Introduction to Ad-hoc Retrieval

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

January 29, 2018

Ad-hoc Retrieval

- Text-based retrieval
- Given a query and a corpus, find the relevant items
  - query: textual description of information need
  - corpus: a collection of textual documents
  - relevance: satisfaction of the user's information need
- "Ad-hoc" because the number of possible queries is (in theory) infinite.

Google

Examples

web search

evo screen capture

[How to **screen capture** on **evo**? - PPCGeeks](#)
forum.ppcgeeks.com › ... › Android HTC Devices › HTC Evo 4G - Cached
Jul 6, 2010 – Is there any app for that ? Sent from my PC36100 using Tapatalk.

- Is it possible to **screen capture** before rooting? - Jul 8, 2011
- Print Screen / **Screen capture** - Sep 12, 2010
- Print Screen / **Screen capture** - Page 2 - Jun 21, 2010

More results from forum.ppcgeeks.com »

How to take screenshots on the HTC **EVO** 4G - Know Your Cell
www.knowyourcell.com/...**evo**.../**evo**.../how_to_take_**screen**shots_o... - Cached
Apr 15, 2010 – On the HTC **EVO** 4G: HTC Desire screen shot. Press the Home icon, ... Click on the Device menu and select **Screen Capture** or use the CTRL-S key ...

HTC **Evo** 4G Apps
www.evo4gforum.net › HTC Evo Media and Miscellaneous - Cached
HTC **Evo** 4G Apps - Talk about HTC **Evo** 4G Apps here. ... Advanced search · Scratch-Proof your HTC **Evo** 4G · Best Screen Protector for HTC **Evo** 4G · Good Price on HTC **Evo** 4G ... **Screen Capture** (updated 9/27/10) « 1 2 . ...

Android Screenshots: No Root Required with **EVO** › AndroidGuys
www.androidguys.com/2010/05/.../android-**screen**shots-root-require... - Cached
May 24, 2010 – We tested this on a stock HTC **EVO** 4G distributed at Google I/O. Let us know in the comments if other **screen capture** apps work on your ...

How to take screenshots on the HTC **EVO** 4G
www.goodand**evo**.net/.../how-to-take-**screen**shots-on-the-htc-**evo**-4... - Cached
May 24, 2010 – **Evo**-ss In general, there are two ways to take screenshots on an Android phone: 1) root it and install a **screen capture** app and 2) connect to ...

Screen Capture/Print Screen App for **EVO** 2.2 - Android Forums
androidforums.com › ... › HTC EVO 4G › EVO 4G - Tips and Tricks - Cached
3 posts - 3 authors - Last post: Aug 11, 2010
I've read several post on **screen capture**, most of which seem to be for advanced users and also risk bricking your phone. Is there a screen ...

Examples

scientific search

PubMed | high fructose corn syrup and obesity | Search

1. Metabolic and behavioural effects of sucrose and **fructose**/glucose drinks in the rat.
   Sheludiakova A, Rooney K, Boakes RA.
   Eur J Nutr. 2011 Jul 29. [Epub ahead of print]
   PMID: 21800086 [PubMed - as supplied by publisher]
   Related citations

2. The impact of **fructose** on renal function and blood pressure.
   Kretowicz M, Johnson RJ, Ishimoto T, Nakagawa T, Manitius J.
   Int J Nephrol. 2011;2011:315879. Epub 2011 Jul 17.
   PMID: 21792388 [PubMed - in process] **Free PMC Article**
   Free full text Related citations

3. The role of salt in the pathogenesis of **fructose**-induced hypertension.
   Soleimani M, Alborzi P.
   Int J Nephrol. 2011;2011:392708. Epub 2011 Jul 18.
   PMID: 21789281 [PubMed - in process] **Free PMC Article**
   Free full text Related citations

4. Survey of American food trends and the growing **obesity** epidemic.
   Shao Q, Chin KV.
   Nutr Res Pract. 2011 Jun;5(3):253-9. Epub 2011 Jun 21.
   PMID: 21779530 [PubMed - in process] **Free PMC Article**
   Free full text Related citations

5. **Obesity** and energy balance: is the tail wagging the dog?
   Wells JC, Siervo M.
   Eur J Clin Nutr. 2011 Jul 20. doi: 10.1038/ejcn.2011.132. [Epub ahead of print]
   PMID: 21772313 [PubMed - as supplied by publisher]
   Related citations

MacRumors
news and rumors you care about

Examples
discussion forum search

thunderbird installation

Search: Keyword(s): **thunderbird**, **installation**

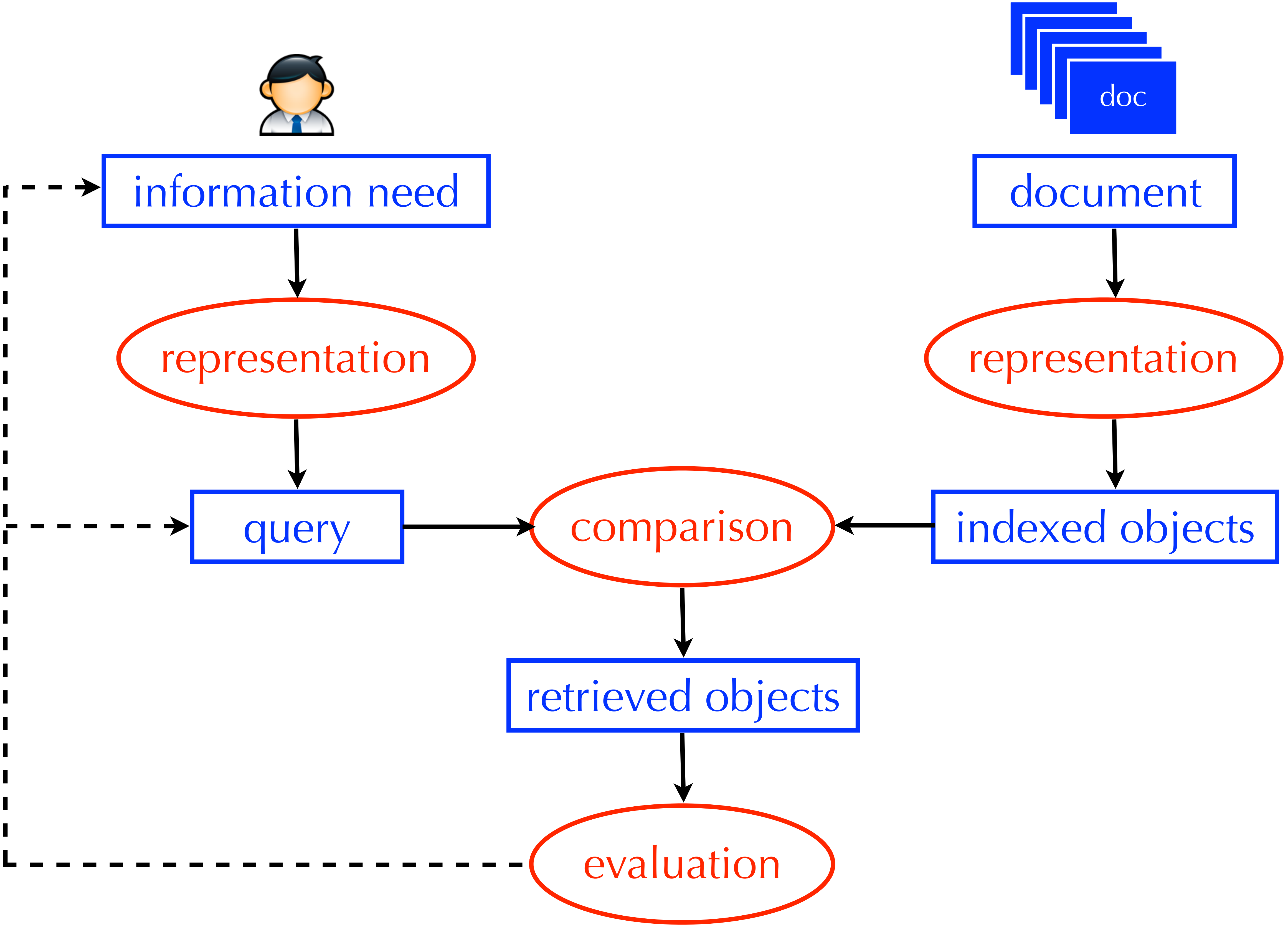
Showing results 1 to 25 of 38
Search took **0.02** seconds.

Thread / Thread Starter	Last Post	Replies	Views	Forum
Pre-Installed Mac Applications (1 2) ' r i S e n	Jul 18, 2011 02:21 AM by RasmusM	34	1,953	Mac Applications and Mac App Store
Translucent mail notify BLOND37	Jun 12, 2011 11:45 PM by jive turkey	8	277	Mac Applications and Mac App Store
How do I move Thunderbird e-mail from PC to Mac donhnick	Oct 12, 2010 08:41 AM by tommcdonald	7	35,011	Mac Applications and Mac App Store
Re-installing 10.6 while preserving user data? Bunker	Feb 28, 2010 10:45 AM by TonyK	5	708	Mac OS X
New to MAC - Dissappointed - text size (1 2 3 4 5 6 ... Last Page) MariekeFJ	Jan 19, 2010 12:14 PM by Don Crosswhite	157	10,115	Mac Basics and Help
Anyone have to "switch back" due to $$? (1 2 3) Schtibbie	Oct 20, 2009 09:30 PM by Kat King123	52	2,688	MacBook
The Saga of Switching ready2switch	May 21, 2009 04:24 PM by Chris.L	4	493	Mac Basics and Help
Apple Mail vs Entourage DJAKO	May 8, 2009 06:30 PM by Benguitar	20	16,768	Mac Applications and Mac App Store
Teacher accuses student using linux of copyright infringement! (1 2 3) LeoFio	Dec 15, 2008 10:14 AM by dilbert4life	56	1,763	Community Discussion
Timemachine Duplicates? MBX	Nov 27, 2008 09:16 AM by scuac	18	2,206	Mac OS X

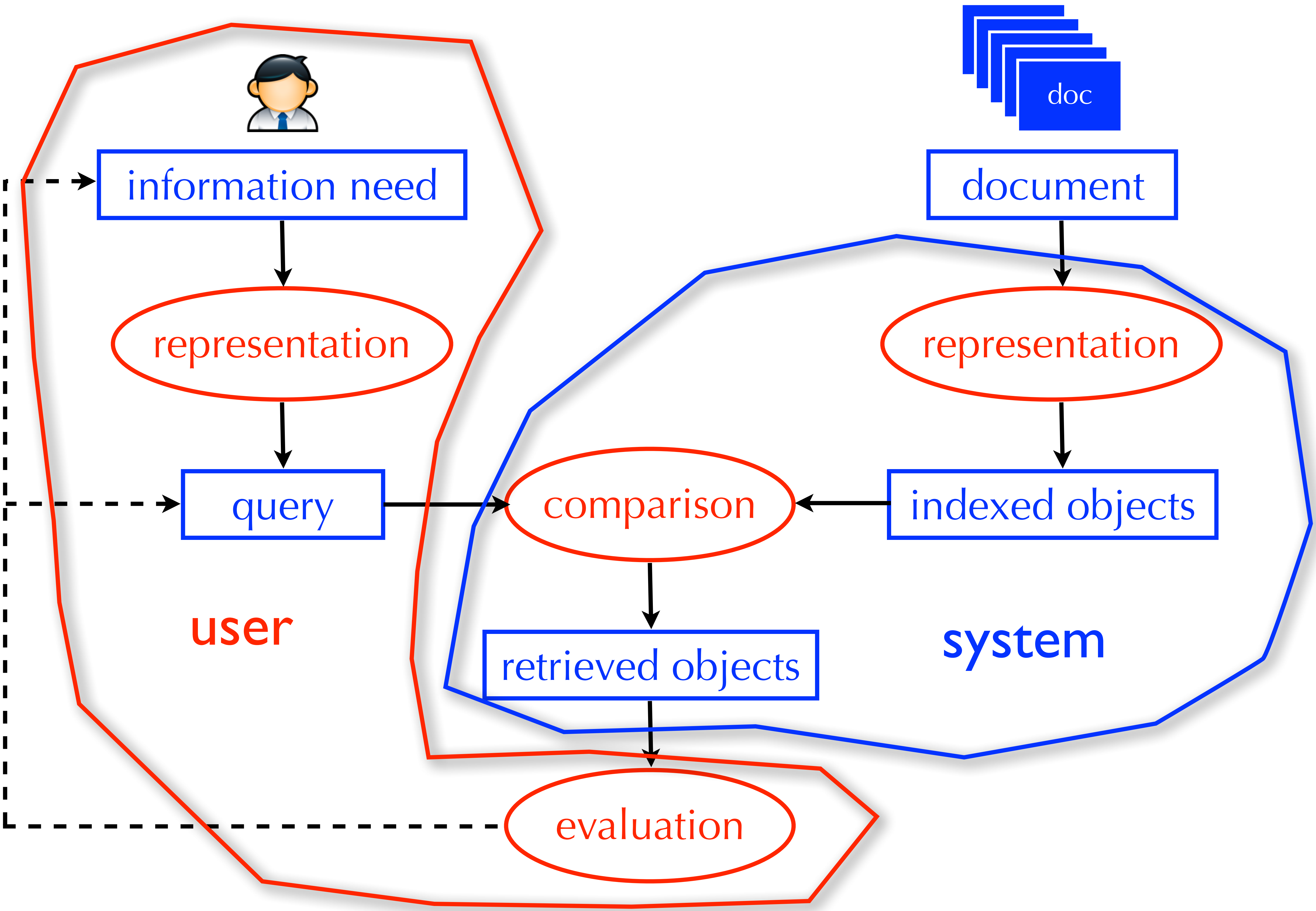
Ad-hoc Retrieval

- We will focus on non-web ad-hoc retrieval
  - more is known about how these systems work
  - more stable solutions - not constantly tweaked
  - not heavily tuned using user-interaction data (e.g., clicks)
  - very common: digital libraries, government and corporate intranets, large information service providers (e.g., Thompson Reuters), social media, your own personal computers

