THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

Extending defoe for the efficient analysis of historical texts at scale

Citation for published version:

Filgueira, R, Grover, C, Karaiskos, V, Alex, B, Van Eyndhoven, S, Gotthard, L & Terras, M 2021, Extending defoe for the efficient analysis of historical texts at scale. in *2021 IEEE 17th International Conference on eScience (eScience)*. IEEE, pp. 21-29, IEEE eScience 2021 - 17th IEEE eScience 2021 International Conference, Innsbruck, Austria, 20/09/21. <https://doi.org/10.1109/eScience51609.2021.00012>

Digital Object Identifier (DOI):

[10.1109/eScience51609.2021.00012](https://doi.org/10.1109/eScience51609.2021.00012)

Link:

[Link to publication record in Edinburgh Research Explorer](#)

Document Version:

Peer reviewed version

Published In:

2021 IEEE 17th International Conference on eScience (eScience)

Publisher Rights Statement:

© 2021 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

General rights

Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Extending *defoe* for the Efficient Analysis of Historical Texts at Scale

Rosa Filgueira

School of Mathematical and Computer Sciences
Heriot-Watt University, UK
R.Filgueira@hw.ac.uk

Beatrice Alex

EFI, School of Literatures, Languages and Cultures
University of Edinburgh, UK
balex@ed.ac.uk

Claire Grover, Vasilios Karaiskos

School of Informatics
University of Edinburgh, UK
{grover, vkaraisk}@ed.ac.uk

Sarah Van Eyndhoven, Lisa Gotthard

PPLS, Linguistics and English Language
University of Edinburgh, UK
{S1890120@ed.ac.uk, l.gotthard}@ed.ac.uk

Melissa Terras

College of Arts, Humanities and Social Sciences
University of Edinburgh, UK
m.terras@ed.ac.uk

Abstract—This paper presents the new facilities provided in *defoe*, a parallel toolbox for querying a wealth of digitised newspapers and books at scale. *defoe* has been extended to work with further Natural Language Processing () tools such as the *Edinburgh Geoparser*, to store the preprocessed text in several storage facilities and to support different types of queries and analyses. We have also extended the collection of XML schemas supported by *defoe*, increasing the versatility of the tool for the analysis of digital historical textual data at scale. Finally, we have conducted several studies in which we worked with humanities and social science researchers who posed complex and interested questions to large-scale digital collections. Results shows that *defoe* allows researchers to conduct their studies and obtain results faster, while all the large-scale text mining complexity is automatically handled by *defoe*.

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

I. INTRODUCTION

Libraries, archives, and museums [1], [2] offer a wide range of digital collections of textual resources, which have the potential to provide an invaluable resource to historians, humanities, and computational linguistics researchers. However, enabling these users to take advantage of advanced computing environments, analytics frameworks, and text mining techniques to mine large scale digital collections effectively remains challenging [3].

We have contributed to this area by creating *defoe* [4]¹, a scalable, parallel and portable toolbox which aims to enable the analysis of digital historical datasets in a consistent way and returns results for further analysis and interpretation. *defoe* can be used in any HPC cluster or cloud platform, as it has

been designed to avoid imposing any specific constraints on the compute environment.

Another aim of the design of *defoe* was to make it flexible so it can consume historical textual collections scanned via OCR from a variety of XML schemas. In this work we have added a new XML schema in *defoe* to extract the knowledge from the data collections offered by the National Library of Scotland² (NLS), so that the system now supports four different XML schemas. Among the digital collections offered by NLS, we have the *Encyclopaedia Britannica* (see Figure 1).

Fig. 1: Page 36 of the First Edition of the *Encyclopaedia Britannica* (Volume 1), and its corresponding XML file.

Working with domain researchers we discovered key functionalities missing from *defoe*. For example, it did not provide support for i) storing ingested and preprocessed data, ii) running several text-mining queries at once, iii) running queries across different XML schemas, among others. In this paper we have investigated how to extend *defoe*, so it supports the needs of the research communities by enabling a better complex

¹<https://github.com/defoe-code>

²<https://data.nls.uk/>

analysis of digital datasets at scale. The new contributions of this work are:

- Supporting the ingestion of NLS digital collections.
- Employing new techniques to mitigate OCR errors
- Integrating geoparsing capabilities
- Extracting commonalities across digital collections schemas
- Storing pre-processed data in different storage solutions
- Providing different query and analysis modalities.

We demonstrate the feasibility and benefits of this work using eight case studies, two computer infrastructures, and four digital collections.

The remainder of the paper is structured as follows. Section II presents *defoe* background and Section III reviews relevant text analysis tools. Section IV describes the new *defoe* functionalities. Section V details the features of the computer infrastructures used in this work. Section VI describes the use cases conducted. We conclude with a summary of achievements and outline future work.

