Analysing Historical Newspapers and Books Using Apache Spark and Cray Urika-GX

Mike Jackson, Rosa Filgueira, Anna Roubickova

Alan Turing Institute / EPCC, The University of Edinburgh

16 August 2019

1. Introduction

In this report, explorations of historical newspapers and books using Apache Spark[1] and the Alan Turing Institute[2]'s Cray Urika-GX system are described. These explorations were done in collaboration with Melissa Terras, College of Arts, Humanities and Social Sciences (CAHSS)[3], The University of Edinburgh. These explorations continue earlier work described in a report on "Analysing Humanities Data using Cray Urika-GX"[4] and its complementary blog post[5]. This work was funded by Scottish Enterprise as part of the Alan Turing Institute-Scottish Enterprise Data Engineering Program.

1.1. Cray Urika-GX

The Cray Urika-GX system [6] [7] [8] is a high-performance analytics cluster with a pre-integrated stack of popular analytics packages, including Apache Spark, Apache Hadoop and Jupyter notebooks[9], all managed using the Apache Mesos resource manager. These are complemented with myriad tools and frameworks to allow data analytics applications to be developed in Python, Scala, R and Java.

The Alan Turing Institute's deployment of the Cray Urika-GX system (hereon called Urika) includes 12 compute nodes (each with 2x18 core Intel® Xeon® "Broadwell" CPUs), 256GB of memory and 60TB of storage (within a Lustre high-performance parallel file system) and 2 login nodes. Both compute and login nodes run the CentOS 7.4 operating system.

---

[1] https://spark.apache.org
[2] https://www.turing.ac.uk/
[3] https://www.ed.ac.uk/arts-humanities-soc-sci
[4] Rosa Filgueira and Mike Jackson (2018) "Analysing Humanities Data using Cray Urika-GX", Alan Turing Institute / EPCC, The University of Edinburgh, 31st July 2018, https://ati-rescomp-service-docs.readthedocs.io/en/latest/_downloads/ATI-SE-Humanities-Report.pdf
[5] Rosa Filgueira and Mike Jackson (2018) "Analysing humanities data using Cray Urika-GX", EPCC blog, 11 October 2018. https://www.epcc.ed.ac.uk/blog/2018/10/11/analysing-humanities-data-using-cray-urika-gx
[6] https://www.cray.com/products/analytics/urika-gx
[7] https://ati-rescomp-service-docs.readthedocs.io/en/latest/cray/introduction.html
[8] https://www.epcc.ed.ac.uk/facilities/other-facilities/cray-urika-gx
[9] http://jupyter.org

2. Datasets

The historical books and newspapers analysed range from the 16th to the early 20th centuries. The books and newspapers had been scanned via optical character recognition (OCR) into XML documents. The books and newspapers datasets that were explored were as follows.

Dataset	Period	Structure	XML Schema/Document Types	Scale	Space	Availability
British Library Books[10] (BLB)	1510-1899	One ZIP file per book with one XML metadata document per book and one XML document per page	XML metadata document: Metadata Encoding and Transmission Standard (METS)[11] XML page document: ALTO (Analysed Layout and Text Object)[12]	63700 ZIP files	~220GB	Available, under licence, from Gale[13], a division of CENGAGE[14]. Also available under an open, public domain, licence.
British Library Newspapers[15] (BLN)	1714-1950	One XML document per issue	Unclear[16]	179669 XML documents	~424GB	Available, under licence, from Gale, a division of CENGAGE.
Times Digital Archive[17] (TDA)	1785-2009	One XML document per issue	Unclear, but with significant overlap to the above	69699 XML documents	~362GB	Available, under licence, from Gale, a division of CENGAGE.
Papers Past New Zealand and Pacific newspapers[18] (NZPP)	1839-1863	One XML document for a set of 22 articles	Unspecified. Each XML document corresponds to results from a search via an API.	13411 XML documents	~4GB	Available, under licence, from Papers Past[19]. Data can be accessed via API calls which return search results in the form of XML documents.

