Part III Project
Dissertation Draft 1

Patrick Lewis
Queens' College

March 2016

This dissertation is submitted in partial fulfilment of the requirements for Part III Chemistry. It describes work carried out in the Department of Chemistry in the Michaelmas Term 2015 and the Lent Term 2016. Unless otherwise indicated, the research described is my own and not the product of collaboration.

Signed:

Date:

Acknowledgements

Acknowledgements will go here

Abstract

A Large Dataset of Chemistry Literature metadata was built up by automated scraping from freely available online sources. A UK Chemistry Department dataset of Chemical literature meta-data was built up by a similar method. Novel Natural Language Processing algorithms were used to develop powerful models to represent the Chemical Semantic space. These models were analysed and visualisation techniques were developed. The utility of the models was demonstrated by finding relationships between researchers at the University of Cambridge Chemistry Department.

Contents

List of Figures

Glossary

NLP blah.

ACS blah.

API blah.

Bag Of Words blah.

BagOfCitations blah.

CBOW blah.

CCD blah.

Cluster Map blah.

Communities blah.

Corpus blah.

Cosine Similarity blah.

Crawling blah.

Crossref blah.

Dendrogram blah.

Doc2Vec blah.

DOI blah.

EuclideanSimilarity blah.

Gensim blah.

Gephi blah.

Hadamard Division blah.

Hadamard Square Root blah.

HTML blah.

Hyperparameters blah.

IDF blah.

IP Address blah.

Lancaster blah.

Machine Learning blah.

Meta-data blah.

MongoDB blah.

Neural Net blah.

Paragraph Vectors blah.

PCA blah.

Perl blah.

PILA blah.

Porter blah.

Python blah.

Regex blah.

RSC blah.

SciFinder® blah.

Scraping blah.

Skipgram blah.

Snowball blah.

Stemmer blah.

Stop Words blah.

Taylor & Francis blah.

TF-IDF blah.

Training Dataset blah.

Training Epoch blah.

TSNE blah.

Unicode blah.

UPGMA blah.

UTF-8 blah.

Web Of Science ™ blah.

Word2Vec blah.

WordNet blah.

XML blah.

XPath blah.

Zipf's Law blah.

1. Introduction

1.1 Modern Scientific Publishing

The widespread adoption of the internet in the late 1990s and 2000s, brought fundamental changes to the academic publishing landscape. The information revolution allowed publishers‘ costs to fall, and there was a mood shift in the academic sphere away from subscription based models, towards giving open and free access to some or all of journal article contents. Simultaneously, learnèd institutions (such as university websites) began to post records of recent publications and other chemical information freely online. Publishers still protect the vast majority of journal article content and some metadata. Data is valuable and the insights within, powerful. As such, publishers are unwilling to grant free access to their data, preferring to perform in-house analysis. Article meta-data, such as authors, titles and abstracts may, however, be available, and it is this dataset which the project is focussed on.

1.2 Motivation

By collecting metadata on papers found on the internet, a large, representative dataset of chemical academic writing language can be built up. Machine Learning techniques can then be applied to find novel connections between articles, research communities, authors, institutions and fields. Machine Learning is a rapidly progressing field and data science can reveal key, non-obvious relationships to aid the scientific process. In an increasingly data-dense world, scientists require smarter tools to streamline research in order to be more productive. Several publishers provide services that perform large scale analysis and provide literature tools, such as SciFinder®and Web of Knowledge^{TM}. The techniques used and motivations behind the corporate bodies that own these services are not necessarily clear and thus there is much to be gained from independent, original analyses of the online publishing landscape.

1.3 Aims

The aims of the project are set out below:

- Collect large quantities of article meta-data from articles pertaining to chemistry as a general discipline
  - Identify websites that might contain useful chemical information
  - Write web-scraping programs that can scrape to identify and extract chemical information
  - Store information in human readable, computer readable, scalable and stable formats
- Develop novel machine learning techniques to enable meta-data to be interpreted in new ways
  - Sanitise input data effectively
  - Devise machine learning models to interpret article titles and abstracts to attempt extract their chemical meaning
  - Quantitatively represent an article's content using its collected meta-data
- Validate the model and provide evidence of their efficacy
  - Develop visualisation techniques for interpretation of algorithm output.
  - Analyse datasets using the developed model to demonstrate new and useful information
  - Provide usable code for future analyses to be performed with

This project is thus an informatics/data project, which split naturally into two sections. The first half of the project was concerned with acquiring data. This is covered in detail in §2. Programs were written in the python programming language, and two databases were created, one of UK Department Chemistry, and a very large database of unrestricted chemistry related material.

Once the databases were set up, focus was shifted to how to use the data to find valuable insights. §**??** and §**??** provide the background of the algorithms selected used and the process of applying them to create useful models.

