- NOT Q1
  - All documents except the one in set Q1

Other Boolean Operators

- Q1 \ Q2
  - Logical “minus” all documents from Q1 except those that belong to Q2
  - Used also as “binary NOT” (Q1 NOT Q2)
- Q1 XOR Q2
  - Exclusive OR - documents that belong to exactly one set: Q1 or Q2, but not both
  - In other words (Q1 OR Q2) \ (Q1 AND Q2)

Some Special Operators

- Proximity (TNT)
  - (Information within 1 word from Retrieval)
  - This is really a special elementary query to structured text fields
- NOF (N of)
  - 2 of (Sashimi, Sushi, Shabu-Shabu)
  - This is a simple form that can be expressed by a regular query

Complex Queries

- We can use braces to determine the order of operation
  - (NOT (Q1 OR Q2 OR Q3) AND (Q4 AND NOT Q5) OR Q6
- Complex queries are hard to write, understand and perform
- Normalization - converting a query to one of the *normal forms*

Normal Forms

- Every Boolean query can be recast into:
- Conjunctive Normal Form (CNF)
  - Q1 AND Q2 AND Q3 AND Q4
  - Each Q is a *disjunct*
- Disjunctive Normal Form (DNF)
  - Q1 OR Q2 OR Q3 OR Q4
  - Each Q is a *conjunct*

Disjunctive Normal Form (DNF)

- Term: positive or negated elementary fact
  - Year = 1999; NOT (Director = Spielberg);
  - Fire; NOT Retrieval
- Conjuncts: conjunction of terms
  - Year = 1999 AND NOT (Director = Spielberg)
- Query: disjunction of conjuncts
  - (a AND b) OR (c AND d) OR (NOT a AND d)

Boolean queries: DNF

- The advantage is that the query can be split into many different queries. The results of the different queries is merged to produce the response to the original query:

(a AND b) OR — First query
(c AND d) OR — Second one
(NOT a AND d) — Third one

Conjunctive Normal Form (CNF)

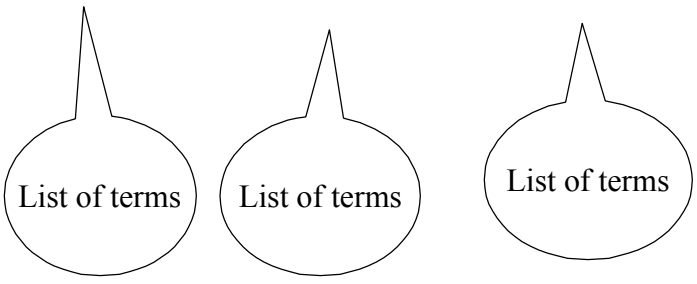
- Term: positive or negated elementary fact
  - Year = 1999; NOT (Director = Spielberg);
  - Fire; NOT Retrieval
- Disjuncts: disjunction of terms
  - Year = 1999 OR NOT (Director = Spielberg)
- Query: conjunction of disjuncts

(a OR b) AND (c OR d) AND (NOT a OR d)

Boolean queries: CNF

- Using a thesaurus can be easy to expand a CNF query to have a broader query:

(a OR b) AND (c OR d) AND (NOT a OR d)



Normalization

- Convert each Boolean function to NFs
- Key: Truth Table
  - Example: (A OR NOT B) AND C
- Normalization to DNF
  - Consider T rows
- Normalization to DNF
  - Consider F rows

A	B	C	not B	Aor(notB)	[Aor(notB)]and C
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	1

Example: Normalize the query [Aor(notB)] and C to DNF

A	B	C	not B	Aor(notB)	[Aor(notB)]and C	
0	0	0	1	1	0	
0	0	1	1	1	1	←
0	1	0	0	0	0	
0	1	1	0	0	0	
1	0	0	1	1	0	
1	0	1	1	1	1	←
1	1	0	0	1	0	
1	1	1	0	1	1	←

Example: Normalize the query [Aor(notB)] and C to DNF

A	B	C	not B	Aor(notB)	[Aor(notB)]and C
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	1

[(not A)and(notB)andC] or
[A and (notB) and C] or
[A and B and C]

Normalization to CNF

- A way to obtain a CNF is to use the false (0) row of the table and simplifying them using DeMorgan's laws and double negation law

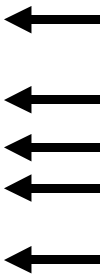
DeMorgan's laws

1st law: `not(A and B) = (not A) or (not B)`
2nd law: `not (A or B) = (not A) and (not B)`

double negation
`not(not A) = A`

Example: Normalize the query [Aor(notB)] and C to CNF

A	B	C	not B	Aor(notB)	[Aor(notB)]and C
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	1

