Advanced Topics in Information Retrieval

Vinay Setty
(vsetty@mpi-inf.mpg.de)

Jannik Strötgen
(jtroetge@mpi-inf.mpg.de)
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
Organization

- **Lectures:** Thursdays, 14:15 - 15:45, Weekly, room 023, E14, MPI-INF
  - Except for lectures on 16th and 23rd June, they will be held in room 0.01 in building E 1.7 (MMCI)

- **Tutorials:** Mondays 14:15 - 15:45, Biweekly, Room 023, E14, MPI-INF
  - Except for tutorial on 13th of June, room 0.01 in building E 1.7 (MMCI)

- **Lecturers:**
  - Vinay Setty (vsetty@mpi-inf.mpg.de) Appointments only by email (no fixed office hours)
  - Jannik Strötgen (jtroetge@mpi-inf.mpg.de) Appointments only by email (no fixed office hours)

- **Tutor:** We are the tutors!
Background Literature

  http://www.informationretrieval.org

- S. Büttcher, C. L.A. Clarke, G.V. Cormack, Information Retrieval, MIT Press, 2010

Required Background Knowledge

- Preferably passed IRDM lecture
- Basic programming skills (any language of your choice)
- LaTeX basics
Exercise Sheets

- Biweekly (almost) exercise sheets
  - six exercise sheets each with up to six problems
  - handed out during the lecture on Thursday (almost biweekly)
    - Refer to the course page for exact dates!
  - due by Thursday 11:59 PM of the following week
  - submit electronically as
    - PDF to atir16@mpi-inf.mpg.de
      (best: typeset using LaTeX, worst: scans of your handwriting)
    - If programming questions are given, also include the zip/tar of the source code
Tutorials

- Biweekly (almost) tutorials
  - on Mondays after due dates
  - Refer to the course page for exact dates!
  - we’ll grade your solutions as (P)resentable, (S)erious, (F)ail
  - no example solutions
Requirements for 6 ECTS

‣ Submit **serious or better** solutions to at least 50% of problems

‣ Present solutions in tutorial
  ‣ **at least once** during the semester

‣ additional presentations score you **bonus points**
  (one grade per bonus point, at most three, at most one per session)

‣ **Pass oral exam** at the end of the semester
Registration & Password

- You’ll have to register for this course and the exam in HISPOS.
- Please register by email to atir16@mpi-inf.mpg.de
  - Full name
  - Student number
  - Preferred e-mail address
- Some materials (e.g. papers and data) will be made available in a password-protected area on the course website.
  - Username: atir16 / Password: you should know it from the first lecture, if not send an email to atir16@mpi-inf.mpg.de
Questions/Suggestions?
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
What is this Course About?

- IR Basics recap (today)
  - Different retrieval models
  - Indexing and Query processing
  - Link analysis
  - IR Tools
- NLP for IR (April 28)
  - Tokenization, stop word removal, lemmatization
  - Part-of-speech tagging, dependency parsing, named entity recognition
  - Information extraction
  - IR evaluation measures
What is this Course About?

- Efficiency and Scalability issues in IR (May 12)
  - Index construction and maintenance
  - Index pruning
  - Query Processing
  - Web archives (versioned documents)
- Mining and Organizing (May 19)
  - Clustering
  - Classification
  - Temporal mining
  - Event mining and timelines
What is this Course About?

- Diversity and Novelty (Jun 2)
  - Diversification techniques: implicit and explicit
  - Diversification measures
- Semantic search (Jun 9)
  - Semantic web
  - Knowledge graphs
  - Entity linking and disambiguation
  - Semantic search, geographic IR
What is this Course About?

- Temporal Information Extraction (Jun 16)
  - Temporal expressions
  - Temporal tagging
  - Temporal scopes, document creation time
  - Temporal reasoning
  - Temporal information extraction
  - Demo: HeidelTime and SUTime

- Temporal Information Retrieval 1 (Jun 23)
  - Searching with temporal constraints
  - Temporal question answering
  - Temporal document and query profiles
  - Language models for temporal expressions
  - Historical document retrieval, Language evolution - Cultoronomics
What is this Course About?

- Temporal Information Retrieval 2 (tentative) (Jun 30)
  - ?
- Social Media (Jul 7)
  - Blogosphere mining TREC TSIT
  - Opinion retrieval
  - Spam/hoax detection
  - TDT and Event mining
  - Feed Distillation
- Learning to rank (Jul 14)
- Q&A (Jul 21)
- Oral Exam (Jul 28)
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
Information Need
Information Need

**Search Query:**

`how to kill python`

**Results:**

1. *linux* - How to kill a running python process? - Stack Overflow
   
   `stackoverflow.com/questions/.../how-to-kill-a-running-python-process`
   
   Oct 23, 2013 - This question has been asked before and already has an answer. If those answers do not fully address your question, please ...

