CHOOSING SCRAPY[*]

Daniel Myers and James W. McGuffee
Department of Computer Science
Northern Kentucky University
Highland Heights, KY 41099
(859) 572-6930
http://cs.nku.edu

ABSTRACT

Scrapy is an open-source web-crawling platform implemented in Python. As part of an undergraduate research project, the lead author created a web crawler with Scrapy that utilizes regular expressions to extract relevant data from geographically targeted websites. This student project is part of a larger, longer term research project to determine the viability of creating an interactive geographically constrained social services client matching system. This paper describes the reason Scrapy was chosen for this project and explores the viability of Scrapy as a powerful and robust tool for undergraduate projects in the computing sciences.

1. BACKGROUND AND MOTIVATION

There are numerous entities and organizations that exist to provide social services. These organizations may be governmental, non-profit, and/or parochial. The services provided may be long term institutionalized support, seasonal (e.g. holiday meal), or limited time benefits. Beneficiaries of social services may be restricted by age, gender, family status, citizenship, and income.

Given this variability, we envision the usefulness of creating an interactive online geographically constrained social services client matching system. In this system, a client (or a social worker on behalf of a client) would enter basic demographic information and the system would return a set of available social services within a specific geographic area that addressed the clients stated needs. In addition to the obvious social benefits, this project is attractive because it fits a pedagogical need. Namely, to identify a longer term

[*] Copyright © 2015 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

sustainable project that is non-trivial yet also accessible for undergraduate students to make a significant contribution in development. The benefits of engaging undergraduate STEM students in research at an early stage have been well documented [2, 5].

In the next section of this paper, we give a broad overview of the history of the work done and challenges encountered in extracting geographic information from the web. In section 3, we briefly describe Scrapy, an open-source framework that is implemented using the Python programming language. The details of the student project conducted in the spring of 2014 are given in section 4.

2. HISTORY OF ASSOCIATING GEOGRAPHIC LOCATIONS WITH WEB CONTENT

As early as 1999, a team led by Orkut Buyukkoten reported on a Java application they had written to demonstrate the feasibility of extracting geographical location information from web pages [3]. They determined that trying to determine geographical information from an analysis of network IP addresses was problematic because many sites used remote hosting services where the physical location of the network was unrelated to the geographical location of the content of the website. The team argued that a much more accurate way to determine the geographical location of information on the site was to attempt to extract zip codes from the web sites.

In 2000, Ding, Gravano, and Shivakumar formally described the geographical scope of a web resource, w, as the geographical area that the creator of the w intends to reach [4]. In addition to examining author intent, Ding and his collaborators advocated the usefulness of examining the geographical distribution of hyperlinks to a web page to assist in determining the geographical scope of the page. The idea is that by examining the location of the pages that link to a page you can begin to begin to analyze the geographic reach of a specific site. By examining both the textual content of a page and the geographical distribution of hyperlinks referencing a page, they were trying to create a more personalized search engine that would constrain browsing relevance by geographic location.

In 2004, the Web-A-Where project created a gazetteer based on contextualized content of web pages [1]. The goal of the project was to locate any mention of a place and determine the exact geographic location that was being referenced. The project's authors described two major types of ambiguities that add challenge to this process. The first ambiguity is when a word can have a geographic and a non-geographic meaning (e.g. Turkey). The second ambiguity is when a word can reference two or more distinct geographical locations (e.g. Springfield). Despite these challenges, the Web-A-Where project reported a precision of up to 82% in the creation of their gazetteer.

In contrast to the Web-A-Where project, Wang and colleagues developed a system to create a gazetteer based on the structures of hyperlinks and user logs [9]. In 2006, Tezuka and colleagues criticized previous work in this area as superficial and merely an extension of current GIS technologies [11]. They argued that to better understand the geographical reach of a website you need to know and analyze the geographical location of the user and the geographical search history of that user.

More recently, in 2011, Sengstock and Gertz proposed a system to determine geographic information sources on the web using distance statistics [9]. They proposed the use of spatial proximity relationships such as co-location patterns and spatial association rules. Sengstock and Gertz demonstrated the feasibility of identifying groups of geographic features with distinct geographic semantics.

