Part III Project
Dissertation Draft 1

Patrick Lewis
Queens' College

March 2016

Abstract

Abstract goes here

Contents

List of Figures

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, 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, as publication meta-data they is valuable and powerful. There is a well known saying in Data Science from Tim OReilly, The Guy with The Most Data Wins CITE. As such, publishers are unwilling to grant free access their data for analysis by the public, 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. Several publishers provide services that perform large scale analysis and search, such as SciFinder®and Web of Knowledge^{TM}. The techniques used and motivations behind the corporate bodies owning 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 website 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 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
  - Devise tests of algorithm to ascertain whether it performs as hypothesised
- Analyse datasets using the developed model to demonstrate new and useful information
- Provide usable code for future analyses to be performed with

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 over the internet to the user, and the browser e.g. Firefox, interprets it and displays the webpage in a human readable format.

A program written to automatically interpret webpages and 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, see 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'.

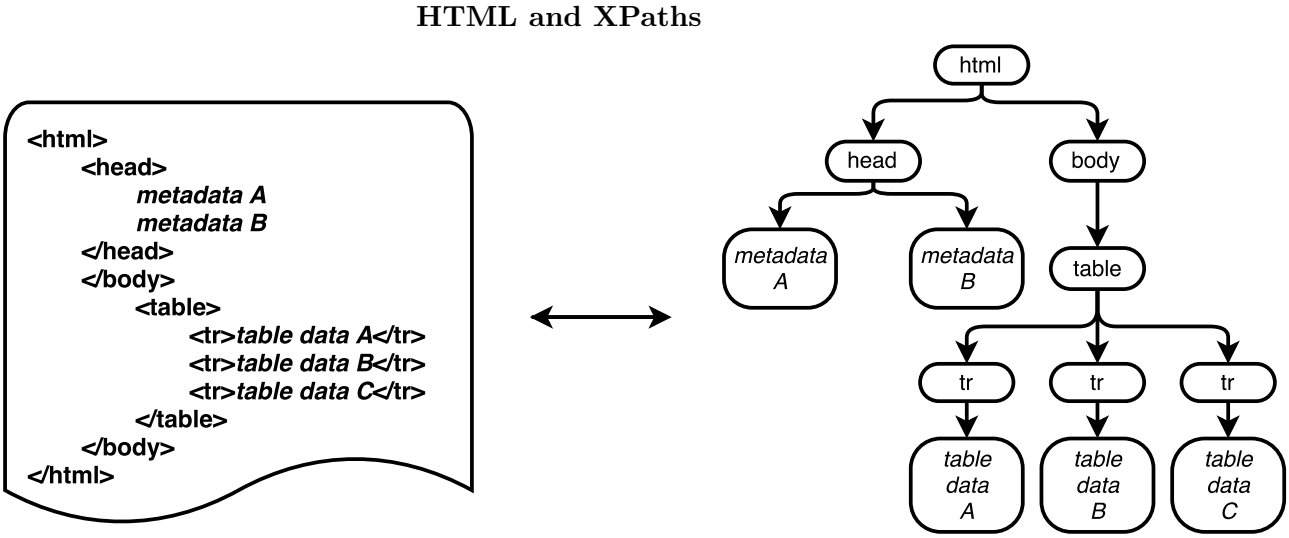

Figure 2.1: Tree representation of HTML code. The html code here displays a table with 3 rows. The page has two peices of metadata associated with it, stored in the 'head'.

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

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

In order to scrape information on a page, the xpath must be known. This presents an immediate problem, as scraping a millions of webpages requires millions of potentially different xpaths to be known. It is clearly impractical to specify them manually. Two possible solutions were attempted. The challenge of large scale scraping is how to identify useful data on the page and collect it, without manually specifying very many xpaths.

2.2 Automatic Xpath Generation

The initial approach was to attempt 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 'repeated structure'. 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, laid out in repeating fashion, but performs poorly for smaller tables or lists 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

The goal set was to build a database of chemical information freely available on the internet. This section describes how this task was addressed. As generating xpaths proved unsuitable, a new strategy was required. Chemical information is often disseminated as published articles. Modern papers are accompanied by a DOI. By programmatically

collecting DOIs, (see section 2.3.1) it is possible to build up a large database of chemical information (see section **??**)

2.3.1 Document Object Identifiers

DOIs (document object identifiers) are digital labels for journal 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 will redirect 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.

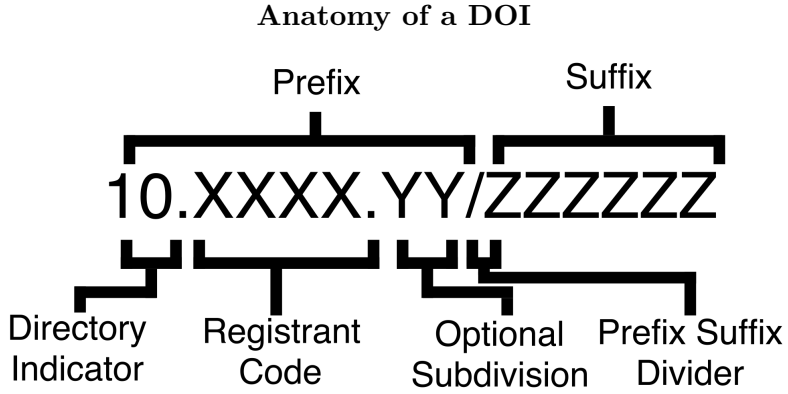

Figure 2.2: Doi structure. The structure consists of a numeric prefix (X and Y must be integers) and alphanumeric suffix (Z can be any Unicode encoded character)

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. Registrant codes can have further 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 due to this defined structure.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