II. BACKGROUND

This section gives an overview of our previous work on *defoe* [4], providing the necessary background for the functionalities presented in Section IV.

defoe is a Python toolbox that uses Apache Spark [5], an open-source distributed HPC framework for large-scale data processing. Spark *Resilient Distributed Datasets (RDD)* are the fundamental data structures of Apache Spark, which is an immutable distributed collection of objects partitioned across cluster nodes.

The historical textual collections we consider can have different XML schemas describing either the structure of a digital document or the actual textual content, depending on the OCR scanning technique employed. All of these schemas are slightly different from each other, and none of them is universally used. Therefore, we initially created three object models (*PAPERS*, *NZPP* and *ALTO*) to map the physical representations and XML schemas. These allow *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. This means that we could only ingest collections that share the same object model at the same time.

Previously, in the first version of *defoe*, we provided *on-the-fly single query analysis*, which consisted of 1) ingesting a digital collection (by using the corresponding object model) into RDDs in memory; 2) pre-processing data; 3) running a single text mining query in parallel; and 4) returning results as YAML files. These four steps (we refer to them as the *text analysis pipeline* – see Figure 4) are repeated every time we want to run a text mining query over one or several collections.

All *defoe* text mining queries are based on a number of operations (e.g. *filter*, *flatMap*, *map*, *reduce*, etc.) that are combined to perform text mining analyses. Figure 2 shows as an implementation example, the *total_pages* query, in which a *flatMap* operation is applied to an archive

to return the list of documents it contains (e.g. volumes, books). For each document, the *map* operation extracts the number of pages, gathering the total number of documents (volumes, books) and the total number of pages within those.

```
def do_query(archives, config_file=None, logger=None, \
             context=None):
    # [archive, archive, ...]
    documents = archives.flatMap(lambda archive:\
                                list(archive))
    # [num_pages, num_pages, ...]
    num_pages = documents.map(lambda document:\
                              document.num_pages)
    result = [documents.count(), num_pages.reduce(add)]
    return {"num_documents": result[0],\
            "num_pages": result[1]}
```

Fig. 2: *defoe* *total_pages* query: Iterates through archives and counts the total number of documents (e.g. volume, book etc) and total number of pages.

As an example, we run this query using the six volumes of the 1st Edition of the *Encyclopaedia Britannica* (see Figure 3).

```
Result:
num_documents: 6
num_pages: 5462
```

Fig. 3: *defoe* *total_pages* query results using the six volumes of the 1st edition of the *Encyclopaedia Britannica*. This archive comprises six documents, one per volume.

defoe was complemented with *defoe visualisation*, a repository of Jupyter notebooks enabling researchers to visualise and explore further the results obtained.

III. RELATED WORK

There are other computational text analysis tools that allow scholars to explore their texts and perform textual analysis. Some of the most relevant among those include:

- *Voyant* [6]³ is a web-based, dashboard-style tool that allows users to upload a corpus and visualize patterns in various ways. For instance, users can experiment with colorful word clusters that represent word frequency and visualize how specific words and phrases appear across texts in line graphs.
- *Mallet* [7] is a machine learning software program that is used through the command line with Python. Though it requires some technical skill to install and run, it can produce powerful results by generating ‘topics’, or lists of words that frequently appear together in corpora.
- *Natural language toolkit (NLTK)* [8] is a suite of Python tools that allows users to analyze human language data using classification, tokenizing, tagging, and more. Though it requires some technical expertise to run, it can be very useful for both teaching and analysis.
- *spaCy* [9] is an open-source library for advanced natural language processing in Python. It is designed specifically

³<https://voyant-tools.org/>

for production use and helps build applications that process large volumes of text. Some of the features provided by *spaCy* are: Tokenization, Part-of-Speech (PoS) Tagging, Text Classification and Named Entity Recognition (NER).

- *Google N-Grams* [10] ⁴ allows users to plot single words and short phrases over time in a large subset (5 million books) of the corpus.
- The *Stanford Natural Language Processing Group* ⁵ provides statistical NLP, deep learning NLP, and rule-based NLP tools for major computational linguistics problems, which can be incorporated into applications with human language technology needs. These packages are widely used in industry, academia, and government.
- *Weka* [11] is Java-based data mining, visualization, and machine learning software. The algorithms can either be applied directly to a dataset or called from your own Java code. *Weka* contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization.

defoe complements these tools, as it has been designed to analyse digital historical texts. With a single command-line, users indicate a directory(ies) containing the XML files describing a collection(s), and *defoe* analyses them in parallel automatically.

IV. NEW FUNCTIONALITIES

The main improvements we have implemented set across the *text analysis pipeline* mentioned in Section II and the *defoe* query submission and analysis systems (see Figure 4). The following subsections detail each of them.

Fig. 4: Overview of *defoe*.

