defoe: A Spark-based Toolbox for Analysing Digital Historical Textual Data

Rosa Filgueira Michael Jackson Anna Roubickova Amrey Krause
EPCC, University of Edinburgh
Edinburgh, UK
{r.filgueira, m.jackson, a.roubickova, a.krause} @epcc.ed.ac.uk

Melissa Terras
CAHSS, University of Edinburgh
Edinburgh, UK
m.terras@ed.ac.uk

Tessa Hauswedell Julianne Nyhan
SELCS, University College London
London, UK
{t.hauswedell, j.nyhan}@ucl.ac.uk

David Beavan Timothy Hobson Mariona Coll Ardanuy
The Alan Turing Institute
London, UK
{dbeavan, thobson, mcollardanuy}@turing.ac.uk

Giovanni Colavizza James Hetherington
The Alan Turing Institute
London, UK
gcolavizza, jhetherington@turing.ac.uk

Ruth Ahnert
Queen Mary University of London and Alan Turing Institute
London, UK
r.r.ahnert@qmul.ac.uk

***Abstract*—This work presents *defoe*, a new scalable and portable digital eScience toolbox that enables historical research. It allows for running text mining queries across large datasets, such as historical newspapers and books in parallel via *Apache Spark*. It handles queries against collections that comprise several XML schemas and physical representations. The proposed tool has been successfully evaluated using five different large-scale historical text datasets and two HPC environments, as well as on desktops. Results shows that *defoe* allows researchers to query multiple datasets in parallel from a single command-line interface and in a consistent way, without any HPC environment-specific requirements.**

***Index Terms*—text mining, distributed queries, Apache Spark, High-Performance Computing, XML schemas, digital tools, digitised primary historical sources, humanities research**

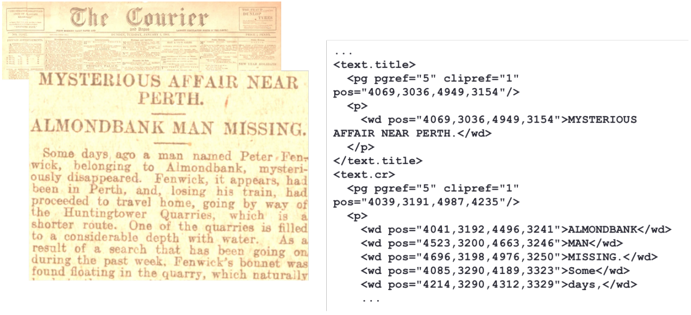
I. Introduction

Over the past three decades, large scale digitisation has been transforming the collections of libraries, archives, and museums [1], [2]. The volume and quality of available digitised text now makes searching and linking these data feasible, where previous attempts were restricted due to limited data availability, quality, and lack of shared infrastructures [3].

There is hunger for large scale text mining facilities from the humanities community, with commercial providers allowing limited access to their own digitised collections [4]. However, there are barriers to querying the wealth of newspapers and books that now exist in digitised, openly licensed form at scale, which would allow humanists to be in control of their text mining research [5].

For example, although the product of most digitisation of historical text is structured XML[1] files derived from Optical Character Recognition (OCR) (see Figure 1), the schemas, structure, and size of the datasets are heterogeneous, and they are often difficult to link and cross-query. Additionally, the humanities community has limited capacity and/or skills to use High-Performance Computing (HPC) environments and analytic frameworks to create applications to mine large-scale digital collections effectively.

This work focuses on removing some these obstacles by enabling complex analysis of digital datasets at scale. By bringing together computer scientists with humanities and computational linguistics researchers, we have created a new



The Courier

MYSTERIOUS AFFAIR NEAR PERTH.

ALMONDBANK MAN MISSING.

Some days ago a man named Peter Fenwick, belonging to Almondbank, mysteriously disappeared. Fenwick, it appears, had been in Perth, and, losing his train, had proceeded to travel home, going by way of the Huntingtower Quarries, which is a shorter route. One of the quarries is filled to a considerable depth with water. As a result of a search that has been going on during the past week, Fenwick's bonnet was found floating in the quarry, which naturally

```
...
<text.title>
  <pg pgref="5" clipref="1"
pos="4069,3036,4949,3154"/>
  <p>
    <wd pos="4069,3036,4949,3154">MYSTERIOUS
AFFAIR NEAR PERTH.</wd>
  </p>
</text.title>
<text.cr>
  <pg pgref="5" clipref="1"
pos="4039,3191,4987,4235"/>
  <p>
    <wd pos="4041,3192,4496,3241">ALMONDBANK</wd>
    <wd pos="4523,3200,4663,3246">MAN</wd>
    <wd pos="4696,3198,4976,3250">MISSING.</wd>
    <wd pos="4085,3290,4189,3323">Some</wd>
    <wd pos="4214,3290,4312,3329">days,</wd>
    ...
```

Fig. 1: Digitisation example of the first *The Courier and Angus* newspaper issue of 1901.

[1] XML has been commonly adopted by the cultural heritage and digital humanities community for the delivery of large scale datasets, commonly with some relationship to the Text Encoding Initiatives guidelines for structuring such texts: https://tei-c.org.

portable humanities research toolbox, called *defoe*[2] for interrogating large and heterogeneous text-based archives. *defoe* uses the power of analytic frameworks, such as Apache Spark [6], Jupyter Notebooks, and HPC environments to manipulate and mine huge digitised archives in parallel at great speed using a simple command line. It can be installed and used in different computing environments (cloud systems, HPC clusters, laptops), making humanities research portable and reproducible.

