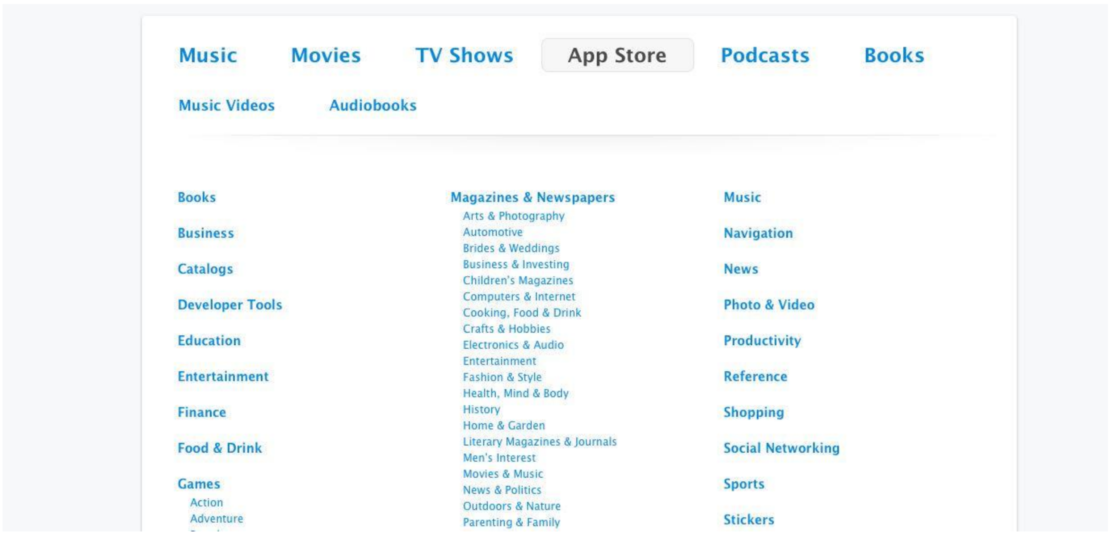
CS6200-IR-Final Project Report

Author: Yuchen Xie

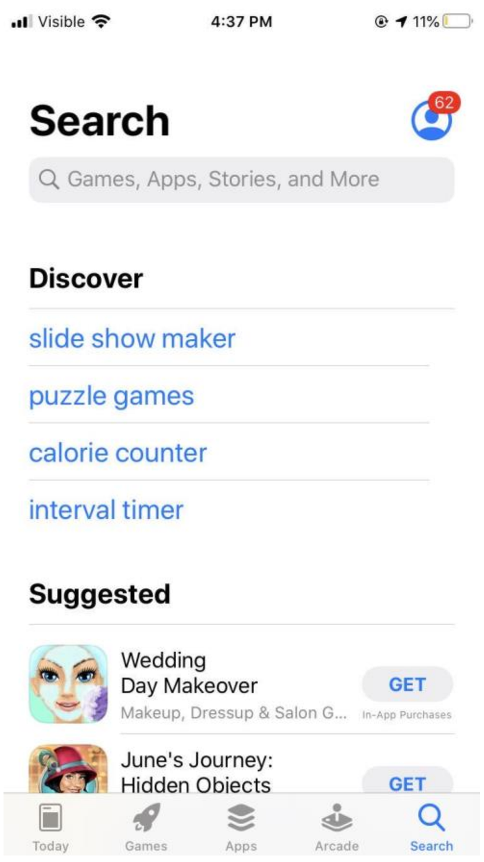
Background and Problems

As we can see from graph1 which is the official site of apps.apple.com, there is no search user interface for the applications in itunes store. The official site is where we can see all the applications' complete data, each one of them is displayed on apple's official site, but we can not search on these apps. In other words, the search interface on itunes and mobile app store does not have transparent progress to users.

Complete data is available on the official site, and the project’ s goals is to build a search engine on these data.



