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

Previous Text Analysis Code

- Code for mining (BLN and TDA) Newspapers
  - Analysis in **Python** and queries via the **Apache Spark**
- Code for mining (BLB) Books
  - Analysis in **Python** and queries via **MPI**

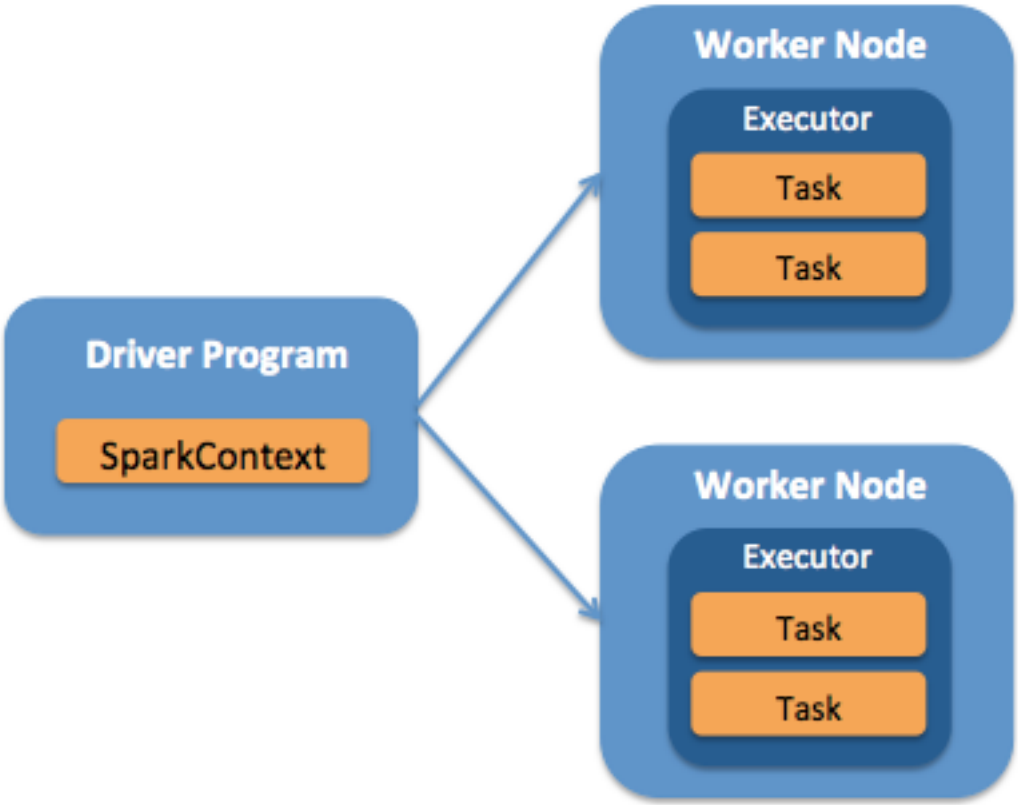
developed by RITS-UCL

- Different parallel enginees
- HPC-specific code
- Each supports only a specific XML schema and a physical representation

Apache Spark

- Open Source
- Fast general engine for large-scale data processing
- It is much faster and much easier than Hadoop MapReduce to use due its rich APIs
- Large community
- Goes far beyond batch applications to support a variety of workloads:
  - including interactive queries, streaming, machine learning, and graph processing

Programming Environment – Spark Cluster

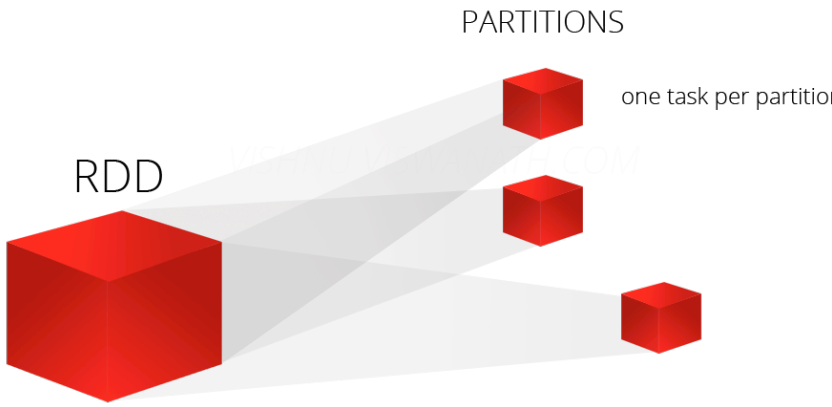


Resilient Distributed Data sets (RDDs)

- Main Spark programming abstraction
- Represent data or transformations on data
  - It is distributed collection of items - partitions
  - Read-only → they are immutable
  - Enables operations to be performed in parallel

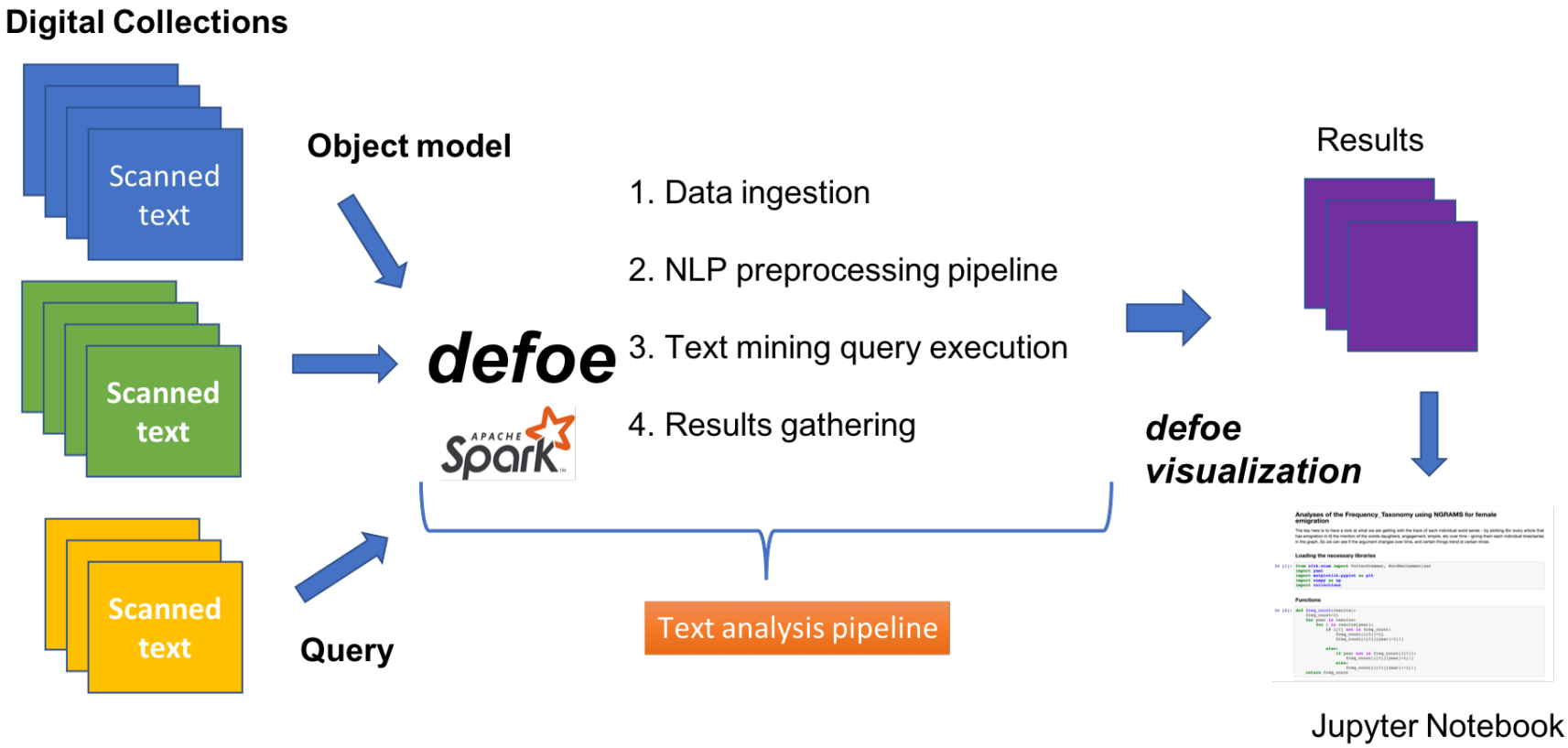
All work is expressed as either:

