Language and Computers: Searching

Based on Dickinson, Brew, and Meurers 2013
Many slides based on slides by Markus Dickinson
Katrin Erk
The University of Texas at Austin
Searching

Gigantic number of information resources:
- Books
- Databases
- The Web
- Newspapers
- ...

How to locate relevant information? Searching these resources, which often are written texts:
- Searching in a library catalogue
- Searching the web using a search engine
- Advanced search in text corpora
  - for example using regular expressions, which specify patterns in text:
    let the .* cat .* out of the .* bag
Searching in speech

Searching through

◦ Interview recordings
◦ Podcasts
◦ ...

This mostly does not work yet

In the following, we focus on searching in written text
Structured data, unstructured data

**Structured data**: organized in categories
- For example, a library database
- Searchable by categories like author, title, book vs journal, topic, ...
- Someone has to do the structuring

**Unstructured data**: just text
- Much greater quantities available than for structured data, for example on the web
- Keyword search can be highly effective
Structured data, unstructured data

Wait a minute – text is unstructured?

A wordle is unstructured, but text has a highly sophisticated structure. Too sophisticated maybe.

How difficult do you think it is to automatically identify the structure in a text?
Searching through structured data: Library catalogues

To find books and other items in a library, a library generally provides:
- a database with information on the items it has and
- a database frontend for users to interact with the database.

Users search for the occurrence of strings occurring in the author, title, keywords, cell number, ...
Basic searching with scoUT

Literal strings
No distinction of upper-case, lower-case letters
Adjacent words are searched as a phrase:
  ◦ art therapy
  ◦ vitamin c

Handling of stop words:
  ◦ Search for “language and computers” first gets you a book called “language and computers”, while “language computers” gets you books on computer languages
  ◦ In some search engines, **stop words** are ignored:
    ◦ a, an, at, be, but, by, do, for, if, in, is, it, of, on, the, to
    ◦ Stop words are an important concept for search in general. Can you guess why?
Special characters and operators

In addition to querying literal strings, the query language also supports special Boolean expressions for combining query strings

- "No general method for the solution of questions in the theory of probabilities can be established which does not explicitly recognise, not only the special numerical bases of the science, but also those universal laws of thought which are the basis of all reasoning, and which, whatever they may be as to their essence, are at least mathematical as to their form"

Use AND and OR to specify multiple strings to be matched

art OR therapy

Russell AND Norvig

In principle, you can also do more complex combinations, using parentheses to group parts (but not in scoUT):

art therapy NOT ((music OR dance) therapy)
Structured data

Do you know examples other than libraries?

Who does the structuring, and why do people put in the money to do it?
Structured data

Do you know examples other than libraries?

Who does the structuring, and why do people put in the money to do it?

- Google knowledge graph (similarly in other search engines)
- Wikidata
  - https://blog.wikimedia.org/2013/04/25/the-wikidata-revolution/
- The semantic web
The semantic web

Idea:
- People annotate their own webpages with structured information

Taking the example apart:
- Start with an ordinary webpage.
- Add structured information
- Using an ontology made and shared by many

Later more on the specification language used in the example.
Search in unstructured data

Unstructured means:
- There may be structure, but we don’t know what it is
- ... or it is not uniformly applied or standardized

Example:
- Webpages that can be switched between two or more languages
- Humans are good at spotting this, even when there is no uniform structure in where the language switch button is located
- Can you find it on this website? [http://www.gu.se/](http://www.gu.se/). How did you do this?

Search in unstructured data:
- No explicit categorization of documents
- Related to doing a keyword search in structured data
- Scale is different: billions of web pages to search
Information need

Searching involves *information need*: the information a searcher is seeking

Searcher translates their information need into a query

This is an imperfect process

Example:
- Information need: one or more Russian translations of the English word “table”
- Possible query: russian translation table
- Could be looking for a table (chart) of Russian translations
- First hit on DuckDuckGo: chart of romanizations of Russian characters

Information need is unambiguous; query is ambiguous
Searching the web

Computer user wants to find something on the web

Uses a search engine: a program that matches documents to a user’s search requests

Enters a query: a request for information

Receives a ranked list of web sites that might be relevant

Evaluates the results: either picks a website from the list or reformulates the query
Evaluation:
What makes a good search engine?