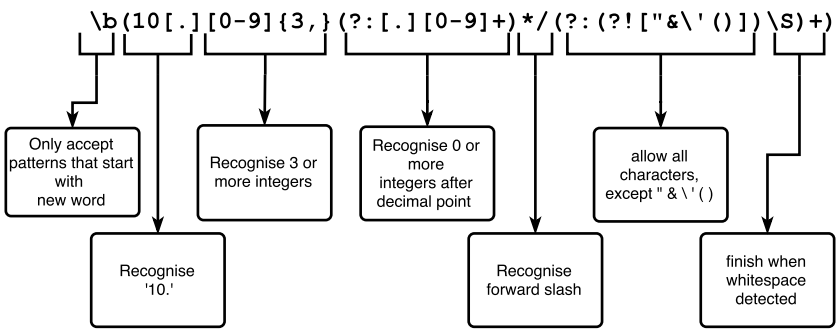

Figure 2.3: Perl Syntax Code that can identify the vast majority of DOIs within free text)

Despite this, the regex is sufficient 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 any xpath in order to extract dois from a webpage. This facilitates large scale scraping from a large set of websites. The meta-data associated with a DOI can be accessed using an online API exposed by Crossref.Further meta-data can be accessed by following the `http://dx.doi.org/{DOI}` redirecting service by DOI®.org. to visit publishers's websites to collect any remaining metadata.

The scaping program was thus 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 not a great many different publisher websites, only 26 publisher xpaths were required for decent capture coverage.

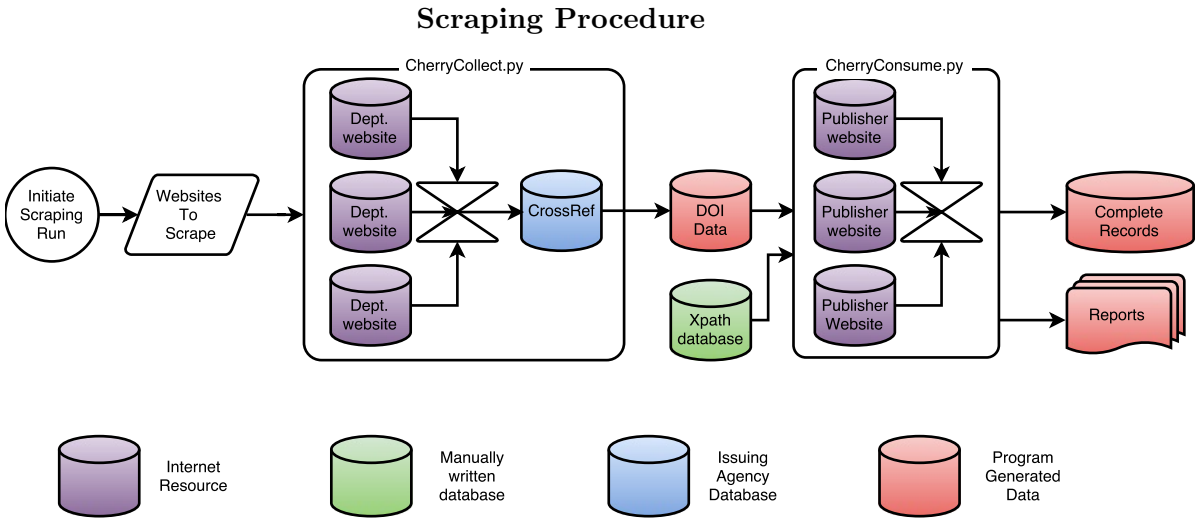

Figure 2.4: The data flow of the scraping program. An inputted list of websites to scrape are visited and dois are extracted in the process described in 2.3.1. The Crossref API service is then used to verify the extracted dois, and collects available meta-data. The program then accesses publisher webpages and collects the 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 htlm 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 is also able to feed directly to a MongoDB database instance. Obtaining an input list of websites to scrape was the next area of focus which is described in sections 2.4.1 and **??**

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 for efficiency of landing targets to give a list of 68 departments[3]. The program was set to collect dois on all the pages it could find at these websites and collect metadata in the described in section2.3.2. This resulted in a collection of: Conversion losses were due to 4 compo-

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%

nents. 45 losses were due to non-existant dois. 2753 losses were due to request errors such as could be due to 404 : not-found errors or permission problems. 133 conversion losses were due to the program throwing internal errors. 3148 conversions were lost due missing publication xpaths. The 26 specified xpaths were sufficient to convert 83.8% of successful requests. This was deemed acceptable, as most major publishers had been covered [4], 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 2.5

[3]Details can be found in the appendix

[4]see appendix for list of covered publishers

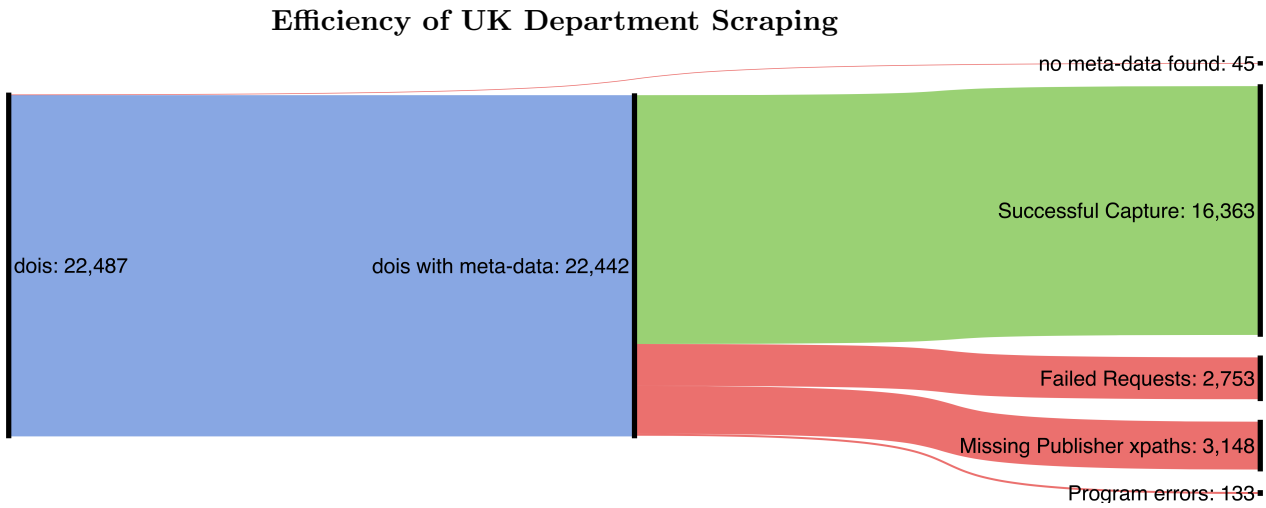

Figure 2.5: The loss processes are coloured red, successfully captured full records in green, and the maximum possible yield in blue.

