Team Logistics
09/18/2020

Team Letter: G
Team Members:

Neha Deshpande
Bowen Gong*
Callum Hauber
Justin Mai
Matthew Scanlon

- **Team meeting schedule:** W/F 10:10-12:00
- **How we will run team meetings:** Team meetings will begin going around in a circle and doing the following tasks:
    - Discuss the progress you've made this week
    - Discuss any issues you've had or still have and what you tried to troubleshoot it (document it as you troubleshoot and log it to the issue tracker)
    - Share insights
    - Discuss any related topics including deadlines and where to learn more about your issue
    - One member during the meeting will take notes, which will then be uploaded to the shared repository
- **Where are we storing our project work:** We will be using a GitHub repository, which can be accessed here:
    - https://github.com/boweng520/LSPT_TeamG

Thinking about Search

1. **Identify at least three examples of where search is not well implemented**
    a. Reddit Search results look for exact matches, not implied matches + users don't always title things correctly for SEO
    b. OSX finder does not have ability to not search through Library (OS) files resulting in hectic search results when looking for a .doc file in an unknown location
    c. Facebook messages search results are indexed in "linked list" format in which each keyword search only has access to what is in front and behind it
2. **Identify at least three example searches in a search engine that produce poor results**
    a. Google search (Computer) Mouse → Mice (The animal)
    b. .py in finder search yields A LOT of unwanted results (OS files)
    c. Tesla Cat → Tesla Car (not even a did you mean tesla car)
3. **Choose the top three search engine sub-components that you'd like to work on**
    a. Text Transformation
    b. Ranking
    c. Text Acquisition/Crawling

Success Metrics

- Precision
    - The results should be what the user was looking for
    - RPI-specific results
    - Manually review documents of search terms of average length to determine relevance
    - Precision = # of relevant items / # of retrieved items
- Recall
    - Most or all relevant items should be retrieved
    - Recall = # of relevant items retrieved / # of relevant items
    - Manually check a select group of documents with search terms
- Scalability
    - What happens when 10, 100, 1000 users are searching at the same time.
    - Make everything as efficient as possible(Memory, performance, etc…)
    - Lots of different websites to search through should not affect the performance too much

Initial Requirements

- Text Transformation
    - Identify the documents we want to store
    - The number of crawled documents should cover every aspect of the topic
    - Documents should have identifiable and quantifiable features
    - Transform documents into index terms
    - Removes common terms
    - Able to recognize words
    - Able to recognize structural elements like heading, title, etc
    - Recognize Markup Language to identify structure
    - Search terms should also return documents related to the word groups (i.e searching for “computer” would also return “computers”, “computing”, etc.)
    - Identify metadata for documents (i.e a document about chicken should have the label “chicken” to make searching faster)
    - Able to extract “important” words like RPI
    - Able to recognize typos/slang and still cache them unless the typo is a common term
- Ranking
    - User can specify to rank by most recent
    - User can specify to rank by most relevant
    - Rank should accurately depict how relevant the document is
    - The rank of each search result is calculated using a ranking algorithm
    - The effectiveness of a ranking algorithm should be measurable and can be compared with alternatives.
    - Ranking function exists to take document features as input and produces a score.
    - The document with the highest score will come first.

    - Cache results such that the ranking can be reused.
    - Rankings should be updated when documents are updated
    - Rankings should be measured and tuned to be more effective
        - I.e number of times the top ranked results are clicked / total links clicked
    - Irrelevant documents should not even be ranked
    - User can specify to search by title of the document rather than the contents
- Text Acquisition/Crawling
    - Identify and acquire documents
    - Able to recognize typos
    - Able to recognize synonyms
    - Has measures to assess the freshness of a document
    - Keep documents up to date and maintain the freshness of the document collections
    - Clean the stale copy when the collection is refreshed
    - Must efficiently find a large number of web pages
    - Converting a variety of documents into a consistent format
    - Able to convert different encodings
    - An efficient database system must be used
    - The crawler should have multiple threads to work simultaneously
    - Use politeness policies when crawling
        - Do not crawl too many pages at the same time
        - Wait for some time between two crawls