How do you evaluate whether you like a search engine?
Evaluation: What makes a good search engine?

Do you remember the definitions of precision and recall from our discussion of document classification?

**Precision**: Out of the web pages that the search engine returned, how many are the ones we want?

**Recall**: Out of the web pages in existence that would have met our information need, how many did the search engine find?
How search engines work (roughly)

Store a copy of all web pages

The web is huge, and users are impatient:
- How many seconds would you be willing to wait for your search result?

Create an index for efficient access
- Google: 1 trillion URLs in 2008

Some words may be treated as exchangeable:
- Capitalization
- Stemming
- Tubingen/Tuebingen/Tübingen

Rank query results by some web page rank
Search engine indexing
(Manning/Raghavan/Schütze 2008)

As a search engine crawls the web, it builds a **term-by-document matrix:**
Which terms appear in which documents?

Example, for some mystery novels:

<table>
<thead>
<tr>
<th></th>
<th>Affair at Styles</th>
<th>Secret Adversary</th>
<th>Sherlock Holmes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poirot</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Sherlock</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>adventure</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>exceedingly</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>strychnine</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>subsided</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1 = word appears. 0 = word does not appear

Stop words may be omitted from indexing: I, the, of, and, ...
Search engine indexing

**Inverted index:** map
- from a word
- to the documents (or IDs of documents) in which they appear

More efficient search: directly have a term point to the documents in which it appears

<table>
<thead>
<tr>
<th>Term</th>
<th>IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poirot</td>
<td>1, 4, 13, 15, 45, ...</td>
</tr>
<tr>
<td>Sherlock</td>
<td>3, 111, ...</td>
</tr>
<tr>
<td>adventure</td>
<td>1, 2, 3, 4, 5, 9, 15, ...</td>
</tr>
<tr>
<td>exceedingly</td>
<td>1, 3, 11, 25, ...</td>
</tr>
<tr>
<td>strychnine</td>
<td>1, 15, 60, ...</td>
</tr>
<tr>
<td>subsided</td>
<td>1, 3, 12, 13, 25, ...</td>
</tr>
</tbody>
</table>

Boolean operators again: What should the search engine do when you search for poirot AND adventure AND strychnine?

Intersect the document ID lists for the 3 terms
Ranking results

Ideally, the most relevant pages should show up first, on the first page

- Users don’t frequently even click through to the second page

Search engine does not understand language. Then how can it determine the relevance of a page?

How would you do this?
Ranking webpages with PageRank

Larry Page, Sergey Brin in the 1990s. This algorithm was the basis of Google.

Pay attention to hyperlinks:
- If a site has many incoming links, it must be well known
- This could be because people find it useful

Importance of a webpage computed based on its popularity:
- number of incoming links
- how popular the pages are that link to it
Ranking webpages with PageRank

Pages X, Y, Z all link to A.

Pages V, W, X all link to B. Are those links better or worse than the links to A?
That depends on how popular/authoritative X, Y, Z are versus V, W, X.
Ranking webpages with PageRank

Pages X, Y, Z all link to A.

Pages V, W, X all link to B. Are those links better or worse than the links to A?
That depends on how popular/authoritative X, Y, Z are versus V, W, X.
Represent that popularity by giving the sites a score.
Ranking webpages with PageRank

What to do with the incoming scores?

Simplest option: add them up.

But then A has a score of 45, more than any of the sites that linked to it.

That is too much.
Ranking webpages with PageRank

Idea: Score = number of votes. Website spreads out its number of votes among the pages that it links to.
But where does $X$’s score come from?

The same way we compute the score for $A$: from its incoming links.

We just do multiple rounds of voting.