Having built the models, it was now necessary to examine their outputs and develop methods to interpret results, which is covered in §**??**. Finally, when the models were shown to be performing successfully, they were used in an analytical setting; To examine the relationships between authors and research communities in the University of Cambridge Chemistry Department and eventually to recommend specific collaborations between staff that were predicted to be fruitful.

2. Data Acquisition

2.1 Background

2.1.1 HTML and Xpath

Internet webpages are written in a markup language called HTML, (HyperText Markup Language). When a webpage is accessed, the html code is sent to the user, and the browser processes and displays the webpage in a human readable format.

A program written to automatically interpret webpages to extract information, is known as a 'scraping' program. The program must process the raw HTML file and access the useful information on the page in an automated fashion. Information is arranged in an html document in a tree-like structure (figure 2.1). This example page would display in a browser as a table with 3 rows, each row containing 'Table Data A/B/C'. The method of tree traversal is by specifying a path through the document tree on the right, using an 'xpath'.

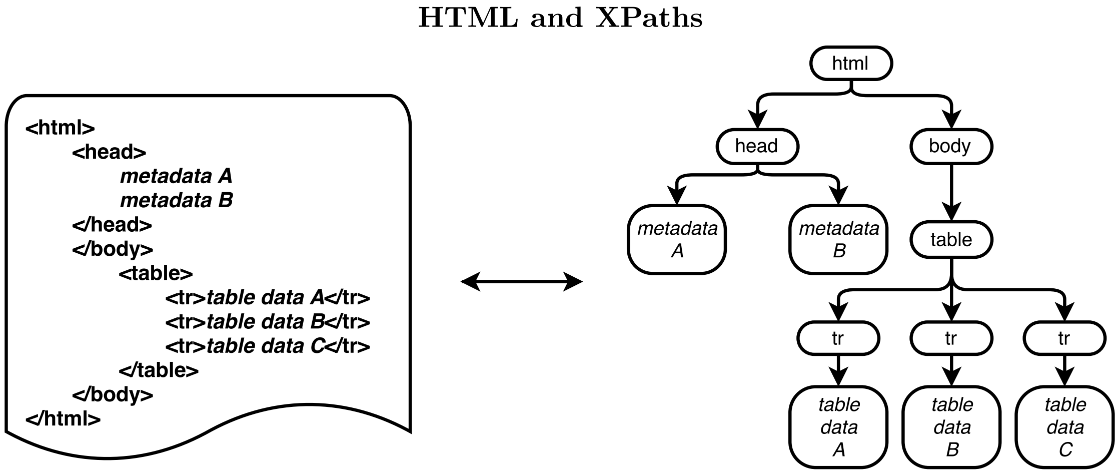

Figure 2.1: Tree representation of HTML code. The html code here displays a table with 3 rows. The page has two peices of metadata associated with it, stored in the 'head'.

Xpaths are just paths through the document tree to the desired information. In order the 'scrape' the data in the table, the following xpath could be used:

```
//html/body/tr/*
```

This presents an immediate problem, as scraping a millions of webpages requires millions of potentially different xpaths to be known. It is impractical to specify them manually, thus the challenge of large scale scraping is how to identify and collect useful data on webages without manually specifying many xpaths.

2.2 Automatic Xpath Generation

The initial approach was to to analyse the html tree to automatically recognise where useful tabulated or listed data was. The program started at the root of the tree and repeatedly followed the branch with the most 'repeating substructure'. The recursive algorithm is summarised below:

```
1. Count # of descendents of each child node
2. (a) Calculate the pairwise similarities between all child nodes
   (b) Consider two nodes similar if pairwise similarity is above a
       heuristic threshold
   (c) Calculate proportion of nodes that are considered similar
3. If proportion calculated in (c) is above a heuristic threshold,
   this node represents a store of information, and the xpath has
   been found.  Otherwise, move to child node with highest # of
   descendants, return to step (1)
```

The heuristic thresholds are adjustable parameters. The approach was successful for webpages with large numbers of records, formatted in repeating fashion, but performed poorly for smaller collections of data. As such it was not flexible enough for the task of scraping large for chemical data, and was not implemented in final solution.

2.3 Collection Strategy

As generating xpaths proved unsuitable, a new strategy was required. Chemical information is usually disseminated as journal articles, mostly accompanied by a DOI. By programmatically collecting DOIs, (§2.3.1) it was possible to build up a large database of chemical information (see §2.3.2)

2.3.1 DOIS : Document Object Identifiers

DOIs are computer-friendly labels for articles. DOIs are issued by a number of accredited bodies, with the vast majority of chemistry related articles issued by Crossref.[1] [**?**]. By pre-pending a DOI string with the url stub `http://dx.doi.org/`, the International DOI foundation (IDF) service redirects the request to the publisher's website to display the article the DOI refers to. The structure of a DOI is shown in Figure 2.2.

