Abstraction of search engine architecture
Web crawler

- An automatic program that systematically browses the web for the purpose of Web content indexing and updating
  - Synonyms: spider, robot, bot
How does it work

• In pseudo code

```python
Def Crawler(entry_point) {
    URL_list = [entry_point]
    while (len(URL_list)>0) {
        URL = URL_list.pop();
        if (isVisited(URL) or !isLegal(URL) or !checkRobotsTxt(URL))
            continue;
        HTML = URL.open();
        for (anchor in HTML.listOfAnchors()) {
            URL_list .append(anchor);
        }
        setVisited(URL);
        insertToIndex(HTML);
    }
}
```

Which page to visit next?
Is it visited already?
Or shall we visit it again?
Is the access granted?
Visiting strategy

• Breadth first
  – Uniformly explore from the entry page
  – Memorize all nodes on the previous level
  – As shown in pseudo code

• Depth first
  – Explore the web by branch
  – Biased crawling given the web is not a tree structure

• Focused crawling
  – Prioritize the new links by predefined strategies
Focused crawling

• Prioritize the visiting sequence of the web
  – The size of Web is too large for a crawler (even Google) to completely cover
  – Not all documents are equally important
  – Emphasize more on the high-quality documents
• Maximize weighted coverage

In 1999, no search engine indexed more than 16% of the Web
In 2005, large-scale search engines index no more than 40-70% of the indexable Web

Weighted coverage till time $t$
Importance of page $p$
Pages crawled till time $t$
Focused crawling

• Prioritize by in-degree [Cho et al. WWW’98]
  – The page with the highest number of incoming hyperlinks from previously crawled is crawled next

• Prioritize by PageRank [Abiteboul et al. WWW’07, Cho and Uri VLDB’07]
  – Breadth-first in early stage, then compute/approximate PageRank periodically
  – More consistent with search relevance [Fetterly et al. SIGIR’09]
Focused crawling

• Prioritize by topical relevance
  – In vertical search, only crawl relevant pages [De et al. WWW’94]
    • E.g., restaurant search engine should only crawl restaurant pages
  – Estimate the similarity to current page by anchor text or text near anchor [Hersovici et al. WWW’98]
  – User given taxonomy or topical classifier [Chakrabarti et al. WWW’98]
Avoid duplicate visit

• Given web is a graph rather than a tree, avoid loop in crawling is important
• How to check
  – trie or hash table
• What to check
  – URL: must be normalized, not necessarily can avoid all duplication
    • http://dl.acm.org/event.cfm?id=RE160&CFID=516168213&CFTOKEN=99036335
    • http://dl.acm.org/event.cfm?id=RE160
  – Page: minor change might cause misfire
    • Timestamp, data center ID change in HTML
Politeness policy

• Crawlers can retrieve data much quicker and in greater depth than human searchers

• Costs of using Web crawlers
  – Network resources
  – Server overload

• Robots exclusion protocol
  – Examples: CNN, UVa
Robot exclusion protocol examples

• Exclude specific directories:
  
  User-agent: *
  Disallow: /tmp/
  Disallow: /cgi-bin/
  Disallow: /users/paranoid/

• Exclude a specific robot:
  
  User-agent: GoogleBot
  Disallow: /

• Allow a specific robot:
  
  User-agent: GoogleBot
  Disallow:

  User-agent: *
  Disallow: /
Analyze crawled web pages

• What you care from the crawled web pages
Analyze crawled web pages

• What machine gets from the crawled web pages
Basic text analysis techniques

- Need to analyze and index the crawled web pages
  - Extract informative content from HTML
  - Build machine accessible data representation
HTML parsing

• Generally difficult due to the free style of HTML

• Solutions
  – Shallow parsing
    • Remove all HTML tags
    • Only keep text between <title></title> and <p></p>
  – Automatic wrapper generation [Crescenzi et al. VLDB’01]
    • Wrapper: regular expression for HTML tags’ combination
    • Inductive reasoning from examples
  – Visual parsing [Yang and Zhang DAR’01]
    • Frequent pattern mining of visually similar HTML blocks
HTML parsing

• **jsoup**
  – Java-based HTML parser
    • Scrape and parse HTML from a URL, file, or string to DOM tree

• Python version: **Beautiful Soup**
  – Scrape and parse HTML from a URL, file, or string to DOM tree
    • find and extract data, using DOM traversal or CSS selectors
      – `children()`, `parent()`, `siblings()`
      – `getElementsByClass()`, `getElementsByAttributeValue()`
How to represent a document

- Represent by a string?