As is seen by the review of projects over the past decade and a half, there have been many attempts to identify geographic information associated with websites. However, there has been no consensus as to the best way to solve this problem. Also, much of the work done in this area was specific to domains outside the scope of our research project (e.g. geographical scope of the readers of an online newspaper). The goal for our project was much more limited and narrowly defined. Specifically we wished to examine the feasibility of data mining based on geographical filters using Scrapy, an open source tool [6]. Our project attempted to assess the viability of using Scrapy as a tool to geographically constrain the scraping of data from web pages. Specifically, the project wanted to test how easy it was to scrape data with the geographic constraints of zip codes.

3. SCRAPY

Scrapy is an open-source web crawling framework written in Python. It's main purpose is to provide support for web scraping. Scrapy was first released on June 2, 2008 and as of March 2015 the latest version is Scrapy 0.24 and is compatible with Python 2.7 [7]. There are several ongoing Scrapy based projects and many companies use Scrapy as a tool to scrape the web. Some of these companies are CareerBuilder, DayWatch, PriceWiki, and Tarlabs [8].

Scrapy is an integrated system that includes an engine for controlling the data flow between all components, a scheduler for receiving requests, a downloader for fetching web pages, and custom classes (called spiders) written by users to parse responses and extract items [7]. For a graphical overview of Scrapy that shows how these parts interact with each other, please visit doc.scrapy.org/en/latest/topics/architecture.html.

The task of capturing and structuring data mined from the web is usually divided into two distinct phases: the crawling and the scraping portions of the task. For this project, Scrapy is ideal because it is a Python based framework that provides tools for both scraping and crawling [6]. Scrapy was also chosen because of the learning curve needed to be a competent user of Scrapy. Several platforms and tools were initially considered by the student but Scrapy was the one that was the most straightforward to learn and implement. Also, as an open source product, Scrapy has a robust community willing and able to assist new users [7,8].

4. STUDENT PROJECT

The main focus of this project was to write a web crawler that utilizes regular expressions to extract relevant data from geographically targeted websites. Specifically, the student written project used Scrapy to find all web mentions of physical addresses in the zip codes of three specific counties in northern Kentucky (Boone, Kenton, and Campbell). The two main tasks for this project were to use Scrapy's spider class to scrape the data and to create a Scrapy item to manage the data.

In Scrapy, the spider is the class which defines which web sites will be scraped, how they will be scraped, and what data will be collected and placed into the item. Our crawler is designed to extract street addresses from HTML pages. It first looks for a zip code that is in Boone, Kenton, or Campbell counties as defined in an external file. After a zip code is identified the crawler then looks for a string consisting of the state, city and street address. The full source code for the spider class is given below:

```
class locationSpider(Spider):
name=”location”          #Name of spider
# Define URLs to search
  gsearch = pygoogle('campbell county social services')
  start_urls =  gsearch.get_urls()
  def parse(self, response):
# Set up xpath selectors
  addresses = hxs.xpath(“//*/text()”)
# Create list to hold results
  items = []
# Define regular expression for finding zip codes.
 nky_zip_regex = re.compile(r'\b(' + '|'.join(locationSpider.zip_codes)
+
                  r')', re.IGNORECASE)
  for current in addresses:
   item = LocationCrawlerItem()
# If crawler identifies a zip code, try to get all data from current
node.
  zip_string = re.search(nky_zip_regex, current.extract())
if(zip_string):
# Store page title and zip code in the LocationCrawlerItem
# Compile regex statements for finding state followed by a valid zip
code.
  state_abb_regex = re.compile(r'\bky[.,]? ' + re.escape(item[“zip”])
,
                    re.IGNORECASE)
        state_proper_regex    =    re.compile(r'\bkentucky    '    +
re.escape(item[“zip”]),
                       re.IGNORECASE)
# Search current for regex patterns
# If state abbreviation is found, place it in the LocationCrawlerItem
  if(state_abb_string):
    item[“state”] = “Ky”
    temp_address_str = state_abb_string.group()
  elif(state_proper_string):
    item[“state”] = “Kentucky”
    temp_address_str = state_proper_string.group()
  else:
    item[“state”] = None
    temp_address_str = None
# Compile regex statement to find city and execute.
  city_regex = re.compile(r'\b(' + '|'.join(locationSpider.cities) +
'[,]? '
               + re.escape(temp_address_str) + r')', re.IGNORECASE)
  city_string = re.search(city_regex, current.extract())
  if(city_string):
        street_address_regex  =  re.compile(r'([0-9]+)([A-Za-z  ]+)',
re.IGNORECASE)
                    s t r e e t _ a d d r e s s _ s t r i n g     =
re.search(street_address_regex,current.extract())
```

```
  if(street_address_string):
    item[“street_address”] = street_address_string.group()
  else:
    item[“street_address”] = None
# Add LocationCrawlerItem to list of objects.
  items.append(item)
return items
```