Anatomy of a DOI

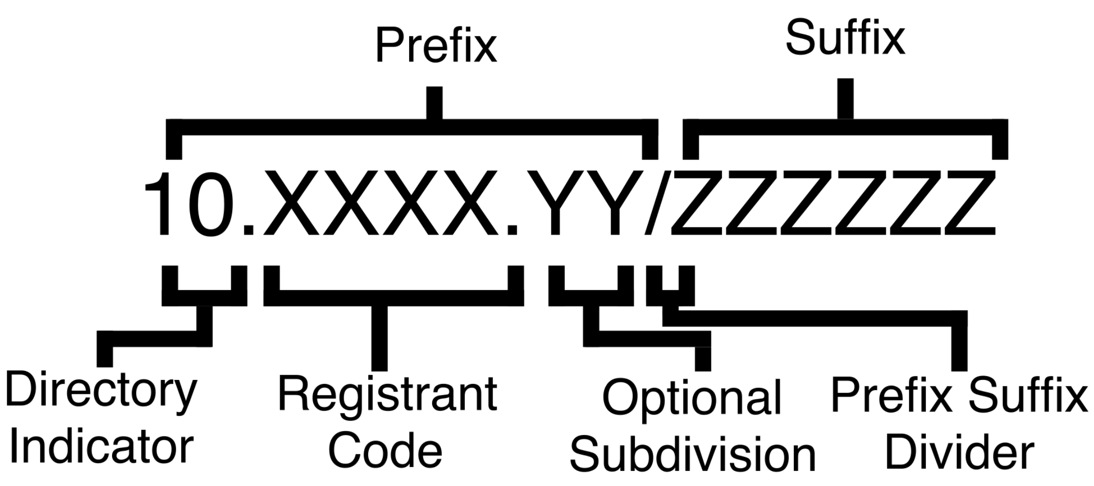

Figure 2.2: Doi structure. The structure consists of a numeric prefix (X and Y must be integers) and alphanumeric suffix (Z can be any Unicode encoded character)

DOIs consist of a prefix and suffix. The prefix is subdivided into the Directory Indicator (always integer 10) separated from the Registrant Code, assigned by the issuing body. Registrant codes are numeric and can be a minimum of 3 integers, with further otpional subdivisions separated by full stops. The suffix is provided by the registrant themselves. It can take any form as long as it is encodable by UTF-8.

It was possible to write a 'Regular Expression' pattern matcher (regex) to automatically recognise DOIs within a body of text, (See Figure 2.3) The flexibility of the registrant code specification means that DOIs cannot always be unambiguously identified in html documents.

[1]Crossref is a not-for-profit body comprised from Publishers International Linking Association (PILA), an association of many academic publishers

Pattern Matching Procedure for DOIs

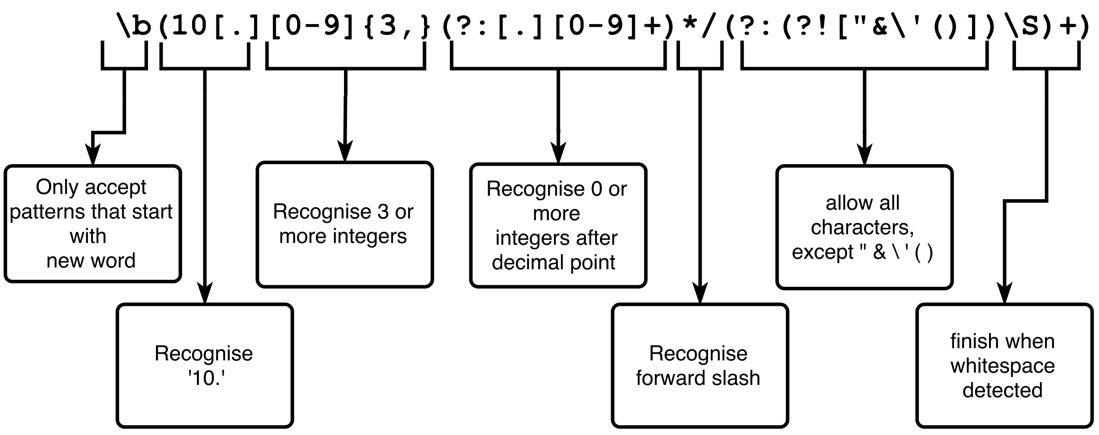
\b(10[.][0-9]{3,}(?:[.][0-9]+)*/(?:(?!["&\'()])\S)+)
Only accept patterns that start with new word
Recognise '10.'
Recognise 3 or more integers
Recognise 0 or more integers after decimal point
Recognise forward slash
allow all characters, except " & \ ' ( )
finish when whitespace detected

Figure 2.3: Perl Syntax Regex Code that can identify the vast majority of DOIs within free text)

Despite this, the regex was able to identify 90.4% of the dois on the Cambridge University Chemistry Department website `http://www.ch.cam.ac.uk/publications`.

2.3.2 Scraping Program

The Regex approach does not require xpaths in order to extract dois from a webpage. This facilitates large scale scraping from a large set of websites. Some meta-data associated with a DOI can be accessed using an online API exposed by Crossref. Further metadata can be accessed by following the `http://dx.doi.org/{DOI}` redirecting service by DOI ® .org. to visit publishers' websites to collect remaining metadata.