1 *Official Site of apple*


Visible 4:37 PM 11%

Search

Games, Apps, Stories, and More

Discover

slide show maker

puzzle games

calorie counter

interval timer

Suggested

Wedding Day Makeover — Makeup, Dressup & Salon G... — GET — In-App Purchases

June's Journey: Hidden Objects — GET

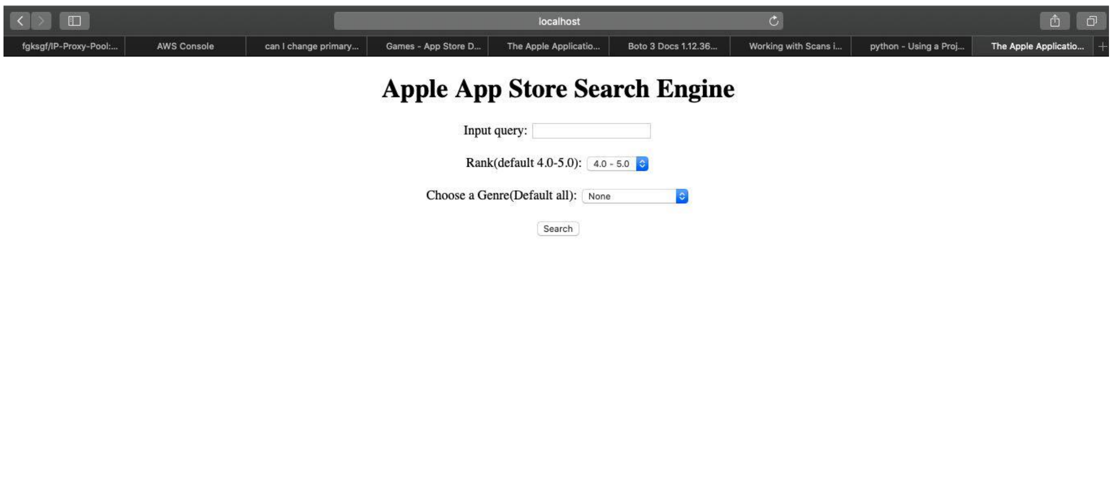
2 *Itunes search page on mobile phone*

On the mobile page, the users only have a input search interface, no other settings can be made, like genre setting or ranking settings. When we are trying to search some game and other tools in photos & videos popping out would make people frustrating. In the meantime, we do not know how apple ranks the results that it's returning to us, some applications got returned top of the list with no reason, maybe developers can trade higher recommendation ranks with apple, there are always some promotion on the page, we don't know, but the ranking is clearly vague.

When we are actually using the search function apple has offered us, sometimes we forget about the name of the application, we would try to type in as much characters as we remember, but the search can only be based on names of the apps and other obvious aspects like zombie games or dungeon games. We want to search all the applications by all information, we want to search on developers, description and genres. Thus, the goal of the project is to come up with a more comprehensive ad-hoc search engine, plus some of the vertical search function.

For this search engine, I want it to has as much data as possible, with no other influence, the engine would rank the return results only on its similarity and relevance to the query and previous settings.

Overview of the search engine



Apple App Store Search Engine

Input query:

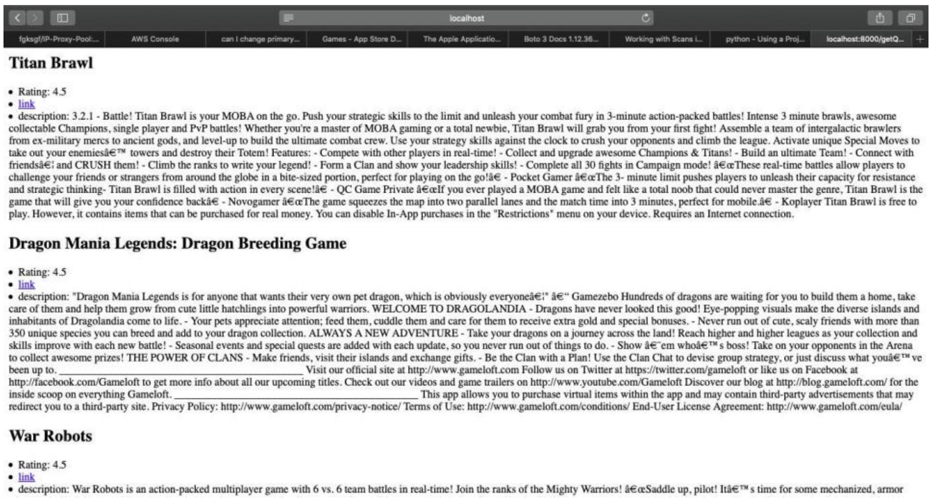
Rank(default 4.0-5.0): 4.0 - 5.0

Choose a Genre(Default all): None

3 *Search Engine page*

The page of my search engine is showed above, it has one query input, which would allow users to type in whatever they want.

In addtion, there are two bar tools that would allow users to select which genre and which rank interval the users want to set before search. If the genres and rankings has been put, the search tends to cost less time, since the data set is smaller, and there is less things to compute.



Titan Brawl

- Rating: 4.5
- link
- description: 3.2.1 - Battle! Titan Brawl is your MOBA on the go. Push your strategic skills to the limit and unleash your combat fury in 3-minute action-packed battles! Intense 3 minute brawls, awesome collectable Champions, single player and PvP battles! Whether you're a master of MOBA gaming or a total newbie, Titan Brawl will grab you from your first fight! Assemble a team of intergalactic brawlers from ex-military mercs to ancient gods, and level-up to build the ultimate combat crew. Use your strategy skills against the clock to crush your opponents and climb the league. Activate unique Special Moves to take out your enemiesâ€™ towers and destroy their Totem! Features: - Compete with other players in real-time! - Collect and upgrade awesome Champions & Titans! - Build an ultimate Team! - Connect with friendsâ€¦ and CRUSH them! - Climb the ranks to write your legend! - Form a Clan and show your leadership skills! - Complete all 30 fights in Campaign mode! â€œThese real-time battles allow players to challenge your friends or strangers from around the globe in a bite-sized portion, perfect for playing on the go!â€ - Pocket Gamer â€œThe 3- minute limit pushes players to unleash their capacity for resistance and strategic thinking- Titan Brawl is filled with action in every scene!â€ - QC Game Private â€œIf you ever played a MOBA game and felt like a total noob that could never master the genre, Titan Brawl is the game that will give you your confidence backâ€ - Novogamer â€œThe game squeezes the map into two parallel lanes and the match time into 3 minutes, perfect for mobile.â€ - Koplayer Titan Brawl is free to play. However, it contains items that can be purchased for real money. You can disable In-App purchases in the "Restrictions" menu on your device. Requires an Internet connection.