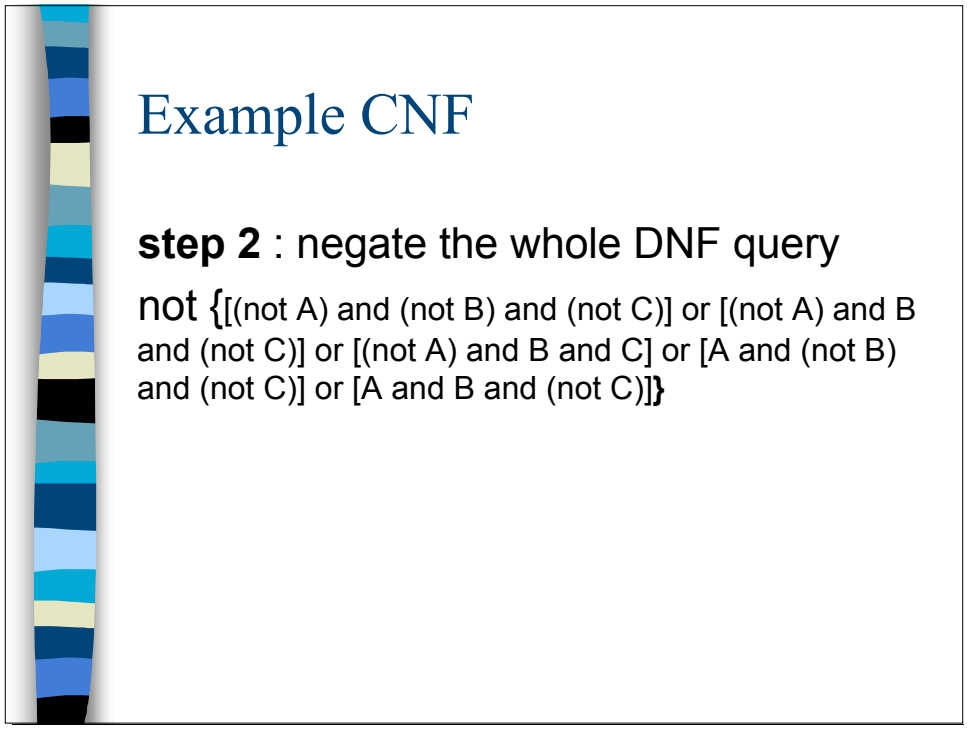


Example CNF

Step 1: form the full DNF query using the false row of the table

A	B	C	not B	Aor(notB)	[Aor(notB)]and C
0	0	0	1	1	0
0	0	1	1	1	1
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	0	1	0
1	1	1	0	1	1

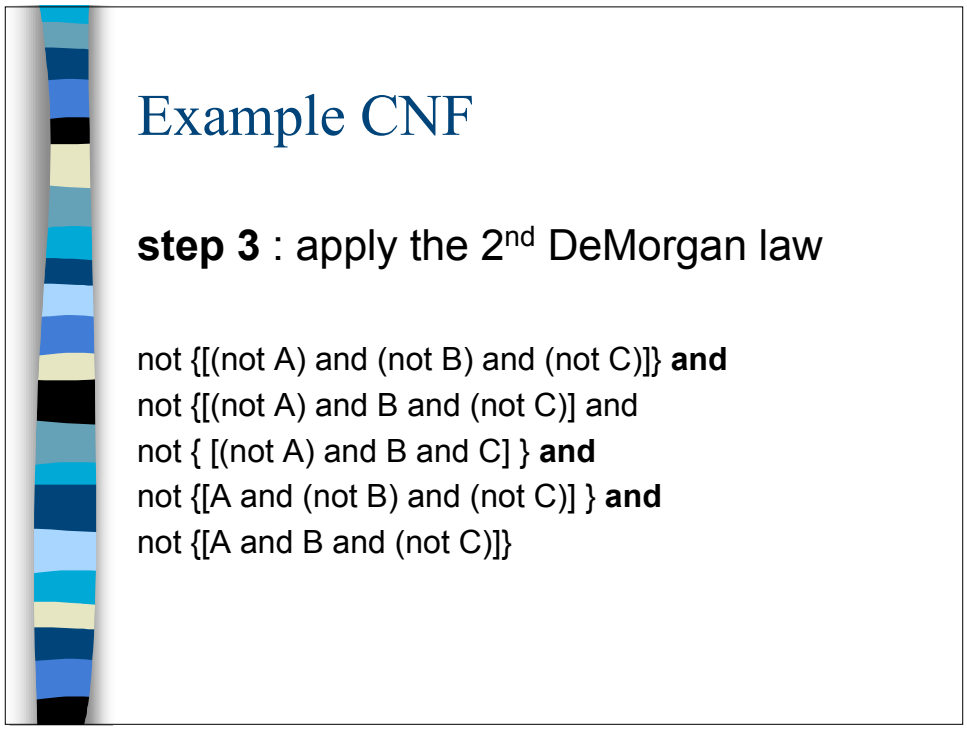
[(not A) and (not B) and (not C)] or
[(not A) and B and (not C)] or
[(not A) and B and C] or
[A and (not B) and (not C)] or
[A and B and (not C)]



Example CNF

step 2 : negate the whole DNF query

not {[(not A) and (not B) and (not C)] or [(not A) and B and (not C)] or [(not A) and B and C] or [A and (not B) and (not C)] or [A and B and (not C)]**}**



