Branch: master Big-Data / README.md Find file Copy path

jb-diplom links to images 4b2503a 1 minute ago

1 contributor

148 lines (126 sloc) 13.1 KB Raw Blame History

Big-Data

Python project to scrape RSS news feeds, derive topic maps and analye sentiment. Multiple visualizations are displayed in a Jupyter Notebook

Getting Started

The easiest approach is to clone this repository (https://github.com/jb-diplom/Big-Data/edit/master/README.md) and test the Notebook RSSNewsIdentification.ipynb on Jupyter.

Prerequisites

To run all the implemented features you'll need:

- Python 3.7 or newer
- Jupyter (version 6.0.3 was used for the project)
- Anaconda (version 1.9.12 was used for the project)
- Gensim (for corpora, topic modelling, word wmbedding tools and downloade)

- Natural Language Toolkit (NLTK) for text processing (Lemmatization)
- VaderSentiment (Sentiment analysis)
- Bokeh (visualization widgets and coloring)
- Feedparser (scraping of RSS Feeds)
- pyLDAvis (visualization of LDA analysis)
- ipyvolume (3d interactive scatterplots)
- ipydatawidgets (required for the interactive parts of the notebook and to ensure correct display of the progress bars)
- Beautiful Soup (parsing and cleaning of HTML/XML)
- sklearn (spectral clustering and dimension reduction using the PCA method)

Installation

Several Python packages are necessary to complete analysis/computations and for an adequate visualization in Jupyter notebook

- `conda install bokeh`
- `conda install feedparser`
- `conda install genism`
- `conda install -c conda-forge pyldavis` see also https://github.com/bmabey/pyLDAvis
- `conda install -c conda-forge ipyvolume`
- `conda install -c conda-forge nodejs`
- `conda install -c conda-forge ipywidgets`
- `conda install -c conda-forge ipydatawidgets`
- `import nltk`
- `nltk.download()` --> choose Models --> vader_lexicon
- `nltk.download()` --> choose Models --> punkt
- `nltk.download()` --> choose Corpora --> wordnet

- `nltk.download()` --> choose Corpora --> stopwords

Usage

The central element for accessing the funtionality of the project work is by loading the RSSNewsIdentification.ipynb Jupyter Notebook. Virtually all of the useful configurative settings can be specified or modified in the run parameters, an editable Python dictionary in a cell near the beginning of the notebook. Plausible parameters are documented as comments in the code (see below).

The Run Parameters

```
runParams={'allFeeds':       getFeedDict(),   # A collection of 50 URLs for RSS Feeds, where the names can be
chosen freely
                                              # example: {'Buzzfeed': 'https://www.buzzfeed.com/world.xml',
                                              #            'Al Jazeera':
'http://www.aljazeera.com/xml/rss/all.xml'}
           'collectFeeds':   False,           # Set to True to do more data-scraping
           'inputDir':       './data',        # Directory containing pre-collected RSS-Feed data in pickle
format
           'articleLimit':   10000,           # Optional specification limiting number of articles to process
(for test
                                              # purposes only)
           'numAuthDispl':   30,              # specify limit of number of authors to display/plot
           'numTagsDispl':   30,              # specify limit of number of tags to display/plot
           'numTopics':      30,              # specify limit of number of topics to be found via TFIDF
Vectorization
           'numLDATopics':   30,              # specify limit of number of topics to be found via Latent
Dirichlet Allocation
           'numTopicsDispl': 30,              # specify limit of number of topics to display/plot
           'removeTopics':   True,            # specify if articles now referenced by topics should be removed
(saves
                                              # computation time later)

           'ngramRange':     (3,3),           # define min and max ngrams as tuple for deriving topic maps
           'weModel':        'glove-wiki-gigaword-50', # Word Embedding model for soft cosine similarity
           'weModel':         'glove-wiki-gigaword-50', # Word Embedding model for soft cosine similarity
calculation
                                              # options are e.g. glove-wiki-gigaword-50,fasttext-wiki-news-
subwords-300
                                              # CAUTION: although higher dimension models produce more nuanced
results, load
                                              # times are VERY long. In particular the first time use requires
a download of
                                              # massive amounts of data which can easily take as much as an
hour
           'thresholdFuzzy': 60,              # define cut-off threshold for fuzzy (Levenshtein Distance
comparison) in percent
           'thresholdCosine':0.60,            # define cut-off threshold for soft cosine similarity as fraction
(0.0 to 1.0)
           'fuzzyDocLimit':  None,            # optional integer limit to number of articles used for fuzzy
relevance of
                                              # topics (only for test purposes: default None)
           'matrixDir':      './outdata',     # relative path for saving matrices to file
           'saveMatrix':     False}           # whether to save matrix to file or not
```

A few Words on Performance

Due to the potentially large volume of data involved, running the notebook in its entirety can take many hours. This may yield very interesting results, but you may alternatively be interested in a quick overview. Some key parameters and options have been designed to help you (and especially me) to run through tests fairly quickly. The following settings are critical for performance and running times:

- `weModel` : Tests have shown - not surprisingly - far better quality and nuance of results with higher dimension word embeddings. However, start times and running times in calculating the soft cosine similarity matrix easily run into hours (or a dead end) with `fasttext-wiki-news-subwords-300` . For speed, definitely use the preconfigured `glove-wiki-gigaword-50` .
- `removeTopics` : If set to `False` , then the running time will be typically (depending also on number of topics required and specified thresholds) a factor 3 or 4 longer. Theoretically you may miss out on some possible clustering which the

soft cosine method finds but TF-IDF doesn't by the removal of articles, but it's probably sufficient to specify a few topics more to compensate and save hours of computational time, by using this flag.

