Information Retrieval, Information Extraction, Question Answering
Overview of Methods for Information Access

Jörg Tiedemann
jorg.tiedemann@lingfil.uu.se
Department of Linguistics and Philology
Uppsala University

Information Retrieval is part of our life!

Terminology

**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 Extraction (IE)** is extracting structured information from unstructured machine-readable documents by means of natural language processing (NLP).

**Question Answering (QA)** is answering a question posed in natural language and has to deal with a wide range of question types including: fact, list, definition, how, why, hypothetical, semantically constrained, and cross-linguual questions.

What are the challenges?

▶ huge dynamic collections of diverse (multi-modal) material
▶ users expect immediate answers
▶ relevance ranking & personalization
▶ access to everything everywhere (in various languages)
Information Retrieval is more than general domain keyword search.

### Outline for today

- **Information Retrieval**
  - Boolean Retrieval
  - Ranked Retrieval
  - Web Retrieval

- **Information Extraction**

- **Question Answering**

---

**Literature**

Most information comes from this book:

*Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze: Introduction to Information Retrieval*, Cambridge University Press, 2008

Freely available at:


---

**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).*

**Questions to be answered**

- How do we match queries with documents?
- How do we measure relevance?
- What is a document?
- How can we make this really fast?
- How can we deal with HUGE diverse collections and MILLIONS of users?
Information Retrieval

Clues for relevance

- document contents (text)
- links between documents & their labels
  (hyperlinks, references, citations)
- tags & captions
- clicks & queries

→ clue importance is shifting ...

Boolean retrieval

**Boolean model:** document = set of terms

<table>
<thead>
<tr>
<th>(term document matrix)</th>
<th>Anthony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</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>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Boolean Query: **BRUTUS AND CAESAR AND NOT CALPURNIA**

- Do a (bitwise) **AND** on three vectors
- **110100 AND 110111 AND 101111 = 100100**

Problem solved?

Inverted Index

For each term \( t \), store a **sorted** list of all doc’s that contain \( t \).

- \( \text{BRUTUS} \rightarrow 1, 2, 4, 11, 31, 45, 173, 174 \)
- \( \text{CAESAR} \rightarrow 1, 2, 4, 5, 6, 16, 57, 132, \ldots \)
- \( \text{CALPURNIA} \rightarrow 2, 31, 54, 101 \)

(... and this is still a very small collection ...)

Bigger collections

- Consider \( N = 10^6 \) documents (and ca. \( 10^8 \) tokens)
- Assume there are \( M = 500,000 \) distinct terms

\( M = 500,000 \times 10^6 = \) half a trillion 0s and 1s.

→ Term-document matrix is extremely sparse!

(... and this is still a very small collection ...)

Information Retrieval Information Extraction Question Answering

Jörg Tiedemann 9/57

Jörg Tiedemann 10/57

Jörg Tiedemann 11/57

Jörg Tiedemann 12/57
Simple conjunctive query (two terms)

Query = BRUTUS AND CALPURNIA

- BRUTUS → 1 → 2 → 4 → 11 → 31 → 45 → 173 → 174
- CALPURNIA → 2 → 31 → 54 → 101

Intersection → 2 → 31

- This is linear in the length of the postings lists!
- Query optimization & skip pointers → make it even faster!
- Exercise: BRUTUS AND CAESAR AND NOT CALPURNIA

Phrase queries with Positional Indexes

- Example query: “to₁ be₂ or₃ not₄ to₅ be₆”

TO, 993427:
{ 1: (7, 18, 33, 72, 86, 231);
  2: (1, 17, 74, 222, 255);
  4: (8, 16, 190, 429, 433);
  5: (363, 367);
  7: (13, 23, 191); …}

BE, 178239:
{ 1: (17, 25);
  4: (17, 191, 291, 429, 430);
  5: (14, 19, 101); …}

→ Also allows proximity search! (“employment /4 place”)

Rapidly scanning the results

Note scan pattern:

<table>
<thead>
<tr>
<th>Page</th>
<th>Result 1</th>
<th>Result 2</th>
<th>Result 3</th>
<th>Result 4</th>
<th>Result 3</th>
<th>Result 2</th>
<th>Result 4</th>
<th>Result 5</th>
<th>Result 6</th>
<th>&lt;click&gt;</th>
</tr>
</thead>
</table>

Q: Why do this?
A: What's learned later influences judgment of earlier content.

(Slides from Dan Russell, Google)
**Looking vs. Clicking**

![Bar chart showing the number of times results are selected vs. time spent in the abstract.