For an animated illustration of PageRank, see this page: https://www.r-bloggers.com/from-random-walks-to-personalized-pagerank/
PageRank, simplified method

PageRank value can be seen as: Probability that a person, clicking around at random, will land on this page

More likely that they will end up on a page that many other pages link to

Method:
- Compute using multiple passes through the collection of all pages
- Approximate PageRank gets adjusted over time
- Start: equal probability for all pages
PageRank, simplified method

PageRank value can be seen as: Probability that a person, clicking around at random, will land on this page

More likely that they will end up on a page that many other pages link to

Method:
- Compute using multiple passes through the collection of all pages
- Approximate PageRank gets adjusted over time
- Start: equal probability for all pages

\[
P(A) = \frac{P(B)}{\text{deg}(B)} + \frac{P(C)}{\text{deg}(C)} + \frac{P(D)}{\text{deg}(D)} = \frac{P(B)}{2} + \frac{P(C)}{1} + \frac{P(D)}{3} = 0.458
\]
PageRank, simplified method

PageRank value can be seen as: Probability that a person, clicking around at random, will land on this page

More likely that they will end up on a page that many other pages link to

Method:
- Compute using multiple passes through the collection of all pages
- Approximate PageRank gets adjusted over time
- Start: equal probability for all pages. After re-computing for all nodes:
PageRank, simplified method

PageRank value can be seen as: Probability that a person, clicking around at random, will land on this page

More likely that they will end up on a page that many other pages link to

Method:
- Compute using multiple passes through the collection of all pages
- Approximate PageRank gets adjusted over time
- Start: equal probability for all pages. After re-computing for all nodes:

Re-computing A now:

\[
P(A) = \frac{P(B)}{\text{deg}(B)} + \frac{P(C)}{\text{deg}(C)} + \frac{P(D)}{\text{deg}(D)} =
\]

\[
P(B)/2 + P(C)/1 + P(D)/3 = 0.08/2 + 0.21 + 0.25/3 = 0.33
\]
PageRank, simplified method

PageRank value can be seen as: Probability that a person, clicking around at random, will land on this page

More likely that they will end up on a page that many other pages link to

Method:
- Compute using multiple passes through the collection of all pages
- Approximate PageRank gets adjusted over time
- Start: equal probability for all pages. After re-computing for all nodes:
PageRank, the rest of the story

PageRank values do not sum to one after all (not probabilities), but to some value N.

"Damping factor": The random web surfer gets tired at some point and stops surfing. At any point there is a probability d that the surfer will stop now.
HTML, the Hyper-Text Mark-up Language

Language that webpages have often been written in

Specifies layout, but does not specify everything about it

- example: HTML code specifies that some text is a “header”, browser selects font size and bolding

In some browsers, you can “View Page Source”, the HTML code of a webpage. Here are example web pages where this works well:

What is computational linguistics? http://www.coli.uni-saarland.de/~hansu/what_is_cl.html

Or the main page of Wikipedia, https://en.wikipedia.org/wiki/Main_Page

In Firefox, to view the source, right-click on the webpage, and choose “View Page Source”.

HTML

HTML consists of **HTML tags:**

- `<li>` is for a list item, `</li>` is the end of the list item
- `<ul>` is the beginning of a list, `</ul>` is the end of the list
- `<b>` starts boldface text, and `</b>` ends it

General pattern: `< MNEMONIC_LABEL >` for starting some formatting, `</MNEMONIC_LABEL>` to end it.

Many tags have obvious effects on the display of the page (like the ones above)

Others do not: `<META>` provides information for web crawlers about the content of a page
HTML

Formatting hyperlinks in HTML:

\<a href="/wiki/Portal:Biography">Biography</a\>

This extends the pattern we saw earlier:

\<LABEL ATTRIBUTE=VALUE> .... </LABEL\>

Can you spot more examples of this on one of the pages we viewed?
HTML: Formatting versus content type

HTML: a mixture of straight formatting commands and commands that characterize types of content

Formatting commands:
- `<i>...</i>` italics,
- `<b>...</b>` boldface

Type of content:
- `<title>`
- `<a href=...>` hyperlink
- `<ul>` list

For the latter kind, the browser decides how to format

XML: Flexible representation of content type only
**XML: What does it look like?**

```xml
<note>
  <to>Kim</to>
  <from>Sandy</from>
  <heading>Shopping</heading>
  <body>Don’t forget to bring gummy bears. </body>
</note>
```
XML: Text