The copies of the datasets used are hosted at the University of Edinburgh within the University's DataStore[20] provided by Information Services' Research Data Services. The authors were unaware of the processes by which the data was transferred from data providers to the University's DataStore, but the authors did discover that some of the datasets were incomplete. Parts 4 to 6 of the BLN

[10] https://data.bl.uk/digbks/
[11] http://www.loc.gov/standards/mets/
[12] https://www.loc.gov/standards/alto/
[13] https://www.gale.com
[14] https://www.cengage.com/
[15] https://www.gale.com/uk/s?query=british+library+newspapers
[16] These conform to an XML Schema for which no XSD or DTD files seem available. However, it is very similar to that documented in "Appendix E. Description of the DTD fields" of Jisc's "British Newspapers 1620 – 1900 Final Report", 19 August 2009, https://webarchive.nationalarchives.gov.uk/20140702175911/http://www.jisc.ac.uk/media/documents/programmes/digitisation/blfinal.pdf
[17] https://www.gale.com/intl/c/the-times-digital-archive
[18] http://paperspast.natlib.govt.nz/newspapers
[19] http://paperspast.natlib.govt.nz/
[20] https://www.ed.ac.uk/information-services/research-support/research-data-service/working-with-data/data-storage

were missing and there were only TDA files spanning 1785-1848. Melissa arranged for access to the full data sets which were delivered via hard drive, then copied, using rsync[21], into the DataStore.

The DataStore directories were mounted onto Urika using SSHFS[22] and were then copied into Urika's Lustre high-performance file system, via rsync which was also used to filter out any content that was not needed e.g. backup files, scratch files or raw image files which were present in the data directories. A local copy is necessary because, unlike Urika's login nodes, Urika's compute nodes have no network access and so cannot access the DataStore via SSHFS network mount points. Equally importantly, for efficient processing, data movement and network transfers need to be minimised.

As the authors are unaware of the processes by which the other data was accessed and then deposited into the DataStore, some of the datasets used during this work may still be incomplete. The above table reflects that actual numbers of files used within this work.

3. Historical books and newspapers analysis code

The work described in "Analysing Humanities Data using Cray Urika-GX" used two text analysis codes and a set of complementary Jupyter notebooks for visualisation and results analysis. The codes were initially developed by UCL with the British Library and further developed by EPCC as described in the original report. The codes and notebooks were as follows:

- i_newspaper_rods (epcc-master branch)[23]: Python code that uses Apache Spark to run queries over both BLN and TDA datasets. It provides an object model that represents newspapers in terms of issues and articles. XPath[24] queries are used to extract information from XML documents. These queries exploit the commonality between the XML Schema used in BLN and TDA with conditionals handling minor differences.
- cluster-code (sparkrods branch) [25]: Python code that uses the Apache Spark cluster computing framework to run queries over BLB datasets. It provides an object model that represents books in terms of ZIP archives, books and pages. XPath queries are used to extract information from XML documents. These queries assume that the XML documents are compliant with METS (for books) and ALTO (for pages).
- visualisations (epcc-master branch)[26]: Jupyter notebooks to visualise and analyse the outputs from cluster-code, plus sample query results.

While i_newspaper_rods and cluster-code share common code and behaviour they are implemented and are run in slightly different ways. They also include code to support the use of various HPC environments at UCL. It was proving challenging for the authors to continue to develop both these codes especially given there was no way for the authors to test the HPC-specific code. Consequently, i_newspaper_rods and cluster-code were merged and refactored into a single new code, "defoe"[27].

[21] https://rsync.samba.org/
[22] https://en.wikipedia.org/wiki/SSHFS
[23] https://github.com/alan-turing-institute/i_newspaper_rods/tree/epcc-master
[24] https://www.w3.org/TR/xpath/all/
[25] https://github.com/alan-turing-institute/cluster-code/tree/epcc-sparkrods
[26] https://github.com/alan-turing-institute/cluster-code-visualisations/tree/epcc-master
[27] https://github.com/alan-turing-institute/defoe

3.1. defoe

defoe allows for BLB, BLN and TDA datasets to be queried from a single command-line interface and in a consistent way. HPC environment-specific code was removed – defoe assumes that the code is executed via Apache Spark's "spark-submit"[28] command (additional HPC environment-specific configuration and deployment code can stored in other repositories). Improvements to usability were made so that the user specifies the data files to query, the object model to use, the query to use and any query-specific files via the command-line. Co-locating the object models for each type of dataset should make it easier, in future, to identify and extract out commonality across the object models, and also to develop queries which operate across any object model transparently to users (this is discussed further in section 5 below).

The merging and refactoring of i_newspaper_rods and cluster-code into defoe was done by the authors as part of the Living with Machines[29] project, a collaboration between the Alan Turing Institute, the British Library and a number of universities, and funded by the Arts and Humanities Research Council and UK Research & Innovation. Living with Machines seeks to understand the impact of technology across society during the Industrial Revolution by studying newspapers, journals, pamphlets, census data and other publications from that era.

3.2. NZPP object model

As part of the work with Melissa, defoe was extended with a new object model to support the NZPP dataset. The object model represents NZPP XML documents in terms of collections of articles and single articles within these. XPath queries are used to extract information from these XML documents.

