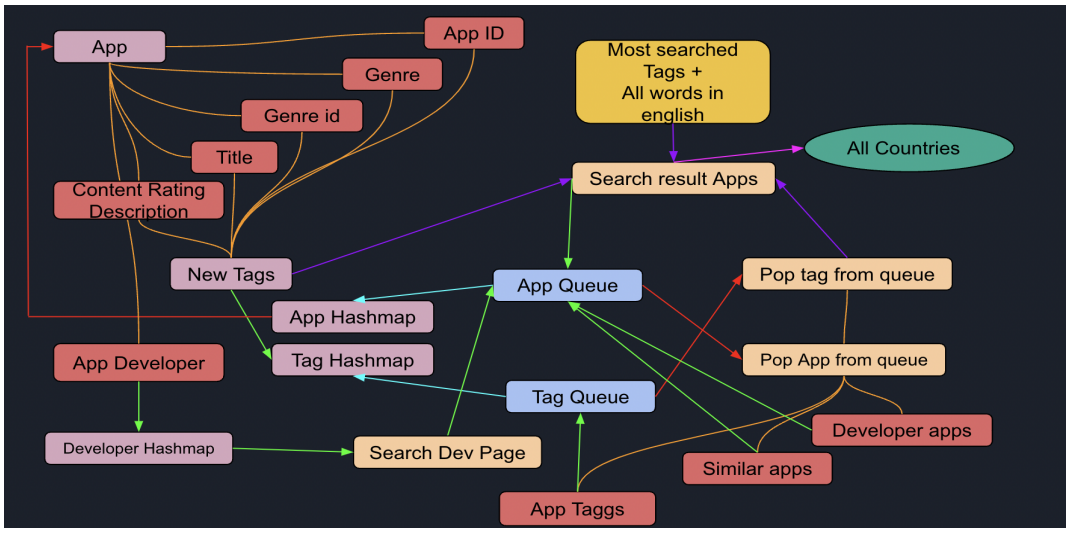
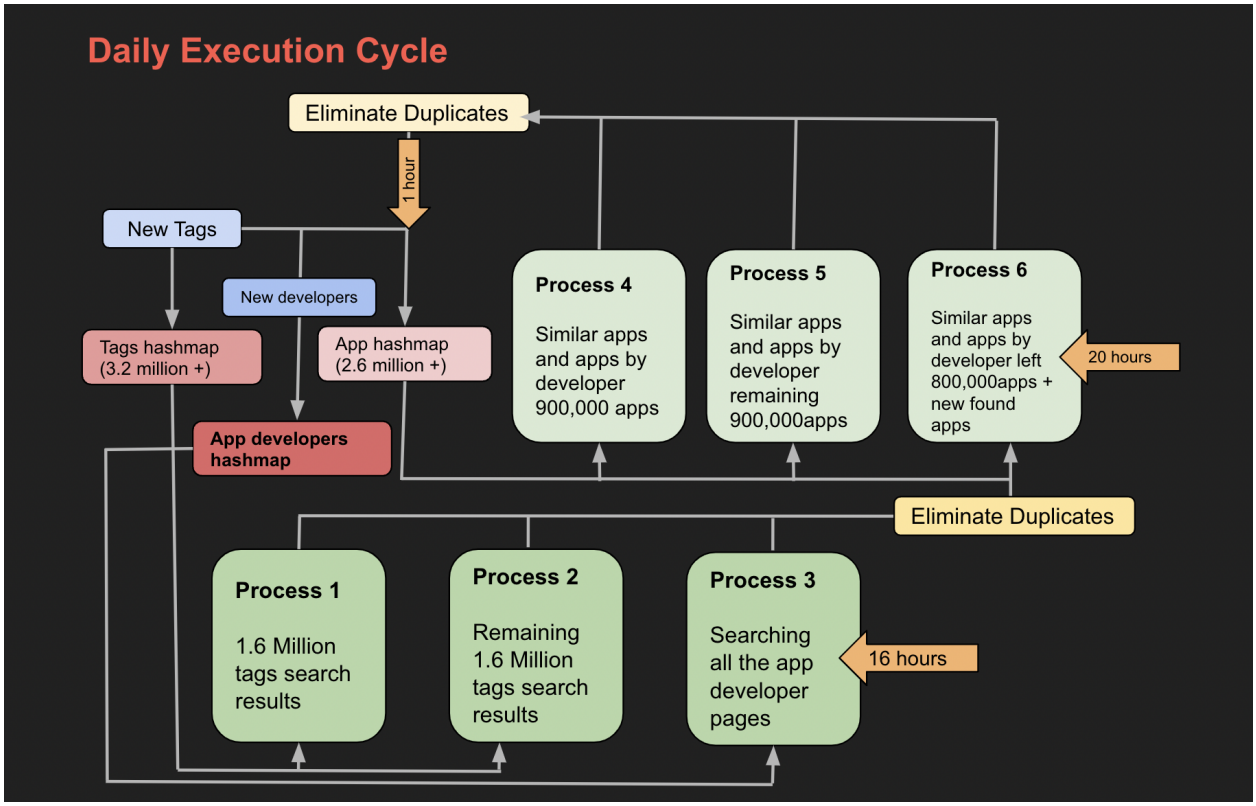
Recursive Web Crawling Algorithm for Efficient Information Extraction from App Stores and Websites