Dragon Mania Legends: Dragon Breeding Game

- Rating: 4.5
- link
- description: "Dragon Mania Legends is for anyone that wants their very own pet dragon, which is obviously everyoneâ€¦" â€“ Gamezebo Hundreds of dragons are waiting for you to build them a home, take care of them and help them grow from cute little hatchlings into powerful warriors. WELCOME TO DRAGOLANDIA - Dragons have never looked this good! Eye-popping visuals make the diverse islands and inhabitants of Dragolandia come to life. - Your pets appreciate attention; feed them, cuddle them and care for them to receive extra gold and special bonuses. - Never run out of cute, scaly friends with more than 350 unique species you can breed and add to your dragon collection. ALWAYS A NEW ADVENTURE - Take your dragons on a journey across the land! Reach higher and higher leagues as your collection and skills improve with each new battle! - Seasonal events and special quests are added with each update, so you never run out of things to do. - Show â€˜em whoâ€™s boss! Take on your opponents in the Arena to collect awesome prizes! THE POWER OF CLANS - Make friends, visit their islands and exchange gifts. - Be the Clan with a Plan! Use the Clan Chat to devise group strategy, or just discuss what youâ€™ve been up to. ______________________________ Visit our official site at http://www.gameloft.com Follow us on Twitter at https://twitter.com/gameloft or like us on Facebook at http://facebook.com/Gameloft to get more info about all our upcoming titles. Check out our videos and game trailers on http://www.youtube.com/Gameloft Discover our blog at http://blog.gameloft.com/ for the inside scoop on everything Gameloft. ______________________________ This app allows you to purchase virtual items within the app and may contain third-party advertisements that may redirect you to a third-party site. Privacy Policy: http://www.gameloft.com/privacy-notice/ Terms of Use: http://www.gameloft.com/conditions/ End-User License Agreement: http://www.gameloft.com/eula/

War Robots

- Rating: 4.5
- link
- description: War Robots is an action-packed multiplayer game with 6 vs. 6 team battles in real-time! Join the ranks of the Mighty Warriors! â€œSaddle up, pilot! Itâ€™s time for some mechanized, armor

4 *Search result*

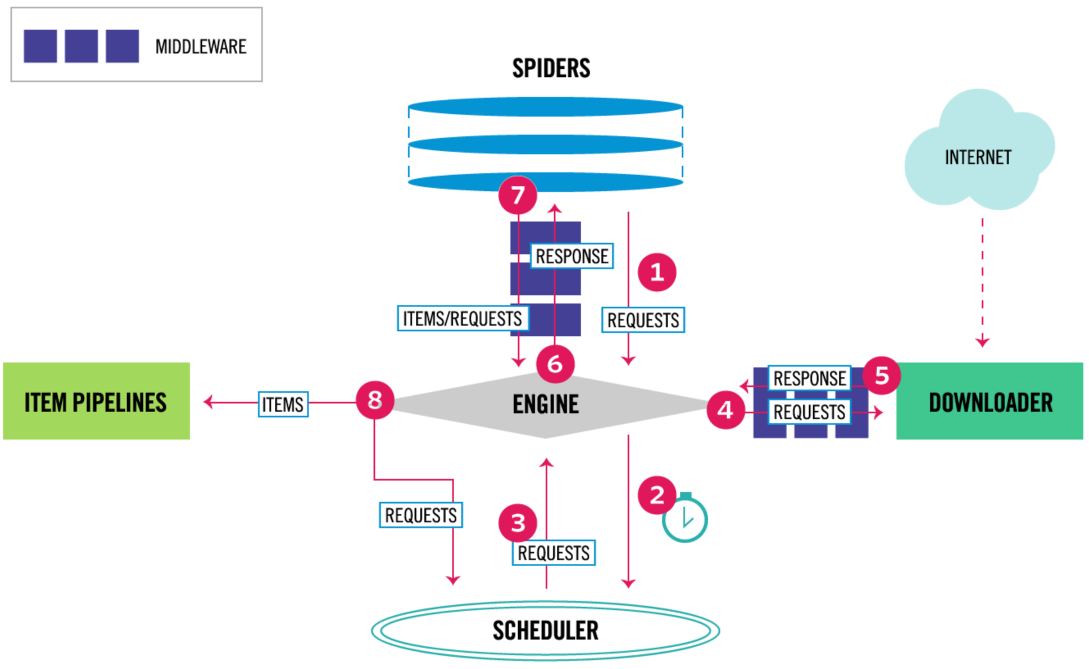
The search results of the engine have simple structure, it would return all the information that users want, actually what I chose to show on the result page is limited, in storage I have all kinds of data like how many people rated this app, what age should be appropriate for the application, but the users only need some information for them to see. In such case, I chose to show Title, ranking and description. The link here can lead users to the application's actual apple store page.

How it works

The whole project can be split into 4 parts to demonstrate: **Crawler**, **Data storage and synchronize**, **Ranking** and **web server**. Let me explain the project by detail.