3.3. defoe_visualization

To complement defoe, a new repository of Jupyter notebooks, defoe_visualization[30], was also developed. These allow researchers to explore the query results and also to post-processing on the results to extract information of use to them. The notebooks are complemented with YAML and comma-separated values (CSV) files with the query results produced by the authors.

4. Querying historical newspapers and books

Using defoe, three types of analysis were undertaken at Melissa's request. The analyses and results were as follows[31].

4.1. Reporting the Krakatoa eruption of 1883

Krakatoa (Krakatau in Indonesian)[32] erupted over 26-27th August 1883 and was one of the most spectacular volcanic eruptions of contemporary times. Melissa was interested in references to the eruption within contemporary newspapers in the year 1883. An existing query was run,

---

[28] https://spark.apache.org/docs/latest/submitting-applications.html
[29] https://www.turing.ac.uk/research/research-projects/living-machines
[30] https://github.com/alan-turing-institute/defoe_visualization
[31] The version of defoe used was https://github.com/alan-turing-institute/defoe/tree/59dbbd93db2a2db8bb72bd70bbd861345e0415be and the results are in defoe_visualisation version https://github.com/alan-turing-institute/defoe_visualization/tree/b6a3f9105fddee4259c15f41fbc55594e177d3fb
[32] https://en.wikipedia.org/wiki/Krakatoa

"keyword_and_concordance_by_date" [33], which searches for occurrences of any word in a list of keywords and returns information on each matching article including title, matching keyword, article text and filename. Results are grouped by the publication dates of the newspapers. This query was run with the keywords "krakatoa" and "krakatau" across BLN[34] and TDA[35]. Within 1883 itself, 62 articles referencing "Krakatoa" or "Krakatau" were identified within BLN and 17 within TDA.

This query motivated the development of an object model for NZPP. However, it was discovered during this work that the dataset predated 1883, spanning 1839-1863.

4.2. Origin of the term "stranger danger"

A social sciences colleague of Melissa's was interested in how the phrase "stranger danger" has been used over time and where and when it might have originated from.

A query was implemented, "colocates_by_year"[36], which searches for two words which are co-located and separated by a maximum number of intervening words. For each match, information about the matching book/issue, including the book/article title, the matching words, the intervening words, and the book/newspaper file name are returned. These results are grouped by the publication dates of the books/newspapers.

This query was run, with the words "stranger" and "danger" and a maximum number of intervening words of 12, across BLB[37].

4.3. Exploring female emigration

Melissa was also interested in exploring female emigration. To assist with this exploration two existing queries were modified:

- "target_and_keywords by_year" [38], which searches for occurrences of a target word occurring with any word in a list of keywords and returns counts of the number of articles which include the target word and a subset of the keywords. The results are grouped by year.
- "target_and_keywords_count_by_year" [39], which searches for occurrences of a target word (occurring with any word in a list of keywords and returns counts of occurrences of each target word and these keywords. The results are grouped by year.

---

[33] See defoe query modules defoe.papers.queries.keyword_concordance_by_date (for BLN and TDA) and defoe.nzpp.queries.keyword_concordance_by_date (for NZPP).
[34] CSV results: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Krakatoa_1883/results_krakatoa_blnewspapers_1883.csv
[35] CSV results: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Krakatoa_1883/results_tda_1883.csv
[36] See defoe query modules defoe.alto.queries.colocates_by_year (for BLB) and defoe.papers.queries.colocates_by_year (for BLN and TDA).
[37] Jupyter notebook: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Stranger_Danger/Stranger_Danger.ipynb
[38] See defoe query module defoe.papers.queries.target_and_keywords_by_year (for BLN and TDA).
[39] See defoe query module defoe.papers.queries.target_and_keywords_count_by_year (for BLN and TDA).

As an example of the difference between these two queries, “target_and_keywords by_year” could return:

```
1751:
- count: 1
  target_word: emigration
  words: [daughter, empire, indecent, marriage, mother]
- count: 1
  target_word: emigration
```

For the same keywords, target word and data, “target_and_keywords_count_by_year” would return:

```
1751:
- [indecent, 1]
- [mother, 2]
- [emigration, 3]
- [empire, 6]
- [marriage, 1]
- [daughter, 1]
```

In addition, two new queries were developed:

- “keysentence_by_year” [40], which searches for occurrences of a sentence (or phrase) and returns counts of the number of articles which include the sentence. The results are grouped by year.
- “target_concordance_collocation_by_date” [41], which searches for occurrences of a target word occurring with any word in a list of keywords and returns the keyword plus its concordance (the text surrounding the keyword, in this case the 5 words preceding and following the keyword). The filename in which the match occurs and the OCR quality is also returned. The results are grouped by year.