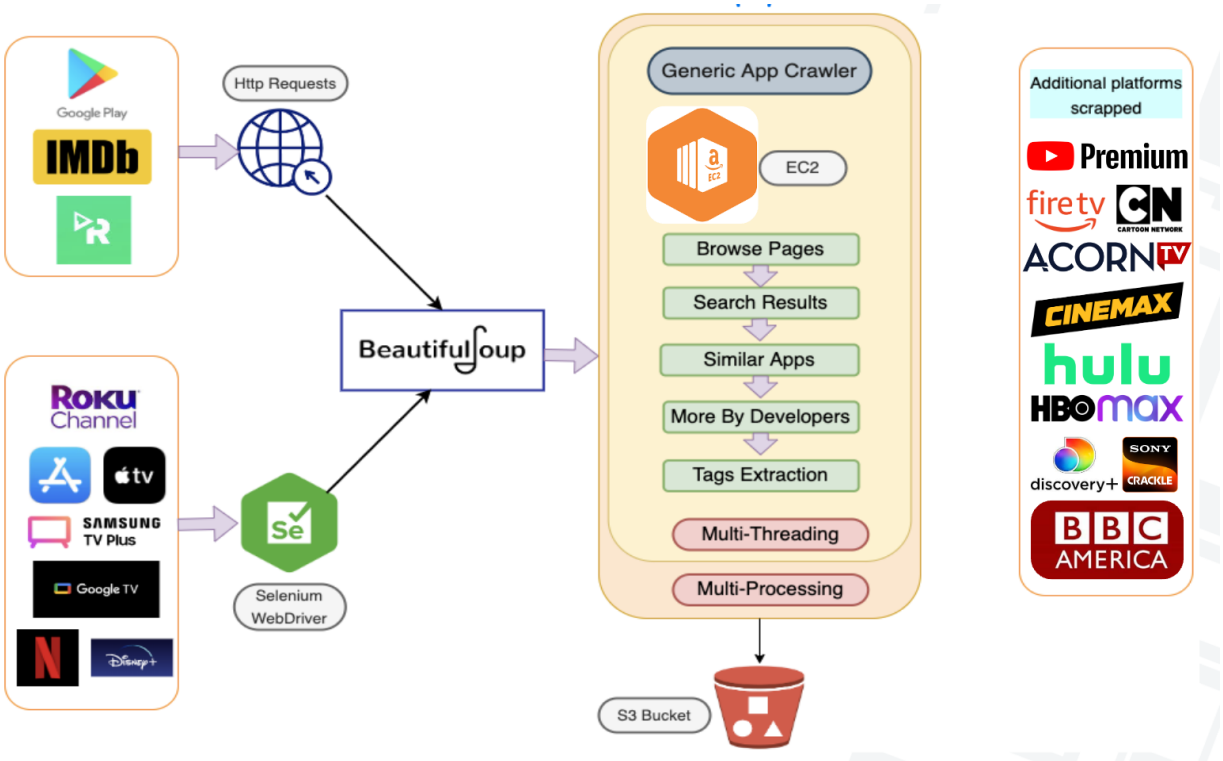
Abstract

This code is an implementation of a multithreaded web crawler for the Google Play Store or any App store. The crawler is used to search for apps and related tags on the Play Store. The main goal is to collect app data up to a specified limit. The code utilizes multithreading to improve efficiency and speed up the crawling process. It uses the threading module in Python to create and manage multiple threads that perform simultaneous tasks. The crawling process involves searching for apps and tags based on a list of words provided in a file. The code iterates through the list of words and creates multiple threads to search for apps and related tags using the Google Play Store search functionality. The search results are parsed using BeautifulSoup to extract relevant information. The visited apps and tags are stored in pickle files to keep track of the crawled data and avoid duplicate entries. The app URLs are added to an app queue for further processing, and the related tags are added to a search queue for subsequent searches. The code also includes error handling to save the crawled data periodically to prevent data loss in case of any issues or interruptions. Overall, this code demonstrates how the algorithm can be used to crawl the Google Play Store and various app stores efficiently and collect app data and related tags.

