cse.ust.hk

crawler

1. fetch with BFS (use HTML parser)
2. extract links to recursively process more pages.
3. build the file structure containing the parent/child link relation.

pages

indexer

defined in modals

1. remove all stop words from the file.
2. transform words into stems using Porter's algorithm (Perform stemming to the words)
3. Store the stems into inverted_titles and inverted_bodies (same type)
4. support phrase search e.g. "hong kong"

inverted files

crawler service

indexer service uses

DAO Layer

Database Manager

mapdb

app

provide APIs:

- Method: GET
- Path: api/search
- Parameter: 1. query (user's search query)
  2. limit (# of returned result)
  3. offset (skip previous
- Return: pageId, title, url, lastModified, snippet, score, pageSize, keywords, parentLinks, childLinks

- Method: GET
- Path: api/health
- Parameter: None
- Return: status (Boolean)

Search

Provide search service to app.

1. Retrieve inverted indices from DAO. stem and remove stopword?
2. Compare list of query terms against the inverted indices.
3. rank according to a vector-space model $\alpha \approx 0.7$
   i.e. Final Score = $\alpha \cdot$ Term Score + $(1-\alpha) \cdot$ PageRank Score

Term-based:

$$\frac{tf \times idf}{\max(tf)}$$

Link-based:

$$PR(p_i) = (1-d) + d \cdot \sum_{p_j \in B(p_i)} \frac{PR(p_j)}{L(p_j)}, \quad d = 0.85$$

title matches boost the score

Database Schemas

inverted_titles, inverted_bodies

Word-ID → {Page-ID, Freq, [pos]}

dummy

lightweight forward index

page ID → {keywords}

↳ top 10 stemmed keywords only.

one to one map

URL ↔ pageID

word ↔ wordID

many-to-many

parent → child

adjacency list representing the link structure

Page properties

PageID → {title, URL, last_modified, size, content}

↳ we can check this before

phase 1 tester program

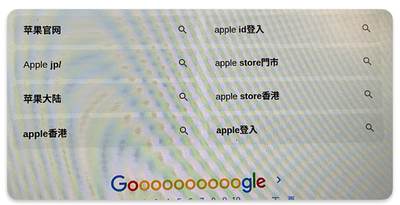
↓

txt

Bonus

F 1. relevance feedback

F 2. Allow user to:



F 3. UI-friendly (Theme: neuromorphism, colour: bg: white, primary colour: #9966cc (Amethyst), secondary colour: #ffff88 (yellow), font: Poppins, #f5f5f5 for highlighting)

F 4. Query history (up to 5 Query)

5. Other search engine func:
   1. word pos B
   2. Result sort by time F

B 6. Ranking (Page rank + Term based)

B 7. do caching, threading (speed)

FIFO, LFU, LRU?