1. Crawler

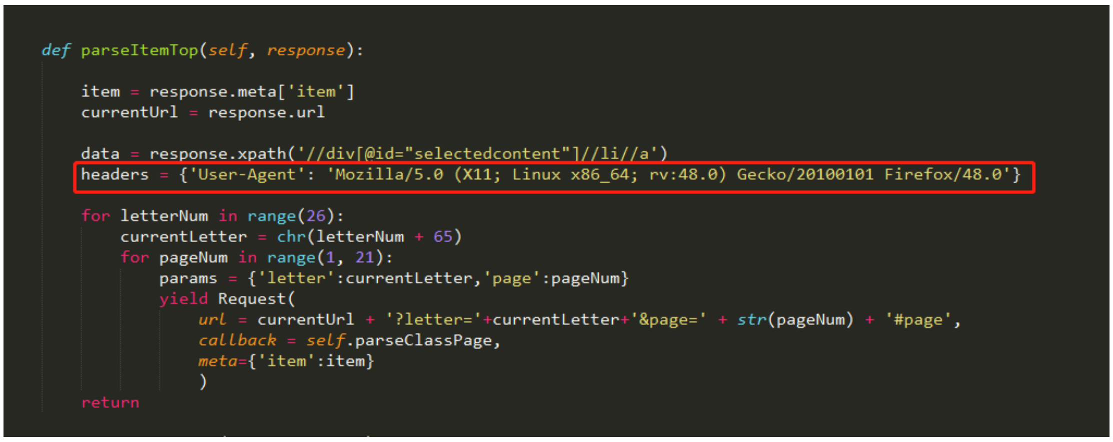
For crawler I use scrapy as crawler framework, which has some advantages like default concurrency, download delays and rotating IP proxy pool. With customized middlewares and pipelines, users can optimize their crawlers' performance and alter them according to actual usage.



5 *Scrapy Architecture*

Due to the IP banning policy toward robots of Apple, I need to change the download delay to be 10 seconds per page, if I download the pages faster, like 300 pages per minutes, my IP would be banned for 3 to 4 hours, which is unacceptable. Apparently there' s a limit number of pages on Apple that each IP can open in a fixed period.

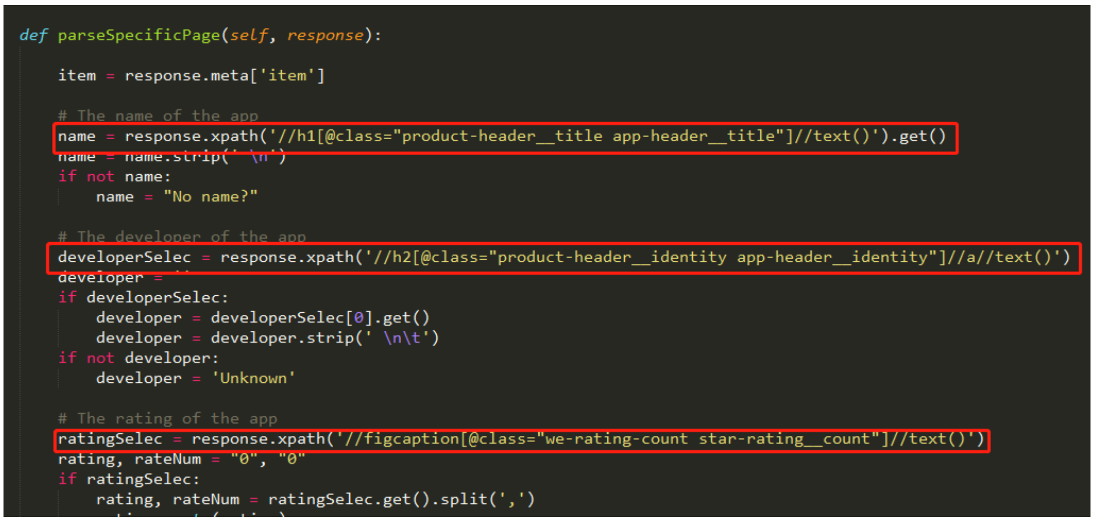
For the crawler' s each request, I would disguise the crawler with customized user-agent:



```python
def parseItemTop(self, response):

    item = response.meta['item']
    currentUrl = response.url

    data = response.xpath('//div[@id="selectedcontent"]//li//a')
    headers = {'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64; rv:48.0) Gecko/20100101 Firefox/48.0'}

    for letterNum in range(26):
        currentLetter = chr(letterNum + 65)
        for pageNum in range(1, 21):
            params = {'letter':currentLetter,'page':pageNum}
            yield Request(
                url = currentUrl + '?letter='+currentLetter+'&page=' + str(pageNum) + '#page',
                callback = self.parseClassPage,
                meta={'item':item}
                )
    return
```