A. Ingestion of NLS digital collections

The NLS Digital Scholarship Service ⁶, has published 19 different digital textual collections. The *Encyclopaedia Britannica*, the *Gazetteers of Scotland*, or the *Chapbooks printed*

in Scotland are three of these. They use the same two XML schemas in all their collections: METS XML schema⁷ for descriptive, structural, technical and administrative metadata; and ALTO XML schema⁸ for encoding OCR text.

Each NLS digital collection (archive) usually comprises several volumes or books. For example, the *Encyclopaedia Britannica* has 195 volumes, and each of them has a METS XML file. Then, for each of those volumes, it provides an ALTO XML and one image file per page. These make up a total of 195 METS files, 155,388 ALTO XML files, and 155,388 image files for the entire collection.

In our previous work, we created an *ALTO* object model to ingest METS and ALTO XMLS files. We used this object model to ingest the *British Library Books* and *Find My Past* collections. However, the NLS METS and ALTO XML schemas have some differences from those: i) they employ a different naming of documents and namespaces that affects their XML files; ii) the XML schemas provide a larger number of attributes. To accommodate these, we created a new object model, called *NLS*. See Figure 5.

Fig. 5: New *NLS* object model. This allows *defoe* to ingest NLS digital collections into Spark RDDs.

The new *NLS* object model is added to our previous collection of object models [4]. We have used it to analyse the *Encyclopaedia Britannica*, the *Gazetteers of Scotland*, and the *Chapbooks printed in Scotland*, as described in Section VI. Note that this object model can be used to analyse any of the other digitised collections offered by NLS.

B. Mitigation of OCR errors

Another significant improvement over previous versions of the tool has been the mitigation of two types of optical character recognition (OCR) errors. To do so, we updated the NLP preprocessing pipeline by adding a new cleaning step after each sentence has been tokenised. See Figure 6.

This new step employs two techniques, described in [12], to mitigate the ‘long-S’ and the line-break hyphenation OCR errors. In most works printed before about 1800 two forms

⁴<https://books.google.com/ngrams>

⁵<https://nlp.stanford.edu/>

⁶<https://data.nls.uk/data/digitised-collections/>

⁷<http://www.loc.gov/standards/mets/>

⁸<https://www.loc.gov/standards/alto/>

of the lower-case ‘s’ were used. One was the ‘s’ that is still in use today; the other was the ‘long-S’, a character which looks like ‘f’ without the right-hand part of its crossbar. These ‘long-S’ characters often generate OCR errors where they are mistaken for ‘f’ characters, resulting in misspellings (e.g. ‘Congress’ misspelled as ‘Congrefs’). Regarding hyphenation, the OCR scanned text usually preserves the line breaks from the original paper, which include hyphenated words (e.g. ‘department’). However, it is still desirable to remove the hyphen (e.g. department) in order to increase text mining accuracy. Both techniques are able to detect and fix automatically both issues, increasing the quality of tokens generated.

Fig. 6: Updated *defoe* NLP pre-processing pipeline.

Finally, after cleaning and normalising each token, we apply stemming and lemmatisation. Details of other steps of this pipeline can be found in [4].

C. DCOM: *defoe* Common Object Model

One of the drawbacks of the previous tool was only being able to run queries across collections that share exactly the same XML schemas, and therefore use the same object model. To remove this barrier and allowing users to run queries across collections, independently of their XML schemas, we identified and extracted commonality across the four object models (see Section IV-A for more details) currently supported. Following this, we created a new model with these common attributes. Table I contains a description of each *DCOM* attribute and an example of mapping an *Encyclopaedia Britannica* page to *DCOM*.

In order to map digital collections to our new common object model, we have created a special query called *DCOM_mapping* query (see Figure 7). This query ingests a collection using the appropriate object model (e.g. *NLS*, *PAPERS*, *NZPP*, *ALTO*) into Spark RDDs, pre-processes the text of each of the text units of the collection, maps them as *DCOM* objects, and stores them in one of our storage solutions.

Note that for each of the text units of a given collection, *DCOM_mapping* creates a new Spark RDD with a *DCOM*