Interestingly, 9467 out of 16363 successful collections were sourced from `http://www.ch.cam.ac.uk`. This could be because the Cambridge Chemistry department has its own website domain name where most of its data is hosted, whereas other departments' data is hosted on central university domain names. The scraping program was confined to scrape only webpages belonging directly to chemistry department websites, 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 collection procedure above was successful. However, in order for the machine learning analysis on the collected meta-data to be successful, much more training data was required. To this end, it was necessary to collect many more records. 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 it's 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 thus set up to scrape the search results pages of these queries. Because the scraping job was so large, the program was instructed to run the scrape in two distinct sessions. Firstly, it was to collect DOIs and get easily avaiable metadata. The results of this scrape were to be examined before setting off the second stage, collecting abstracts from publisher web pages.

After the doi and-meta data scraping run, the program had collected 1,267,495 records,

which was deemed very 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 and Francis

Some publishers automatically track number of requests 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 paywalls, or protected. However, during the scraping run, a bug in the randomisation of request frequencies resulted in the scraping being detected by publishers ACS (American Chemical Society) and Taylor and 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 and 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 predicted to cause problems.

The ACS banning occurred because of a nuanced bug in the randomisation of requests. The program was instructed to take a random publisher's doi per request. However, since the largest publisher of chemistry articles was ACS, the program eventually the other publishers papers, until it had only ACS papers 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).

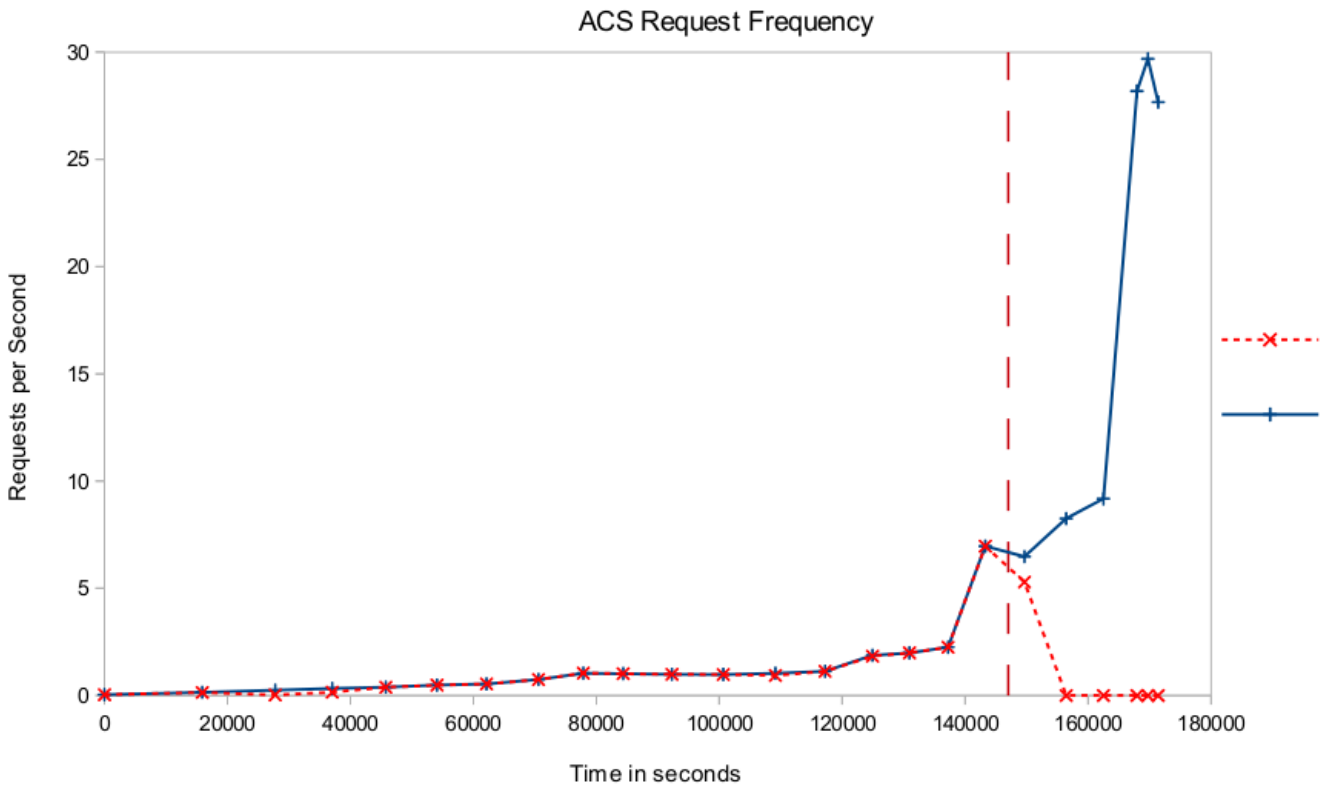

Figure 2.6: The request frequency is plotted in blue, the received pages frequency in red. The vertical dashed line shows where the server detected the scape and banned the IP.

The program was capable of making a total number of approximately 30 requests per seconds.As can be 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 going to ACS. The banning occurred after approximately 150,000 results, after which there were no more responses to requests.

