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, learn`ed 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 article meta-data. This 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 should, 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 language can be built. Machine Learning techniques can be applied to find novel connections between articles, 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 productive. Several publishers perform large-scale analysis and provide literature tools, such as SciFinder and Web of Knowledge. 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 quantity 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 identify and extract chemical informa tion

Store information in human-readable, computer readable, stable, scalable for mats

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 these models 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 databases were created (UK Department chemistry databases and very large databases of unrestricted chemistry-related material).  Once the databases were set up, focus was shifted to how to mine the data for 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, covered in §5. Finally, when the models were shown to be performing successfully, they were used in an analytical setting to examine relationships between authors and research communities in the University of Cambridge Chemistry Department, and eventually to recommend specific collaborations between staff (§6).  The appendix goes into further detail, discusses results, implications, additional methods as well as technical details and recommended further work. There is an extensive glossary for terms used in the project.

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 it in a human-readable format.

A scraping 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 as a table with three rows, each row containing ‘Table Data A/B/C’. This data is accessible programmatically using an ‘XPath’.

XPaths are simply paths through this tree to the desired information. In order the ‘scrape’ the data in the table, the following XPath could be used:

Scraping millions of webpages potentially requires millions of different XPaths. 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 data generate XPaths1. When this strategy proved unsuitable, a new method was required. Chemical information is usually disseminated as journal articles accompanied by a DOI. By programmatically collecting DOIs (§2.2.1) it was possible to build up a large database of chemical information (§2.2.2)

DOIs are computer-friendly labels for articles. DOIs are issued by a number of accredited bodies, with the majority indexed by Crossref[6]. By pre-pending a DOI with the url stub http://dx.doi.org/, The IDF service redirects the request to the publisher’s website to display the article the DOI indexes. The DOI structure is shown in figure 2.2.

DOIs consist of a prefix and a suffix. The prefix is subdivided into the Directory Indicator2 and the Registrant Code, assigned by the issuing body[24]. Registrant codes are a minimum of three integers, with further optional subdivisions separated by full stops. The suffix is provided by the registrant and can be any form of unicode-encoded text[24].

It was possible to write a ‘Regular Expression’ pattern (REGEX) to automatically recognise DOIs within a body of text (figure 2.3). The flexibility of the registrant code specification means that DOIs cannot always be unambiguously identified in HTML.

The REGEX was able to identify 90.4% of the DOIs on http://www.ch.cam.ac.uk/ publications.

This REGEX approach does not require XPaths in order to extract DOIs from a webpage. This facilitates large-scale scraping from many websites. Some meta-data3 associated with a DOI can be accessed using an online API exposed by Crossref. The remaining meta-data can be accessed by following the http://dx.doi.org/{DOI} link to visit publishers’ webpages.

With this methodology in place, a scraping program was written to collect DOIs from a list of webpages, collecting meta-data in a two stage process. The Crossref API provides article titles, journals, authors, publisher and date of publication meta-data, but not article abstracts, which had to be collected by visiting publisher webpages, and using hand-written XPaths4. The procedure is summarised in figure 2.4.

The programmatic steps depicted in figure 2.4 are:

Request the webpage from the inputted list  Process the html and extract DOIs  Using the Crossref Online API, verify the extracted DOIs exist Crossref yields metadata:

Title  Journal Publisher  Authors  Publication Date

For each DOI, follow http://dx.doi.org/{DOI} link Use XPath to collect article abstracts

The program exports complete records as .json files, but also feeds to a MongoDB database. Once the program was written, a list of webpages to scrape was required (§2.3.1, §2.3.2).

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 edited to give a list of 68 departments5. The program was run using this list, 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 to request errors (404: not-found errors or permission problems), 133 to the program errors and 3148 to missing publication XPaths. The 26 specified XPaths6 were sufficient to convert 83.8% of successful requests. This was deemed acceptable, as most major publishers had been covered7, and the missing publishers each covered a small number of articles8. 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 hosts the majority of its information under its own domain name, whereas most other departments’ data are hosted on central university domains. The program was instructed to only scrape webpages belonging to department domains, not university domains as a whole. It is noted that the Cambridge chemistry department may be overrepresented in ∆2.

