Introduction to Ad-hoc Retrieval

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

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 Tapataalk.

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_screenshots_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-screenshots-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 screenshot apps work on your ...

How to take screenshots on the HTC EVO 4G
www.goodandevo.com/...how-to-take-screenshots-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

- Metabolic and behavioural effects of sucrose and fructose/glucose drinks in the rat.
  1. Sheludiakova A, Rooney K, Boakes RA.
     PMID: 21800086 [PubMed - as supplied by publisher]
     Related citations

- The impact of fructose on renal function and blood pressure.
  2. Kretowicz M, Johnson RJ, Ishimoto T, Nakagawa T, Manitius J.
     Free full text Related citations

- The role of salt in the pathogenesis of fructose-induced hypertension.
  3. Soleimani M, Alborzi P.
     Free full text Related citations

  4. Shao Q, Chin KV.
     Free full text Related citations

- Obesity and energy balance: is the tail wagging the dog?
  5. Wells JC, Siervo M.
     PMID: 21772313 [PubMed - as supplied by publisher]
     Related citations
Examples
discussion forum search

<table>
<thead>
<tr>
<th>Thread / Thread Starter</th>
<th>Last Post</th>
<th>Replies</th>
<th>Views</th>
<th>Forum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Installed Mac Apps</td>
<td>Jul 18, 2011</td>
<td>34</td>
<td>1,953</td>
<td>Mac Applications and Mac App Store</td>
</tr>
<tr>
<td>Translucent mail notify</td>
<td>Jun 12, 2011</td>
<td>8</td>
<td>277</td>
<td>Mac Applications and Mac App Store</td>
</tr>
<tr>
<td>How do I move Thunderbird e-mail from PC to Mac</td>
<td>Oct 12, 2010</td>
<td>7</td>
<td>35,011</td>
<td>Mac Applications and Mac App Store</td>
</tr>
<tr>
<td>Re-installing 10.6 while preserving user data?</td>
<td>Feb 28, 2010</td>
<td>5</td>
<td>708</td>
<td>Mac OS X</td>
</tr>
<tr>
<td>New to MAC - Dissatisfied - text size</td>
<td>Jan 19, 2010</td>
<td>157</td>
<td>10,115</td>
<td>Mac Basics and Help</td>
</tr>
<tr>
<td>Anyone have to &quot;switch back&quot; due to $$?</td>
<td>Oct 20, 2009</td>
<td>52</td>
<td>2,688</td>
<td>MacBook</td>
</tr>
<tr>
<td>The Saga of Switching</td>
<td>May 21, 2009</td>
<td>4</td>
<td>493</td>
<td>Mac Basics and Help</td>
</tr>
<tr>
<td>Apple Mail vs Entourage</td>
<td>May 8, 2009</td>
<td>20</td>
<td>16,768</td>
<td>Mac Applications and Mac App Store</td>
</tr>
<tr>
<td>Teacher accuses student using linux of copyright infringement!</td>
<td>Dec 15, 2008</td>
<td>56</td>
<td>1,763</td>
<td>Community Discussion</td>
</tr>
<tr>
<td>Timemachine Duplicates?</td>
<td>Nov 27, 2008</td>
<td>18</td>
<td>2,206</td>
<td>Mac OS X</td>
</tr>
</tbody>
</table>
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

information need → representation → query → comparison → retrieved objects → evaluation

- document
- indexed objects
- representation
- retrieved objects
Basic Information Retrieval Process

- Information need
  - Representation
  - Query
  - Comparison
  - Retrieved objects
  - Evaluation

System:
- Document
  - Representation
  - Indexed objects
Next Two Lectures

1. Document representation
2. Information need representation
3. Retrieved objects evaluation
4. Query comparison
Most Basic View of a Search Engine

- A search engine does not 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

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 of 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
If we view option 1 and option 2 as two extremes, where does this particular index fit in?
Document Representation
option 1: controlled vocabulary

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

Assigned to documents by annotators (or automatically)
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