creating new RDDs
transforming existing RDDs
operations on RDDs to compute a result.



Each RDD is split into multiple **partitions**, which can be computed on different nodes of the cluster.

Defoe Overview



https://github.com/alan-turing-institute/defoe

https://github.com/alan-turing-institute/defoe_visualization

Defoe – Data Ingestion

It has support for newspapers and books conforming for three physical representations:
1) one XML document per issue;
2) one XML document with search results including several articles
3) one XML metadata document and a XML per page

It also supports the following digital library standards:

- METS XML
- MODS XML
- ALTO XML
- British Library-specific
- PaperPast-specific XML

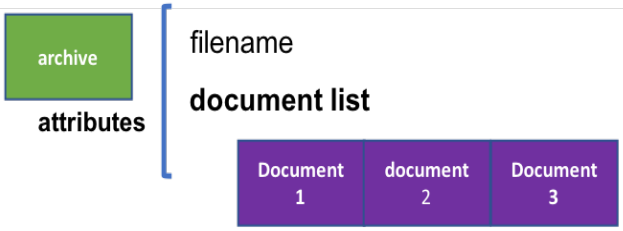
3 different object models that are used for loading data into RDD

ALTO object model

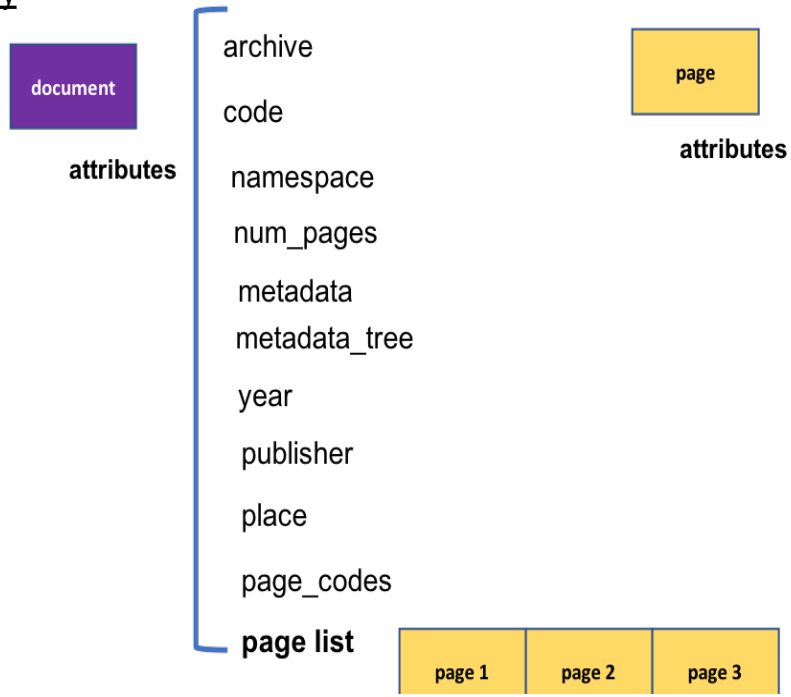
FMP!!

RDDs

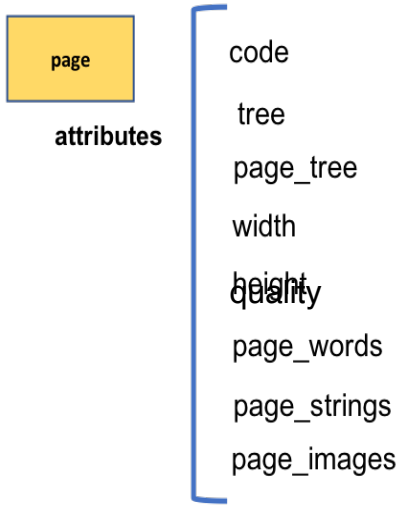
Class Archive → Representation of a zipped archive



Class Document → Representation of a metadata document (XML in METS/MODS) and an ALTO directory



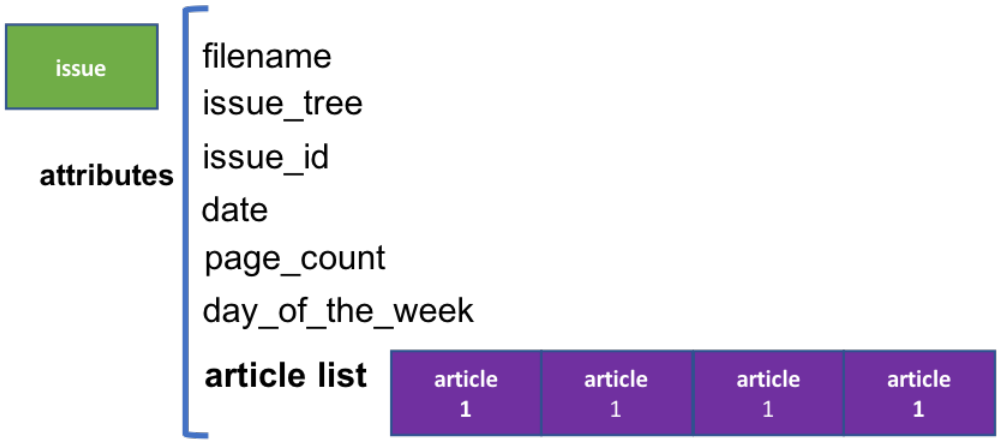
Class Page → Representation of a page (XML in ALTO).