2. With Florida python hunt about to begin, humane killing urged
   
   `www.palmbeachpost.com/.../python...killing.../nTps...`
   
   The Palm Beach Post
   
   Jan 7, 2013 - Decapitating Burmese pythons — a sanctioned method for killing the invasive snakes in the upcoming Python Challenge contest — is ...

3. Record-setting 128lb python killed by knife-wielding Miami ...
   
   `www.dailymail.co.uk/.../Record-setting-128lb-python-killed-k...`
   
   Daily Mail
   
   May 20, 2013 - Record-breaking 128lb Python that measures 18ft and 8in long is captured and killed in Florida. ... The biggest ever Python snake found in Florida, measuring 18 feet and 8 inches long, has been captured and killed. ... The Burmese
Information Need

Google search for "how to kill python"

About 12,900,000 results (0.67 seconds)

Linux - How to kill a running python process? - Stack Overflow
stackoverflow.com/questions/.../how-to-kill-a-running-python-process
Oct 23, 2013 - This question has been asked before and already has an answer. If those answers do not fully address your question, please ...

With Florida python hunt about to begin, humane killing urged
www.palmbeachpost.com/.../python...killing/...nTps...
The Palm Beach Post
Jan 7, 2013 - Decapitating Burmese pythons — a sanctioned method for killing the invasive snakes in the upcoming Python Challenge contest — is ...

Record-setting 128lb python killed by knife-wielding Miami ...
www.dailymail.co.uk/.../Record-setting-128lb-python-killed-k...
Daily Mail
May 20, 2013 - Record-breaking 128lb Python that measures 18ft and 8in long is captured and killed in Florida. ... The biggest ever Python snake found in Florida, measuring 18 feet and 8 inches long, has been captured and killed. ... The Burmese
Information Need

What is the capital of Saarland

About 17,100,000 results (0.76 seconds)

Saarland / Capital

Saarbrücken

Events and points of interest
Information Need

Google

What is the capital of Saarland

About 17,100,000 results (0.76 seconds)

Saarland / Capital

Saarbrücken

Events and points of interest

Knowledge Base
Information Need

Google search for "saarbrucken weather"

About 102,000 results (0.45 seconds)

Saarbrücken, Germany
Wednesday 12:00 PM
Sunny

57°F | 1°C
Precipitation: 0%
Humidity: 46%
Wind: 12 mph

Temperature chart:
- 57°F
- 59°F
- 58°F
- 50°F
- 46°F
- 44°F
- 42°F
- 48°F

Forecast:
- Wed: 61°F to 41°F
- Thu: 68°F to 45°F
- Fri: 64°F to 45°F
- Sat: 46°F to 32°F
- Sun: 49°F to 31°F
- Mon: 46°F to 34°F
- Tue: 48°F to 34°F
- Wed: 51°F to 33°F
Information Need

Google

**saarbrucken weather**

About 102,000 results (0.45 seconds)

**Saarbrücken, Germany**

Wednesday 12:00 PM  
Sunny

57°F °C  
Precipitation: 0%  
Humidity: 46%  
Wind: 12 mph

External Service
Information Need

log 2000 + sin 45

About 37,600,000 results (0.66 seconds)

log(2000) + sin(45 radians) = 4.1519335202
Information Need

Google: log 2000 + sin 45

About 37,600,000 results (0.66 seconds)

The result of the calculation is 4.1519335202.

Computation
Information Need

Google search: 100 € to $ shows that 100 Euro equals 113.71 US Dollar.
Information Need

Google search for "100 € to $" showing that 100 Euro equals 113.71 US Dollar.
Information Need

2014 FIFA World Cup Final - Wikipedia, the free encyclopedia
The 2014 FIFA World Cup Final was a football match that took place on 13 July 2014 at the ... The result marked Germany's fourth World Cup title, their first since German ...
The 2006 quarter-final game, where Germany won 4–2 in the shootout after .... German President Joachim Gauck and Chancellor Angela Merkel were ...
Background - Road to the final - Match ball - Match officials

World Cup 2014: how 'Germany's 12th man' Angela Merkel ...
www.telegraph.co.uk › ... › Teams › Germany - The Daily Telegraph
Jul 14, 2014 - Football-loving German chancellor Angela Merkel enjoys surge in support ... sight at Germany matches since the country hosted the World Cup in 2006. .... Germany deserved to win though Argentina could have also won had ...
Information Need

Temporal QA?
Documents

- Webpages, news articles etc
- Social media tweets, forums, Facebook status etc.
- Books, Journals, scholars article etc
- Knowledge Graphs
- Files/data on your personal computer
- Apps/data on your smartphones
What is Information Retrieval?
What is Information Retrieval?

- Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need (usually a query) from within large collections (usually stored on computers). - Manning et. al.
Information Retrieval

Information need

Documents
Information Retrieval

Information need → Query Representation

Documents → Document Representation
Information Retrieval

Information need

Documents

Query Representation

How to match?