After crawling and determining which pages to scrape, the item object was used. In Scrapy, the item defines an object to place the data in. This allows the crawler to extract structured data from unstructured sources. In the example provided we are attempting to extract all of the information necessary to identify a single physical address and place it into the item. The item can later be used to place the data into a database or print it to a file in JSON, XML or CSV.

```
class LocationCrawlerItem(Item):
    page_title = Field()
    street_address = Field()
    city = Field()
    state = Field()
    zip = Field()
```

Below is an example output after the Item has been converted to JSON and written to a file.

```
[{“city”: “Covington”, “state”: “Kentucky”, “page_title”: [“Welcome To
Kenton County “], “street_address”: “303 Court Street Covington”, “zip”:
“41011"},
{“city”: “Covington”, “state”: “Ky”, “page_title”: [“\r\n\tNorthern
Kentucky Community Action Commission\r\n”], “street_address”: “717
Madison Avenue”, “zip”: “41011"},
{“city”: “Covington”, “state”: “Ky”, “page_title”: [“Kenton County
Attorney's Office :: Departments :: Child Support”], “street_address”:
“41011 “, “zip”: “41011"},
{“city”: “Covington”, “state”: “Ky”, “page_title”: [“\nKenton County
Jail Tracker? - Ask.com\n”], “street_address”: “303 Court Street”,
“zip”: “41011"},
{“city”: “Covington”, “state”: “Ky”, “page_title”: [“\nKenton County
Jail Tracker? - Ask.com\n”], “street_address”: “303 Court Street”,
“zip”: “41011"},
{“city”: “Covington”, “state”: “Ky”, “page_title”: [“Kenton County Homes
\u2013 The Point ARC of N. KY”], “street_address”: “104 West Pike Street
Covington”, “zip”: “41011"},
{“city”: “Newport”, “state”: “Ky”, “page_title”: [“Campbell County
Sheriff”], “street_address”: “1098 Monmouth Street Suite “, “zip”:
“41071"},
{“city”: “Newport”, “state”: “Ky”, “page_title”: [“Campbell County
assistance programs | Newport”], “street_address”: “437 West Ninth
Street”, “zip”: “41071"},
{“city”: “Newport”, “state”: “Ky”, “page_title”: [“Campbell County
assistance programs | Newport”], “street_address”: “1098 Monmouth
Street”, “zip”: “41071"}]
```

This limited scope project demonstrated the feasibility of using Scrapy as part of a project to create an interactive geographically constrained social services client matching system. The next phase of the project will involve the verification of the data scraped and adding filters to try and disambiguate data related to viable social services from all other zip code related date. In other words, does the information collected via scraping correspond to reliable current information regarding real world organizations and services offered?

5. CONCLUSION

By using the open source tool Scrapy, the problem of extracting geographically relevant information becomes more manageable when the scope of the search is narrowed to a specific geographic area. For instance, instead of maintaining a gazetteer (database of complete locations) in order to verify location information, it was possible to compile lists containing all cities and zip codes in northern Kentucky to allow the spider to utilize in regular expressions. Due to the reduced data set, extracting information becomes faster and more accurate.