With this methodology in place, a scraping program was written in python to collect DOIs from a list of webpages and collect metadata in a 2 stage process. The Crossref API provides article titles, journals, authors, publisher and publication date meta-data, but not article abstracts. These had to be collected by visiting publisher webpages, and collecting with hand written xpaths. [2] The procedure is summarised in figure 2.4 below:

[2]Since there are comparatively few publisher websites, only 26 publisher xpaths were required for decent capture coverage.

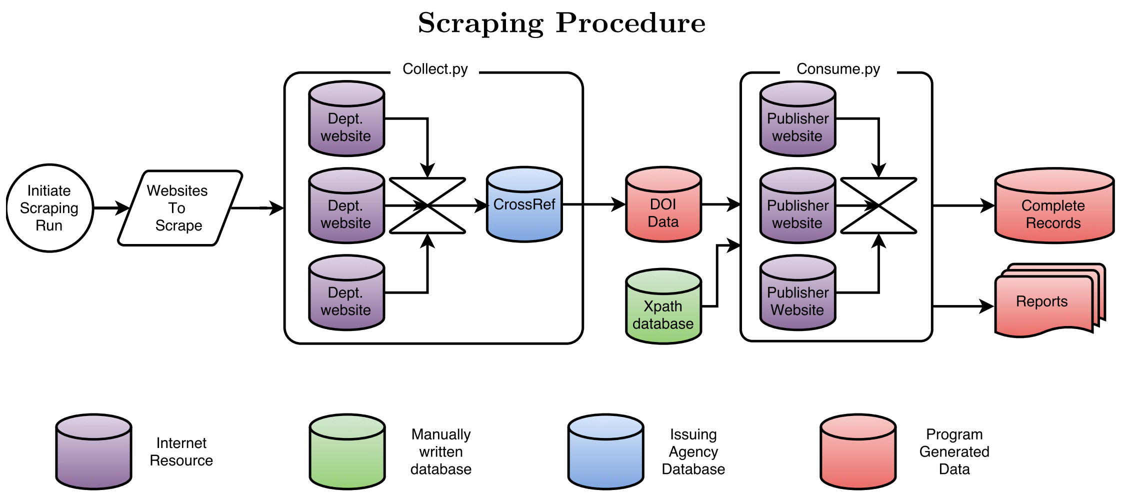

Figure 2.4: The data flow of the scraping program. Websites from an inputted list of websites are visited and dois are extracted in the process described in §2.3.1. The Crossref API service is then used to verify the extracted dois, and collects available meta-data. The program then accesses publisher webpages and collects abstracts. The program also produces explanation of capture failures and some general statistics

The programmatic steps depicted in 2.4 are:

1. `Request the webpage from the inputted list`
2. `Process the html and extract dois`
3. `Using the Crossref Online API, verify the extracted dois exist.`
4. `Crossref yields metadata:`
   - `Title`
   - `Journal`
   - `Publisher`
   - `Authors`
   - `Publication Date`
5. `for each doi, follow the doi using http://dx.doi.org/{doi}`
6. `use xpath to collect article abstracts.`

The program exports complete records as .json files, but also feeds directly to a MongoDB database. Once the program was written, the next priority was to obtain a list of webpages to scrape. This is described in §2.4.1 and §2.4.2

2.4 Collection Results

2.4.1 UK University Department scraping

The program was first used to collect the data from the UK. The Goodman group's website hosts a list of UK chemistry departments `http://www-jmg.ch.cam.ac.uk/data/c2k/uk.html`. The list was manually checked and some urls were changed to give a list of 68 departments[3]. The program was run with this list as an input, the results of which are detailed in table 2.1. Conversion losses were due to 4 components. 45 losses

Table 2.1: UK Scraping results

Process	# records	% of maximum yield
Dois collected	22442	100.0%
Dois found with metadata	22397	99.8%
Articles successfully resolved	16363	72.9%
Losses due to failed requests	2753	12.3%
Program errors	133	0.6%
Missing Publication Errors	3148	14.0%

for non-existant dois, 2753 losses to request errors (404 : not-found errors or permission problems), 133 to the program throwing internal errors and 3148 conversions were lost due missing publication xpaths. The 26 specified xpaths [4] were sufficient to convert 83.8% of successful requests. This was deemed acceptable, as most major publishers had been covered [5], and the missing publishers each covered a small number of articles it would take another 11 xpaths of the missing most popular publishers to increase the conversion rate from 83.8% to 90%. The efficiency is depicted in figure 2.5

[3]Details can be found in the appendix
[4]Corresponding to 37 publishers
[5]see appendix for list of covered publishers

