

Text Mining: Web Scraper Project

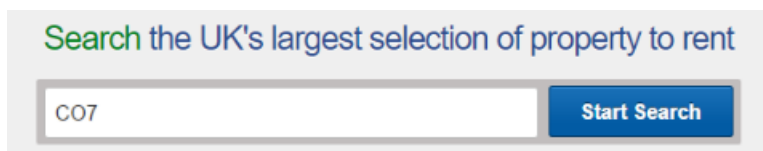
Roger Cusco, Matthew Sudmann-Day, Miquel Torrens

We chose to scrape the website ‘[RightMove](#)’ which advertises properties for sale and to let throughout the UK. We chose to limit ourselves to properties to let. 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.

You can find our code and this document in our Github repository: <https://github.com/m-sudmann-day/BGSE-text-mining/tree/master/scrapper>.

Navigation

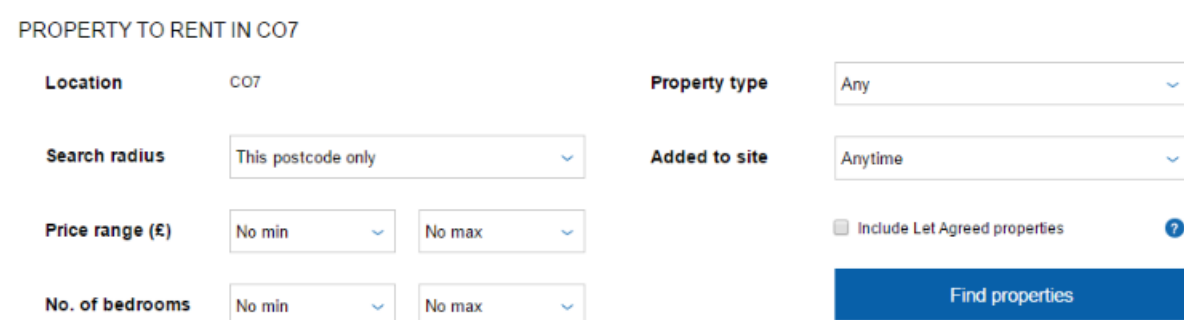
The web scraper mimics the actions of a user. The first thing the user must do is choose an “outcode”. In the UK, an outcode is the 3- or 4- digit prefix of a full postcode.



Search the UK's largest selection of property to rent

CO7 Start Search

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



PROPERTY TO RENT IN CO7

Location	CO7	Property type	Any
Search radius	This postcode only	Added to site	Anytime
Price range (£)	No min		<input type="checkbox"/> Include Let Agreed properties
No. of bedrooms	No min		

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.

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>

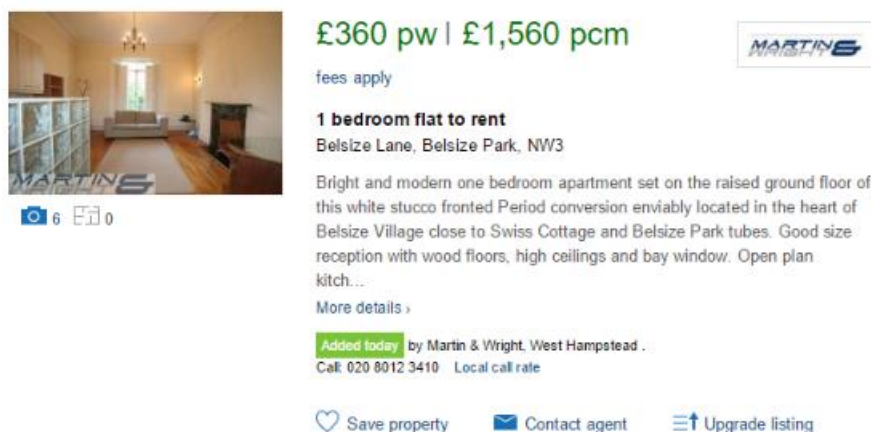
This is the format of URL that the scraper uses to extract pages of search results. Note that the outcode has been converted by the website from CO7 to 520. 520 is RightMove's internal ID for the outcode CO7, and the only ways to discover the mappings between outcodes and the internal IDs is manual discovery 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 use 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:



£360 pw | £1,560 pcm

fees apply

1 bedroom flat to rent

Belsize Lane, Belsize Park, NW3

Bright and modern one bedroom apartment set on the raised ground floor of this white stucco fronted Period conversion enviably located in the heart of Belsize Village close to Swiss Cottage and Belsize Park tubes. Good size reception with wood floors, high ceilings and bay window. Open plan kitch...