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 tablulated in **??** The overall efficiency of the process is 56.4%. This is mainly down to ACS ban reducing the number of successfully collected articles. Excluding the ACS lost records, the program's efficiency jumps to 74.0%, similar to the efficiency of the UK scraping run (section 2.4.1). The efficiency is shown graphically in 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%

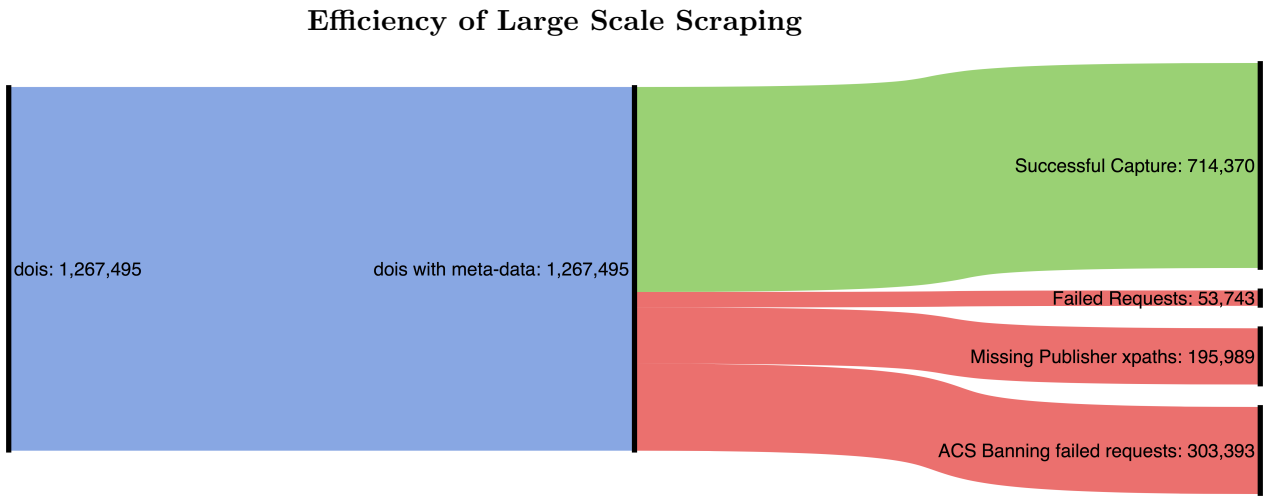

Figure 2.7: The loss processes are coloured red, successfully captured full records in green, and the maximum possible yield in blue.

The successfully captured 714,370 records were then inspected and merged with the UK results. Records were then rejected with short titles or short abstracts (likely to be addenda, informal articles, retractions etc.) Records were also removed if the majority of the title and abstract were not written in acscii characters [5] (removing majority Japanese and Chinese script). This was done to provide better quality data for training the algorithm described in chapter 4. 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.

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

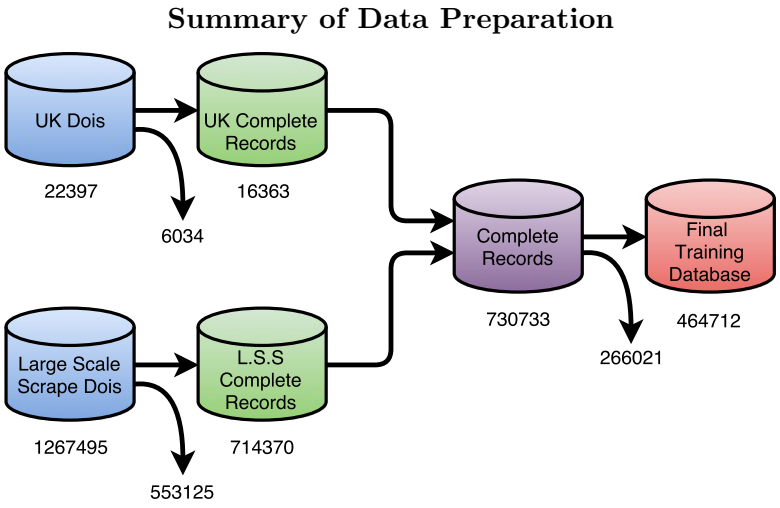

Figure 2.8: Blue databases represent data with dois and metadata. Green databases represent meta-data, dois and abstracts. The purple database is the combined complete records, and the red database is the data deemed suitable for the training algorithm. database sizes and losses are annotated.

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 Large Scale Scape Doi database, shown in 2.9.

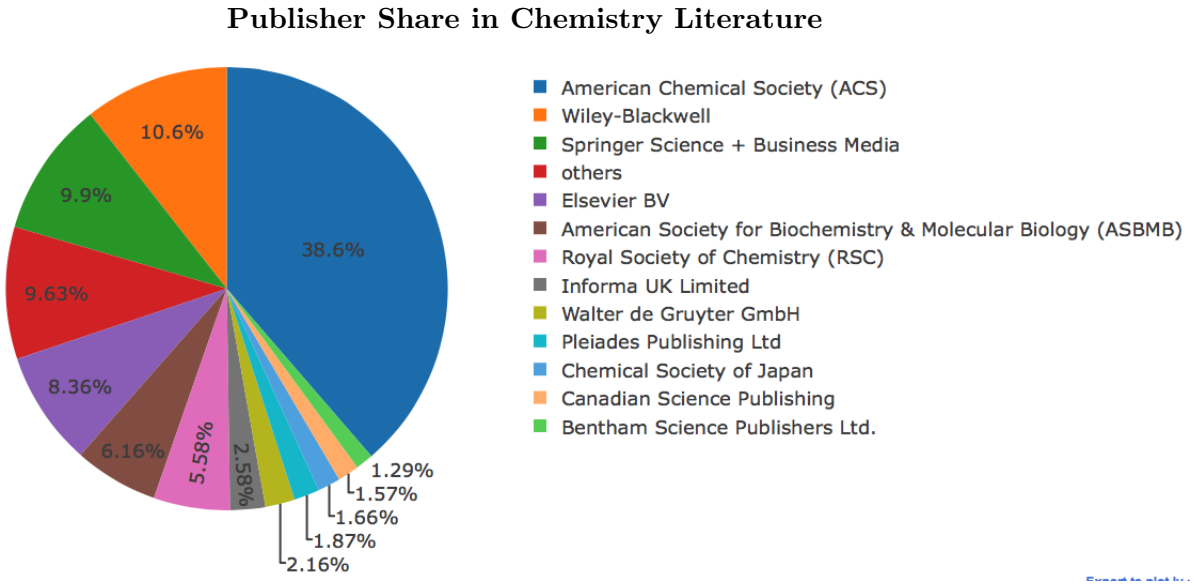

Figure 2.9: Articles grouped by publisher in the Large Scale Scrape doi database. Only the top 12 publishers are shown.