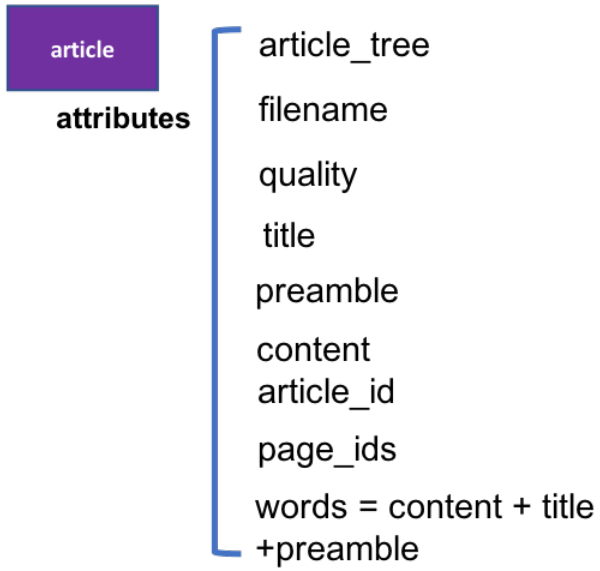

```
(urika-py27) -bash-4.2$ ls
0000408_18350101_0001.xml  0000408_18520101_mets.xml  0000408_18570101_0014.xml  0000408_18630101_0007.xml  0000408_18630101_mets.xml  0000408_18800101_0014.xml
0000408_18350101_0002.xml  0000408_18570101_0001.xml  0000408_18570101_0015.xml  0000408_18630101_0008.xml  0000408_18800101_0001.xml  0000408_18800101_0015.xml
0000408_18350101_0003.xml  0000408_18570101_0002.xml  0000408_18570101_0016.xml  0000408_18630101_0009.xml  0000408_18800101_0002.xml  0000408_18800101_0016.xml
0000408_18350101_0004.xml  0000408_18570101_0003.xml  0000408_18570101_0017.xml  0000408_18630101_0010.xml  0000408_18800101_0003.xml  0000408_18800101_0017.xml
0000408_18350101_mets.xml  0000408_18570101_0004.xml  0000408_18570101_0018.xml  0000408_18630101_0011.xml  0000408_18800101_0004.xml  0000408_18800101_0018.xml
0000408_18460101_0001.xml  0000408_18570101_0005.xml  0000408_18570101_0019.xml  0000408_18630101_0012.xml  0000408_18800101_0005.xml  0000408_18800101_0019.xml
0000408_18460101_0002.xml  0000408_18570101_0006.xml  0000408_18570101_0020.xml  0000408_18630101_0013.xml  0000408_18800101_0006.xml  0000408_18800101_0020.xml
0000408_18460101_0003.xml  0000408_18570101_0007.xml  0000408_18570101_mets.xml  0000408_18630101_0014.xml  0000408_18800101_0007.xml  0000408_18800101_mets.xml
0000408_18460101_0004.xml  0000408_18570101_0008.xml  0000408_18630101_0001.xml  0000408_18630101_0015.xml  0000408_18800101_0008.xml  0101.zip
0000408_18460101_mets.xml  0000408_18570101_0009.xml  0000408_18630101_0002.xml  0000408_18630101_0016.xml  0000408_18800101_0009.xml
0000408_18520101_0001.xml  0000408_18570101_0010.xml  0000408_18630101_0003.xml  0000408_18630101_0017.xml  0000408_18800101_0010.xml
0000408_18520101_0002.xml  0000408_18570101_0011.xml  0000408_18630101_0004.xml  0000408_18630101_0018.xml  0000408_18800101_0011.xml
0000408_18520101_0003.xml  0000408_18570101_0012.xml  0000408_18630101_0005.xml  0000408_18630101_0019.xml  0000408_18800101_0012.xml
0000408_18520101_0004.xml  0000408_18570101_0013.xml  0000408_18630101_0006.xml  0000408_18630101_0020.xml  0000408_18800101_0013.xml
(urika-py27) -bash-4.2$
```

PAPERS object model

Class Issue → Representation of an issue (XML document)
Each XML holds articles that belong to the same issue



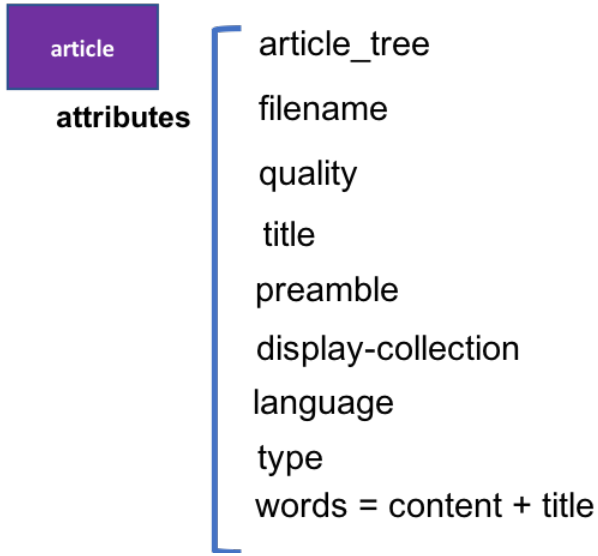
Class Article → Representation of an article



NZPP object model

Class Issue → Representation of an issue (collection of XML documents)
Each XML holds <u>one or more articles</u> that <u>belong to one or more issues</u>.

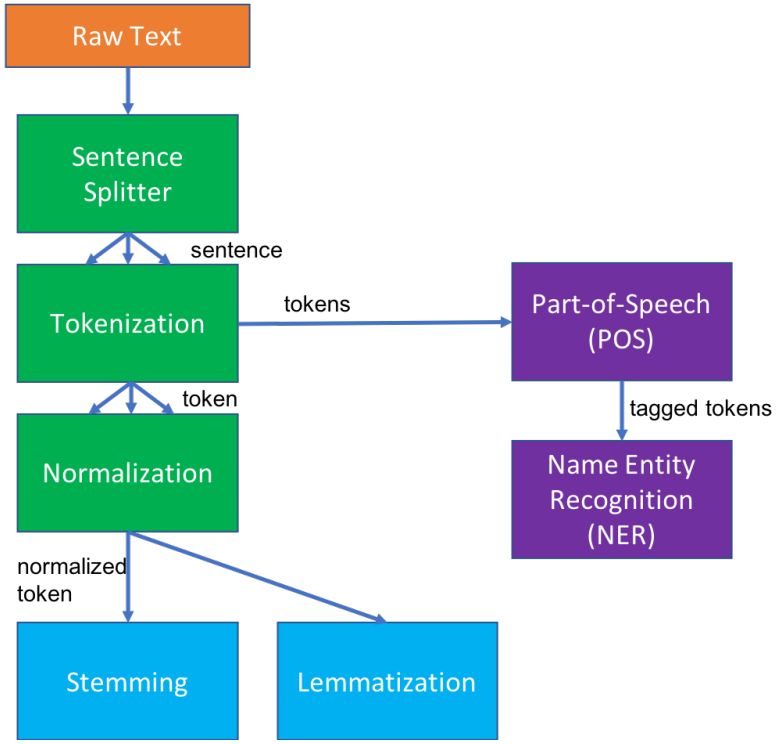
Class Article → Representation of an article