<table>
<thead>
<tr>
<th>MeSH Heading</th>
<th>Eukaryota</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree Number</td>
<td>B01</td>
</tr>
<tr>
<td>Annotation</td>
<td>do not confuse with EUKARYOTIC CELLS; specific algae and protozoa are located under various groups tree under Eukaryota</td>
</tr>
<tr>
<td>Scope Note</td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entry Term</th>
<th>Eucarya</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry Term</td>
<td>Eukarya</td>
</tr>
<tr>
<td>Entry Term</td>
<td>Eukaryotes</td>
</tr>
<tr>
<td>Allowable Qualifiers</td>
<td>CH CL CY DE EN GD GE IM IP ME PH PY RE UL VI</td>
</tr>
<tr>
<td>Previous Indexing</td>
<td>Eukaryotic Cells (1986-2009)</td>
</tr>
<tr>
<td>History Note</td>
<td>2010</td>
</tr>
<tr>
<td>Date of Entry</td>
<td>20090706</td>
</tr>
<tr>
<td>Unique ID</td>
<td>D056890</td>
</tr>
</tbody>
</table>
Controlled Vocabularies example

MeSH

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:
- adverse effects
- classification
- contraindications
- economics
- history
- instrumentation
- methods
- nursing
- psychology
- standards
- statistics and numerical data
- supply and distribution
- trends
- utilization
- 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-headings
sub-tree within the hierarchy
entry-terms
Controlled Vocabularies example

Results: 1 to 20 of 2697

1. Burning daylight: balancing vitamin D requirements with sensible sun exposure.
   Stalgis-Bilinski KL, Boyages J, Salisbury EL, Dunstan CR, Henderson SI, Talbot PL.
   PMID: 21470084 [PubMed - Indexed for MEDLINE]
   Related citations

   Chae JB, Lee JY, Yang SJ, Kim JG, Yoon YH.
   Free full text  Related citations

3. Metal stenting to resolve post-photodynamic therapy stricture in early esophageal cancer.
   Cheon YK.
   Free full text  Related citations

   Goldberg DJ, Hussain M.
   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
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”)
Gerard Salton (6 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 Spärck-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.
Gerard Salton (6 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 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.
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. He was editor-in-chief of the International Journal of Man-Machine Studies and was awarded the SIGIR Award for outstanding contributions to study of information retrieval (1983) – now called the Gerard Salton Award.
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.
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

Gerard Salton
From Wikipedia, the free encyclopedia

Gerard Salton (6 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 well as automatic hypertext generation". He published over 150 research articles and 3 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.

<a href="/wiki/Association_for_Computing_Machinery">ACM</a>
Free-text Indexing

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
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

• **Step 2: down-casing**

• Can change a word’s meaning, but we do it anyway
  ‣ Information = information ???
  ‣ SMART = smart ???

---

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.
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

• **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

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 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

<table>
<thead>
<tr>
<th></th>
<th>Cost of assigning index terms</th>
<th>Ambiguity of index terms</th>
<th>Detail of representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Controlled Vocabularies</strong></td>
<td>High/Low?</td>
<td>Ambiguous/Unambiguous?</td>
<td>Can represent arbitrary level of detail?</td>
</tr>
<tr>
<td><strong>Free-text Indexing</strong></td>
<td>High/Low?</td>
<td>Ambiguous/Unambiguous?</td>
<td>Can represent arbitrary level of detail?</td>
</tr>
</tbody>
</table>
## Document Representation
### controlled vocabulary vs. free-text indexing

<table>
<thead>
<tr>
<th></th>
<th>Cost of assigning index terms</th>
<th>Ambiguity of index terms</th>
<th>Detail of representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controlled Vocabularies</td>
<td>High</td>
<td>Not ambiguous</td>
<td>Can’t represent arbitrary detail</td>
</tr>
<tr>
<td>Free-text Indexing</td>
<td>Low</td>
<td>Can be ambiguous</td>
<td>Any level of detail</td>
</tr>
</tbody>
</table>

- 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
- Query
- Comparison
- Retrieved objects
- Evaluation

Document

Indexed objects
Information Need Representation

- Information need
- Representation
- Query
- Comparison
- Retrieved objects
- Evaluation
- Document
- Indexed objects
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

- $X \text{ AND } Y$

![Diagram showing the intersection of two sets: collection, docs that satisfy X, docs that satisfy Y, and the intersection X AND Y.](image)
Boolean Retrieval

- $X \text{ OR } Y$
Boolean Retrieval

- $X \text{ AND NOT } Y$
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

1. information need
2. representation
   - query
   - comparison
   - retrieved objects
   - evaluation
3. document
   - representation
   - indexed objects
Evaluation

information need

representation

document

representation

indexed objects

query

comparison

retrieved objects

evaluation

3
**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

![Diagram showing collection, relevant docs, and retrieved docs](image-url)
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

![Diagram showing the relationship between collection, relevant documents, and retrieved documents.]

