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, learned institutions (such as university websites) began to post records of recent publications and other chemical information freely online. Pubishers 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.

By collecting meta-data 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 SciFinderand Web of KnowledgeTM. 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.

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 to 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 with which future analysis may be performed
  + 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. §3 and §4 provide the background of the algorithms 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 §5. 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.

Internet webpages are written in HTML. 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 three 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’.

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.

The initial approach was to analyse the HTML tree to automatically recognise useful tabulated or listed data. The program started at the tree’s root and repeatedly fol- lowed the branch with the most ‘repeating substructure’. The recursive algorithm is summarised below:

1. Count # of descendents of each child node

Calculate the pairwise similarities between all child nodes

Consider two nodes similar if pairwise similarity is above a

heuristic threshold

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 sufficiently flexible for the task of scraping large quantities of chemical data, and was not developed further.

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)

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.2 [6]. 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.

DOIs consist of a prefix and a suffix. The prefix is subdivided into the Directory Indicator (always integer 10) separated from the Registrant Code, assigned by the issuing body [24]. Registrant codes are numeric and can be a minimum of three integers, with further optional subdivisions separated by full stops. The suffix is provided by the registrant themselves and can be any form of unicode-encoded text [24].

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.

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.

The Regex approach does not require XPaths in order to extract DOIs from a web- page. 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. Fur- ther metadata can be accessed by following the http://dx.doi.org/{DOI} redirecting service by DOI􏰀.org. to visit publishers’ websites to collect remaining meta-data.

With this methodology in place, a scraping program was written to collect DOIs from a list of webpages and collect meta-data in a two 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 XPaths3. The procedure is summarised in figure 2.4.

The programmatic steps depicted in figure 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 http://dx.doi.org/{DOI} link

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

The program was first used to collect the data from the UK. The Goodman group’s web- site 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 departments4. The program was run with this list as input, the results of which are detailed in table 2.1. The DOIs collected were stored in database ∆1 and the complete results were stored in database ∆2. Conversion losses were due to four components. 45 losses for non-existant DOIs, 2753 losses to request errors (404 : not-found errors or permission problems), 133 to the program errors and 3148 conversions were lost due missing publication XPaths. The 26 specified XPaths 5 were sufficient to convert 83.8% of successful requests. This was deemed acceptable, as most major publishers had been covered6, and the missing publishers each covered a small number of articles7 The efficiency is depicted in figure 2.5.

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 bearing in mind that the Cambridge chemistry department may be overrepresented in the UK chemistry data set.

Much more data would be required to train a successful machine learning model. One approach would have been to expand the scrape to world-wide chemistry departments, and other learned 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 large, it was programmed to ‘pause’ before publisher abstract collection. The results 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. This database was labelled ∆3.

The publisher distributions were then 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 three days.

Some publishers automatically track request volumes sent to their site as they wish to discourage automatic scraping of their data. Scraping their websites constitutes fair use and is legal in the UK , and despite the university owning a Licence to full access of these publishers’ publications, the material collected was freely available without licence[25] [26]. However, during the scraping run, a bug in the randomisation of request frequencies resulted in detection by the ACS 8 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 scraping runs would be performed.

Taylor & Francis banned the IP address after it detected over 100 requests were made within five minutes. This corresponds to a request every three seconds. This was a 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 increase broke the threshold of allowed requests at the ACS server which then banned the IP (approximately 10 requests a second).

The program was capable of making a total number of approximately 30 requests per second. 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 to ACS. The ban occurred after approximately 150,000 seconds, after which there were no more responses received.

The yield of the global-scale scraping run was cut significantly by the ACS banning event. A summary is tabulated in table 2.2 and shown graphically in figure 2.7. The complete records were stored in a database ∆4.

The overall efficiency of the process is 56.4%, but excluding lost ACS records, the pro- gram’s efficiency jumped to 74.0%, similar to the efficiency of the UK scraping run (§2.4.1). 9

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 written in ascii characters10 (removing majority Japanese and Chinese script). This was done to provide higher-quality data for training the algorithm described in §4. This filtering resulted in a final training database of 464712 articles. This dataset is henceforth referred to as ∆6. The entire database formation process is shown in figure 2.8 and table 2.3.