It can be seen that 90% of all chemistry literature collected was published by 12 publishers. The majority of these were from ACS, Wiley-Blackwell, Springer and Elsevier BV. Looking at the UK scraping DOI dataset (Figure **??**), 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, the distribution of publications is more even between the large publishers.

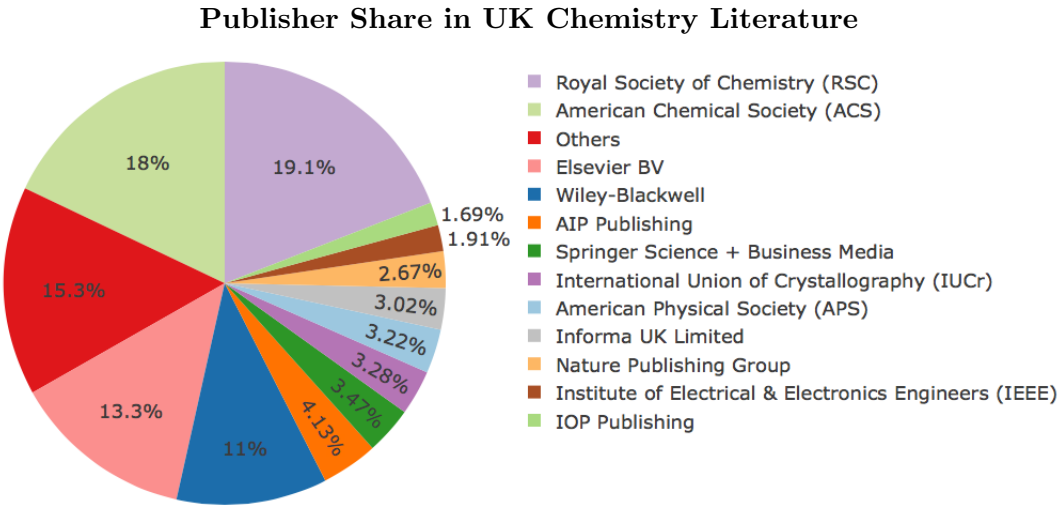

Figure 2.10: Articles grouped by publisher in the UK Doi database published by by each publisher. Only the top 12 publishers are shown.

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[6] See figure 2.11

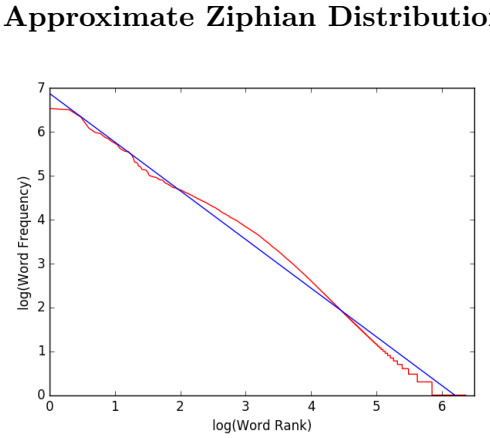

Figure 2.11: The log Frequency of words vs the log of their position in the rank in the word frequency table in blue. Best fit line in Red, gradient = -1.11, intercept 6.3.

A summary of the corpus statistics are shown below:

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

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

3. Techniques for Language Processing

3.1 Background

Natural Language Processing (henceforth NLP) is the application of computer science to study, model and understand human languages using computers. Machine learning, a class of algorithms for fitting and predicting patterns in data, is a powerful technique for Natural Language processing. NLP of chemistry journal articles enables subtle patterns in the to be detected. This section explores approaches to representing journal articles in a quantitative manner.

3.2 Bag of Words

A simple approach to representing a document is a *bag of words* model. The document is split into component words in an unordered set. The model first computes the number of distinct words that occur in a corpus of documents, N. It then assigns a each document in the corpus an N dimensional vector $\mathbf{v}$. If the document contains word i 2 times, then v_i is set to 2. A simple example is given below:

Document A: `A good yield was obtained for a nucleophile`

Document B: `The nucleophile is a good donor`

Table 3.1: Bag of words

Vocabulary	$\mathbf{v}_A$	$\mathbf{v}_B$
A	2	1
Good	1	1
Nucleophile	1	1
Yield	1	0
For	1	0
Is	0	1
The	1	0
Donor	0	1
Was	1	0

Table 3.2 shows the vector representations for Documents A and B. The higher the scalar product of normalised $\mathbf{V}_A \cdot \mathbf{V}_A$, the more similar the documents are predicted to be. This model is used extensively in industry.

3.3 Bag of Citations

The *bag of citations* model is also widely used for analysis for academic papers. The principle is the same as the bag of words model, but vector components are the presence or absence of citations.

3.4 Word2Vec

The *Bag of Words* model treats words as atomic units. This is beneficial for robust and fast computations. However, words can have degrees of similarity to each other, and these relationships are not captured by bag of words models **?**. Distributed representations of words have been used to address this for some time **?**.

A recent successful approach has been the Word2Vec algorithm Mikolov et al. [2013c] Mikolov et al. [2013a]. The Word2Vec algorithm attempts to use a neural net to represent words as vectors in a continuous space rather than a discrete one. Vectors for words with similar meanings will point in similar directions in this 'Semantic space'.The Word2Vec algorithm is fed a corpus of language sentence by sentence. The words within the sentences have a semantic relationship, which the algorithm tries to infer.

This is achieved with two architectures, the Continuous Bag of Words (henceforth CBOW) and skip-gram models. The CBOW architecture uses a neural net type learning algorithm to predict a word's vector by summing the vectors of the words that appear near it in a training sentence and comparing this to the model's current vector for the word. The skip-gram algorithm compares the predicted and current vectors of the words surrounding the current training word. By training with many input sentences, prediction vectors are gradually improved.