- collection
- relevant docs
- retrieved docs
• **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 = \( \frac{|B|}{|C|} \)

\( B = \) intersection of \( A \) and \( C \)

\( A = \) relevant docs

\( C = \) retrieved docs
Boolean Retrieval
evaluation

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

$B = \text{intersection of } A \text{ and } C$

- $A = \text{relevant docs}$
- $C = \text{retrieved docs}$
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 relevant 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

- information need
- representation
- query
- comparison
- retrieved objects
- evaluation
- document
- representation
- indexed objects
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
### Inverted Index Full-text Representation

<table>
<thead>
<tr>
<th></th>
<th>aardvark</th>
<th>abacus</th>
<th>abba</th>
<th>able</th>
<th>...</th>
<th>zoom</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3421</td>
<td>df=22</td>
<td>df=19</td>
<td>df=2</td>
<td>df=44</td>
<td></td>
<td>df=1</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>2</td>
<td>33</td>
<td>66</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>56</td>
<td>10</td>
<td>150</td>
<td>134</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>86</td>
<td>15</td>
<td>176</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>::</td>
<td>::</td>
<td>::</td>
<td>::</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1022</td>
<td>1011</td>
<td>231</td>
<td>432</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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**

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
<th>Jack AND and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>df=5</td>
<td>count=1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
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*

<table>
<thead>
<tr>
<th></th>
<th>Jack</th>
<th>and</th>
<th></th>
<th>Jack</th>
<th>AND</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>1</td>
<td>1</td>
<td>df=5</td>
<td>3</td>
<td>3</td>
<td>count=2</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td>\</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>\</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
<th>Jack AND and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>df=5</td>
<td>count=2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>
### Merging (Variable-Length) Inverted Lists

**Query:** *Jack AND and*

1. *If docids are equal, add docid to results and increment both pointers*

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>df=5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

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*

<table>
<thead>
<tr>
<th>Jack AND and</th>
</tr>
</thead>
<tbody>
<tr>
<td>count=3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>
Merging (Variable-Length) Inverted Lists

AND

• Query: Jack AND and

1. If docids are equal, add docid to results and increment both pointers

<table>
<thead>
<tr>
<th></th>
<th>Jack</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>df=5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>count=3</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

2. If docids are not equal, increment pointer with lowest docid

<table>
<thead>
<tr>
<th></th>
<th>Jack</th>
<th>AND</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>df=5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>count=3</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

3. Repeat until (1) end of one list and (2) docid from other list is greater

stop!
Merging (Variable-Length) Inverted Lists

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

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>df=5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Jack AND and</th>
</tr>
</thead>
<tbody>
<tr>
<td>count=3</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>35</td>
</tr>
</tbody>
</table>

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

If docids are equal, add docid to results and increment both pointers.

If docids are not equal, increment pointer with lowest docid.

Repeat until (1) end of one list and (2) docid from other list is greater.
Merging (Variable-Length) Inverted Lists

- **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.

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
<th>Jack AND and</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>35</td>
<td>35</td>
<td>35</td>
</tr>
</tbody>
</table>

**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

- **Query:** *Jack OR and*

1. *If docids are equal, add docid to results and increment both pointers*

<table>
<thead>
<tr>
<th>Jack df=3</th>
<th>and df=5</th>
<th>Jack OR and count=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

2. *If docids are not equal, add lowest docid and increment its pointer*

3. *Repeat until end of both lists*
Merging (Variable-Length) Inverted Lists

- **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

<table>
<thead>
<tr>
<th>Jack df=3</th>
<th>and df=5</th>
<th>Jack OR and count=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>
### Merging (Variable-Length) Inverted Lists

**Query:** Jack **OR** and

1. **If docids are equal, add docid to results and increment both pointers**

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>df=5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2. **If docids are not equal, add lowest docid and increment its pointer**

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

3. **Repeat until end of both lists**
Merging (Variable-Length) Inverted Lists

- **Query:** Jack OR and

1. *If docids are equal, add docid to results and increment both pointers*

<table>
<thead>
<tr>
<th></th>
<th>Jack</th>
<th>and</th>
<th>Jack OR and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>df=5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