We have demonstrated the feasibility of *defoe* by using two case studies as undertaken at the University of Edinburgh (*Eruption of Krakatoa Volcano in 1883* and *Female Emigration*), two HPC clusters (*Cray Urika* and *Eddie*), and five different digital collections.

This paper is structured as follows. Section II presents background. Section III discusses *defoe* design features. Section IV presents two case studies for testing feasibility of the tool. Section V describes the computing environments used for our studies. We conclude with a summary of achievements and outline future work.

II. Related work

The work presented in this paper builds on two previous Python text analysis packages, *cluster-code*[3] and *i_newspaper_rods*[4], created by members of the author team in collaboration with the Research IT Services at University College London (UCL) [5]

cluster-code [7] analyses British Library (BL) books conforming to a particular XML schema, while *i_newspaper_rods* has been designed for analysing British Library newspapers conforming to another XML schema. XML schemas describe either the structure of a digital object or the actual textual content of the object including the word content, styles, and layout elements. However, the two XML schemas used by both packages are slightly different in terms of metadata attributes, tags, and document structure.

Even though *cluster-code* and *i_newspaper_rods* share a common behaviour, they are implemented and run in different ways. *cluster-code* uses *mpi4py* [8] (a wrapper to MPI) to query data, while *i_newspaper_rods* uses the *Apache Spark* framework. Each defines a single object model based on the physical representation and XML schemas of the data that each has been designed to ingest. Furthermore, each was originally designed to extract data held within a UCL deployment of the data management software, and run queries on UCLs HPC services. This means, that we can not use *cluster-code* for analysing BL newspapers or *i_newspapers* for British Library books, and we have to make several modifications to both codebases to run them in different HPC environments.

Among the range of text mining queries supported by *cluster-code* we can count the total number of pages across all books, count the frequencies of a given list of words, find the locations of figures, etc. Some examples of *i_newspapers_rods* queries are counting the number of articles per year, the frequencies of words in a given list of words, and finding expressions that match a given pattern.

In terms of preprocessing the raw text data, both libraries only include a basic normalisation strategy by removing non-'a-z—A-Z' characters and converting all text to lower case.

With *defoe* we aimed to create a portable tool for analysing historical collections that conform to either of several XML schemas and different physical representations, expanding the range of supported queries, and including natural language processing capabilities. All of these functionalities should be provided using Apache Spark as the distributed and parallel engine. More details about *defoe* are given in the next section.

III. The defoe Digital Humanities Tool

defoe is the result of merging, extending and refactoring codebases introduced in Sec. II in the same place, allowing queries from a single command-line interface in a consistent way. The main component of *defoe* is the text analysis pipeline (see Figure 2), which resides at its core and has been implemented using *Apache Spark*.

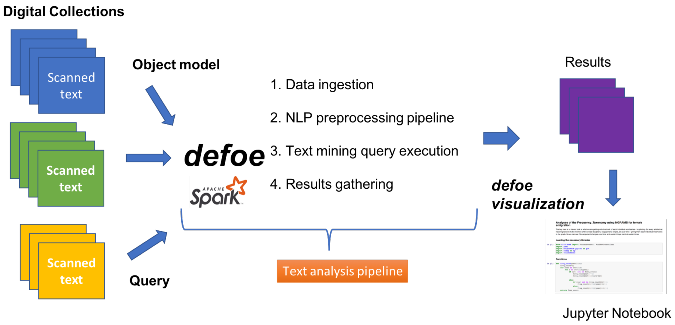


Fig. 2: *defoe* overview.

Apache Spark is an open source HPC distributed computing framework for large-scale data processing. It uses a directed acyclic graphs (DAG) that allow for processing multi-stage pipelines chained in one job. Apache Spark provides an ability to cache large datasets in memory between stages of the calculation and to reuse intermediate results of the computation in iterative algorithms. *Resilient Distributed Datasets (RDD)* are the fundamental data structures of Apache Spark, which is an immutable distributed collection of objects. RDDs are partitioned into logical partitions across cluster nodes and operate in parallel through *transformations* and *actions*. A *transformation* is a function that produces new RDD from existing ones. Transformations are not executed immediately, only after an action is called. On the other hand, action are operations that trigger execution of transformation and return values.

Therefore, we chose Apache Spark because it provides automatic parallelisation and scalability for the four steps of our text analysis pipeline. It loads digital collections into RDDs in

[2] Daniel Defoe was both a journalist and a novelist, which are the two datasets that this toolkit work with. Therefore, we decided to name it after him. *defoe* source code available at https://github.com/alan-turing-institute/defoe

[3] https://github.com/UCL-dataspring/cluster-code

[4] https://github.com/UCL/i_newspaper_rods

[5] https://www.ucl.ac.uk/isd/services/research-it-services.

memory, cleans data and runs text mining queries in parallel, returning results as YAML files. To complement *defoe*, a repository of Jupyter notebooks, called *defoe visualisation*, has been developed to allow researchers to visualise and explore further results obtained with *defoe*.