- `collectFeeds` : If you just want to play with the test data and don't need fresh articles, the scraping stage can be skipped by setting this to `False`
- `articleLimit` : By setting this to 1000 or less you can get a quick overview of the analysis results. If you want to be sure to miss none of the article content, you have to set this to `None` , though. Probably time to grab a capuccino, in that case :-)
- `thresholdFuzzy` and `thresholdCosine` : in practice you get to see some optically interesting results with lower values here. However this costs time and tends to include clustering which will effectively produce more noise than anything else.

Test Data

Two crucial sets of data are supplied to help the user get up and running very quickly.

1. A set of 50 fully tested RSS-Feed URLs, which can be conveniently loaded from the function `getFeedDict()` . The user can simply specify custom values in an equivalently structured dictionary, .e.g.

```
{'Buzzfeed': 'https://www.buzzfeed.com/world.xml',
'Al Jazeera': 'http://www.aljazeera.com/xml/rss/all.xml',
'The Guardian': 'https://www.theguardian.com/world/rss',
'CNBC': 'https://www.cnbc.com/id/100727362/device/rss/rss.html',
'RT': 'https://www.rt.com/rss/news/'}
```

2. A corpus of over 11000 articles which have been collected between 02.05.2020 and 31.05.2020 daily from the 50 RSS-feeds mentioned in 1.
   An article limit `articleLimit` can be specified for test purposes to take a randomized sample of documents from the test corpus. It should be mentioned that with 11000 documents the runtime for the complete notebook may run into into several hours, dependant on CPU power and RAM.

Features

Initially the user can load data from a pre-configured set of 50 RSS-Feeds, or specify custom feeds for scraping. This takes typically around 75s (can be skipped using runParam `collectFeeds` . Additionally there is a large corpus of test data in the directory data directory which may be loaded for test purposes.

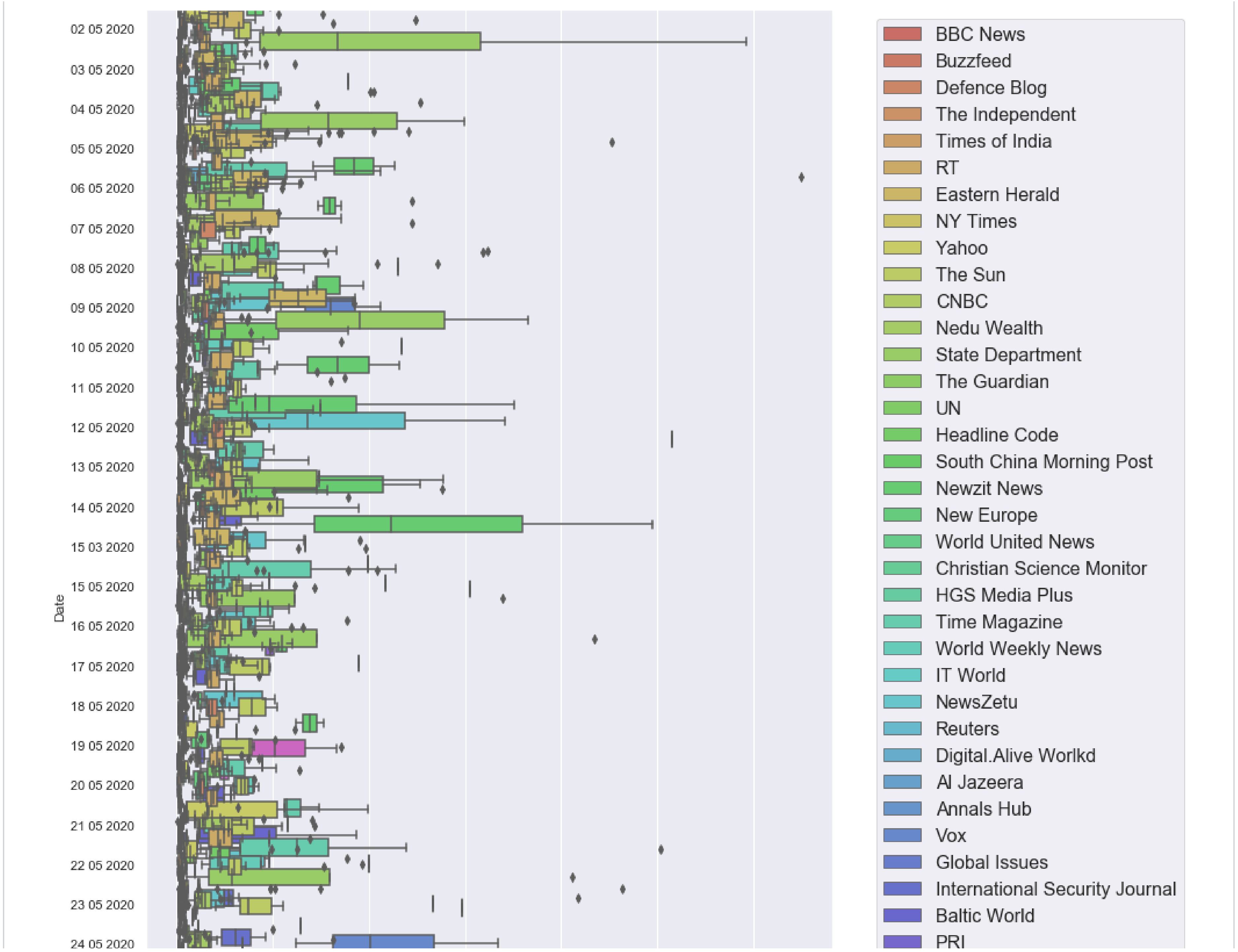
The steps which can be walked through are:

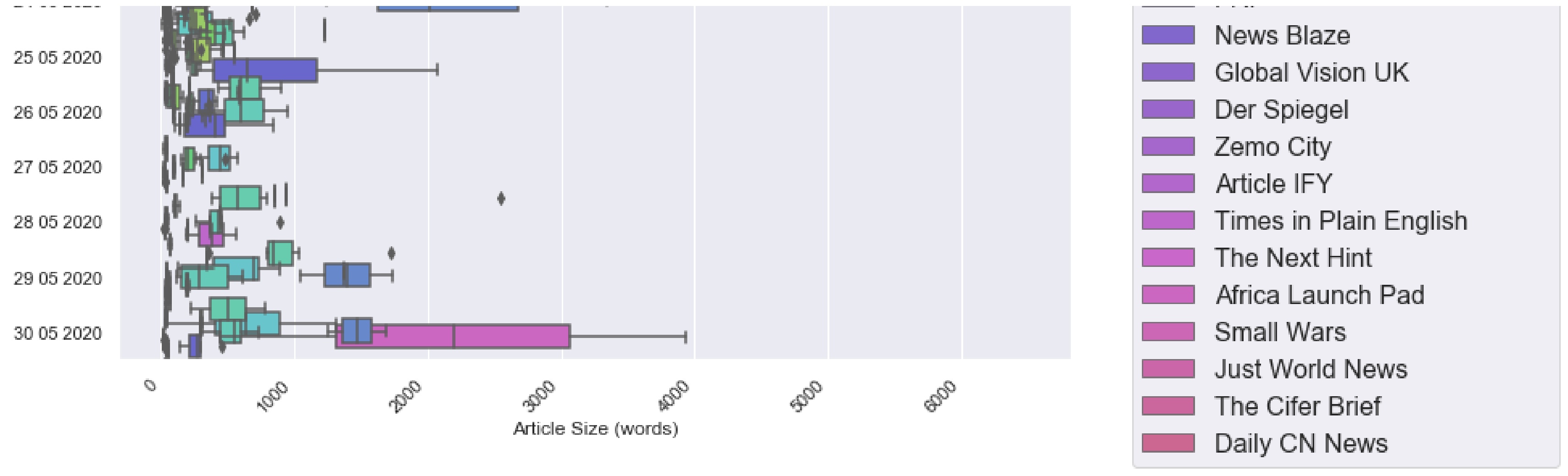
1. Custom configuration of `runParam`
2. Data collection from the specified RSS-feeds. The collected data is saved in pickle format to the data directory
3. All data from the test corpus in data directory are loaded. The number of articles may be reduced according to runParam articleLimit`. The HTML content is parsed and stripped of tags, some data cleaning occurs to remove some articles which are known to not contain sinle stories, but rather collections of all the day's news.
4. A brief summary of the articles scraped/loaded is displayed in tabular form as a quick check (note that a more thorough view of the content is available later using the content viewer
5. To get an impression of the range of publish dates and article sizes per feed a box-plot is created
6. A statistical overview of the most prolific authors is displayed
7. Statistical overviews of the most used tags - as specified by the publishers - are displayed (bar chart and scatterplot)
8. Stop words are removed from the content, TF-IDF vectorization with Lemmatization is used to convert the amassed content of the corpus into documents to a matrix of TF-IDF features deriving the specified number of topics. TF-IDF Transformer additionally scales down the impact of tokens that occur very frequently in the given corpus and that are hence empirically less informative than features occuring in a small fraction of the corpus. The user can specify not only the preferred number of features but also the n-gram range
9. The topics from 8. are compared using a Levenshtein Distance calculation to each article in the corpus. Topics within a relevant tolerance (a threshold level is specified per parameter) are held as a reference with each article including the value for relevance as a percentage. Articles may have - any often do have - references to several topics.
10. An overview of the most frequent topics and frequency of occurence for each feed publisher are displayed in a scatter plot,
11. The overall topic frequency is visualized in a histogram