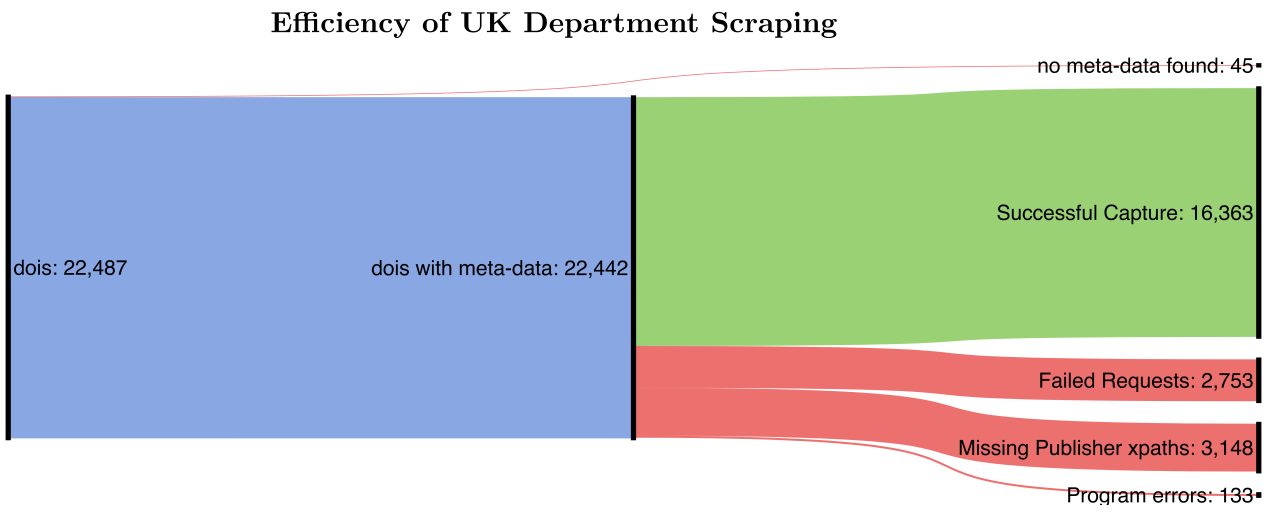

Figure 2.5: The loss processes are coloured red, successfully captured full records in green, and the maximum possible yield in blue.

Interestingly, 9467 out of 16363 successful collections were sourced from `http://www.ch.cam.ac.uk`. This could be because the department at Cambridge has an extensive website and also hosts the majority of its information under its own domain name, whereas other departments‘ data are hosted on central university domains. The scraping program was confined to scrape only webpages belonging directly to chemistry department domains, not the university website as a whole. As a result, it is worth baring in mind that the Cambridge chemistry department may be overrepresented in the UK chemistry data set.

2.4.2 Very Large Scale Scraping

The UK scrape was a respectable start, but much more data would be required to train a machine learning model successfully. One approach would have been expand the scrape to world-wide chemistry departments, and other learnèd bodies. However, Crossref also exposes a search service that can be used to query its vast internal database. The program was then set up to query the Crossref service for search terms ‘Chemistry’, ‘Chemical’, ‘Molecule’ and Molecular’ for journal articles and journal titles. This suggested possible yields in the millions of articles.

The program was instructed to scrape the search results pages of these queries. Because the scraping job was so large, the program was set up to ‘pause’ before the publisher abstract collection stage.The results of up scrape up to this point were examined before setting off the second stage to collect abstracts.

At the intermediate point, the program had collected 1,267,495 records, which was deemed successful, and would provide enough data to train a powerful machine learning algorithm.

The records were then inspected and publisher distributions were considered. Some of this analysis is presented in §2.4.4. After careful considerations of request server loads and predicting capture probabilities, the second half of the scraping routine was set off to run for 3 days.

2.4.3 Problems with ACS and Taylor & Francis

Some publishers automatically track request volumes sent to their site as they wish to discourage automatic scraping of their data. Scraping their websites is not illegal, and the data collected was freely available, not behind pay-walls, or protected. However, during the scraping run, a bug in the randomisation of request frequencies resulted in detection by the ACS [6] and Taylor & Francis. Both publishers responded by banning the IP address of the computer running the program. The department librarians were able to restore access, and it was agreed that no further large scale scraping runs would be performed.

Taylor & Francis banned the IP address after it detected over 100 requests were made within 5 minutes. This corresponds to a request every 3 seconds. This is modest server load compared to other publishers, and was not foreseen to cause problems.

The ACS banning occurred because of a bug in the randomisation of requests. The program was instructed to take a doi from a random publisher every time it made a request, rather than just a random doi. Since the largest publisher was ACS, the program eventually exhausted dois from the other publishers, until there were only ACS dois to 'randomly' draw requests from. This meant the request frequency to the ACS server went up dramatically. This uptick broke the threshold of allowed requests at the ACS server which then banned the IP (approximately 10 requests a second).