6 *Crawler-User agent*

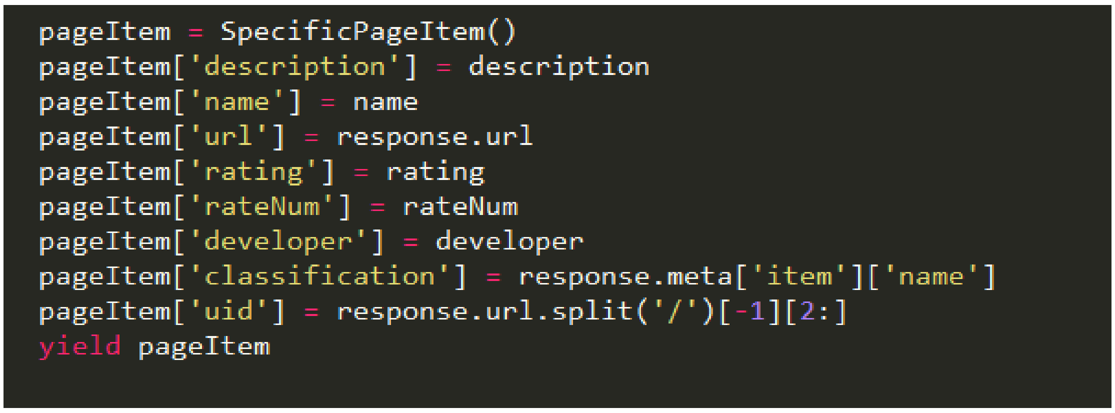
After downloading the page, I used scrapy built-in function xpath to parse the response text by selecting elements with specific id or class.



```python
def parseSpecificPage(self, response):

    item = response.meta['item']

    # The name of the app
    name = response.xpath('//h1[@class="product-header__title app-header__title"]//text()').get()
    name = name.strip(' \n')
    if not name:
        name = "No name?"

    # The developer of the app
    developerSelec = response.xpath('//h2[@class="product-header__identity app-header__identity"]//a//text()')
    developer = ''
    if developerSelec:
        developer = developerSelec[0].get()
        developer = developer.strip(' \n\t')
    if not developer:
        developer = 'Unknown'

    # The rating of the app
    ratingSelec = response.xpath('//figcaption[@class="we-rating-count star-rating__count"]//text()')
    rating, rateNum = "0", "0"
    if ratingSelec:
        rating, rateNum = ratingSelec.get().split(',')
```

7 *Xpath example*

When the parsing is done, the crawler would pass an *item* to pipeline, then it would be stored into database.



```python
pageItem = SpecificPageItem()
pageItem['description'] = description
pageItem['name'] = name
pageItem['url'] = response.url
pageItem['rating'] = rating
pageItem['rateNum'] = rateNum
pageItem['developer'] = developer
pageItem['classification'] = response.meta['item']['name']
pageItem['uid'] = response.url.split('/')[-1][2:]
yield pageItem
```

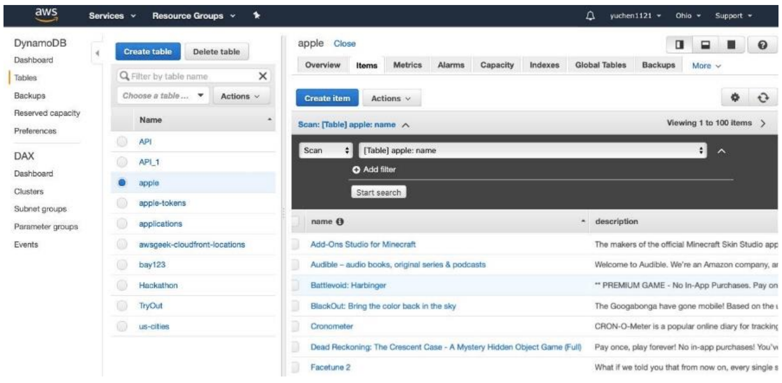
8 *Scrapy Item*

We can see from graph 7, there are 8 elements in the item, which would be the features/keys of the data that we store into the database.

There are several things need to be set previously in setting of scrapy, like DOWNLOAD_DELAY, ROBOTSTXT_OBEY and concurrency etc., if pipelines or items are being implemented, we need to set them active in setting as well.

2. Data Storage and Synchronization

When the crawler has finished downloading the web page and parsing it, it will be passed to pipeline of scrapy, in the pipeline, I would do the tokenization, and then store all I got into database: AWS Dynamodb.



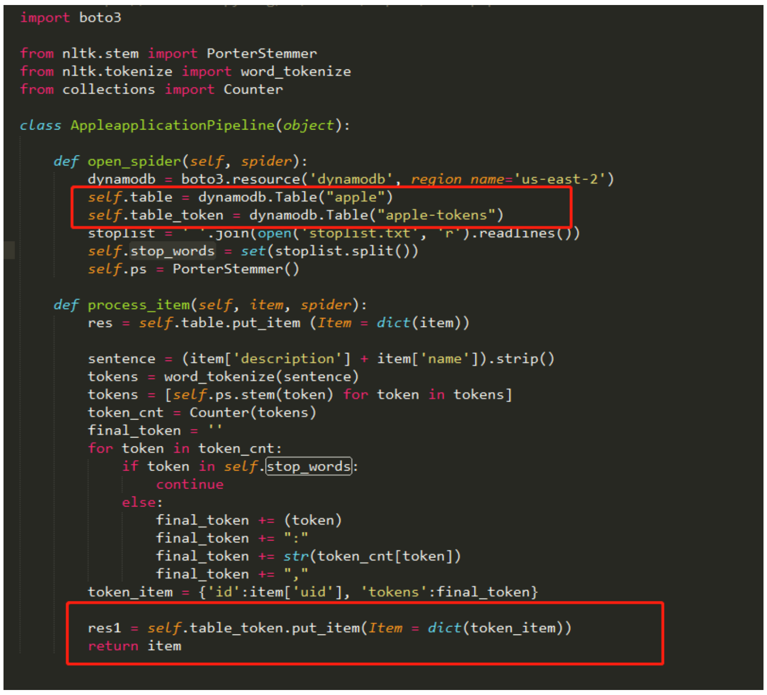
Name
API
API_1
apple
apple-tokens
applications
awsgeek-cloudfront-locations
bay123
Hackathon
TryOut
us-cities