12. The soft cosine similarity matrix is derived for the corpus to ascertain similarities between every combinational pair of articles. This employs word embeddings for the calculation of the comparison vectors (see also https://www.machinelearningplus.com/nlp/cosine-similarity/)
13. Using the soft cosine similarity matrix, multiple 3d visualizations of the clusters are produced. The colourization is simply achieved through the application of spectral clustering, a representation of the coordinates of the "bubbles" is achieved through dimension reduction using the PCA method, to create an optical separation according to the cosine similarity. The size of each bubble is calculated to be proportional to the highest topical relevance from the article's row taken from the similarity matrix. The user may specify the number of topics to display and the cut-off threshold for deciding if an article should be represented in the plot.
14. A specified number of topics are representen in the pyLDAvis visualization.
15. A sentiment analysis of each article is conducted using Vader. A 3d scatterplot of the sentiment for topically grouped sets of articles. The coordinates are derived from the positiv, negativ and neutral sentiment values. The colour represents the compound sentiment. The size of the bubbles are exponentially proportional to the strength of topicality (cf topical relevance percentage in 9. above) for the preferred topic which can be conveniently chosen from a pull-down list of possible topics. A change of topic triggers a redraw of the 3d-visualization.
16. Finally the user can browse through the article content grouped by topic. The article content together with a subset of meta-data for the article is displayed in a viewer.