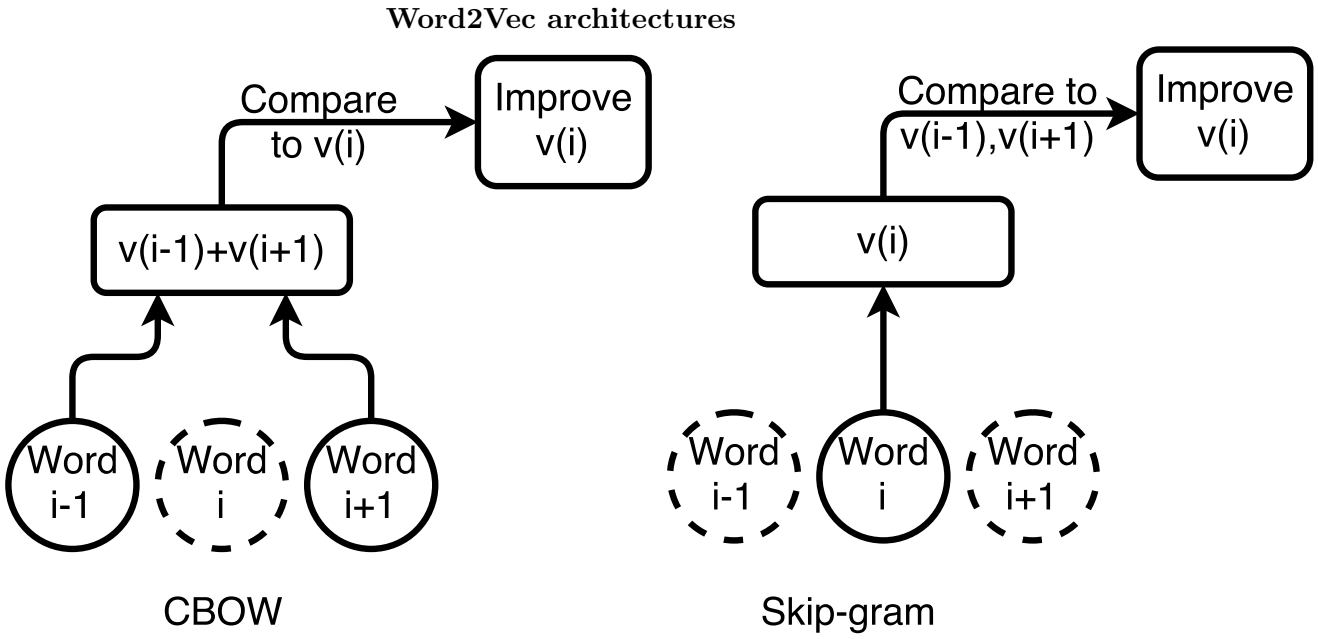

Figure 3.1: The training architectures of the Word2Vec training algorithm. Word vectors are denoted $v(i)$ for word i. In CBOW word i is predicted by the vector found by summing vectors surrounding i, and $v(i)$ is adjusted to be closer to this prediction. In skip-gram, word i's vector is pairwise compared to its context words, here i-1 and i+1 as a basis to improve $v(i)$.)

The training process is shown in figure 3.1. CBOW uses a fixed window of words surround current training word. The order of words within the window does not matter, but because the window 'slides' along as the algorithm considers words i+1, i+2... word order is represented in the model . In Skip-gram, a random number of nearby words are used for the prediction vectors for word i.

The model has added sophistications built in to reduce the importance of very commonly occurring words, and to identify phrases. The word vectors that are produced encapsulate both semantic and syntactic meanings and can be manipulated to represent concepts and relationships.

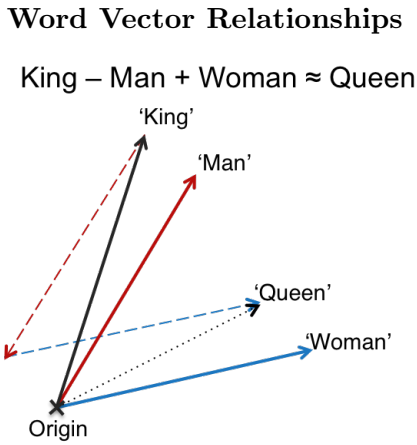

Figure 3.2: Schematic Representation of how concepts can be represented in word vector space. Word2Vec is able to replicate this behaviour. The vector found by vec(King)-vec(Man)+vec(Woman) is approximately equal to vec(Queen). The model has been tested on thousands of similar examplesMikolov et al. [2013a]Mikolov et al. [2013b].

Word2Vec models are able to represent concepts by vector algebraic operations on their word representations. Figure **??** shows one famous example a Word2Vec model trained on the 'Google News' text corpus was able to identify.

3.5 Doc2Vec

The Doc2Vec algorithm Řehůřek and Sojka [2010](implementation of Paragraph Vectors Le and Mikolov [2014]) allows the Word2vec process to directly learn vectors that represent paragraphs. The CBOW model is adapted so that, in addition to word vectors, each document is associated with its own vector that contributes to the vector sum predictions in training. The result is that documents can be represented in their own semantic space.

The nature of the collected metadata detailed in section **??** lends itself naturally to the Word2Vec and Doc2Vec algorithms, as it is a large store of natural language,rich with encoded information. The focus of the machine learning analysis phase of the project was directed at applying the Word2Vec and Doc2Vec algorithms to the dataset to try and automatically learn and classify chemical semantic concepts. These investigations are detailed in the following sections.

4. Algorithm Development

4.1 Premise

The aim of the machine learning phase was to apply the Word2Vec and Doc2Vec algorithms to the training dataset described in 2. An article was considered to be represented by a document consisting of its title and abstract. The aim was to represent these documents as vectors in semantic space, so that advanced analyses could be performed.

4.2 Data Sanitisation

The documents (titles + abstracts) in the training dataset required preprocessing before they effective used in training. The training process requires inputs to be as clean as possible in order to get good results (encapsulated by the well known computer science idiom 'Garbage in, Garbage out').

The first step was to cast all words to lower case, so that the algorithm did not produce different vectors for 'Molecule' and 'molecule'.