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

The publisher ‘market share’ can be approximated from examining ∆3.

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.

The corpus of combined titles and abstracts in ∆6 was then examined. The word fre- quencies across all the data were found to be approximately Zipfian, with a gradient of -1.1111 See figure 2.11

A summary of the corpus statistics are shown below:

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, with many applications in NLP. This section explores approaches to representing journal articles in a quantitative manner using NLP.

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 computes the number of distinct words in a corpus of documents, N. It then assigns each document in the corpus an N dimensional vector v. If document A contains word i 2 times, then vA,i = 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 3.2 shows the vector representations for Documents A and B. The higher the scalar product of normalised vA ·vA, the more similar the documents are predicted to be. This model is used extensively in industry. A related model, the bag of citations model, sets vector components according to the presence or absence of citations. Both models are widely used in industry for analysing the publishing landscape.

The Bag of Words model treats words as atomic units, beneficial for robust and fast computation. However, words can have degrees of similarity to each other, and these relationships are not captured by bag of words models [27]. Distributed representations of words have been used to address this for some time [28].

A recent successful approach has been the Word2Vec algorithm [18] [19]. The Word2Vec algorithm uses a neural net to represent words as vectors in a continuous rather than discrete space. Vectors for words with similar meanings will point in similar directions in this ‘semantic space’. The Word2Vec algorithm is fed a language corpus sentence by sentence. The words within the sentences have a semantic relationship, which the algorithm uses to infer word meanings.

This is achieved with two architectures, Continuous Bag of Words (henceforth CBOW) and skip-gram. The CBOW architecture uses a shallow neural net to predict a word’s vector by summing or averaging the vectors of the surrounding words in a training sentence. The skip-gram algorithm predicts the vectors of words surrounding the current training word. By training with many input sentences, prediction vectors are gradually improved.

The training process is shown in figure 3.1. CBOW uses a fixed window of words surrounding the 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 ordering 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 sophistication built in to reduce the importance of very com- monly 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.

Word2Vec models are able to represent concepts by vector algebraic operations on their word representations. Figure 3.2 shows one famous example a Word2Vec model trained on the ‘Google News’ text corpus was able to identify. 1

The Doc2Vec algorithm[29] (an implementation of Paragraph Vectors [30]) allows the Word2vec process to directly learn vectors that represent documents. 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 an entire document can be represented by a vector in a document semantic space.

The nature of the collected meta-data detailed in §2 lends itself naturally to the Word2Vec and Doc2Vec algorithms, as a large store of natural language, rich with encoded infor- mation. The focus of the machine learning analysis phase of the project was directed at applying the Word2Vec and Doc2Vec algorithms to ∆6 to try and automatically learn and classify chemical semantic concepts. These investigations are detailed in the following sections.

The aim of the machine learning phase was to apply the Word2Vec and Doc2Vec algo- rithms to dataset ∆6 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 computational analyses and statistical methods could be performed.

The documents (titles and abstracts) in ∆6 required preprocessing before they could be effectively 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 e.g. ‘Molecule’ and ‘molecule’.

The raw documents also frequently contained artefacts from the source webpages (un- wanted whitspace, vestigial HTML tags, ‘newline’ characters and carriage returns). The algorithm training word vectors for these symbols is clearly undesired behaviour, so these were removed and whitespace normalised.

It was also observed that, as unicode text scraped from a wide variety of sources, there was varied and redundant punctuation. Punctuation would be treated as separate words by the algorithm, so had to be carefully removed. Unicode has very wide variety of different punctuations. For example, unicode encodes 24 different types of hyphen. Table 4.1 shows the punctuation that was filtered out of the documents. Large sections of unicode script (sections of non-western languages) was also removed as the algorithm works best on a smaller vocabulary.

Removing hyphens and primes also meant chemical names were fragmented. This was considered acceptable as the fragment words had greater freedom than specific (possi- bly singleton) fully-formed names, 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 only associate with mentions of the much less frequent 5-methyl-1-heptanol.

