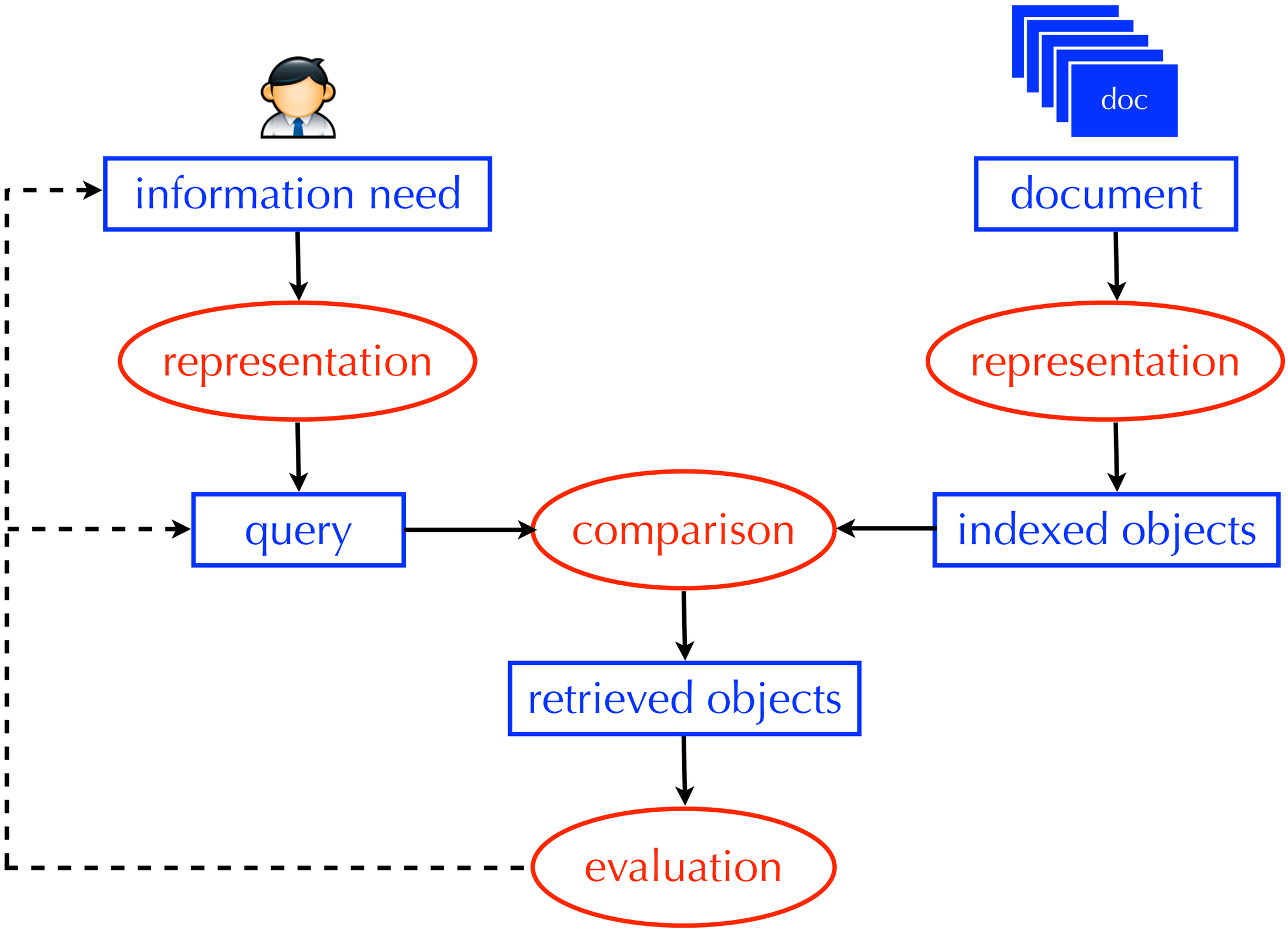