The raw documents also frequently contained artefacts from the source webpages, such as unwanted white space, vestigial html tags, 'newline' characters and carriage returns. The algorithm training a word vector for a '\n' is clearly undesired behaviour, so these special symbols were all removed and whitespace normalised.

It was also observed that, as unicode text scraped from a wide variety of sources, there was varied and reduntant punctuation. Punctuation would be treated as separate words by the algorithm, so had to be carefully removed. This was not straight forward as unicode has very wide variety of different punctuations. For example, unicode encodes 24 different types of hyphen. Table 4.2 shows the punctuation that was filtered out of the documents. Large sections of unicode script (Non-western language) was also removed as the algorithm works best on a smaller vocabulary.

Filtered Punctation

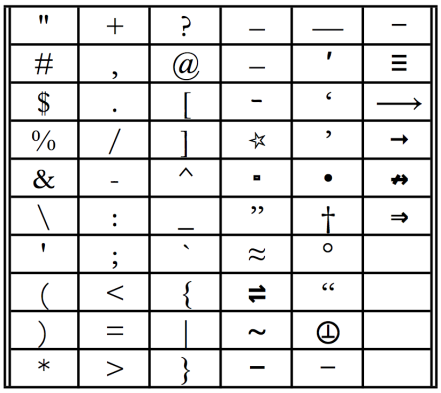

Figure 4.1: All the punctuation removed in scraping. Only these were found in appreciable quantities in the training dataset.

Removing hyphens and primes also meant chemical names were fragmented. This was considered acceptable as the fragment words had greater freedom than very specific (possibly singleton) fully formed words, e.g. `5-methyl-1-heptanol` is split to `5 methyl 1 heptanol`, this allows the `heptanol` fragment to be associated with other mentions of heptanol in the training set, rather than assoicate with much less frequent 5-methyl-1-heptanol.

Next, English stopwords were removed [1] from the Porter stopwords corpus**? ?**. From inspection of the zipfian frequency table, (section 2.4.4), it was apparent that chemistry literature also generates stopwords. Table **??** details 'Chemistry' stopwords that were identified and removed, as they carried little specific information.

Table 4.1: Chemistry stopwords

chemistry	containing	7	six	water
structure	novel	8	seven	also
structural	study	9	eight	method
study	studies	0	nine	molecular
new	1	zero	ten	studied
using	2	one	phase	
based	3	two	based	
reaction	4	three	compounds	
reactions	5	four	high	
chemical	6	five	results	

[1] Stopwords are commonly occurring words in a corpus that hold little information, e.g. ('the', 'a', 'and'...)

Finally, the processed words were sent through a 'stemming algorithm'[2]. Several stemming algorithms were assessed (Porter **?**, Snowball **??**, Lancaster **?** and the Wordnet Lemmatizer **?**,**?**,**?**). The Snowball [3]) stemmer was found to strike a good balance between making an appreciable number of contractions (Wordnet) whilst minimising conflations and over-contraction (Lancaster). See Table **??** The document preprocessing pipeline is

Table 4.2: Stemming

Word	Porter Stemmed	Snowball Stemmed	Comment
phyllenthus	phyllenthu	phyllenthus	Overagressive stemming by Port
angularly	angularli	angular	Adverbs map to root better
infinitly	infinitli	infinit	Snowball maps these to correct ro
infinite	infinit	infinit	
Word	**Lancaster Stemmed**	**Snowball Stemmed**	**Comment**
pigment	pig	pigment	Lancaster collapses too far
conductive	conduc	conduct	Lancaster conflates these different v
conducive	conduc	conduct	
scripting	scripting	script	Lancaster doesn't consider present par
aroma	arom	arom	For chemistry work,preferable not to map aron
aromatic	arom	aromat	

shown diagrammatically in figure **??**:

Pre processing pipeline

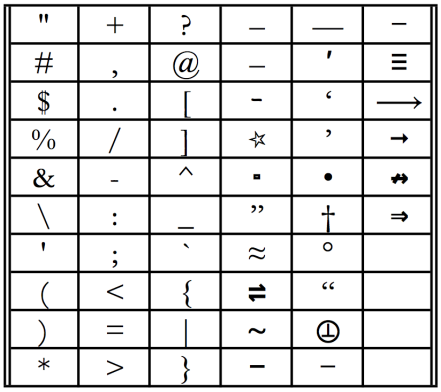

Figure 4.2: All documents in the training database were preprocessed with this pipeline schema before being used in training models

The process is best illustrated by real collected abstract from the dataset:

[2]A stemming algorithm seeks to map derived words onto the same root, such as polymer and polymers, but some also attempt more complex cases such as morphologic and morphology

[3]Also known as Porter2

< p > n A 9-silafluorene-containing biphenolic monomer, 9,9-bis(4-hydroxyphenyl)-9-silaf
was prepared from 9,9-dichloro-9-silafluorene and employed for the synthesis
of polyesters using a fluorene-based homoditopic acid chloride. < \ p >.
?

is processed into:

silafluoren biphenol monom bis hydroxyphenyl silafluoren prepar dichloro silafluoren
employ synthesi polyest fluoren homoditop acid chlorid

Whilst difficult for a human to read, text order is preserved and low information words (or words with complicated, diverse meanings such as numbers) have been removed to give good quality input data. Note how chemical names have been fragmented so that more, simpler, chemical vectors can be learned, rather than the fewer complex vectors (9,9-dichloro-9-silafluorene vs dichloro andsilafluoren).

4.3 Word2Vec Models

The processed data was used to train two Word2Vec models using the gensim python implementation Řehůřek and Sojka [2010].The hyperparameters used for training were consistent for the two models. Training was carried out on all documents in the training dataset. The model was trained with sentences formed by simple splitting of documents using full stops[4]. One model was was trained using the CBOW architecture, the other using Skip-gram. After examination of different hyperparameters, the models were run using mainly default hyperparameters, representing good balance of specificity, speed and generalityŘehůřek and Sojka [2010]. Namely the hyperparameters used were:

Table 4.3: Chemistry stopwords

Model Parameter	CBOW and skipgram
Vector Dimensionality	100
Minimum word frequency	1 (all words)
Initial learning rate α	0.025
Minimum learning rate α_{min}	0.0001
Epochs of training	24
Sliding word window size	5
Negative sampling	Yes
Downsampling parameter	0.001
Hierarchical Softmax	No

[4]Whilst not perfect, this method was a good compromise for partitioning on actual sentences and false partitioning.