DataSets

	Dataset	Period	Structure	XML Schema	Space	Model
ATI-SE	**British Library Books (BLB)**	1510-1899	ZIP per book - XML metadata - XML per page	METS and ALTO schemas	~220GB	ALTO
ATI-SE	**British Library Newspapers (BLN)**	1714-1950	XML per issue	GALEN Schema	~1TB	PAPERS
ATI-SE	**Times Digital Archive (TDA)**	1785-2009	XML per issue	GALEN Schema	~324GB	PAPERS
ATI-SE	**Papers Past New Zealand and Pacific newspapers (NZPP)**	1839-1863	XML per 22 articles	XML from a search via an API	~4GB	NZPP
LwM	**FindMyPast (FPM)**	1752 - 1957	-XML metadata -XML per newspaper page	METS and ALTO schemas	~20TB	ALTO

Defoe - NLP Preprocessing



Introducing NLP to our queries: NLTK and spaCy

```
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 ./.)
```

Defoe - Text Mining Queries

ALTO Model

- colocates_by_year.py
- keyword_by_word.py
- keyword_by_year.py
- keyword_concordance_by_word.py
- keyword_concordance_by_year.py
- normalize.py
- ocr_quality_by_year.py
- total_documents.py
- total_pages.py
- total_words.py

PAPERS Model

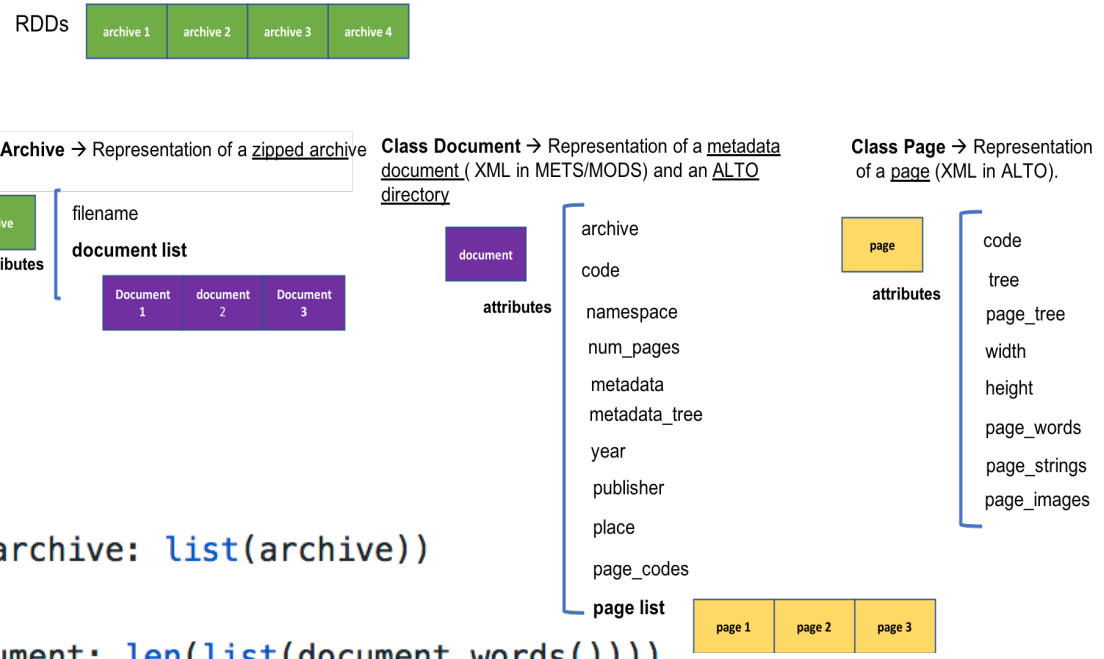
- colocates_by_year.py
- keysentence_by_year.py
- keyword_by_year.py
- keyword_concordance_by_date.py
- keywords_by_year.py
- lda_topics.py
- normalize.py
- ocr_quality_by_year.py
- target_and_keywords_by_year.py
- target_and_keywords_count_by_ye...
- target_concordance_collocation_b...
- total_articles.py
- total_issues.py
- total_words.py
- unique_words.py

NZPP Model

- keyword_by_year.py
- keyword_concordance_by_date.py
- normalize.py
- total_articles.py
- total_words.py

Defoe - Text Mining Queries

After preprocessing the raw text data, defoe executes the selected text mining query over clean RDDs.

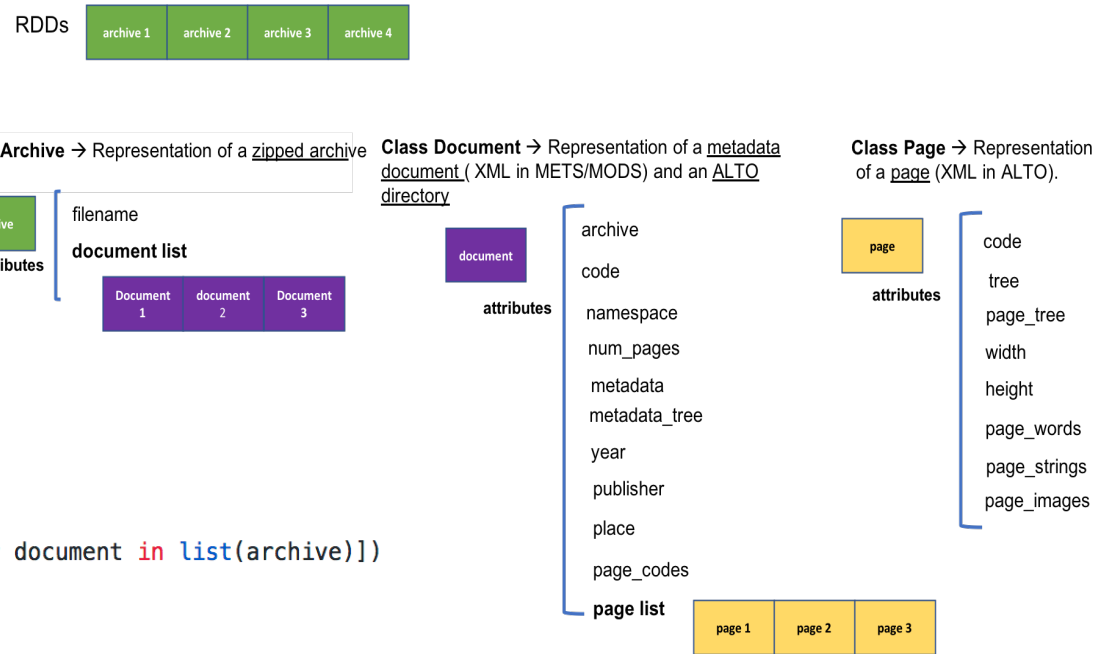



```
# [archive, archive, ...]
documents = archives.flatMap(lambda archive: list(archive))
# [num_words, num_words, ...]
num_words = documents.map(lambda document: len(list(document.words())))
result = [documents.count(), num_words.reduce(add)]
return {"num_documents": result[0],
        "num_words": result[1]}
```

total_words query: Iterates through archives and count total number of documents and total number of words.