Furthermore, a lot of work has gone into *defoe* to remove all HPC environment-specific requirements, allowing us to run *defoe* in any computing environment that has Apache Spark installed. The following subsections give a detailed explanation of each step of the *defoe* text analysis pipeline.

A. Data Ingestion

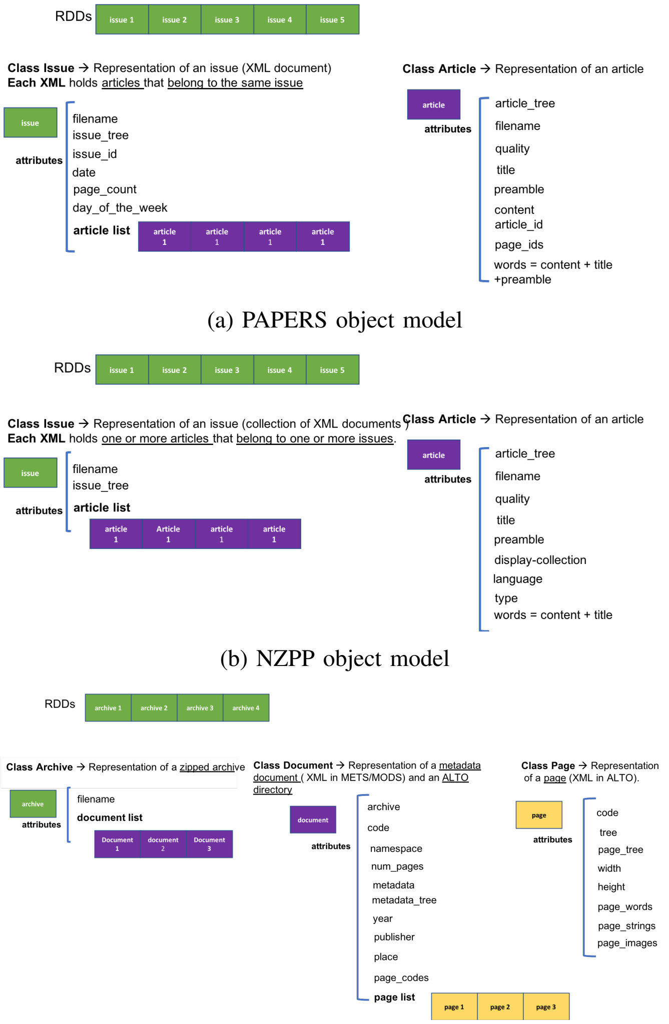
We are able to able to consume historical textual collections scanned via OCR into XML documents using *defoe*. It has specific support for newspapers and books conforming any of these three physical representations: 1) one XML document per issue; 2) one XML document with search results including several articles; and 3) one XML metadata document and XML page. It also supports the following digital library standards:

- METS XML schema[6] for descriptive, structural, technical and administrative metadata;
- MODS XML[7] for descriptive and bibliographic metadata;
- ALTO XML schema[8] for encoding OCR text
- British Library-specific XML schema[9] for OCR text; and
- PaperPast-specific XML schema [10] for encoding OCR text.

As we mentioned before, XML schemas describe either the structure of a digital document or the actual textual content. All of these are slightly different, and none of them is universally used. Therefore, we have created three object models (*PAPERS*, *NZPP* and *ALTO*) to map the physical representations and XMLs schemas (see Figure 3) mentioned before. These allows *defoe* to ingest digital collections into Spark RDDs, which is the only requirement to select the appropriate object model the digital collection to analyse is mapped onto.

For example, if we want to import a historical newspaper collection to be analysed, first we have to check whether the collection has been digitised using a document per issue and the British Library-specific XML schema, in which case the *PAPERS* object model needs to be selected. If, on the other hand, each XML document follows the PaperPast-specific XML schema and holds a collection of articles that corresponds to one more issues, the *NZPP* object should be selected instead. In the case of analysing ALTO archives, the *ALTO* model should be selected.

We have also explored the option of ingesting XML documents using *Spark SQL* and DataFrames. Unlike an RDD, DataFrames organise data into named columns, and impose a structure on a distributed collection of data. To use this option, we also need to infer the different XML schemas. Therefore, we developed two new python parsers[11] based on the *databricks spark-xml* package[12], which has been designed to read XML documents as DataFrames.



(a) PAPERS object model

(b) NZPP object model

(c) ALTO object model

Fig. 3: Object models available in *defoe*

However, due to the nested nature of the historical XML schemas (such as ALTO, METS and British-Library specific XML), the *spark-xml* package is not able to infer them automatically, requiring a lot of manual work to flatten their attributes. Since this option did not add further benefits to what we already had, we decided to continue with our original implementation using RDDs.

Once the data is loaded into RDDs with a particular object model, *defoe* continues the analysis with the Natural Language Processing (NLP) step for cleaning the text. Note that Apache Spark ingests the data in parallel which is automatically partitioned (in memory) across the Spark cluster.

To evaluate *defoe*, we have used a range of digital collections, which main features are summarised in Table I.

