

BGSE / Data Science / Text Mining / Web Scraper

Roger Cuscó, Matthew Sudmann-Day, Miquel Torrens

We scraped the website ‘[RightMove](#)’ which advertises properties for sale and for rent throughout the UK. We chose to limit ourselves to properties for rent. The documents we extracted are the descriptions of the properties as written by the owner or agent of the property. Along with the document text, we also scraped a number of metadata attributes about each property.

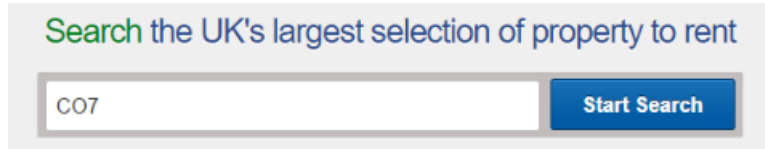
The scraper was written in Python with the Beautiful Soup screen-scraping library. You can find our code and this document in our Github repository: <https://github.com/m-sudmann-day/BGSE-text-mining/tree/master/scraper>.

Table of Contents

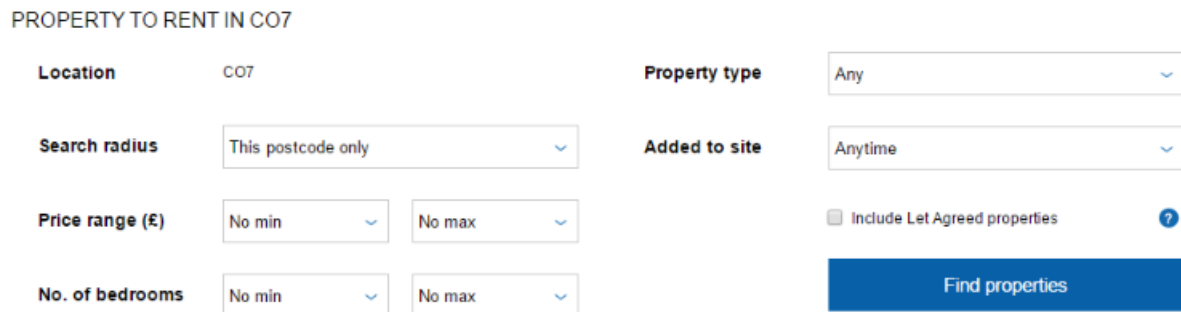
| | |
|--------------------------------|----|
| Navigation | 2 |
| Being a Good Citizen | 5 |
| Application | 6 |
| Document Storage | 6 |
| Sample Output | 7 |
| Properties of The Corpus..... | 8 |
| Handling Failure | 8 |
| Restarting After Failure | 9 |
| Simultaneity | 9 |
| Creating the Database..... | 10 |
| Running the Application | 10 |

Navigation

The web scraper mimics the actions of a user. When looking at properties for rent, the first thing the user must do is choose an “outcode”. In the UK, an outcode is the 3- or 4- digit prefix of a full postcode.

A search bar with the text "Search the UK's largest selection of property to rent" in green and blue. Below the text is a white input field containing "CO7" and a blue button labeled "Start Search".

The user is then presented with a search screen for that outcode:

A search results screen titled "PROPERTY TO RENT IN CO7". It features several filters: "Location" (CO7), "Property type" (Any), "Search radius" (This postcode only), "Added to site" (Anytime), "Price range (£)" (No min, No max), and "No. of bedrooms" (No min, No max). There is a checkbox for "Include Let Agreed properties" and a blue button labeled "Find properties".

We choose not to enter any filters. We do not check ‘Let Agreed’ properties as this will introduce some bias as we don’t know what RightMove will choose to show and not show here. So we limit ourselves to properties that are not yet let agreed, i.e. just those that are still available for rent.

Upon clicking ‘Find Properties’, we are navigated to the following URL:

http://www.rightmove.co.uk/property-to-rent/find.html?searchType=RENT&locationIdentifier=OUTCODE%5E520&insId=3&radius=0.0&minPrice=&maxPrice=&minBedrooms=&maxBedrooms=&displayPropertyType=&maxDaysSinceAdded=&sortByPriceDescending=&_includeLetAgreed=on&primaryDisplayPropertyType=&secondaryDisplayPropertyType=&oldDisplayPropertyType=&oldPrimaryDisplayPropertyType=&letType=&letFurnishType=&houseFlatShare=false

Manual experimentation tells us that the following simpler URL will have the same effect:

<http://www.rightmove.co.uk/property-to-rent/find.html?locationIdentifier=OUTCODE%5E520>

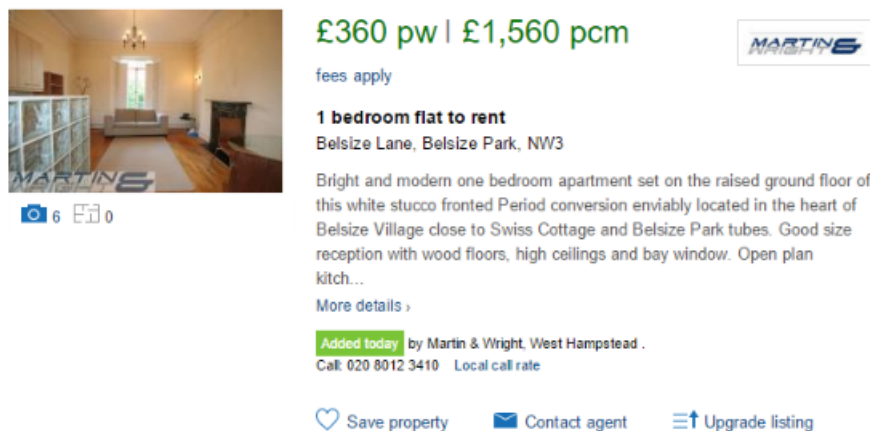
This is the format of URL that the scraper uses to extract pages of search results. Note that the outcode has been converted by the website from CO7 to 520. 520 is RightMove’s internal ID for the outcode CO7, and the only ways to discover the mappings between outcodes and the internal IDs is manual discovery or an automated loop to try them all. We discovered that the internal IDs covering the whole of the UK ranged from 1 to 2917.

To scrape the entire UK would take approximately a week and would not be likely to add a lot of intuition to this exercise. Therefore, we decided to limit our scope for actual text extraction to the city of Edinburgh. This corresponds to the outcodes EH1 through EH17. The RightMove internal IDs for these outcodes, in order, are 793, 804, 815, 826, 837, 843, 844, 845, 846, 794, 795, 796, 797, 798, 799, 800, 801. We used these internal IDs to construct new URL's to get lists of search results.

We further modify the URL to get as many results as permitted on a single page. The website offers users the option to see 10, 20, or 50 properties on a page. Unfortunately, we are not able to enter a number larger than 50.

<http://www.rightmove.co.uk/property-to-rent/find.html?searchType=RENT&locationIdentifier=OUTCODE%5E801&numberOfPropertiesPerPage=50>

As a user executing this query, we see a page of (up to 500) results that look like this:



At the bottom of the page, we see the paging controls:

page 1 of 5 previous **1** 2 3 4 5 next

The web scraper must simulate clicking through each page of results to the last page. This is done by further modifying the URL. The query element index specifies the starting index of the first ad on the page. As we are permitted a maximum of 50 results per page, we request indexes 0, 50, 100, etc. In the URL below, we request the third page of properties.

<http://www.rightmove.co.uk/property-to-rent/find.html?locationIdentifier=OUTCODE%5E801&numberOfPropertiesPerPage=50&index=100>

The first piece of actual “scraping” that is necessary is to extract the ID of each property on a page. This is embedded in the link under the image and the title of the property. The property shown above has an ID of 59453477. Although some relevant property data can be scraped directly from the search results, most is not, and the description is truncated. Therefore, we “step into” every property on every page of every search query. To visit the page for the property above, we must use the following URL.

<http://www.rightmove.co.uk/property-to-rent/property-59453477.html>

The page for the property has the following components of interest to us:

Title and cost:

2 bedroom maisonette to rent
Lower Ground, South Hill Park, Hampstead, NW3

£385 pw | £1,668 pcm
fees apply

Metadata:

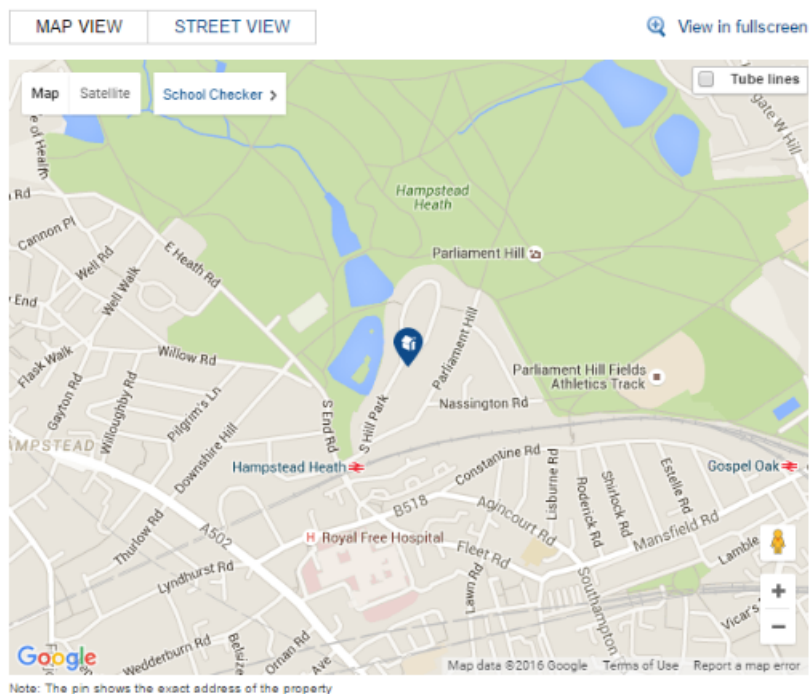
Letting information:

Date available: 11/06/2016
Furnishing: Furnished or unfurnished, landlord is flexible
Added on Rightmove: 16 May 2016 (16 minutes ago)

Full description

A spectacular lower ground apartment. The property comprises one large double bedroom, second single bedroom, large reception room, kitchen and family bathroom. Perfectly placed on a quiet road close to buses, Hampstead heath rail and Belsize park station.

Location (which we extract from the Google map link):



To summarize the navigation challenges, here is pseudocode to describe the journey our web scraper takes:

```

For each outcode:
{
    Index = 0
    Navigate to the search results using RightMove's internal ID for the outcode
    starting with
    ad index Index.
    While there are no search results not yet seen:
    {
        For each ad on the search results page:
        {
            Extract the property ID.
            Navigate to the property page.
            Extract description text.
            Extract price.
            Extract location.
            Extract other metadata.
        }
    }
    Index = Index + 50
}

```

Being a Good Citizen

When scraping a site, it is appropriate to honour the instructions provided by the owners of the website in the robots.txt file.

The only robot whose identity we would match is the wildcard user agent '*'. Under the rules for this robot, a large number of paths are disallowed. We scrape only the following paths:

```
/property-to-rent/property-*.html
```

```
/property-to-rent/find.html
```

These do not match any of the disallowed paths. Therefore, we have not violated any of the rules in the robots.txt file.

No features were specifically implemented to adhere to these rules.

It takes approximately two seconds for our scraper to scrape one property. We intentionally add a delay of another two seconds between properties. This is to avoid the website from rejecting us, but also to reduce the burden on the site. To keep ourselves somewhat less noticeable, we run the scraper on the Edinburgh data in the middle of the day when a lot of other users would also be using the site.

Application

The web scraper was written in Python 2.7 and used the following packages:

- `bs4` (Beautiful Soup 4, a screen-scraping library)
- `requests` (for requesting HTML from a URL)
- `re` (for regular expressions)
- `time` (for implementing intentional delays)
- `MySQLdb` (a connector for MySQL)

Document Storage

To associate our documents more easily with their metadata, we stored both the metadata and the documents together in a MySQL database. Our database has the following tables:

- **outcode**
 - `id` (primary key) – the internal ID used by RightMove to identify the outcode
 - `completed` – 1 if scraping of the outcode has completed
- **property**
 - `id` (primary key) – the ID used by RightMove to identify the property
 - `outcode_id` (foreign key to `outcode(id)`) – the outcode ID
 - `url` – the URL of the scraped property, useful for validation purposes
 - `title` – the short title used to advertise the property
 - `monthly_price` – the price per month advertised
 - `weekly_price` – the price per week advertised
 - `latitude` – the latitude of the location of the property
 - `longitude` – the longitude of the location of the property
 - `description` – the text that defines our ‘document’, the description of the property that appears in the advertisement
- **error**
 - `obj_type` – the type of the Python object that reported the error (Outcode or Property)
 - `id` – the ID of the Python object that reported the error
 - `text` – the text of the Python exception

Another approach would have been to store the metadata within a text file that also contained the document. This would merely force another round of scraping, to correctly parse the metadata out of the text file. This approach keeps the various fields associated with the document separate. It also allows queries to be run across documents in the database if that makes more sense than loading all of them and then doing an analysis in memory.