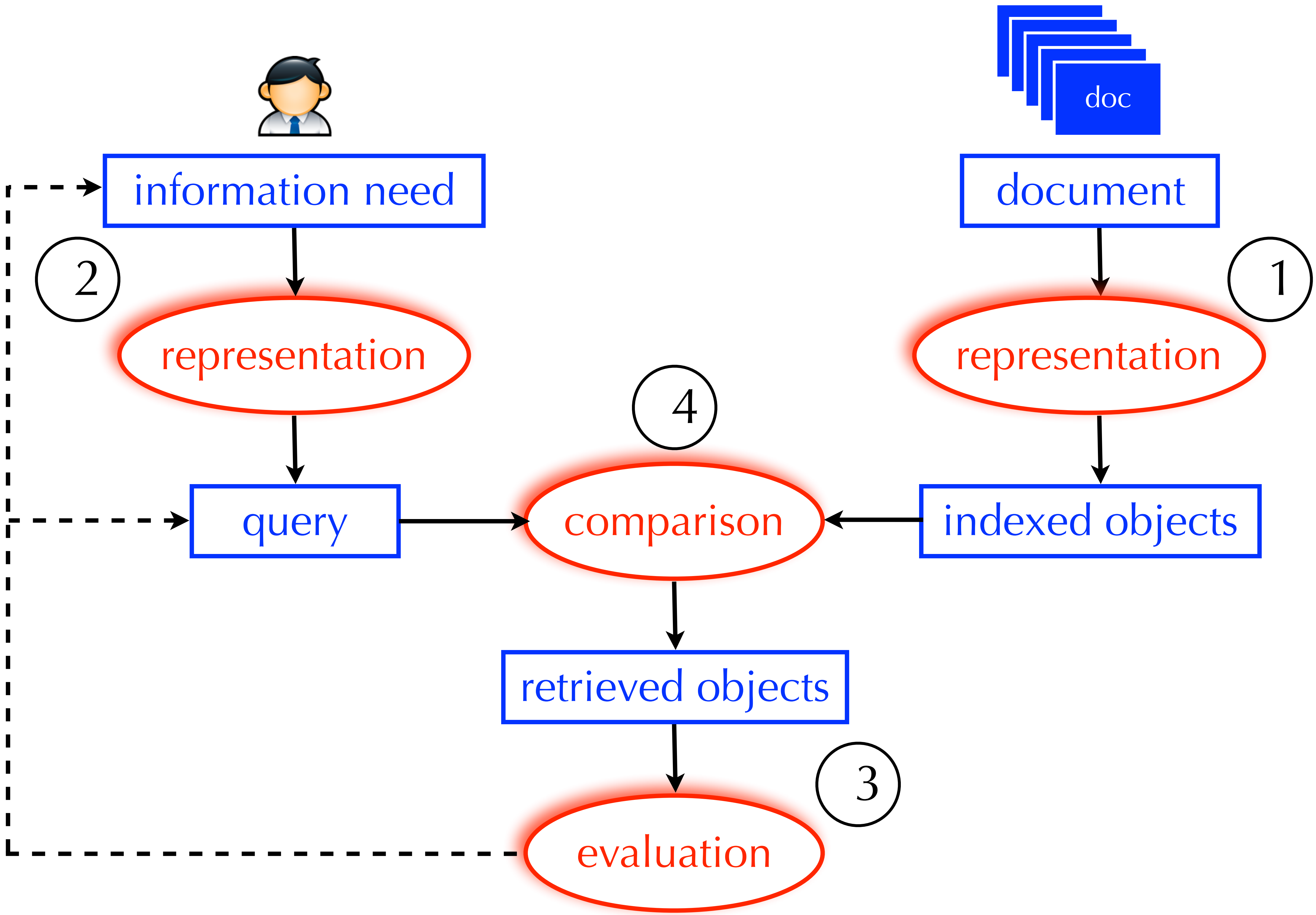
Basic Information Retrieval Process



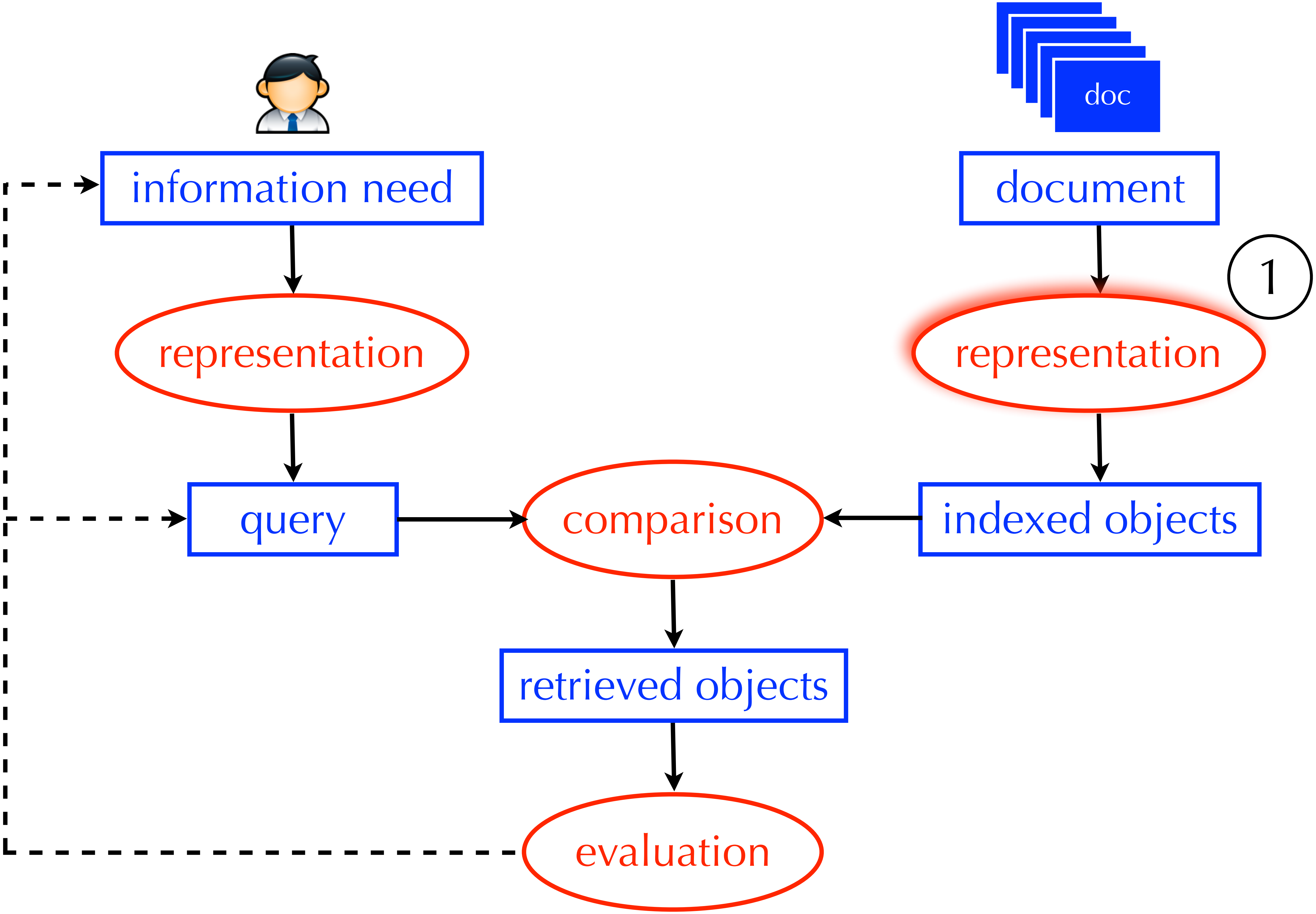
Basic Information Retrieval Process



Next Two Lectures



Document Representation



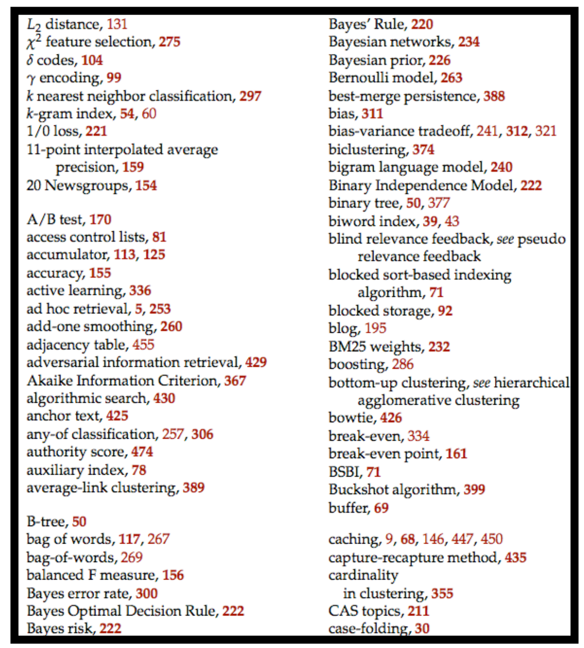
Most Basic View of a Search Engine

- A search engines <u>does not</u> scan each document to see if it satisfies the query
- It uses an index to quickly locate the relevant documents
- Index: a list of concepts with pointers to documents (in this case, pages) that discuss them

Index from Manning et al., 2008

Most Basic View of a Search Engine

input query: A/B testing



output document: docid: 170

- So, what goes in the index is important!
- How might we combine concepts (e.g., patent search + A/B testing)?

Document Representation

- Document representation: deciding what concepts should go in the index
- Option 1 (controlled vocabulary): a set a manually constructed concepts that describe the major topics covered in the collection
- Option 2 (free-text indexing): the set of individual terms that occur in the collection

Document Representation

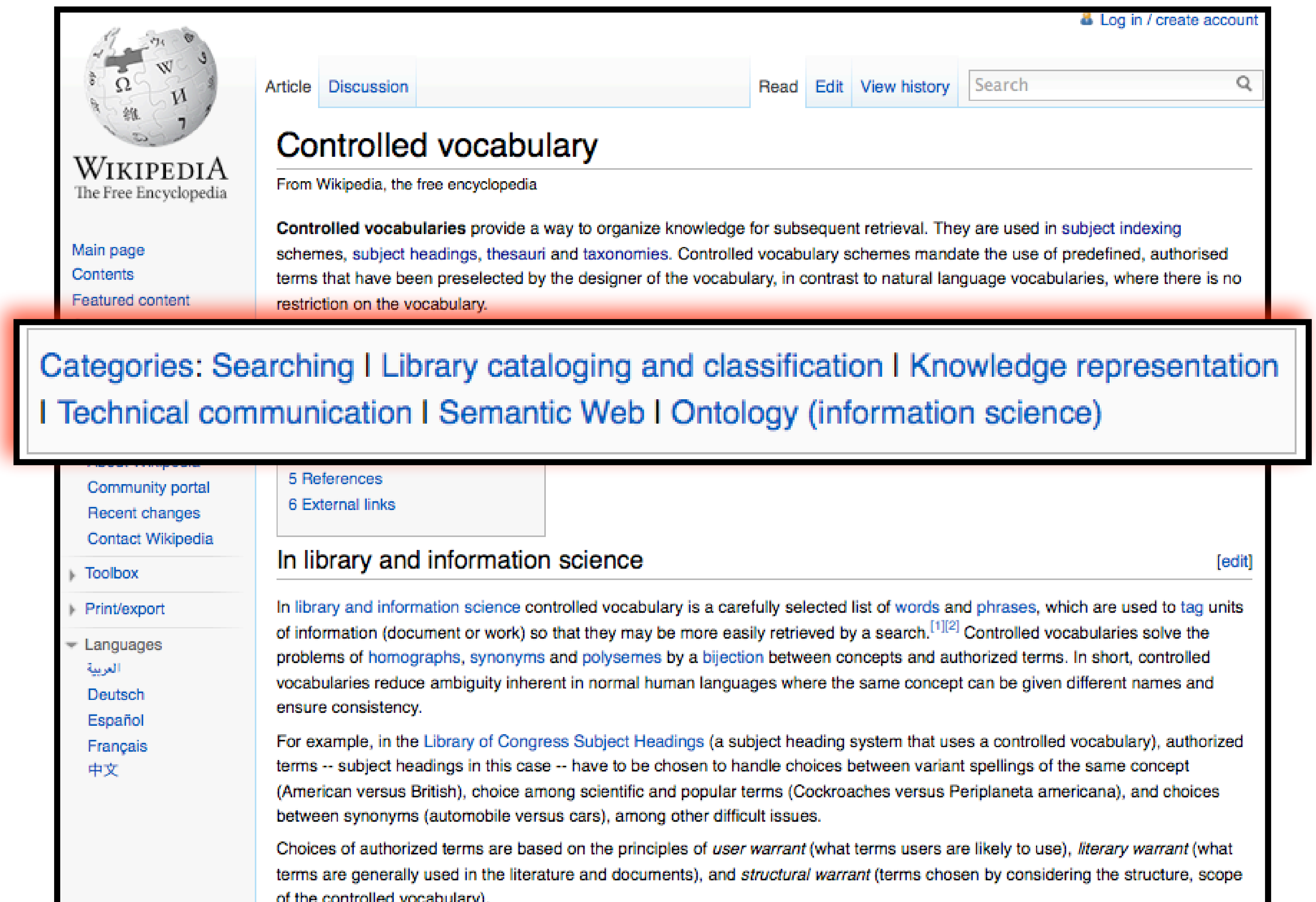
- If we view option 1 and option 2 as two extremes, where does this particular index fit in?

Index from Manning et al., 2008

Document Representation

option 1: controlled vocabulary

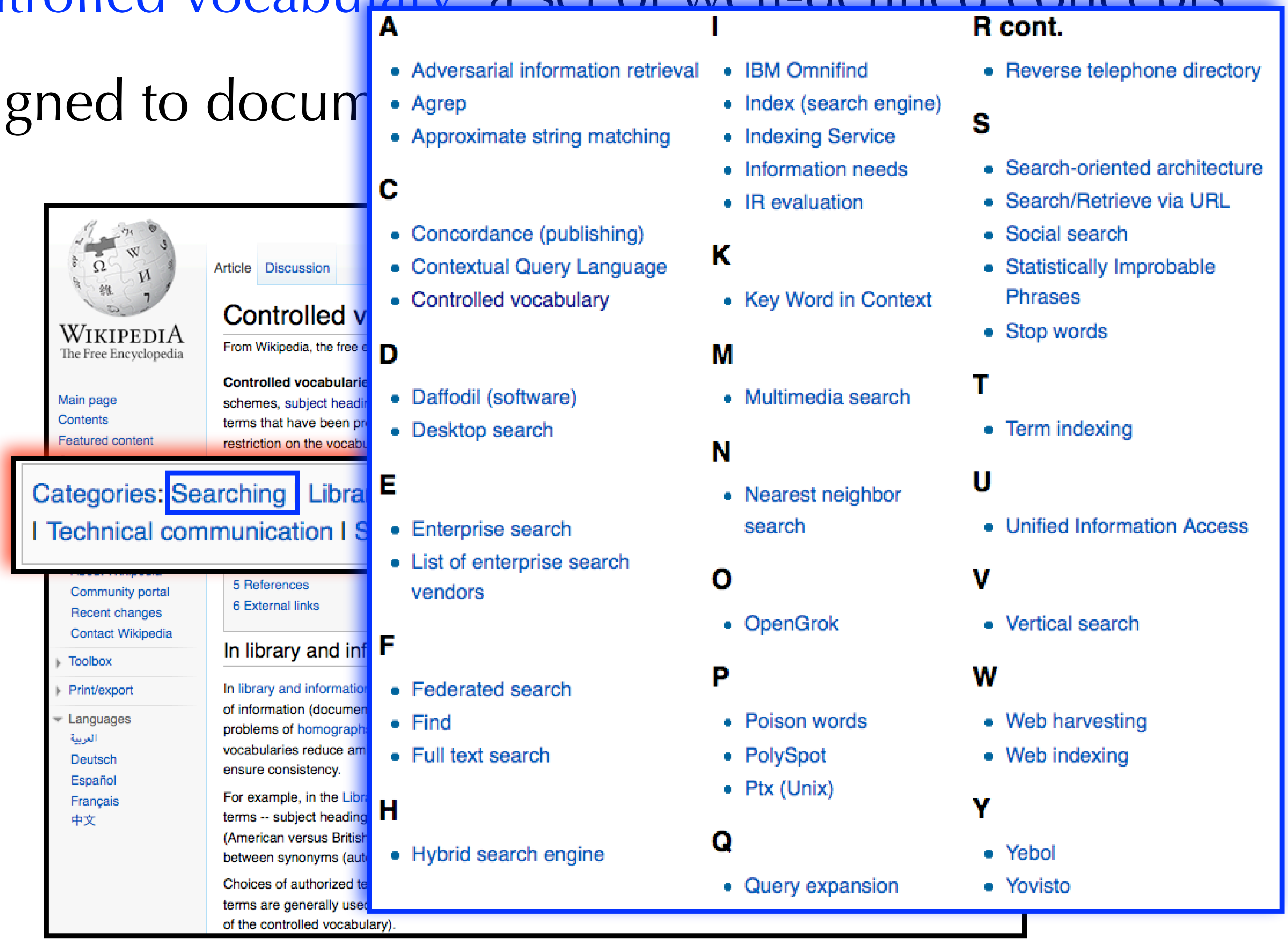
- Controlled vocabulary: a set of well-defined concepts
- Assigned to documents by humans (or automatically)



Document Representation

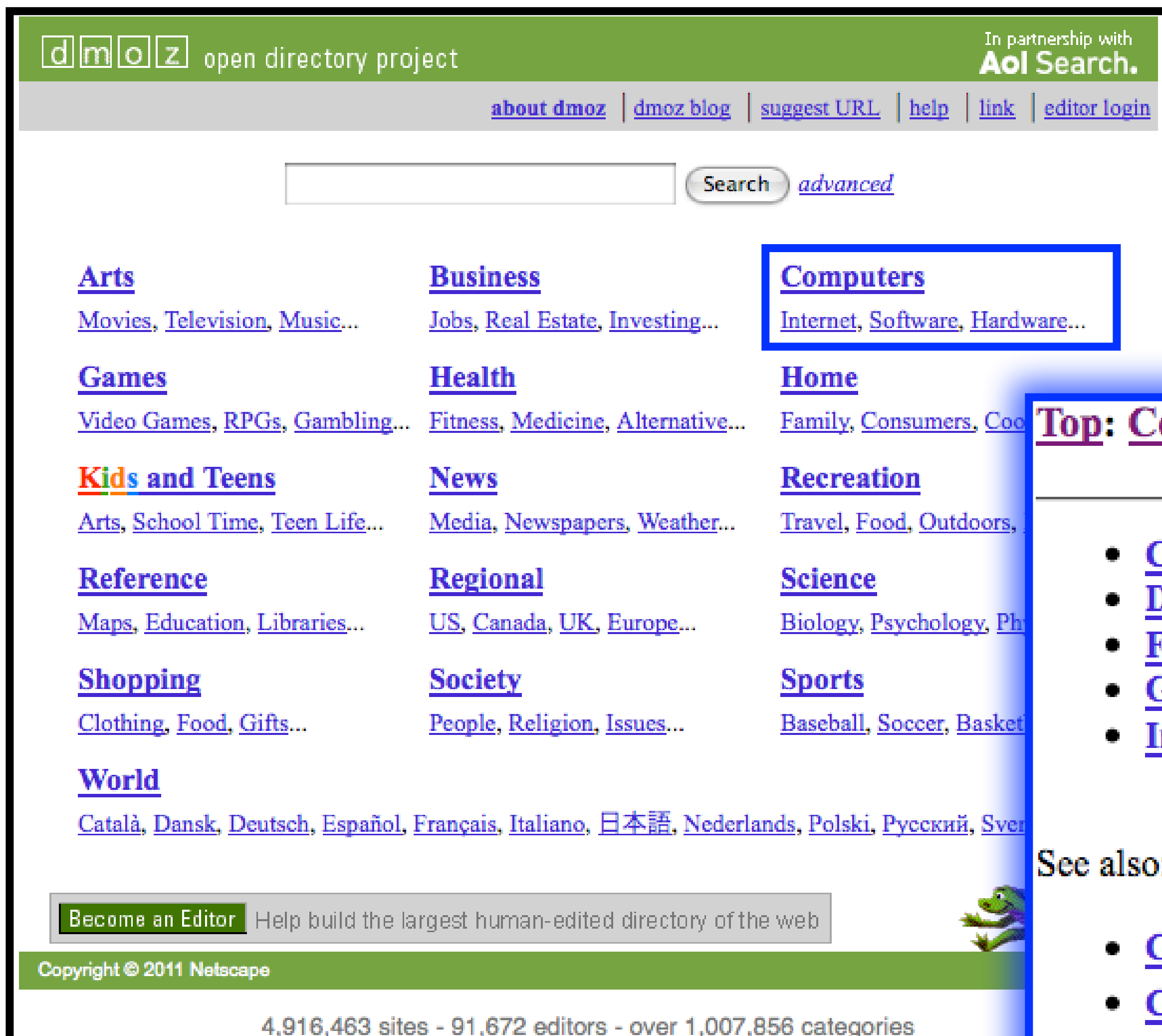
option 1: controlled vocabulary

- Controlled vocabulary: a set of well-defined concepts
- Assigned to docum

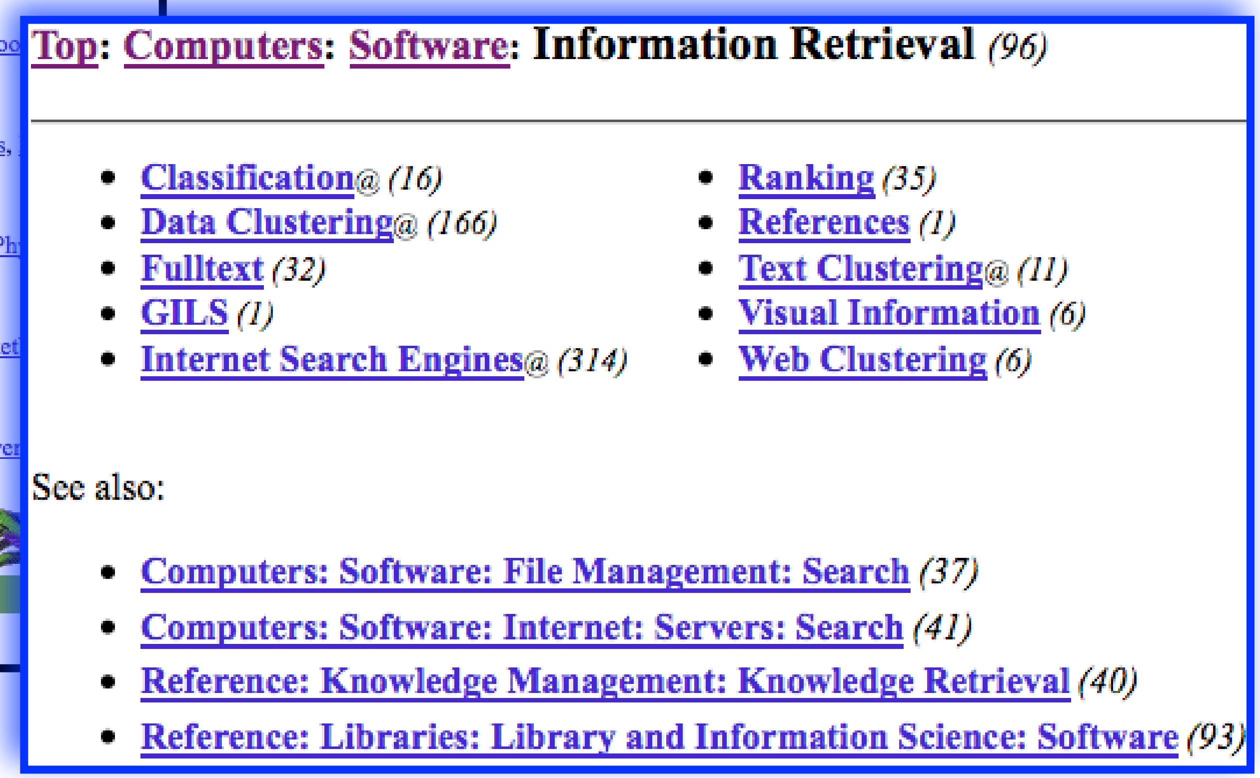


Controlled Vocabularies

- May include (parent-child) relations b/w concepts
- Facilitates non-query-based browsing and exploration