Attribute	Description, Options and Examples
Title	Title of the collection Example: <i>Encyclopaedia Britannica</i>
Subtitle	Subtitle of the collection Example: <i>Null</i>
Edition	Edition of the collection Example: <i>Seventh edition, Volume 13, LAB-Magnetism</i>
Authors	Authors related with the collection Example: <i>Dugald Stewart, Sir James Mackintosh, etc ...</i>
Publishers	Publishers of the collection Example: <i>A. & C. Black</i>
Genre	Genre of the collection Example: <i>Null</i>
Topic	Topic of the collection Example: <i>Null</i>
Year	Year of the collection Example: <i>1842</i>
Place	Place of the collection Example: <i>Edinburgh</i>
Archive_Filename	Path of the collection Example: <i>../EB/193108323/</i>
Text_Unit	Unit representing each XML Options: Page, Article Example: <i>Page</i>
Source_Text_Filename	Path of each text unit Example: <i>alto/193202213.34.xml</i>
Text_Unit_ID	Id of the unit Example: <i>Page72</i>
Text_Unit_Title	Title of the text unit (e.g. Article title) Example: <i>None</i>
Num_Text_Unit	Number of text units of the collection Example: <i>810</i>
Type_Archive	Type of archive of the collection Options: Newspaper, Book, Volume Examples: <i>Volume</i>
Model	Object model used to ingest the collection Options: <i>PAPERS, NZPP, ALTO, NLS</i> Examples: <i>NLS</i> .
Source_Text_Raw	Raw text extracted from each text unit (XML) Example: <i>62 LANG Language, having lived ... , merely by considering the nature of speech, and the men- tal ...</i>
Source_Text_Clean	Cleaned text from each text unit (XML) Example: <i>62 LANG Language, having lived ... , merely by considering the nature of speech, and the mental ...</i>
Source_Text_Norm	Normalized text from each text unit (XML) Example: <i>lang language having lived ... merely by considering the nature of speech and the mental ...</i>
Source_Text_Lemma	Lemmatized text from each text unit (XML) Example: <i>lang language having ... merely by considering the nature of speech and the mental ...</i>
Source Text Stem	Stemmed text from each text unit (XML) Example: <i>lang languag have live ... mere by consid the natur of speech and the mental ..</i>
Num_Words	Number of words detected in the Source_Text_Raw Example: <i>1409</i>

TABLE I: Description of *DCOM* attributes, using examples from mapping page 72 of the Seventh Edition (Volume 13), of the *Encyclopaedia Britannica*⁹ to *DCOM*. Note that some attributes for this example do not have values, as, for this particular collection, this information is not available in its original XML files.

object in it. For example, the Volume 13 of the *Encyclopaedia*

⁹<https://digital.nls.uk/encyclopaedia-britannica/archive/193108323#?c=0&m=0&s=0&cv=71&xywh=-348%2C-499%2C6940%2C5144>

Fig. 7: Overview of `DCOM_mapping` query, which maps collections to `DCOM` and stores them in the selected storage facility.

Britannica has 810 pages, which means that our query has generated a list of 810 `DCOM-RDDs`. Once we have the collections represented with the same model, we can run queries across them, independently of their XML schemas.

D. DCOM Storage

The improved version of the tool provides additional solutions to store digital collections as `DCOM` objects:

- *Hadoop HDFS* file system [13]: Very often Apache Spark and Hadoop Clusters are installed together. Therefore, *Hadoop HDFS* is very often used in combination with Spark applications to write or read data to/from RDDs.
- *ElasticSearch* engine [14]: This popular engine provides storing and indexing facilities for structured or unstructured text, which enables fast searches.
- *PostgreSQL (PSQL)* database [15]: As a powerful, open source object-relational database system that uses and PSQL combines SQL with many features to safely store and scale data workloads.

These storage solutions bring different benefits to *defoe*. While many *Spark* applications use *Hadoop HDFS* for storing RDDs, *PSQL* is very popular among researchers. *ElasticSearch* offers additional functionalities with text data that complement *defoe* very well. We provide support for all of these by exporting `DCOM` objects to these formats, allowing users to choose which storage solution to use depending on their familiarity with them or availability in their computing facilities.

E. Query Submission and Analyses Types

Initially, *defoe* only supported *on-the-fly* analysis, ingesting and pre-processing data every time a query was submitted, and limited the analysis (performed in the same query submission) to collections that share the same XML schema.

We have now enabled *defoe* to run analyses using the pre-processed collections previously stored using any of the solutions listed in Section IV-D. We refer to this feature as *pre-processed* analysis, and it enables us to query collections

without the necessity to re-ingest and pre-process data repeatedly across different XML schemas.

We have also improved the ability of *defoe* to run more than a single query at once. In the presented version, we can indicate a list with multiple text mining queries that we want to run using either of the two analyses previously described. With this new query submission modality, data is ingested once, and all queries indicated in the list are run against the data loaded in memory. This modality is very efficient when we already know what queries will be run on a collection.

F. Integrating the Edinburgh Geoparser

A further important improvement is the integration of *defoe* with the *Edinburgh Geoparser* [16] and *spaCy* NLP tools for automatically tagging place names in text and resolving them to their correct latitude and longitude coordinates or gazetteer entry. The *Edinburgh Geoparser* is a language processing tool designed to detect place name references in English text and ground them against an authoritative gazetteer, so that they can be plotted on a map. The available download on the *Edinburgh Geoparser* website¹⁰ was developed for contemporary newspaper text but the Geoparser team has also adapted it to historical [17] and literary text [18].

The geoparser is implemented as a pipeline with two steps:

- *geotagging*, in which place name entities are identified, and
- *georesolution*, which grounds place name entities against locations contained in a gazetteer.

the available download on our website was developed for contemporary newspaper text but that the Geoparser team has also adapted to historical [1] and literary text [2].

