Introduction to Information Retrieval
http://informationretrieval.org

IIR 1: Boolean Retrieval

Hinrich Schütze

Institute for Natural Language Processing, University of Stuttgart

2011-08-29
Models and Methods

1. Boolean model and its limitations (30)
2. Vector space model (30)
3. Probabilistic models (30)
4. Language model-based retrieval (30)
5. Latent semantic indexing (30)
6. Learning to rank (30)
Models and Methods

1. Boolean model and its limitations (30)
2. Vector space model (30)
3. Probabilistic models (30)
4. Language model-based retrieval (30)
5. Latent semantic indexing (30)
6. Learning to rank (30)
Take-away
Take-away

- **Boolean model and Inverted index**: The Boolean model and the basic data structure of most IR systems
Take-away

- **Boolean model and Inverted index**: The Boolean model and the basic data structure of most IR systems
- **Processing Boolean queries**
Take-away

- **Boolean model and Inverted index**: The Boolean model and the basic data structure of most IR systems
- **Processing Boolean queries**
- **Why is Boolean retrieval not enough?** or Why do we need ranked retrieval?
Outline

1. Boolean model and Inverted index
2. Processing Boolean queries
3. Why ranked retrieval?
Definition of information retrieval

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
Definition of *information retrieval*

Information retrieval (IR) is *finding* material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
Definition of *information retrieval*

Information retrieval (IR) is finding material *(usually documents)* of an unstructured nature *(usually text)* that satisfies an information need from within large collections *(usually stored on computers)*.
Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an *unstructured* nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an *information need* from within large collections (usually stored on computers).
Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
Definition of *information retrieval*

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

**The adhoc retrieval problem:** Given a user information need and a collection of documents, the IR system determines how well the documents satisfy the query and returns a subset of relevant documents to the user.
Boolean retrieval
The Boolean model is arguably the simplest model to base an information retrieval system on.
The Boolean model is arguably the simplest model to base an information retrieval system on.

Queries are Boolean expressions, e.g., **CAESAR AND BRUTUS**
The Boolean model is arguably the simplest model to base an information retrieval system on.

Queries are Boolean expressions, e.g., *CAESAR AND BRUTUS*

The search engine returns all documents that satisfy the Boolean expression.
Model collection: The works of Shakespeare
Model collection: The works of Shakespeare
Model collection: The works of Shakespeare
Model collection: The works of Shakespeare

Each of Shakespeare’s tragedies, comedies etc is a document in this collection.
Term-document incidence matrix
Term-document incidence matrix

<table>
<thead>
<tr>
<th>Anthony and Cleopatra</th>
<th>Anthony</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ANTHONY</strong></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td><strong>BRUTUS</strong></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>CAESAR</strong></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>CALPURNIA</strong></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>CLEOPATRA</strong></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>MERCY</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>WORSER</strong></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Entry is 1 if term occurs. Example: CALPURNIA occurs in *Julius Caesar*. Entry is 0 if term doesn’t occur. Example: CALPURNIA doesn’t occur in *The tempest*. 
## Term-document incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony and Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cleopatra</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anthony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Worse</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Entry is 1 if term occurs. Example: **Calpurnia** occurs in *Julius Caesar*. Entry is 0 if term doesn’t occur. Example: **Calpurnia** doesn’t occur in *The Tempest*. 
### Term-document incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony and Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Worse</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Entry is 1 if term occurs. Example: **Calpurnia** occurs in *Julius Caesar*.

Entry is 0 if term doesn’t occur. Example: **Calpurnia** doesn’t occur in *The Tempest*. 
## Term-document incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>Anthony and Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Worser</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Entry is 1 if term occurs. Example: Calpurnia occurs in *Julius Caesar*. Entry is 0 if term doesn’t occur. Example: Calpurnia doesn’t occur in *The Tempest*.

We will return to this matrix many times in this class.
We can’t build the incidence matrix for large collections
We can’t build the incidence matrix for large collections

- Size of incidence matrix: number of documents times number of terms → too large for large collections
We can’t build the incidence matrix for large collections

- Size of incidence matrix: number of documents times number of terms → too large for large collections
- But the matrix is very sparse – mostly 0s, few 1s.
We can’t build the incidence matrix for large collections

- Size of incidence matrix: number of documents times number of terms → too large for large collections
- But the matrix is very sparse – mostly 0s, few 1s.
- Inverted index: We only record the 1s.
Inverted Index
Inverted Index

For each term $t$, we store a list of all documents that contain $t$.