[6] American Chemical Society

ACS BANNING

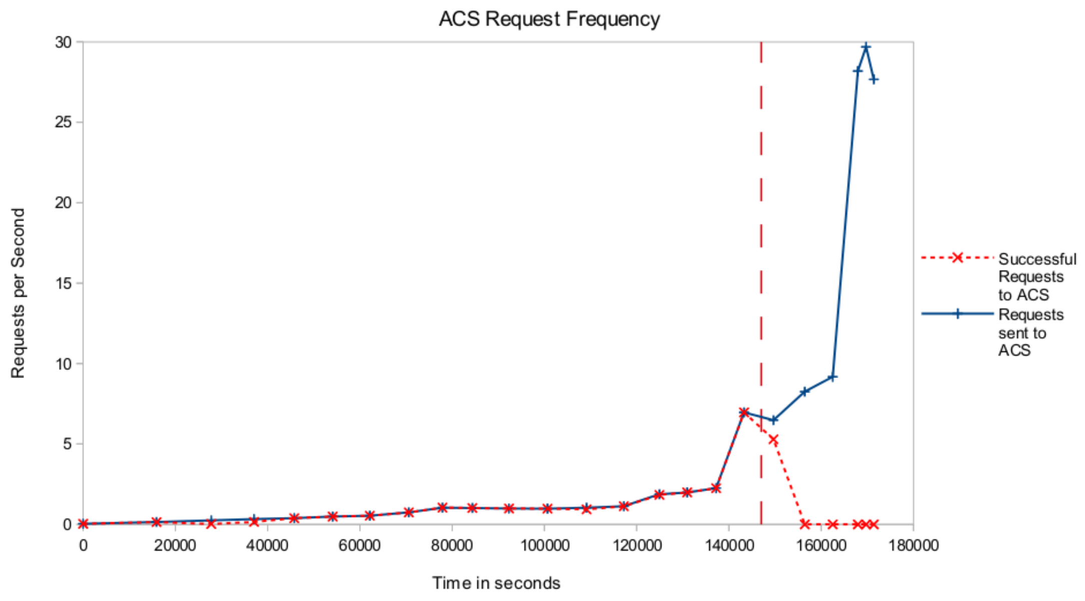

Figure 2.6: The request frequency is plotted in blue, the received pages frequency in red. The vertical dashed line shows where the server detected the scape and banned the IP.

The program was capable of making a total number of approximately 30 requests per seconds. As canbe seen in figure 2.6, the program began to run out of requests to other publishers after approximately 140,000 seconds, resulting in an increase in the proportion of total requests per second to ACS. The ban occurred after approximately 150,000 seconds, after which there were no more responses received.

2.4.4 Analysis of Collected data

The yield of the large scale scraping run was cut significantly by the ACS banning event. A summary is tabulated in 2.2, and shown graphically in figure 2.7.

Table 2.2: Large Scale Scraping Results

Process	# records	% of maximum yield
DOIS collected in stage 1	1267495	100.0%
Predicted maximum capture	1071506	84.5%
Predicted Capture without ACS	581797	45.9%
Articles successfully captured	714370	56.4%
Losses to failed requests (excluding ACS)	53743	4.2%
Losses to ACS banning	303393	23.9%
Missing Publications & Program Errors	195989	15.5%

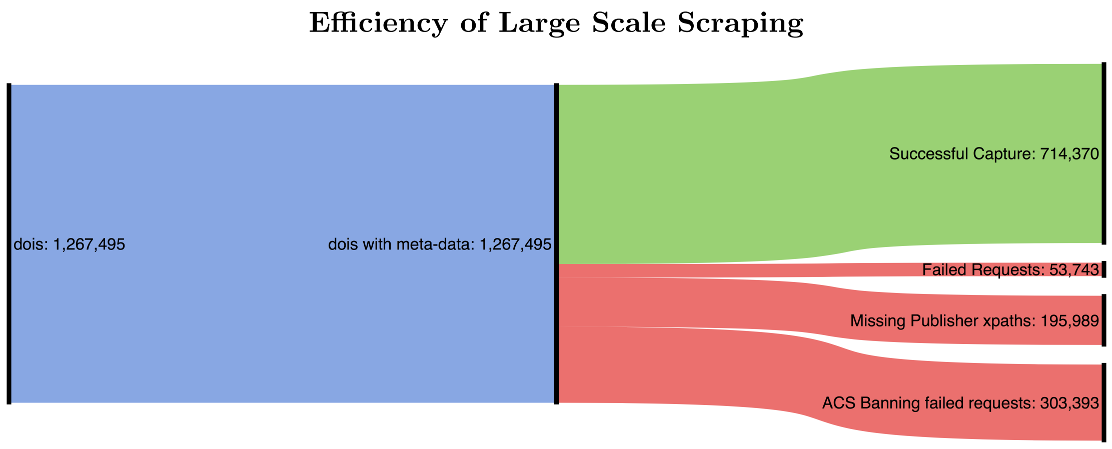

Figure 2.7: The loss processes are coloured red, successfully captured full records in green, and the maximum possible yield in blue.