With the integration of the *Edinburgh Geoparser* and *spaCy* we have added more ready-to-use text-mining analyses to *defoe*, and enabled parallel geoparsing of text data. We also support two different geoparsing methods by using different tokenization and NER techniques (the *geotagging* step above). These are:

- The original *Edinburgh Geoparser* for *geotagging* and *georesolution* of historical text.
- Using *spaCy* for *geotagging* in combination with *Edinburgh Geoparser georesolution*.

Our aim behind this new functionality was not only to provide a parallel geoparsing facility, but also to employ different tokenization and NER techniques. Such re-combining of techniques helped to develop more accurate geoparsing for historical text collections.

For each of the previous geoparsing methods, we created two different queries (*original_geoparser* for the first method and *spacy_georesolver* for the second one). Both queries work either loading the preprocessed clean stored text (using *Source_Text_Clean* attribute - see Table I) or directly ingesting the collections' XML files and executing the queries after the NLP pre-processing pipeline shown in Figure 6. Next, each query identifies entities by employing

¹⁰<https://www.ltg.ed.ac.uk/software/geoparser/>

a different *geotagging* technique. Finally, both queries apply *Edinburgh Geoparser georesolution* to place name entities.

G. Further improvements

We also improved other aspects of *defoe*, while we were working on the use cases presenting in Section VI. For example, we have doubled the number of text mining queries supported by *defoe*, offering now a total of 64. We have developed new Jupyter notebooks¹¹ for analyzing and visualizing the results obtained with the new *defoe* queries.

V. COMPUTE INFRASTRUCTURES

We have run the use cases presented in the next Section VI with the new updated *defoe* in two compute environments:

- *Cirrus* HPC-Cluster¹²: This environment uses a state-of-the-art SGI ICE XA system with 280 compute nodes with Lustre as the file system and CentOS Linux as the OS, and Slurm to schedule jobs. *Cirrus* standard compute nodes contain two 2.1 GHz, 18-core Intel Xeon E5-2695 (Broadwell) series processors and 256 GB of memory with support for hyperthreading. This means that we can use up to 72 processes per node for running *defoe*. In our case, we used up to three nodes (216 processes), depending on the size of the collection to analyse and the cluster availability.
- *Google Cloud Platform*¹³: Offered by Google, it is a suite of cloud computing services (such as computers and hard disk drives, or virtual machines (VMs) that runs on the same infrastructure that Google uses internally for its end-user products. For this work, we have configured a Debian VM with 500GB of disk, 96 vCPU and 1.4 TB of memory.

Since *Apache Spark* is not available in *Cirrus* as a module, we have created a new set of scripts to provision an *Spark Cluster* on demand and for a specific period of time within a *slurm job*. It starts the Spark master, Spark workers, and registers all workers with the master¹⁴. We also have created another set of *slurm jobs* for running our text-mining analyses on the provisioned *Spark Cluster*¹⁵. *defoe* was installed in our HOME directory using a *conda* Python3 environment.

In the case of the *Google Cloud Platform*, we installed *Apache Spark* in the VM along with *defoe*, also using a *conda* Python3 environment. Once again, we have documented the full installation process¹⁶ to provide full reproducibility.

VI. USE CASES

New *defoe* functionalities have been evaluated using a number of studies funded by the University of Edinburgh Centre for Data, Culture & Society (CDCS) Text Mining Lab¹⁷. In

these studies, we worked with several humanities and social science researchers, using *defoe* to mine the *Times Digital Archive* (TDA) newspapers and the NLS collections displayed in Figure 8.

Dataset	Period	Structure	XML Schema	Space	Model
British Library Books (BLB)	1510-1899	ZIP per book - XML metadata - XML per page	METS and ALTO schemas	~220GB	<u>ALTO</u>
British Library Newspapers (BLN)	1714-1950	XML per issue	GALEN Schema	~1TB	<u>PAPERS</u>
Times Digital Archive (TDA)	1785-2009	XML per issue	GALEN Schema	~324GB	<u>PAPERS</u>
Papers Past New Zealand and Pacific newspapers (NZPP)	1839-1863	XML per 22 articles	XML from a search via an API	~4GB	<u>NZPP</u>
Gazetteers of Scotland, 1803-1901	1803-1901	- XML metadata - XML per page	METS and ALTO schema	4.7 GB (includ. Fig)	<u>NLS</u>
Encyclopaedia Britannica	1768-1860	- XML metadata - XML per page	METS and ALTO schemas	46GB (includ. figs)	<u>NLS</u>
Chapbooks printed in Scotland	17 th to 19 th Century	- XML metadata - XML per page	METS and ALTO schemas	7.7 G (includ. figs)	<u>NLS</u>

Fig. 8: Updated list of digital collections analysed with *defoe*. The last three rows are the new additions to this list.