$\Rightarrow$ For each term $t$, we store the 1s in its row in the incidence matrix

<table>
<thead>
<tr>
<th>BRUTUS</th>
<th>$\rightarrow$</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>11</th>
<th>31</th>
<th>45</th>
<th>173</th>
<th>174</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAESAR</td>
<td>$\rightarrow$</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>16</td>
<td>57</td>
<td>132</td>
</tr>
<tr>
<td>CALPURNIA</td>
<td>$\rightarrow$</td>
<td>2</td>
<td>31</td>
<td>54</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[\ldots\]\n
dictionary

\[\ldots\]\npostings
Outline

1. Boolean model and Inverted index
2. Processing Boolean queries
3. Why ranked retrieval?
Simple conjunctive query (two terms)
Simple conjunctive query (two terms)

Consider the query: **Brutus AND Calpurnia**
Simple conjunctive query (two terms)

- Consider the query: **Brutus AND Calpurnia**
- To find all matching documents using inverted index:
Simple conjunctive query (two terms)

- Consider the query: **Brutus AND Calpurnia**
- To find all matching documents using inverted index:
  1. Locate **Brutus** in the dictionary
Simple conjunctive query (two terms)

Consider the query: **Brutus AND Calpurnia**

- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file
Simple conjunctive query (two terms)

- Consider the query: **Brutus AND Calpurnia**
- To find all matching documents using inverted index:
  1. Locate **Brutus** in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate **Calpurnia** in the dictionary
Consider the query: **Brutus AND Calpurnia**

To find all matching documents using inverted index:

1. Locate **Brutus** in the dictionary
2. Retrieve its postings list from the postings file
3. Locate **Calpurnia** in the dictionary
4. Retrieve its postings list from the postings file
Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate Calpurnia in the dictionary
  4. Retrieve its postings list from the postings file
  5. Intersect the two postings lists
Consider the query: **BRUTUS AND CALPURNIA**

To find all matching documents using inverted index:
1. Locate **BRUTUS** in the dictionary
2. Retrieve its postings list from the postings file
3. Locate **CALPURNIA** in the dictionary
4. Retrieve its postings list from the postings file
5. Intersect the two postings lists
6. Return intersection to user
Intersecting two postings lists

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

**Calpurnia** → 2 → 31 → 54 → 101

Intersection →
Intersecting two postings lists

**BRUTUS** → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

**CALPURNIA** → 2 → 31 → 54 → 101

Intersection →
Intersecting two postings lists

**Brutus**  →  \[1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174\]

**Calpurnia**  →  \[2 \rightarrow 31 \rightarrow 54 \rightarrow 101\]

Intersection  →

Schütze: Boolean retrieval 14 / 30
Intersecting two postings lists

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

**Calpurnia** → 2 → 31 → 54 → 101

Intersection → 2
Intersecting two postings lists

\[ \text{Brutus} \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174 \]

\[ \text{Calpurnia} \rightarrow 2 \rightarrow 31 \rightarrow 54 \rightarrow 101 \]

\[ \text{Intersection} \rightarrow 2 \]
Intersecting two postings lists

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

**Calpurnia** → 2 → 31 → 54 → 101

**Intersection** → 2
Intersecting two postings lists

Brutus: \[1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174\]

Calpurnia: \[2 \rightarrow 31 \rightarrow 54 \rightarrow 101\]

Intersection: \[2\]
Intersecting two postings lists

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

**Calpurnia** → 2 → 31 → 54 → 101

**Intersection** → 2 → 31
Intersecting two postings lists

**Brutus**

$$\rightarrow \ 1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174$$

**Calpurnia**

$$\rightarrow \ 2 \rightarrow 31 \rightarrow 54 \rightarrow 101$$

**Intersection**

$$\Rightarrow \ 2 \rightarrow 31$$
Intersecting two postings lists

Brutus → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174
Calpurnia → 2 → 31 → 54 → 101
Intersection → 2 → 31
Intersecting two postings lists