[More details](#)

Added today by Martin & Wright, West Hampstead

Call: 020 8012 3410 Local call rate

[Save property](#) [Contact agent](#) [Upgrade listing](#)

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

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 **£385 pw | £1,668 pcm**
Lower Ground, South Hill Park, Hampstead, NW3 fees apply

Metadata:

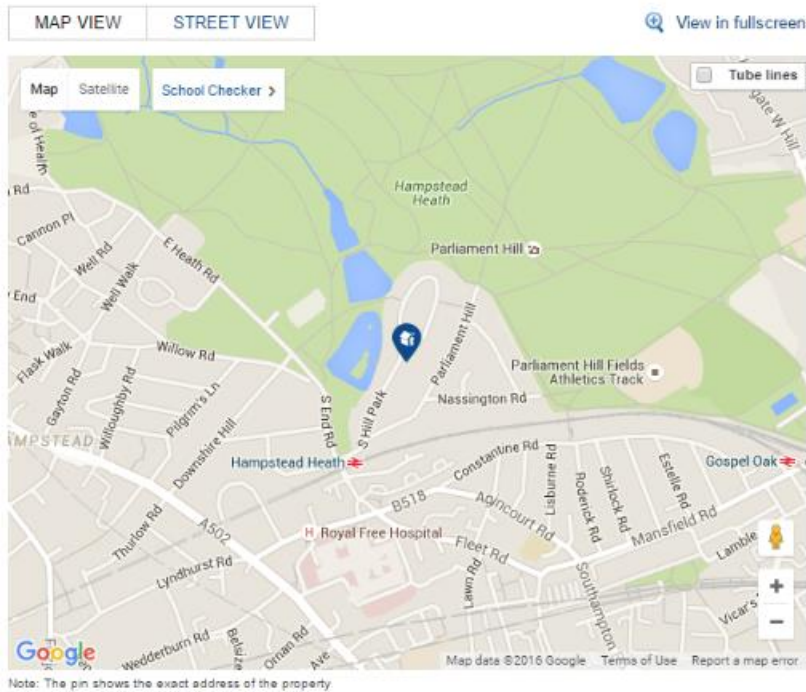
Letting information:

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 the search results are not empty:
    {
        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.

Storing Our Documents

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.

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 metadata 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. 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:

```
try:
    self.start()
    self.scrape()
    self.finish()
except Exception as ex:
    LogError(self.db, "Outcode", outcodeId, ex)
```

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;

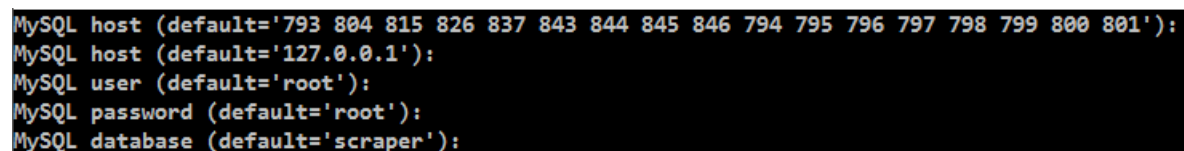
CREATE TABLE `property` (
  `id` int(11) NOT NULL,
  `outcode_id` int(11) NOT NULL,
  `url` varchar(250) NOT NULL,
  `title` varchar(250) DEFAULT NULL,
  `description` text,
  `monthly_price` int(11) DEFAULT NULL,
  `weekly_price` int(11) DEFAULT NULL,
  `latitude` float DEFAULT NULL,
  `longitude` float 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.

Answer the following questions about the list of outcodes you would like to scrape and the connection to your MySQL database:

A screenshot of a terminal window showing MySQL command line prompts. The text is as follows:

```
MySQL host (default='793 804 815 826 837 843 844 845 846 794 795 796 797 798 799 800 801'):
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 each property. In

this example, you can see that a number of the properties had already been scraped and were skipped this time.

```
===== Scraping outcode 793
=== Search results page 1
Already scraped property 48006964
Already scraped property 53692108
Already scraped property 54173659
Already scraped property 41844144
Already scraped property 41843841
Already scraped property 41826849
Already scraped property 41748489
Already scraped property 44997508
Already scraped property 54082849
Already scraped property 59387120
Already scraped property 41713656
Already scraped property 53786089
Already scraped property 59294882
Property 37468431
Property 49065919
Property 53935693
```

Restarting

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.