Open Directory Project (ODP)



Controlled Vocabularies
example

- MeSH: Medical Subject Headings
- Created by the National Library of Medicine to index biomedical journals and books
- About 25,000 subject headings arranged in a hierarchy
- Used to search PubMed

Controlled Vocabularies

example

1. ⊞ Anatomy [A]
2. ⊞ Organisms [B]
3. ⊞ Diseases [C]
4. ⊞ Chemicals and Drugs [D]
5. ⊞ Analytical, Diagnostic and Therapeutic Techniques and Equipment [E]
6. ⊞ Psychiatry and Psychology [F]
7. ⊞ Phenomena and Processes [G]
8. ⊞ Disciplines and Occupations [H]
9. ⊞ Anthropology, Education, Sociology and Social Phenomena [I]
10. ⊞ Technology, Industry, Agriculture [J]
11. ⊞ Humanities [K]
12. ⊞ Information Science [L]
13. ⊞ Named Groups [M]
14. ⊞ Health Care [N]
15. ⊞ Publication Characteristics [V]
16. ⊞ Geographicals [Z]

Controlled Vocabularies

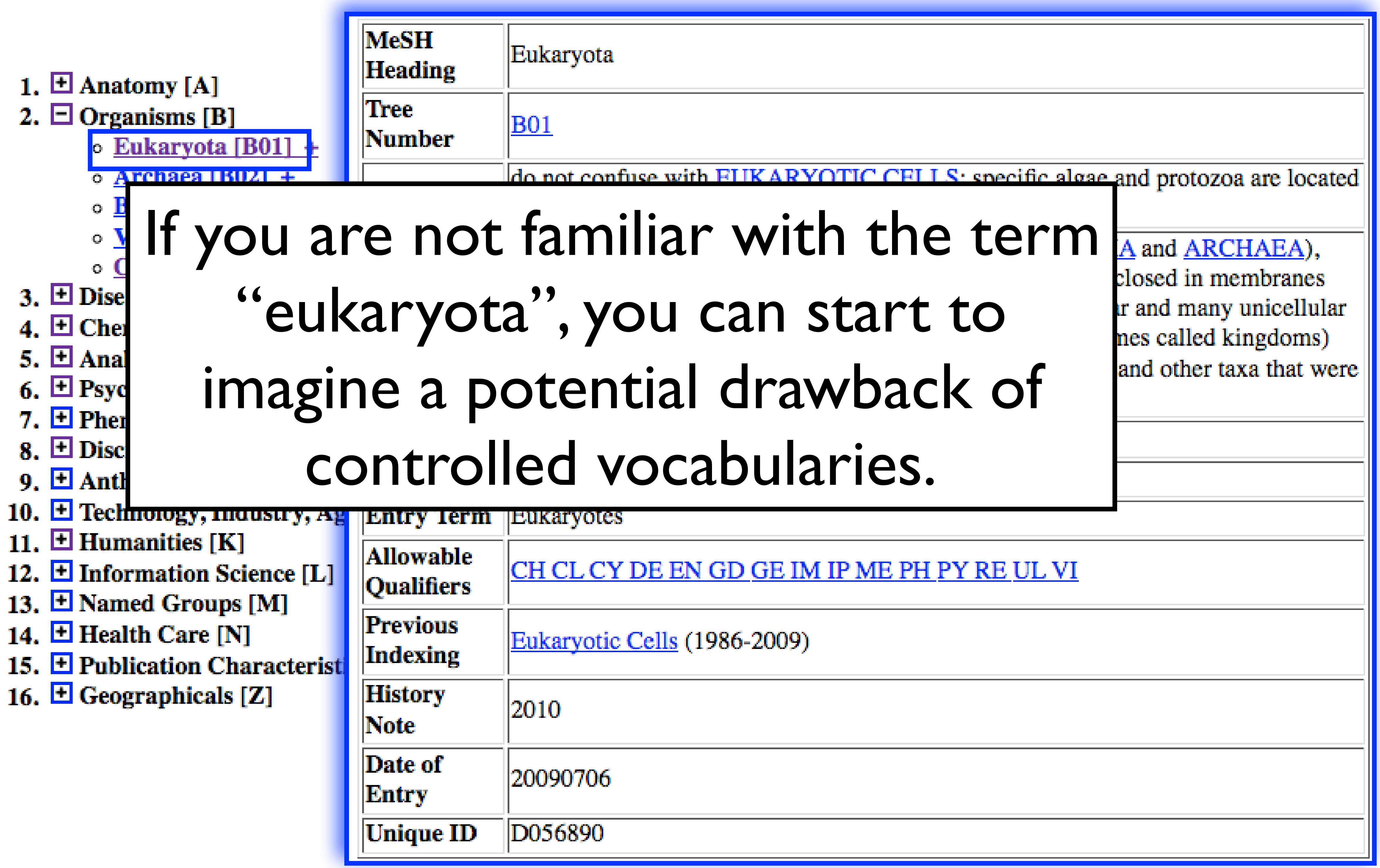
example

1. ⊞ Anatomy [A]
2. ⊟ Organisms [B]
   - Eukaryota [B01] +
   - Archaea [B02] +
   - Bacteria [B03] +
   - Viruses [B04] +
   - Organism Forms [B05
3. ⊞ Diseases [C]
4. ⊞ Chemicals and Drugs [D]
5. ⊞ Analytical, Diagnostic an
6. ⊞ Psychiatry and Psycholog
7. ⊞ Phenomena and Processe
8. ⊞ Disciplines and Occupati
9. ⊞ Anthropology, Education
10. ⊞ Technology, Industry, Ag
11. ⊞ Humanities [K]
12. ⊞ Information Science [L]
13. ⊞ Named Groups [M]
14. ⊞ Health Care [N]
15. ⊞ Publication Characterist
16. ⊞ Geographicals [Z]

MeSH Heading	Eukaryota
Tree Number	B01
Annotation	do not confuse with EUKARYOTIC CELLS; specific algae and protozoa are located under various groups treed under Eukaryota
Scope Note	One of the three domains of life (the others being BACTERIA and ARCHAEA), also called Eukarya. These are organisms whose cells are enclosed in membranes and possess a nucleus. They comprise almost all multicellular and many unicellular organisms, and are traditionally divided into groups (sometimes called kingdoms) including ANIMALS; PLANTS; FUNGI; and various algae and other taxa that were previously part of the old kingdom Protista.
Entry Term	Eucarya
Entry Term	Eukarya
Entry Term	Eukaryotes
Allowable Qualifiers	CH CL CY DE EN GD GE IM IP ME PH PY RE UL VI
Previous Indexing	Eukaryotic Cells (1986-2009)
History Note	2010
Date of Entry	20090706
Unique ID	D056890

Controlled Vocabularies

example



If you are not familiar with the term "eukaryota", you can start to imagine a potential drawback of controlled vocabularies.

Controlled Vocabularies

example

NCBI Resources How To

MeSH | MeSH | light therapy | Search

Phototherapy

Treatment of disease by exposure to light, especially by variously concentrated light rays or specific wavelengths.

Year introduced: 1981

PubMed search builder options

Subheadings:

sub-headings

adverse effects	instrumentation	statistics and numerical data
classification	methods	supply and distribution
contraindications	nursing	trends
economics	psychology	utilization
history	standards	veterinary

All MeSH Categories
- Analytical, Diagnostic and Therapeutic Techniques and Equipment Category
  - Therapeutics
    - **Phototherapy**
      - Color Therapy
      - Heliotherapy
      - Laser Therapy, Low-Level
      - Photochemotherapy
        - Hematoporphyrin Photoradiation
      - Ultraviolet Therapy
        - PUVA Therapy +

Entry Terms:

- Phototherapies
- Therapy, Photoradiation
- Photoradiation Therapies
- Therapies, Photoradiation
- **Light Therapy**
- Light Therapies
- Therapies, Light
- Therapy, Light
- Photoradiation Therapy

sub-tree within the hierarchy

entry-terms

Controlled Vocabularies

example

NCBI Resources How To

PubMed.gov
US National Library of Medicine
National Institutes of Health

PubMed | phototherapy/adverse effects | Search

RSS Save search Limits Advanced

Results: 1 to 20 of 2697

<< First < Prev Page 1 of 135 Next > Last >>

1. Burning daylight: balancing vitamin D requirements with sensible sun exposure.
   Stalgis-Bilinski KL, Boyages J, Salisbury EL, Dunstan CR, Henderson SI, Talbot PL.
   Med J Aust. 2011 Apr 4;194(7):345-8.
   PMID: 21470084 [PubMed - indexed for MEDLINE]
   Related citations

2. Time-lag between subretinal fluid and pigment epithelial detachment reduction after polypoidal choroidal vasculopathy treatment.
   Chae JB, Lee JY, Yang SJ, Kim JG, Yoon YH.
   Korean J Ophthalmol. 2011 Apr;25(2):98-104. Epub 2011 Mar 11.
   PMID: 21461221 [PubMed - indexed for MEDLINE] **Free PMC Article**
   Free full text Related citations

3. Metal stenting to resolve post-photodynamic therapy stricture in early esophageal cancer.
   Cheon YK.
   World J Gastroenterol. 2011 Mar 14;17(10):1379-82.
   PMID: 21455341 [PubMed - indexed for MEDLINE] **Free PMC Article**
   Free full text Related citations

4. A study of multiple full-face treatments with low-energy settings of a 2940-nm Er:YAG fractionated laser.
   Goldberg DJ, Hussain M.
   J Cosmet Laser Ther. 2011 Apr;13(2):42-6.
   PMID: 21401375 [PubMed - indexed for MEDLINE]
   Related citations

Controlled Vocabularies

example

Burning daylight: balancing vitamin D requirements with sensible sun exposure.

Stalgis-Bilinski KL, Boyages J, Salisbury EL, Dunstan CR, Henderson SI, Talbot PL.

Westmead Breast Cancer Institute, University of Sydney, Sydney, NSW, Australia. Kellie.Bilinski@bci.org.au

Abstract

OBJECTIVE: To examine the feasibility of balancing sunlight exposure to meet vitamin D requirements with sun protection guidelines.

DESIGN AND SETTING: We used standard erythemal dose and Ultraviolet Index (UVI) data for 1 June 1996 to 30 December 2005 for seven Australian cities to estimate duration of sun exposure required for fair-skinned individuals to synthesise 1000 IU (25 μg) of vitamin D, with 11% and 17% body exposure, for each season and hour of the day. Periods were classified according to whether the UVI was < 3 or ≥ 3 (when sun protection measures are recommended), and whether required duration of exposure was ≤ 30 min, 31-60 min, or > 60 min.

MAIN OUTCOME MEASURE: Duration of sunlight exposure required to achieve 1000 IU of vitamin D synthesis.

RESULTS: Duration of sunlight exposure required to synthesise 1000 IU of vitamin D varied by time of day, season and city. Although peak UVI periods are typically promoted as between 10 am and 3 pm, UVI was often ≥ 3 before 10 am or after 3 pm. When the UVI was < 3, there were few opportunities to synthesise 1000 IU of vitamin D within 30 min, with either 11% or 17% body exposure.

CONCLUSION: There is a delicate line between balancing the beneficial effects of sunlight exposure while avoiding its damaging effects. Physiological and geographical factors may reduce vitamin D synthesis, and supplementation may be necessary to achieve adequate vitamin D status for individuals at risk of deficiency.

MeSH Terms

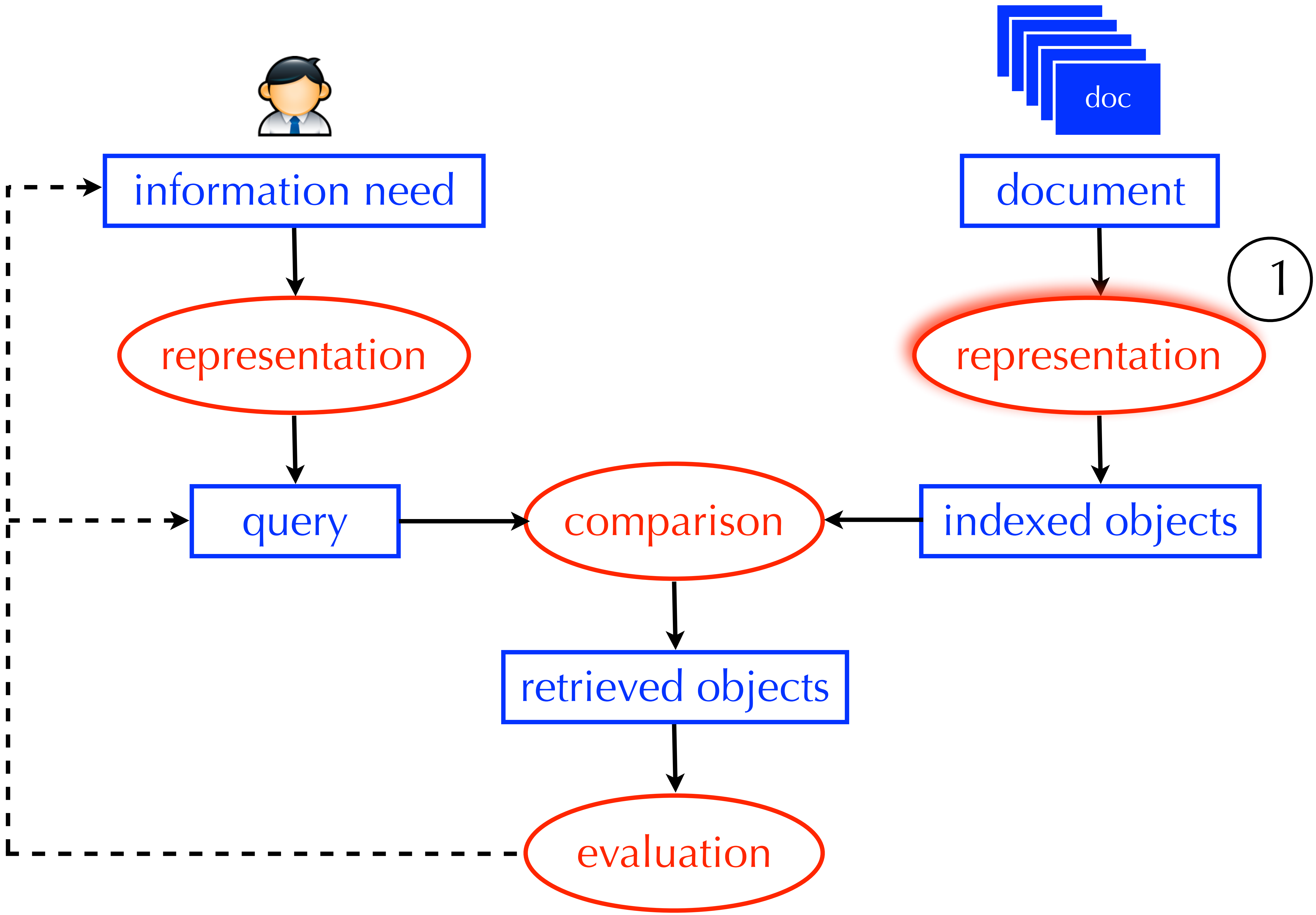
Australia
Dose-Response Relationship, Radiation
Guideline Adherence
Health Policy*
Heliotherapy/adverse effects
Heliotherapy/methods*
Humans
Seasons
Skin Pigmentation
Sunlight/adverse effects*
Time Factors
Vitamin D/biosynthesis*
Vitamin D Deficiency/prevention & control*

Controlled Vocabularies

advantages

- Concepts do not need to appear explicitly in the text
- Relationships between concepts facilitate non-query-based navigation and exploration (e.g., ODP)
- Developed by experts who know the data and the users
- Represent the concepts/relationships that users (presumably) care the most about
- Describe the concepts that are most central to the document
- Concepts are unambiguous and recognizable (necessary for annotators and good for users)

Document Representation



Document Representation

option 2: free-text indexing

- Represent documents using terms within the document
- Which terms? Only the most descriptive terms? Only the unambiguous ones? All of them?
- Usually, all of them (a.k.a. full-text indexing)
- The user will use term-combinations to express higher level concepts
- Query terms will hopefully disambiguate each other (e.g., “volkswagen golf”)
- The search engine will determine which terms are important (we’ll talk about this during “retrieval models”)

Free-text Indexing

Log in / create account

Wikipedia
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Toolbox

Print/export

Languages
Deutsch
Español
Bahasa Indonesia

Article | Discussion | Read | Edit | View history

Gerard Salton

From Wikipedia, the free encyclopedia

Gerard Salton (8 March 1927 in Nuremberg - 28 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

Salton was born Gerhard Anton Sahlmann on March 8, 1927 in Nuremberg, Germany. He received a Bachelor's (1950) and Master's (1952) degree in mathematics from Brooklyn College, and a Ph.D. from Harvard in Applied Mathematics in 1958, the last of Howard Aiken's doctoral students, and taught there until 1965, when he joined Cornell University and co-founded its department of Computer Science.

Salton was perhaps most well known for developing the now widely used Vector Space Model for Information Retrieval[1]. In this model, both documents and queries are represented as vectors of term counts, and the similarity between a document and a query is given by the cosine between the term vector and the document vector. In this paper, he also introduced TF-IDF, or term-frequency-inverse-document frequency, a model in which the score of a term in the a document is the ratio of the number of terms in that document divided by the frequency of the number of documents in which that term occurs. (The concept of inverse document frequency, a measure of specificity, had been introduced in 1972 by Karen Sparck-Jones[2].) Later in life, he became interested in automatic text summarization and analysis[3], as well as automatic hypertext generation[4]. He published over 150 research articles and 5 books during his life.

Salton was editor-in-chief of the Communications of the ACM and the Journal of the ACM, and chaired SIGIR. He was an associate editor of the ACM Transactions on Information Systems. He was an ACM Fellow (elected 1995), received an Award of Merit from the American Society for Information Science (1989), and was the first recipient of the SIGIR Award for outstanding contributions to study of information retrieval (1983) -- now called the Gerard Salton Award.