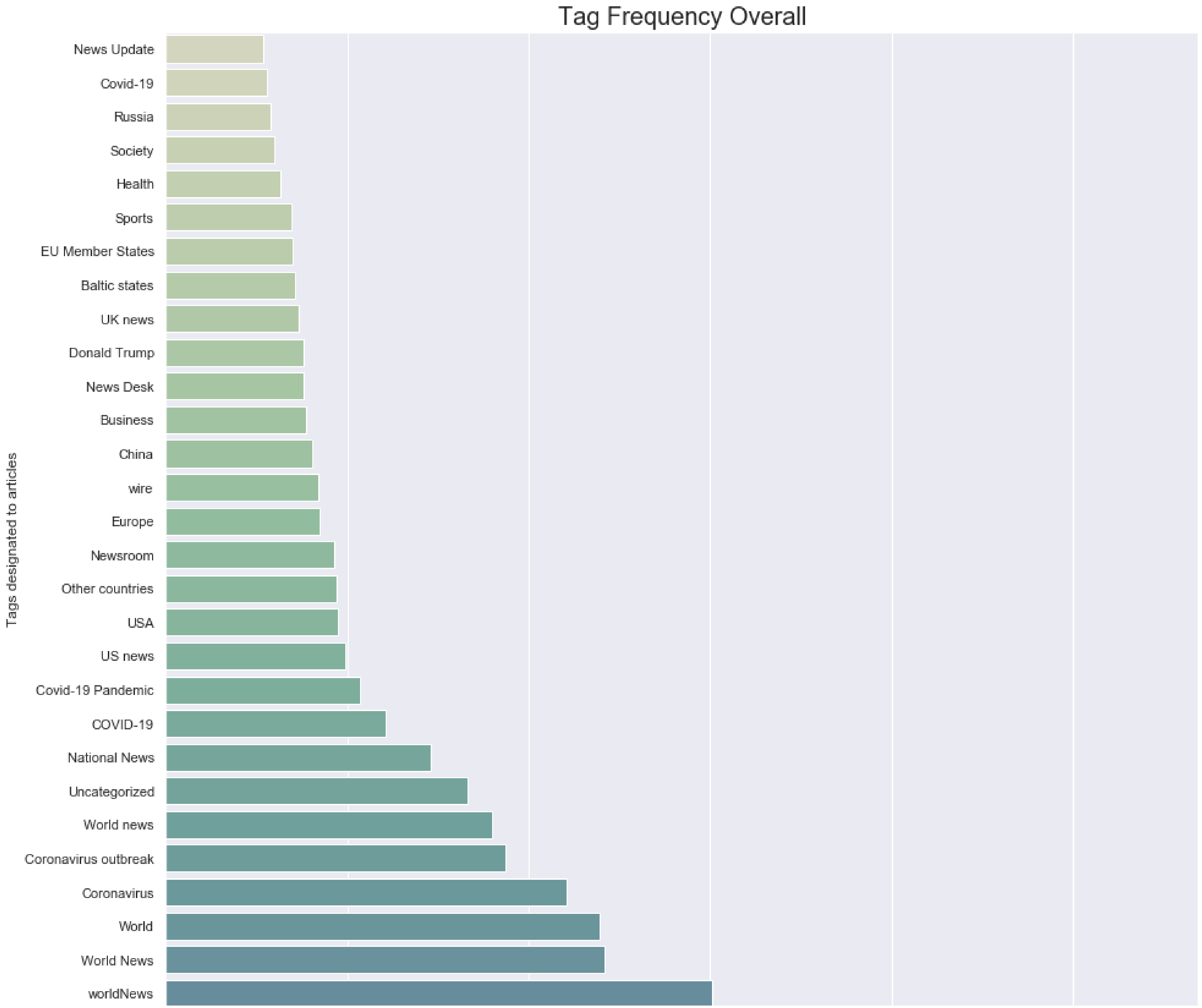
Some Graphics from the Notebook

Article Sizes

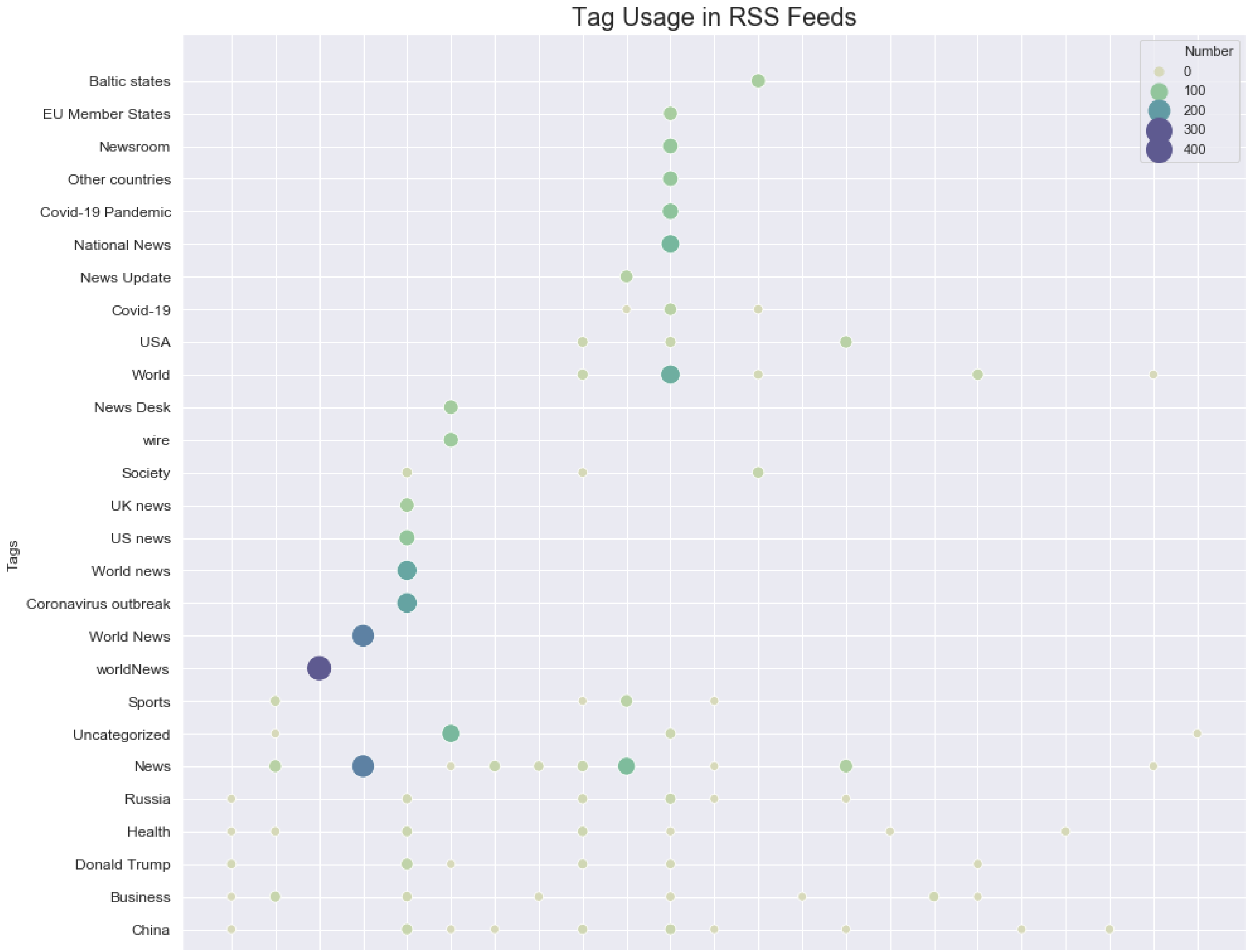

02 05 2020
03 05 2020
04 05 2020
05 05 2020
06 05 2020
07 05 2020
08 05 2020
09 05 2020
10 05 2020
11 05 2020
12 05 2020
13 05 2020
14 05 2020
15 03 2020
15 05 2020
16 05 2020
17 05 2020
18 05 2020
19 05 2020
20 05 2020
21 05 2020
22 05 2020
23 05 2020
24 05 2020
Date
BBC News
Buzzfeed
Defence Blog
The Independent
Times of India
RT
Eastern Herald
NY Times
Yahoo
The Sun
CNBC
Nedu Wealth
State Department
The Guardian
UN
Headline Code
South China Morning Post
Newzit News
New Europe
World United News
Christian Science Monitor
HGS Media Plus
Time Magazine
World Weekly News
IT World
NewsZetu
Reuters
Digital.Alive Worlkd
Al Jazeera
Annals Hub
Vox
Global Issues
International Security Journal
Baltic World
PRI