Document Representation
Information Retrieval in a Nutshell
Information Retrieval in a Nutshell

Crawling

strategies for crawl schedule and priority queue for crawl frontier
Information Retrieval in a Nutshell

Crawling: strategies for crawl schedule and priority queue for crawl frontier

Extract, & Clean: handle dynamic pages, detect duplicates, detect spam,
Information Retrieval in a Nutshell

- Crawling: strategies for crawl schedule and priority queue for crawl frontier
- Extract, & Clean: handle dynamic pages, detect duplicates, detect spam,
- preprocess: tokenization, stop word removal, stemming, lemmatization
Information Retrieval in a Nutshell

Crawling
- strategies for crawl schedule and priority queue for crawl frontier
- handle dynamic pages, detect duplicates, detect spam,

Extract, & Clean
- tokenization, stop word removal, stemming, lemmatization

Preprocess

Index
- build and analyze web graph, index all tokens, compress index
Information Retrieval in a Nutshell

Crawling
- strategies for crawl schedule and priority queue for crawl frontier
  - handle dynamic pages, detect duplicates, detect spam,

Extract, & Clean
- tokenization, stop word removal, stemming, lemmatization

preprocess

Index
- build and analyze web graph, index all tokens, compress index

Process Queries
- fast top-k queries, query logging, auto-completion
Information Retrieval in a Nutshell

Crawling
- strategies for crawl schedule and priority queue for crawl frontier
- handle dynamic pages, detect duplicates, detect spam,

Extract, & Clean
- tokenization, stop word removal, stemming, lemmatization

Preprocess
- build and analyze web graph, index all tokens, compress index

Index
- fast top-k queries, query logging, auto-completion

Process Queries
- scoring function over many data and context criteria, diversification

Ranking
Information Retrieval in a Nutshell

- **Crawling**: Strategies for crawl schedule and priority queue for crawl frontier
  - Handle dynamic pages, detect duplicates, detect spam,

- **Extract, & Clean**: Tokenization, stop word removal, stemming, lemmatization
  - Index all tokens, compress index

- **Preprocess**: Build and analyze web graph
  - Fast top-k queries, query logging, auto-completion

- **Ranking**: Scoring function over many data and context criteria, diversification
  - GUI, user guidance, personalization, visualization

- **Present**: Process Queries
Pre-processing of documents and queries typically includes:

- **tokenization** (e.g., splitting them up at white spaces and hyphens)
- **stemming** or **lemmatization** (to group variants of the same word)
- **stopword removal** (to get rid of words that bear little information)

This results in a **bag (or sequence) of indexable terms**

Investigators entered the company’s HQ located in Boston MA on Thursday.

{ investig enter compani hq locat boston ma thursdai }

{ investig enter compani hq locat boston ma thursdai }

{ investig enter compani hq locat boston ma thursdai }
Documents & Queries

- Pre-processing of documents and queries typically includes
  - **tokenization** (e.g., splitting them up at white spaces and hyphens)
  - **stemming** or **lemmatization** (to group variants of the same word)
  - **stopword removal** (to get rid of words that bear little information)
- This results in a **bag (or sequence)** of indexable terms

Investigators entered the company’s HQ located in Boston MA on Thursday.

More in next lecture
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Tools for IR - Elasticsearch
Retrieval Models

- Retrieval model defines for a given document collection \( D \) and a query \( q \) which documents to return and in which order
  - Boolean retrieval
  - Probabilistic retrieval models (e.g., binary independence model)
  - Vector space model with \( \text{tf.idf} \) term weighting
  - Language models
  - Latent topic models (e.g., LSI, pLSI, LDA)
Boolean Retrieval

- **Boolean variables** indicate presence/absence of query terms
- **Boolean operators** AND, OR, and NOT
- **Boolean queries** are arbitrary compositions of those, e.g.:
  - Frodo AND Sam AND NOT Gollum
  - NOT ((Saruman AND Sauron) OR (Smaug AND Shelob))
- Extensions of Boolean retrieval (e.g., proximity, wildcards, fields) with rudimentary ranking (e.g., weighted matches) exist
## Processing Boolean Queries

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frodo</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sam</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gollum</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Saruman</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gandalf</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Sauron</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

How to Process the query:

Frodo AND Sam AND NOT Gollum
Processing Boolean Queries

- Take the term vectors (Frodo, Sam, and Gollum)
- Flip the bits for terms with NOT (e.g. Gollum)
- bitwise AND the vectors finally the documents which return 1 are relevant

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frodo</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sam</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gollum</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Processing Boolean Queries

- Take the term vectors (Frodo, Sam, and Gollum)
- Flip the bits for terms with NOT (e.g. Gollum)
- bitwise AND the vectors finally the documents which return 1 are relevant

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frodo</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sam</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gollum</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Processing Boolean Queries

- Take the term vectors (Frodo, Sam, and Gollum)
- Flip the bits for terms with NOT (e.g. Gollum)
- Bitwise AND the vectors finally the documents which return 1 are relevant

<table>
<thead>
<tr>
<th></th>
<th>d1</th>
<th>d2</th>
<th>d3</th>
<th>d4</th>
<th>d5</th>
<th>d6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Frodo</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sam</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gollum</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

d1 and d4 are the relevant documents for Frodo AND Sam AND NOT Gollum
Vector Space Model

- Vector space model considers queries and documents as vectors in a common high-dimensional vector space.
- Cosine similarity between two vectors $q$ and $d$ is the cosine of the angle between them.

$$sim(q, d) = \frac{q \cdot d}{\|q\| \|d\|} = \frac{\sum_v q_v d_v}{\sqrt{\sum_v q_v^2} \sqrt{\sum_v d_v^2}}$$
tf.idf

- How to set the components of query and document vectors?