The overall efficiency of the process is 56.4%, but excluding the acs records lost in the ban, the program's efficiency jumps to 74.0%, similar to the efficiency of the UK scraping run (§2.4.1).

The successfully captured 714,370 records were inspected and merged with the UK results. Records were rejected with short titles or abstracts (likely to be addenda, informal articles, retractions etc.) Records were also removed if the majority of the title and abstract were not writen in ascii characters [7] (removing majority Japanese and Chinese script). This was done to provide better quality data for training the algorithm described in §**??**. This filtering resulted in a final training database of 464712 articles. This dataset is henceforth referred to as *the training dataset*. The entire database formation process is shown in figure 2.8.

[7]ascii is an encoding for English characters a-z, A-Z, some punctuation and 1-9.

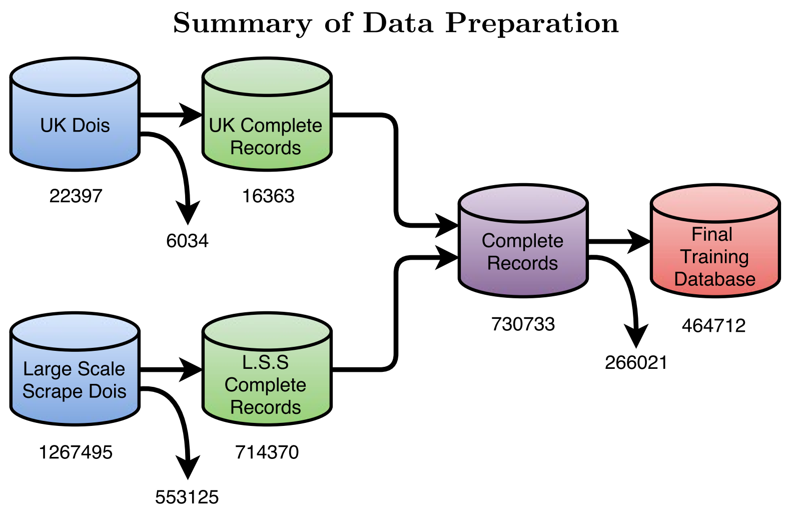

Figure 2.8: Blue databases represent data with dois and metadata. Green databases represent meta-data, dois and abstracts. The purple database is the combined complete records, and the red database is the data deemed suitable for the training algorithm. database sizes and losses are annotated.

It was instructive to examine these databases and derive some simple statistical results. The following section briefly explores some of these.

Observations

The publisher 'market share' can be approximated from examining the database[8].

[8]Data was taken from 'Large Scale Scrape Dois' database, shown in figure 2.8

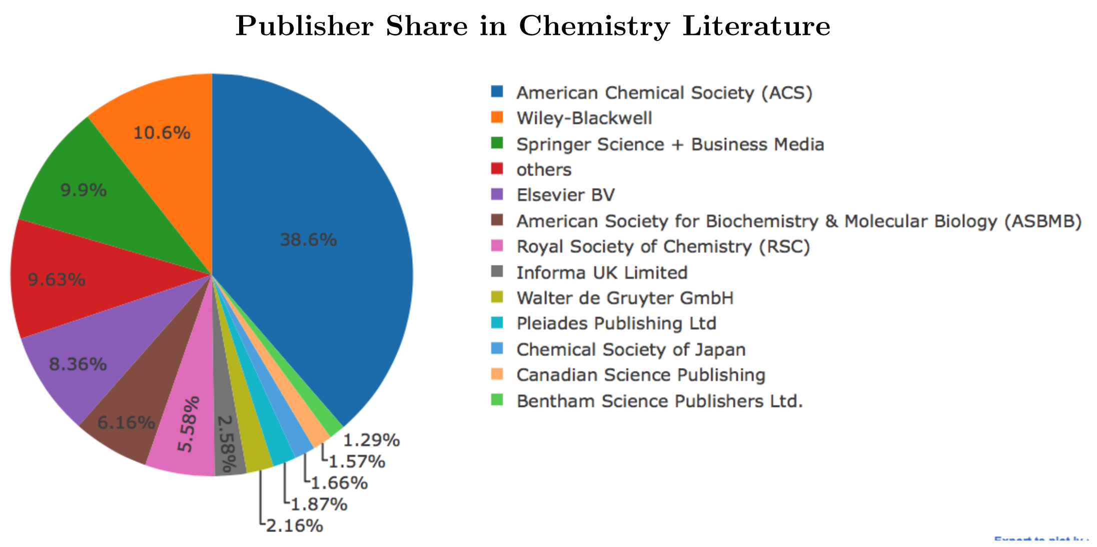

Figure 2.9: Articles grouped by publisher in the Large Scale Scrape doi database (marked 'Large Scale Scrape Dois' in figure 2.8). Only the top 12 publishers are shown.