name	description
Add-Ons Studio for Minecraft	The makers of the official Minecraft Skin Studio app
Audible – audio books, original series & podcasts	Welcome to Audible. We're an Amazon company, ar
Battlevoid: Harbinger	** PREMIUM GAME - No In-App Purchases. Pay on
BlackOut: Bring the color back in the sky	The Googabonga have gone mobile! Based on the
Cronometer	CRON-O-Meter is a popular online diary for tracking
Dead Reckoning: The Crescent Case - A Mystery Hidden Object Game (Full)	Pay once, play forever! No in-app purchases! You'v
Facetune 2	What if we told you that from now on, every single s

9 *AWS Dynamodb*

The biggest reason I am using Dynamodb is, cheap. It has some free tier and credit for students. Other consideration like it is a relational database which can prevent duplicate by setting the primary key which would overwrite the records with the same key attribute; It has user interreact page which can offer us better way to monitor the tables; Dynamodb provides two different ways to retrieve the data, scan and query, that suits different scenarios; Convenient API in Python, etc.

Before putting the whole data into the database, I will preprocess the data, which is stemming and tokenizing, since I am using cosine similarity to rank the return result, it will improve the efficiency if I parse the tokens at the time when crawler is waiting to download another page, this action would save some time. In stemming and tokenizing I am using nltk to perform such actions.



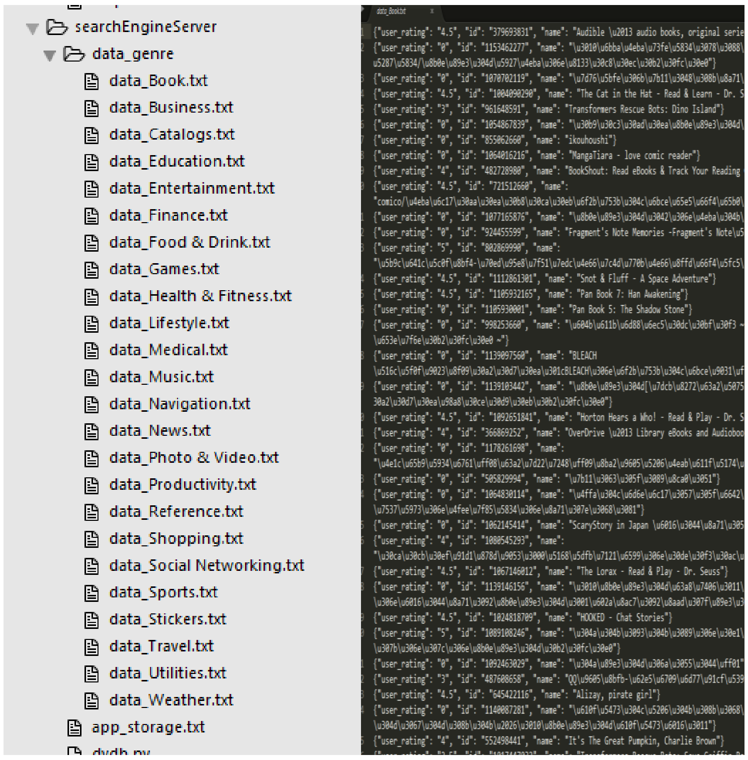
```
import boto3

from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from collections import Counter

class AppleapplicationPipeline(object):

    def open_spider(self, spider):
        dynamodb = boto3.resource('dynamodb', region_name='us-east-2')
        self.table = dynamodb.Table("apple")
        self.table_token = dynamodb.Table("apple-tokens")
        stoplist = ' '.join(open('stoplist.txt', 'r').readlines())
        self.stop_words = set(stoplist.split())
        self.ps = PorterStemmer()

    def process_item(self, item, spider):
        res = self.table.put_item (Item = dict(item))

        sentence = (item['description'] + item['name']).strip()
        tokens = word_tokenize(sentence)
        tokens = [self.ps.stem(token) for token in tokens]
        token_cnt = Counter(tokens)
        final_token = ''
        for token in token_cnt:
            if token in self.stop_words:
                continue
            else:
                final_token += (token)
                final_token += ":"
                final_token += str(token_cnt[token])
                final_token += ","
        token_item = {'id':item['uid'], 'tokens':final_token}

        res1 = self.table_token.put_item(Item = dict(token_item))
        return item
```

10 *Dynamodb put item*

Due to the big dataload on the official site of Apple, the crawler is consistent running, the data is continually being put into the database. When the search engine is running, if the engine retrieves the data online from the database, it would not only be incredibly slow, but also it can result in dirty read and conflicts.

Thus, I would daily synchronize the data into local storage files in order to accelerate the search and compute. (You can see searchEngineServer / refresh_app.py). All data are store in searchEngineServer/data_genre/*.