[6] http://www.loc.gov/standards/mets/

[7] http://www.loc.gov/standards/mods/

[8] https://www.loc.gov/standards/alto/

[9] http://www.jisc.ac.uk/media/documents/programmes/digitisation/blfinal.pdf

[10] https://paperspast.natlib.govt.nz/about

[11] https://github.com/rosafilgueira/SparkSQL_DataFrames_Defoe

[12] https://github.com/databricks/spark-xml

Dataset	Period	Structure	Scale	Size	Object Model
British Library Books (BLB)[13]	1510-1899	ZIP per book with a XML metadata (METS schema) and a XML per page (ALTO schema)	63700 ZIP files	220 GB	ALTO model
British Library Newspapers (BLN)[14]	1714-1950	XML document per issue (British Library-specific XML schema).	179669 XML documents	424 GB	PAPERS model
Times Digital Archive (TDA)[15]	1785-1848	XML (British Library-specific XML schema) per issue	69699 XML documents	362 GB	PAPERS model
Papers Past New Zealand and Pacific newspapers (NZPP)[16]	1839-1863	XML document (with 22 articles) corresponds to results from a search via an API)	13411 XML documents (PaperPast-specific XML schema).	4 GB	NZPP model
Find My Past (FPM)	1752 - 1957	Folder per issue, with a XML metadata document (METS) and XML file per issue page (ALTO)	2067235 XML documents	1.8TB	ALTO model

TABLE I: Main features, abbreviations and *defoe* object models for the digital collections explored.

B. *NLP Text Preprocessing*

The poor quality that often emerges from large-scale text digitisation is well-documented [9], [10]. This can create issues for searching and research based on imperfect digital copies of text. Basic string-matching searching is unlikely to return all the results intended by the user. We have implemented a NLP preprocessing pipeline (see Figure 4) that deals with character-level errors in the OCR, and which is a first step towards allowing a more semantically meaningful exploration of the data. Our NLP pipeline consists in the following steps. After the raw text (from pages or articles) is ingested, the first step is to split it into sentences. Then, each sentence is tokenised separately, which results in a new sequence of tokens that roughly correspond to 'words'. Later each token is normalised by removing non-'a-z—A-Z' characters and making them lower-case. Once tokens are normalised, users can choose whether to apply stemming or lemmatisation. Stemming reduces words to their word stem (base or root form), whereas lemmatisation reduces inflectional forms to a common base form.

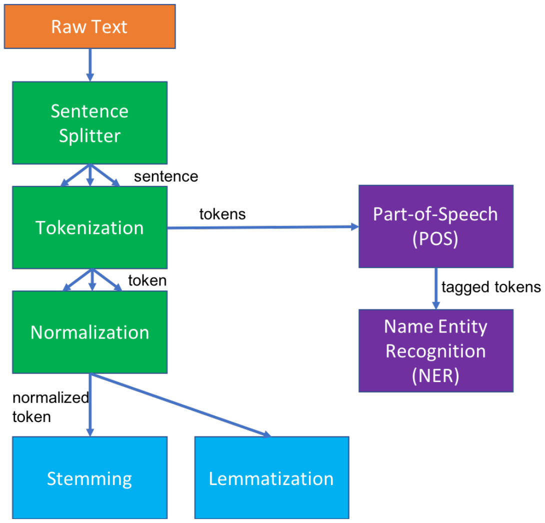


Fig. 4: *defoe* NLP preprocessing pipeline implemented.

After that, the pipeline performs *Part-Of-Speech* (POS) tagging, which assigns parts of speech to each token (tagging them as nouns, verbs, adjectives, and others) based on its definition and context. Finally, the pipeline proceeds with the *named-entity recognition* (NER) step to find named entities in the tagged tokens and classify them into pre-defined categories (names of persons, locations, organisations, times, etc).

Since the NLP pipeline is implemented in Apache Spark, each step is a transformation, in which a function (e.g. sentence splitter, tokenisation, normalisation, etc) produces new RDDs from the existing ones.

Furthermore, we have implemented two versions of this pipeline using widely used NLP frameworks: *NLTK* [11] and *spaCy* [12] (see Listing 5). *spaCy* is an open-source software library for advanced NLP written in Python and Cython. *NLTK* is a suite of libraries and programs for symbolic and statistical NLP for English written in Python. Their main features are shown in Table II.

Feature	***spaCy***	***NLTK***
Easy installation	Y	Y
Python API	Y	Y
Multi Language support	N	Y
Tokenization	Y	Y
Part-Of-Speech tagging	Y	Y
Sentence segmentation	Y	Y
Dependency parsing	Y	N
Name Entity Recognition	Y	Y
Integrated word vectors	Y	N
Sentiment analysis	Y	Y
Conference resolution	N	N

TABLE II: Comparison of *spaCy* and *NLTK* features.

We are currently evaluating end-to-end performance by comparing the outputs of the two versions of the pipeline