/html> Crowds in Liverpool to Mark 10th Anniversary of John Lennon's Death<//html>
/html> LIVERPOOL, England (AP) <//html>
/html> 

- Dozens of fans of rock legend and former Beatle John Lennon gathered in the snow on a windy Saturday for a ceremony marking the 10th anniversary of his death. Liverpool's mayor, Dorothy Gavlin, led Lennon devotees who laid wreaths at the foot of a bronze statue of The Beatles in the city's Cavern Walks shopping center. The center was built on the original site of the Cavern Club, made famous when The Beatles played there in the 1960s, and has become a place of pilgrimage. "Give peace a chance," the title of one of singer-songwriter Lennon's greatest hits, was the theme for the day.

... Lennon and his wife, Yoko Ono, were returning to their apartment in New York's Dakota apartment building after a recording session on Dec. 8, 1980, when Lennon was shot to death by Mark David Chapman, a deranged fan to whom Lennon had given his autograph only hours before. Lennon was 40. A spokesman for the Lennon family said Ms. Ono and the couple's son, Sean, were in Europe and would spend the anniversary privately.

... Peebles said late in 1980 that Lennon had just recovered from a period when he had "gone off the rails" and his relationship with Ms. Ono had suffered. "But (when I saw him) they'd had the baby, Sean had been born, and everything was great."<//html>

-- Bag-of-Words representation!
Tokenization

• Break a stream of text into meaningful units
  – Tokens: words, phrases, symbols
  • Input: It’s not straight-forward to perform so-called “tokenization.”
  • Output(1): 'It’s', 'not', 'straight-forward', 'to', 'perform', 'so-called', ‘“tokenization.”’
  • Output(2): 'It', '', 's', 'not', 'straight', '-', 'forward', 'to', 'perform', 'so', '-', 'called', '', 'tokenization', '.', ‘’

  – Definition depends on language, corpus, or even context
Tokenization

• Solutions
  – Regular expression
    • \[w\]+: so-called -> ‘so’, ‘called’
    • \[S\]+: It’s -> ‘It’s’ instead of ‘It’, “s’
  – Statistical methods
    • Explore rich features to decide where is the boundary of a word
      – Apache OpenNLP (http://opennlp.apache.org/)
    • Online Demo
      – UIUC (http://cogcomp.cs.illinois.edu/curator/demo/index.html)
Full text indexing

• Bag-of-Words representation
  – Doc1: Information retrieval is helpful for everyone.
  – Doc2: Helpful information is retrieved for you.

<table>
<thead>
<tr>
<th></th>
<th>information</th>
<th>retrieval</th>
<th>retrieved</th>
<th>is</th>
<th>helpful</th>
<th>for</th>
<th>you</th>
<th>everyone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Doc2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Word-document adjacency matrix
Full text indexing

• Bag-of-Words representation
  – Assumption: word is independent from each other
  – Pros: simple
  – Cons: grammar and order are missing
  – *(Used to be)* The most frequently used document representation
    • Image, speech, gene sequence
Full text indexing

• Improved Bag-of-Words representation
  – N-grams: a contiguous sequence of n items from a given sequence of text
    • E.g., Information retrieval is helpful for everyone
    • Bigrams: ‘information_retrieval’, ‘retrieval_is’, ‘is_helpful’, ‘helpful_for’, ‘for_everyone’
  – Pros: capture local dependency and order
  – Cons: purely statistical view, increase vocabulary size $O(V^N)$
Full text indexing

• Index document with all the occurring word
  – Pros
    • Preserve all information in the text (hopefully)
    • Fully automatic
  – Cons
    • Vocabulary gap: cars v.s., car
    • Large storage: e.g., in N-grams $O(V^N)$
  – Solution
    • Construct controlled vocabulary
Statistical property of language

• Zipf’s law
  – Frequency of any word is inversely proportional to its rank in the frequency table.
  – Formally:
    \[ f(k) = \frac{1}{k^s} \]
    where \( k \) is rank of the word, \( N \) is the vocabulary size, and \( s \) is a language-specific parameter.