- Users view results one and two more often / thoroughly.
- Users click most frequently on result one.

> Boolean retrieval is not enough! Relevance ranking is important!

---

**Bag of Words Model**

Each document is represented by a count vector $\in N^{|V|}$ (term frequency).

<table>
<thead>
<tr>
<th>Anthony</th>
<th>Julius</th>
<th>The</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anthony</td>
<td>157</td>
<td>73</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>4</td>
<td>157</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>232</td>
<td>227</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Mercy</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>Worse</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

... 

How do we rank documents according to relevance?

> Probably the best known weighting scheme in IR:

**tf-idf weighting**

- **term frequency** ($tf$) = measure of **relevance**
- **inverse document frequency** ($idf$) = measure of **informativeness** of the term

$$w_{t,d} = (1 + \log tf_{t,d}) \cdot \log \frac{N}{df_{t}}$$

- increases with
  - number of occurrences within a document
  - rarity of the term in collection

> probably the best known weighting scheme in IR
**Documents and queries as vectors**

- **d1:** Ranks of starving poets swell  
  - q: Rich, Poor  
  - d2: Engineers’ pay up by 5%  
  - d3: Record baseball salaries in 2009

→ Measure proximity between query and documents!  
→ Why not Euclidean distance?

**Cosine similarity between query and document**

\[
\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{|\vec{q}| \cdot |\vec{d}|} = \sum_{i=1}^{V} \frac{q_i}{\sqrt{\sum_{i=1}^{V} q_i^2}} \cdot \frac{d_i}{\sqrt{\sum_{i=1}^{V} d_i^2}}
\]

- \(q_i\) is the tf-idf weight of term \(i\) in the query.  
- \(d_i\) is the tf-idf weight of term \(i\) in the document.  
- \(|\vec{q}|\) and \(|\vec{d}|\) are the lengths of \(\vec{q}\) and \(\vec{d}\).  
- \(\vec{q}/|\vec{q}|\) and \(\vec{d}/|\vec{d}|\) are length-1 vectors (= normalized).

**Time for Questions & Short Break**

To sum up ....

- IR = searching relevant information in unstructured data collections  
- Boolean model: find exact matches  
- Inverted index: important data structure for IR  
- Ranking by relevance is important!  
- Vector-space model, tf-idf & cosine similarity
Part II

- Tweaking performance for ranked retrieval
- Web-based information retrieval
- Link Analysis

Components of various tf-idf weighting schemes

<table>
<thead>
<tr>
<th>Term frequency</th>
<th>Document frequency</th>
<th>Normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>n (natural)</td>
<td>n (no)</td>
<td>n (none)</td>
</tr>
<tr>
<td>l (logarithm)</td>
<td>t (idf)</td>
<td>t (cosine)</td>
</tr>
<tr>
<td>a (augmented)</td>
<td>p (prob idf)</td>
<td>u (pivoted unique)</td>
</tr>
<tr>
<td>b (boolean)</td>
<td>1 if ( t_{i,d} &gt; 0 )</td>
<td>( 1/\text{CharLength}^\alpha ), ( \alpha &lt; 1 )</td>
</tr>
<tr>
<td>L (log ave)</td>
<td>( 1/\log(\text{CharLength}) )</td>
<td>( 1/\log(\text{CharLength}) )</td>
</tr>
</tbody>
</table>

- Different weightings for queries & documents → qqq.ddd
- Example: ltn.lnc:
  - Query: logarithmic tf, idf, no normalization
  - Document: logarithmic tf, no df weighting, cosine normalization

Efficient scoring

Comparing each query with ALL documents is too expensive!

- Use inverted index! (accumulate scores for doc's in postings)
- Store term frequencies in postings (space-efficient)!
- Store document frequencies in dictionary!

Don’t need full ranking! → efficient ranking with binary min heap!
### Binary min heap for selecting top $k$

- **binary min heap** = binary tree in which each node’s value is less than the values of its children

- always keep top $k$ documents seen so far
- takes $O(J \log k)$ operations to construct
- read off $k$ winners in $O(k \log k)$ steps

### Inexact top $k$ retrieval

- Cosine comparison is still expensive!