Much more data would be required to train a successful machine learning model. One approach would have been to expand 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 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-result 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. This database was labelled ∆3. Publisher distributions and potential server loads were then carefully considered and capture probabilities were predicted before the second half of the scraping routine was set off to run for three days9.

Most publishers track request volumes sent to their servers to discourage automatic downloading. Scraping constitutes fair use and complies to UK copyright law. Despite the university owning full-access licences to these publishers’ publications, the collected material was freely available without licence[25][26]. During the scraping run, a bug in the randomisation of requests resulted in detection by ACS and Taylor & Francis, which responded by banning the computer’s IP address10. The department librarians were able to restore access, and it was agreed that no further scraping would be performed.

The yield of the global-scale scraping run was cut significantly by the ACS banning. A summary is tabulated in table 2.2 and shown graphically in figure 2.6. The complete records were stored in database ∆4.

The overall efficiency of the process is 56.4%, but excluding lost ACS records, the program’s efficiency was 74.0%, similar to the efficiency of the UK scrape11 (§2.3.1).

The successful 714,370 records were merged with the UK results12. Records were rejected with short titles or abstracts, or if the majority of the title and abstract were not written in ascii characters13. This was done to provide higher-quality data for algorithm training (§4). This filtering resulted in a final training database of 464712 articles. This dataset was labelled ∆6. The database formation process is summarised in figure 2.7 and table 2.3.

It was instructive to examine these databases and derive some simple statistical results, which is explored in §8.8.2.

Natural Language Processing is the application of computer science to study human languages using computers. Machine learning, a class of algorithms for predicting patterns in data, finds 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 A simple example is given below:

Table 3.1 shows vector representations for Documents A, B and C. The higher the scalar product of normalised, the more similar the documents are predicted to be:

and so documents A and B are the more similar. The related bag of citations model sets vector components according to the presence of citations. Both models are used by the scientific publishing industry12.

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

A recent successful approach has been the Word2Vec algorithm[18][19]. Word2Vec uses a neural net to represent words as vectors in a continuous space. Vectors for words with similar meanings will point in similar directions in this ‘semantic space’. Word2Vec is trained by feeding in a corpus sentence-by-sentence. The words within the sentences are semantically related, which the algorithm uses to infer word meanings.

This is achieved with two architectures, Continuous Bag of Words (CBOW) and skipgram. Both use a shallow neural net to predict a word’s vector using vectors of surrounding words in a sentence. CBOW attempts to make words similar the sum or average of the surrounding words, skipgram attempts to minimise distance to surrounding words. 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 surrounding words. The order of words within the window does not matter, but because the window ‘slides’ along as the algorithm considers words word ordering is represented in the model . In skipgram, a random number of surrounding words are used for the prediction vectors for word i.

The model has added sophistication to reduce the importance of 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.

Word2Vec models may 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 identify3.

The Doc2Vec algorithm[29] (an implementation of Paragraph Vectors[30]) allows the Word2vec process to directly learn vectors representing documents. The CBOW architecture is adapted so that, in addition to word vectors, every document is associated with its own vector that contributes to predictions in training. The result is that documents can be represented by vectors in a document semantic space.

The meta-data collected in §2 (a large store of natural language) lends itself to the Word2Vec and Doc2Vec algorithms. The focus of the machine learning analysis phase of the project was directed at applying Word2Vec and Doc2Vec to ∆6 to automatically learn and classify chemical semantic concepts.

The aim of the machine learning phase was to apply the Word2Vec and Doc2Vec algorithms to dataset ∆6 (§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. This section constitutes an experimental section.

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 (unwanted 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. Figure 4.1 shows the punctuation that was filtered out of the documents. Large sections of unicode symbols (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 (possibly 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 removed12. From inspection of the zipfian frequency table, (§8.8.2), it was apparent that chemistry literature also has stopwords. Table 4.1 details ‘chemistry’ stopwords that were identified and removed. The chemistry stopwords were chosen from high on the rank frequency table (they appeared extremely commonly) and because they were considered to encode little information; for instance, the digits appeared so frequently and in such an wide set of contexts, no meaningful vector would be trained. The word structure appeared so frequently, so as to encode very little actual information.