Brutus → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174
Calpurnia → 2 → 31 → 54 → 101
Intersection → 2 → 31
Intersecting two postings lists

**BRUTUS**  →  1 → 2 → 4 → 11 → 31 → 45 → 173 → 174

**CALPURNIA**  →  2 → 31 → 54 → 101

**Intersection**  →  2 → 31
Intersecting two postings lists

\[ \text{Brutus} \longrightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 11 \rightarrow 31 \rightarrow 45 \rightarrow 173 \rightarrow 174 \]

\[ \text{Calpurnia} \longrightarrow 2 \rightarrow 31 \rightarrow 54 \rightarrow 101 \]

Intersection \[ \longrightarrow 2 \rightarrow 31 \]

- This is linear in the length of the postings lists.
The example was a simple conjunctive query . . .
The example was a simple conjunctive query . . .

. . . the Boolean retrieval model can answer any query that is a Boolean expression.
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use **AND**, **OR** and **NOT** to join query terms.
  - Views each document as a **set** of terms.
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
- Primary commercial retrieval tool for 3 decades
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
- Primary commercial retrieval tool for 3 decades
- Many professional searchers (e.g., lawyers) still like Boolean queries.
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.

- Primary commercial retrieval tool for 3 decades
- Many professional searchers (e.g., lawyers) still like Boolean queries.
  - You know exactly what you are getting.
Boolean queries

- The example was a simple conjunctive query . . .
- . . . the Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
- Primary commercial retrieval tool for 3 decades
- Many professional searchers (e.g., lawyers) still like Boolean queries.
  - You know exactly what you are getting.
- Many search systems you use are also Boolean: search system on your laptop, in your email reader, on the intranet etc
The example was a simple conjunctive query . . .

. . . the Boolean retrieval model can answer any query that is a Boolean expression.

- Boolean queries are queries that use AND, OR, and NOT to join query terms.
- Views each document as a set of terms.
- Is precise: Document matches condition or not.

Primary commercial retrieval tool for 3 decades

Many professional searchers (e.g., lawyers) still like Boolean queries.

- You know exactly what you are getting.

Many search systems you use are also Boolean: search system on your laptop, in your email reader, on the intranet etc.

So are we done?
Outline

1. Boolean model and Inverted index
2. Processing Boolean queries
3. Why ranked retrieval?
The Boolean model: Pros and Cons
The Boolean model: Pros and Cons

- Key property: Documents either match or don’t.
The Boolean model: Pros and Cons

- Key property: Documents either match or don’t.
- **Good for expert users** with precise understanding of their needs and of the collection.
The Boolean model: Pros and Cons

- Key property: Documents either match or don’t.
- Good for expert users with precise understanding of their needs and of the collection.
- Also good for applications: Applications can easily consume 1000s of results.
The Boolean model: Pros and Cons

- Key property: Documents either match or don’t.
- **Good for expert users** with precise understanding of their needs and of the collection.
- Also **good for applications**: Applications can easily consume 1000s of results.
- Not good for the majority of users
The Boolean model: Pros and Cons

- **Key property:** Documents either match or don’t.
- **Good for expert users** with precise understanding of their needs and of the collection.
- **Also good for applications:** Applications can easily consume 1000s of results.
- **Not good for the majority of users**
- Most users are not capable of writing Boolean queries . . .
The Boolean model: Pros and Cons

- **Key property:** Documents either match or don’t.
- **Good for expert users** with precise understanding of their needs and of the collection.
- **Also good for applications:** Applications can easily consume 1000s of results.
- **Not good for the majority of users**
- **Most users are not capable of writing Boolean queries . . .**
  - . . . or they are, but they think it’s too much work.
The Boolean model: Pros and Cons

- **Key property:** Documents either match or don’t.
- **Good for expert users** with precise understanding of their needs and of the collection.
- **Also good for applications:** Applications can easily consume 1000s of results.
- **Not good for the majority of users**
- Most users are not capable of writing Boolean queries . . . 
  - . . . or they are, but they think it’s too much work.
- Most users don’t want to wade through 1000s of results.
The Boolean model: Pros and Cons

