Information retrieval

Lab

Purpose

- Web crawling
- Basic NLP techniques
- Store and query web information
- Ranked query processor

Task description – Android app search engine

- The number of apps developed for the web and the mobile devices is increasing tremendously.
- Therefore efficient and effective search engine for apps is of vital importance.
- In this lab, you will learn to construct a simple Android app search engine in Python.
- For each App there is a piece of text describing it.
- Your task is to extract this kind of information from the specific websites and build inverted index over the documents. Moreover, the term frequency and document frequency are computed and stored in the inverted index. Based on this index structure, the search engine should take the keyword query as input and evaluate the query over the index using the ranked query algorithm we learned from the class.

Instructions

- Step 1
  - Access web site such as Google Play and extract at least 1000 app descriptions (No constraint on what kind of app)
- Step 2
  - Preprocess app descriptions: tokenization, normalization, etc.
  - Compute and store tf, idf in the inverted index.
- Step 3
  - Write a ranked query processor

Step 1 – example approach

- Access Google Play app section
  (e.g. https://play.google.com/store/apps/top)
  (e.g. https://play.google.com/store/apps/category/GAME/collection/topselling_free)

- Extract app urls and add them to the list of app links
  - (e.g. using regular expression `"href=\"/store/apps/details.*?\""`)
  - re.findall(pattern, string)
- Access links one by one – `urllib.request.urlopen(url).read().decode('utf8')`
  - Add `&hl=en` to acquire only descriptions (some apps in Swedish will be returned)
  - Extract description
    - (e.g. using regular expression
      `"itemprop=\"description.*?\">.*?<div jsname=\".*?\">.*?</div>"`)
  - Extract all app urls on this webpage and add them to the list of app links (be careful with reviews)
  - Repeat until *x* descriptions extracted (minimum 1000)
- Store the description text in the files

Step 2 – example approach

Preprocess descriptions:

- Remove non alpha-numeric characters
- Tokenize
- Lowercase words
- Remove stopwords
- Stem

Construct Vector Model

- Compute tf and idf (numpy or methods from scikit-learn module)

Step 3 – example approach

- Preprocess search string (same approach as before)
- Compute tf-idf
- Compute an angle
- Extract top k elements

Report

- A short report describing:
  - The implementation
    - Web crawling
    - What is the input and how you store them
    - How is the angle calculated, etc.
  - Some test runs

Notes

- Test your code fragments and functions as soon as you achieve
  - Make sure you have correctly obtained at least 1000 apps' urls first
  - Then access the app webpage
- Avoid call crawling function after you have saved app description in local files
- Run
  - *source /home/TDDE16/labs/environment/bin/activate*