More broadly, we were pleased that an undergraduate student was able to get a Scrapy based project up and running within a short period of time. One of the observed advantages is that the syntax of Python aids in the learning curve of the Scrapy toolset. As reported recently in the *Communications of the ACM*, the Python programming language tops the list of programming languages used to teach introductory courses [10]. The same relatability reasons that make Python a good choice as an introductory language, make Scrapy a good choice for web scraping projects.

As mentioned earlier, the effort to create a geographically constrained social services locator is intended to be an ongoing student project. If there is interest in joining the effort, please feel free to contact the faculty author. There has been some discussion of making this an open source project using github. Thus, there would be no geographic constraints for collaboration. More broadly, we would encourage you to explore Scrapy and see if you can find a project that you find interesting. If you do, please let us know about your project.

REFERENCES

[1] Einat Amitay, Nadav Har'El, Ron Sivan, and Aya Soffer. 2004. Web-A-Where: geotagging web content. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '04). ACM, New York, NY, USA, 273-280. DOI=10.1145/1008992.1009040

[2] Bethany Bowling, Heather Bullen, Maureen Doyle, and John Filaseta. 2013. Retention of STEM majors using early undergraduate research experiences. In *Proceedings of the 44th ACM Technical Symposium on Computer Science Education (SIGCSE '13)*, ACM, New York, NY USA, 171-176. DOI=10.1145/2445196.2445249

[3] Orkut Buyukkoten, Junghoo Choo, Hector Garcia-Molina, Luis Gravano, and Narayan Shivakumar. 1999. Exploiting geographical location information of web pages. In *Proceedings of the ACM SIGMOD Workshop on the Web and Databases* (WebDB '99). ACM, New York, NY, USA. Accessed electronically <www.cs.columbia.edu/~gravano/Papers/1999/webdb.pdf> on 24 MAY 2014.

[4] Junyan Ding, Luis Gravano, and Narayanan Shivakumar. 2000. Computing Geographical Scopes of Web Resources. In *Proceedings of the 26th International Conference on Very Large Data Bases* (VLDB '00), Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 545-556.

[5] Anne-Barrie Hunter, Sandra L. Laursen, and Elaine Seymour. 2007. Becoming a scientist: the role of undergraduate research in students' cognitive, personal, and professional development. *Science Education*, 91, 36-74. DOI=10.1002/sce.20173

[6] Kate Matsudaira. 2014. Capturing and structuring data mined from the web. *Communications of the ACM* 57, 3 (March 2014), 10-11. DOI=10.1145/2567664

[7] Scrapy documentation, <doc.scrapy.org>, accessed 17 March 2015.

[8] Scrapy wiki, <github.com/scrapy/scrapy/wiki>, accessed 15 March 2015.

[9] Christian Sengstock and Michael Gertz. 2011. Exploration and comparison of geographic information sources using distance statistics. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (GIS '11), Divyakant Agrawal, Isabel Cruz, Christian S. Jensen, Eyal Ofek, and Egemen Tanin (Eds.). ACM, New York, NY, USA, 329-338. DOI=10.1145/2093973.2094017

[10] Esther Shein. 2015. Python for Beginners. *Communications of the ACM* 58, 3 (March 2015), 19-21. DOI=10.1145/2716560

[11] Taro Tezuka, Takeshi Kurashima, and Katsumi Tanaka. 2006. Toward tighter integration of web search with a geographic information system. In *Proceedings of the 15th International Conference on the World Wide Web* (WWW '06). ACM, New York, NY, USA, 277-286. DOI=10.1145/1135777.1135821

[12] Chuang Wang, Xing Xie, Lee Wang, Yansheng Lu, and Wei-Ying Ma. 2005. Detecting geographic locations from web resources. In *Proceedings of the 2005 Workshop on Geographic Information Retrieval* (GIR '05). ACM, New York, NY, USA, 17-24. DOI=10.