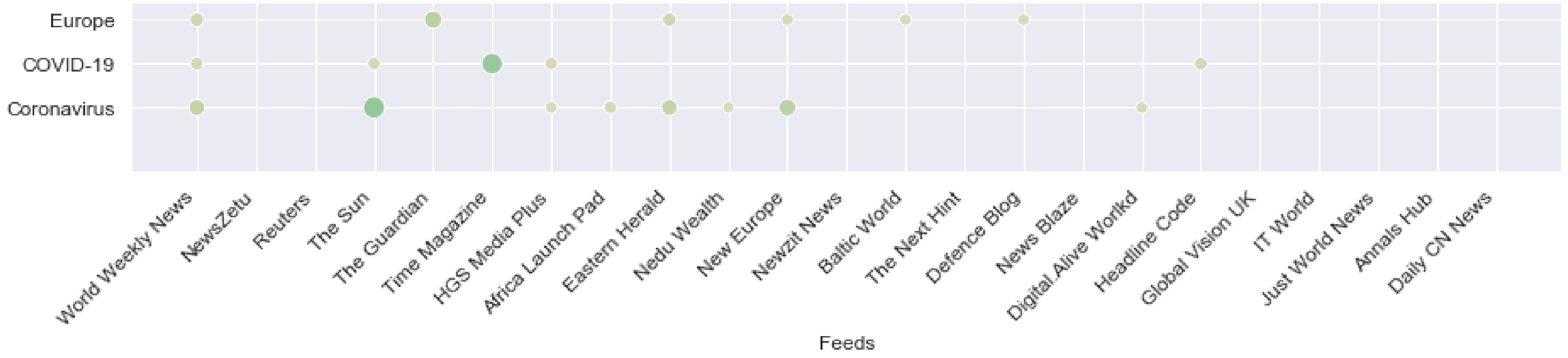

25 05 2020
26 05 2020
27 05 2020
28 05 2020
29 05 2020
30 05 2020
0
1000
2000
3000
4000
5000
6000
Article Size (words)
News Blaze
Global Vision UK
Der Spiegel
Zemo City
Article IFY
Times in Plain English
The Next Hint
Africa Launch Pad
Small Wars
Just World News
The Cifer Brief
Daily CN News