Finally, the processed words were sent through a ‘stemming algorithm’3. Several stemming algorithms were assessed (Porter[12], Snowball[13][31], Lancaster[9] and the Wordnet Lemmatizer[20][21][22]). The Snowball4 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:

processed into:

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 goodquality input data. Note how chemical names have been fragmented so that multiple chemical vectors can be learned, rather than the fewer complex vectors

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 consistent 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 hyperparameters 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 (Term-frequency Inverse-Document-Frequency) 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:

where is the raw frequency of a term w in a document d, multiplied by inverse document frequency where is the number of documents in the corpus,

1 if word w is in document d, 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 architecture6[29] using ∆6 with the same sanitation 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 specificity7, and also so that dimensions were consistent across all models. The Doc2Vec model was trained for 24 epochs, with hyperparameters detailed in table 4.5

The model took longer to train than Word2Vec, as there were more work required per document. Negative sampling was disabled as per recommendations 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 challenging to assess model quality, due to a lack of concrete metrics for comparisons1. The Word2Vec development team tested with approximately 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 obtained by direct comparison of their word vectors. A possible metric is to compute euclidean distance. For words α and β, with vectors and

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

were examined using these metrics. For a given word, the models were requested to return the three words in the corpus with highest similarity. The cbow and skipgram models some examples are given in tables 5.1 and 5.2

As shown above, the models perform well, returning intuitively similar words to the test word2. In most cases, chemical inference is represented in some way3.

It was observed that the skipgram model gave misleading positives more frequently4. CBOW was considered to be superior for word-word comparisons. It was also noted that

CBOW had closer agreement between and, however, euclidean similarity gave poorer general performance5. It was noted that 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 TF-IDF models (CBOW-TFIDF-S, CBOW-TFIDF-W, SG-TFIDF-S, SG-TFIDFW) suffered from mathematical conditioning problems, giving poor predictions. The remaining models’ most similar documents 6 for this test document are shown in table 5.3:

The document vectors generated by the Doc2Vec model had considerably better general performance, and were selected as the model of choice for further analysis.

High Dimensional systems are difficult to visualise but there are several methods available to visualise high-dimensional data. PCA[11] and TSNE[14][34] techniques allow 100-dimensional document vectors to be collapsed to points on an arbitrary 2D plane, to give a visual ‘snapshot’ of the semantic space7. 10,000 documents were randomly selected from ∆6 (referred to henceforth as database ∆R). Figures 5.1 and 5.2 show PCA and TSNE reductions on the Doc2Vec vectors of ∆R[35].

The PCA reduction shows a dark central area, suggesting most vectors are ‘smeared’ about a common direction. The map is not symmetric which is what would be expected for random vectors. It was expected that document vectors would be distributed in clusters representing particular research fields within the literature. This is indeed seen in the TSNE reduction, which resolved many clusters. There are document vectors scattered between dark cluster spots, which may be could interpreted as ‘interdisciplinary’8. TSNE is based upon euclidean distance, which is noted not to be the best similarity measure. Whilst qualitatively useful, TSNE maps were interpreted cautiously.

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

where w and v are document vectors. C was then defined such that element i,j contains

the cosine between ith vector in A and jth vector in B:

where ⊘ and 2 indicate Hadamard division and Hadamard square root, diag(Q) the

1 × n matrix formed from the diagonal of Q.

C represents a network where each document in A is a node with an edge to every document in B. Edge weights were equal to the cosine similarity. If A=B, then the matrix is a fully connected network9. This network can be visualised using specialist software[36]. Figure 5.3 visualises ∆R as a network graph.

Concentrations of documents also form in the network visualisation. There are noticeable outlier documents far from the central clusters10. 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 analysis 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 a smaller subset of ∆6, 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 modularity algorithm[4][5]. The result is shown in figure 6.1.