Sample Output

Sample output as stored in the property table looks like this:

| id | outcode_id | url | title | description | monthly_price | weekly_price | latitude | longitude |
|----------|------------|---|----------------------------------|---|---------------|--------------|----------|-----------|
| 39856245 | 1822 | http://www.rightmove.co.uk/property-to-rent/property-39856245.html | 2 bedroom house to rent | We are pleased to offer this STUNNING, fully furni... | 1100 | 254 | 52.8329 | 1.2503 |
| 40190835 | 1822 | http://www.rightmove.co.uk/property-to-rent/property-40190835.html | 3 bedroom detached house to rent | Opening to kitchen, latched wood door utility room | 1200 | 277 | 52.8758 | 1.32855 |

An example document as contained within the description field looks something like this:

We are pleased to offer this STUNNING, fully furnished, 2 bedroom Coach House on the Grange Estate, set in the rural village of Erpingham. The property sits within a lovely quiet spot, deep in the country side. **FREE ADMIN** Porch leading to Open plan lounge/kitchen. Fully furnished character filled Lounge area with gas fireplace, double aspect windows and fully tiled floor, leading through to a stunning vaulted ceiling kitchen area, with an impressive island, ideal for entertaining or food preparation. Kitchen is fully fitted with integrated appliances, including 6 ring electric hob and dishwasher. Electric heater. Master Bedroom AMAZING LARGE en-suite double bedroom, double bed, fitted wardrobe and dressing table. The Bedroom also benefits from field views. En-suite wet room with fully tiled shower area and wash basin. Electric heater. Carpet. Bedroom 2 En-suite DOUBLE bedroom, with side aspect window. Includes bed and double wardrobe. En-suite includes Feature freestanding ROLL TOP BATH, toilet, bidet and wash basin. Electric heater. Carpet. Hallway Large tiled hallway with glass doors leading to parking space. Currently being used as a dining area. Electric heater. As part of our application process, fees will become due for referencing, tenancy agreement administration and an inventory check, these will be charged in addition to the Rent and Deposit that will be payable before the tenancy starts. Please contact our Branch for full details of the fees payable before you make any decision about this property or before you decide to view this property. Our Branch staff can provide you with an explanation of how these fees are calculated, please note that the referencing fees are charged per individual and should a Guarantor be required, this would attract additional referencing fees. While every reasonable effort is made to ensure the accuracy of descriptions and content, we should make you aware of the following guidance or limitations. (1) MONEY LAUNDERING REGULATIONS "pro prospective tenants will be asked to produce identification documentation during the referencing process and we would ask for your co-operation in order that there will be no delay in agreeing a tenancy. (2) These particulars do not constitute part or all of an offer or contract. (3) The text, photographs and plans are for guidance only and are not necessarily comprehensive. (4) Measurements: These approximate room sizes are only intended as general guidance. You must verify the dimensions carefully to satisfy yourself of their accuracy. (5) You should make your own enquiries regarding the property, particularly in respect of furnishings to be included/excluded and what parking facilities are available. (6) Before you enter into any tenancy for one of the advertised properties, the condition and contents of the property will normally be set out in a tenancy agreement and inventory. Please make sure you carefully read and agree with the tenancy agreement and any inventory provided before signing these documents.

Properties of The Corpus

Our corpus of documents containing advertisements for properties for rent in Edinburgh has the following properties:

- There are 871 documents. (If scraping the entire UK, this number would be closer to 200,000.)
- 27 documents have no length.
- Of those that have length:
 - The minimum document length is 14 terms.
 - The average document length is 149 terms.
 - The maximum document length is 1028 terms.
 - The standard deviation of the document length is 102 terms.

Handling Failure

There are countless ways that a web scraper can fail. An obvious one is that the website could change in large or subtle ways breaking the scraper's code. In our case, we struggled with the fact that the website would sometimes conclude that we were requesting a mobile site. This prevented us from using the enabled/disabled status of the 'next' button to determine whether there were more pages of search results to scrape. As a result, we had to use a slightly less elegant solution in which we kept asking for more pages until we received no further results that we had not previously seen.

Sometimes, data is not clean:

A rare property would fail to provide a weekly price. For those, we store `NULL` in the database and leave it to the application that uses that `metadat` to handle that case.

We used regular expressions and were forced to explicitly handle the cases in which there were no matches, too many matches, and so forth.

`BeautifulSoup` did not seem completely robust and would not correctly scan for element classes. In the case of links, of which there are an enormous number on a single page, we were forced to scan every link and interrogate the attributes ourselves.