[13] https://data.bl.uk/digbks/

[14] https://www.gale.com/uk/s?query=british+library+newspapers

[15] https://www.gale.com/intl/c/the-times-digital-archive

[16] http://paperspast.natlib.govt.nz/newspapers

against a gold standard dataset of historical texts that is being manually annotated. This will inform which library is most suitable for preprocessing historical newspapers and books. At the time of writing, both implementations are available in *defoe*.

```
Sentence: "And devoted some time to social work in London."

|spaCy preprocessing|
 Word    Normaliz. Lemma      PoS     Tag   NER
 And     and       and       CCONJ    CC
 devoted devoted   devote    VERB     VBN
 some    some      some      DET      DT
 time    time      time      NOUN     NN
 to      to        to        ADP      IN
 social  social    social    ADJ      JJ
 work    work      work      NOUN     NN
 in      in        in        ADP      IN
 London  london    London    PROPN    NNP   GPE
 .                 .         PUNCT    .
-----------------------------------------------------------
|NLTK preprocessing|
 Word    Normaliz. Lemma   Stem    PoSTag     NER
 And     and       and     and     CC      (S And/CC)
 devoted devoted   devoted devot   VBN     (S devoted/VBN)
 some    some      some    some    DT      (S some/DT)
 time    time      time    time    NN      (S (NP time/NN))
 to      to        to      to      TO      (S to/TO)
 social  social    social  social  JJ      (S social/JJ)
 work    work      work    work    NN      (S (NP work/NN))
 in      in        in      in      IN      (S in/IN)
 London  london    london  london  NNP  (S(GPE London/NNP))
.                  .       .       .       (S ./.)
```

Fig. 5: *defoe* preprocessing output example. The sentence belongs to the FMP digital corpus.

C. Text mining queries

After preprocessing the raw text data, *defoe* executes the selected text mining query over clean RDDs. It offers users a wide range of queries, some of which were directly imported from *i_newspapers_rods* (such as `total_issues`, `total_articles` or `normalize`). Others were imported from `cluster-code` (such as `total_pages` or `total_books`), but these need to be re-implemented first in Apache Spark, since they were written in *mpi4py*. We also augmented these query types with a number of new ones, such as:

- `keyword_and_concordance_by_date`: 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 dates.
- `collocates_by_year`: Searches for two co-located words 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. Results are grouped by dates.
- `keysentence_by_year`: Searches for occurrences of a sentence (or phrase) and returns counts of the number of articles that include the sentence. Results are grouped by year.
- `target_and_keywords_by_year`: 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. Results are grouped by year.
- `target_and_keywords_count_by_year`: 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. Results are grouped by year.
- `target_concordance_collocation_by_date`: 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). The filename in which the match occurs and the OCR quality is also returned. Results are grouped by dates.

All *defoe* queries are based on a number of transformations (e.g. `filter`, `flatMap`, `reduceByKey`, etc.) and actions (e.g. `reduce`, `collect`, etc.) that are combined to perform text mining analyses. Listing 6 shows as an implementation example, the `ocr_quality_by_year` query, in which a `flatMap transformation` is applied to issues RDDs. For each article stored inside a given issue RDD, the transformation extracts the information from the `quality` and `years` attributes (check these on the PAPERS object model shown in Figure 3.a) and returns it as a new qualities RDD. Later the `reduceByKey` transformation is applied to group the qualities by years. Finally the *collect action* is performed to trigger the execution of previous transformations and to gather the result in a YAML file.

```
def do_query(issues, config_file=None, logger=None):
    # [(year, [quality]), ...]
    qualities = issues.flatMap(
        lambda issue: [(issue.date.year, [article.quality])\
        for article in issue.articles])
    result = qualities \
        .reduceByKey(concat) \
        .collect()
    return result
```

Fig. 6: *defoe* `ocr_quality_by_year query`: Gets the information on the OCR accuracy of each article and groups the results by year.

Apache Spark excels at distributing these transformations and actions across a cluster while abstracting away many of the underlying implementation details, and runs them in parallel.

D. Results

As we mentioned before, *defoe* gathers results into YAML files. Listing 7 shows the result obtained by the `ocr_quality_by_year query` using a small subset of BLN digital corpus. For simplification, only OCR qualities from 1850 and 1851 articles are displayed.

Furthermore, a new repository of Jupyter Notebooks, called *defoe visualisation*[17], has been developed to visualise and evaluate further results/answers obtained from *defoe*. For example,

[17] available at https://github.com/alan-turing-institute/defoe_visualization

```
1850: [72.5, 85.97, 88.51, 85.41, 52.11, 82.01 85.97,
       75.51, 86.28, 72.13, 75.55, 74.77, 80.05, 73.14,
       88.96, 89.3, 92.01, 93.42, 91.3, 89.21, 86.29,
       84.86, 80.41, 76.07, 80.41, 88.11, 90.6, 75.07,
       81.62, 65.82, 85.61, 89.84, 76.15, 82.65, 84.63]
 1851: [86.51, 82.82, 82.86, 82.98, 76.74, 82.59, 86.79,
        88.26, 87.61, 57.51, 79.4, 68.52, 85.11, 90.07,
        93.29, 85.73, 76.71, 78.2, 89.42, 92.78, 92.67,
        88.92, 92.76, 89.41, 90.57, 92.77, 88.7, 85.43]
```

Fig. 7: *defoe* `ocr_quality_by_year query` results using a subset of BLN digital corpus

we have developed a Jupyter Notebook to analyse the results of the `colocates_by_year` query (described at Sec. III-C, that examines the presence and rate of occurrence of the ‘Stranger Danger’ phrase over the time using the BLB digital corpus. With the developed notebook we were able to compare results, plot them by year, normalise the results to account for increased used of the phrase over time (see Figure 8) and perform sentiment analyses, or visualise which words appear more often near the ‘Stranger Danger’ phrase (see Figure 9).