- Key property: Documents either match or don’t.
- **Good for expert users** with precise understanding of their needs and of the collection.
- Also **good for applications**: Applications can easily consume 1000s of results.
- **Not good for the majority of users**
  - Most users are not capable of writing Boolean queries . . .
  - . . . or they are, but they think it’s too much work.
- Most users don’t want to wade through 1000s of results.
- This is particularly true of web search.
Problem with Boolean search: Feast or famine
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (\(\leq 0\)) or too many (\(1000\text{s}\)) results.
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (=0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650]
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (≈0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650]
  → 200,000 hits – feast
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (≈0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650] → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found]
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650]
  → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found]
  → 0 hits – famine
Problem with Boolean search: Feast or famine

- Boolean queries often result in either too few (≈0) or too many (1000s) results.
- Query 1 (boolean conjunction): [standard user dlink 650] → 200,000 hits – feast
- Query 2 (boolean conjunction): [standard user dlink 650 no card found] → 0 hits – famine
- In Boolean retrieval, it takes a lot of skill to come up with a query that produces a manageable number of hits.
Feast or famine: No problem in ranked retrieval
Feast or famine: No problem in ranked retrieval

- With ranking, large result sets are not an issue.
Feast or famine: No problem in ranked retrieval

- With ranking, large result sets are not an issue.
- Just show the top 10 results and the user won’t be overwhelmed
Feast or famine: No problem in ranked retrieval

- With ranking, large result sets are not an issue.
- Just show the top 10 results and the user won’t be overwhelmed.
- Premise: the ranking algorithm works: More relevant results are ranked higher than less relevant results.
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
  - Interview them
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
  - Interview them
  - Eye-track them
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
  - Interview them
  - Eye-track them
  - Time them
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
  - Interview them
  - Eye-track them
  - Time them
  - Record and count their clicks
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
  - Interview them
  - Eye-track them
  - Time them
  - Record and count their clicks
- The following slides are from Dan Russell’s 2007 JCDL talk
Empirical investigation of the effect of ranking

- How can we measure how important ranking is?
- Observe what searchers do when they are searching in a controlled setting
  - Videotape them
  - Ask them to “think aloud”
  - Interview them
  - Eye-track them
  - Time them
  - Record and count their clicks

- The following slides are from Dan Russell’s 2007 JCDL talk
- Dan Russell was at the “Über Tech Lead for Search Quality & User Happiness” at Google.
So, did you notice the FTD official site?

To be honest, I didn’t even look at that. At first I saw “from $20” and $20 is what I was looking for. To be honest, 1-800-flowers is what I’m familiar with and why I went there next even though I kind of assumed they wouldn’t have $20 flowers.

And you knew they were expensive?

I knew they were expensive but I thought “hey, maybe they’ve got some flowers for under $20 here…”

But you didn’t notice the FTD?

No I didn’t, actually… that’s really funny.
Rapidly scanning the results

Note scan pattern:

Page 3: 
Result 1  
Result 2  
Result 3  
Result 4  
Result 3  
Result 2  
Result 4  
Result 5  
Result 6 <click>

Q: Why do this? 
A: What’s learned later influences judgment of earlier content.
How many links do users view?

Total number of abstracts viewed per page

Mean: 3.07   Median/Mode: 2.00
Looking vs. Clicking

- Users view results one and two more often / thoroughly
- Users click most frequently on result one
Presentation bias – reversed results

- Order of presentation influences where users look AND where they click

![Bar chart showing the probability of click for normal and swapped conditions]
Importance of ranking: Summary
Importance of ranking: Summary

- **Viewing abstracts**: Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).
Importance of ranking: Summary

- **Viewing abstracts:** Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).

- **Clicking:** Distribution is even more skewed for clicking
Importance of ranking: Summary

- **Viewing abstracts**: Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).
- **Clicking**: Distribution is even more skewed for clicking.
- There is a very strong bias to click on the top-ranked page.
Importance of ranking: Summary

- **Viewing abstracts:** Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).

- **Clicking:** Distribution is even more skewed for clicking.

- There is a very strong bias to click on the top-ranked page.

- Even if the top-ranked page is not relevant, 30% of users will click on it.
Importance of ranking: Summary

- **Viewing abstracts:** Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).