Example CNF

step 3 : apply the 2nd DeMorgan law

not {[(not A) and (not B) and (not C)]} **and**
not {[(not A) and B and (not C)] and
not { [(not A) and B and C] } **and**
not {[A and (not B) and (not C)] } **and**
not {[A and B and (not C)]}

Example CNF

step 4 : apply the 1st DeMorgan law

[not (not A) **or** not(not B) **or** not(not C)] **and**
[not (not A) **or** (not B) **or** not (not C)] **and**
[not (not A) **or** (not B) **or** (not C)] **and**
[(not A) **or** not(not B) **or** not(not C)] } **and**
[(not A) **or** (not B) **or** not(not C)]

Example CNF

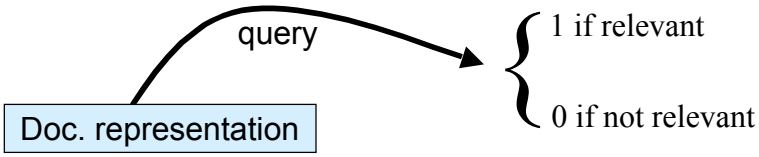
step 5 : apply the double negation law

[A **or** B **or** C] **and**
[A **or** (not B) **or** C] **and**
[A) **or** (not B) **or** (not C)] **and**
[(not A) **or** B **or** C] } **and**
[(not A) **or** (not B) **or** C]

Boolean Matching

- The query is not a part of the document space. It is a Boolean expression that represent some conditions on the index terms or representation of the desired documents

```
(computer AND memory) AND (NOT chip)
```



Boolean Matching

- How to define match for elementary query?
- How to define match for main boolean operations?
  - Q1 AND Q2
  - Q1 OR Q2
  - NOT Q1
- How to order?

Efficiency for elementary queries

- Fast ways to match
- Inverted Indexing (primary and secondary key):
  - Fixed value search (year = 1999)
  - Full term search (Adaptive in text)
- Hashing
- Range search (Year < 1990)
  - Sorted array search

Boolean queries and inverted file

Doc1: `the cat is on the mat`

Doc2: `the mat is on the floor`

Inverted file

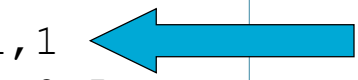
```
cat:doc1,1
floor:doc2,5
mat:doc1,5;doc2,1
```

Boolean queries and inverted file

Query : cat

Inverted file

```
cat:doc1,1
floor:doc2,5
mat:doc1,5;doc2,1
```



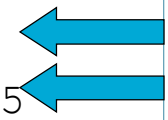
Answer : doc1

Boolean queries and inverted file

Query: cat AND floor

Inverted file

```
cat:doc1,1
floor:doc2,5
mat:doc1,5;doc2,1
```



The algorithm looks at these two lists and try to found elements in common

Answer : No document

Boolean queries and Term-document matrix

- Rows represent document terms
- Columns represent documents

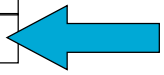
Doc1: `the cat is on the mat`
Doc2: `the mat is on the floor`

	Doc1	Doc2
cat	1	0
floor	0	1
mat	1	1

Boolean queries and Term-document matrix

Query: Mat

	Doc1	Doc2
cat	1	0
floor	0	1
mat	1	1



The algorithm looks to this row

Answer: Doc1 and Doc2

More efficiency

- Slower ways to match
- Full comparison
  - Exact comparison (size < 5.9)
  - Text matching
- Combination
  - Do some fast search first, defining a smaller subset, then comprehensive search

Matching

- The case of Exact Match (Boolean characteristic function)
  - The result is either in or not
- Softening Boolean queries
  - The case of A or B or C
  - Document that satisfy A and B is better than A but worse than A and B and C

Homework (Part 1)

Using the system at the URL (Boolean query servlet)
http://kt1.sis.pitt.edu:8080/ir/uquery/matrix.html
answer to the following questions (and report the query used):

- Did Mr. Flanagan write any book about "algorithms"?
  Answer to the question and report the query used
- Did. Prof. Brusilovsky write any book about Java ?
  Answer to the question and report the query used
- List the books about Java or Perl language? (look at the title)
  Answer to the question and report the query used
- List the books related to Java were published in 1999?
  Answer to the question and report the query used
- List the books published by Java Soft were not authored by Flanagan ?
  Answer to the question and report the query used

Homework (Part 2)

- Take an example of an advanced online search system (the one you have used in HW1 or any other)retrieval systems and analyze it from the prospect of IR models presented at the lectures 3 and 4. What kind of model this system is based upon? Can you recognize one of the models we have learned? What kind of queries the system allows? How you could use boolean operators or perform Boolean search with this engine? What kinds of usual Boolean operators it has? Whay kind of proximity operators it uses? Does it provide a form-based search for users unprepared to work with complex boolean expressions? We have discussed all that (and a number of other issues) and now you should be able to apply your knowledge for thinking in a real world context. In addition to that, please, specify what other conditions you can specify in your search request to fine-tune the matching. Does the system enable you to restrict the search to a subset of documents or to consider for matching only a specific areas of documents?