4.3.1. Normalization, stemming and lemmatization

“target_and_keywords by_year”, “target_and_keywords_count_by_year”, “keysentence_by_year” and “target_concordance_collocation_by_date” allow a user to specify the preprocessing that should be applied to words in documents and to words provided as part of the query (e.g. keywords). There are three types of preprocessing available: normalization, stemming and lemmatization.

Normalization removes all non-'a-z|A-Z' characters and making the words lower-case. All queries in defoe have been implemented to normalize words by default.

Stemming reduces normalized words to their word stems (for example, “books” becomes “book” or “looked” becomes “look”). The Python Natural Language Toolkit (NLTK)[42] implementation of the

[40] See defoe query module defoe.papers.queries.keysentence_by_year (for BLN and TDA).
[41] See defoe query module defoe.papers.queries.target_concordance_collocation_by_date (for BLN and TDA).
[42] https://www.nltk.org/

Porter stemming algorithm[43] is used, which removes common morphological and inflexional endings from words.

Lemmatization reduces inflectional forms of normalized words to a common base form. As opposed to stemming, lemmatization does not simply chop off inflections. Instead, lexical knowledge bases are to get the correct base forms of words. Lemmatization is done using the NLTK WordNet Lemmatizer[44].

To see the differences, here are the results of running the "target and keywords count by year" query over BLN and looking for occurrences of a target word, "emigration", co-located with keywords from a taxonomy of terms which includes the words "Colony", "Colonies" and "Colonial". Using normalization only, the occurrences of these three words are, for 1901:

```
- [colonial, 16]
- [colonies, 28]
- [colony, 16]
```

Using normalization and stemming:

```
- [coloni, 105]
```

Using normalization and lemmatization:

```
- [colonial, 16]
- [colony, 45]
```

At Melissa's request, both normalization and lemmatization were applied.

4.3.2. Normalised frequencies of the names of specific female emigration societies

"keysentence_by_year" was used to search for references to female emigration societies[45] (e.g. "The East End Emigration Fund" or "The South African Colonisation Society"). The query was run over both BLN[46] and TDA[47].

[43] https://www.nltk.org//api/nltk.stem.html#module-nltk.stem.porter
[44] https://www.nltk.org/api/nltk.stem.html#module-nltk.stem.wordnet
[45] For a full list, see defoe's queries/emigration_societies.txt.
[46] Raw results: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/BLN_Parts1-6/results_BLN/results_bln_society_1850_1914.txt, Jupyter notebook: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/BLN_Parts1-6/Visualization_Frequency_Societies_BLN.ipynb
[47] Raw results: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/TDA/results_TDA/results_tda_society_1850_1914, Jupyter notebook: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/TDA/Visualization_Frequency_Societies_TDA.ipynb

4.3.3. Normalised frequencies of female emigration taxonomy terms co-located with the word "emigration"

"target_and_keywords by_year" and "target_and_keywords_count_by_year" were used to search BLN and TDA for occurrences of a target word, "emigration", co-located with keywords from a taxonomy of terms relating to "female emigration" [48] (e.g. "colony", "service", or "female", "governess"). The query was run over both BLN and TDA, and results can be found at BLN_Results[49] and TDA[50] and N-grams and Jupyter notebooks at BLN[51] and TDA[52].

4.3.4. Concordance and collocation analysis of the term "emigration"

"target_concordance_collocation_by_date" was used to search for occurrences of a target word, "emigration". It was also used to search BLN and TDA for references to female emigration societies. The query was run over both BLN and TDA datasets, and results can be found at BLN_Results[53] and TDA[54].

4.3.5. Concordance and collocation analysis of female emigration taxonomy terms co-located with the word "emigration"

"target_concordance_collocation_by_date" was used to search for occurrences of a target word, "emigration", co-located with keywords from the taxonomy of terms relating to "female emigration". It was also used to search BLN and TDA for references to female emigration societies. The query was run over both BLN and TDA, and results can be found at BLN_Results[55], and TDA_Results[56].

5. Future work

Future work on analysing historical newspapers and books and on defoe include the following.