A pair of elements can enclose another pair of elements, or text

\[
\text{<body>Don’t forget to bring gummy bears. </body>}
\]

Representations are predefined for some special characters

- à: &agrave;
- ö: &oumlaut;
- ~
XML: another example

```xml
<books>
  <book pages="232" isbn="78-1-4051-8305-5" price="34.95">
    <title>Language and Computers</title>
    <authors>
      <author>Markus Dickinson</author>
      <author>Chris Brew</author>
      <author>Detmar Meurers</author>
    </authors>
  </book>
</books>
```
XML: Elements and attributes

Elements:
- `<author>...</author>`

Leaf elements:
- `<word/>` (label then slash, not slash then label)

Attributes:
- `<book pages="232" isbn="78-1-4051-8305-5" price="34.95">`
- Attribute-value pairs, value encased in “”
XML: what does the notation mean?

<something>…</something>
is like a set of labeled brackets [ something ]

A set of labeled brackets can contain another set of labeled brackets

A different way of visualizing this: as a tree
  ◦ https://codebeautify.org/xmlviewer
XML: what can I do with this?

You can design your own set of labeled brackets:
- List of books
- Menu for a restaurant
- Formatting for blogs
- Formatting for text documents (used in Word)

Describe arbitrary tree-shaped annotation
XML: Applications

News feeds
Office applications
Industry data standards
...and a common data exchange format in general
Semantic Web, once more

Vision: Common framework for data re-use across sites

A web of data that can be processed by machines

How to describe this data?

Resource Description Framework (RDF), a general framework for modeling data in web pages

Can be implemented in XML

◦ https://www.w3schools.com/XML/xml_rdf.asp
Regular expressions

Reminder: Boolean expressions: combining pieces using AND, OR, NOT

Sometimes this is not enough for describing complex patterns of text
- I want to find addresses with a zip code starting with 911 (around Pasadena, CA). But I don’t want all occurrences of emergency phone numbers in the document
- I want to find all .utexas.edu email addresses in a long document

To match complex patterns in text, we can use regular expressions.
What are regular expressions?

A regular expression is a compact description of a set of strings (the ones that match the expression).

In formal language theory, such a set of strings is called a “language”.

Compact descriptions of formal languages can be classified by the complexity of the string sets they describe. There are more complex ones that cannot be described by regular expressions.

So some patterns cannot be described with regular expressions (more on this later)

A formal language need not be a natural language. But we can use it to specify patterns occurring in natural languages.
What tools use regular expressions?

Some Unix tools:
- grep, sed

Some editors
- emacs, jEdit

Programming languages

Implementations are very efficient

The exact syntax of regular expressions differs across tools

Testing regular expressions: https://regexr.com/
Regular expression syntax

You can search for:

- Strings of literal characters:
  - kitten
  - 30 years?!
  - 911
  - natural language

- Disjunction:
  - normal disjunction with “|”:
    - devoured|ate
    - famil(y|ies)
  - character classes:
    - [Tt]he
    - bec[oa]me
  - ranges:
    - [A-Z][a-z] [0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]

- negation:
  - [^a] any symbol but a
  - [^A-z0-9] (not an uppercase letter or number)
Regular expression syntax

- Counters and optionality:
  - Optionality: `sings?` (will match either “sing” or “sings”). `colou?r`
  - Any number of occurrences: `*` (“Kleene star”)
    - `[0-9]* years`
  - At least one occurrence: `+` (“Kleene plus”)
    - `[0-9]+ dollars`
- Wildcard for any character: `. (period)`
  - `beg.n` will match “begin”, “began”, “beg!n”, “beg never”
Regular expression syntax

◦ To literally match a character with a special meaning, such as * + . [ ] ( ): “escape” them by putting a backslash before them.
  ◦ \. matches an actual period

◦ Shortcuts for frequent character classes:
  ◦ \w uppercase or lowercase letter or digit
  ◦ \d digit
  ◦ \s space

◦ Beginning and end of word: \b
  To match “begin” and “began” but not “beg never”, use “beg.n\b”

Trying it out: https://regexr.com/