Free-text Indexing

what you see

Log in / create account

WIKIPEDIA
The Free Encyclopedia

- Main page
- Contents
- Featured content
- Current events
- Random article
- Donate to Wikipedia

Interaction

- Help
- About Wikipedia
- Community portal
- Recent changes
- Contact Wikipedia

Toolbox

Print/export

Languages

- Deutsch
- Español
- Bahasa Indonesia

Article | Discussion | Read | Edit | View history

Gerard Salton

From Wikipedia, the free encyclopedia

Gerard Salton (8 March 1927 in Nuremberg - 28 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

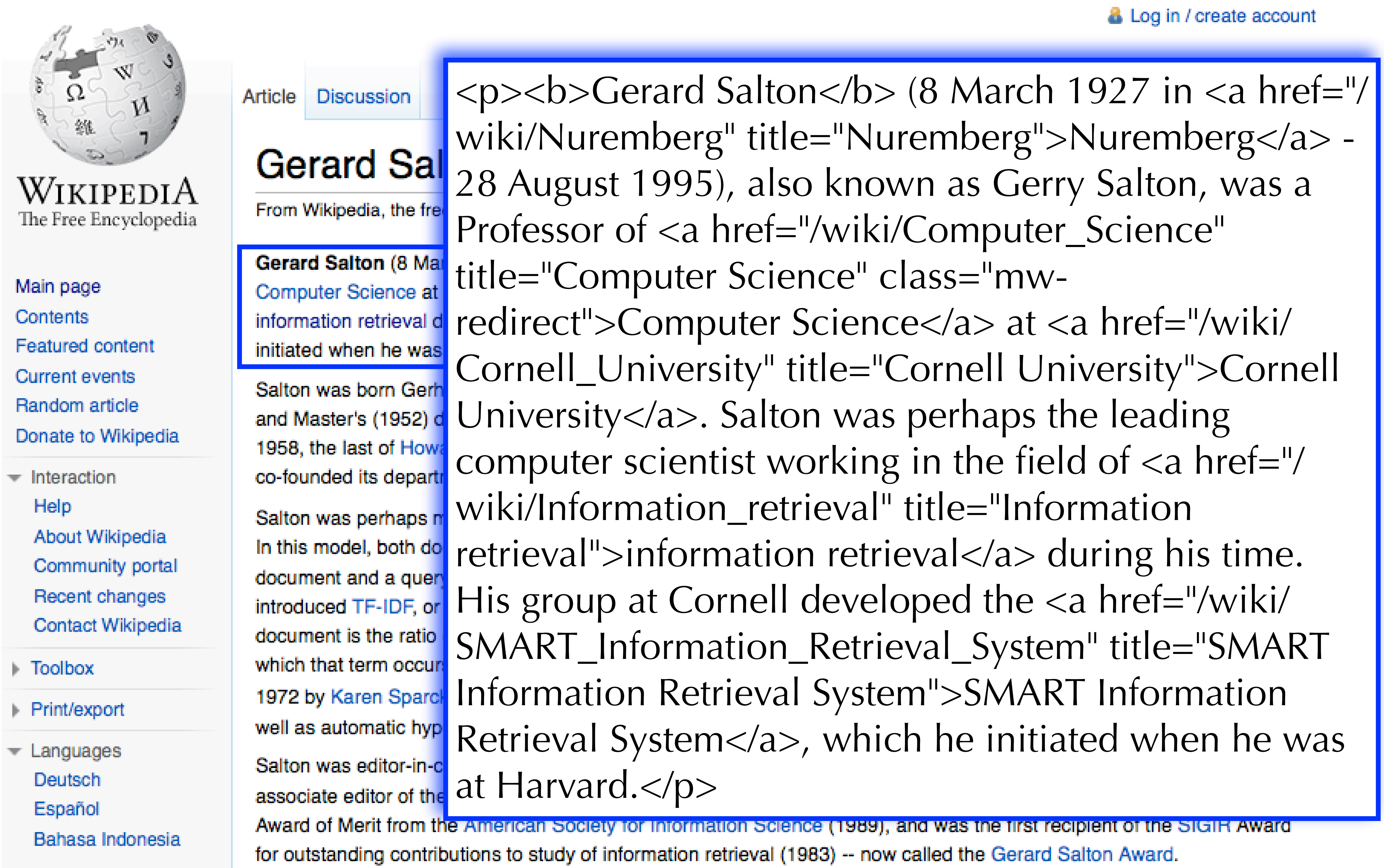
Salton was born Gerhard Anton Sahlmann on March 8, 1927 in Nuremberg, Germany. He received a Bachelor's (1950) and Master's (1952) degree in mathematics from Brooklyn College, and a Ph.D. from Harvard in Applied Mathematics in 1958, the last of Howard Aiken's doctoral students, and taught there until 1965, when he joined Cornell University and co-founded its department of Computer Science.

Salton was perhaps most well known for developing the now widely used Vector Space Model for Information Retrieval[1]. In this model, both documents and queries are represented as vectors of term counts, and the similarity between a document and a query is given by the cosine between the term vector and the document vector. In this paper, he also introduced TF-IDF, or term-frequency-inverse-document frequency, a model in which the score of a term in the a document is the ratio of the number of terms in that document divided by the frequency of the number of documents in which that term occurs. (The concept of inverse document frequency, a measure of specificity, had been introduced in 1972 by Karen Sparck-Jones[2].) Later in life, he became interested in automatic text summarization and analysis[3], as well as automatic hypertext generation[4]. He published over 150 research articles and 5 books during his life.

Salton was editor-in-chief of the Communications of the ACM and the Journal of the ACM, and chaired SIGIR. He was an associate editor of the ACM Transactions on Information Systems. He was an ACM Fellow (elected 1995), received an Award of Merit from the American Society for Information Science (1989), and was the first recipient of the SIGIR Award for outstanding contributions to study of information retrieval (1983) -- now called the Gerard Salton Award.

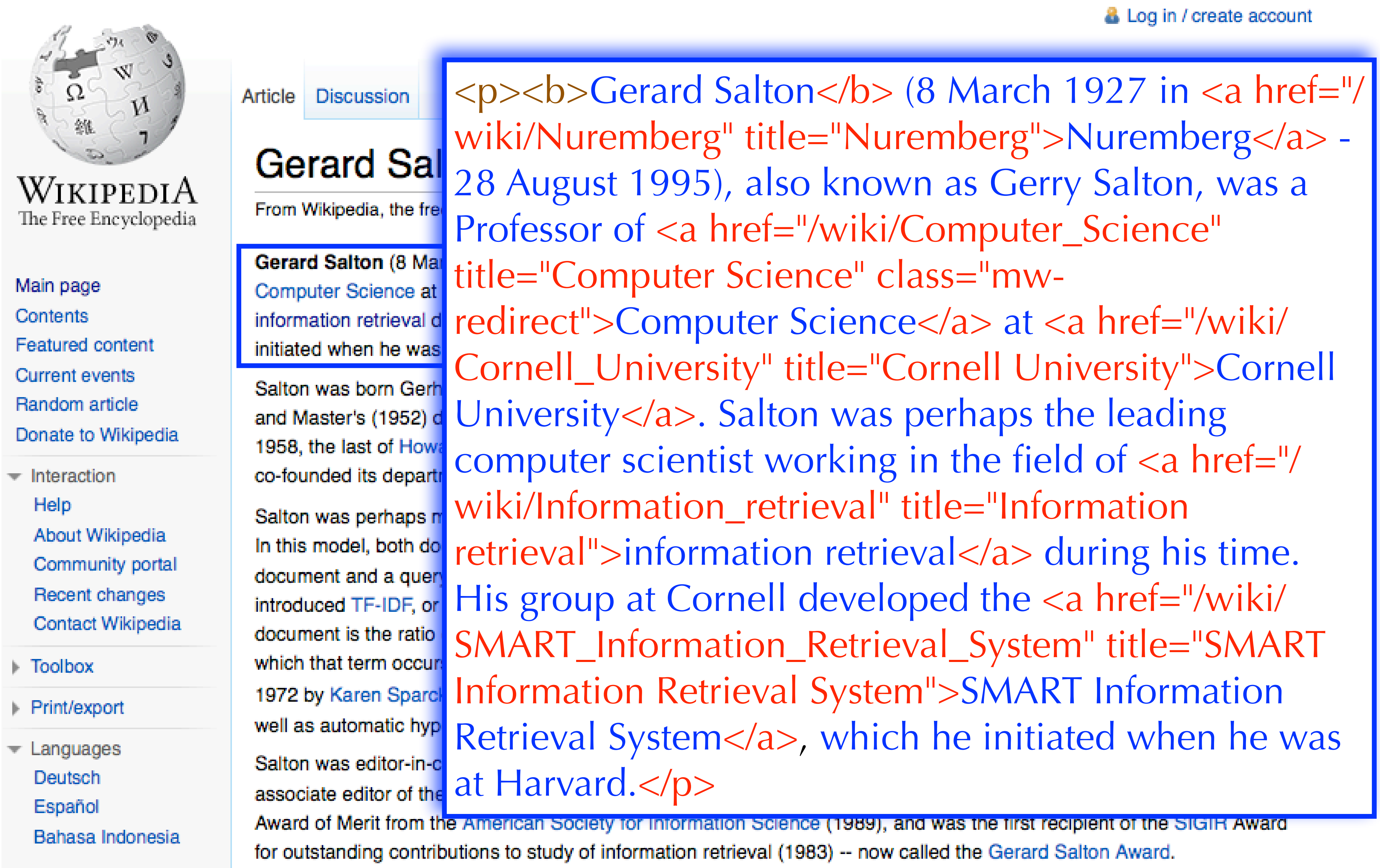
Free-text Indexing

what your computer sees



Free-text Indexing

mark-up vs. content



Free-text Indexing

mark-up

- Describes how the content should be presented
  - e.g., your browser interprets HTML mark-up and presents the page as intended by the author
- Can also define relationships with other documents (e.g., hyperlinks)
- Can provide evidence of what text is important for search
- It may also provide useful, “unseen” information!

Free-text Indexing

mark-up

Log in / create account

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact Wikipedia

Languages
Deutsch
Español
Bahasa Indonesia

Article | Discussion | Read | Edit | View history

Gerard Salton

From Wikipedia, the free encyclopedia

Gerard Salton (8 March 1927 in Nuremberg - 28 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

Salton was born Gerhard Anton Sahlmann on March 8, 1927 in Nuremberg, Germany. He received a Bachelor's (1950) and Master's (1952) degree in mathematics from Brooklyn College, and a Ph.D. from Harvard in Applied Mathematics in 1958, the last of Howard Aiken's doctoral students, and taught there until 1965, when he joined Cornell University and co-founded its department of Computer Science.

Salton was perhaps most well known for developing the now widely used Vector Space Model for Information Retrieval[1]. In this model, both documents and queries are represented as vectors of term counts, and the similarity between a document and a query is given by the cosine between the term vector and the document vector. In this paper, he also introduced TF-IDF, or term-frequency-inverse-document frequency, a model in which the score of a term in the a

ACM

Salton was editor-in-chief of the Communications of the ACM and the Journal of the ACM, and chaired SIGIR. He was an associate editor of the ACM Transactions on Information Systems. He was an ACM Fellow (elected 1995), received an Award of Merit from the American Society for Information Science (1989), and was the first recipient of the SIGIR Award for outstanding contributions to study of information retrieval (1983) -- now called the Gerard Salton Award.

Free-text Indexing

text-processing

<p>Gerard Salton (8 March 1927 in Nuremberg - 28 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.</p>

- Step 1: mark-up removal

Free-text Indexing
text-processing

Gerard Salton (8 March 1927 in Nuremberg 28 August 1995), also known as Gerry Salton, was a Professor of Computer Science at Cornell University. Salton was perhaps the leading computer scientist working in the field of information retrieval during his time. His group at Cornell developed the SMART Information Retrieval System, which he initiated when he was at Harvard.

- Step 1: mark-up removal

Free-text Indexing
text-processing

gerard salton (8 march 1927 in nuremberg 28 august 1995), also known as gerry salton, was a Professor of computer science at cornell university. salton was perhaps the leading computer scientist working in the field of information retrieval during his time. his group at cornell developed the smart information retrieval system, which he initiated when he was at harvard.

- Step 2: down-casing
- Can change a word's meaning, but we do it anyway
  - Information = information ???
  - SMART = smart ???

Free-text Indexing
text-processing

gerard salton 8 march 1978 in nuremberg 28 august 1995 also know as gerry salton was professor of computer science at cornell university salton was perhaps the leading computer scientist working in the field of information retrieval during his time his group at cornell developed the smart information retrieval system which he initiated when he was at harvard

- Step 3: tokenization
- Tokenization: splitting text into words (in this case, based on sequences of non-alphanumeric characters)
- Problematic cases: ph.d. = ph d, isn’t = isn t

Free-text Indexing
text-processing

gerard salton 8 march 1978 in nuremberg 28 august 1995 also know as gerry salton was professor of computer science at cornell university salton was perhaps the leading computer scientist working in the field of information retrieval during his time his group at cornell developed the smart information retrieval system which he initiated when he was at harvard

- Step 4: stopword removal
- Stopwords: words that we choose to ignore because we expect them to not be useful in distinguishing between relevant/non-relevant documents for any query

Free-text Indexing

text-processing

gerard salton 8 march 1978 nuremberg 28 august 1995 know gerry salton professor computer science cornell university salton perhaps leading computer scientist working field information retrieval time group cornell developed smart information retrieval system initiated harvard

- Step 4: stopword removal
- Stopwords: words that we choose to ignore because we expect them to not be useful in distinguishing between relevant/non-relevant documents for <u>any</u> query

Free-text Indexing
text-processing

gerard salton 8 march 1978 nuremberg 28 august 1995 gerry salton professor computer science cornell university salton leading computer scientist working field information retrieval during time group cornell developed smart information retrieval system initiated harvard

- Step 5: do this to every document in the collection and create an index using the all terms appearing in the collection

Document Representation

controlled vocabulary vs. free-text indexing

	Cost of assigning index terms	Ambiguity of index terms	Detail of representation
Controlled Vocabularies	High/Low?	Ambiguous/ Unambiguous?	Can represent arbitrary level of detail?
Free-text Indexing	High/Low?	Ambiguous/ Unambiguous?	Can represent arbitrary level of detail?