- Intuitions behind tf.idf term weighting:
  - documents should profit if they contain a query term more often
  - terms that are common in the collection should be assigned a lower weight

- Term frequency $tf(v, d)$ – # occurrences of term $v$ in document $d$

- Document frequency $df(v)$ – # documents containing term $v$

- Components of document vectors set as

$$d_v = tf(v, d) \log \frac{|D|}{df(v)}$$
Statistical Language Models

- Models to describe language generation
- Traditional NLP applications: Assigns a probability value to a sentence
  - Machine Translation — \( P(\text{high snowfall}) > P(\text{large snowfall}) \)
  - Spelling Correction — \( P(\text{in the vineyard}) > P(\text{in the vinyard}) \)
  - Speech Recognition — \( P(\text{It's hard to recognize speech}) > P(\text{It's hard to wreck a nice beach}) \)
- Question Answering
- Goal: compute the probability of a sentence or sequence of words:
  \[ P(S) = P(w_1, w_2, w_3, w_4, w_5 \ldots w_n) \]
Language Model of a Document

- Language model describes the probabilistic generation of elements from a formal language (e.g., sequences of words)
- Documents and queries can be seen as samples from a language model and be used to estimate its parameters
- Maximum Likelihood Estimate (MLE) for each word is the most natural estimate

\[
P[v | \theta_d] = \frac{tf(v, d)}{\sum_w tf(w, d)}
\]

\[
P[a | \theta_d] = \frac{16}{25}
\]
\[
P[b | \theta_d] = \frac{6}{25}
\]
\[
P[c | \theta_d] = \frac{3}{25}
\]
Unigram Language Models

- Unigram Language Model provides a probabilistic model for representing text
- With unigram we can also assume terms are independent

<table>
<thead>
<tr>
<th>Words</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>0.15</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>0.12</td>
</tr>
<tr>
<td>Frodo</td>
<td>0.01</td>
<td>0.0002</td>
</tr>
<tr>
<td>Sam</td>
<td>0.01</td>
<td>0.0001</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>that</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Rosie</td>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>Gandalf</td>
<td>0.003</td>
<td>0.015</td>
</tr>
<tr>
<td>Saruman</td>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>

\[ P(\text{Frodo said that Sam likes Rosie}) \]
Unigram Language Models

- Unigram Language Model provides a probabilistic model for representing text.
- With unigram we can also assume terms are independent.

<table>
<thead>
<tr>
<th>Words</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>0.15</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>0.12</td>
</tr>
<tr>
<td>Frodo</td>
<td>0.01</td>
<td>0.0002</td>
</tr>
<tr>
<td>Sam</td>
<td>0.01</td>
<td>0.0001</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>that</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Rosie</td>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>Gandalf</td>
<td>0.003</td>
<td>0.015</td>
</tr>
<tr>
<td>Saruman</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[
P(\text{Frodo said that Sam likes Rosie}) = P(\text{Frodo}) \times P(\text{said}) \times P(\text{that}) \times P(\text{Sam}) \times P(\text{likes}) \times P(\text{Rosie})
\]
Unigram Language Models

- Unigram Language Model provides a probabilistic model for representing text
- With unigram we can also assume terms are independent

<table>
<thead>
<tr>
<th>Words</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>0.15</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>0.12</td>
</tr>
<tr>
<td>Frodo</td>
<td>0.01</td>
<td>0.0002</td>
</tr>
<tr>
<td>Sam</td>
<td>0.01</td>
<td>0.0001</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>that</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Rosie</td>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>Gandalf</td>
<td>0.003</td>
<td>0.015</td>
</tr>
<tr>
<td>Saruman</td>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>

\[
P(\text{Frodo said that Sam likes Rosie}) = P(\text{Frodo}) \times P(\text{said}) \times P(\text{that}) \times P(\text{Sam}) \times P(\text{likes}) \times P(\text{Rosie})
\]
Unigram Language Models

- Unigram Language Model provides a probabilistic model for representing text
- With unigram we can also assume terms are independent