Next, English stopwords were removed 1 (stopwords were taken from the Porter stop- words corpus[31] [12]). From inspection of the zipfian frequency table, (§2.4.4), it was apparent that chemistry literature also generates stopwords. Table 4.1 details ‘Chem- istry’ stopwords that were identified and removed2.

Finally, the processed words were sent through a ‘stemming algorithm’3. Several stem- ming algorithms were assessed (Porter [12], Snowball [13][31], Lancaster [9] and the Wordnet Lemmatizer [20][21][22]). The Snowball 4 stemmer was found to strike a good balance between making an appreciable number of contractions (superior to Wordnet) whilst minimising conflations and over-contraction (superior to Lancaster and Porter). See Table 4.2

The document preprocessing pipeline is shown diagrammatically in figure 4.2:

The process is best illustrated by real example from ∆6:

< p > n A 9-silafluorene-containing biphenolic monomer, 9,9-bis(4-hydroxyphenyl)-9-silafluorene, was prepared from 9,9-dichloro-9-silafluorene and employed for the synthesis of polyesters using a fluorene-based homoditopic acid chloride. < \ p >. [32] processed into:

silafluoren biphenol monom bis hydroxyphenyl silafluoren prepar dichloro silafluoren

employ synthesi polyest fluoren homoditop acid chlorid

Whilst challenging to read, word order is preserved and low information words (or words with complex, diverse meanings such as numbers) have been removed to give good- quality input data. Note how chemical names have been fragmented so that multiple chemical vectors can be learned, rather than the fewer complex vectors (9,9-dichloro-9-silafluorene vs dichloro and silafluoren).

The processed data was used to train two Word2Vec models (one CBOW, one skipgram) using the gensim implementation [29].The hyperparameters used for training were con- sistent for the two models. Training was carried out on all documents in ∆6. The model

was trained with sentences formed by simple splitting of documents using full stops5. After examination of different hyperparameters, the models were run using hyperparam- eters representing good balance of specificity, speed and generality. The hyperparameters used are detailed in table 4.3.

In order to represent documents as vectors using these models, the component word vectors had to be aggregated into a single vector. There were several possible aggregation techniques, described below.

TF-IDF 6 is an empirical metric for weighting the importance of words in a sentence. If averaging word vectors, it is intuitive that equal weighting should not be given to information heavy and trivial words. The TF-IDF weight, defined as term frequency: TF(w,d) = fw∈d where f(w) is the raw frequency of a term w in a document d,

multiplied by inverse document frequency IDF(w) = log 􏰃 |D| 􏰄 where |D| is the 2 􏰈ddfw∈d)

number of documents in the corpus, df is 1 if word w is in document d, otherwise otherwise 0 [29]. TF-IDF assigns small weights to words that are common across the corpus. It assigns high weights to words that appear often in a document but rarely in the corpus.

Document vectors could be created by averaging word vectors into sentence vectors, followed by averaging sentence vectors into document vectors, or by simpy averaging word vectors directly into documents. 8 models for document vectors composed of Word2Vec models were constructed, detailed in table 4.4

A Doc2Vec model was trained with a distributed memory architecture7 using the gen- sim framework in python [29] using ∆6 with the same sanitiation pipeline as for the Word2Vec models. The training sentences were labelled with the document (journal article DOI) they came from. 100 dimensional vectors were chosen as a compromise of training speed and specificity8, and also so that dimensions were consistent across all models. The Doc2Vec model was trained for 24 epochs, with hyperparamters detailed in table 4.4

The model took considerably longer to train than Word2Vec, as there were appreciably more work required per document. Negative sampling was disabled as per recommen- dations in the literature. [29] [30]. The Doc2Vec and Word2Vec models are assessed in §5.

The models created in §4 were then examined and assessed. As an unsupervised learning algorithm, it is difficult to assess model quality, due to a lack of concrete metrics to com- parisons1. The Word2Vec development team tested models against ∼ 10,000 semantic and syntactic relationships (See Figure 3.2)[18] [19] [27]. The scope of this project does not extend to such elaborate tests. In the section, some examples of model strengths are given and techniques for using word vectors and visualisation are presented.

