URL Parsing Challenge

William Baron, Tiffany Lee, Emma Peatfield, Yudhvir Singh, Sterling Tarng
Computer Engineering Department
San Jose State University (SJSU)San Jose, CA, USA
Email: {william.baron, tiffany.lee, emma.peatfield, yudhvir.singh. sterling.tarng}@sjsu.edu

***Abstract*—Being able to efficiently parse URLs and extract info from them is a very important skill for a data scientist.**

Firstly, if you join a very early stage startup, it might be that a lot of data are just stored in the URLs visited by the users. And, therefore, you will have to build a system that parses the URLs, extracts fields out of it, and saves them into a table that can be easily queried.

Secondly, often using external data can help a lot your models. A way to get external data is by scraping websites. And this is often done by being able to play with a given site URL structure (assuming it is allowed by the external site ToS or it is not allowed, and you don't get caught).

The goal of this project is to parse a sequence of URLs about user searches and extract some basic info out of it.

Keywords—Data Science, Data Processing, URL, Parsing

I. Introduction

In this URL parsing challenge, we are asked to look into Company XYZ and see what we can do for their raw data they receive. Company XYZ is an Online Travel Agent site, such as Expedia, Booking.com, etc.

They haven't invested in data science yet and all the data they have about user searches are simply stored in the URLs generated when users search for a hotel. If you are not familiar with URLs, you can run a search on any OTA site and see how all search parameters are present in the URL.

You are asked to: Create a clean data set where each column is a field in the URL, each row is a given search and the cells are the corresponding URL values.

For each search query, how many amenities were selected? Often, to measure the quality of a search algorithm, data scientists use some metric based on how often users click on the second page, third page, and so on. The idea here is that a great search algorithm should return all interesting results on the first page and never force users to visit the other pages (how often do you click on the second page results when you search on Google? Almost never, right?).

Create a metric based on the above idea and find the city with the worst search algorithm.

II. Data Preprocessing

A. The Data Given

We are given raw URL data in a CSV file called "url_list", which is a list of search URLs generated by the users when searching for hotels. We are given the check-in date, minimum price filter, maximum price filter, if the cancellation is free, what hotel star rating they might be searching for (they can choose any combination of stars), the max score and min score of the hotels they would like to stay at, and the number of adults and children that might be staying in a room. There is also check-out information, if there are certain amenities (that the user can search for), as well as the page rank of the search - which tells us what page the customer is on in terms of search pages. Figure 1 shows what our raw URL data looks like from the file.

```
1 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  19&hotel.stars_4=yes&hotel.min_score=4&hotel.adults=3&hotel.city=New+York,+NY,+United+States&hotel.checkout=2015-09-
  20&hotel.search_page=1
2 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  14&hotel.stars_3=yes&hotel.min_score=4&hotel.adults=3&hotel.city=London,+United+Kingdom&hotel.checkout=2015-09-15&hotel.search_page=1
3 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  26&hotel.customMaximumPriceFilter=175&hotel.stars_4=yes&hotel.min_score=5&hotel.adults=2&hotel.city=New+York,+NY,+United+States&hotel.che
  ckout=2015-09-27&hotel.search_page=1
4 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  02&hotel.stars_5=yes&hotel.stars_4=yes&hotel.min_score=4&hotel.adults=1&hotel.city=Hong+Kong,+Hong+Kong&hotel.checkout=2015-09-
  03&hotel.search_page=1
5 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  20&hotel.customMaximumPriceFilter=275&hotel.min_score=5&hotel.adults=3&hotel.city=London,+United+Kingdom&hotel.checkout=2015-09-
  29&hotel.search_page=1
6 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  14&hotel.freeCancellation=yes&hotel.adults=2&hotel.city=San+Francisco,+California,+United+States&hotel.checkout=2015-09-
  16&hotel.search_page=1
7 http://www.mysearchforhotels.com/shop/hotelsearch?hotel.checkin=2015-09-
  20&hotel.customMaximumPriceFilter=125&hotel.stars_2=yes&hotel.min_score=5&hotel.adults=1&hotel.city=Hong+Kong,+Hong+Kong&hotel.checkout=2
  015-09-21&hotel.search_page=1
```

Fig. 1. Our raw URL data given to us.