In total, we conducted eight CDCS text mining studies¹⁸. We are describing three of them in the following subsections.

A. Extracting articles within *Encyclopaedia Britannica*

As mentioned before, for each page of the *Encyclopaedia Britannica* (see Figure 8) we have an ALTO XML with the OCR recognized text. However, these do not tell us when an article of the encyclopaedia starts or when it finishes within pages. In this study we have focused on extracting the articles represented in the encyclopaedias, so they can be analysed independently without the surrounding text.

We have developed a new query, called `extract_articles_eb` for this end, which loads pages directly into memory from one of our storage solutions (using the *Source_Text_Clean* attribute - see Table I). Alternative, the query can also ingest the *Encyclopaedia Britannica* XML files into memory using the *NLS* object Model introduced in Section 5, and to pre-process the text of each page by applying the updated NLP pre-processing pipeline (see Figure 6). After the text in each page is cleaned and loaded in memory, the query detects in parallel the articles contained within each page by making use of page headers and text patterns. We have detected two types of articles with two different patterns at page level:

- Short articles we refer to as ‘articles’: Usually presented by a *term* in the main text in uppercase, followed by a “,” (e.g. *ACQUEST*), and then a description of the term (similar to an entry in a dictionary). This description is normally one or two paragraphs long, but of course there are exceptions. An example of a page containing ‘articles’ is shown in Figure 9

¹⁸https://github.com/defoe-code/CDCS_Text_Mining_Lab

¹¹https://github.com/defoe-code/defoe_visualization

¹²<https://www.cirrus.ac.uk/>

¹³<https://cloud.google.com/>

¹⁴[https://github.com/defoe-code/CDCS_Text_Mining_Lab#](https://github.com/defoe-code/CDCS_Text_Mining_Lab#1-spark-installation-steps)

1-spark-installation-steps

¹⁵[https://github.com/defoe-code/CDCS_Text_Mining_Lab/blob/master/](https://github.com/defoe-code/CDCS_Text_Mining_Lab/blob/master/Round2.slurm)

¹⁶<https://github.com/defoe-code/defoe/blob/master/docs/setup-VM.md>

¹⁷<https://www.cdcs.ed.ac.uk/cdcs-text-mining-lab-call-projects>

- [illegible]

Our query is able to detect both ‘articles’ and ‘topics’, and classify each page accordingly. An *Encyclopaedia Britannica* page can contain just ‘articles’, or a ‘topic’, or can have a mix of both²⁰. An *Encyclopaedia Britannica* page may also contain information not related to articles, such as the cover of a volume, preface, tables, figures, etc. Therefore, our query performs pages classification (‘articles’, ‘topic’, ‘mix’, ‘other’) depending on the type of articles detected.

In summary, the `extract_articles_eb` query automatically extracts the articles (if any) within pages, along with metadata, such as the number of articles per page, the number of words per article, header, and page classification. Figure 9 shows a list of ‘articles’ extracted by our new query using the example introduced in Section I.

The *Gazetteers of Scotland*²¹ is a collection of twenty volumes of the most popular descriptive historical gazetteers of Scotland in the 19th century, providing ‘dictionaries’ of places such as towns, cities, castles and antiquities. In this study, we have

15 Places most mentioned using the Original Geoparser across all Scottish Gazetteers

Place	Frequency
Scotland	12500
Edinburgh	10000
Glasgow	8500
Inverness	5200
Perth	4500
Perthshire	4500
Clyde	4200
Aberdeen	3900
Stirling	3200
Ork	3200
Ayr	3000
England	2900
Ayrshire	2900
Aberdeenshire	2800
Tay	2600

Region	Frequency of Places
Scotland	11800
Glasgow	9800
Edinburgh	9300
Inverness	5000
Perth	4600
Perthshire	4300
Aberdeen	4200
Ayr	3100
Ayrshire	3100
Stirling	2900
Fife	2800
Lanarkshire	2700
Aberdeenshire	2600
Dumfries	2600
England	2600

Figure 10 shows the 15 most frequent places names extracted using the `original_geoparser` (above) and `spacy_georesolver` (bellow) queries from the *Gazetteers of Scotland*. Note that the five most frequent locations mentioned among both techniques are *Edinburgh, Scotland, Glasgow, Inverness* and *Perth*. More preliminary results of geoparsing the *Gazetteers of Scotland* can be found in [19].