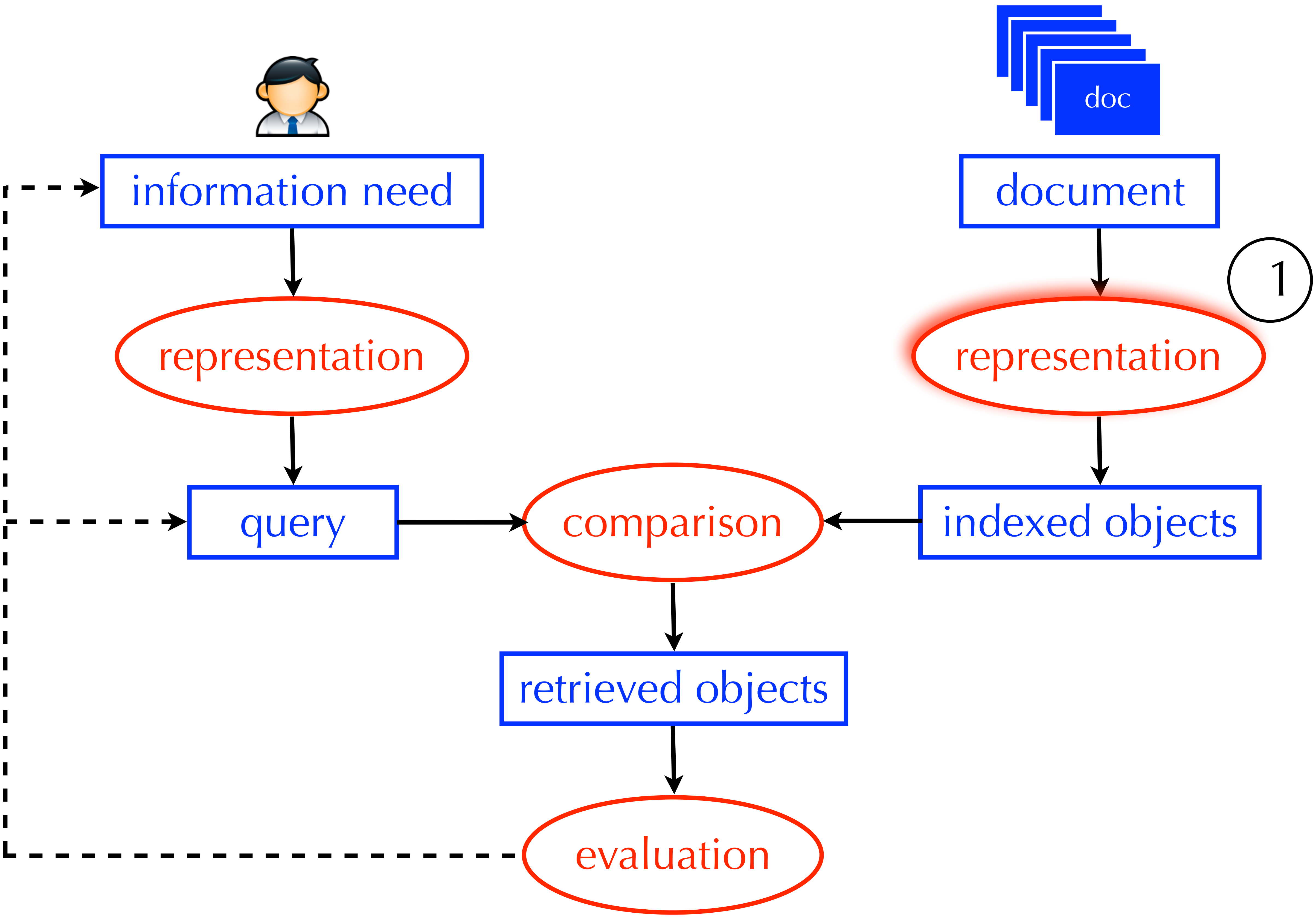
Document Representation

controlled vocabulary vs. free-text indexing

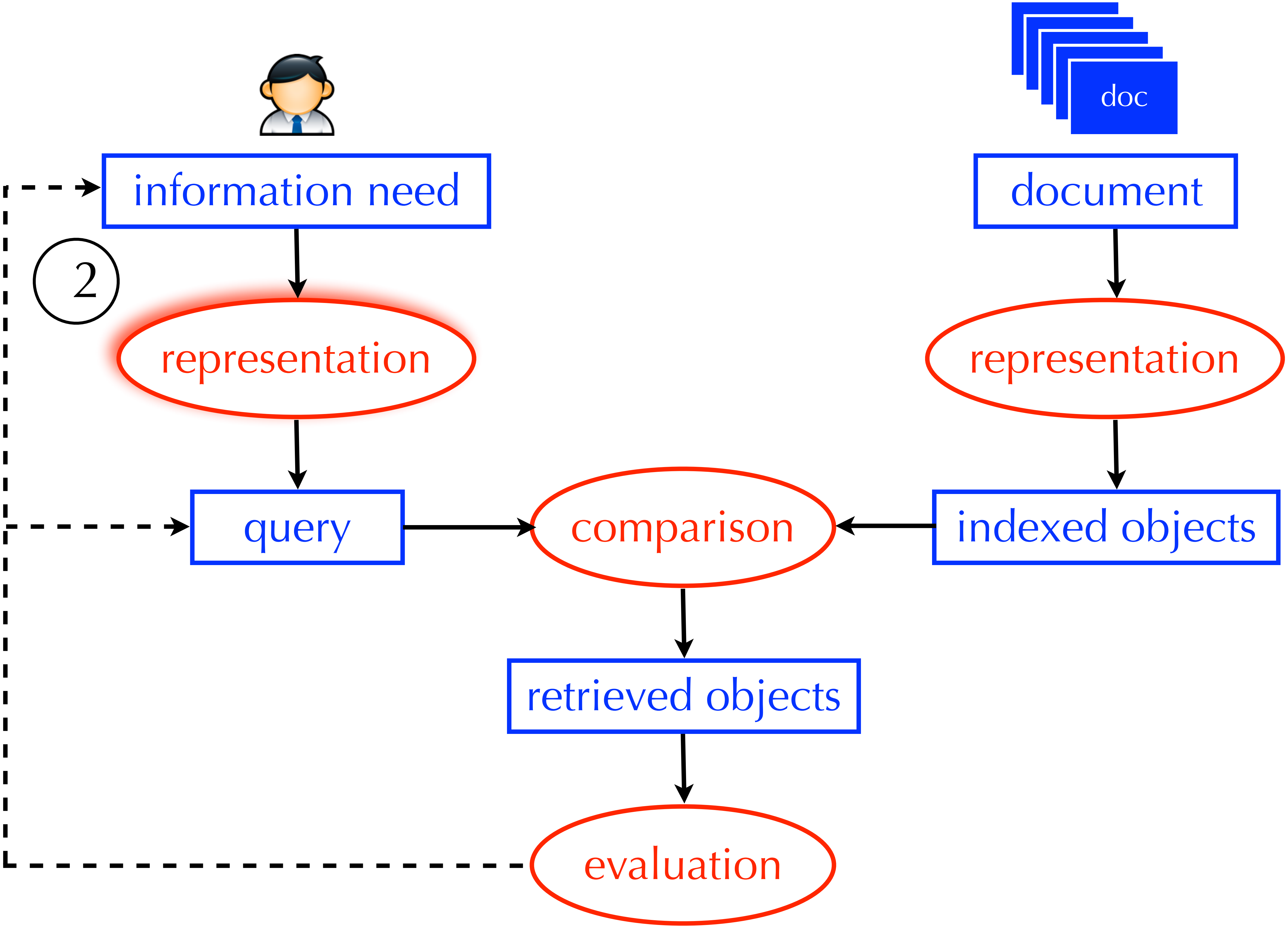
	Cost of assigning index terms	Ambiguity of index terms	Detail of representation
Controlled Vocabularies	High	Not ambiguous	Can't represent arbitrary detail
Free-text Indexing	Low	Can be ambiguous	Any level of detail

- Both are effective and used often
- We will focus on free-text indexing in this course
  - cheap and easy
  - most search engines use it (even those that adopt a controlled vocabulary)

Document Representation



Information Need Representation

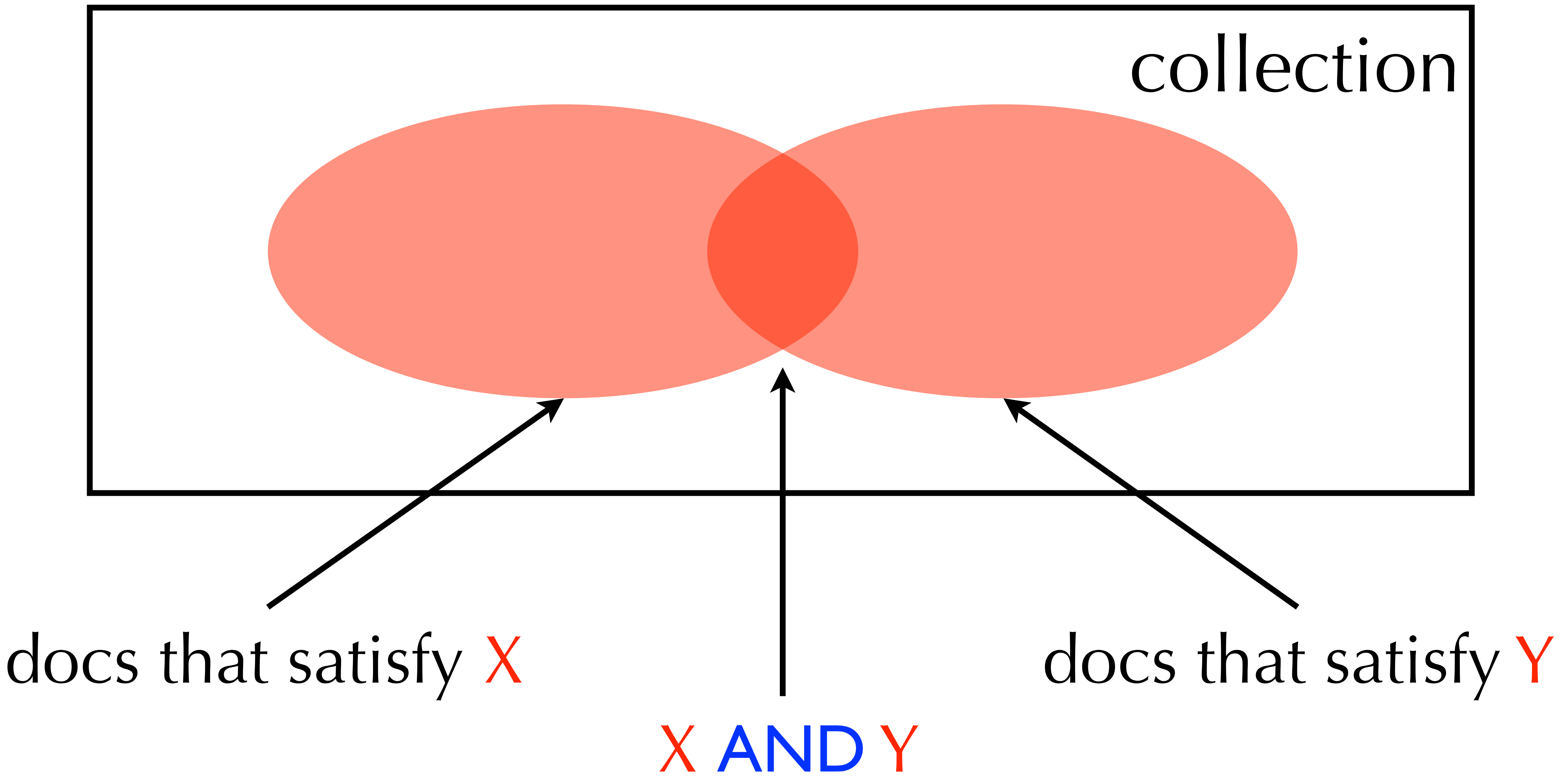


Boolean Retrieval

- Assumption: the user can represent their information need using boolean constraints: AND, OR, and AND NOT
  - lincoln
  - president AND lincoln
  - president AND (lincoln OR abraham)
  - president AND (lincoln OR abraham) AND NOT car
  - president AND (lincoln OR abraham) AND NOT (car OR automobile)
- Parentheses specify the order of operations
  - A OR (B AND C) does not equal (A OR B) AND C