In the Brown Corpus of American English text, the word "the" is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences; the second-place word "of" accounts for slightly over 3.5% of words.
Zipf’s law tells us

• Head words may take large portion of occurrence, but they are semantically meaningless
  – E.g., the, a, an, we, do, to
• Tail words take major portion of vocabulary, but they rarely occur in documents
  – E.g., dextrosinistral
• The rest is most representative
  – To be included in the controlled vocabulary
Automatic text indexing

- Remove non-informative words
- Remove 1s
- Remove 0s
- Remove rare words
## Stopwords

Useless words for query/document analysis.

- Not all words are informative.
- Remove such words to reduce vocabulary size.
- No universal definition.
- Risk: break the original meaning and structure of text.

### Examples
- E.g., this is not a good option -> option
to be or not to be -> null

### Nouns
1. time
2. person
3. year
4. way
5. day
6. thing
7. man
8. world
9. life
10. hand
11. part
12. child
13. eye
14. woman
15. place
16. work
17. week
18. case
19. point
20. government
21. company
22. number
23. group
24. problem
25. fact

### Verbs
1. be
2. have
3. do
4. say
5. get
6. make
7. go
8. know
9. take
10. see
11. come
12. think
13. look
14. want
15. give
16. use
17. find
18. tell
19. ask
20. work
21. seem
22. feel
23. try
24. leave
25. call

### Adjectives
1. good
2. new
3. first
4. last
5. long
6. great
7. little
8. own
9. other
10. old
11. right
12. big
13. high
14. different
15. small
16. large
17. next
18. early
19. young
20. important
21. few
22. public
23. bad
24. same
25. able

### Prepositions
1. to
2. of
3. in
4. for
5. on
6. with
7. at
8. by
9. from
10. up
11. about
12. into
13. over
14. after
15. beneath
16. under
17. above
18. an
19. will
20. my
21. one
22. all
23. would
24. there
25. their

---

The OEC: Facts about the language
Normalization

• Convert different forms of a word to normalized form in the vocabulary
  – U.S.A -> USA, St. Louis -> Saint Louis

• Solution
  – Rule-based
    • Delete periods and hyphens
    • All in lower case
  – Dictionary-based
    • Construct equivalent class
      – Car -> “automobile, vehicle”
      – Mobile phone -> “cellphone”
Stemming

• Reduce inflected or derived words to their root form
  – Plurals, adverbs, inflected word forms
    • E.g., ladies -> ladi, referring -> refer, forgotten -> forget
  – Bridge the vocabulary gap
  – Risk: lose precise meaning of the word
    • E.g., lay -> lie (a false statement? or be in a horizontal position?)
  – Solutions (for English)
    • Porter stemmer: pattern of vowel-consonant sequence
    • Krovetz Stemmer: morphological rules
Abstraction of search engine architecture

Indexed corpus

Crawler

Doc Analyzer

Doc Representation

1. Visiting strategy
2. Avoid duplicated visit
3. Re-visit policy

1. HTML parsing
2. Tokenization
3. Stemming/normalization
4. Stopword/controlled vocabulary filter

BagOfWord representation!
Automatic text indexing

• In modern search engine
  – No stemming or stopword removal, since computation and storage are no longer the major concern
  – More advanced NLP techniques are applied
    • Named entity recognition
      – E.g., people, location and organization
    • Dependency parsing

Query: “to be or not to be”
What you should know

- Basic techniques for crawling
- Zipf’s law
- Procedures for automatic text indexing
- Bag-of-Words document representation
Today’s reading

- Introduction to Information Retrieval
  - Chapter 20: Web crawling and indexes
    - Section 20.1, Overview
    - Section 20.2, Crawling
  - Chapter 2: The term vocabulary and postings lists
    - Section 2.2, Determining the vocabulary of terms
  - Chapter 5: Index compression
    - Section 5.1, Statistical properties of terms in information retrieval
Reference I

Reference II