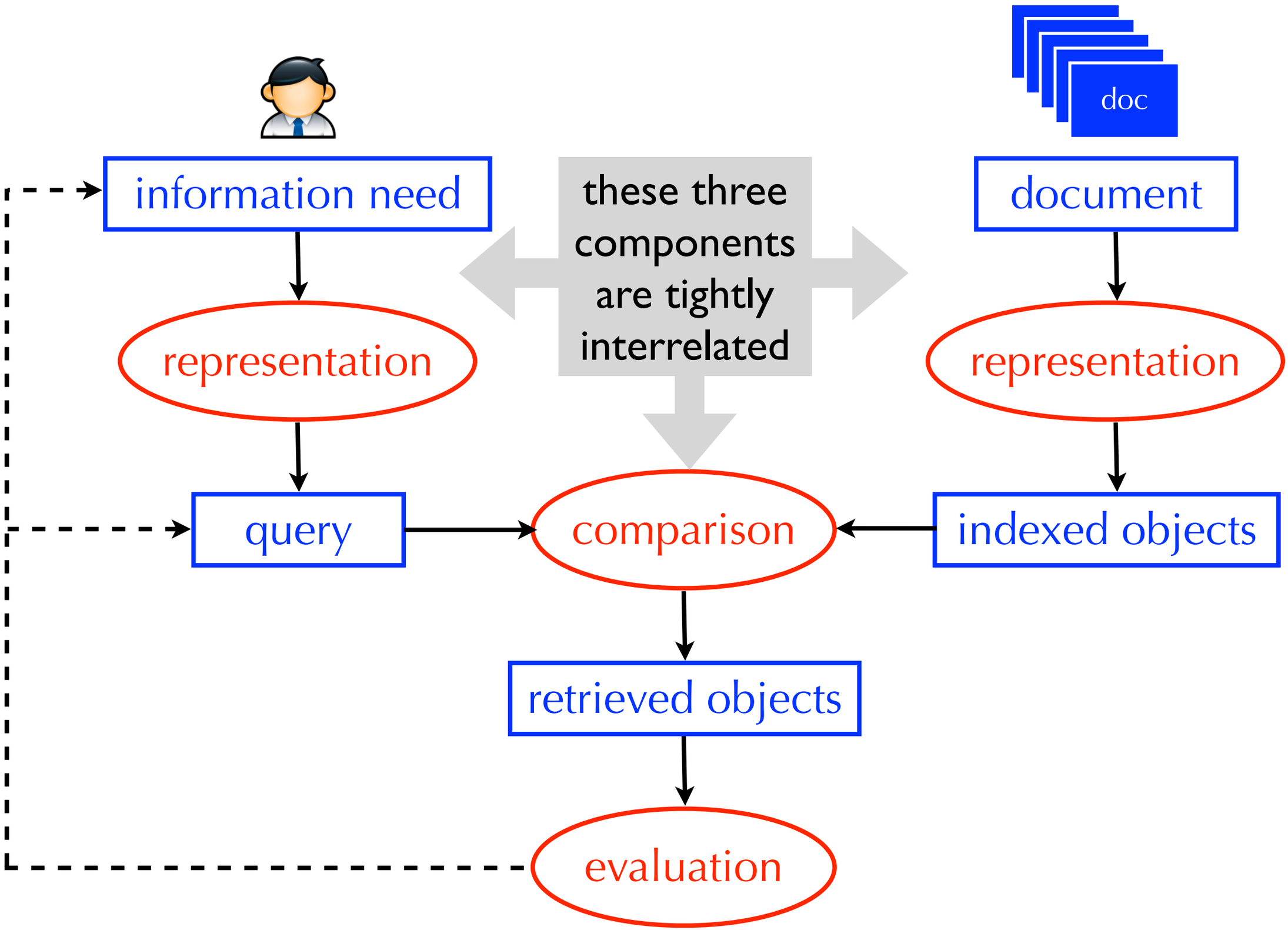
Indexing and Query Processing