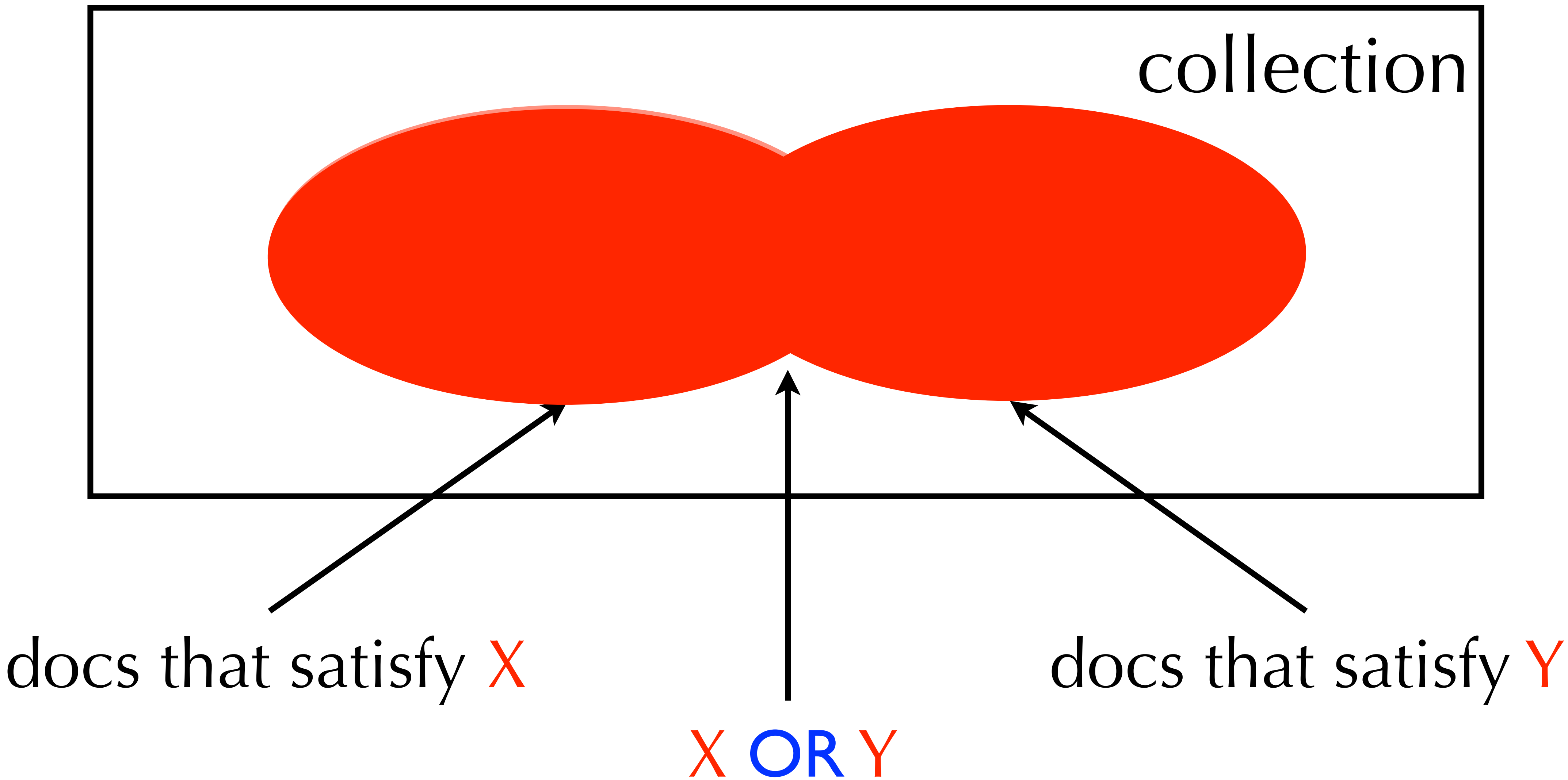
Boolean Retrieval

- X AND Y



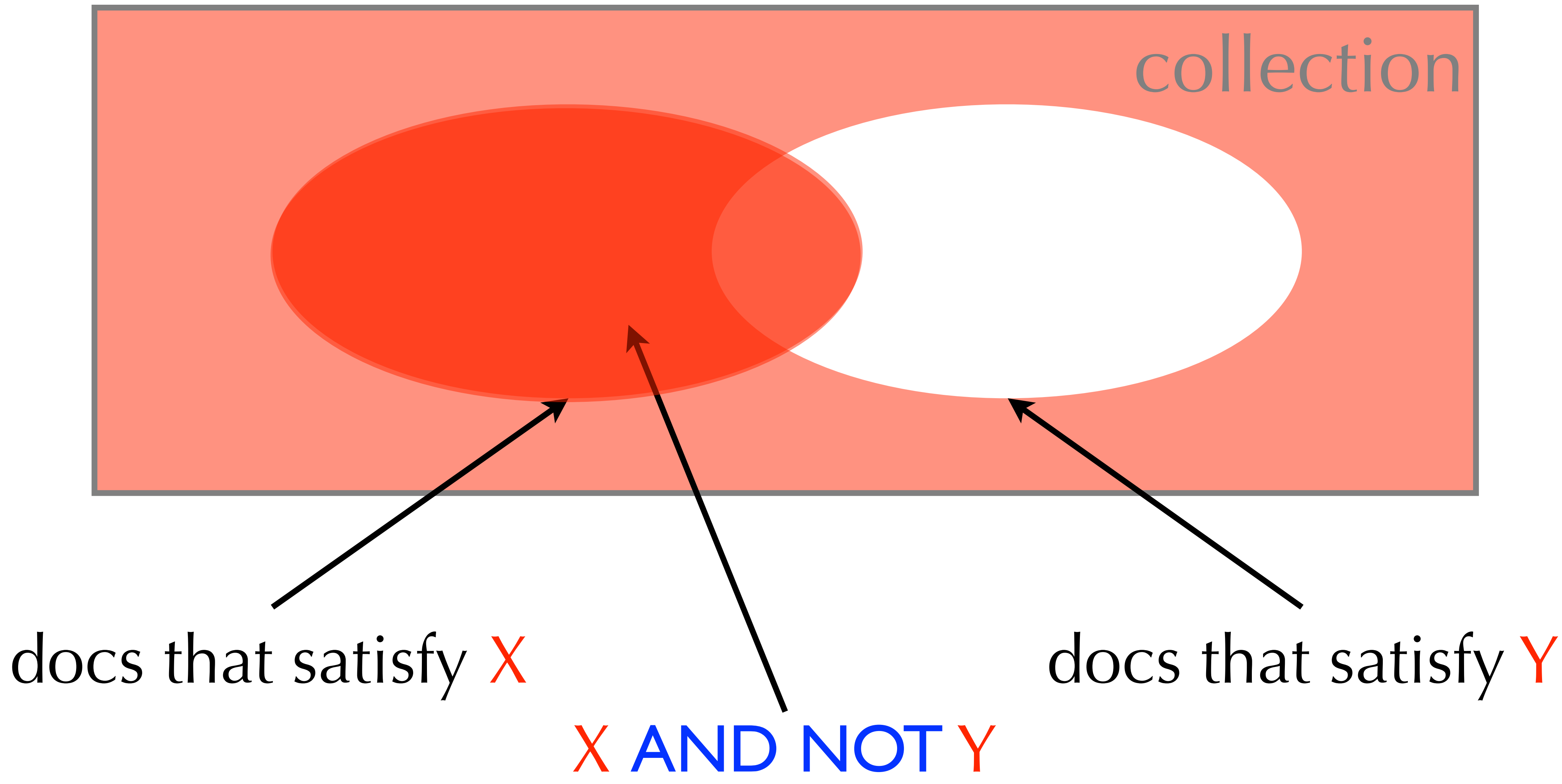
Boolean Retrieval

- X OR Y



Boolean Retrieval

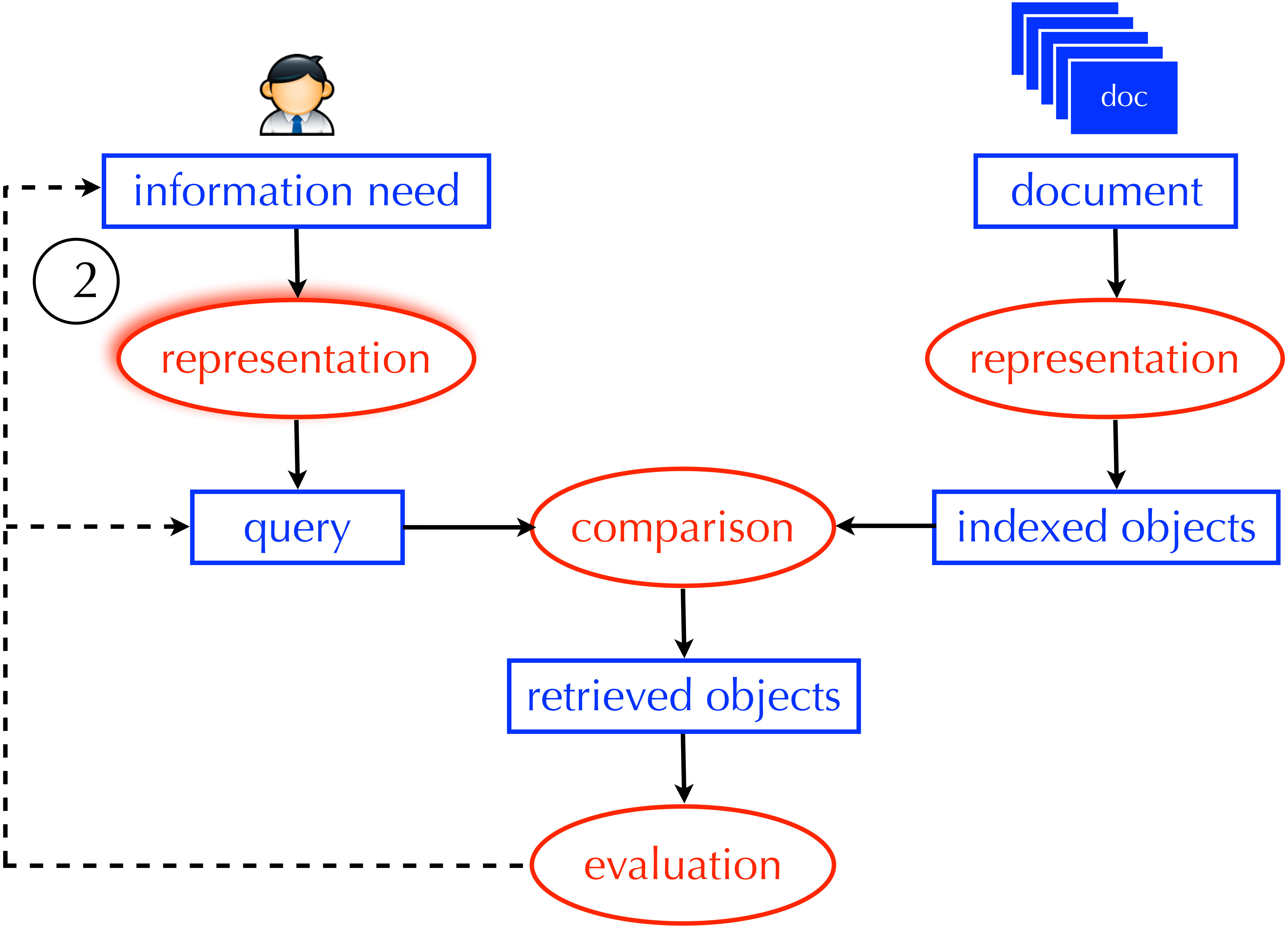
- X AND NOT Y



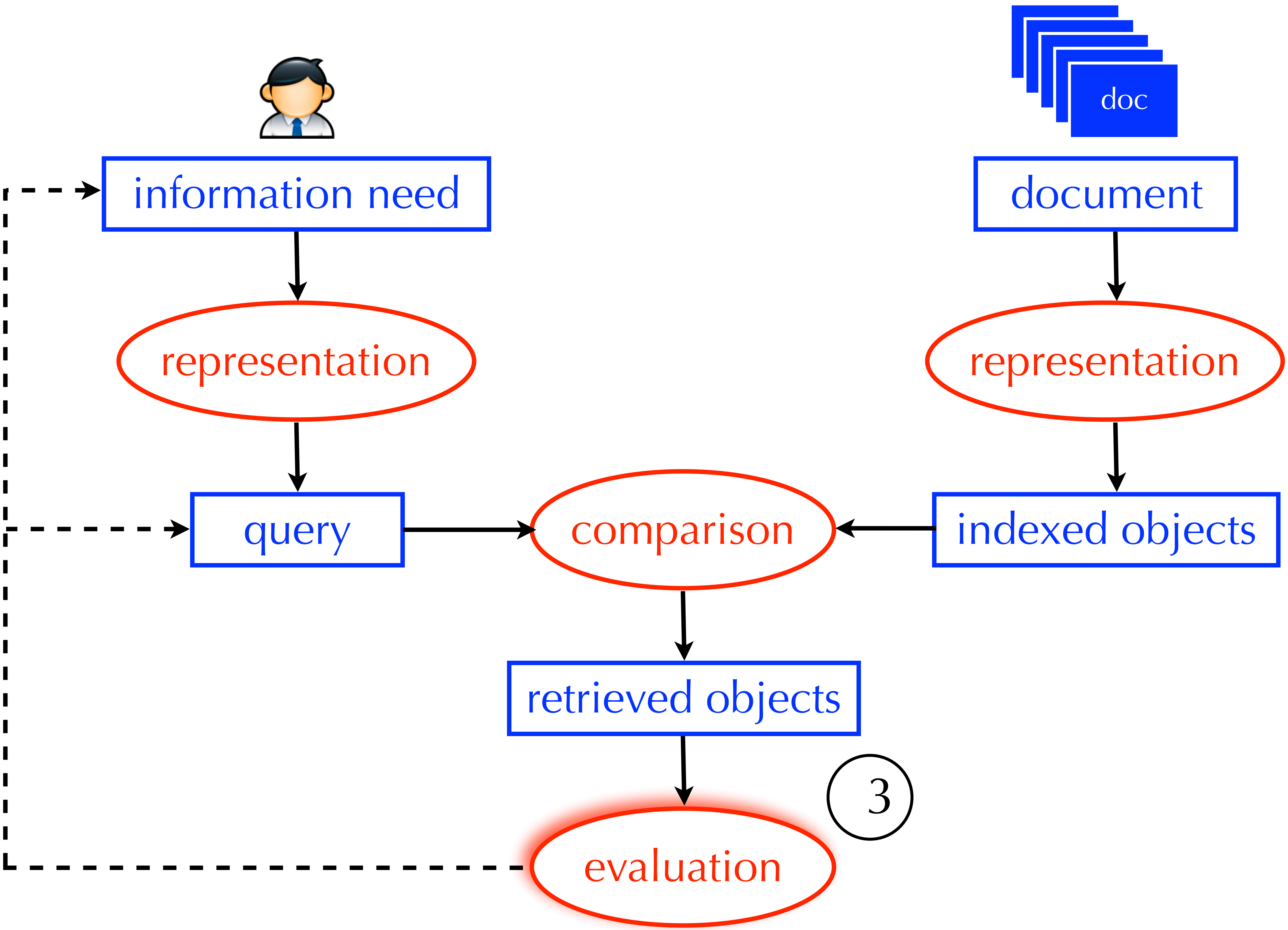
Boolean Retrieval
advantages

- Easy for the system (no ambiguity in the query)
  - the burden is on the user to formulate the right query
- The user gets transparency and control
  - lots of results ➔ the query is too broad
  - no results ➔ the query is too narrow
- Common strategy for finding the right balance:
  - if the query is too broad, add AND or AND NOT constraints
  - if the query is too narrow, add OR constraints

Information Need Representation



Evaluation

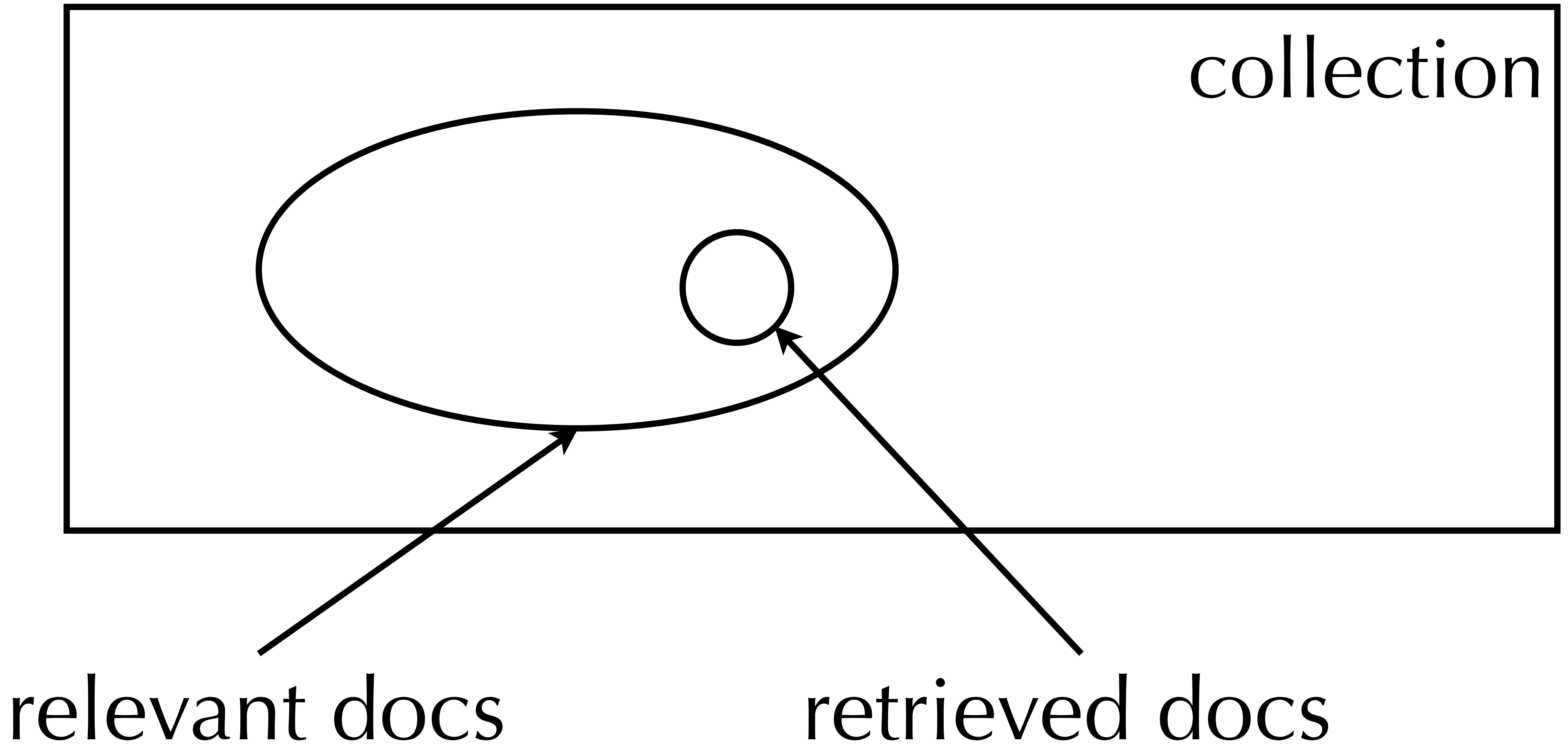


doc
information need
document
representation
representation
query
comparison
indexed objects
retrieved objects
3
evaluation

Boolean Retrieval

evaluation

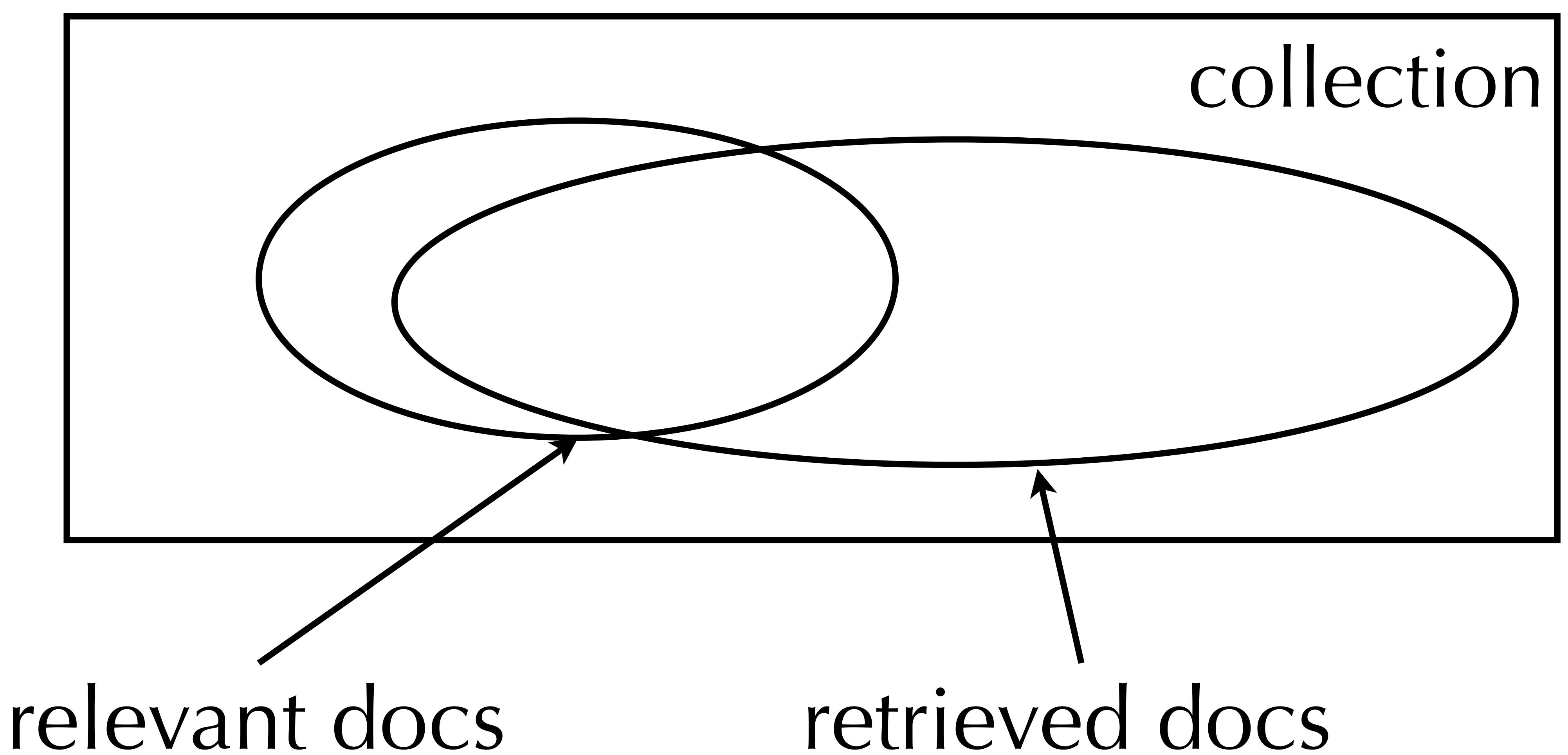
- Assumption: the user wants to find all the relevant documents and only the relevant documents
- If the query is too specific, it may retrieve relevant documents, but not enough



Boolean Retrieval

evaluation

- Assumption: the user wants to find all the relevant documents and only the relevant documents
- If the query is too broad, it may retrieve many relevant documents, but also many non-relevant ones



Boolean Retrieval

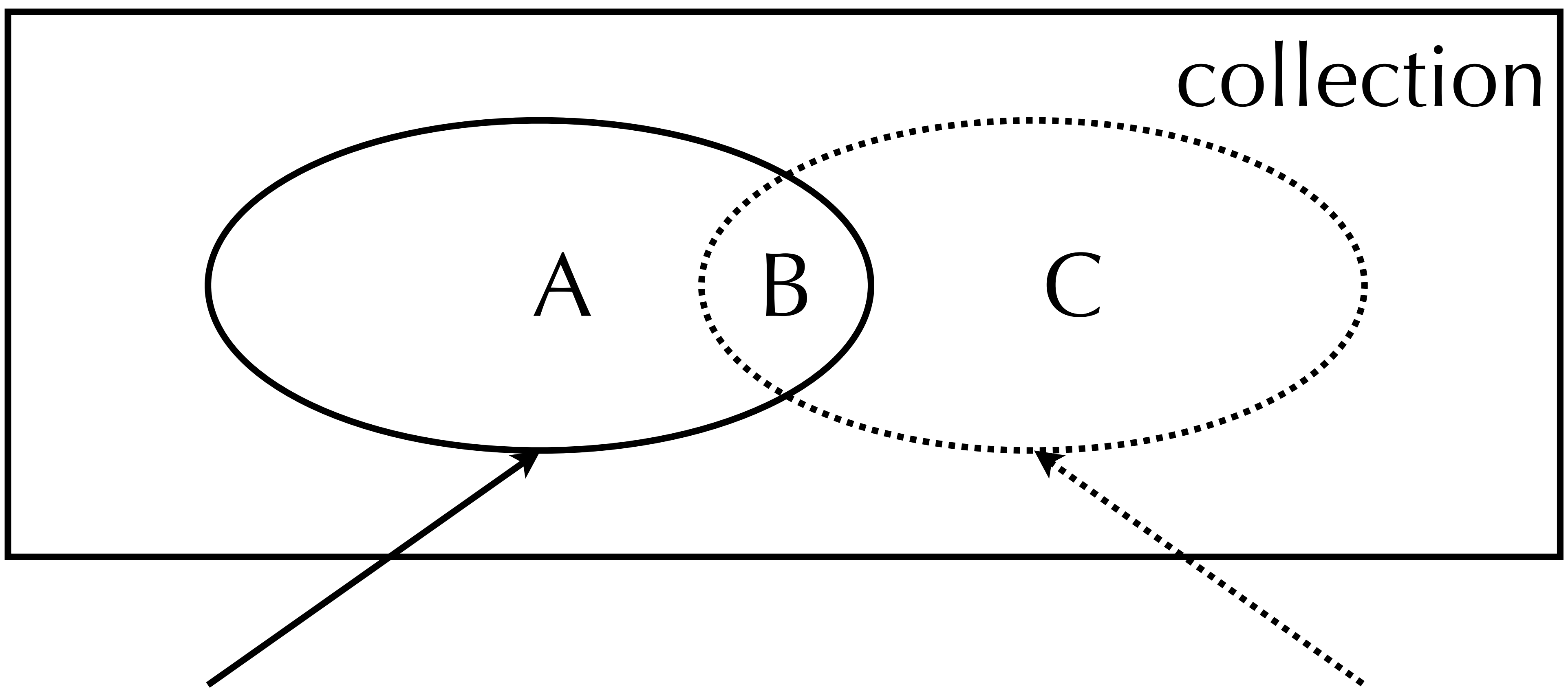
evaluation

- Assumption: the user wants to find all the relevant documents and only the relevant documents
- Precision: the percentage of retrieved documents that are relevant
- Recall: the percentage of relevant documents that are retrieved
- The goal of the user is to find the right balance between precision and recall
- These are important evaluation measures that we will see over and over again