Tag Usage

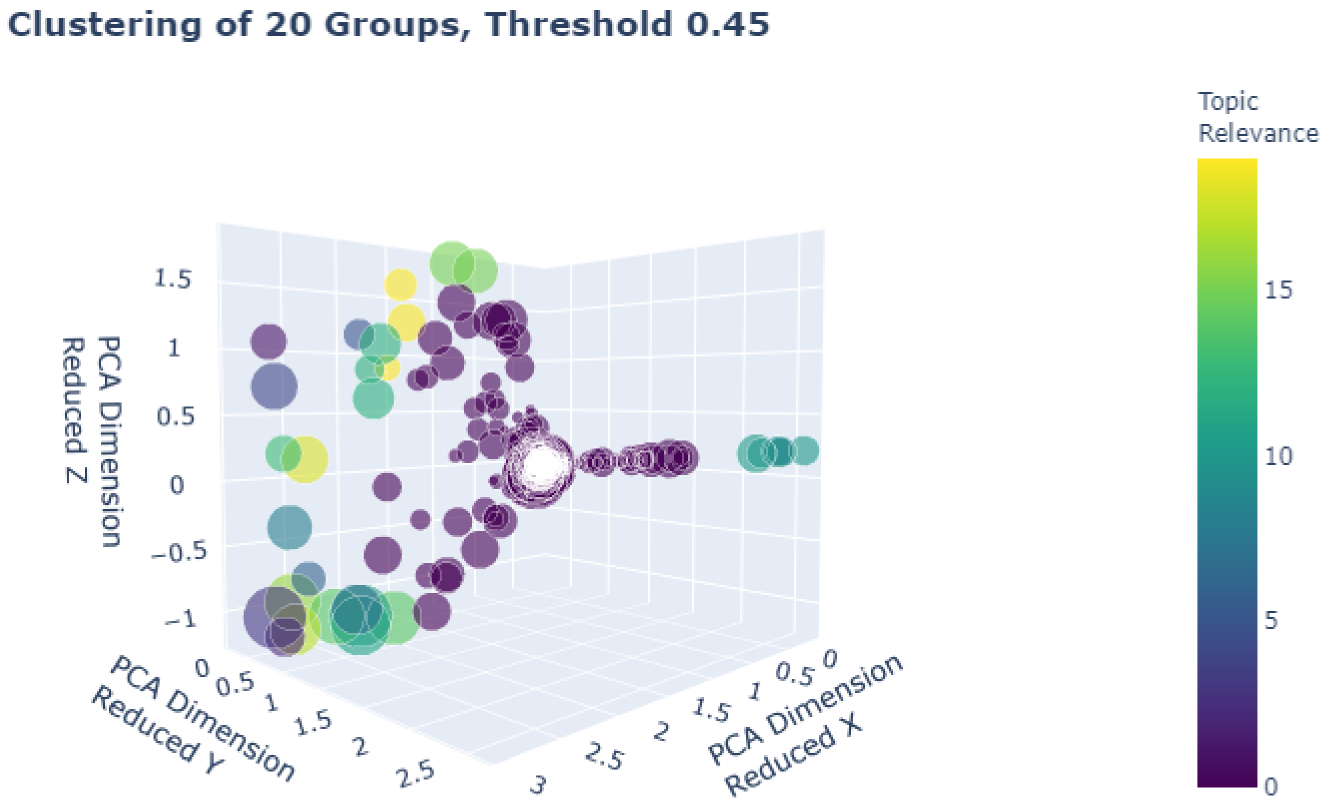

Tag Frequency Overall
Tags designated to articles
News Update
Covid-19
Russia
Society
Health
Sports
EU Member States
Baltic states
UK news
Donald Trump
News Desk
Business
China
wire
Europe
Newsroom
Other countries
USA
US news
Covid-19 Pandemic
COVID-19
National News
Uncategorized
World news
Coronavirus outbreak
Coronavirus
World
World News
worldNews