As shown in shown in figure 2.9, it can be seen that 90% of all the chemistry literature collected was published by just 12 publishers, the majority from ACS, Wiley-Blackwell, Springer and Elsevier BV. Looking at the UK scraping DOI dataset (Figure 2.10), the same large publishers are represented, but the Royal Society of Chemistry has a much larger share. This is to be expected, as the RSC is a UK based body. In the UK, there is a more even distribution between the large publishers.

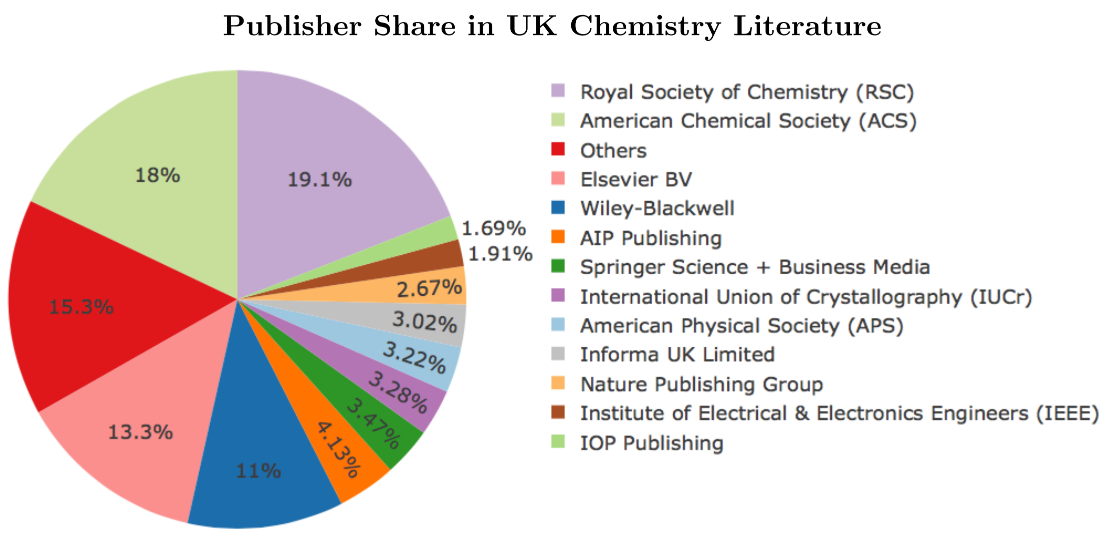

Figure 2.10: Articles grouped by publisher in the UK Doi database published by by each publisher. Only the top 12 publishers are shown.

The corpus of combined titles and abstracts of the complete training database was then examined. The word frequencies across all the data were found to be approximately Zipfian, with a gradient of -1.11[9] See figure 2.11

[9]A Zipfian distribution is a subset of the Pareto distribution, stating that the frequency of a word is proportional to its ranking in the word frequencies table. Ideally, the gradient of a log(frequency) vs log(rank) should be -1.0 [**?**]

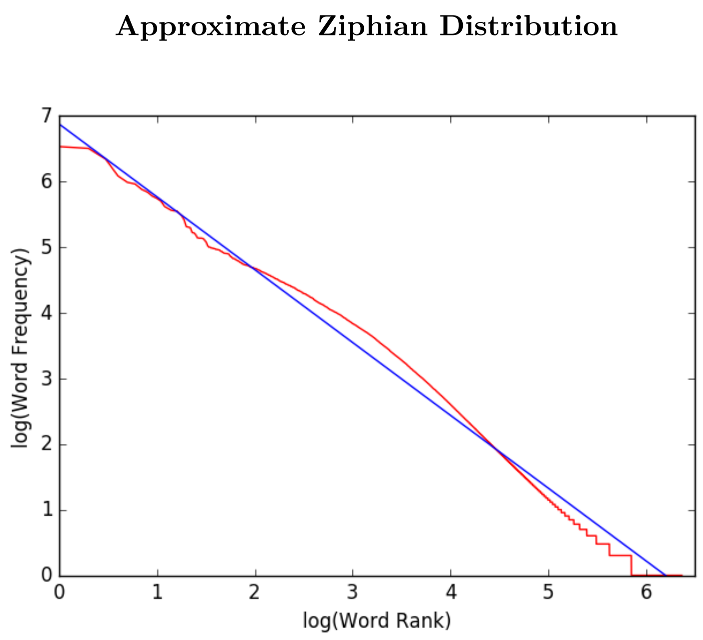

Figure 2.11: The log Frequency of words vs the log of their position in the rank in the word frequency table in blue. Best fit line in red, gradient = -1.11, intercept 6.3.

A summary of the corpus statistics are shown below:

Table 2.3: Titles and Abstracts in Training Database

Total Word Count	61,296,410
Total Unique Words	2,326,725
Total Document Count	464,712
Mode Words per Title	11
Mean Words per Title	12.2
Mode Words per Abstract	156
Mean Words per Abstract	119.7
Mode Sentences per Abstract	4
Mean Sentences per Abstract	5.4