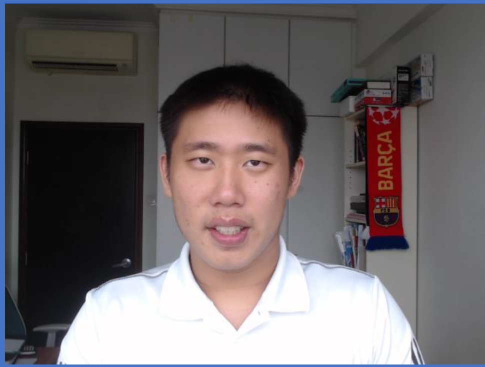


A Tier-Structured, Modular Scraper with a User Verification System to Profile Data Brokers

CMSC 25900 Final Project

Bruce Wen

Outline

Project Goals

Architecture and Description of Code Base

Scraping, MongoDB, Vue.JS front-end, hosting on Heroku

Code Demo

Privacy Considerations

Findings, Future Work

Goal

To build a tier-structured, modular scraper that has the following features:

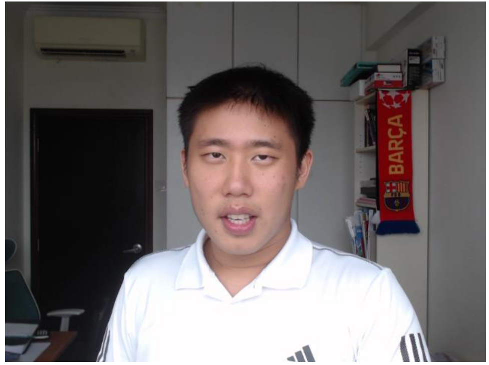
Easily add on new data-broker websites to scrape

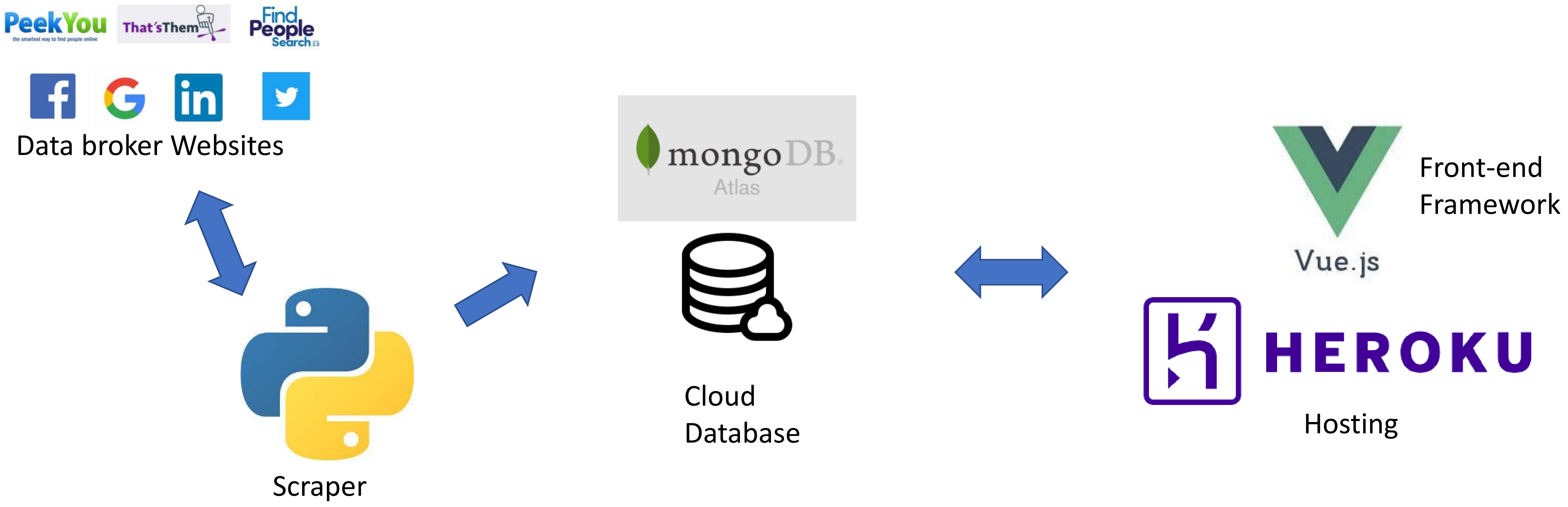
Easily add on subjects to scrape their data from data-brokers

Front-End dynamic website linked to server that allows User verification of authenticity of data

Conduct Preliminary Analysis of Data Broker Websites using my code base and a small surveyed sample of ~20 participants