Algorithm




Daily Execution Cycle
Eliminate Duplicates
1 hour
New Tags
New developers
Tags hashmap (3.2 million +)
App hashmap (2.6 million +)
App developers hashmap
Process 4
Similar apps and apps by developer 900,000 apps
Process 5
Similar apps and apps by developer remaining 900,000apps
Process 6
Similar apps and apps by developer left 800,000apps + new found apps
20 hours
Eliminate Duplicates
Process 1
1.6 Million tags search results
Process 2
Remaining 1.6 Million tags search results
Process 3
Searching all the app developer pages
16 hours


Google Play
IMDb
Http Requests
Roku Channel
SAMSUNG TV Plus
Google TV
Selenium WebDriver
BeautifulSoup
Generic App Crawler
EC2
Browse Pages
Search Results
Similar Apps
More By Developers
Tags Extraction
Multi-Threading
Multi-Processing
S3 Bucket
Additional platforms scrapped
Premium
fire tv
ACORN TV
CINEMAX
hulu
HBO max
discovery+
SONY CRACKLE
BBC AMERICA

Software required to run the code

- Python 3.8 and above
- Beautiful soup library in python
- Google play scraper library in Python

Steps to install above software

1. Install python from official website of Python repository
2. pip install beautifulsoup4
3. pip install google-play-scraper

