Artificial Intelligence Programming

Comments on Project 1

Chris Brooks

Department of Computer Science
University of San Francisco

9-2: Web crawlers

- A web crawler is a program that crawls the web.
- Starts with a set of root pages.
- Extracts links to other pages.
- Fetch each of those pages.
- Extract their links and repeat.

9-3: Web crawling as search

- We can consider web crawling as a form of search.
- Documents are states.
- By controlling the order in which new pages are explored, we can achieve different search strategies (BFS, DFS, etc.)

9-4: Focused crawlers

- A focused crawler is a crawler that looks for pages that satisfy a particular set of criteria.
- In this project, we'll build a focused crawler.
- Two methods for users to indicate criteria:
  - Keywords
  - Example pages

9-5: The Crawler in a nutshell

9-7: How to do this?

- You'll want to build a helper class that can parse HTML.
- I recommend subclassing the SGMLParser class.
  - Dive Into Python shows how to do this
  - You may have done this in homework 1.
- This will let you extract anchors, headings, title and text.

9-8: Filtering content

- You'll then remove non-useful content
  - Any words with a non-alphabetic character
  - Any stop words.

9-9: Outward Links

- Extract the URLs from all anchor tags and add them to the list of outward links.
- You may discard links to docs other than text or HTML (can do this either here or later).
- You'll also need to deal with relative URLs.
  - No 'http' or hostname, only a relative path to the document.
  - Before adding them to the list of outward links, fix them to work as standalone URLs.

9-10: Hints and Issues

- You'll need to deal gracefully with malformed HTML.
- You'll need to deal gracefully with remote server timeouts.
- You may need to deal with authentication.
- Remember Python's motto: "Batteries included"

9-11: the FocusedCrawler class

- The FocusedCrawler will manage the process of creating Documents according to their expected value.
  - Much like the 'search()' function in homework 3.
- Maintain a heapq of (score, URL) tuples.
- Dequeue the best URL, check to see if we've visited it before, create a Document, score it, enqueue (score, URL) tuples for its outward Links.

9-12: Closed dictionary

- We'll use a simple closed list (dictionary, actually)
- Keep a dictionary of previously visited URLs.
- Before creating a new document, look to see if that URL has been previously visited.
- What are some limitations of this approach?

9-13: Dealing with robots.txt

- Your crawler should also repsect robots.txt
- Before fetching a URL, use the robotparser module to determine if that URL is allowed.

9-14: Scoring a Document

- Now we know how to fetch Documents, process them, and collect their outward links.
- How to determine the 'goodness' of a document?
- Assumption: the estimated score of a document will be determined by the quality of documents that link to it.

9-15: Query-driven search

- To begin, we'll implement a KeywordQueryScorer.
- This will allow the user to specify combinations of keywords that should appear in documents that are collected.
- We'll use the following query format:
  - "word1 word2 (word3 word4 word5) word6 word7 (word8 word9) ..."
  - Where combinations inside parens are OR clauses, and other words are ANDed together.

9-16: Query-driven search

- To score a document, count the number of ANDed words that are contained within the document, along with the number of ORed clauses that have any word in the document.
- To normalize this, we then divide by the length of the query, which is the number of ANDed terms plus the number of ORed clauses.
- This gives us a number between 0 and 1.

9-17: Example

- Let's say our document contains 'cat cat dog dog bunny squirrel fish'
- Our query is 'dog (cat snake) bunny lion'
- Our score is 3/4 = 0.75.
- Note that multiple occurrences of a query term count as one match.

9-18: Searching by topic

- Our KeywordQuery scorer has some weaknesses.
  - Doesn't account for likelihood of words occurring.
  - Doesn't count frequency of a query's occurrence
  - Requires our users to be able to write Boolean queries.