Overall Architecture of Code Base





Data broker Websites

Scraper

Cloud Database

Front-end Framework

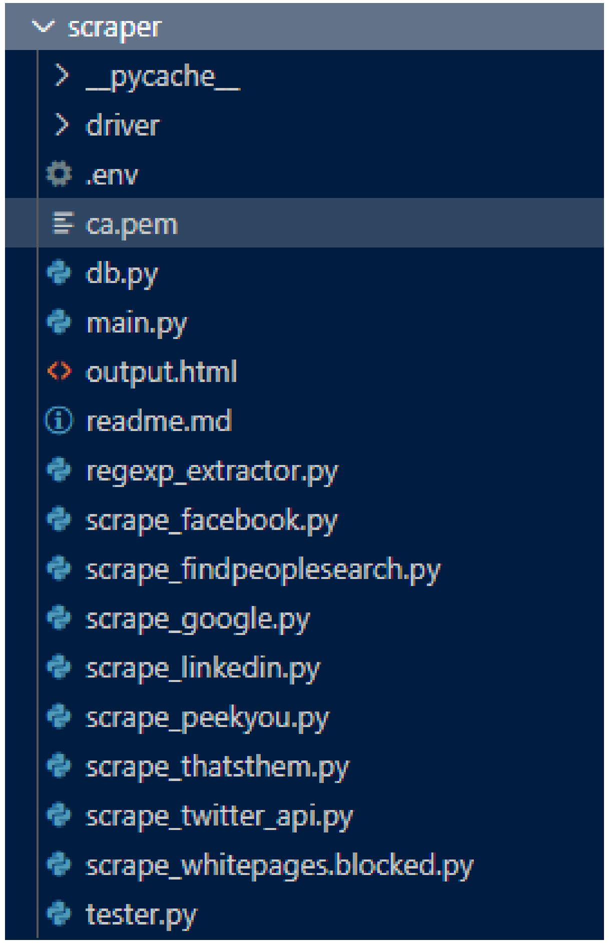
Hosting

Scrape Data

Store Data

User Verification

Scraping



```
⌄ scraper
  > __pycache__
  > driver
  .env
  ca.pem
  db.py
  main.py
  output.html
  readme.md
  regexp_extractor.py
  scrape_facebook.py
  scrape_findpeoplesearch.py
  scrape_google.py
  scrape_linkedin.py
  scrape_peekyou.py
  scrape_thatsthem.py
  scrape_twitter_api.py
  scrape_whitepages.blocked.py
  tester.py
```