Steps to run the code

After installation of all the dependencies above code can be run using command python3 code.py

```python
# Python code to get search results

import json
import threading

import requests
from bs4 import BeautifulSoup
import re
from google_play_scraper import app
import pickle

Apps_Data_limit = 21000

country_codes = ['al', 'dz', 'ao', 'ag', 'ar', 'am', 'aw', 'au', 'at', 'az', 'bs', 'bh', 'bd', 'by', 'be', 'bz', 'bj',
                 'bo', 'ba', 'bw', 'br', 'bg', 'bf', 'kh', 'cm', 'ca', 'cv', 'cl', 'co', 'cr', 'ci', 'hr', 'cy', 'cz',
                 'dk', 'do', 'ec', 'eg', 'sv', 'ee', 'fj', 'fi', 'fr', 'ga', 'de', 'gh', 'gr', 'gt', 'gw', 'ht', 'hn',
                 'hk', 'hu', 'is', 'in', 'id', 'ie', 'il', 'it', 'jm', 'jp', 'jo', 'kz', 'ke', 'kw', 'kg', 'la', 'lv',
                 'lb', 'li', 'lt', 'lu', 'mk', 'my', 'ml', 'mt', 'mu', 'mx', 'md', 'ma', 'mz', 'na', 'np', 'nl', 'an',
                 'nz', 'ni', 'ne', 'ng', 'no', 'om', 'pk', 'pa', 'pg', 'py', 'pe', 'ph', 'pl', 'pt', 'qa', 'ro', 'ru',
                 'rw', 'sa', 'sn', 'rs', 'sg', 'sk', 'si', 'za', 'kr', 'es', 'lk', 'se', 'ch', 'tw', 'tj', 'tz', 'th',
                 'tg', 'tt', 'tn', 'tr', 'tm', 'ug', 'ua', 'ae', 'gb', 'us', 'uy', 'uz', 've', 'vn', 'ye', 'zm', 'zw']

visited_apps_file = open('mvisited_apps', 'rb')
visited_apps = pickle.load(visited_apps_file)
visited_tags_file = open('mvisited_tags', 'rb')
visited_search =pickle.load(visited_tags_file)
search_queue = []
app_queue_file = open('mapp_queue', 'rb')
app_queue = pickle.load(app_queue_file)
count = len(visited_apps)
#headers = {'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.95 Safari/537.36'}

search_url1 = 'https://play.google.com/store/search?q='
search_url2 = '&c=apps&gl=fi'
search_url2_base='&c=apps&gl='
search_url2_index = 15
start_index = 0


def search_apps(tag):
    if tag not in visited_search or True:
        visited_search[tag] = 1
        req = requests.get(search_url1 + tag + search_url2)
        soup = BeautifulSoup(req.content, 'html.parser')
        list_apps = soup.find_all('a', {'class': 'Si6A0c Gy4nib'})
        #print(str(len(list_apps)) + "  " + str(len(search_queue)))
        #print(str(count) + "  " + str(i))
        print(str(len(visited_apps))+"  "+str(len(app_queue))+"  "+tag)

        for app in list_apps:
            n_link = 'https://play.google.com' + app.attrs['href']
            if n_link not in visited_apps:
                #count += 1


                visited_apps[n_link] = 1
                app_queue.append(n_link)



f = open("words.txt", "r")


i = start_index
word_list = f.read().split('\n')
ri = 0
n = 100
word_list = word_list[:len(word_list)-start_index]
while len(word_list)>0:

    threads = []
    for j in range(n):
        if(len(word_list)>0):
            temp_thread = threading.Thread(target=search_apps, args=(word_list.pop(),))
            threads.append(temp_thread)
            temp_thread.start()




    for thread in threads:
        thread.join()

    i+=n
    ri+=n
    print(i)
    if ri%5000==0:
        try:

            visited_apps_file = open('mvisited_apps', 'wb')
            pickle.dump(visited_apps, visited_apps_file)
            visited_apps_file.close()

            visited_tags_file = open('mvisited_tags', 'wb')
            pickle.dump(visited_search, visited_tags_file)
            visited_tags_file.close()
```

```
            app_queue_file = open('mapp_queue', 'wb')
            pickle.dump(app_queue, app_queue_file)
            app_queue_file.close()
            print("saved")

        except:
            print("Something went wrong")




try:
                visited_apps_file = open('mvisited_apps', 'wb')
                pickle.dump(visited_apps, visited_apps_file)
                visited_apps_file.close()

                visited_tags_file = open('mvisited_tags', 'wb')
                pickle.dump(visited_search, visited_tags_file)
                visited_tags_file.close()

                app_queue_file = open('mapp_queue', 'wb')
                pickle.dump(app_queue, app_queue_file)
                app_queue_file.close()
                print("saved")

except:
                print("Something went wrong")




#Python code to search related apps, tags
import json
import threading

import requests
from bs4 import BeautifulSoup, SoupStrainer
import re
from google_play_scraper import app
import pickle
import lxml
#import cchardet

Apps_Data_limit = 21000

visited_apps_file = open('mvisited_apps', 'rb')
visited_apps = pickle.load(visited_apps_file)
visited_tags_file = open('mvisited_tags', 'rb')
visited_search = pickle.load(visited_tags_file)
search_queue = []
app_queue_file = open('mapp_queue', 'rb')
app_queue = pickle.load(app_queue_file)
count = len(visited_apps)
new_app_queue_file = open('new_app_queue', 'rb')
new_app_queue = pickle.load(new_app_queue_file)

search_url1 = 'https://play.google.com/store/search?q='
search_url2 = '&c=apps'

only_a_tags = SoupStrainer("a")
only_a_tags1 = SoupStrainer("a", class_=['WpHeLc VfPpkd-mRLv6', 'WpHeLc VfPpkd-mRLv6 VfPpkd-RLmnJb'])
only_a_tags2 = SoupStrainer("a", class_=['Si6A0c ZD8Cqc'])
session_object = requests.Session()

def search_apps(url):


        new_app_queue.append(url)
        while len(search_queue) > 0:
            tag = search_queue.pop()