It is important to note that if a field is not in the URL, it means the user didn't filter by it. So, for instance, if in this search the user didn't filter by a certain max/min price, amenities, etc. that means that the information was optional and not necessary.

B. Considerations

Before we create a table, we have to consider important factors that will make our data processing easier. We need to display all the information in a easy to read, easy to query format. Making an optimal table is of utmost importance as we will perform operations on the table to get the information we need out of it. It is important to take into consideration that not every URL has all the potential information, as some of the selections are optional. Our data set is also very large, so we need to make sure we are transcribing the data into our table in an efficient manner as well.

III. Solution

A. Data Processing

```
from urllib.parse import urlparse, parse_qs, urlsplit
n = pd.DataFrame(data={parse_qs(urlsplit(data[0][line]).query) for line in range(len(data))})
for column in n:
    n[column] = n[column].str[0]
```

```
n = n.rename(columns={'hotel.adults': 'Adults', 'hotel.amenities': 'Amenities', 'hotel.checkin': 'Check-In',
                      'hotel.checkout': 'Check Out', 'hotel.children':'Children', 'hotel.city':'City',
                      'hotel.couponCode':'Coupon', 'hotel.customMaximumPriceFilter':'Max-Price',
                      'hotel.customMinimumPriceFilter':'Min-Price','hotel.freeCancellation':'Free Cancellation',
                      'hotel.max_score':'Max-Score','hotel.min_score':'Min-Score','hotel.search_page':'Search Page',
                      'hotel.stars_1':'1 Star','hotel.stars_2':'2 Stars','hotel.stars_3':'3 Stars','hotel.stars_4':'4 Stars',
                      'hotel.stars_5':'5 Stars'})
```

Fig. 2. Processing the URLs.

We discovered a library, urllib, that lets us perform URL parsing and splitting, amongst other things. This proved extremely useful in our task to parse the data into the format we needed. From there, we split the data into the format we wanted to by creating a column for each variable data we had. Refer to Figure 3 for a snapshot of what our data looked like after we parsed it.

	Adults	Amenities	Check-In	Check Out	Children	City	Coupon	hotel.customMaximumPriceFilter	Min-Price	Free Cancellation	Max-Score	Min-Score	Search Page
0	3	NaN	2015-09-19	2015-09-20	NaN	New York, NY, United States	NaN	NaN	NaN	NaN	NaN	4	1
1	3	NaN	2015-09-14	2015-09-15	NaN	London, United Kingdom	NaN	NaN	NaN	NaN	NaN	4	1
2	2	NaN	2015-09-26	2015-09-27	NaN	New York, NY, United States	NaN	175	NaN	NaN	NaN	5	1
3	1	NaN	2015-09-02	2015-09-03	NaN	Hong Kong, Hong Kong	NaN	NaN	NaN	NaN	NaN	4	1

Fig. 3. Processing the URLs.

Here, you can see that for the data that was not given to us, a 'Null' was put in that cell. We can now easily tally or query through the data and can proceed to our ultimate task, which is to figure out how to query our new table and get valuable information out of it.

To gather how many users had searched for amenities, for example, all we needed was to count through our amenities column and count how many had searched for amenities - which values were not null, and return that value. Likewise, to find what cities had the best page rank displays, we just tallied which cities had what ranks and divided by how many times the cities appeared to find the best page ranks.

References

[1] https://docs.python.org/3/library/urllib.html

[2] https://docs.python.org/3/howto/urllib2.html

[3] https://pythonprogramming.net/urllib-tutorial-python-3/