The data that I would query actually looks like this:



```
searchEngineServer
  data_genre
    data_Book.txt
    data_Business.txt
    data_Catalogs.txt
    data_Education.txt
    data_Entertainment.txt
    data_Finance.txt
    data_Food & Drink.txt
    data_Games.txt
    data_Health & Fitness.txt
    data_Lifestyle.txt
    data_Medical.txt
    data_Music.txt
    data_Navigation.txt
    data_News.txt
    data_Photo & Video.txt
    data_Productivity.txt
    data_Reference.txt
    data_Shopping.txt
    data_Social Networking.txt
    data_Sports.txt
    data_Stickers.txt
    data_Travel.txt
    data_Utilities.txt
    data_Weather.txt
  app_storage.txt
```

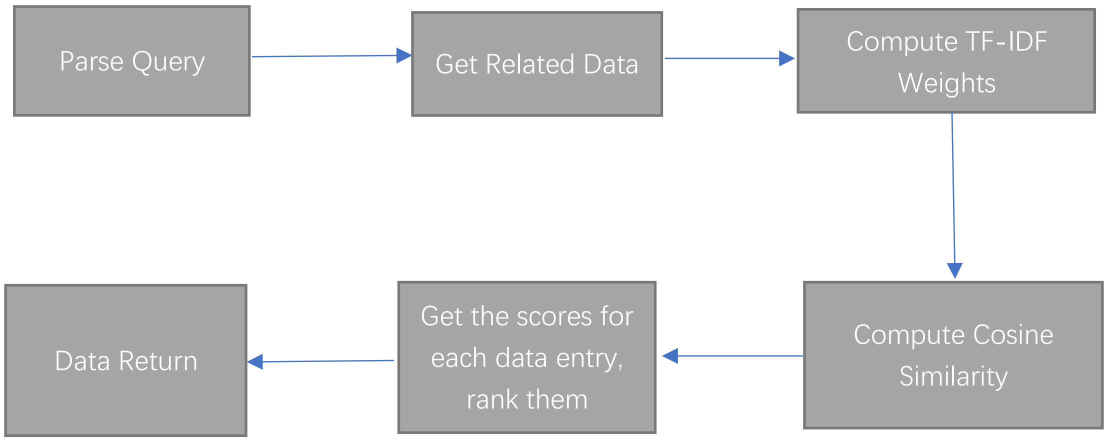
```
{"user_rating": "4.5", "id": "379693831", "name": "Audible \u2013 audio books, original serie
{"user_rating": "0", "id": "1153462277", "name": "\u3010\u6bba\u4eba\u73fe\u5834\u3078\u3088\
u5287\u5834/\u8b0e\u89e3\u304d\u5927\u4eba\u306e\u8133\u30c8\u30ec\u30b2\u30fc\u30e0"}
{"user_rating": "0", "id": "1070702119", "name": "\u7d76\u5bfe\u306b\u7b11\u3048\u308b\u8a71\
{"user_rating": "4.5", "id": "1004090290", "name": "The Cat in the Hat - Read & Learn - Dr. S
{"user_rating": "3", "id": "961648591", "name": "Transformers Rescue Bots: Dino Island"}
{"user_rating": "0", "id": "1054867839", "name": "\u30b9\u30c3\u30ad\u30ea\u8b0e\u89e3\u304d\
{"user_rating": "0", "id": "855062660", "name": "ikouhoushi"}
{"user_rating": "0", "id": "1064016216", "name": "MangaTiara - love comic reader"}
{"user_rating": "4", "id": "482728980", "name": "BookShout: Read eBooks & Track Your Reading
{"user_rating": "4.5", "id": "721512660", "name":
"comico/\u4eba\u6c17\u30aa\u30ea\u30b8\u30ca\u30eb\u6f2b\u753b\u304c\u6bce\u65e5\u66f4\u65b0\
{"user_rating": "0", "id": "1077165876", "name": "\u8b0e\u89e3\u304d\u3042\u306e\u4eba\u304b\
{"user_rating": "0", "id": "924455599", "name": "Fragment's Note Memories -Fragment's Note\u5
{"user_rating": "5", "id": "802869990", "name":
"\u5b9c\u641c\u5c0f\u8bf4-\u70ed\u95e8\u7f51\u7edc\u4e66\u7c4d\u770b\u4e66\u8ffd\u66f4\u5fc5\
{"user_rating": "4.5", "id": "1112861301", "name": "Snot & Fluff - A Space Adventure"}
{"user_rating": "4.5", "id": "1105932165", "name": "Pan Book 7: Han Awakening"}
{"user_rating": "0", "id": "1105930001", "name": "Pan Book 5: The Shadow Stone"}
{"user_rating": "0", "id": "998253660", "name": "\u604b\u611b\u6d88\u6ec5\u30dc\u30bf\u30f3 ~
\u653e\u7f6e\u30b2\u30fc\u30e0 ~"}
{"user_rating": "0", "id": "1139097560", "name": "BLEACH
\u516c\u5f0f\u9023\u8f09\u30a2\u30d7\u30ea\u301cBLEACH\u306e\u6f2b\u753b\u304c\u6bce\u9031\uf
{"user_rating": "0", "id": "1139103442", "name": "\u8b0e\u89e3\u304d[\u7dcb\u8272\u63a2\u5075
30a2\u30d7\u30ea\u98a8\u30ce\u30d9\u30eb\u30b2\u30fc\u30e0"}
{"user_rating": "4.5", "id": "1092651841", "name": "Horton Hears a Who! - Read & Play - Dr. S
{"user_rating": "4", "id": "366869252", "name": "OverDrive \u2013 Library eBooks and Audioboo
{"user_rating": "0", "id": "1178261698", "name":
"\u4e1c\u65b9\u5934\u6761\uff08\u63a2\u7d22\u7248\uff09\u8ba2\u9605\u5206\u4eab\u611f\u5174\u
{"user_rating": "0", "id": "505829994", "name": "\u7b11\u3063\u305f\u3089\u8ca0\u3051"}
{"user_rating": "0", "id": "1064830114", "name": "\u4ffa\u304c\u6d6e\u6c17\u3057\u305f\u6642\
\u7537\u5973\u306e\u4fee\u7f85\u5834\u306e\u8a71\u307e\u3068\u3081"}
{"user_rating": "0", "id": "1062145414", "name": "ScaryStory in Japan \u6016\u3044\u8a71\u305
{"user_rating": "4", "id": "1080545293", "name":
"\u30ca\u30cb\u30ef\u91d1\u878d\u9053\u3000\u5168\u5dfb\u7121\u6599\u306e\u30de\u30f3\u30ac\u
{"user_rating": "4.5", "id": "1067146012", "name": "The Lorax - Read & Play - Dr. Seuss"}
{"user_rating": "0", "id": "1139146156", "name": "\u3010\u8b0e\u89e3\u304d\u63a8\u7406\u3011\
\u306e\u6016\u3044\u8a71\u3092\u8b0e\u89e3\u304d\u3001\u602a\u8ac7\u3092\u8aad\u307f\u89e3\u3
{"user_rating": "4.5", "id": "1024818709", "name": "HOOKED - Chat Stories"}
{"user_rating": "5", "id": "1089108246", "name": "\u304a\u304b\u3093\u304b\u3089\u306e\u30e1\
\u307b\u306e\u307c\u306e\u8b0e\u89e3\u304d\u30b2\u30fc\u30e0"}
{"user_rating": "0", "id": "1092463029", "name": "\u304a\u89e3\u304d\u306a\u3055\u3044\uff01"
{"user_rating": "3", "id": "487608658", "name": "QQ\u9605\u8bfb-\u62e5\u6709\u6d77\u91cf\u539
{"user_rating": "4.5", "id": "645422116", "name": "Alizay, pirate girl"}
{"user_rating": "0", "id": "1140087281", "name": "\u610f\u5473\u304c\u5206\u304b\u308b\u3068\
\u304d\u3067\u304d\u308b\u304b\u2026\u3010\u8b0e\u89e3\u304d\u610f\u5473\u6016\u3011"}
{"user_rating": "4", "id": "552498441", "name": "It's The Great Pumpkin, Charlie Brown"}
```