Boolean Retrieval
evaluation

- Precision = $\frac{?}{?}$

B = intersection of A and C



A = relevant docs

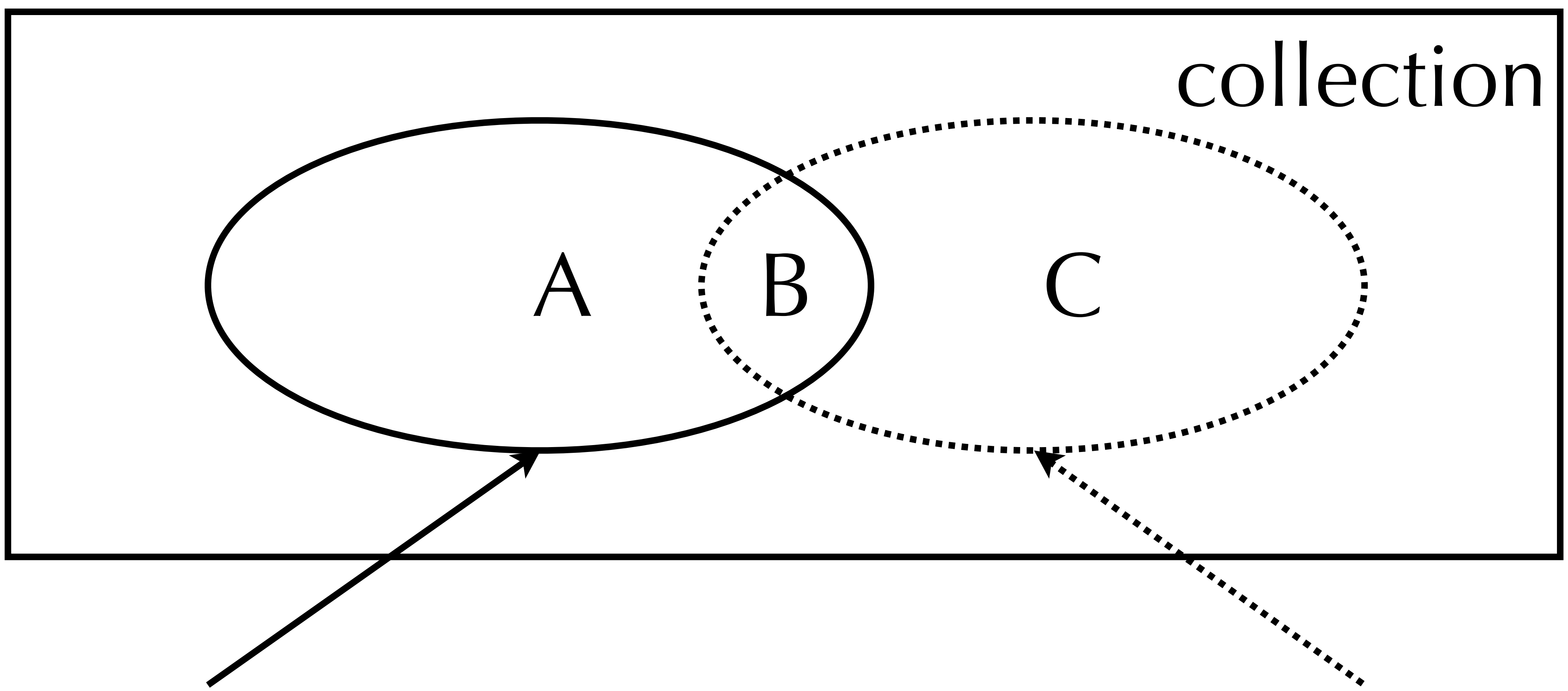
C = retrieved docs

Boolean Retrieval

evaluation

- Precision = $\frac{|B|}{|C|}$

B = intersection of A and C



A = relevant docs

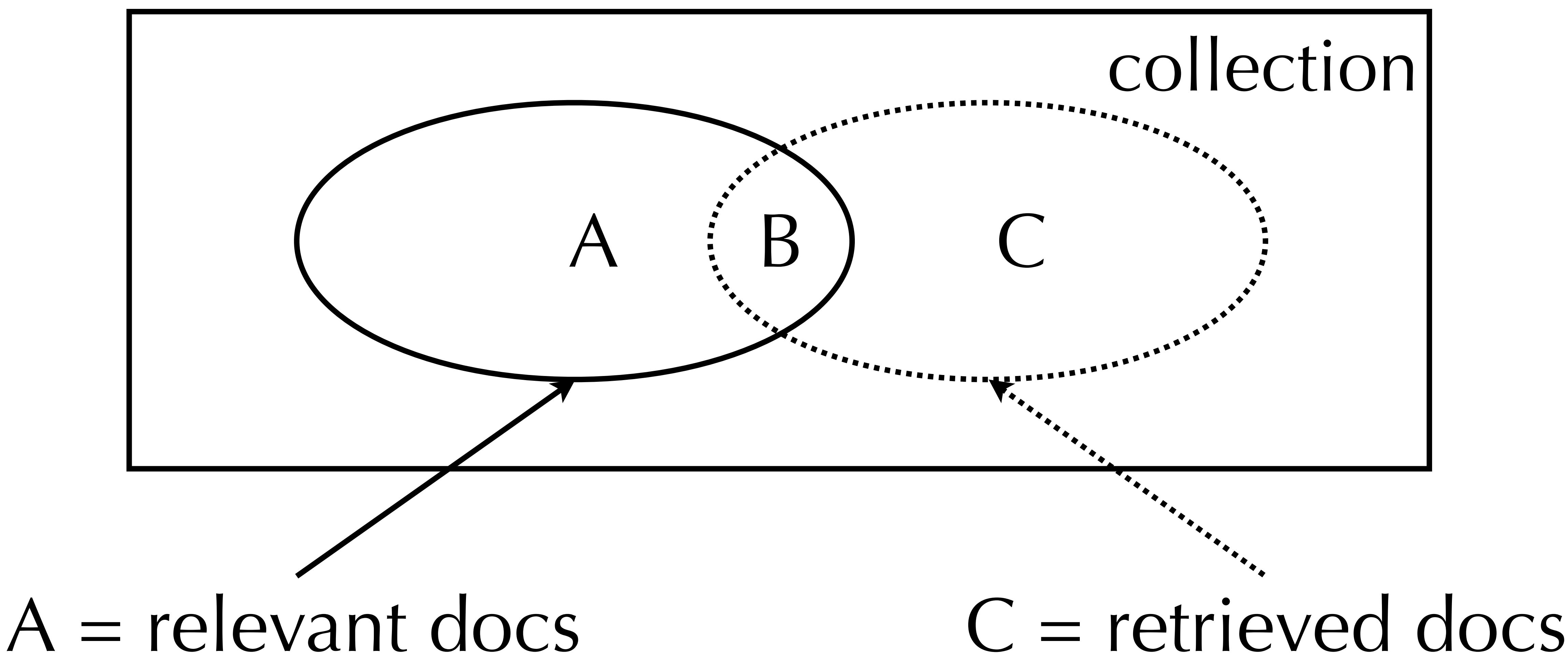
C = retrieved docs

Boolean Retrieval

evaluation

- Recall = $\frac{?}{?}$

B = intersection of A and C

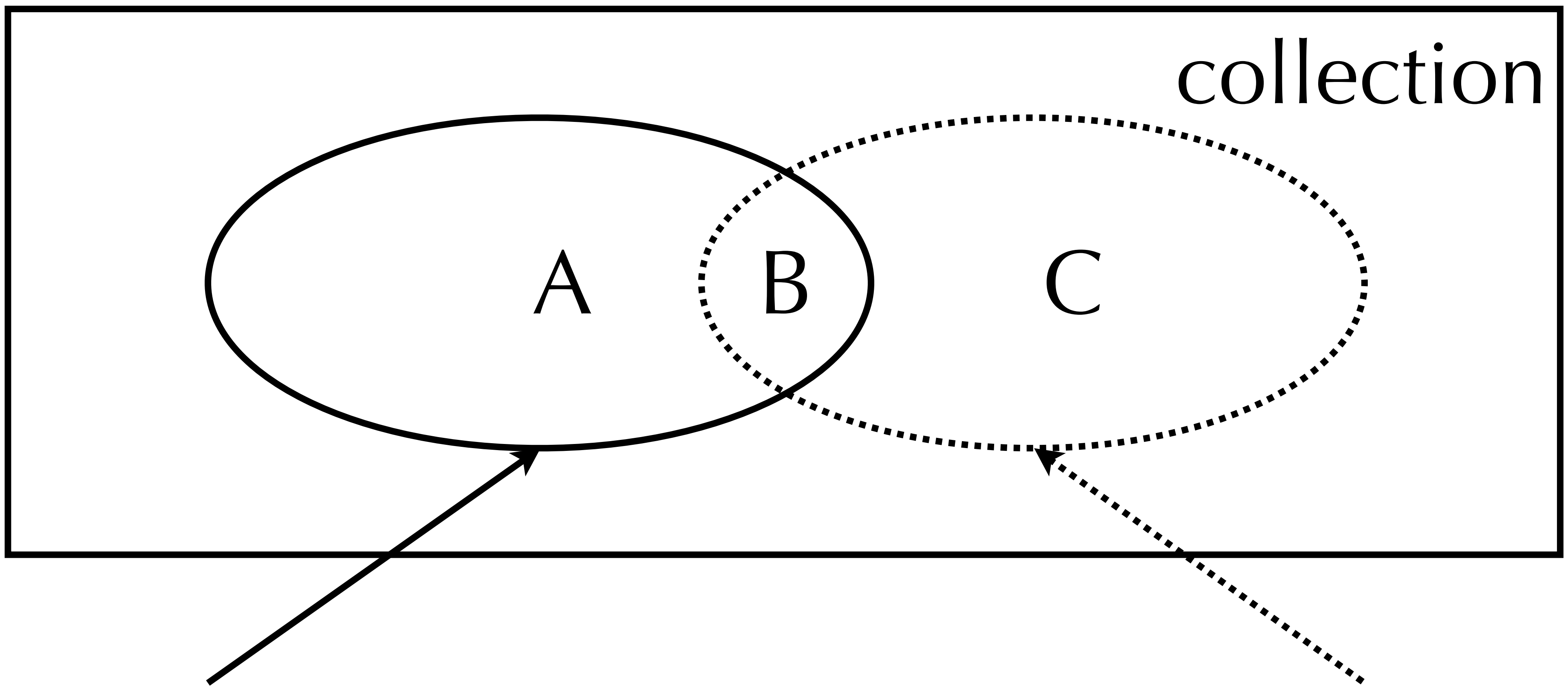


Boolean Retrieval

evaluation

- Recall = $\frac{|B|}{|A|}$

B = intersection of A and C



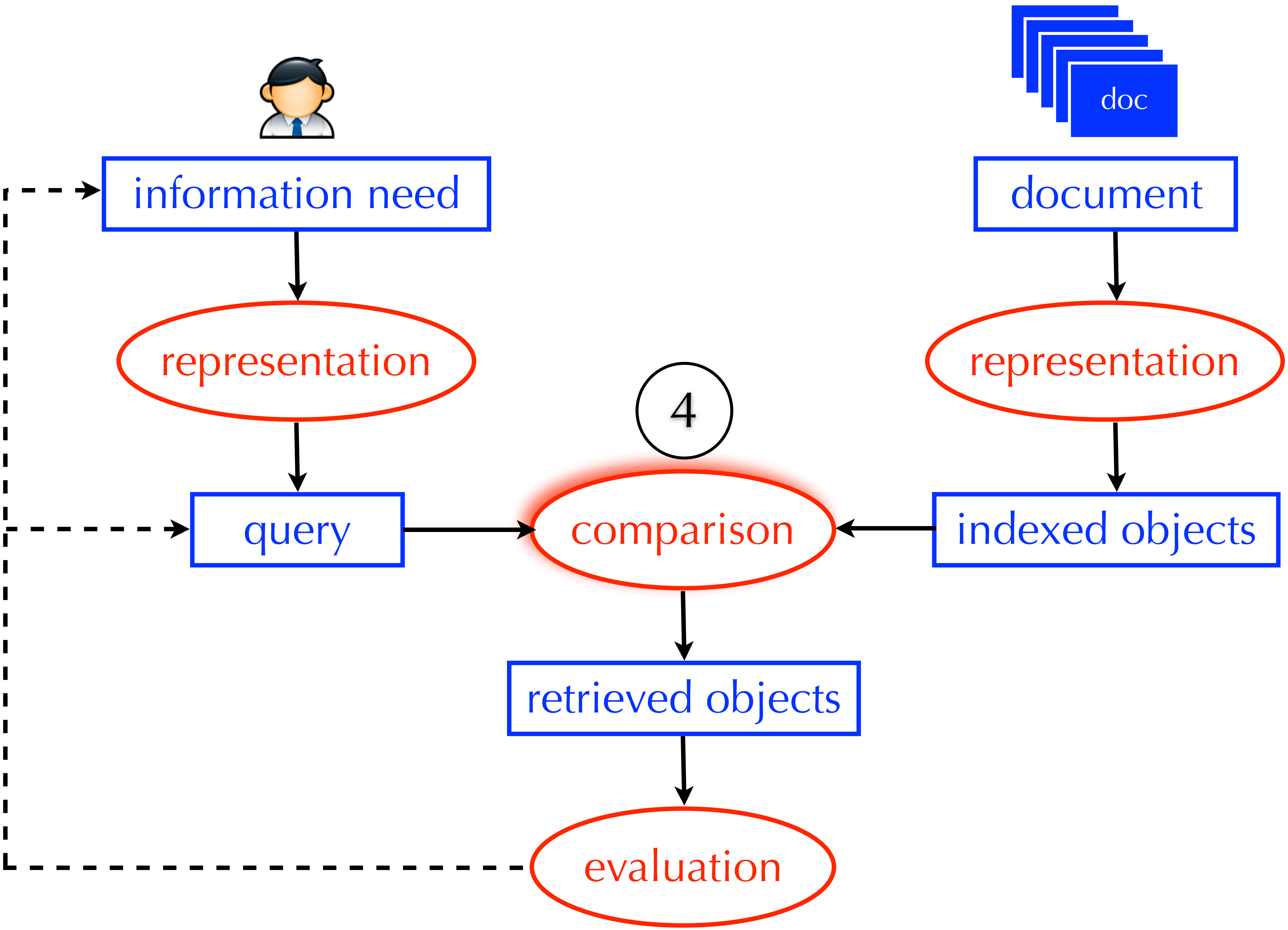
A = relevant docs

C = retrieved docs

Boolean Retrieval
evaluation

- If the query is too specific, precision may be high, but recall will probably be low
- If the query is too broad, recall may be high, but precision will probably be low
- Extreme cases:
  - a query that retrieves a single <u>relevant</u> document will have perfect precision, but low recall (unless only that one document is relevant)
  - a query that retrieves the entire collection will have perfect recall, but low precision (unless the entire collection is relevant)

Performing Retrieval



Most Basic View of a Search Engine

- A search engines does not scan each document to see if it satisfies the query
- That may be effective, but not efficient
- It uses an index to quickly locate the relevant documents
- Index: a list of concepts and pointers to documents that discuss them

Index from Manning et al., 2008

Indexing and Query Processing

- Next, we will see two types of indices and how they can be used to retrieve documents
- Bit-map index vs. variable-length inverted-list index
- In particular, we'll focus on how they can be used to evaluate boolean queries quickly
- Both produce the same output
- However, they go about it in different ways

Binary Full-text Representation
bitmap index

	a	aardvark	abacus	abba	able	...	zoom
doc_1	1	0	0	0	0	...	1
doc_2	0	0	0	0	1	...	1
::	::	::	::	::	::	...	0
doc_m	0	0	1	1	0	...	0

- 1 = the word appears in the document at least once
- 0 = the word does not appear in the document

Binary Full-text Representation
bitmap index

	a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
doc_1	1	0	0	0	0	...	1
doc_2	0	0	0	0	1	...	1
::	::	::	::	::	::	...	0
doc_m	0	0	1	1	0	...	0

- 1 = the word appears in the document at least once
- 0 = the word does <u>not</u> appear in the document
- Does not represent word frequency, order, or location information

Binary Full-text Representation

bitmap index

	a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
doc_1	1	0	0	0	0	...	1
doc_2	0	0	0	0	1	...	1
::	::	::	::	::	::	...	0
doc_m	0	0	1	1	0	...	0

- This type of document representation is known as a bag of words representation
- Term location information is lost
  - dog bites man = man bites dog
- Simplistic, but surprisingly effective for search

Binary Full-text Representation

bitmap index

	a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
doc_1	1	0	0	0	0	...	1
doc_2	0	0	0	0	1	...	1
::	::	::	::	::	::	...	0
doc_m	0	0	1	1	0	...	0

- Every indexed term is associated with an inverted list
- Inverted list: marks the docs where the term appears at least once
- This type of inverted list is called a bit-vector
- In a bit-map index, all inverted lists (or bit-vectors) have the same number of elements

Processing a Boolean Query

- Query: *Jack* AND *Jill*

doc_1	Jack and Jill went up the hill
doc_2	To fetch a pail of water.
doc_3	Jack fell down and broke his crown,
doc_4	And Jill came tumbling after.
doc_5	Up Jack got, and home did trot,
doc_6	As fast as he could caper,
doc_7	To old Dame Dob, who patched his nob
doc_8	With vinegar and brown paper.

Processing a Boolean Query

- Query: *Jack* AND *Jill*

docid	*text*	*Jack*	*Jill*
doc_1	Jack and Jill went up the hill	1	1
doc_2	To fetch a pail of water.	0	0
doc_3	Jack fell down and broke his crown,	1	0
doc_4	And Jill came tumbling after.	0	1
doc_5	Up Jack got, and home did trot,	1	0
doc_6	As fast as he could caper,	0	0
doc_7	To old Dame Dob, who patched his nob	0	0
doc_8	With vinegar and brown paper.	0	0

Processing a Boolean Query

- Query: *Jack* AND *Jill*

	Jack	*Jill*	*Jack* AND *Jill*
doc_1	1	1	?
doc_2	0	0	?
doc_3	1	0	?
doc_4	0	1	?
doc_5	1	0	?
doc_6	0	0	?
doc_7	0	0	?
doc_8	0	0	?

Processing a Boolean Query

- Query: *Jack* AND *Jill*

	Jack	*Jill*	*Jack* AND *Jill*
doc_1	1	1	1
doc_2	0	0	0
doc_3	1	0	0
doc_4	0	1	0
doc_5	1	0	0
doc_6	0	0	0
doc_7	0	0	0
doc_8	0	0	0

Processing a Boolean Query

- Query: *Jack* OR *Jill*

docid	*text*	*Jack*	*Jill*
doc_1	Jack and Jill went up the hill	1	1
doc_2	To fetch a pail of water.	0	0
doc_3	Jack fell down and broke his crown,	1	0
doc_4	And Jill came tumbling after.	0	1
doc_5	Up Jack got, and home did trot,	1	0
doc_6	As fast as he could caper,	0	0
doc_7	To old Dame Dob, who patched his nob	0	0
doc_8	With vinegar and brown paper.	0	0

Processing a Boolean Query

- Query: *Jack* OR *Jill*

	Jack	*Jill*	*Jack* OR *Jill*
doc_1	1	1	?
doc_2	0	0	?
doc_3	1	0	?
doc_4	0	1	?
doc_5	1	0	?
doc_6	0	0	?
doc_7	0	0	?
doc_8	0	0	?

Processing a Boolean Query

- Query: *Jack* OR *Jill*

	Jack	*Jill*	*Jack* OR *Jill*
doc_1	1	1	1
doc_2	0	0	0
doc_3	1	0	1
doc_4	0	1	1
doc_5	1	0	1
doc_6	0	0	0
doc_7	0	0	0
doc_8	0	0	0

Processing a Boolean Query

- Query: *Jack* AND (*up* OR *down*)

	up	*down*	*up* OR *down*	*Jack*	*Jack* AND (*up* OR *down*)
doc_1	1	0	?	1	?
doc_2	0	0	?	0	?
doc_3	0	1	?	1	?
doc_4	0	0	?	0	?
doc_5	1	0	?	1	?
doc_6	0	0	?	0	?
doc_7	0	0	?	0	?
doc_8	0	0	?	0	?

Processing a Boolean Query

- Query: *Jack* AND (*up* OR *down*)

	up	*down*	*up* OR *down*	*Jack*	*Jack* AND (*up* OR *down*)
doc_1	1	0	1	1	?
doc_2	0	0	0	0	?
doc_3	0	1	1	1	?
doc_4	0	0	0	0	?
doc_5	1	0	1	1	?
doc_6	0	0	0	0	?
doc_7	0	0	0	0	?
doc_8	0	0	0	0	?

Processing a Boolean Query

- Query: *Jack* AND (*up* OR *down*)

	up	*down*	*up* OR *down*	*Jack*	*Jack* AND (*up* OR *down*)
doc_1	1	0	1	1	1
doc_2	0	0	0	0	0
doc_3	0	1	1	1	1
doc_4	0	0	0	0	0
doc_5	1	0	1	1	1
doc_6	0	0	0	0	0
doc_7	0	0	0	0	0
doc_8	0	0	0	0	0

Processing a Boolean Query

- Query: *Jack* AND NOT *Jill*

	Jack	*Jill*	*Jack* AND NOT *Jill*
doc_1	1	1	
doc_2	0	0	
doc_3	1	0	
doc_4	0	1	
doc_5	1	0	
doc_6	0	0	
doc_7	0	0	
doc_8	0	0	

Processing a Boolean Query

- Query: *Jack* AND NOT *Jill*

	Jack	*Jill*	NOT *Jill*	*Jack* AND NOT *Jill*
doc_1	1	1	?	?
doc_2	0	0	?	?
doc_3	1	0	?	?
doc_4	0	1	?	?
doc_5	1	0	?	?
doc_6	0	0	?	?
doc_7	0	0	?	?
doc_8	0	0	?	?