5.1. Run Spark and defoe on EDDIE

Not all researchers have access to Urika. It is useful to enable researchers to defoe on other HPC environments upon which, unlike Urika, Spark has not been pre-installed and configured. As part of

[48] For a full list, see defoe's queries/emigration_taxonomy.txt.
[49] CSV results: https://drive.google.com/file/d/1bUB2TWZcP8kGiyOM6UI5V5-QvU-4f-vN/view
[50] CSV results: https://drive.google.com/file/d/1ewliFhAIUqaNHEkeWkugYalrBKYnBCN4/view
[51] Raw results: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/BLN_Parts1-6/results_BLN/results_bln_ngram_1850_1914, Jupyter notebook: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/BLN_Parts1-6/Visualization_Frequency_Taxonomy_Ngrams_BLN.ipynb
[52] Raw results: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/TDA/results_TDA/results_tda_ngram_1850_1914, Jupyter notebook: https://github.com/alan-turing-institute/defoe_visualization/blob/b6a3f9105fddee4259c15f41fbc55594e177d3fb/Female_Emigration/TDA/Visualization_Frequency_Taxonomy_Ngrams_TDA.ipynb
[53] CSV results: https://drive.google.com/file/d/1bUB2TWZcP8kGiyOM6UI5V5-QvU-4f-vN/view
[54] CSV results: https://drive.google.com/file/d/1ewliFhAIUqaNHEkeWkugYalrBKYnBCN4/view
[55] TXT results: https://drive.google.com/file/d/1s_xRAsCm8Tp8KS8-9kKdu6_7bl9Wljov/view?usp=sharing
[56] CSV results: https://drive.google.com/file/d/1fhsLJQ_nMiZjuUaDWjIQYXefJqF9B7j_/view

the BioExcel project[57], Rosa developed scripts[58] to deploy and configure a Spark cluster on-demand as part of a traditional batch job submitted. These were tested on the Cirrus[59] HPC service.

As part of an internal project at the University of Edinbugh, Rosa is working with Melissa and David Fergusson of University of Edinburgh's Information Services' Research Data Services to customise her scripts to run on the University's Linux Computing Cluster (EDDIE) [60]. This will allow defoe analyses to be run by researchers who do not have access to Urika.

5.2. Redesign object models

Owing to its origins as two separate codes, defoe now has a number of object models[61]:

- Issues and articles. BLN and TDA datasets are parsed into this model.
- ZIP archives, documents (compliant with METS) and pages (compliant with ALTO). BLB datasets are parsed into this model. Living with Machines uses a newspapers dataset which is also represented as METS XML documents (one per newspaper) and ALTO XML documents (one per newspaper page). These datasets are also parsed into this model. An abstract model represents the ZIP archives, documents, and pages and there are concrete submodels for BLB and the newspapers dataset in use by Living with Machines. The two submodels handle differences in how XML documents are arranged and named within ZIP archives.
- Collections of articles and single articles: NZPP datasets are parsed into this model.

There are two problems with this design:

- The object models, and the associated query code, are based upon the physical representation of the data. For example, BLN, NZPP and newspapers data used within Living with Machines all contain newspapers, consisting of issues and articles, but the object model used is issues/articles, sets of articles/articles, and issues/pages respectively.
- Python code implementing queries are also object model-specific. So, for example there are three queries to count the number of words, one for each object model.

It would be very useful to introduce conceptual object models (e.g. books/pages, issues/articles) and to introduce a layer to parse the physical representations of the datasets into these conceptual object model. For example, newspapers from BLN, NZPP and the newspapers dataset used within Living with Machines would all be mapped to a conceptual model of issues/articles. This would allow query code to be specific to conceptual models (e.g. "count number of articles"). A base model could represent information common to all documents (e.g. books, newspapers etc) with a complementary set of basic queries (e.g. "count number of words").

The ease of mapping from the physical representation to the conceptual object model depends upon the physical representation. For example, for BLN it is straightforward, since each XML document represents a single issue and each article is marked-up within that XML document. In NZPP, each XML document contains one or more articles, but articles for a single issue may be spread across

[57] https://bioexcel.eu/
[58] Rosa Filgueira (2019) "Spark-based genome analysis on Cray-Urika and Cirrus clusters", EPCC blog, 16 Jan 2019, https://www.epcc.ed.ac.uk/blog/2019/spark-based-genome-analysis-cray-urika-cirrus-clusters
[59] http://www.cirrus.ac.uk/
[60] https://www.ed.ac.uk/information-services/research-support/research-computing/ecdf/high-performance-computing
[61] For more detail, see defoe's design and implementation notes, https://github.com/alan-turing-institute/defoe/blob/master/docs/design-implementation.md

multiple XML files – to construct issues requires parsing multiple XML documents and reconstructing each issue from its articles, using newspaper names and publication dates/times as the linking criteria. For the newspaper dataset used within Living with Machines, this would require parsing the METS metadata document and, from this, identifying which articles are present and parts of what ALTO page documents contain the text for each article (as an article may span a number of pages) and then parsing these page documents in turn.

Development of these conceptual object models will be done as part of Living with Machines.