<table>
<thead>
<tr>
<th>Words</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>0.15</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>0.12</td>
</tr>
<tr>
<td>Frodo</td>
<td>0.01</td>
<td>0.0002</td>
</tr>
<tr>
<td>Sam</td>
<td>0.01</td>
<td>0.0001</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>that</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Rosie</td>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>Gandalf</td>
<td>0.003</td>
<td>0.015</td>
</tr>
<tr>
<td>Saruman</td>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>

\[
P(\text{Frodo said that Sam likes Rosie}) = P(\text{Frodo}) \times P(\text{said}) \times P(\text{that}) \times P(\text{Sam}) \times P(\text{likes}) \times P(\text{Rosie})
\]

<table>
<thead>
<tr>
<th>s</th>
<th>Frodo</th>
<th>said</th>
<th>that</th>
<th>Sam</th>
<th>likes</th>
<th>Rosie</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>0.01</td>
<td>0.03</td>
<td>0.04</td>
<td>0.01</td>
<td>0.02</td>
<td>0.005</td>
</tr>
<tr>
<td>M2</td>
<td>0.0002</td>
<td>0.03</td>
<td>0.04</td>
<td>0.0001</td>
<td>0.04</td>
<td>0.01</td>
</tr>
</tbody>
</table>

\[
P(s|M1) = 0.0000000000012\]
\[
P(s|M2) = 0.00000000000096\]
Zero Probability Problem

- what if some of the queried terms are absent in the document?
- frequency based estimation results in a zero probability for query generation

<table>
<thead>
<tr>
<th>Words</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>0.15</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>0.12</td>
</tr>
<tr>
<td>Frodo</td>
<td>0.01</td>
<td>0.0002</td>
</tr>
<tr>
<td>Sam</td>
<td>0.01</td>
<td>0.0001</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>that</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Rosie</td>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>Gandalf</td>
<td>0.003</td>
<td>0.015</td>
</tr>
<tr>
<td>Saruman</td>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>
Zero Probability Problem

- what if some of the queried terms are absent in the document?
- frequency based estimation results in a zero probability for query generation

<table>
<thead>
<tr>
<th>Words</th>
<th>M1</th>
<th>M2</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
<td>0.15</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
<td>0.12</td>
</tr>
<tr>
<td>Frodo</td>
<td>0.01</td>
<td>0.0002</td>
</tr>
<tr>
<td>Sam</td>
<td>0.01</td>
<td>0.0001</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>likes</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>that</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Rosie</td>
<td>0.005</td>
<td>0.01</td>
</tr>
<tr>
<td>Gandalf</td>
<td>0.003</td>
<td>0.015</td>
</tr>
<tr>
<td>Saruman</td>
<td>0.001</td>
<td>0.002</td>
</tr>
</tbody>
</table>

\[
P(\text{"Frodo", "Gollum"}|M1) = 0.01 \times 0 \]
\[
P(\text{"Frodo", "Gollum"}|M2) = 0.0002 \times 0 \]
Smoothing

- Need to smooth the probability estimates for terms to avoid zero probabilities
- Smoothing introduces a **relative term weighting** (idf-like effect) since more common terms now have higher probability for all documents
- Parameter estimation from a **single document or query** bears the **risk of overfitting** to this very limited sample
- Smoothing methods estimate parameters considering the entire document collection as a background model
Jelinek-Mercer smoothing

- Linear combination of document and corpus statistics to estimate term probabilities

\[
P[v \mid \theta_d] = \alpha \cdot \frac{tf(v, d)}{\sum_w tf(w, d)} + (1 - \alpha) \cdot \frac{tf(v, D)}{\sum_w tf(w, D)}
\]

- Collection frequency: fraction of occurrence of term in the document \(d\)

- Document frequency: fraction of document occurrence of term in the entire collection \(D\)
Jelinek-Mercer smoothing

- Linear combination of document and corpus statistics to estimate term probabilities

\[
P[v | \theta_d] = \alpha \cdot \frac{tf(v, d)}{\sum_w tf(w, d)} + (1 - \alpha) \cdot \frac{tf(v, D)}{\sum_w tf(w, D)}
\]

- Collection frequency: fraction of occurrence of term in the document \(d\)

- Document frequency: fraction of document occurrence of term in the entire collection \(D\)
Jelinek-Mercer smoothing

- Linear combination of document and corpus statistics to estimate term probabilities

\[ P[v | \theta_d] = \alpha \cdot \frac{tf(v, d)}{\sum_w tf(w, d)} + (1 - \alpha) \cdot \frac{tf(v, D)}{\sum_w tf(w, D)} \]

- Collection frequency: fraction of occurrence of term in the document \(d\)
- Document frequency: fraction of document occurrence of term in the entire collection \(D\)
**Jelinek-Mercer smoothing**

- Linear combination of document and corpus statistics to estimate term probabilities

\[ P[v | \theta_d] = \alpha \cdot \frac{tf(v, d)}{\sum_w tf(w, d)} + (1 - \alpha) \cdot \frac{tf(v, D)}{\sum_w tf(w, D)} \]