2. *If docids are not equal, add lowest docid and increment its pointer*

3. *Repeat until end of both lists*
Merging (Variable-Length) Inverted Lists

- **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*

<table>
<thead>
<tr>
<th>Jack</th>
<th>and</th>
<th>Jack OR and</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=3</td>
<td>df=5</td>
<td>count=5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
<td>5</td>
</tr>
</tbody>
</table>

• This table shows the merging process for two inverted lists, where the “Jack” and “and” columns represent the first list, and the “Jack OR and” column represents the merged list. The `df` values indicate document frequency, and the `count` value is the final count for the merged list.
Merging (Variable-Length) Inverted Lists

• Query: Jack OR and

1. If docids are equal, add docid to results and increment both pointers

\[
\begin{array}{ccc|c}
\text{Jack} & \text{and} & \text{Jack} & \text{OR} & \text{and} \\
\text{df} = 3 & \text{df} = 5 & \text{count} = 5 \\
1 & 1 & 1 \\
3 & 3 & 3 \\
5 & 4 & 4 \\
5 & 5 & 5 \\
8 & & 8 \\
\end{array}
\]

2. If docids are not equal, add lowest docid and increment its pointer

3. Repeat until end of both lists

stop!
Merging (Variable-Length) Inverted Lists

- **Query:** *Jack OR and*

1. **If docids are equal, add docid to results and increment both pointers**

<table>
<thead>
<tr>
<th>Jack df=3</th>
<th>and df=5</th>
<th>Jack OR and count=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

2. **If docids are not equal, add lowest docid and increment its pointer**

   |   | 3          | 3                  | 3                   |
   |   | 5          | 4                  | 4                   |
   |   | 5          | 5                  | 5                   |
   |   | 8          |                    | 8                   |

3. **Repeat until end of both lists**

- **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 probably the least effective order of operations?
Merging (Variable-Length) Inverted Lists

- Which is probably the most effective order of operations?

<table>
<thead>
<tr>
<th></th>
<th>president</th>
<th>abraham</th>
<th>lincoln</th>
</tr>
</thead>
<tbody>
<tr>
<td>df</td>
<td>df=302</td>
<td>df=45</td>
<td>df=5</td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td>xx</td>
<td>xx</td>
</tr>
<tr>
<td></td>
<td>::</td>
<td>::</td>
<td></td>
</tr>
<tr>
<td></td>
<td>xx</td>
<td>xx</td>
<td></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>University</th>
<th>North</th>
<th>Carolina</th>
<th>UNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=6</td>
<td>df=4</td>
<td>df=3</td>
<td>df=5</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 5</td>
<td>10, 5</td>
<td>10, 1</td>
</tr>
<tr>
<td>15, 2</td>
<td>16, 1</td>
<td>16, 1</td>
<td>16, 4</td>
</tr>
<tr>
<td>16, 1</td>
<td>68, 1</td>
<td></td>
<td>33, 2</td>
</tr>
<tr>
<td>33, 5</td>
<td></td>
<td></td>
<td>56, 10</td>
</tr>
<tr>
<td>67, 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- `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:** \((\text{University AND North AND Carolina}) \text{ OR UNC}\)

<table>
<thead>
<tr>
<th>University</th>
<th>North</th>
<th>Carolina</th>
<th>UNC</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=6</td>
<td>df=4</td>
<td>df=3</td>
<td>df=5</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 5</td>
<td>10, 5</td>
<td>10, 1</td>
</tr>
<tr>
<td>15, 2</td>
<td>16, 1</td>
<td>16, 1</td>
<td>16, 4</td>
</tr>
<tr>
<td>16, 1</td>
<td>68, 1</td>
<td></td>
<td>33, 2</td>
</tr>
<tr>
<td>33, 5</td>
<td></td>
<td></td>
<td>56, 10</td>
</tr>
<tr>
<td>68, 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **AND** ➔ min
- **OR** ➔ sum
Retrieval Model 2: Ranked Boolean

• **Query:** \((University\ AND\ North\ AND\ Carolina)\ OR\ UNC\)

<table>
<thead>
<tr>
<th>University</th>
<th>North</th>
<th>Carolina</th>
<th>Result_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=6</td>
<td>df=4</td>
<td>df=3</td>
<td>count=??</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td></td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 5</td>
<td>10, 5</td>
<td></td>
</tr>
<tr>
<td>15, 2</td>
<td>16, 1</td>
<td>16, 1</td>
<td></td>
</tr>
<tr>
<td>16, 1</td>
<td>68, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33, 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68, 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **AND** ➔ **min**