²²<https://openlayers.org/>

²¹<https://data.nls.uk/data/digitised-collections/gazetteers-of-scotland/>

Historic map

OpenStreetMap **Abercrombly, 1167**

Transparency:

Placenames source: **OpenStreetMap** **1803-1901**

Period:

Search placenames:

All placenames types ☒

All countries ☒

Non-georeferenced placenames

Abercrombly ☒

Abercrombly Bank ☒

Abercrombly ☒

Abercrombly ☒

Articles

Singapore

...18 feet on the upper **Abercrombly** lock-gate : thus reducing the...[1842](#), p. 271

...in which the **Abercrombly** locks occur. The railway...[1842](#), p. 271

...over the upper gate-**Abercrombly** lock. This shows an...[1842](#), p. 271

...the canal; and the **Abercrombly** regulating lock was so adapted...[1842](#), p. 271

While the other two use cases presented in Section VI were run on *Cirrus*, this particular use case was run on *Google Cloud*. This is mainly because the *Edinburgh Geoparser georesolution* needs access to online gazetteers to ground place names. *Cirrus* nodes (as many other HPC-Clusters) cannot directly access the Internet because they have a private IP. This is not the case for *Google Cloud*.

Since the 16th century, Scots has been increasingly influenced by English, particularly in writing. Written Scots came to be seen as less appropriate or lucrative for text genres such as newspapers. On the basis of this, in this study we explored the use of written Scots in the digitised 19th century *Chapbooks* collection ²⁴, to investigate (i) whether Scots survives in the printed world, and, if so, (ii) in what contexts Scots is used and how this indicates the status it had as a written medium during this time.

²³<https://maps.nls.uk/joins/747.html>

²⁴<https://data.nls.uk/data/digitised-collections/chapbooks-printed-in-scotland/>

²⁵https://github.com/defoe-code/CDCS_Text_Mining_Lab/tree/master/Round2_Requirements/Sarah_Lisa#defoe-queries

²⁶https://github.com/defoe-code/CDCS_Text_Mining_Lab/blob/master/Round2_Requirements/Sarah_Lisa/Lexicon_Scots_English.xlsx

The figure consists of two vertically stacked line plots. Both plots share the same x-axis, labeled 'years', which ranges from 0 to 1000. The y-axis for both plots is labeled 'Normalized frequency'.

The top plot has a y-axis scale from 0.000 to 0.008. It displays four data series: 'nile' (blue line with diamond markers), 'bagg' (orange line with diamond markers), 'gaard' (green line with diamond markers), and 'land' (red line with diamond markers). The 'land' series shows a major peak of approximately 0.0048 around year 1000, and another sharp peak of about 0.0078 around year 1050. The other three series remain very close to zero throughout the period.

The bottom plot has a y-axis scale from 0.000 to 0.020. It displays the same four data series. The 'land' series shows a very prominent peak of approximately 0.020 around year 1000. The 'nile' series shows several smaller peaks, with the highest being around 0.012 around year 1000. The 'bagg' and 'gaard' series remain relatively flat and close to zero.

Despite these anglicising pressures, the results obtained from the previous *defoe* queries indicate that Scots clearly had not disappeared from the public eye entirely. Through *defoe*, we were able to extract Scots tokens from previously unmined historical data, but also important metadata attributes about the text such as *genre* and *place* — factors that we will take into consideration in our analysis of which factors encouraged or suppressed possible use of Scots in these publications. To obtain these attributes, we created a new query, `extract_metadata`, to extract all the information associated with this collection (see Figure 13).

Fig. 13: Snapshot of metadata from the first 12 *Chapbooks*. We are currently updating the `extract_metadata` query to extract metadata in Dublin-Core format [20].

The results indicate that written Scots was still present in print, but primarily in narrative prose, with a clear preference for Scots in garlands, humorous material and verse. Thus, using the results from the *defoe* queries, we can statistically analyse the normalised frequency of Scots by chapbook, topic,

genre and author. The notebooks and visualisations²⁷ were useful to get an overview of the data, and to do initial explorations of the results.

VII. CONCLUSIONS

The new *defoe* functionalities described in this paper enable us to mine different historical collections more efficiently, supporting the research of different disciplines such as the social sciences, history, and computational linguistics. *defoe* has been enriched with a larger number of analyses readily available to users and has also been integrated with further NLP tools such as the *Edinburgh Geoparser*. We have also enabled *defoe* to run multiple queries at once across digital collections. We have proposed a new *DCOM* model to map collections independently of their XML schemas, and store them in several storage facilities. We provide support for analysing both i) XML files (*on-the-fly* analysis); and ii) pre-processed data stored (*preprocessed* analysis). And we have also created a new *NLS* object model for ingesting NLS digital collections and employing new techniques to mitigate more OCR errors. With all these improvements, we have made *defoe* more efficient for searching and analysing large-scale historical textual data. Most of the current analyses supported by *defoe* are keyword searches or frequencies, in which users can configure several parameters. In the future we plan to extend this work, and employ more sophisticated analyses by using more advanced machine learning and NLP algorithms.