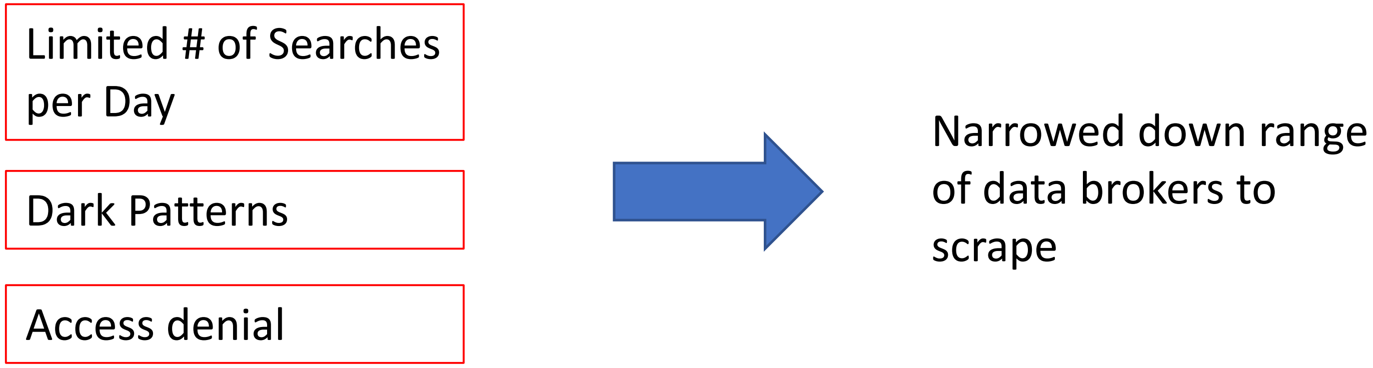
A total of 7 sites were scraped

PeekYou
the smartest way to find people online

That'sThem

Find People Search version 2.0

Difficulties Overcome and Techniques Employed



Limited # of Searches per Day

Dark Patterns

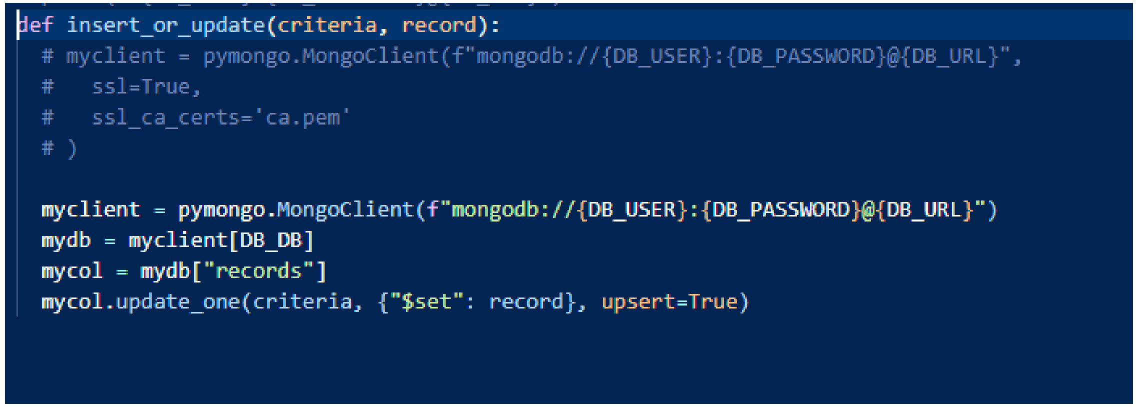
Access denial

Narrowed down range of data brokers to scrape

- Selenium: LinkedIn (login with my personal account), PeekYou, ThatsThem, Findpeoplesearch
- Twitter: login and utilize Twitter API
- Dynamically generated HTML and random CSS class names in Facebook

MongoDB Cloud Database Implementation

- A database is required to dynamically feed the front-end with newly scraped data
- MongoDB ATLAS 30 day free trial
- Call function in db.py to update the database upon scrape



```
def insert_or_update(criteria, record):
  # myclient = pymongo.MongoClient(f"mongodb://{DB_USER}:{DB_PASSWORD}@{DB_URL}",
  #   ssl=True,
  #   ssl_ca_certs='ca.pem'
  # )

  myclient = pymongo.MongoClient(f"mongodb://{DB_USER}:{DB_PASSWORD}@{DB_URL}")
  mydb = myclient[DB_DB]
  mycol = mydb["records"]
  mycol.update_one(criteria, {"$set": record}, upsert=True)
```

mongoDB® Atlas

Vue.JS front-end Framework

Initial exploratory analysis suggested that data were highly inconsistent between the different data broker websites

Optimal methodology will be to ask users to verify the scraped data themselves

Thus, the need to setup a front-end framework

Node.JS front-end server is also server to provide the restAPI for front-end, which queries MongoDB dynamically.

Queries are displayed in UI.

Vue.js

Hosted
On

HEROKU

DEMO

Privacy Considerations

- Survey was implemented to ask for user’s consent before scraping their data on the variety of websites and storing the data on MongoDB cloud database
- MongoDB stored data will be deleted at the conclusion of the project.

Goals Achieved

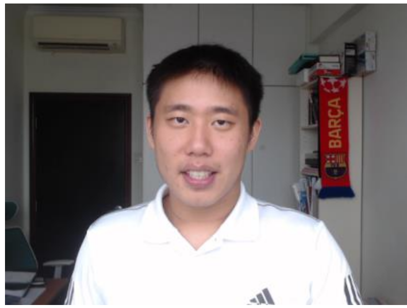
Created a dynamic code base that is modular: can easily add on additional data broker websites and additional user subjects to scrape data

Front-end framework to display data and ask for user verification, easily implement additional data brokers and users

Implemented pilot study on ~20 test subjects (university students, adults, professors), and preliminary findings suggest that:

Much of the publicly available and free information on data broker websites are unreliable

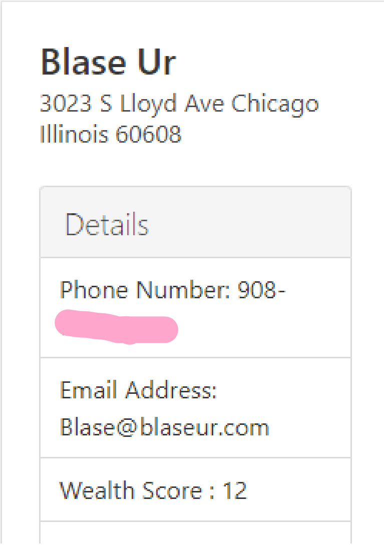
Searches through Google, LinkedIn, Facebook, and Twitter can offer more insights into personal data



Preliminary Findings

- Certain data on a same individual from a data broker website may be accurate while others may refer to another person of a similar name
- Combined with pre-knowledge on certain individuals, cross-verification across different data broker websites allow us to generate certain (*possibly*) accurate results on target individuals

We know that Blase lives in Chicago, since he is a professor at UChicago.