Defoe - Text Mining Queries

After preprocessing the raw text data, defoe executes the selected text mining query over clean RDDs.




```
# [(year, document), ...]
documents = archives.flatMap(
    lambda archive: [(document.year, document) for document in list(archive)])

# [(year, [quality]), ...]
qualities = documents.flatMap(
    lambda document: [(document[0], [page.pc, calculate_words_within_dictionary(page)]) for page in document[1]])
result = qualities \
    .groupByKey() \
    .map(lambda year_q:
         (year_q[0], list(year_q[1]))) \
    .collect()
return result
```

defoe ocr_quality_by_year query: Gets the information on the OCR accuracy of each page and groups the results by year.

Submit the source code to Spark along with information about your query:

```
spark-submit --py-files defoe.zip defoe/run_query.py <DATA_FILE> <MODEL_NAME> <QUERY_NAME> <QUERY_CONFIG
```

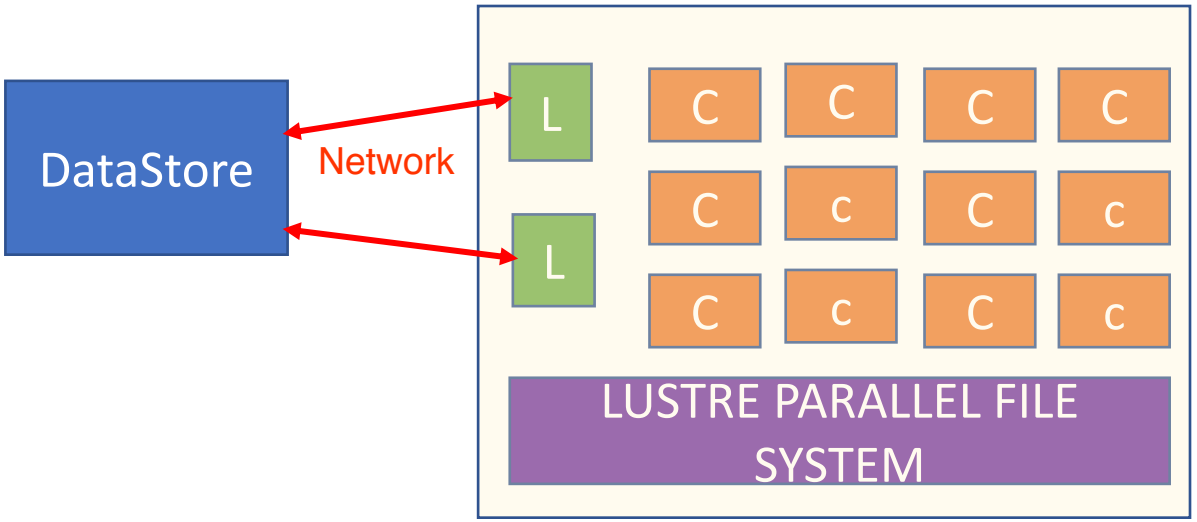
where:

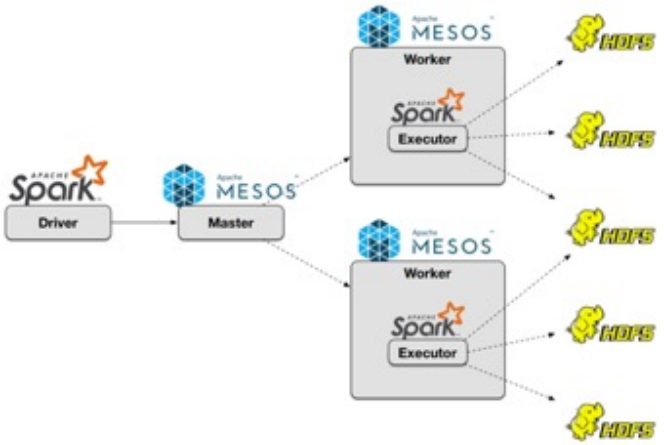
- `<DATA_FILE>` is a file that lists either URLs or file paths which are the files over which the query is to be run, one per line. Either URLs or file paths should be exclusively used, not both.
- `<MODEL_NAME>` specifies which text model is to be used, one of:
  - `books` : British Library Books
  - `papers` : British Library Newspapers
  - `fmp` : Find My Past Newspapers
  - `nzpp` : Papers Past New Zealand and Pacific newspapers
  - `generic_xml` : Arbitrary XML documents
  - For example, `books` tells the code that the data files listed in `data.txt` are books so should be parsed into a books data model.
- `<QUERY_NAME>` is the name of a Python module implementing the query to run, for example `defoe.alto.queries.find_words_group_by_word` or `defoe.papers.queries.articles_containing_words`. The query must be compatible with the chosen model.
- `<QUERY_CONFIG_FILE>` is a query-specific configuration file. This is optional and depends on the query implementation.
- `<RESULTS_FILE>` is the query results file, to hold the query results in YAML format. If omitted the default is `results.yml`.
- `<ERRORS_FILE>` is the errors file, to hold information on any errors in YAML format. If omitted the default is `errors.yml`.
- `<NUM_CORES>` is the number of computer processor cores requested for the job. If omitted the default is 1.

HPC Environment

Alan Turing Institute's **Cray Urika-GX system**

- High-performance analytics cluster
- Pre-integrated stack of popular analytics packages:
  - **Apache Spark**, Apache Hadoop, Jupyter Notebooks, etc.
- 12 computing nodes: 36 Broadwell CPUs and 256GB – 2 login nodes
- 60TB of storage – HDFS & Lustre
- Link: https://www.cray.com/products/analytics/urika-gx





