CS 537 Handout Dec 10

- Goal:
    - Enter search phrase
    - Get high quality, relevant results *quickly*
- Stages:
    - Crawling
    - Computing PageRank
    - Computing inverted index
    - Searching index
- Input to search engine: crawled web pages
    - Crawler takes “snapshot” of websites
    - Snapshots are then indexed
        - MapReduce often used to do this
- Output: search results
    - Results obtained from indexes over snapshots
- What determines quality of results?
    - The snapshots
    - The indexes
- Crawling
    - Maintain a list of pages to crawl
    - Grab item from list, save page
        - Convert domain name to IP address
            - Use DNS caching to speed this up
        - Fetch page from server at IP address
    - Identify links from saved page, add to list
        - What if saved page had link back to itself?
            - Avoid via heuristics
    - robots.txt can used to tell crawlers whats off-limit
- Indexes
    - A lot of web pages out there
    - How do we determine what are “good” pages?
    - PageRank
        - count backlinks to a webpage
        - not all backlinks are equal: cnn.com backlink is worth more
            - if you are a “good” website, ur link matters more
            - Problem: recursive definition
            - Solution:
                - compute new rank
                - update existing rank
                - compute change from last iteration
                - repeat until change is below threshold

- Personalized Search
    - Quality is subjective
    - Bias algorithm towards pages relevant to user
        - Such pages get higher rank
- Inverted Index
    - Important != Relevant
    - Ascertain relevancy from page content
    - Index over page content
    - Given a word, find all documents containing word
    - docID to each unique page
    - wordID for each unique word on web (not document)
    - Forward index: docID -> list of wordIDs
    - Inverted index: wordID-> list of docIDs
    - Include text of hyperlinks pointing to each docID
        - cat picture
    - Include other extra information like word position, text type etc
    - Run MapReduce jobs to construct inverted index
        - Mapper: read words from files => (word, file name)
        - Reduce: (word, list of filenames)
- Retrieving results
    - Query converted into wordIDs
    - docIDs for each wordID retrieved
    - These sets can be unioned or intersected
    - Search for words near other in document
    - Resource intensive!

Questions

1. How does Google handle pages that only refer to each other?
2. What does the quality of search results depend on?
3. How does Google personalize search for each user?
4. The formulation for page rank is recursive. Naively, we could keep updating the page rank forever. How does Google solve this problem?
5. How does the crawling process use MapReduce?
6. For PageRank, why is just using the number of inbound links a bad idea?
7. List the steps involved in crawling a URL.