- Collection frequency: fraction of occurrence of term in the document \( d \)
- Document frequency: fraction of document occurrence of term in the entire collection \( D \)
Jelinek-Mercer smoothing

- Linear combination of document and corpus statistics to estimate term probabilities

\[ P[v \mid \theta_d] = \alpha \cdot \frac{tf(v, d)}{\sum_w tf(w, d)} + (1 - \alpha) \cdot \frac{tf(v, D)}{\sum_w tf(w, D)} \]

- Collection frequency: fraction of occurrence of term in the document \( d \)
- Document frequency: fraction of document occurrence of term in the entire collection \( D \)
Dirichlet Smoothing

- Smoothing with Dirichlet Prior:

\[
P[v | \theta_d] = \frac{tf(v, d) + \mu \frac{tf(v, D)}{\sum_w tf(w, D)}}{\sum_w tf(w, d) + \mu}
\]

- Takes the corpus distribution as a prior to estimating the prob. for terms
Dirichlet Smoothing

- Smoothing with Dirichlet Prior:

\[
P[v | \theta_d] = \frac{tf(v, d) + \mu \sum_w tf(w, D)}{\sum_w tf(w, d) + \mu}
\]

- Takes the corpus distribution as a prior to estimating the prob. for terms
Dirichlet Smoothing

- **Smoothing with Dirichlet Prior:**

\[
P[v | \theta_d] = \frac{tf(v, d) + \mu \frac{tf(v, D)}{\sum_w tf(w, D)}}{\sum_w tf(w, d) + \mu}
\]

- Takes the corpus distribution as a prior to estimating the prob. for terms
Smoothing with Dirichlet Prior:

\[ P[v \mid \theta_d] = \frac{tf(v, d) + \mu \frac{tf(v, D)}{\sum_w tf(w, D)}}{\sum_w tf(w, d) + \mu} \]

- Takes the corpus distribution as a prior to estimating the prob. for terms
Query Likelihood vs. Divergence

- **Query-likelihood approaches** rank documents according to the probability that their language model generates the query

\[
P(q | \theta_d) \propto \prod_{v \in q} P(v | \theta_d)
\]

- **Divergence-based approaches** rank according to the Kullback-Leibler divergence between the query language model and language models estimate from documents

\[
KL(\theta_q \| \theta_d) = \sum_v P(v | \theta_q) \log \frac{P(v | \theta_q)}{P(v | \theta_d)}
\]
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
Link Analysis

- Link analysis methods consider the Web’s hyperlink graph to determine characteristics of individual web pages.

\[ A = \begin{bmatrix}
0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0
\end{bmatrix} \]
PageRank

- PageRank (by Google) is based on the following random walk
  - jump to a random vertex (1 / |V|) in the graph with probability $\varepsilon$
  - follow a random outgoing edge (1 / out(v)) with probability $(1-\varepsilon)$

$$p(v) = (1 - \varepsilon) \cdot \sum_{(u,v) \in E} \frac{p(u)}{out(u)} + \frac{\varepsilon}{|V|}$$

- PageRank score $p(v)$ of vertex $v$ is a measure of popularity and corresponds to its stationary visiting probability
PageRank scores correspond to components of the dominant Eigenvector $\pi$ of the transition probability matrix $P$ which can be computed using the power-iteration method.
PageRank scores correspond to components of the dominant Eigenvector $\pi$ of the transition probability matrix $P$ which can be computed using the power-iteration method.

\[ P = \begin{bmatrix} 0.05 & 0.45 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix} \]

$\epsilon = 0.2$
PageRank scores correspond to components of the dominant Eigenvector $\pi$ of the transition probability matrix $P$ which can be computed using the power-iteration method.
PageRank scores correspond to components of the dominant Eigenvector $\pi$ of the transition probability matrix $P$ which can be computed using the power-iteration method.

$$\pi^{(0)} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}, \quad \pi^{(1)} = \begin{bmatrix} 0.15 & 0.15 & 0.25 & 0.45 \end{bmatrix}$$

$$P = \begin{bmatrix} 0.05 & 0.45 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix}$$

$\epsilon = 0.2$
PageRank scores correspond to components of the dominant Eigenvector $\pi$ of the transition probability matrix $P$ which can be computed using the power-iteration method.

$$P = \begin{bmatrix} 0.05 & 0.45 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.05 & 0.85 \\ 0.45 & 0.05 & 0.05 & 0.45 \\ 0.05 & 0.05 & 0.85 & 0.05 \end{bmatrix}$$

$$\pi^{(0)} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix}$$

$$\pi^{(1)} = \begin{bmatrix} 0.15 & 0.15 & 0.25 & 0.45 \end{bmatrix}$$

$$\pi^{(2)} = \begin{bmatrix} 0.15 & 0.11 & 0.41 & 0.33 \end{bmatrix}$$

$$\pi^{(10)} = \begin{bmatrix} 0.18 & 0.12 & 0.34 & 0.36 \end{bmatrix}$$

$\epsilon = 0.2$
HITS operates on a subgraph of the Web induced by a keyword query and considers

- hubs as vertices pointing to good authorities
- authorities as vertices pointed to by good hubs

Hub score $h(u)$ and authority score $a(v)$ defined as

\[
h(u) \propto \sum_{(u,v) \in E} a(v) \quad a(v) \propto \sum_{(u,v) \in E} h(u)
\]

Hub vector $h$ and authority vector $a$ are Eigenvectors of the co-citation matrix $AA^T$ and co-reference matrix $A^TA$

\[
h = \alpha \beta AA^T h \quad a = \alpha \beta A^T A a
\]
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
Indexing & Query Processing

- Retrieval models define which documents to return for a query but not how they can be identified efficiently.