Processing a Boolean Query

- Query: *Jack* AND NOT *Jill*

	Jack	*Jill*	NOT *Jill*	*Jack* AND NOT *Jill*
doc_1	1	1	0	?
doc_2	0	0	1	?
doc_3	1	0	1	?
doc_4	0	1	0	?
doc_5	1	0	1	?
doc_6	0	0	1	?
doc_7	0	0	1	?
doc_8	0	0	1	?

Processing a Boolean Query

- Query: *Jack* AND NOT *Jill*

	Jack	*Jill*	NOT *Jill*	*Jack* AND NOT *Jill*
doc_1	1	1	0	0
doc_2	0	0	1	0
doc_3	1	0	1	1
doc_4	0	1	0	0
doc_5	1	0	1	1
doc_6	0	0	1	0
doc_7	0	0	1	0
doc_8	0	0	1	0

The Binary Full-text Representation

	a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
doc_1	1	0	0	0	0	...	1
doc_2	0	0	0	0	1	...	1
::	::	::	::	::	::	...	0
doc_m	0	0	1	1	0	...	0

- These are <u>fixed-length</u> inverted lists, each of size *m* (the number of documents in the collection)
- Are these inverted lists efficient in terms of storage?

Statistical Properties of Text

sneak preview!

- IMDB collection (movies, artist/role, plot descriptions)
  - number of documents: **230,721**
  - number of term occurrences: **36,989,629**
  - number of unique terms: ????

- Term Statistics
  - Most terms occur very infrequently
  - ??? of all terms occur only once
  - ??? occur 5 times or less
  - ??? occur 10 times or less
  - Only ??? occur 50 times or more

Statistical Properties of Text

sneak preview!

- IMDB collection (movies, artist/role, plot descriptions)
  - number of documents: **230,721**
  - number of term occurrences: **36,989,629**
  - number of unique terms: **424,035**

- Term Statistics
  - Most terms occur very infrequently
  - ??? of all terms occur only once
  - ??? occur 5 times or less
  - ??? occur 10 times or less
  - Only ??? occur 50 times or more

Statistical Properties of Text

sneak preview!

- IMDB collection (movies, artist/role, plot descriptions)
  - number of documents: 230,721
  - number of term occurrences: 36,989,629
  - number of unique terms: 424,035

- Term Statistics
  - Most terms occur very infrequently
  - 44% of all terms occur only once
  - 77% occur 5 times or less
  - 85% occur 10 times or less
  - Only 6% occur 50 times or more

Sparse Representation of an Inverted List

- Most terms appear in only a few documents
- Most bit-vectors have many 0′s and only a few 1′s
- A bitmap index is very inefficient
- Alternative: represent only the 1′s:
  - aardvark: 00101011....
  - aardvark: *df* = 18; 3, 5, 7, 8, ...
- *df* = number of documents in which the term appears at least once
- Each document has a unique identifier (docid)

Inverted Index Full-text Representation

a	*aardvark*	*abacus*	*abba*	*able*	...	*zoom*
df=3421	*df=22*	*df=19*	*df=2*	*df=44*		*df=1*
1	33	2	33	66		54
33	56	10	150	134		
45	86	15		176		
::	::	::		::		
1022	1011	231		432		

- Variable-length inverted lists
- Each document has a unique identifier (docid)
- Why are the inverted lists sorted by docid?
- Why do we store the ***df's*** in the index?

Merging (Variable-Length) Inverted Lists

AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, increment pointer with lowest docid*
3. *Repeat until (1) end of one list <u>and</u> (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
df=3	*df=5*	*count=1*
1	1	1
3	3	
5	4	
	5	
	8	

Merging (Variable-Length) Inverted Lists
AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, increment pointer with lowest docid*
3. *Repeat until (1) end of one list and (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
df=3	*df=5*	*count=2*
1	1	1
3	3	3
5	4	
	5	
	8	

Merging (Variable-Length) Inverted Lists
AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, increment pointer with lowest docid*
3. *Repeat until (1) end of one list and (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
df=3	*df=5*	*count=2*
1	1	1
3	3	3
5	4	
	5	
	8	

Merging (Variable-Length) Inverted Lists

AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, increment pointer with lowest docid*
3. *Repeat until (1) end of one list and (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
df=3	*df=5*	*count=3*
1	1	1
3	3	3
5	4	5
	5	
	8	

Merging (Variable-Length) Inverted Lists

AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, increment pointer with lowest docid*
3. *Repeat until (1) end of one list and (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
df=3	*df=5*	*count=3*
1	1	1
3	3	3
5	4	5
	5	
	8	

stop!

Merging (Variable-Length) Inverted Lists
AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, increment pointer with lowest docid*
3. *Repeat until (1) end of one list <u>and</u> (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
df=3	*df=5*	*count=3*
1	1	1
3	3	3
5	4	5
	5	
	8	
	10	
	35	

If the inverted list for "and" was longer, would it make sense to continue? Why or why not?

Merging (Variable-Length) Inverted Lists
AND

- Query: *Jack* AND *and*

1. *If docids are equal, add docid to results[…] increment both[…]*
2. *If docids are n[…] increment poin[…] lowest docid*
3. *Repeat until (1) end of one list <u>and</u> (2) docid from other list is greater*

Jack	*and*	*Jack* AND *and*
	[…]=5	*count=3*
	1	1
	3	3
	4	5
	5	
	[illegible]	
	10	
	35	

This is why the inverted lists are sorted in ascending order of docid!

If the inverted list for "and" was longer, would it make sense to continue? Why or why not?

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	3
5	4	4
	5	5
	8	8

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	
5	4	
	5	
	8	

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	3
5	4	
	5	
	8	

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	3
5	4	4
	5	
	8	

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	3
5	4	4
	5	5
	8	

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	3
5	4	4
	5	5
	8	8

stop!

Merging (Variable-Length) Inverted Lists
OR

- Query: *Jack* OR *and*

1. *If docids are equal, add docid to results and increment both pointers*
2. *If docids are not equal, add lowest docid and increment its pointer*
3. *Repeat until end of both lists*

Jack	*and*	*Jack* OR *and*
df=3	*df=5*	*count=5*
1	1	1
3	3	3
5	4	4
	5	5
	8	8

- Which is more expensive (on average) AND or OR?

Merging (Variable-Length) Inverted Lists

- In some cases, the search engine has a choice in the order of operations
- Query: *Abraham* AND *Lincoln* AND *President*
  - option 1: (*Abraham* AND *Lincoln*) AND *President*
  - option 2: *Abraham* AND (*Lincoln* AND *President*)
  - option 3: (*Abraham* AND *President*) AND *Lincoln*
- Which is <u>probably</u> the least effective order of operations?

Merging (Variable-Length) Inverted Lists

- Which is probably the most effective order of operations?

president	*abraham*	*lincoln*
df=302	*df=45*	*df=5*
XX	XX	XX
XX	XX	XX
XX	XX	XX
XX	XX	XX
XX	XX	XX
::	::	
XX	XX	

Retrieval Model 1: Unranked Boolean

- Retrieves the set of documents that match the boolean query (an “exact-match” retrieval model)
- Returns results in no particular order (ordered by date?)
- This is problematic with large collections
  - requires complex queries to reduce the result set to a manageable size
- Can we do better?

Retrieval Model 2: Ranked Boolean

University	*North*	*Carolina*	*UNC*
df=6	*df=4*	*df=3*	*df=5*
1, 4	1, 4	1, 4	1, 4
10, 1	10, 5	10, 5	10, 1
15, 2	16, 1	16, 1	16, 4
16, 1	68, 1		33, 2
33, 5			56, 10
67, 7			

- *docid = document identifier*
- *tf = term frequency (# of times the term appears in the document)*

Retrieval Model 2: Ranked Boolean

- At each step, keep a list of documents that match the query and their scores (a.k.a. a "priority queue")
- Score computation:
  - A **AND** B: adjust the document score based on the minimum frequency/score associated with expression A and expression B
  - A **OR** B: adjust the document score based on the sum of frequencies/scores associated with expression A and expression B

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

University	*North*	*Carolina*	*UNC*
df=6	*df=4*	*df=3*	*df=5*
1, 4	1, 4	1, 4	1, 4
10, 1	10, 5	10, 5	10, 1
15, 2	16, 1	16, 1	16, 4
16, 1	68, 1		33, 2
33, 5			56, 10
68, 7			

- AND → min
- OR → sum

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

University	*North*	*Carolina*	*Result_1*
df=6	*df=4*	*df=3*	*count=??*
1, 4	1, 4	1, 4	
10, 1	10, 5	10, 5	
15, 2	16, 1	16, 1	
16, 1	68, 1		
33, 5			
68, 7			

- AND → min
- OR → sum

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

University	*North*	*Carolina*	*Result_1*
df=6	*df=4*	*df=3*	*count=3*
1, 4	1, 4	1, 4	1, 4
10, 1	10, 5	10, 5	10, 1
15, 2	16, 1	16, 1	16, 1
16, 1	68, 1		
33, 5			
68, 7			

- AND → min
- OR → sum

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

Result_1	*UNC*	*Query*
count=3	*df=5*	*count=??*
1, 4	1, 4	
10, 1	10, 1	
16, 1	16, 4	
	33, 2	
	56, 10	

- AND → min
- OR → sum

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

Result_1	*UNC*	*Query*
count=3	*df=5*	*count=5*
1, 4	1, 4	1, 8
10, 1	10, 1	10, 2
16, 1	16, 4	16, 5
	33, 2	33, 2
	56, 10	56, 10

- AND → min
- OR → sum

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

University	*North*	*Carolina*	*UNC*	*Query*
df=6	*df=4*	*df=3*	*df=5*	*count=5*
1, 4	1, 4	1, 4	1, 4	1, 8
10, 1	10, 5	10, 5	10, 1	10, 2
15, 2	16, 1	16, 1	16, 4	16, 5
16, 1	68, 1		33, 2	33, 2
33, 5			56, 10	56, 10
68, 7				

- Conceptually, what do these document scores indicate?

Retrieval Model 2: Ranked Boolean

- Query: (*University* AND *North* AND *Carolina*) OR *UNC*

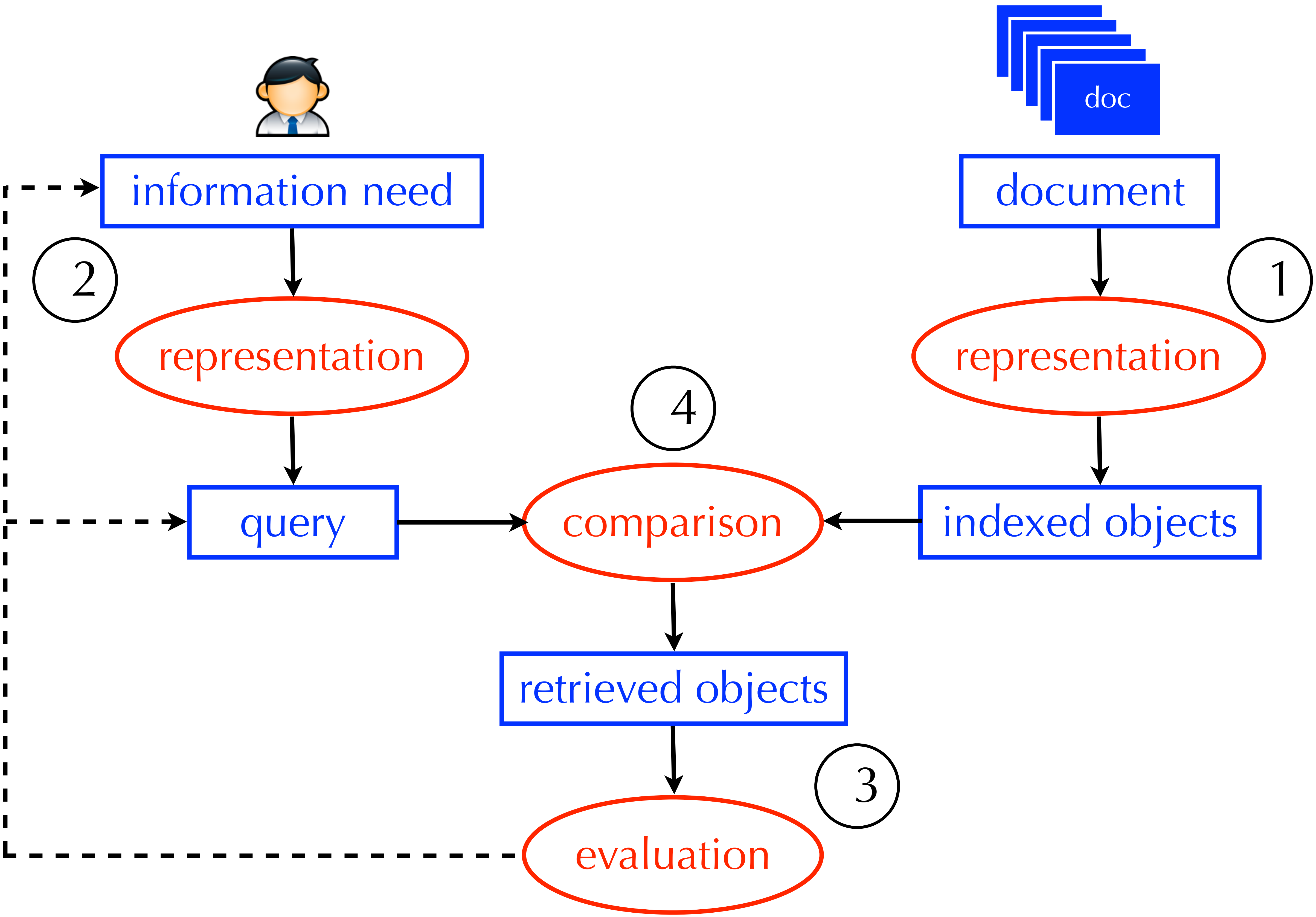
University	*North*	*Carolina*	*UNC*	*Query*
df=6	*df=4*	*df=3*	*df=5*	*count=5*
1, 4	1, 4	1, 4	1, 4	1, 8
10, 1	10, 5	10, 5	10, 1	10, 2
15, 2	16, 1	16, 1	16, 4	16, 5
16, 1	68, 1		33, 2	33, 2
33, 5			56, 10	56, 10
68, 7				

- The *scores* correspond to the number of ways in which the document redundantly satisfies the query

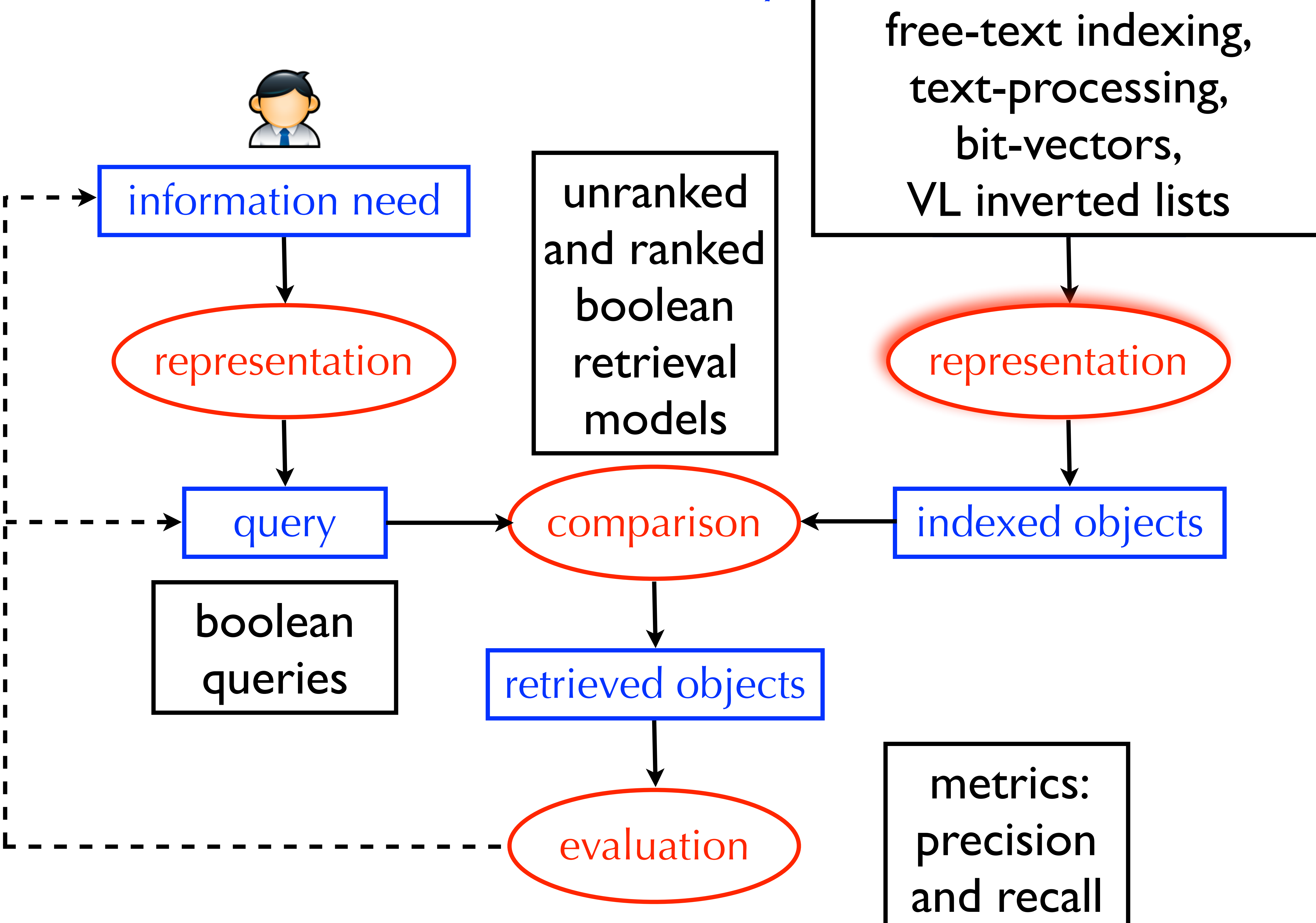
Retrieval Model 2: Ranked Boolean

- Advantages:
  - same as unranked boolean: efficient, predictable, easy to understand, works well when the user knows what to look for
  - the user may be able to find relevant documents quicker and may not need to examine the entire result set
- Disadvantages:
  - same as unranked boolean: works well when the user <u>knows what to look for</u>
  - difficult to balance precision and recall

Summary


doc
information need
document
2
1
representation
representation
4
query
comparison
indexed objects
retrieved objects
3
evaluation

Summary


controlled vocabularies,
free-text indexing,
text-processing,
bit-vectors,
VL inverted lists
information need
representation
unranked
and ranked
boolean
retrieval
models
representation
query
comparison
indexed objects
boolean
queries
retrieved objects
evaluation
metrics:
precision
and recall

Take Home Message

- Congratulations! Now, you know how a boolean search engine works
- How are indexes structured?
- How are boolean queries processed quickly?
- What are some time-saving hacks?
- How are boolean retrieval sets evaluated?
- How can we prioritize documents based on how much they satisfy the boolean constraints?