Word similarities can be be obtained by direct comparison of their word vectors. For words α and β, with vectors να and νβ. A possible metric is to compute euclidean distance between να and νβ,

􏰍􏰌

where D is the dimensionality (D=100). This metric simply describes the distance between the end points of να and νβ. A larger Seuclid indicates weaker similarity. A second similarity metric is the cosine similarity, a measure of the directionality of να and νβ. A value close to 1 corresponds to high similarity of α and β. Cosine similarity is computed as:

the CBOW and skip-gram models were examined using these metrics. For a given word,

they were requested to return the three words examples are given in tables 5.1 and 5.2.

As shown above, the models perform well, returning intuitively similar words to the test word.2. In most cases, chemical inference is represented in some way (e.g. the models understood that catalysts and nanocatalysts are closely-connected concepts).

It was observed that the skip-gram model gave misleading positives more frequently. 3. The CBOW model was considered to be superior for word-word comparisons. It was also noted that CBOW had stronger agreement between euclidean and cosine similarity metrics, however, euclidean similarity gave poorer general perform.4. It is noted that cosine similarity is the accepted similarity metric in the literature [18] [19] [30].

The models detailed in §4 were then tested for document vector similarity. A document was chosen from the corpus, the three most similar articles were computed for each model and results assessed. One test document was DOI: 10.1134/s0036024412120266 [33], the title:

‘Photochemical transformations of anthraquinone in polymeric alcohols’.

The TF-IDF models (CBOW-TFIDF-S, CBOW-TFIDF-W, SG-TFIDF-S, SG-TFIDF- W) suffered from mathematical conditioning problems, and gave poor predictions. The remaining models’ most similar documents 5 for this test document are shown in table 5.3:

The document vectors generated by the Doc2Vec model had considerably better perfor- mance, and thus were taken forward as the model of choice for further analysis.

High Dimensional systems are hard to visualise but there are several methods available to visualise high-dimensional data. PCA6[11], a well-established technique reduces dimen- sionality by a series of orthogonal transformations. T-SNE7, a state of the art technique to reduce dimensionality by preserving spatial clusters of vectors at high dimensions [14]. These techniques allows 100-dimensional document vectors to be collapsed to points on an arbitrary 2D plane, to give a visualised ‘snapshot’ of the semantic space. Figures 5.1 and 5.2 show PCA and TSNE reductions8 on 10,000 document vectors randomly

The PCA reduction shows a dark central area, suggesting most vectors are ‘smeared’ about a common direction. The map is not entirely symmetric which is what would be expected for random vectors. It was expected that document vectors would be dis- tributed into clusters representing particular research fields within the literature. This is borne out by the TSNE reduction, which has resolved many clusters. There is a sig- nificant portion of document vectors scattered between dark cluster spots, which may be could interpreted as ‘interdisciplinary’. TSNE is based upon euclidean distance, which is noted not to be the best similarity measure so, whilst qualitatively useful, TSNE maps were interpreted cautiously.

A similarity matrix C was defined between a sets of documents. For a set of documents A (with a documents) and B (with b documents) define document matrices of document vectors A and B such that

where w and v are document vectors. The cosine matrix C is then defined as Ci,j =

cosine (θi,j ) where element i j contains the cosine between ith document vector in A and jth document vector in B:

where ⊘ and 2 indicate Hadamard division and Hadamard square root , diag(Q) the 1 × n matrix formed from the diagonal of matrix Q. This matrix represents a network where each document in A is a node with an edge to every document in B with a weight equal to the cosine. If A is B, then the matrix is a fully connected network10. This network can be visualised using specialist software11 [35]. Figure 5.3 visualises the same 10,000 document sample as a network graph.

It can be seen that concentrations of documents also form in the network visualisation. There are noticeable outlier documents far from the central clusters 12. Also note that the network visualisation technique is dependent only on cosine similarity, so was considered a more reliable analytical tool than TSNE. Treating the system as a network graph also enables powerful network analytic algorithms to be applied.

Having developed a framework to examine the models, attention was turned to some analyses that could be carried out within the time frame and scope of the project1. With this in mind, it was decided to focus analysis on the a smaller subset of ∆6, namely documents from the University of Cambridge Chemistry Department. This dataset was labelled ∆7.