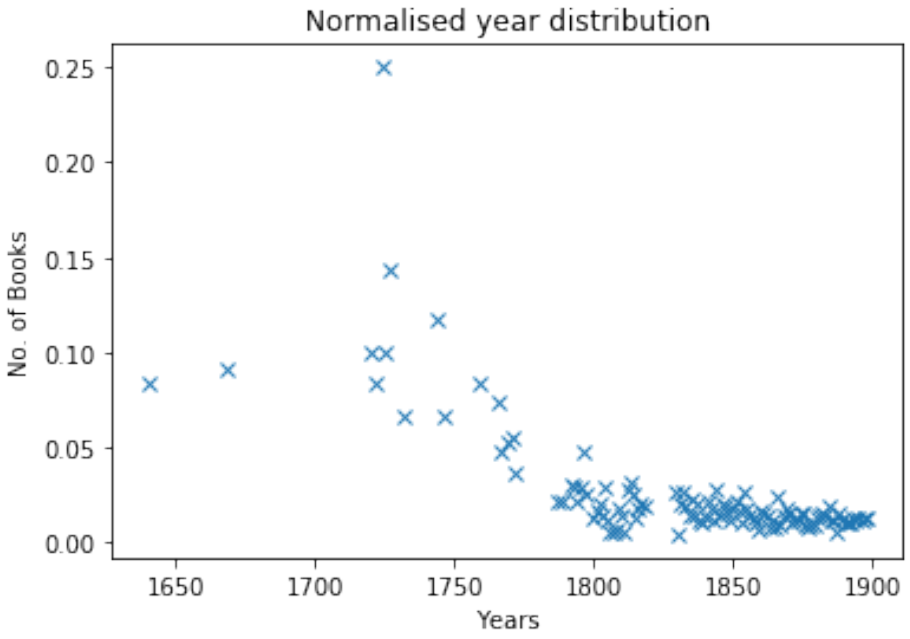
Stranger Danger

- Detect the origin of ‘Stranger Danger’
  - *colocates_by_year* query:
    - This query searches for for sentences where the words “*stranger*” and “*danger*” (**matching criteria**) appear within the same sentence.
    - Results are grouped by the publication dates.

- Jupyter Notebook (*)
  - compare results, plot them by year
  - normalise the results to account for increased used of the phrase sentiment analyses
  - visualise which words appear more often near the phrase

(*) https://github.com/alan-turing-institute/defoe_visualization/blob/master/Stranger_Danger/Stranger_Danger.ipynb

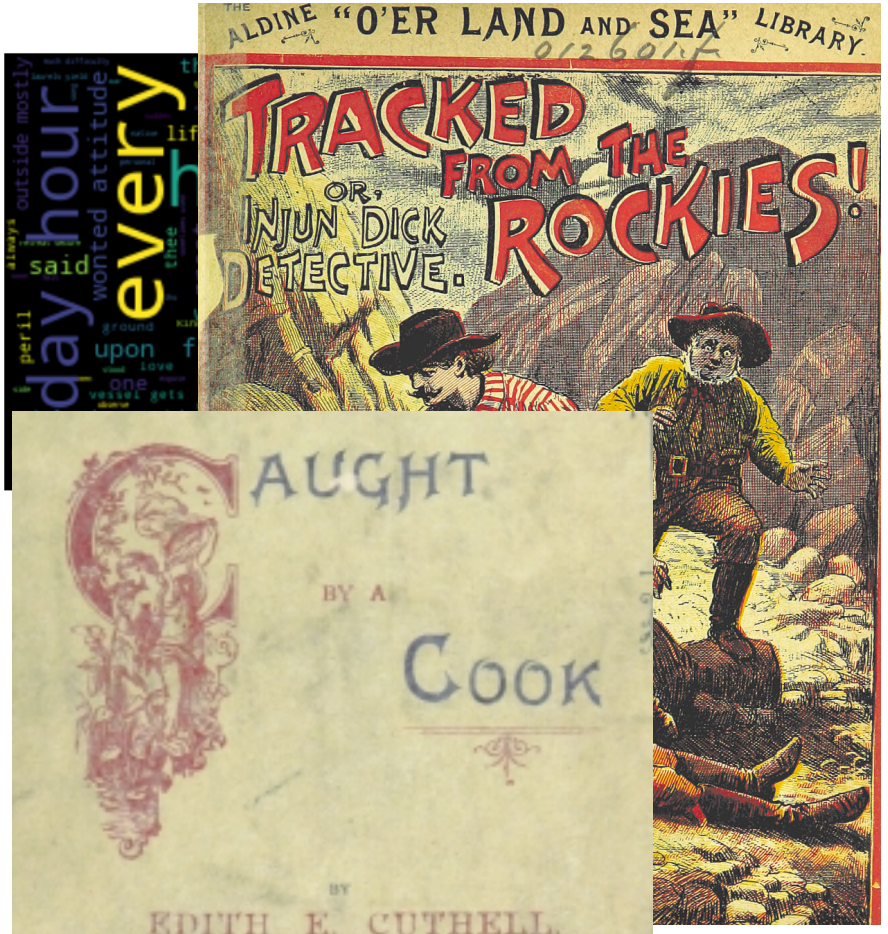
Exploration of Stranger Danger



Visualization of how the 'Stranger Danger' terms are affected by the way that the number of books were published

Getting the first book, which both terms together:
"Caught by a Cook (1895)", Page: 105.
Sentence: "*who knew most of the villagers by sight perceived a stranger danger*."

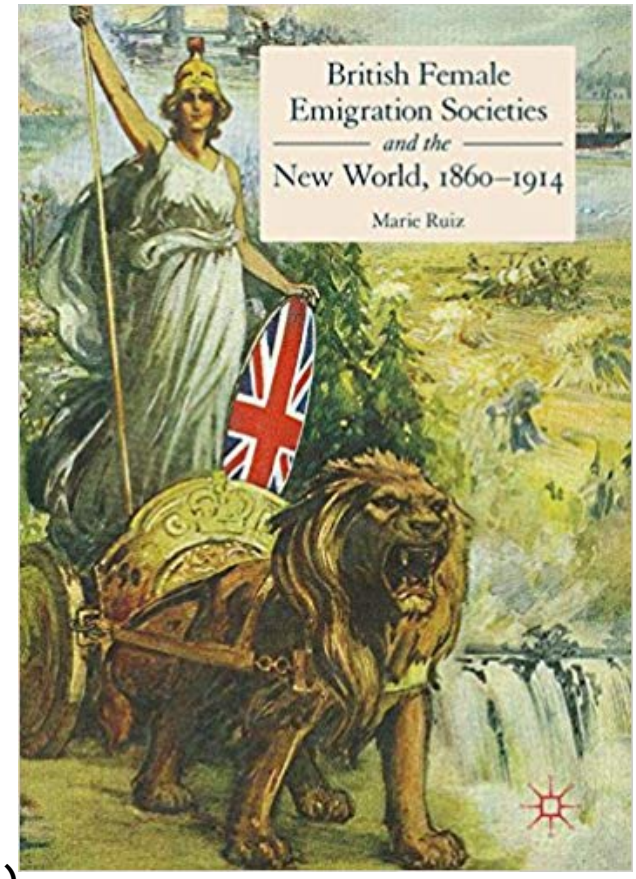
Getting the total number of matching sentences, and the book which has the max. numb of them:
Total sentences found is 1038
The book "*Aldine "O'er Land and Sea"* (1890) ,
has the max number of sentences: 29



Female Emigration

Project Oceanic Exchanges
(https://oceanicexchanges.org/)

Case Study: Mine the TDA and BLN archives for attitudes towards female emigration from Great Britain to the 'Colonies' and North America from 1850 to 1914



Normalised frequencies of female emigration societies ***keysentence_by_year(*)*** & ***normalize*** queries

Normalised frequencies of taxonomy terms relating to female emigration: **`target_and_keywords_by_year(*)`** **`target_and_keywords_count_by_year(*)`** **`& normalize`** queries