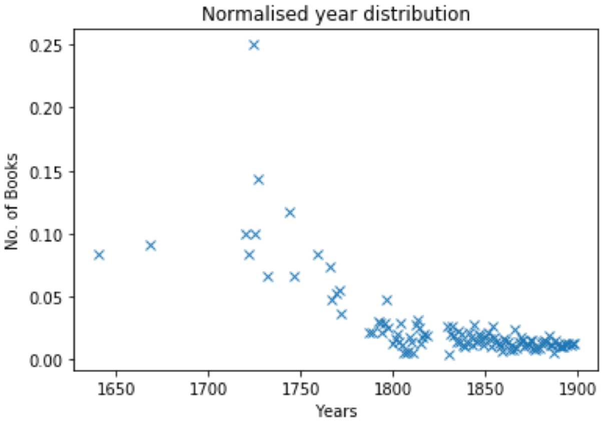


Fig. 8: Examination of the total number of BL books published to understand how much ‘Stranger Danger’ phrase is affected by the way that the number of BL books published increases over the measurement period

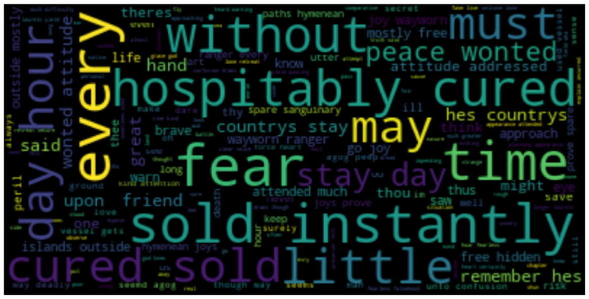


Fig. 9: Exploration of word frequency near ‘Stranger Danger’ using the BLB corpus

IV. Case Studies

There are two common ways that scholars in the humanities use text mining. One is to visualise overall patterns and trends in text. The other is to use the search facility to return a subcorpus (such as whole newspaper articles that include given terms), which can then be subject to further computational analysis as well as manual closer analysis. *defoe* supports both types of use, which we describe further with two different case studies below.

A. Female Emigration

The first case study is part of the Oceanic Exchanges[18] project and aims to mine the *TDA* and *BLN* archives for attitudes towards female emigration from Great Britain to the ‘Colonies’ and North America from 1850 to 1914. It seeks to understand the role that newspapers may have played as an agent of empire, asking the following question: *Can we detect an emphasis on a ‘service of the British empire’ narrative as a motivating factor and driver in the pursuit of female emigration?*.

Therefore, three text mining questions were designed to move from a purely quantitative, birds-eye view into a more semantic analysis of the topic of female emigration:

- Normalised frequencies of the names of female emigration societies[19], for which we used `keysentence_by_year` and `normalise` *defoe* queries (described in Sec. III-C).
- Normalised frequencies of specially chosen taxonomy terms[20] relating to female emigration, for which we used `target_and_keywords_by_year`, `target_and_keywords_count_by_year`, and `normalize` *defoe* queries (described in Sec. III-C).
- Analysing the concordance and collocation of the taxonomy terms , for which we used `target_concordance_collocation_by_date` *defoe* query (described in Sec. III-C).

defoe was not only used for running these queries in parallel using TDA and BLN newspapers, but also for aggregating, transforming results into CVS files (see Figure 10), post-processing analyses, and visualising results. Analysis of the results continues, but we can demonstrate the ability of *defoe* to create n-grams to visualise the occurrences of the words “daughter”, “engagement”, and “empire” across the corpus, as shown in Figure 11.

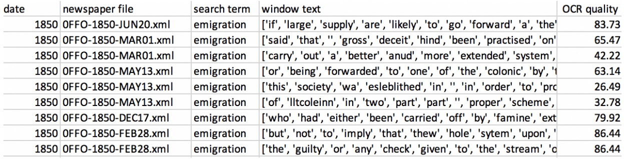
B. Eruption of Krakatoa Volcano in 1883

Krakatoa (Krakatau in Indonesian) erupted over 26-27th August 1883 and was one of the most spectacular volcanic eruptions in contemporary times. *defoe* was used to identify every mention of this eruption using the BLN, TDA, and NZPP

[18] https://oceanicexchanges.org

[19] List of the emigration societies available at https://github.com/alan-turing-institute/defoe/blob/master/queries/emigration_societies.txt

[20] The taxonomy terms derived from close reading of relevant newspaper, journal and periodical archives, as well as pamphlets on the topic of emigration from the time. These are available at https://github.com/alan-turing-institute/defoe/blob/master/queries/emigration_taxonomy.txt



date	newspaper file	search term	window text	OCR quality
1850	0FFO-1850-JUN20.xml	emigration	['if', 'large', 'supply', 'are', 'likely', 'to', 'go', 'forward', 'a', 'the'	83.73
1850	0FFO-1850-MAR01.xml	emigration	['said', 'that', '', 'gross', 'deceit', 'hind', 'been', 'practised', 'on'	65.47
1850	0FFO-1850-MAR01.xml	emigration	['carry', 'out', 'a', 'better', 'anud', 'more', 'extended', 'system',	42.22
1850	0FFO-1850-MAY13.xml	emigration	['or', 'being', 'forwarded', 'to', 'one', 'of', 'the', 'colonic', 'by', 't	63.14
1850	0FFO-1850-MAY13.xml	emigration	['this', 'society', 'wa', 'esleblithed', 'in', '', 'in', 'order', 'to', 'prc	26.49
1850	0FFO-1850-MAY13.xml	emigration	['of', 'lltcoleinn', 'in', 'two', 'part', 'part', '', 'proper', 'scheme',	32.78
1850	0FFO-1850-DEC17.xml	emigration	['who', 'had', 'either', 'been', 'carried', 'off', 'by', 'famine', 'ext	79.92
1850	0FFO-1850-FEB28.xml	emigration	['but', 'not', 'to', 'imply', 'that', 'thew', 'hole', 'sytem', 'upon', '	86.44
1850	0FFO-1850-FEB28.xml	emigration	['the', 'guilty', 'or', 'any', 'check', 'given', 'to', 'the', 'stream', 'c	86.44