∆7 contained 9467 documents. The cosine matrix was calculated and a network was constructed from the matrix. Communities within the network (clusters of strongly- connected nodes) were identified by applying a high throughput modularity algorithm[4][5]. The result is shown in figure 6.1.

It was apparent that ∆7 contained clear communities of documents. This corresponds to different fields of research within the department. Some communities detected were small, but some quite large (green, orange, etc...). The algorithm was then applied only to the ‘green’ community, which revealed subcommunities within the ‘green’ documents.

A program was then written to recursively detect subcommunities in ∆7. This resulted in ∆7 being divided into 300 communities of comparable size. The smallest communities were singleton documents, the largest community was 434 documents, and the mean population was 34.5. The community-finding subdivision process is shown in figure 6.2

Figure 6.2 can be interpreted as showing the relationships between different fields of research within the department. The tree is shallow with highly branched nodes, sug- gesting wide research fields, and much qualitative overlap between fields. The process described constitutes an unsupervised categorisation algorithm. The entire process, from model training to finding communities has been performed without human labelling or intuition. It was therefore instructive to examine what the algorithm defined as commu- nities. Communities were examined closely, to reveal that community clustering made

intuitive sense in the majority of cases. Community 275 is typical:

Table 6.1 shows that this particular research community refers mainly to toxicology stud- ies of neonicotinoids, bees and flowers2. The connections mostly make sense. Note the surprising inclusion of the cosmetics and lanthanide silicate studies. Upon investigation, both studies use very similar analytical techniques used elsewhere in the community, and both examined intercalation. 3

Note also that the mean vector for the community was closest to a paper in the training set that summarised the community extremely well. This paper could be considered as a Summary paper. The uses of this kind of analysis include:

• Analysis of literature field - plotting trees such as figure 6.2 can give a relational understanding of how facets of a field link up together.

Research tool: If researching a paper, identifying its community immediately pro- vides the researcher with papers that are related to it. Crucially this is done without simply following citations, so that interesting, perhaps overlooked, links between papers can be found.

Summarising: If a researcher is required to read many papers from a field, they could find the communities involved and begin by reading the ‘summary’ papers.

It is not only articles themselves that can be grouped and analysed, but articles can be aggregated together to represent higher order concepts, such as staff members or research groups, or potentially even departments.

To investigate this further, http://www.ch.cam.ac.uk/publications/authors was scraped

in order to associate the documents in ∆7 with particular staff members. A staff member

vector f was defined as f = 1 􏰇N vi , for an author with N published articles in ∆7, Ni

with document vectors vi (vector mean).

To investigate author relationships, a cosine matrix was created for each pair of authors A and B, with α and β documents respectively, C(A),(B) (see §5.3.2). The similarity between the author pair was defined as

αβ SA,B =􏰈􏰈C(A),(B)

i,j ij

An author similarity matrix can then be built up, S, with elements SA,B = SA,B. A similar technique to that described in §6.1 could have been used to create clusters of au- thors. Since the sample size was now much smaller (47 authors compared to 9467 papers) a more appropriate technique, Dedicated Hierarchical Clustering, specifically UPGMA was applied [15] 4. This method clusters the authors pairwise in a hierarchical fashion. An effective visualisation of the similarities between staff was to plot a clustermap [2] [3].

Figure 6.3 shows the result of generating S and performing UPGMA hierarchical cluster- ing. The authors are labelled by crsid. The dendrogram tree links authors pair-by-pair, illustrating how the clustering was performed, and how closely related clusters are. An enlarged dendrogram is shown below

A striking feature of figure 6.3 is the cluster in the bottom right corner. The dendrogram tree shows the members of this cluster occupy a separate branch of research space than the rest of the department. The staff members involved (Professors Jones and Pyle, Drs. Harris, Archibald and Kalberer) are all members of the Centre for Atmospheric Science. The unsupervised model thus successfully ‘predicted’ their department, and indicated that their work is quite separate from most of the work in the Chemistry Department. This is a real success for the model. The dendrogram was then further examined and broken into distinct branches. Each branch was examined and manually labelled (see figure 9.3). Most clusters make intuitive sense, but there is one core of well-connected, more disparate members (wj10 to jrn34). These members could be interpreted as forming an interdisciplinary cluster.