(*) new queries

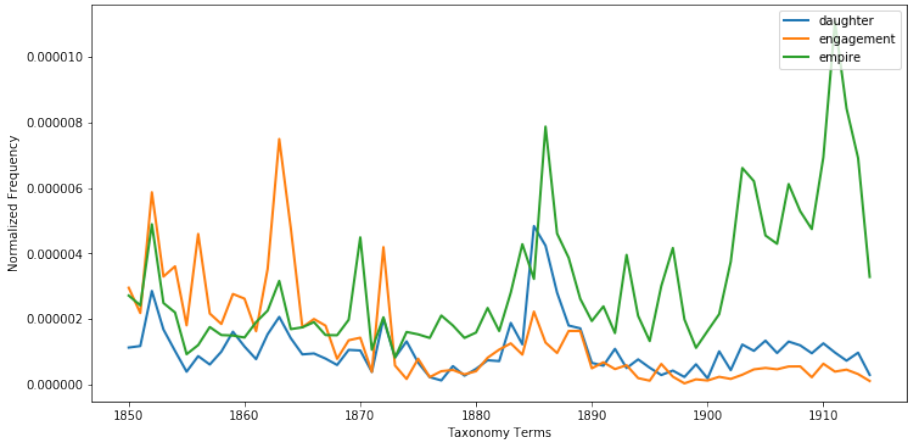
https://github.com/alan-turing-institute/defoe_visualization/tree/master/Female_Emigration

Exploration of Female Emigration

Some of the taxonomy terms

emigration
bookkeeping
Colony
Colonies
Colonial
daughters
engagement
empire
British empire
failure
female labourer
female welfare
feminine
genteel
good character
governess
guardian
guardianship
happiness
hardship

N-grams of female emigration terms co-located with "emigration"



Using TDA corpus:
Only newspapers
From 1850 to 1914.

Normalised frequencies of female societies

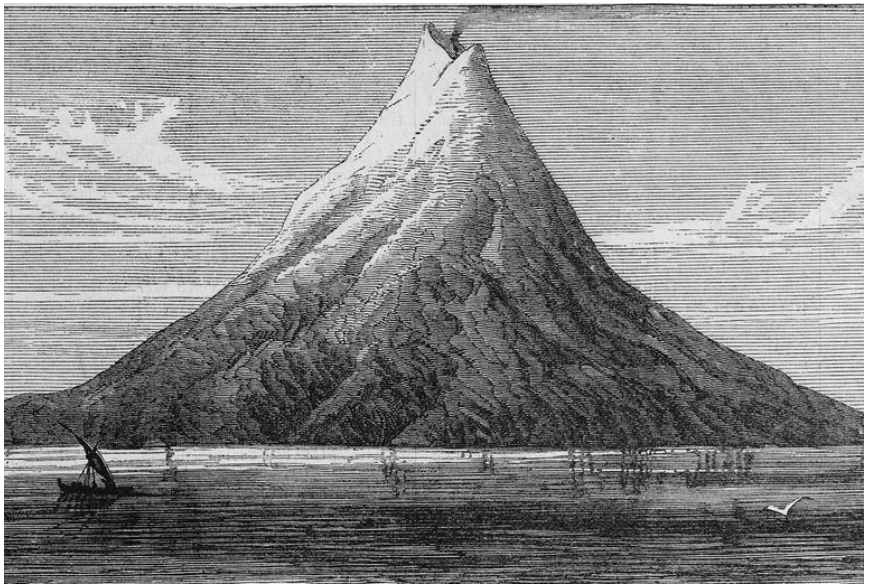
Some of the societies

South African Colonisation Society
Juvenile Emigration Society
Child Emigration Society
Church Emigration Society
Self Help Emigration Society
East End Emigration Fund
Church Army Emigration Department
Carlton Emigration Society to Canada

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.

Case study →Identify this eruption using BLN, TDA and NZPP papers from late 1883



`keyword_and_concordance_by_date` `query:` searches for occurrences of "*krakatoa*" and "*krakatua*" and returns information on each matching article.

https://github.com/alan-turing-institute/defoe_visualization/tree/master/Krakatoa_1883

Exploration of Krakatoa 1883

- Results from the query in YAML files

	date	newspaper title	search term	sequence	text
2	1883-05-24	0000453- The Evening Telegraph	krakatoa	1	VOLCANIC ERUPTION. Rata via, Thursday. A violent eruption ha* be
3	1883-05-25	0000327- The Derby Daily Telegraph	krakatoa	1	VOLCANIC ERUPTION IN THE STRAITS OF SUNDA. A Reuter's teleg
4	1883-08-27	0000321- Nottingham Evening Post	krakatoa	1	ALARMING VOLCANIC DISTURBANCES. DISTURBANCES. A Calami
5	1883-08-27	0000325- The Citizen Gloucester	krakatoa	1	TERRIBLE VOLCANIC DISTURBANCE IN THE WEST INDIES : A VILL
6	1883-08-28	0000321- Nottingham Evening Post	krakatoa	1	THE VOLCANIC ERUPTION. r • -Manager. 2473 c " OFFICII,"^s"~ >
7	1883-08-28	0000452- Edinburgh Evening News	krakatoa	1	GREAT VOLCANIC ERUPTION IN THE EAST INDIES. A Keuter's teleg
8	1883-08-28	0000325- The Citizen Gloucester	krakatoa	1	THE VOLCANIC DISTURBANCES IN THE EAST INDIES. Batavia, Aug
9	1883-08-28	0000327- The Derby Daily Telegraph	krakatoa	1	SUMMARY. ®be Her bp ®axlg ®eUjraplj DERBY, August 28, 1883. Tb
10	1883-08-29	0000327- The Derby Daily Telegraph	krakatoa	1	LATEST NEWS. "Telegraph" Officii, 1.0 P.M. MINISTERIAL CRISIS IN
11	1883-08-29	0000325- The Citizen Gloucester	krakatoa	1	THE VOLCANIC DISTURBANCES IN JAVA: A TOWN DESTROYED BY
12	1883-08-29	0000321- Nottingham Evening Post	krakatoa	1	A TOWN COMPLETELY DESTROYED. S^OIALjEDITION.! POST" OFF

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.