  - Goal: find a set $A$ of “contenders” with $k < |A| << N$
  - return top $k$ documents in $A$

  - Task: selecting appropriate $A = \text{pruning}$ non-contenders
    - index elimination
    - champion lists
    - static quality scores
    - impact ordering
    - cluster pruning

### Index elimination & Champion lists

- only consider high-idf query terms
  - query: **catcher in the rye**
  - accumulate scores for **catcher** and **rye** only

- only consider doc’s containing many query terms
  - for example, fixed proportion (3 out of 4)

- Champion lists:
  - For each term in dictionary: precompute $r$ documents of heighest weight (in the postings of the term)
  - query: only compute scores for documents in union of champion lists for all query terms
  - fixed $r$ or depending on term value (e.g. idf-based)

### Static quality scores

- Reorder posting lists according to “expected relevance”

  - query independent quality $g(d)$ of documents (**authority**)
    - a paper with many citations
    - many bookmarks (del.icio.us, ...)
    - PageRank

  - rank according to net-score:
    \[
    \text{net-score}(q, d) = g(d) + \cos(q, d)
    \]

  Why does this help with efficiency?
Static quality scores

How does this help?

- postings are ordered by \( g(d) \) – still consistent order!
- traverse postings and compute scores as usual
- early termination is possible
  - stop if minimal score cannot be improved
  - time threshold
  - threshold for goodness score
- can be combined with champion lists

Web search Engines

Web search engines must crawl their documents!

- initialize with known seed pages (URLs)
- fetch & parse pages from the URLs in the queue
- extract URLs from those pages & add them to the queue

Efficient crawling needs to address several things:

- need to distribute to scale-up, prioritize
- need (near-)duplicate detection, spam detection
- web politeness, crawler strategies
- need robustness (spider traps, large sites, dynamic pages)
- need to re-crawl periodically (some sites more often)

Link Analysis

- A query “IBM” might not match the contents of the IBM homepage!
- there are many other pages linking to IBM’s homepage

  page A \( \text{anchor text} \rightarrow \) page B

  → hyperlink is a quality signal
  → anchor text (or local context) describes contents of page B

  Index anchor texts & link analyses

  → PageRank

  - model: likelihood that a random surfer arrives at page B
  - markov chain model: web = probabilistic directed connected graph
  - web-page = state, \( N \times N \) probability transition matrix (links)
  - PageRank = long-term visit rate = steady-state probability
### Information Retrieval

- Information Extraction
- Question Answering

### Link Analysis - Example Graph

#### Link Matrix

<table>
<thead>
<tr>
<th></th>
<th>$d_0$</th>
<th>$d_1$</th>
<th>$d_2$</th>
<th>$d_3$</th>
<th>$d_4$</th>
<th>$d_5$</th>
<th>$d_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_0$</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$d_1$</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$d_2$</td>
<td>0.33</td>
<td>0.00</td>
<td>0.33</td>
<td>0.33</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$d_3$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>$d_4$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$d_5$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>$d_6$</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.33</td>
<td>0.33</td>
<td>0.00</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Avoid dead ends → add teleportation rate (e.g. 14% chance to jump to any random page)

### Link Analysis - PageRank

#### Power Method to find $\mathbf{x} = \mathbf{\pi} P$

1. start with random distribution $\mathbf{x}_0 = \mathbf{(x}_{0,1}, \mathbf{x}_{0,2}, \ldots, \mathbf{x}_{0,N})$
2. at step $t$ compute $\mathbf{x}_t = \mathbf{x}_{t-1} P$:
   $$x_{t,k} = \sum_{i=1}^{N} x_{t-1,i} P_{i,k}$$

3. continue with 2. until convergence ($\mathbf{\pi} = \mathbf{x}_m = \mathbf{x}_0 P^m$)
Link Analysis

- PageRank is behind Google’s early success (Larry Page)
- PageRanks are query-independent (pre-computed)
- re-rank retrieved documents by their PageRank
  (together with text match, anchor text match, ...)

Drawback: query-independent scores for all pages

→ Hypertext Induced Topic Search (HITS)
  - retrieve relevant pages
  - page rankings for hubs & authorities
  - problems: on-the-fly computation, topic drift, easy to spam

Time for Questions & Short Break

To sum up part II ....

- several weighting schemes for vector-space models
- data structures for efficient retrieval
- web search & link analysis
- PageRank & HITS

Part III

- Some words about building practical IR engines
- Information Extraction
- Question Answering

Other practical issues when building IR systems

- Tiered indexes
  - cascaded query processing
  - index with most important terms & doc’s first
- Zones
  - different indexes for various parts of a doc (title, body ...)
- Query term proximity
  - more relevant: keywords in close proximity to each other
- Query parsing
  - check syntax
  - create actual index queries
  - combine results

→ All parts need careful tuning! (→ Evaluation is important!)
Information Extraction (IE) is extracting structured information from unstructured machine-readable documents by means of natural language processing (NLP).