• **OR** ➔ **sum**
Retrieval Model 2: Ranked Boolean

- **Query:** \((\text{University AND North AND Carolina}) \lor \text{UNC}\)

<table>
<thead>
<tr>
<th>University</th>
<th>North</th>
<th>Carolina</th>
<th>Result_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=6</td>
<td>df=4</td>
<td>df=3</td>
<td>count=3</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 5</td>
<td>10, 5</td>
<td>10, 1</td>
</tr>
<tr>
<td>15, 2</td>
<td>16, 1</td>
<td>16, 1</td>
<td>16, 1</td>
</tr>
<tr>
<td>16, 1</td>
<td>68, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33, 5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68, 7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **AND** $\rightarrow$ min

- **OR** $\rightarrow$ sum
Retrieval Model 2: Ranked Boolean

- **Query:** \((\text{University AND North AND Carolina}) \text{ OR UNC}\)

<table>
<thead>
<tr>
<th>Result_1</th>
<th>UNC</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>count=3</td>
<td>df=5</td>
<td>count=??</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td></td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 1</td>
<td></td>
</tr>
<tr>
<td>16, 1</td>
<td>16, 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33, 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>56, 10</td>
<td></td>
</tr>
</tbody>
</table>

- **AND** → min
- **OR** → sum
Retrieval Model 2: Ranked Boolean

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

<table>
<thead>
<tr>
<th>Result_1</th>
<th>UNC</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>count=3</td>
<td>df=5</td>
<td>count=5</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 8</td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 1</td>
<td>10, 2</td>
</tr>
<tr>
<td>16, 4</td>
<td>16, 4</td>
<td>16, 5</td>
</tr>
<tr>
<td>33, 2</td>
<td>33, 2</td>
<td></td>
</tr>
<tr>
<td>56, 10</td>
<td>56, 10</td>
<td></td>
</tr>
</tbody>
</table>

- **AND** $\rightarrow$ **min**
- **OR** $\rightarrow$ **sum**
## Retrieval Model 2: Ranked Boolean

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

<table>
<thead>
<tr>
<th>University</th>
<th>North</th>
<th>Carolina</th>
<th>UNC</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=6</td>
<td>df=4</td>
<td>df=3</td>
<td>df=5</td>
<td>count=5</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 8</td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 5</td>
<td>10, 5</td>
<td>10, 1</td>
<td>10, 2</td>
</tr>
<tr>
<td>15, 2</td>
<td>16, 1</td>
<td>16, 1</td>
<td>16, 4</td>
<td>16, 5</td>
</tr>
<tr>
<td>16, 1</td>
<td>68, 1</td>
<td></td>
<td>33, 2</td>
<td>33, 2</td>
</tr>
<tr>
<td>33, 5</td>
<td></td>
<td></td>
<td>56, 10</td>
<td>56, 10</td>
</tr>
<tr>
<td>68, 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Conceptually, what do these document scores indicate?
# Retrieval Model 2: Ranked Boolean

- **Query:** \((\text{University AND North AND Carolina}) \text{ OR } \text{UNC}\)

<table>
<thead>
<tr>
<th>University</th>
<th>North</th>
<th>Carolina</th>
<th>UNC</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>df=6</td>
<td>df=4</td>
<td>df=3</td>
<td>df=5</td>
<td>count=5</td>
</tr>
<tr>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 4</td>
<td>1, 8</td>
</tr>
<tr>
<td>10, 1</td>
<td>10, 5</td>
<td>10, 5</td>
<td>10, 1</td>
<td>10, 2</td>
</tr>
<tr>
<td>15, 2</td>
<td>16, 1</td>
<td>16, 1</td>
<td>16, 4</td>
<td>16, 5</td>
</tr>
<tr>
<td>16, 1</td>
<td>68, 1</td>
<td>33, 2</td>
<td>56, 10</td>
<td>56, 10</td>
</tr>
<tr>
<td>33, 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>68, 7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 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 knows what to look for
  - Difficult to balance precision and recall
Summary

1. Document representation
2. Information need
3. Evaluation
4. Comparison

- Query
- Retrieved objects
Summary

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