The value of this method is self-evident. Clustering staff members informs the depart- ment about the width of research (number of clusters), and how resources are partitioned (size of clusters). It should also be stressed that authors are associated without any hu- man preconceptions or bias. Thus perhaps the most valuable author associations are the unexpected ones, and authors should be encouraged to examine their cluster and consider their ‘neighbours’.

As a final data examination, the topic communities found in §6.1 were linked to the staff members. Different metrics for author similarity were developed to investigate if they correlated with the maps produced in §6.2. Firstly, for a topic community C, with documents d ∈ C, and an author A with documents δ ∈ A, we can associate the author with the community if C ∩ A ̸= {}, 5. The function fassoc was defined as

It was noted that there was significant variation in the number of communities that researchers were associated with. A plot of 􏰇Cc fassoc (Cc, A) for each author is shown below:

It can be seen that some authors are widely distributed between communities, whereas others are concentrated. It should be appreciated that communities are not uniformly distributed. For example, there are many communities in ‘Life Sciences’ but few in Atmospheric Chemistry, as such, interpretation of high values in Figure 6.6 directly corresponding to wide research interests should be tentative6.

An association metric Scoincidence between authors A and B was then defined as Where C is the total number of communities. An association matrix was created, SAssoc = Scoincidence (A, B), where high values for author pair A, B indicate they ap-

A,B pear in many research communities together. The matrix was then scaled such that:

SAssoc,scaled = SAssoc/ 􏰂SAssoc + SAssoc􏰃, and scaled from 0 → 1. This was a measure A,B A,B A,A B,B

of how often authors could be found in the same communities. The matrix is shown below:

Figure 6.7 displays where authors have been detected to have similar research com- munity occupations. High values should indicate that authors should ideally collabo- rate/communicate because they publish in the same research communities. Note also that the square patterns of higher values close to the diagonal of the map reproduces the clustering in figure 6.3, lending weight to the validity of both analyses.7.

Having defined a framework for finding where authors share research interest, the next step was to find where authors were actually collaborating. It was possible to identify approximately 700 documents in ∆7 that were co-authored by staff members in the analysis. A heatmap for actual collaboration between authors is shown below, as well as a metric equivalent to the Sassoc,scaled with elements as the sum of the number of communities both staff members have collaborated in.

Both pictures show the same qualitative picture. Similar author pairs (close to diagonal) are more likely to collaborate.

As a final data step, a matrix defined as the difference between an author similarity matrix (e.g figures 6.3, 6.7) and an author collaboration matrix (e.g. figures 6.8, 6.9)

could be interpreted as a recommended collaboration matrix, i.e. where values close to 1 indicate high similarity but low evidence of collaboration, values close to 0 indicate effective collaboration and values close to -1 indicate high collaboration but low author similarity. Author Pairs with values to 1 should be encouraged to consider working together. This matrix is shown below:

This final piece of the analysis section illustrates how the framework developed over the research project may reveal where it might be profitable for authors to collaborate. Returning to the Centre for Atmospheric Science, which was highlighted as a tight, separate research community, it can be seen that there are recommendations for greater collaboration between particular authors within the Centre. Table 6.2 shows the top 20 scores in the Recommended Collaboration Matrix, where there is stronger evidence to suggest these author pairs should collaborate but little evidence was found that they are collaborating.

The matrix row for a particular staff member (Professor Goodman) is plotted below by way of example of what the model considers a staff member’s recommendations to be.

The aim is that these maps and plots may trigger new, constructive debate, and promote effective collaboration in the department. The analyses presented in this section are not exhaustive, and there is potential for more fruitful insights to be found. Please see §8. It should also be noted that the evidence for collaboration is from quite a small sample, and the collaboration metric could be improved by considering other factors than just co-authorship. It is also possible some co-authorships could not be resolved due to data incompatibilities between databases.