- **Clicking:** Distribution is even more skewed for clicking
  - There is a very strong bias to click on the top-ranked page.
  - Even if the top-ranked page is not relevant, 30% of users will click on it.

→ Getting the ranking right is very important.
Importance of ranking: Summary

- **Viewing abstracts**: Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).
- **Clicking**: Distribution is even more skewed for clicking.
  - There is a very strong bias to click on the top-ranked page.
  - Even if the top-ranked page is not relevant, 30% of users will click on it.

→ Getting the ranking right is very important.
→ Getting the top-ranked page right is most important.
Importance of ranking: Summary

- **Viewing abstracts:** Users are a lot more likely to read the abstracts of the top-ranked pages (1, 2, 3, 4) than the abstracts of the lower ranked pages (7, 8, 9, 10).

- **Clicking:** Distribution is even more skewed for clicking.
  - There is a very strong bias to click on the top-ranked page.
  - Even if the top-ranked page is not relevant, 30% of users will click on it.

→ **Getting the ranking right is very important.**

→ **Getting the top-ranked page right is most important.**
Take-away

- **Boolean model and Inverted index**: The Boolean model and the basic data structure of most IR systems
- **Processing Boolean queries**
- Why is Boolean retrieval not enough? or *Why do we need ranked retrieval?*
Resources

- Chapter 1 of Introduction to Information Retrieval
- Resources at http://informationretrieval.org/essir2011
  - List of useful information retrieval resources
  - Shakespeare search engine
  - Daniel Russell’s home page
Exercise
Exercise

Does Bing/Google use the Boolean model?
Exercise

Does Bing/Google use the Boolean model?
Does Spotlight use the Boolean model?
Does web search engines use the Boolean model?
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \land w_2 \ldots \land w_n]\) is \(w_1 \land w_2 \land \ldots \land w_n\)
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \land w_2 \land \ldots \land w_n]\) is \(w_1 \land w_2 \land \ldots \land w_n\)

- Cases where you get hits that do not contain one of the \(w_i\):
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \land w_2 \ldots \land w_n]\) is \(w_1 \land w_2 \land \ldots \land w_n\)
- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \land w_2 \land \ldots \land w_n]\) is \(w_1 \land w_2 \land \ldots \land w_n\)
- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
  - page contains variant of \(w_i\) (morphology, spelling correction, synonym)
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \ AND \ w_2 \ AND \ldots \ AND \ w_n]\)

- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
  - page contains variant of \(w_i\) (morphology, spelling correction, synonym)
  - long queries \((n\ \text{large})\)
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 w_2 \ldots w_n]\) is \(w_1 \text{ AND } w_2 \text{ AND } \ldots \text{ AND } w_n\)
- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
  - page contains variant of \(w_i\) (morphology, spelling correction, synonym)
  - long queries (\(n\) large)
  - conjunctive boolean query generates very few hits
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \ AND \ w_2 \ AND \ldots \ AND \ w_n]\)
- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
  - page contains variant of \(w_i\) (morphology, spelling correction, synonym)
  - long queries (\(n\) large)
  - conjunctive boolean query generates very few hits

- Simple Boolean vs. Ranking of result set
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1, w_2, \ldots, w_n]\) is \(w_1 \text{ AND } w_2 \text{ AND } \ldots \text{ AND } w_n\)

- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
  - page contains variant of \(w_i\) (morphology, spelling correction, synonym)
  - long queries (\(n\) large)
  - conjunctive boolean query generates very few hits

- Simple Boolean vs. Ranking of result set
  - Simple Boolean retrieval returns matching documents in no particular order.
Does web search engines use the Boolean model?

- Default interpretation of a query by web search engines: \([w_1 \land w_2 \ldots \land w_n]\) is \(w_1 \land w_2 \land \ldots \land w_n\)
- Cases where you get hits that do not contain one of the \(w_i\):
  - anchor text
  - page contains variant of \(w_i\) (morphology, spelling correction, synonym)
  - long queries (\(n\) large)
  - conjunctive boolean query generates very few hits
- Simple Boolean vs. Ranking of result set
  - Simple Boolean retrieval returns matching documents in no particular order.
  - Google (and most well designed Boolean engines) rank the result set – they rank good hits (according to some estimator of relevance) higher than bad hits.