4.3.1 Aggregation Techniques

To Do

4.4 Doc2Vec Models

To Do

5. Validation of Algorithm

5.1 Visualisation Techniques

5.2 Simulated Inputs

5.3 Word Similarities

5.4 Document Similarities

6. Analysis with Sample Dataset

To Do

7. Conclusions

To Do

8. Recommendations for Further Work

TO DO

Bibliography

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013a. URL `http://arxiv.org/abs/1310.4546`.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT-2013)*. Association for Computational Linguistics, May 2013b. URL `http://research.microsoft.com/apps/pubs/default.aspx?id=189726`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013c. URL `http://arxiv.org/abs/1301.3781`.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL `http://arxiv.org/abs/1405.4053`.

9. Appendix

9.1 UK Departments scraped

Department	URL
Aberdeen	http://www.abdn.ac.uk/chemistry/
Aston	http://www.aston.ac.uk/eas/about-eas/academic-groups/ceac/
Bangor	http://www.bangor.ac.uk/chemistry/index.php
Bath	http://www.bath.ac.uk/chemistry/
Belfast (Queen's)	http://www.qub.ac.uk/schools/SchoolofChemistryandChemicalEnginee
Birmingham	http://www.birmingham.ac.uk/schools/chemistry/index.aspx
Bradford	http://www.brad.ac.uk/acad/chemistry/
Brighton	http://about.brighton.ac.uk/pharmacy/
Bristol	http://www.bris.ac.uk/Depts/Chemistry/Bristol_Chemistry.html
Cambridge	http://www.ch.cam.ac.uk/
Cardiff	http://www.cardiff.ac.uk/chemistry
Dundee	http://www.lifesci.dundee.ac.uk
Durham	http://www.dur.ac.uk/chemistry/
Edinburgh	http://www.chem.ed.ac.uk/
Essex	http://www.essex.ac.uk/bs/
Glasgow	http://www.chem.gla.ac.uk/
Greenwich	http://www.gre.ac.uk/engsci/study/pharchemenv
Heriot-Watt	http://www.eps.hw.ac.uk/institutes/chemical-sciences.htm
Hertfordshire	http://www.herts.ac.uk/research/hhsri/research-areas-hhsri/pharm
Huddersfield	http://www.hud.ac.uk/sas/chemistry/
Hull	http://www2.hull.ac.uk/science/chemistry.aspx
Keele	http://www.keele.ac.uk/chemistry/
Kent at Canterbury	http://www.kent.ac.uk/bio/
Kingston	http://sec.kingston.ac.uk/research/research-centres/
Lancaster	http://www.lancaster.ac.uk/chemistry/
Leeds	http://www.chem.leeds.ac.uk/
Leicester	http://www.le.ac.uk/chemistry/
Lincoln	https://www.lincoln.ac.uk/home/chemistry/
Liverpool	http://www.liv.ac.uk/chemistry/
Liverpool John Moores	https://www.ljmu.ac.uk/about-us/faculties/faculty-of-science/sch
London Metropolitan	http://www.londonmet.ac.uk/faculties/faculty-of-life-sciences-ar
Loughborough	http://www.lboro.ac.uk/departments/chemistry
Manchester	http://www.manchester.ac.uk/chemistry/
Manchester Metropolitan	http://www.sste.mmu.ac.uk
Newcastle	http://www.ncl.ac.uk/chemistry/
Northumbria	https://www.northumbria.ac.uk/about-us/academic-departments/appl

Department	URL
Nottingham	http://www.nottingham.ac.uk/chemistry/
Nottingham Trent University	http://www.ntu.ac.uk/sat/about/academic_teams/chemistry
Open Univserity	http://www.open.ac.uk/science/chemistry/
Oxford	http://www.chem.ox.ac.uk/
University of the West of Scotland	http://www.uws.ac.uk/schools/school-of-science/departme
Plymouth	https://www.plymouth.ac.uk/schools/school-of-geography-
Reading	http://www.reading.ac.uk/chemistry/
Robert Gordon	http://www.rgu.ac.uk/about/faculties-schools-and-depart
St Andrews	http://ch-www.st-and.ac.uk/
Salford	http://www.salford.ac.uk/environment-life-sciences/resea
Sheffield	http://www.sheffield.ac.uk/chemistry
Sheffield Hallam	http://www.shu.ac.uk/schools/sci/chem/
South Wales	http://www.southwales.ac.uk/chemistry/
Southampton	http://www.soton.ac.uk/chemistry/
Strathclyde	http://www.strath.ac.uk/chemistry/
Sunderland	http://www.sunderland.ac.uk/ug/subjectareas/pharmacychem
Surrey	http://www.surrey.ac.uk/chemistry/
Sussex	http://www.sussex.ac.uk/chemistry/
Teesside	http://www.tees.ac.uk/schools/sst/
UEA	http://www.uea.ac.uk/chemistry
Warwick	http://www2.warwick.ac.uk/fac/sci/chemistry/
York	http://www.york.ac.uk/depts/chem/
Bradford Ploymer IRC	http://www.brad.ac.uk/acad/irc/
Cardiff Pharmacy	http://www.cardiff.ac.uk/pharmacy-pharmaceutical-science
Burbeck Chemistry	http://www.bbk.ac.uk/bcs/
Burbeck Crystallography	http://www.cryst.bbk.ac.uk/
Imperial College London	http://www.imperial.ac.uk/chemistry/
King's College London	http://www.kcl.ac.uk/nms/depts/chemistry/index.aspx
Queen Mary London	http://www.sbcs.qmul.ac.uk/
UCL School of Pharmacy	http://www.ucl.ac.uk/pharmacy
University College London	http://www.ucl.ac.uk/chemistry/
Sheffield Computational Chemistry	http://www.sheffield.ac.uk/is/research/groups/chemoinfor