- **Index structures** are an essential building block for IR systems; variants of the inverted index are by far most common.

- **Query processing methods** operate on these index structures:
  - holistic query processing methods determine all query results (e.g., term-at-a-time, document-at-a-time)
Inverted Index

- Inverted index as widely used index structure in IR consists of
  - dictionary mapping terms to term identifiers and statistics (e.g., df)
  - posting list for every term recording details about its occurrences
- Posting lists can be document- or score-ordered and be equipped with additional structure (e.g., to support skipping)
- Postings contain a document identifier plus additional payloads (e.g., term frequency, tf.idf score contribution, term offsets)
Term-at-a-Time

- Processes posting lists for query terms $\langle q_1, \ldots, q_m \rangle$ one at a time

- Maintains an accumulator for each document seen; after processing the first $k$ query terms this corresponds to

$$
acc(d) = \sum_{i=1}^{k} score(q_i, d)
$$

- Main memory proportional to number of accumulators

- Top-k result determined at the end by sorting accumulators
Term-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) **one at a time**

- Maintains an **accumulator for each document** seen; after processing the first \( k \) query terms this corresponds to

\[
acc(d) = \sum_{i=1}^{k} score(q_i, d)
\]

<table>
<thead>
<tr>
<th>a</th>
<th>d_1, 0.2</th>
<th>d_3, 0.1</th>
<th>d_5, 0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>d_5, 0.3</td>
<td>d_7, 0.2</td>
<td></td>
</tr>
</tbody>
</table>

- **Main memory** proportional to number of accumulators

- Top-\( k \) result determined at the end by sorting accumulators
Term-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) one at a time

- Maintains an accumulator for each document seen; after processing the first \( k \) query terms this corresponds to

\[
acc(d) = \sum_{i=1}^{k} \text{score}(q_i, d)
\]

- Main memory proportional to number of accumulators

- Top-k result determined at the end by sorting accumulators
Term-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) one at a time

- Maintains an **accumulator for each document** seen; after processing the first \( k \) query terms this corresponds to

\[
acc(d) = \sum_{i=1}^{k} score(q_i, d)
\]

- **Main memory** proportional to number of accumulators

- **Top-k result** determined at the end by sorting accumulators
Term-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) one at a time
- Maintains an accumulator for each document seen; after processing the first \( k \) query terms this corresponds to

\[
acc(d) = \sum_{i=1}^{k} score(q_i, d)
\]

- Main memory proportional to number of accumulators
- Top-k result determined at the end by sorting accumulators
Term-at-a-Time

- Processes posting lists for query terms $\langle q_1, \ldots, q_m \rangle$ one at a time

- Maintains an **accumulator for each document** seen; after processing the first $k$ query terms this corresponds to

$$acc(d) = \sum_{i=1}^{k} score(q_i, d)$$

- **Main memory** proportional to number of accumulators

- Top-$k$ result determined at the end by sorting accumulators
Term-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) one at a time

- Maintains an accumulator for each document seen; after processing the first \( k \) query terms this corresponds to

\[
acc(d) = \sum_{i=1}^{k} score(q_i, d)
\]

- Main memory proportional to number of accumulators

- Top-k result determined at the end by sorting accumulators
Processes posting lists for query terms $\langle q_1, \ldots, q_m \rangle$ all at once

Sees the same document in all posting lists at the same time, determines score, and decides whether it belongs into top-k

Main memory proportional to $k$ or number of results

Skipping aids conjunctive queries (all query terms required)
Document-at-a-Time

- Processes posting lists for query terms $\langle q_1, \ldots, q_m \rangle$ all at once

- Sees the same document in all posting lists at the same time, determines score, and decides whether it belongs into top-k

  a  
  $d_1$, 0.2  $d_3$, 0.1  $d_5$, 0.5

  b  
  $d_5$, 0.3  $d_7$, 0.2

- Main memory proportional to k or number of results

- Skipping aids conjunctive queries (all query terms required)
Document-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) all at once
- Sees the same document in all posting lists at the same time, determines score, and decides whether it belongs into top-k

\[
\begin{array}{c|c}
\text{a} & d_1, 0.2 \quad d_3, 0.1 \quad d_5, 0.5 \\
\text{b} & d_5, 0.3 \quad d_7, 0.2 \\
\end{array}
\]