11 *Actuall Data*

The Data stored in local are in simpler form, only contains import attributes which would help us to identify the data, like name and id, and other attributes which would help us to rank the data like the ranking on app store. This option would make data stored locally with smaller size and computed more conveniently.

3. Ranking

For the results ranking, I am using the simplest measures to rank them, TF-IDF and cosine similarity.


Parse Query
Get Related Data
Compute TF-IDF Weights
Compute Cosine Similarity
Get the scores for each data entry, rank them
Data Return

Process are as above. Actual code about data computation can be viewed in searchEngineServer/parser.py. Code about data retrieve from local storage can be viewed in searchEngineServer/dydb.py. Stemming and Tokenizing are based on nltk, a python package, tf-idf and cosine care computed by raw code without other packages.

The query would be consisting of a text query, a rank interval (If set), a targeted genre (If set). According to the specific requirement of the query, the ranking function would first request the data from local storage, and then start to compute the scores of each data entry. After the computation is done, the results would be ranked by their relevance firstly, and then sorted by their actual store ranking secondly, which would give users better experience.

The process is relatively easy but functional. The tokens of each data entry are parsed by one application’ s name, description and developers, which can ensure better search upon attributes other than names.

The returned data are in simple form, only has name, id and rank, so that the computation and local storage pressure would be lighter. After we ranked the results, we have to retrieve the data entry in complete form which has all attributes from database, then we can return them to the server.

4. Web Server

The server is based on Python’ s baseHTTPServer, which is a simple http server packet supported by python2, no longer a separate package in Python3.

There’ s no much to talk about the server. The query is passed to backend with POST method, after all the process being done, the results would be returned and showed directly on the web page.

Conclusion

This project let me build the search engine from data crawling, although I thought about using other search engine like Elastic Search and Solr, I decided not to apply other techniques since we have already done the tokenizing and ranking assignments, I think those can really help the final project.

I have learned great knowledge about IR and search engine from this project and this course, I appreciate the help I got from TA Michelle and Gustavo, and Professor Alonso and Professor Ricardo has offered great deal of help, I deeply thank you for educating me such wonderful course.

Github link of the project:

https://github.com/Yuchen112211/CS6200-IR-AppleSearchEngine