VIII. ACKNOWLEDGEMENTS

This work was partly funded by the Data-Driven Innovation Programme as part of the Edinburgh and South East Scotland City Region Deal, by the University of Edinburgh, and by Google Cloud Platform research credits program. The authors wish to thank Lisa Otty at the University of Edinburgh Centre for Data, Culture & Society, Mike Bennett at the University of Edinburgh Library, and Sarah Ames at the National Library of Scotland for their help and support during this work.

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. URL <http://discovery.ucl.ac.uk/171144/>
- [4] R. Filgueira Vicente, M. Jackson, A. Roubickova, A. Krause, R. Ahnert, T. Hauswedell, J. Nyhan, D. Beavan, T. Hobson, M. Coll Ardanuy, G. Colavizza, J. Hetherington, M. Terras, *defoe*: A spark-based toolbox for analysing digital historical textual data, in: 2019 IEEE 15th International Conference on e-Science (e-Science), Institute of Electrical and Electronics Engineers (IEEE), United States, 2020, pp. 235–242, 2019 IEEE 15th International Conference on e-Science (e-Science), e-Science 2019 ; Conference date: 24-09-2019 Through 27-09-2019. doi:10.1109/eScience.2019.00033. URL <https://escience2019.sdsc.edu/>
- [5] 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.
- [6] G. Rockwell, S. Sinclair, Teaching text analysis with voyant, in: 8th Annual International Conference of the Alliance of Digital Humanities Organizations, DH 2013, Lincoln, NE, USA, July 16–19, 2013, Conference Abstracts, Alliance of Digital Humanities Organizations (ADHO), 2013, pp. 21–22. URL <http://dh2013.unl.edu/abstracts/workshops-011.html>
- [7] A. K. McCallum, Mallet: A machine learning for language toolkit, <http://mallet.cs.umass.edu> (2002).
- [8] 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, Association for Computational Linguistics, Stroudsburg, PA, USA, 2002, pp. 63–70. doi:10.3115/1118108.1118117. URL <https://doi.org/10.3115/1118108.1118117>
- [9] 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.
- [10] Google, Google ngram viewer, <http://books.google.com/ngrams/datasets> (2012). URL <http://books.google.com/ngrams/datasets>
- [11] E. Frank, M. A. Hall, G. Holmes, R. Kirkby, B. Pfahringer, I. H. Witten, Weka: A machine learning workbench for data mining., Springer, Berlin, 2005, pp. 1305–1314. URL <http://researchcommons.waikato.ac.nz/handle/10289/1497>
- [12] B. Alex, C. Grover, E. Klein, R. Tobin, Digitised historical text: Does it have to be mediocre?, in: Proceedings of KONVENS 2012 (LThist 2012 workshop), 2012, pp. 401–409.
- [13] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), MSST '10, IEEE Computer Society, USA, 2010, p. 1–10. doi:10.1109/MSST.2010.5496972. URL <https://doi.org/10.1109/MSST.2010.5496972>
- [14] elasticsearch, elasticsearch/elasticsearch (2015). URL <https://github.com/elasticsearch/elasticsearch>
- [15] T. P. G. D. Group, Documentation PostgreSQL 10.3 (2018).
- [16] C. Grover, R. Tobin, K. Byrne, M. Woollard, J. Reid, S. Dunn, J. Ball, Use of the Edinburgh Geoparser for georeferencing digitized historical collections, *Philosophical Transactions of the Royal Society A* 368 (1925) (2010) 3875–3889. URL <https://doi.org/10.1098/rsta.2010.0149>
- [17] B. Alex, K. Byrne, C. Grover, R. Tobin, Adapting the edinburgh geoparser for historical georeferencing, *International Journal of Humanities and Arts Computing* 9 (1) (2015) 15–35. doi:10.3366/ijhac.2015.0136.
- [18] B. Alex, C. Grover, R. Tobin, J. Oberlander, Geoparsing historical and contemporary literary text set in the city of edinburgh, *Language Resources and Evaluation* 53 (4) (2019) 651–675, springer Compact Gold OA. doi:10.1007/s10579-019-09443-x.
- [19] R. Filgueira Vicente, C. Grover, M. Terras, B. Alex, Geoparsing the historical gazetteers of scotland: Accurately computing location in mass digitised texts, in: Proceedings of the 8th Workshop on Challenges in the Management of Large Corpora, European Language Resources Association (ELRA), 2020, p. 24–30, 8th Workshop on the Challenges in the Management of Large Corpora, CMLC-8 ; Conference date: 16-05-2020 Through 16-05-2020. URL <http://corpora.ids-mannheim.de/cmlc-2020.html>
- [20] DCMI Usage Board, DCMI metadata terms, DCMI recommendation, Dublin Core Metadata Initiative, published online on December 18th, 2006 at <http://dublincore.org/documents/2006/12/18/dcmi-terms/> (December 2006). URL <http://dublincore.org/documents/2006/12/18/dcmi-terms/>

²⁷https://github.com/defoe-code/defoe_visualization/tree/master/Round_2/
Lisa_Sarah