It was apparent that ∆7 contained clear communities. This corresponds to different fields of research within the department. Some communities were small, but most large

(green, orange, etc...). The algorithm was then re-applied only to the ‘green’ community, which revealed subcommunities. A program was then written to recursively find subcommunities in ∆7. This resulted in ∆7 being divided into 300 communities of comparable size. The smallest communities were singleton documents, the largest 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 relationships between different fields of research within the department. The tree is shallow with highly branched nodes, suggesting wide research fields, and much qualitative overlap between fields. The process constitutes an unsupervised categorisation algorithm2. It was instructive to examine what the algorithm defined as communities. Communities were examined and 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 studies of neonicotinoids, bees and flowers3. The connections mostly make sense. Note the surprising inclusion of the cosmetics and silicate studies. Upon investigation, both studies used very similar analytical techniques used elsewhere in the community, and both examined intercalation4 5.

Note also that the mean vector for the community was closest to a paper in ∆6 that summarised the community extremely well6. This paper could be considered as a Summary paper. The uses of this kind of analysis include:

• Analysis of literature field trees such as figure 6.2 can give an understanding of how facets of a field link up.

• Research tool: If researching a paper, identifying its community immediately provides the researcher with related papers. This is done without following citations, so that interesting, perhaps overlooked, links 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 ‘summary’ papers.

It is not only articles themselves that can be grouped and analysed. Articles can be aggregated together to represent higher concepts, such as staff members7. 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 cosine matrix was created for each pair of authors A and B, authoring α and β documents respectively,(see §5.3.2). The similarity between the author pair

was defined as An author similarity matrix can then be built up with elements

A similar technique to that described in §6.1 could have been used to create clusters of authors. 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]8. 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

and performing UPGMA hierarchical clustering. The dendrogram tree links authors pair-by-pair, illustrating 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 shows the members of this cluster occupy a separate branch of research space than the rest of the department. The staff members involved9 are all members of the Centre for Atmospheric Science. The unsupervised model thus successfully ‘predicted’ their department, and indicated that their work is separate from most of 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 6.5). Most clusters make intuitive sense, but there is a 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 department 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 human preconceptions/bias. 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

with documents

and an author A with documents, we can associate the author with the community if The function was defined as

It was noted that there was significant variation in the number of communities researchers were associated with. A plot of

for each author is shown below:

It can be seen that some authors were widely distributed between communities, whereas others were concentrated. It was noted that communities were not uniformly distributed. For example, there were 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 tentative10.

An association metric between authors

and was then defined as Where C is the total number of communities. An author association matrix was created where high values for author pair A, B indicate  they appear in many research communities together. The matrix was then scaled such

that

and normalised from 0 → 1. This was a measure of how often authors published in the same communities. The matrix is shown below:

Figure 6.7 displays where authors have similar research community occupations. High values should indicate that authors should ideally collaborate/communicate because they publish in the same research communities. Note also the square patterns of high values close to the diagonal of the map reproduce the clustering in figure 6.3, lending weight to the validity of both analyses.11.

Having defined a framework for finding shared research interests, 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. A heatmap for co-authorship

between authors is shown below, 12, as well as a metric equivalent to the with elements as the sum of the number of communities in which both staff members have co-authored,

Both maps show similar qualitative pictures: 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 matrix13.

Author Pairs with values to 1 should be encouraged to consider working together. Matrix is one possible example, shown below:

This final piece of the analysis section illustrates how the framework developed over the research project reveals where it might be profitable for authors to collaborate. Table 6.2 shows the top 20 scores in, where there is stronger evidence to suggest these author pairs should collaborate but little evidence was found that they are collaborating14.

The matrix row of 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 constructive debate, and promote effective collaboration in the department15. 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-authorship16.

Focussing first on the data acquisition phase, 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 16363 records). The actual number of articles from UK chemistry 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 departments did not host their own website, precluding large parts of many important departments. The data that was successfully resolved was of high relevance, with few false-positive inclusions. The scraping program was robust and efficient.

The data collected in global scraping was sufficiently populous and chemistry-specific to enable effective models to be trained. It should be highlighted all the datasets were created from freely-available sources, requiring no subscription. alternative scraping tactics may be required to avoid IP banning if further scraping is required. This said, it must be acknowledged that the publisher banning was considered as a major failure in the project1.

It should be mentioned that there are existing meta-data stores available (e.g. PubMed). Whilst using one of these datasets may have been easier, there was not a suitable available chemistry dataset with satisfactory breadth of data. ∆6 was heterogeneous and thus a more suitable tool.