Jaime Arguello
INLS 509: Information Retrieval
jarguell@email.unc.edu

Basic Information Retrieval Process



Basic Information Retrieval Process



Indexing and Query Processing

- Query language: defines how users can describe their information needs to the system (e.g., boolean queries)
- Document representation: determines what goes in the index (e.g., terms, term frequencies, etc.)
- Index: facilitates quickly finding the documents that match the query
- Retrieval model: decides whether a document is relevant to the query (and possibly its degree of relevance)

Full-text Representation

(variable-length) inverted lists with term-frequencies

a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
df=1020	*df=2*	*df=1*	*df=3*	*df=102*	...	*df=43*
1, 3	3033, 1	254, 1	43, 5	2, 44	...	323, 6
2, 4	5463, 1		576, 6	3, 32	...	506, 5
::			1878, 3	::	...	::
5023, 55				4543, 12	...	2421, 2

- Bag of words representation (no term-location information)
- Facilitates unranked and ranked boolean retrieval
- Facilitates boolean operators AND, OR, and AND NOT
- Facilitates efficient query processing
  - merging inverted lists can be done quickly

What about other query operators?

- A boolean query language with only AND, OR and AND NOT is too restrictive
- Difficult to balance precision and recall
- Popular systems that employ boolean retrieval (e.g., WestLaw) include <u>other</u> operators

1. proximity operators: impose constraints on query-terms appearing close to each other (ordered or unordered)
2. field restriction: impose constraints on terms appearing in particular parts of the document
3. wild-card operators: impose constraints on matching query-terms with index-terms

Proximity Operators

Proximity Operators
1) ordered window

- A OW/N B: A must appear no more than N terms before B
- Example: (*information* OW/2 *science*)
  - "information library science"
  - "information science"
  - "information and library science"
  - "science of information"

Proximity Operators
1) ordered window

- A OW/N B: A must appear no more than N terms before B
- Example: (*information* OW/2 *science*)
  - "information library science" ✓
  - "information science" ✓
  - "information and library science" ✗
  - "science of information" ✗

Proximity Operators

2) phrase operator

- "A B": A must appear immediately before B
- Example: "*information science*"
  - "information library science" 
  - "information science" 
  - "information and library science" 
  - "science of information" 
- equivalent to (*information* OW/1 *science*)

Proximity Operators

3) unordered window

- A UW/N B: A must appear no more than N terms before or after B
- Example: (*information* UW/2 *science*)
  - "information library science"
  - "information science"
  - "information and library science"
  - "science of information"

Proximity Operators

3) unordered window

- A UW/N B: A must appear no more than N terms before or after B
- Example: (*information* UW/2 *science*)
  - "information library science" 
  - "information science" 
  - "information and library science" 
  - "science of information" 

Full-text Representation

(variable-length) inverted lists with term frequencies

a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
df=1020	*df=2*	*df=1*	*df=3*	*df=102*	...	*df=43*
1, 3	3033, 1	254, 1	43, 5	2, 44	...	323, 6
2, 4	5463, 1		576, 6	3, 32	...	506, 5
::			1878, 3	::	...	::
5023, 55				4543, 12	...	2421, 2

- We cannot handle proximity operators with this index
- What additional information do we need?

Full-text Representation

positional index

Information	*Library*	*Science*
df=3	*df*=4	*df*=3
2, 1 [2]	2, 2 [3,44]	2, 3 [4,45,78]
22, 1 [31]	22, 2 [32,66]	22, 3 [33,67,78]
45, 1 [2]	45, 3 [3,46,101]	45, 2 [5,34]
	3421, 1 [2]	

- *docid, tf* [*pos_1, pos_2, ..., pos_tf*]
- Which document(s) match "*information library science*"?
- What about (*information* OW/2 *science*)?

Field Restrictions

Structured Documents

Automatic Boolean Query Suggestion for Professional Search

Youngho Kim
yhkim@cs.umass.edu

Jangwon Seo
jangwon@cs.umass.edu

W. Bruce Croft
croft@cs.umass.edu

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst, MA 01003, USA

ABSTRACT

In professional search environments, such as patent search or legal search, search tasks have unique characteristics: 1) users interactively issue several queries for a topic, and 2) users are willing to examine many retrieval results, i.e., there is typically an emphasis on recall. Recent surveys have also verified that professional searchers continue to have a strong preference for Boolean queries because they provide a record of what documents were searched. To support this type of professional search, we propose a novel Boolean query suggestion technique. Specifically, we generate Boolean queries by exploiting decision trees learned from pseudo-labeled documents and rank the suggested queries using query quality predictors. We evaluate our algorithm in simulated patent and medical search environments. Compared with a recent effective query generation system, we demonstrate that our technique is effective and general.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – Query Formulation, Search Process.

General Terms

Algorithms, Experimentation.

Keywords

Boolean query suggestion, prior-art search, patentability search.