Fig. 10: Snapshot of the taxonomy terms concordance and collocation results, using a window of 10 words preceding and following each match across the TDA corpus. OCR quality for each article is also captured.

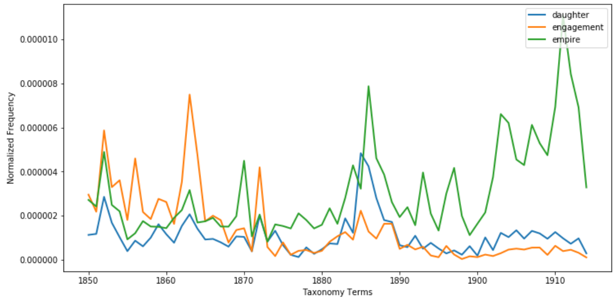


Fig. 11: N-gram for the normalised frequencies of the words “daughter”, “engagement”, and “empire” across the TDA corpus.

newspaper corpus from late 1883 in order to contribute to an international analysis on how the story was reported around the world, looking at news dissemination in the late Victorian period.

For this analysis, we used the `keyword_and_concordance_by_date` query described at Sec. III-C, which searches for occurrences of “krakatoa” and “krakatua” and returns information on each matching article.

The resulting small corpus of approximately 50 newspaper articles from all over England provides a rich data set where we can track copying and transmission of text, and this is being analysed at the time of writing by historians, and visualised in more advanced ways by data scientists. Further results are available in the Jupyter Notebook repository [21]

As we mentioned before, *defoe* allows for querying datasets from a single command-line interface and in a consistent way. It hides all the complexity to users, allowing to query data just specifying the corpus, object model, and query to use via the command line (see Listing 12).

V. Computing Environments

We have run the previous case studies using *defoe* on two HPC clusters, *Urika-GX* and *Eddie*, both hosted at the University of Edinburgh. We briefly descibe these two systems.

A. The Cray Urika-GX system

The Cray *Urika-GX* system is a high-performance analytics cluster with a pre-integrated stack of popular analytics packages, including *Apache Spark*, *Apache Hadoop* and *Jupyter Notebooks*, all managed using the Apache Mesos[22] resource manager. These are complemented witha myriadof tools and frameworks to allow data analytics applications to be developed in Python, Scala, ,R and Java.

```
spark-submit --py-files defoe.zip \
defoe/run_query.py \
bln_files.txt bln  \
defoe.papers.queries.keyword_and_concordance_by_date \
queries/preprocess.yml \
-r results_krakatoa \
-n 324
```

Fig. 12: Example of the command line for submitting the `keyword_and_concordance_by_date` query. Users needs to select the digital corpus to use (`bln_files`), the corresponding object model (papers), the query (`keyword_and_concordance_by_date`), the preprocessing pipeline implementation (`preprocess.yml`), the file to save the results in (`results_krakatoa`) and the number of processes to use (324).

The Alan Turing Institutes deployment of the *Urika-GX* system (hereon called Urika) includes 12 compute nodes (each with 2x18 core 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.

Little work was needed to run the defoe queries in *Urika-GX* using several nodes. The *Urika-GX* software stack includes a fault-tolerant *Spark cluster* configured and deployed to run under Mesos, which acts as the cluster manager.

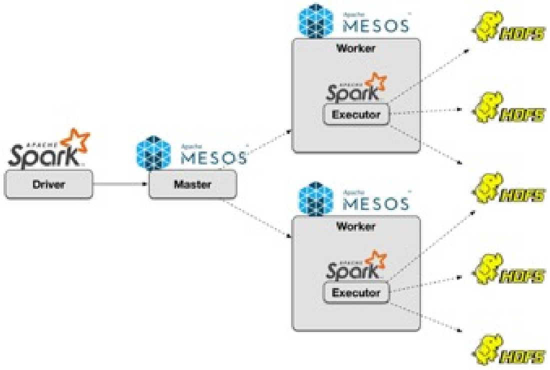


Fig. 13: Apache Spark Architecture in *Urika-GX*.

Therefore, we just needed to adjust four Spark parameters: Spark master with the URL of the Mesos master node; number of executors (worker nodes’ processes in charge of running individual Spark tasks); number of cores per executor (up to 36 cores); and the amount of memory to be allocated to each executor (up to 250GB). Even though *Urika-GX* has 12 nodes, at the time of our experiments, only 9 nodes and 324 cores were available.

[21] https://github.com/alan-turing-institute/defoe_visualization.