Exploring OCR quality

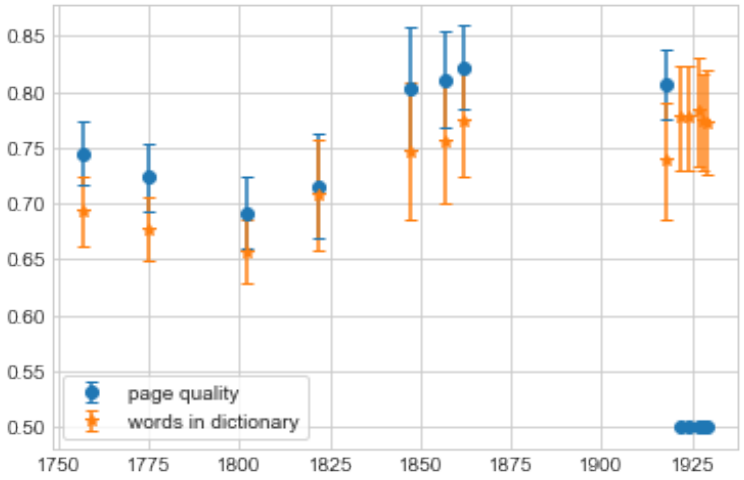
- Extract the OCR Page Quality
- Calculate the normalized number of words found in a dictionary
- Grouping results by year

```
spark-submit --py-files defoe.zip defoe/run_query.py data.txt model
defoe.alto.queries.ocr_quality_by_year -r results -n 324
```

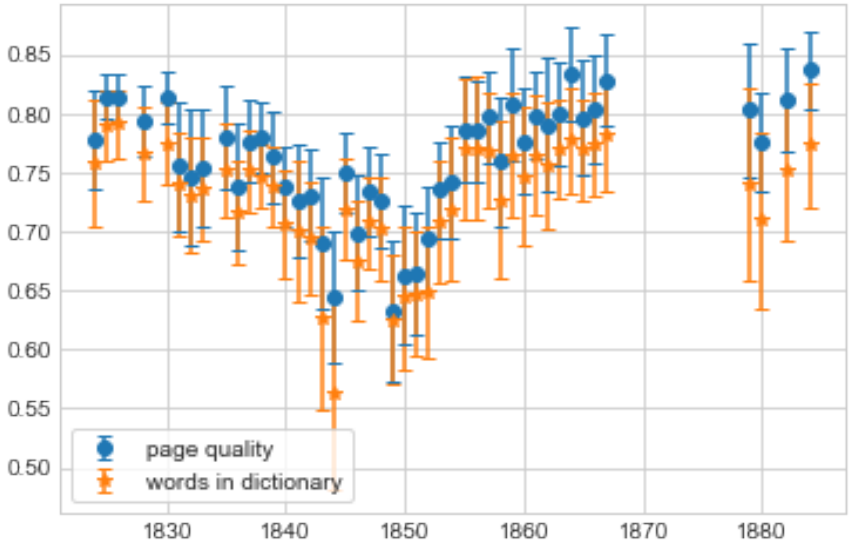
1757:
- ['0.751', '68']
- ['0.748', '66']
- ['0.692', '62']
- ['0.743', '71']
- ['0.785', '73']

1775:
- ['0.703', '67']
- ['0.726', '68']
- ['0.737', '70']
- ['0.743', '71']
- ['0.752', '69']

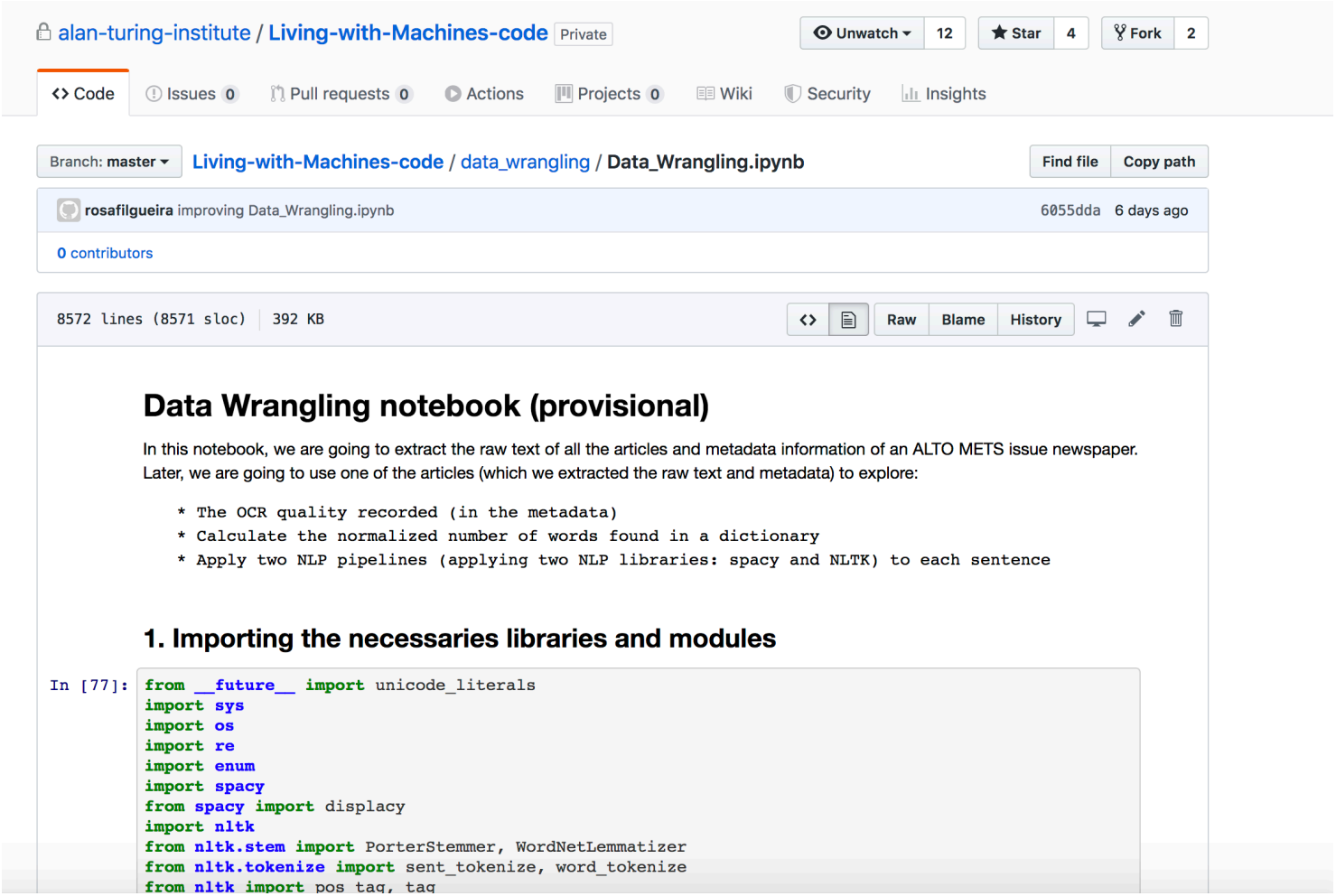
1922:
- ['0.503', '66']
- ['0.500', '80']
- ['0.500', '80']
- ['0.500', '82']
- ['0.500', '80']
- ['0.500', '80']



Small subsample of Lancanshire MET18



Small subsample of Dorsert MET18



* Extract the raw text of all the articles and metadata - (ALTO METS) issue newspaper
* OCR quality recorded (in the metadata)
* Calculate the normalized number of words found in a dictionary
* Apply two NLP pipelines (applying two NLP libraries: spacy and NLTK) to each sentence

https://github.com/alan-turing-institute/Living-with-Machines-code/blob/master/data_wrangling/Data_Wrangling.ipynb