1. INTRODUCTION

Query suggestion is an effective and practical way to help users formulate queries [15, 16]. While there have been many studies on how to provide alternative queries in general web search [15, 16], little work has been done about suggestion for domain-specific search, e.g., patent retrieval, legal search, and medical information search. Many of the users in such domains are search professionals, e.g., patent examiners and information specialists in companies and law firms, who perform specialized search tasks such as prior-art search and legal discovery. Query suggestion techniques should be designed for the unique search characteristics of these domains. For example, professional search is typically more recall-oriented than consumer search. In the patent validity task, for example, patent examiners formulate search queries from a new patent to validate its patentability, and generally spend about 12 hours to complete a single task by examining approximately about 100 patent documents retrieved by 15 different queries on average [1]. Another typical characteristic of professional search is the need to document the searches that are carried out.

For a number of reasons, both historic and technical, Boolean queries are particularly common in professional search. For example, in patent search, recent surveys [1, 2] revealed that the use of Boolean operators is one of the most important features to formulate effective queries from the perspective of patent professionals. Also, according to [2], most patent professionals who participated in the survey did not regard query term weighting and query expansion as important whereas 96.3% of participants agreed that Boolean operators are necessary. This is not because Boolean queries are the most effective. In fact, a number of studies over the years (e.g., [5, 6, 7, 9, 11]) have shown that "keyword" queries are often significantly more effective. Boolean queries, however, are easy for information professionals to manipulate and are essentially self-documenting in that they define precisely the set of documents that are retrieved.

Despite the importance of Boolean queries in professional search, there has not been much research on helping information professionals formulate those queries. Tseng and Wu [3] indicated that the provision of suggested search vocabulary would be helpful in patent search. Other studies on prior-art search that automatically generate queries from patent text (e.g., [6, 7]) did not investigate Boolean query suggestion. Current government or commercial patent search systems [1] used by information professionals all support Boolean queries but not query suggestion.

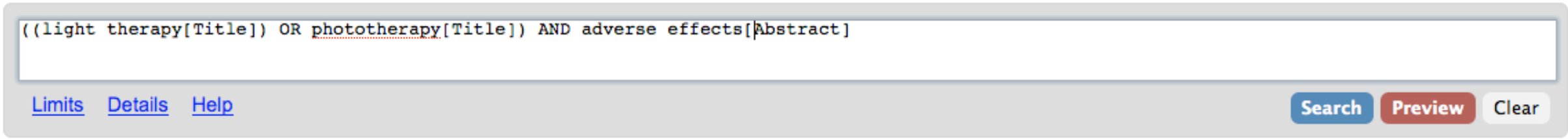
In this paper, we propose a method to suggest Boolean queries for professional search. We define a Boolean query as the sequence of terms associated by conjunction (AND) where each term can be prefixed by negation (NOT). Although the OR operator is often used by professionals to indicate synonym groups, the retrieval evidence shows that AND and NOT have much more impact on effectiveness in domains such as patent search with very detailed documents (e.g., [4]). Adding synonym structure is left for future work. Although the suggested Boolean queries can be generated and used with any search engine, we use a simple statistical Boolean retrieval model for our experiments (explained in Section 5). We do not adopt any additional query processing and term weighting because those features are not generally preferred by professionals and not supported by commercial search systems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR'11, July 24–28, 2011, Beijing, China.
Copyright 2011 ACM 978-1-4503-0757-4/11/07…$10.00.

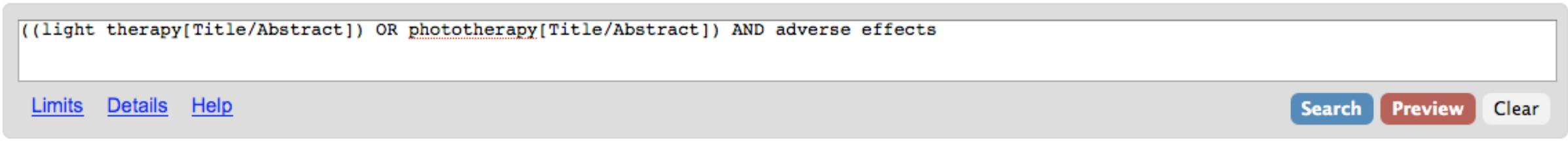
[1]PATENT SCOPE (http://www.wipo.int/patentscope/), PatFT (http://patft.uspto.gov/), DELPHION (http://www.delphion.com/)

Field-Restricted Boolean Retrieval

- PubMed advanced search allows users to build a boolean query that searches different fields:



- ...or field combinations:



- How would we implement this using techniques we already know?