Ultimately, it's not feasible to believe we could handle all errors. Therefore, we capture and log any errors that fall through the cracks. This will also capture the predictable errors relating to the website refusing access, interruptions in the network, and so forth. In a production setting, we would go much further with this piece, storing timestamps, call stacks, etc. For this, we simply logged the object type (outcode or property), its ID, and the text of the error to an 'error' table in the database. The application code reflects this pattern in the following manner:


```

if self.isFinished():
    return

try:
    # Try to scrape the website
    self.start()

    self.scrape()

    if not self.errorOccured:
        self.finish()

except Exception as ex:
    # If an error occurs write the incidence in the log file
    LogError(self.db, "Outcode", outcodeId, ex)

```

An error that was written to the database while we were scraping was this:

| obj_type | id | text |
|----------|----------|---|
| Property | 53999200 | HTTPConnectionPool(host=www.rightmove.co.uk, port=80): Max retries exceeded with url: /property-to-rent/property-53999200.html (Caused by NewConnectionError(<requests.packages.urllib3.connection.HTTPConnection object at 0x00000000069EB780>: Failed to est... |

Restarting After Failure

Sometimes, a scraping session will end without completing successfully, but we do not want to start over from the beginning. Therefore, the Outcode table has a completed flag. If the scraper is instructed to scrape the properties in an outcode, it will return immediately if the Outcode has already been scraped.

If a previous session started, but did not complete scraping the properties of an outcode, then scraping of the outcode search results will start over from the beginning, but individual properties will only be scraped if they do not yet have an entry in the Property table.

These two checks, for completion of the outcode, and for presence of the property, mean that we can dramatically reduce the amount of web scraping that happens after a session terminates without completing.

Simultaneity

An unavoidable issue that the user of the data should be aware is that web scraping takes time. Therefore, there are potential inconsistencies and biases. If we were to scrape the whole UK, some of our outcodes would have data one week older than the others. Perhaps the code could be modified to address this with some random sampling. It could be run multiple times and the unique set of

properties combined such that the differences between the outcodes is blurred. However, at that point, the problem becomes that some properties will have their advertisements updated; do we use the oldest, the newest, keep them both, etc.? These questions can probably never be answered perfectly and the conclusions drawn from the data should probably be broad enough that precise timing is not a significant issue.

Creating the Database

Log into MySQL as an administrator. Run the Scraper.sql file you find in our Github repository. As it is a short script, it is embedded here for ease of access:

```
CREATE DATABASE `scraper2` /*!40100 DEFAULT CHARACTER SET latin1 */;
USE `scraper2`;

CREATE TABLE `outcode` (
  `id` int(11) NOT NULL,
  `completed` int(11) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `property` (
  `id` int(11) NOT NULL,
  `outcode_id` int(11) NOT NULL,
  `url` varchar(250) NOT NULL,
  `title` varchar(250) DEFAULT NULL,
  `description` text,
  `monthly_price` int(11) DEFAULT NULL,
  `weekly_price` int(11) DEFAULT NULL,
  `latitude` float DEFAULT NULL,
  `longitude` float DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `outcode_id` (`outcode_id`),
  CONSTRAINT `property_ibfk_1` FOREIGN KEY (`outcode_id`) REFERENCES `outcode` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `error` (
  `obj_type` varchar(250) DEFAULT NULL,
  `id` varchar(20) DEFAULT NULL,
  `text` text
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Running the Application

Execute the Python script 'Scraper.py'. Make sure that the other scripts in our Github repository are also available in the same location. Also make sure that you have installed the necessary packages as described above (bs4, requests, re, time, and MySQLdb).

You will be asked to answer the following questions about the list of outcodes you would like to scrape and the connection to your MySQL database:

```
Outcode IDs separated by spaces (default='1822'): 942 945 1273
MySQL host (default='127.0.0.1'):
MySQL user (default='root'):
MySQL password (default='root'):
MySQL database (default='scraper'):
```

Below, you can see an example of the output from the scraping process. This shows the outcode being scraped, which page of the search results of that outcode is being scraped, and then the ID of each property. In this example, you can see that a number of the properties had already been scraped and were skipped in the current execution.

```
===== Scraping outcode 942
=== Search results page 1
Already scraped property 59476661
Already scraped property 41895795
Property 54192922
Property 59423210
Property 41865603
```

For a simple example, we suggest using outcode NR11 which covers an area in rural Norfolk. At the time of this writing, RightMove shows only 17 properties in this outcode. The internal RightMove ID for this outcode is 1822.