Focussing first on the data acquisition phase, the scraping procedure was regarded as a modest success. The volumes of data collected from the UK chemistry departments was respectable, as was conversion rate from the potential results to fully-resolved records (72.9% to give a database of 16363). The actual number of articles from UK chem- istry departments can be confidently predicted to be considerably larger. The limited harvest could be down to the input list of scraping websites being too small. The procedure to identify webpages for scraping was limited where the chemistry depart- ments did not host their own website. This precluded large parts of many important departments. Identifying potential webpages to scrape could have been implemented more effectively. However, the data that was successfully resolved was of high relevance, with few false positive inclusions. The scraping program was robust and efficient, and performed well.

The global scraping can be regarded as a success as the data collected was sufficiently populous and chemistry-specific to enable effective models to be trained. It should also be highlighted all of the datasets created were from freely-available sources, requiring no subscription and could be collected by anyone. This said, it must be acknowledged that most publishers discourage automatic scraping, and the publisher banning was considered as a major failure in the project. That said, it was dealt with swiftly, and did not present a lasting issue. 1.

It should be mentioned that there are existing meta-data stores available (such as PubMed). Whilst using one of these datasets would certainly have been easier, there was no real available dataset spanning chemistry with enough breadth of data. The Training dataset, whilst taking considerable time and effort to create, was heterogeneous and thus was a more suitable tool.

The algorithmic development section can be regarded as successful. The premise of quantitative vectorial representation of chemical articles was realised, especially by the Doc2Vec model. It should be mentioned the TF-IDF models failed to produce well- behaved vectors, which is not well understood. The success of the model can begin to be

seen in the §6, where clustering performances were intuitive and instructive. The poten- tial of the models has not fully explored. It is the author’s opinion that another project could be filled developing further uses of the dataset and extending the methodologies presented. Some model design choices may have limited specificity, such as the decision to use 100 dimensional vectors.2.

The analysis that was performed is most interesting, but the usage to chemists is some- what limited. As a chemical project, it should have been a strong focus to produce results directly useful to chemistry. This was achieved to some extent towards the end of the project, but this point was reached probably slightly too late.

Some further useful applications of the methodologies have been alluded to, but most of these take the form of a service rather than concrete universal insight. Whilst the author would be enthusiastic to implement some of these services (on-demand similari- ties, clustering, recommendations of articles to read, research profiling etc.), the project scope had to be limited at some point.

It is concluded that the aims set out in this project have been addressed, and there were no major barriers preventing the fulfilment of the project brief.

As alluded to in the text, there are several recommendations for further work. The code and data will be improved and amended over time, and is freely available under MIT licence on request 1. If attempting to carry out further work on this project, it is recommended to contact the author for in-depth explanations. This list is by no means exhaustive, and it is the author’s belief that literature semantic analysis should be considered an important analytical chemical tool.

The principles behind the methods discussed in the project have been shown to be sound. Models should now be improved. Computing resources should be obtained to train higher dimensional vectors 2. The models should also be trained for longer (> 24) epochs on more data (> 460000 documents). These steps will lead to more expressive models.

This very This is to systematically built up to test model intuition.3 This follows the methodologies set out in the literature [18] [19]. Furthermore, is it possible to predict chemical properties using semantic relationships found in the literature? Vec(Compound A) + Vec(Compound B) + Vec(Lab Technique) may give vec(Product C). If so, it may be possible to find unexpected reactions. This could be coupled with the RInChI database to form a new type of data-driven cheminformatics.

Methods have been described for clustering documents. The cluster centres represent the content of the cluster effectively. By finding early papers in the cluster, is it possible to identify influential papers or authors? By clustering on documents from particular years, is it possible to identify a path for the evolving cluster centre vector? If so, it should be possible to extrapolate to predict near future research directions.

With the increase in open source papers, it should be possible to build up a vast dataset of chemical language for training, using the bodies of articles published on open source platforms, and even to use supplied supporting information.

Chemical names could be smartly preprocessed to classes of chemicals, for example by identifying a compound from its name and mapping to InChI key, then to a chemical class. This would allow better association of chemical fragments in training.

In Training Doc2Vec, by specifying document with more than just their unique identifiers allows more vectors to be associated. By identifying and labelling all documents with a particular concept, e.g. ‘palladium-catalysed’, and then training Doc2Vec, one defines an ’palladium-catalysed’ vector, specifically trained for the concept. These concept vectors would be robust and information-rich4