            visited_search[tag] = 1
            req = requests.get(search_url1 + tag + search_url2)
            soup = BeautifulSoup(req.text, 'html.parser', parse_only=only_a_tags)
            list_apps = soup.find_all('a', {'class': 'Si6A0c Gy4nib'})
            print(len(list_apps))
            for app in list_apps:
                n_link = 'https://play.google.com' + app.attrs['href']
                if n_link not in visited_apps:
                    visited_apps[n_link] = 1
                    app_queue.append(n_link)

        print(url + "  " + str(len(app_queue)) + " " + str(len(new_app_queue)) + " "+str(len(visited_apps)))
        req = session_object.get(url)

        soup = BeautifulSoup(req.text, 'lxml', parse_only=only_a_tags1)
        #print(soup)
        list_apps = soup.find_all('a', {'class': 'WpHeLc VfPpkd-mRLv6'})

        list_search = soup.find_all('a', {'class': 'WpHeLc VfPpkd-mRLv6 VfPpkd-RLmnJb'})

        for i in list_search:
```

```
            tsearch = i.attrs['aria-label']
            if tsearch not in visited_search:
                search_queue.append(tsearch)
                visited_search[tsearch] = 1

        # print(list_apps)
        if (len(list_apps) == 0):
            return
        count_not = 0
        count_yes=0
        for apps in list_apps:
            if apps.attrs['aria-label'] != 'See more information on Data safety':

                suggested_link = 'https://play.google.com' + apps.attrs['href']
                suggested_req = session_object.get(suggested_link)
                suggested_soup = BeautifulSoup(suggested_req.text, 'lxml', parse_only=only_a_tags2)
                #print(suggested_soup)
                for link in suggested_soup.find_all('a', {'class': "Si6A0c ZD8Cqc"}):

                    n_link = 'https://play.google.com' + link.attrs['href']

                    # print(n_link)
                    if n_link not in visited_apps:
                        # writeToJSONFile(n_link)
                        visited_apps[n_link] = 1
                        app_queue.insert(0, n_link)
                        count_yes+=1
                    else:
                        count_not+=1
        #print(str(count_not)+"  "+str(count_yes))

f = open("words.txt", "r")

i = 0
word_list = f.read().split('\n')
ri = 0
n =100
while len(app_queue)>0:

    session_object = requests.Session()
    threads = []
    for j in range(n):
        if(len(app_queue)>0):
            temp_thread = threading.Thread(target=search_apps, args=(app_queue.pop(),))
            threads.append(temp_thread)
            temp_thread.start()


    for thread in threads:
        thread.join()

    """
    tag1 = word_list.pop()
    tag2 = word_list.pop()
    tag3 = word_list.pop()
    tag4 = word_list.pop()
    tag5 = word_list.pop()
    tag6 = word_list.pop()
    tag7 = word_list.pop()
    tag8 = word_list.pop()
    tag9 = word_list.pop()
    tag10 = word_list.pop()



    t1 = threading.Thread(target=search_apps, args=(tag1,))
    t2 = threading.Thread(target=search_apps, args=(tag2,))
    t3 = threading.Thread(target=search_apps, args=(tag3,))
    t4 = threading.Thread(target=search_apps, args=(tag4,))
    t5 = threading.Thread(target=search_apps, args=(tag5,))
    t6 = threading.Thread(target=search_apps, args=(tag6,))
    t7 = threading.Thread(target=search_apps, args=(tag7,))
    t8 = threading.Thread(target=search_apps, args=(tag8,))
    t9 = threading.Thread(target=search_apps, args=(tag9,))
    t10 = threading.Thread(target=search_apps, args=(tag10,))

    t1.start()
    t2.start()
    t3.start()
    t4.start()
    t5.start()
    t6.start()
    t7.start()
    t8.start()
    t9.start()
    t10.start()

    t1.join()
    t2.join()
    t3.join()
    t4.join()
    t5.join()
    t6.join()
    t7.join()
    t8.join()
    t9.join()
    t10.join()
```

```
    """

    i+=n
    ri+=n
    print(i)
    if ri%1000==0:
        try:
            visited_apps_file = open('mvisited_apps', 'wb')
            pickle.dump(visited_apps, visited_apps_file)
            visited_apps_file.close()

            visited_tags_file = open('mvisited_tags', 'wb')
            pickle.dump(visited_search, visited_tags_file)
            visited_tags_file.close()

            app_queue_file = open('mapp_queue', 'wb')
            pickle.dump(app_queue, app_queue_file)
            app_queue_file.close()

            new_app_queue_file = open('new_app_queue', 'wb')
            pickle.dump(new_app_queue, new_app_queue_file)
            new_app_queue_file.close()

            print("saved")

        except:
            print("Something went wrong")




try:
            visited_apps_file = open('mvisited_apps', 'wb')
            pickle.dump(visited_apps, visited_apps_file)
            visited_apps_file.close()

            visited_tags_file = open('mvisited_tags', 'wb')
            pickle.dump(visited_search, visited_tags_file)
            visited_tags_file.close()

            app_queue_file = open('mapp_queue', 'wb')
            pickle.dump(app_queue, app_queue_file)
            app_queue_file.close()

            new_app_queue_file = open('new_app_queue', 'wb')
            pickle.dump(new_app_queue, new_app_queue_file)
            new_app_queue_file.close()

            print("saved")

except:
            print("Something went wrong")
```