Thee premise of quantitative vectorial representation of articles has been realised, especially by the Doc2Vec model. The TF-IDF models failed to produce effective vectors, which is not well understood. The power of the model can begin to be seen in §6, where clustering performances were intuitive and instructive. Some model design choices may have limited specificity, such as the decision to use 100 dimensional vectors2.

There is further potential within this project. Some further useful applications of the methodologies have been alluded to, some of which take the form of a service or tool rather than concrete insights (on-demand similarities, clustering, recommendations of

articles to read, research profiling etc.). The project scope had to be limited at some point. There are detailed discussions of recommended work, and some further investigations/analysis in the appendix §8, especially §8.3 on chemical element analysis3 4.

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.

The appendix both expands on details of the work in the main section and explores other analyses. Of principle interest is §8.3, an investigation into word vectors and chemical elements using complementary methods to §6. Recommendations and technical details for further work is also included. This project was expansive and the appendix largely serves to include work that could not be fit into the main dissertation due to size limits.

As alluded to in the text, there are several recommendations for further work. If carrying out further work on this project, it is recommended to contact the author for in-depth explanations. It is the author’s belief that literature semantic analysis should be considered an important analytical chemical tool.

Models should now be improved. Computing resources should be obtained to train higher dimensional vectors. The models should also be trained for more (> 24) epochs on more data (> 460000 documents) leading to more expressive models.

This project focussed mainly on document vectors. However, word vectors may be very useful (see §8.3). Methods for testing the quality of improved models should be develope, e.g relationship-testing: e.g. Fluorine is to Fluoride as Chlorine is to ... . Thousands of these relationships could be systematically built up to test model intuition1 following the methodologies set out in the literature[18][19]. Is it possible to predict chemical properties using semantic relationships in the literature? Vec(Compound A)

+ Vec(Compound B) + Vec(Lab Technique) may give vec(Product C). If so, it may be possible to highlight unexpected reactions. This could be coupled with the RInChI database to form new types of data-driven cheminformatics.

Methods have been described for clustering documents. The cluster centres represent the cluster content effectively. By finding early papers in the cluster, is it possible to identify influential papers/authors? By clustering on documents from particular years, is it possible to resolve a path for the evolving cluster centre vector? If so, it may be possible to extrapolate to predict 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 documents with more than just unique identifiers allows more vectors to be trained. By identifying and labelling documents with a particular concept, e.g. ‘palladium-catalysed’, and then training Doc2Vec, a ‘palladium-catalysed’ concept vector can be trained. These concept vectors would be easily mine-able2

There was not sufficient time to explore the clustering found by TSNE reductions. TSNE is a popular technique in the current machine learning literature, and should be investigated more thoroughly. K-Means clustering performed on the TSNE maps was briefly investigated before rapid progress was made by other techniques. There was evidence to suggest that TNSE K-Means clustering was potentially useful, but time did not permit investigation.

In the interest of future work, this section details the technical details of artefacts provided with this project.

The python code used in this project was written in a largely self-documenting style. The time limits did not permit for anaconda packages to be provided, but the code is fully commented and docstrings are provided. Documentation is provided in html (recommended) and a 31 page pdf form. There are also comprehensive Jupyter Notebooks as tutorial guides for using the code[37]. The core code has been presented in a ‘package’ style. The module was named fruitbowl with five submodules,