Blase Ur
3023 S Lloyd Ave Chicago
Illinois 60608

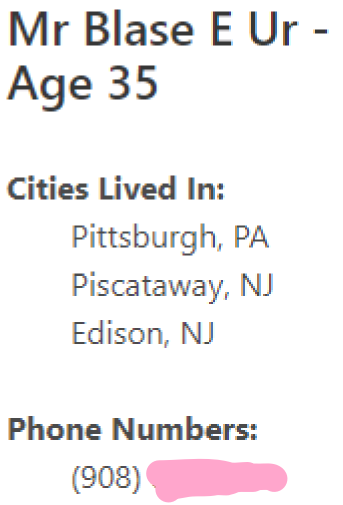
Details

Phone Number: 908- [redacted]

Email Address: Blase@blaseur.com

Wealth Score : 12

thatsthem



Mr Blase E Ur - Age 35

Cities Lived In:
Pittsburgh, PA
Piscataway, NJ
Edison, NJ

Phone Numbers:
(908) [redacted]

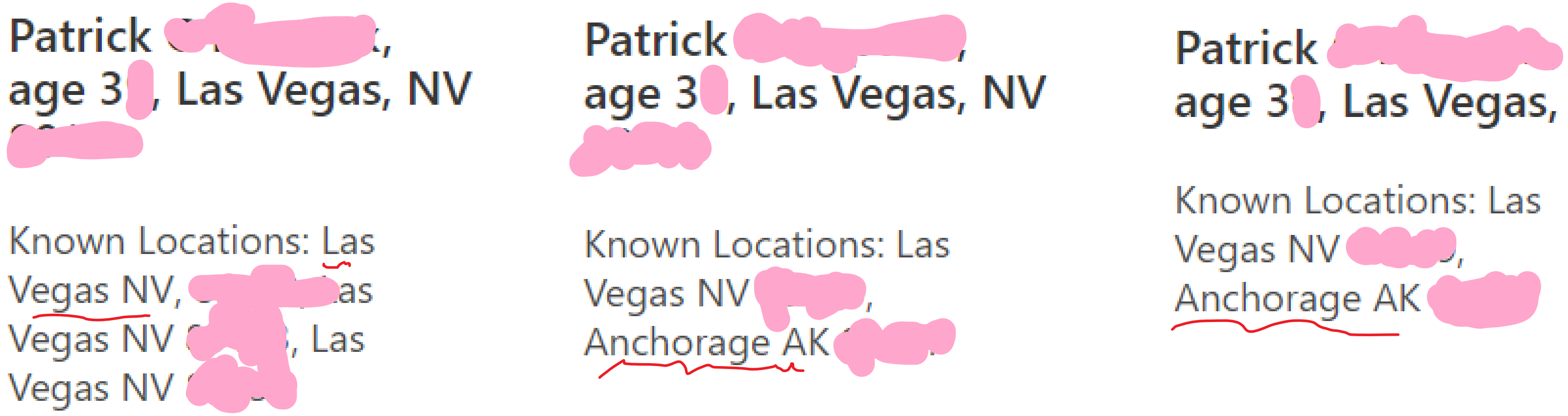
Findpeoplesearch

We can guess Blase's phone number to be (908) [redacted]

***Censored to protect Blase's privacy. Data accuracy verified by Blase.*

Other Preliminary Findings

- Certain relatively unique names that have fewer search results on major search engines like Google, Facebook, seem to generate more accurate results on data broker sites.
- Age Data is often an approximation, and varies even within a single data broker website
- Location Shifts may not be well-documented => Present location indeterminate
- Example: Patrick **



Patrick [redacted],
age 3[redacted], Las Vegas, NV
[redacted]

Known Locations: Las Vegas NV, [redacted]as Vegas NV [redacted], Las Vegas NV [redacted]

Patrick [redacted],
age 3[redacted], Las Vegas, NV
[redacted]

Known Locations: Las Vegas NV [redacted], Anchorage AK [redacted]

Patrick [redacted]
age 3[redacted], Las Vegas, NV

Known Locations: Las Vegas NV [redacted], Anchorage AK [redacted]

From thatsthem. All 3 verified by user.

Future Work

Significantly expand user subjects experimented

Through payments, access more opaque data broker sites such as Pipl which might provide more accurate and complete data

Implement login system so that users can only search for their own personal data on the website

Implement machine learning model to compare across a greater variety of different data brokers and figure out all the possible correct personal details on a target individual