[22] http://mesos.apache.org/

B. Eddie

Not all researchers have access to *Urika-GX*. It is useful to enable researchers to use *defoe* on other HPC environments upon which, unlike *Urika-GX*, Spark has not been pre-installed and configured. *Eddie* is a University of Edinburgh HPC cluster. It consists of some 7000 Intel Xeon cores with up to 3TB of memory available per compute node. The cluster uses the Open Grid Scheduler batch system on Scientific Linux 7.

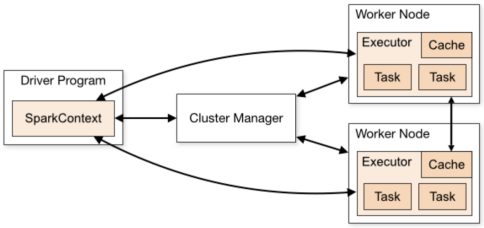


Fig. 14: Apache Spark Architecture in *Eddie*

Since Apache Spark is not available in *Eddie* as a module, we have created a new set of scripts to provision it on demand and for a specific period of time within a batch job. The batch job starts the Spark master, Spark workers, and registers all workers against the master.[23]

This new Spark on-demand cluster facility could be used for running all types of Spark applications, going beyond to the original scope of this work.

In terms of data analytic capacities of both HPC environments used in our case studies, running Spark queries on Eddie requires a more complex process than on Urika-GX. We are also more exposed to failures, since if the Spark master crashes, no new queries can be submitted until we restart the cluster. However, Eddie gives us more flexibility to configure a Spark cluster, having the possibility of using a higher number of executors (and cores) to run our text mining queries.

VI. Conclusions and Future work

In this paper, we have presented a new digital eScience toolbox called *defoe*. It allows for extracting knowledge from historical data by running text analyses across large digital collections in parallel. It offers a rich set of text mining queries, which have been defined by humanities researchers. We have included NLP prepossessing techniques to mitigate against OCR errors and standardise the textual data. We have tested *defoe* portability by running it on different computing environments and using five different digital collections. Furthermore, we have also provided new facilities to provision a multi-node Apache Spark on demand cluster for HPC enviroments that do not support Spark directly.

All this work provides the means to search across large-scale datasets and to return results for further analysis and interpretation by historians.

In the future we would like to extend *defoe* to identify and extract commonality across the object models and run queries across them transparently to users. We are currently exploring different ways to store the preprocessed RDDs (e.g. in object storage) so we will be able to run more than a query reusing these RDDs without the necessity to ingest a digital collection repeatedly. Finally, we plan to create a web interface (e.g. virtual environment) for enabling researchers to select query(ies) to run against selected data collection(s) in a distributed computing environment (in a transparent way for researchers) and obtain the results in this interface. In this way, we hope to make this facility more accessible to humanities researchers hoping to access text mining facilities.

Acknowledgements

This work was funded by Scottish Enterprise as part of the Alan Turing Institute-Scottish Enterprise Data Engineering Programme, and by AHRC as part of the *Living with Machines* via the Strategic Priorities Fund.

References

[1] A. Verheusen, Mass digitisation by libraries: Issues concerning organisation, quality and efficiency., LIBER Quarterly (Issue 18(1)) 28–38. doi:http://doi.org/10.18352/lq.7902.

[2] M. Terras, Opening access to collections: the making and using of open digitised cultural content Volume 39 (Issue 5) (2015) 733–752. doi:10.1108/oir-06-2015-0193.

[3] M. Terras, The potential and problems in using high performance computing in the arts and humanities: the researching e-science analysis of census holdings (reach) project.

[4] L. Vincent, Google book search: Document understanding on a massive scale, in: 9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil, 2007, pp. 819–823. doi:10.1109/ICDAR.2007.125.

[5] A. Wynne, Big data and digital transformations in the humanities: are we there yet? textual digital humanities and social sciences data.

[6] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, I. Stoica, Apache spark: A unified engine for big data processing, Commun. ACM 59 (11) (2016) 56–65. doi:10.1145/2934664.

[7] M. Terras, J. Baker, J. Hetherington, D. Beavan, M. Zaltz Austwick, A. Welsh, H. O'Neill, W. Finley, O. Duke-Williams, A. Farquhar, Enabling complex analysis of large-scale digital collections: humanities research, high-performance computing, and transforming access to British Library digital collections, Digital Scholarship in the Humanities 33 (2) (2017) 456–466. doi:10.1093/llc/fqx020.

[8] L. D. Dalcin, R. R. Paz, P. A. Kler, A. Cosimo, Parallel distributed computing using Python, Advances in Water Resources 34 (9) (2011) 1124–1139. doi:10.1016/j.advwatres.2011.04.013.

[9] S. Tanner, T. Munoz, P. Ros, Measuring mass text digitization quality and usefulness lessons learned from assessing the ocr accuracy of the british librarys 19th century online newspaper archive.

[10] R. Cordell, D. Smith, A research agenda for historical and multilingual optical character recognitiondoi:10.1108/oir-06-2015-0193.

[11] E. Loper, S. Bird, Nltk: The natural language toolkit, in: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, ETMTNLP '02, 2002, pp. 63–70.

[12] M. Honnibal, M. Johnson, An improved non-monotonic transition system for dependency parsing, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1373–1378.

[23] Details about how to deploy, configure, and run *defoe* queries on on-demand Spark clusters can be found at https://github.com/rosafilgueira/Spark_EDDIE_TextMining.