* Cherry for operations concerning scraping and data collection.
* Orange for operations concerning NLP corpus creation and big data memory friendly streaming
* Strawberry for operations concerning Word2Vec and Doc2Vec model Training
* Apple for operations concerning analysis of trained models (visualisation, export  management etc.)
* Pomegranate for operations interfacing with Gephi and community generation.  There are approximately 30 python source files included in the module. If using the code it is recommended to read and adapt the jupyter notebooks Fruitbowl Example 1.ipynb and Fruitbowl Example 2.ipynb. It is recommended to write code in a directory that contains the fruitbowl module. The module is free to distribute and adapt under the MIT licence, which must be included in any copy. The list of dependencies required for fully functional behaviour for the fruitbowl suite is as follows:  • Python 2.7: Developed on Python 2.7.11 (recommended version) • Python 2 external modules required:  – matplotlib 1.5.1: Plotting modules[38] – Seaborn 0.7.0: Extension to plotting modules and data analysis[2] – numpy 1.10.4: Computational Library[39] – Scikit-Learn 0.17: Machine learning library[35] – Scrapy 1.0.3: Scraping framework – Gensim 0.12.2: Natural Langauge Processing library[29] – nltk 3.1: Natural Language ToolKit library[31] – pandas 0.17.1: Data analysis and management library[40] – pymongo 3.0.3: Python driver for MongoDB database – requests 2.9.1: Web scraping library – scipy 0.17.0: Scientific computing library[3] – jupyter 1.0.0: Jupyter notebooks will be required to use the tutorial notebooks[37].  59
* JDK Java Development Kit for Gephi graph analysis via gephi api
* apache-maven-3.3.9 Java dependency manager for Gephi graph analysis via  gephi api
* C Compiler for use in BHTSNE reductions[34].
* mongoDB The program was built around use of MongoDB. Not strictly necessary  but strongly recommended. Recommended versions >3.2.

Data used in the project was dumped from their mongoDB databases is also supplied in .json format. The data provided is as follows:

* Delta1.json : These are the DOIs found in the UK scrape
* Delta2.json : These are the complete meta-data results found in the UK scrape
* Delta3.json : These are the DOIs found in the global scrape
* Delta4.json : These are the complete meta-data results found in the global scrape
* Delta6.json : This is the data used for training and analytical purposes in the  project
* Delta7.json : The subset of ∆6 from Cambridge used in §6
* cbow model : Gensim binary saved model for final cbow Word2Vec model used in  the project
* sg model : Gensim binary saved model for final skipgram Word2Vec model used  in the project
* FULL DOC2VEC : Gensim binary saved model for final Doc2Vec model used in the  project Note ∆5 is not provided to save disk space (It is simply ∆2 combined with ∆4).

As an investigation into the utility of word vectors trained, it was decided to briefly investigate the word similarities between chemical elements. This analysis is included as an appendix as there was not sufficient space for it to be included in the main body, and due to its ‘self-contained‘ nature. It was hoped that this short investigation would provided evidence that methods sketched in §8.1.2 could work.

The similarity matrix was produced for chemical elements mentioned in the text corpus (115 out of 118 known chemical elements). This required mapping both chemical names and symbols together (e.g. chlorin3 and cl for chlorine) to represent the concept vector for the elements in question.

A modified data sanitation pipeline was created to substitute the chemical symbol for the chemical name. This was only done for chemical symbols longer than 1 letter to dissuade conflating different concepts to the same word vector (S could represent Sulfur or a stereochemical label.)

A CBOW model was trained using this modified input data with the same presets as the main CBOW model and ∆6, detailed in table 4.3. The Cosine Similarity matrix was produced for the 115 elements found in the corpus. UPGMA clustering was performed[35], as well as graph visualisation with modularity clustering[4],[5]. The dendrogram of the UPGMA clustering is shown in figure 8.1. The process identified 5 main branches:

* The gold region includes a sub-branch of noble gases, the other branch mainly actinoids.
* The magenta region contains non-metals mostly associated with organic compounds
* The cyan region contains mainly metalloids, actinoids and alkali metals
* the red region contains mainly transition metals
* The green region contains almost exclusively lanthanoid metals  The dendrogram shows that the classifications broadly fall into intelligible categories within the periodic table. There are, however, some surprises, especially the halogens, with bromine in the actinoid subbranch, and chlorine associating with copernicium. This may be because the symbols Cl and Cn occur together often in the literature due to mentions of carbon and nitrogen, not copernicium. Similar reasoning can be used for bromine (cf br could refer to a CFC rather than californium and bromine) This exposes a flaw in the symbol/name association process that could be tackled in further work.  The graph visualisation of the same Cosine Similarity matrix is shown in figure 8.2 (Also the front cover of this dissertation). Period 7 was removed from this graph as
* there were too few mentions in the corpus for reliable vectors4, and to remove cluttering nodes
* 5 distinct communities were identified:
* • The orange community contained mainly alkali metals and alkali earth metals • The blue community contained mainly lanthanoids • The light green community contained mainly transition metals
* The purple community separated into two spatially distinct regions. The northern region generally contained transition metals and metalloid, and the southern region contained organic non-metals.
* • The dark green community contained noble gases.
* The community finding process reflects a similar situation to the UPGMA process, but is perhaps more successful. The removal of the actinoids appears to have improved community finding. The community finding process was repeated with period 7 included, resulting in broadly the same communities, but with bromine and chlorine leaving the community marked in purple to join a a loose community of actinoids, however they remained strongly associated with each other. The degree of connectivity between nodes is similar for most nodes, but larger for some nodes, e.g. beryllium, which is difficult to interpret.
* Attention was turned to a practical example. Palladium is used widely in catalysis but is rare and expensive, and alternatives would be economically and environmentally beneficial[41]. With this in mind, the cosine similarities of all the elements to palladium vector were computed, and a selection of metallic elements with high similarity is shown in figure 8.3.
* Platinum, rhodium, ruthenium, copper and nickel all had very high scores. The models could be interpreted as suggesting that these metals have similar properties to palladium. This is very much the case for platinum and rhodium (pd, pt and rd are all platinium group metals)[42]. Nickel and copper are predicted to be similar to palladium, and there is evidence that nickel could be used for some palladium-catalysed reactions[43], whereas copper is often combined with palladium to form more effective catalysts[44]. Thus it could be argued the models suggest that more attention should be focussed to nickel catalysis.
* This analysis, whilst brief, is promising. This lends weight that more in-depth considerations of word vectors and concept vectors would be fruitful.

This section briefly re-examines community 275, discussed in §6.1. There was not room for this section to be included in the main body, so is included as an appendix.

Table 6.1 details some of the contents of community 275. Most of the articles in community 275 were published by members of staff who are no longer in the department. The only authors currently in the department are Dr. Kalberer (1 out of 15 articles) and Dr. Vignolini (3 out of 15 articles). These two authors work in very different fields. Dr. Vignolini has worked on plant microsctructures (including pollen) and Dr. Kalberer has worked on atmospheric affects of pollen particles. It could be argued that these two researchers could benefit from discussing each others’ work. The program has thus found an unexpected, non-obvious link between these researchers. These unexpected links can be extracted as follows:

The co-occurance of authors is represented in MAuth.Coinc.(figure 6.7 in §6). To emphasise where authors often appear together in communities but do not collaborate, a more extreme recommendation matrix could be defined, by setting values in MAuth.Coinc. to zero if the author pair have ever collaborated.

Because of the UPGMA clustering, the ordering of the authors in the matrix reflects their similarity (authors are adjacent to similar authors), so that pairs near the diagonal are close. If we select the high values that are distant from the diagonal, these are the more ‘unexpected’ pairings. Setting a minimum distance from diagonal of 10 authors (11 was the population of the largest dendrogram branch identified in 6.5 so a distance of 10 ensured author pairs in the same branch were not included), the following heatmap (fig 8.4) of ‘unexpected’ links is created, MUnexpected.

The top 20 ‘unexpected’ results are shown in table 8.1. They are ordered by their distance from the diagonal, to attempt to highlight high scores but also the more ‘unexpected’ links.

Due to the nature of research in the department, even these ‘unexpected’ pairings can be rationalised. This analysis could be taken further to find the most surprising results using more sophisticated or different techniques. This analysis could be generalised to larger sets of researchers, not just those in the department. The analyses presented in this section and §6 are intended to be useful in their own right, but mainly to serve as the basis for further sophistication.

In this section, the suggested collaboration table 6.2 is briefly explored to rationalise some of the suggested pairings. Limitations should be explained. The recommended collaborations arise from a balance of two factors: how similar the pair are, and how often they have collaborated in the past. The evidence that members are collaborating is taken from http://www.ch.cam.ac.uk/publications/authors, finding articles that are in ∆7 and considering authors to be collaborating if they co-author papers. This is not a particularly robust metric, as there were only about 700 co-authored paper found in this process, a small sample. This is why the main body of §6 refers to ‘evidence of collaboration’ rather than concretely stating that authors are collaborating. In order to build better metrics, citations and a wider body of co-authorship data would be required, which is beyond the scope of this project.

