CS: Pod of Delight

Week 13: Search

Logistics

- How is everyone doing?
- Semester is almost over!
- No Pod next week, go home enjoy thanksgiving!
- The week after, party!
- Then you’re done! Off you go into the real world!

So you want to build a search engine?

- Search engines have four main problems
  - Crawling
  - Index
  - Search
  - Ranking

Crawling

- The internet is a massive jungle of links
- Goal: find them all
- How?
    - Follow every link in every page
    - Exponential
- Problems:
    - Where do you start?
    - How do you know if you've already seen a page? (cycles)

Crawling: Implemented

- Need a way to get webpage
  - All webpages are nothing but some text (html/css/js) and media (images/flash/videos/music)
- Need a way to parse source code
  - Parse the html DOM tree, and provide methods for traversing it, querying it, etc…
  - Jsoup

Crawling: Problems

- Where do you start?
  - Google originally started crawling on Larry Page's Stanford personal website
- How do you prevent cycles?
  - Hashtable
  - Bloom filter

Index

- So you found the internet, now what?
- Store what you found
- Efficient representation of the content so you can query it

Inverted Index

- Maps words to locations
  - Map words to documents
    - For a given word map to which documents it can be found in
- How to store?
  - Hashmap
  - B-tree

How to build it?

- Parse every word of content
- Map the word to the document where you found it
- What if there are multiple documents with that word
    - Map to a set
- What if there are multiple occurrences of that word in the document?
    - Doesn’t matter!
    - Or does it?

What if you want to store phrases?

- Could map all word tuples, or triplets!
- Too much space!
- Instead map word to document, and place in document
  - You store all occurrences
  - Advantages:
    - Can search for where in document word is!
    - Can perform phrase searches!

So searching

- You have your index, awesome!
- How do you search it?
    - Look up a word in the index, boom!
- What if you want to search for multiple words
    - Look up all, return the intersection, boom!
- What if you want to search for the union of words?
    - Look up all, return the intersection, boom!
- What if you want to search for the union or intersections?
    - You get the point

Humans

- Biggest problem: English
- Language is imprecise
- Have to parse an English query
- Can have explicit and implicit ANDs and ORs
- Need to parse queries like
    - “the duck is awesome”
    - the duck is awesome
    - the | duck | is | awesome
    - (the duck) | (is awesome)
    - the duck | “is awesome”

Recursive descent parser

- First define a context-free-grammar (CFG) for your language
- Then start parsing it top-down, consuming input as it matches
- Keep parsing until either all the input is consumed or you encounter an error (input doesn’t match what you expected)

RDP: Implemented

- First want to tokenize your input
- StringTokenizer
- Deal with whitespace (either too much, too little, etc)
- Then build a recursive descent parser
  - Start at the start state, build methods to consume input for each of the non-terminal states
  - Store the query in some representation
  - Probably want a tree!

Searching

- You have your query tree
- Then perform it, keeping a list of pages as you go
- Little tricks and optimizations
    - If you have intersect, only search the result of the first query

Ranking

- Cool! You have your list of webpages
- How do you return them? How do you rank?
- Need a way to score a match
  - Number of word occurrences
  - Where in the document the word appears
    - Is it in the title? Big text? small text? colored? underlined?
  - How close two words appear to each other?
  - Exact match vs approximate match?

PageRank

- As you crawl, store all websites that point back to a given website (backlinks)
- The more a website is linked to, the better the content
- Rank higher
- Links from higher ranked websites are more meaningful

Results

- Take all the matches you found
- Score all of them
- Sort them
- Return them to the user
- Profit???

Good luck :)