Field-Restricted Boolean Retrieval

- Solution 1: build a separate index for each field type

title index

a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
df=820	*df=1*	*df=1*	*df=3*	*df=12*	...	*df=43*
1, 3	3033, 2	254, 1	43, 1	2, 2	...	323, 1
2, 4			576, 1	3, 1	...	506, 2
::			1878, 1	::	...	::
5023, 55				4543, 2	...	2421, 2

author index

a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
df=10	*df=0*	*df=0*	*df=0*	*df=1*	...	*df=0*
1, 1				34, 1	...	
2, 2					...	
::					...	
6033, 2					...	

body index

a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
df=1020	*df=2*	*df=1*	*df=3*	*df=102*	...	*df=43*
1, 3	3033, 3	254, 1	43, 5	2, 44	...	323, 6
2, 4	5463, 1		576, 6	3, 32	...	506, 5
::			1878, 3	::	...	::
5023, 55				4543, 12	...	2421, 2

Field-Restricted Boolean Retrieval

- Solution 1: process each field-specific subquery using the appropriate index and merge the results (as usual)
- (*phototherapy*)[*title*] OR (*phototherapy*)[*abstract*]

title index result

phototherapy[*title*]
count=3
1, 8
10, 2
16, 5

author index result

phototherapy[*abstract*]
count=5
1, 3
10, 2
16, 5
33, 2
56, 10

final result

final
count=5
1, 11
10, 4
16, 10
33, 2
56, 10

Field-Restricted Boolean Retrieval

- Solution 2: build a single index, but treat each term-field pair as a separate entry in the index

a.title	*a.author*	*a.body*	*abba.title*	*abba.author*	*abba.body*	...
df=33	*df=2*	*df=543*	*df=3*	*df=1*	*df=3*	...
33, 3	3033, 1	1, 8	1, 1	543, 1	43, 3	...
45, 4	5463, 1	21, 78	341, 1	432, 1	...	
::		::	453, 1	2341, 1	...	
532, 54		567, 4			...	

Field-Restricted Boolean Retrieval

- Solution 2: map each query-term to the appropriate field-specific version
- (*phototherapy*)[*title*] OR (*light* AND *therapy*)[*abstract*] =
- *phototherapy.title* OR (*light.abstract* AND *therapy.abstract*)

a.title	*a.author*	*a.body*	*abba.title*	*abba.author*	*abba.body*	...
df=33	*df=2*	*df=543*	*df=3*	*df=1*	*df=3*	...
33, 3	3033, 1	1, 8	1, 1	543, 1	43, 3	...
45, 4	5463, 1	21, 78	341, 1		432, 1	...
::		::	453, 1		2341, 1	...
532, 54		567, 4				...

Wild-card Operators

Wildcard Operators

- Some query languages allow users to match a single query-term to multiple index-terms
- light AND therap*
- Assumption: the terms **therapy, therapies, therapeutic, etc.** are all equally predictive of relevance
- Do we need to modify the index to allow wildcard operators?

Wildcard Operators

- During index construction, build a dictionary that maps prefixes to full terms
- **therap* : [therapy, therapies, therapeutic, ...]**
- During query-processing, look up each prefix in the dictionary and expand the query using its full terms
- **light AND therap***
- **light AND (therapy XX therapies XX therapeutic XX ...)**
- What boolean operator should XX be?

Wildcard Operators

- You could also imagine doing this with suffixes
- *eutic : [hermeneutic, pharmaceutic, therapeutic, ...]
- light AND *eutic
- light AND (hermeneutic OR pharmaceutic OR therapeutic)

Wildcard Operators

- How might we handle within-term wildcards (e.g., **therap*c**)?

Wildcard Operators

- How might we handle within-term wildcards (e.g., **therap*c**)?
- Fetch list of full-terms for prefix **therap***
- Fetch list of full-terms for suffix ***c**
- Take the intersection

Take Home Message!