This said, the table is still useful as a guide to who these staff members should focus future collaboration with. It does not assert that these authors are not already collaborating, only that they should treat the collaboration recommendation as a useful guide to who would be fruitful to work with, as their work is quite strongly related.

The top suggestion is for Dr. Archibald to work with Prof. Harris. They both work in the Centre of Atmospheric Science. There is one instance of collaboration on http://www. ch.cam.ac.uk/publications/authors but this article was not successfully collected to ∆7. If this evidence was represented in the dataset, it is likely their recommendation score would drop. It is so high is because their work was considered very similar by the models, pushing their recommended collaboration score high.

David Wales and Daan Frenkel were second highest, and there is also evidence to suggest they have collaborated in the past on http://www.ch.cam.ac.uk/publications/ authors. Some of this evidence was represented in the collaboration matrix, but perhaps with more data, the association would have been stronger. Their research was considered very similar and this outweighed the evidence of collaboration to give a high score.

This trend (some evidence of collaboration, but strong similarity) is present in most of the top 20.

It should be noted that date of publications was not a factor considered in the analysis. This goes some way to explaining why authors who were not simultaneously at the department for long periods have high recommended collaboration scores, as they would not have collaborated. For example, this is probably the case for Dr. Andrew Bond, who features several times in the table. As a recent staff member who specialises in crystallographic techniques, there has not been a great deal of time for collaboration and co-authorship. Many authors publish articles mentioning crystallography, and so Dr. Bond’s work will have high similarity to several members in the department.

It was mentioned in §5.3.2 that singletons (articles with few to no connections to others or those that form their own community)in the graph were predicted to be significantly different to most articles. The singletons produced analysing ∆7 in §6.1 were examined to test this hypothesis.

It was certainly the case that some of the 33 singletons produced in the community finding process were poorly formatted or different. There was one article that detailed RSC award winners, one symposium handout, and 14 did not have well resolved abstracts. However the majority of singletons appeared to be mostly normal (if not quite narrowscope) articles. The average number of words in a singleton document was lower than the average for ∆7 (65.7 words vs 125.7). It appears that in the majority of cases, singletons lie ’just outside’ a threshold for inclusion, rather than being freak anomalies.

This section briefly explains the automatic XPath generation method initially developed to collect chemical data. 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 followed the branch with the most ‘repeating substructure’. The recursive algorithm is summarised below:

1. 2.

3.

Count # of descendents of each child node

(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

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 (such as long tables, lists etc. ), but performed poorly for smaller or unstructured 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.

This section provides additional data and explanation on the data acquisition phase of the project.

As mentioned in §2.3.3, Taylor & Francis and ACS banned the scraping computer’s IP address during the second stage of global scraping. This section explores why this occurred.

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 8.5, 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.

There is much to be learnt by examination and simple statistics of the collected data. This section details some of this analysis which was used in development of the scraping program and to inform upon algorithm and processing design choices.

When deciding how many XPaths were required, it was necessary to examine publication profiles. The publisher ‘market share’ can be approximated from examining ∆3.

As shown in shown in figure 8.6, 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 8.7), 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. An understanding of word distributions would inform data sanitiation practices. It is is included here for interes and completeness. The word frequencies across all the data were found to be approximately Zipfian, with a gradient of -1.115 See figure 8.8

A summary of the corpus statistics are shown below: Note that the mean scores are

slightly higher for the UK, suggesting UK universities tend to publish slightly more verbose publications. The mode abstract length for the UK abstracts is significantly below the mean. This is indicative of a skewed, ‘noisy distribution’, which is indeed found when the distribution is plotted (figure 8.9).

As can be seen in the plot, there is significant variation in abstract lengths, with anything from 25 to 200 words commonly observed.

University chemistry departments with suitable websites were considered when building the input list for the scraping program. Table 8.3 details all the departments that were included. A crawler program was written to navigate through these websites and store urls which had DOIs in them for the main program to scrape.

The UK scraping run found articles published by 36 different publishers. These are detailed below in table 8.4.