Motivation:
- unstructured diverse data collections are full of information
- extract & store world knowledge from those collections
- structured collections (databases) for many purposes
  - searchable fact databases
  - question answering
→ for example, turn Wikipedia into a well-structured fact-DB

Sub-tasks:
- named entity recognition
- coreference resolution
- terminology extraction
- relationship extraction

Find patterns like
- PERSON works for ORGANIZATION
- PERSON lives in LOCATION

Precision is usually more important than Recall!

IE often requires heavy NLP and semantic inference

- extract dates of people’s death
  - ...and when on September 18, 1970, Jimi Hendrix died of an overdose, her reaction...
- extract capitals of countries in the world
  - The riders had a tour of Moscow this morning. Tomorrow morning they are leaving the Russian capital..

Simple cases exist (e.g. infoboxes at Wikipedia)
**Question Answering**

*Question Answering (QA)* is answering a question posed in natural language and has to deal with a wide range of question types including: *fact, list, definition, how, why, hypothetical, semantically constrained, and cross-lingual questions.*

**Motivation:**

- retrieve specific information
- do it in a more natural (human) way
- possibly integrate this in a dialogue system
  - follow-up questions (with coreferences)
  - interactive topic specific dialogues
  - multi-modal, speech recognition & synthesis

→ Very ambitious!

---

**Question Answering Systems (Example: Joost)**

![Diagram of Joost system]

1. **IR**
2. **IE**
3. **Relation tables**
4. **Answer Extraction & Selection**
5. **Parijs**

---

**Question Answering: Why is this difficult?**

- When was the unification of Germany?
  - *Already in 1961* he predicted the unification of Germany.

- What is the capital of Ireland?
  - *It is a common joke in Ireland to call Cork the “real capital of Ireland”*
  - *In the middle ages Kilkenny was the capital of Ireland*

- What is RSI?
  - Website with “Common misconceptions about RSI” .... *RSI is the same as a mouse arm.*

---

**Question Answering: How good is it?**

- acceptable for simple factoid questions
- off-line IE is very effective
- still quite bad for more complex cases
- too slow due to heavy NLP

→ needs still a lot of development
→ Google starts integrating some QA functionalities
Question Answering with Google

How did Jimi Hendrix die?

Jimi Hendrix — Cause of Death: Barbiturate Overdose
According to http://www.celebritywonder.com/hitlist/jimmihendrix.html - More sources -

WikiAnswers - How did Jimi Hendrix die
Jimi Hendrix question: How did Jimi Hendrix die? Jimi Hendrix overdosed on sleep aids and drank too much red wine on September 18th, 1970, and checked on ...
wiki.answers.com/Q/How_did_Jimi_Hendrix_die - Cached - Similar

How many people live in Berlin

WikiAnswers - How many people live in Berlin
Germany question: How many people live in Berlin? Berlin proper - that is the (Bundesland) of Berlin - has a population of just over 3.4 million
wiki.answers.com/Q/How_many_people_live_in_Berlin - Cached - Similar

Question Answering vs. IR

Information Retrieval
- input: keywords (+ boolean operators, ...)
- output: links to relevant documents (maybe snippets)
- techniques: vector-space model, bag-of-words, tf-idf, PageRank

→ IR is just a component of question answering!
→ QA → more NLP → more fun (?)!

Question Answering
- input: natural language question
- output: concrete answer (facts or text)
- techniques: shallow/deep NLP, passage retrieval (IR), information extraction

Summary

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).

→ effective shallow techniques — in daily use

Information Extraction (IE) is extracting structured information from unstructured machine-readable documents by means of natural language processing (NLP).

→ (mostly shallow) NLP — used for data mining

Question Answering (QA) is answering a question posed in natural language and has to deal with a wide range of question types including: fact, list, definition, how, why, hypothetical, semantically constrained, and cross-lingual questions.

→ heavy NLP — still more like a research area
What we didn’t talk about ...

- measures for evaluation (very important!)
- index construction
  - pre-processing (conversion, tokenization, lemmatization ...)
  - distributed indexing, dynamic indexing
  - index compression
- relevance feedback, query expansion, tolerant retrieval
- XML & other semi-structured document retrieval
- probabilistic IR & LM-based IR
- text classification & topic clustering
- dimensionality reduction & semantic clusters
- personalization, cross-lingual IR, ...