- Main memory proportional to \( k \) or number of results
- Skipping aids conjunctive queries (all query terms required)
Document-at-a-Time

- Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) all at once

- Sees the same document in all posting lists at the same time, determines score, and decides whether it belongs into top-\( k \)

- Main memory proportional to \( k \) or number of results

- Skipping aids conjunctive queries (all query terms required)
Document-at-a-Time

- Processes posting lists for query terms $\langle q_1, \ldots, q_m \rangle$ all at once

- Sees the same document in all posting lists at the same time, determines score, and decides whether it belongs into top-$k$

- Main memory proportional to $k$ or number of results

- Skipping aids conjunctive queries (all query terms required)
Processes posting lists for query terms \( \langle q_1, \ldots, q_m \rangle \) all at once

Sees the same document in all posting lists at the same time, determines score, and decides whether it belongs into top-k

Main memory proportional to k or number of results

Skipping aids conjunctive queries (all query terms required)
Agenda

- Organization
- Course overview
- What is IR?
- Retrieval Models
- Link Analysis
- Indexing and Query Processing
- Tools for IR - Elasticsearch
Elasticsearch

- Flexible and powerful open source, distributed real-time search and analytics engine

- Features:
  - real time data, real time analytics,
  - distributed, high availability,
  - full text search, document oriented,
  - conflict management, schema free (json)
  - restful api, per-operation persistence,
  - apache 2 open source license, build on top of apache lucene.
Elasticsearch Installation

- curl -L -O https://download.elastic.co/elasticsearch/release/org/elasticsearch/distribution/tar/elasticsearch/2.3.1/elasticsearch-2.3.1.tar.gz
- tar -xvf elasticsearch-2.3.1.tar.gz
- cd elasticsearch-2.3.1/bin
- ./elasticsearch
Elasticsearch - Indexing

- Create Index
  
  ```bash
  $ curl -XPUT 'http://localhost:9200/atirtest'
  ```

- Add records to index
  
  ```bash
  $ curl -XPUT 'http://localhost:9200/atirtest/doc/1' -d '{
    "title" : "Ecuador earthquake: Aid agencies step up efforts",
    "pub_date" : 1461230434627,
    "content" : "Aid agencies are stepping up help following Saturdays devastating earthquake in Ecuador, amid concerns over the conditions faced by survivors."
  }'
  
  $ curl -XPUT 'http://localhost:9200/atirtest/doc/2' -d '{
    "title" : "Syria conflict: Air strikes on Idlib markets kill dozens",
    "pub_date" : 1461230434457,
    "content" : "At least 44 people have been killed and dozens hurt in Syrian government air strikes on markets in two rebel-held towns in Idlib province, activists say."
  }'
  ```
curl -XGET 'localhost:9200/atirtest/_search?pretty' -d ' {
  "query":{
    "query_string": {
      "default_field": "content",
      "query": "earthquake AND Ecuador AND NOT Syrian"
    }
  }
}'
Elasticsearch has built-in language tools for

- tokenization
- stop word removal
- stemming

For Arabic, Armenian, Basque, Brazilian, Bulgarian, Catalan, CJK, Czech, Danish, Dutch, English, Finnish, French, Galician, German, Greek, Hindi, Hungarian, Indonesian, Irish, Italian, Latvian, Lithuanian, Norwegian, Persian, Portuguese, Romanian, Russian, Sorani, Spanish, Swedish, Turkish, Thai.
"settings": {
  "analysis": {
    "filter": {
      "stop_filter": {
        "type": "stop",
        "stopwords": ["_english_"]
      },
      "custom_english_stemmer": {
        "type": "stemmer",
        "name": "minimal_english"
      }
    },
    "analyzer": {
      "custom_lowercase_stemmed": {
        "tokenizer": "standard",
        "filter": [
          "stop_filter",
          "custom_english_stemmer",
          "lowercase"
        ]
      }
    }
  }
}
Elasticsearch - TF-IDF

- Query with TF-IDF score

```bash
curl -X GET 'localhost:9200/atirtest/_search?pretty' -d \
   '{
     "query": {
       "match": {
         "content": "Earthquake in Ecuador"
       }
     }
   }'
```
Okapi BM25 in Elasticsearch

curl -XPUT 'localhost:9200/atirtest/' -d'
{
    "mappings": {
        "doc": {
            "properties": {
                "title": {
                    "type": "string",
                    "similarity": "BM25"
                },
                "pub_date": {
                    "type": "date"
                },
                "content": {
                    "type": "string",
                    "similarity": "BM25"
                }
            }
        }
    }
}'}
Caveats

- For debugging: Most errors can be because of a comma missing or a bracket not closed
- Use Marvel sense UI to compose your elasticsearch code
  - It compiles the code and points the error for you
Credits

Thanks to Prof. Klaus Berberich and Dr. Avishek Anand for allowing us to use their slides!