News
Tags
0
100
200
300
400
500

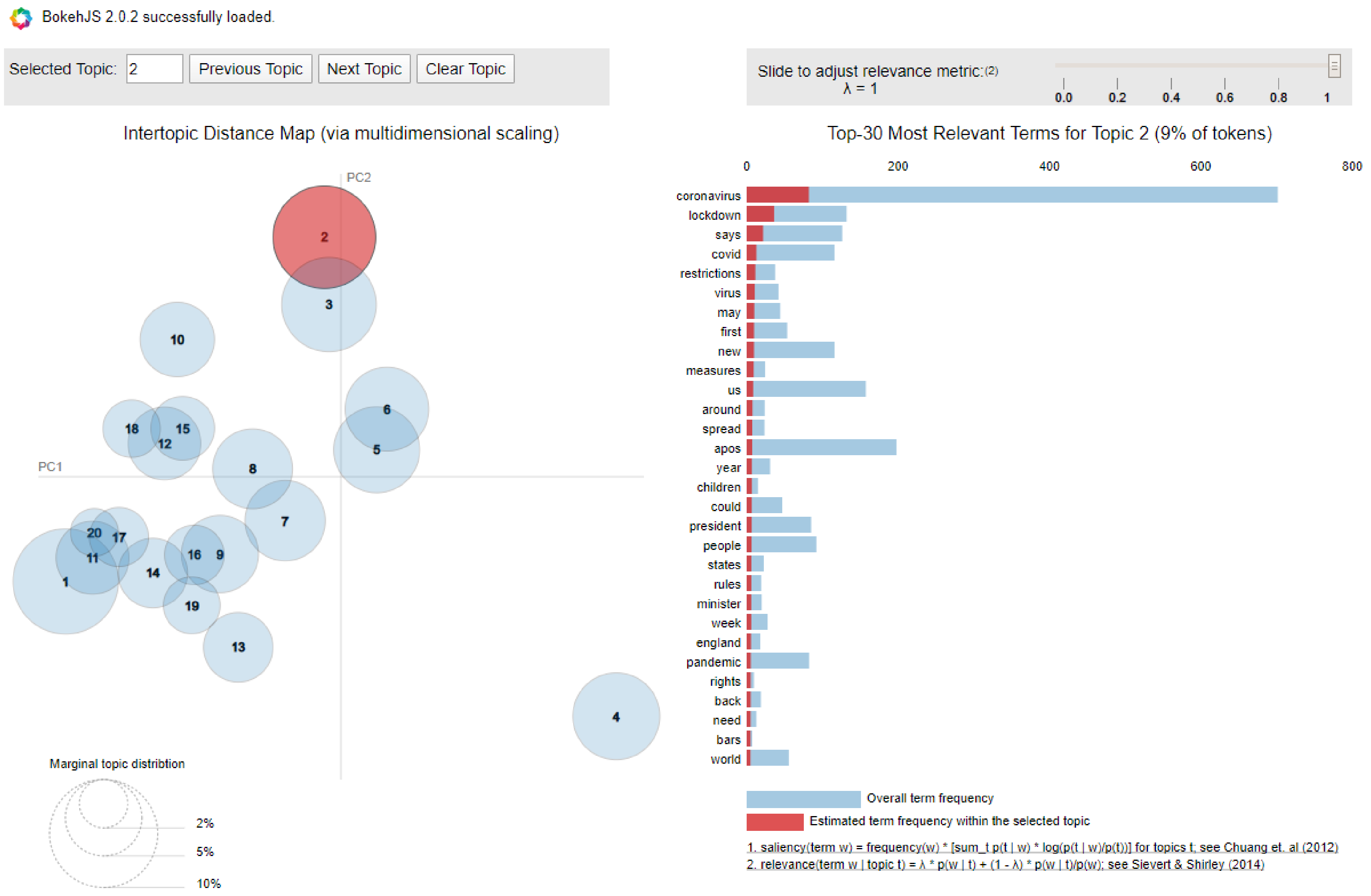

Tag Usage in RSS Feeds
Number
0
100
200
300
400
Tags
Baltic states
EU Member States
Newsroom
Other countries
Covid-19 Pandemic
National News
News Update
Covid-19
USA
World
News Desk
wire
Society
UK news
US news
World news
Coronavirus outbreak
World News
worldNews
Sports
Uncategorized
News
Russia
Health
Donald Trump
Business
China



Cosine Similarity (3d interactive plot)

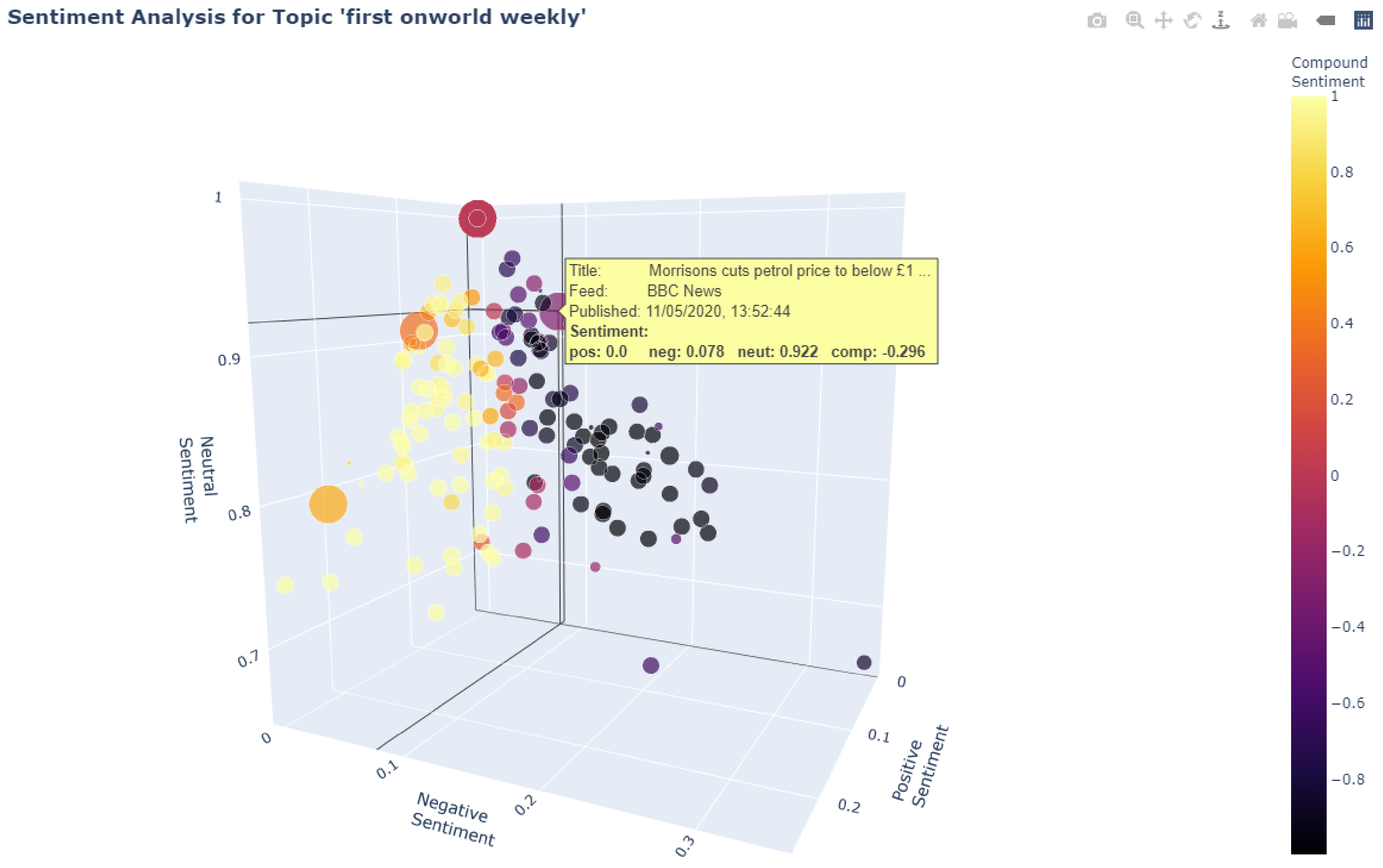


LDA Visualization (pyLDAvis interactive viewer)


BokehJS 2.0.2 successfully loaded.

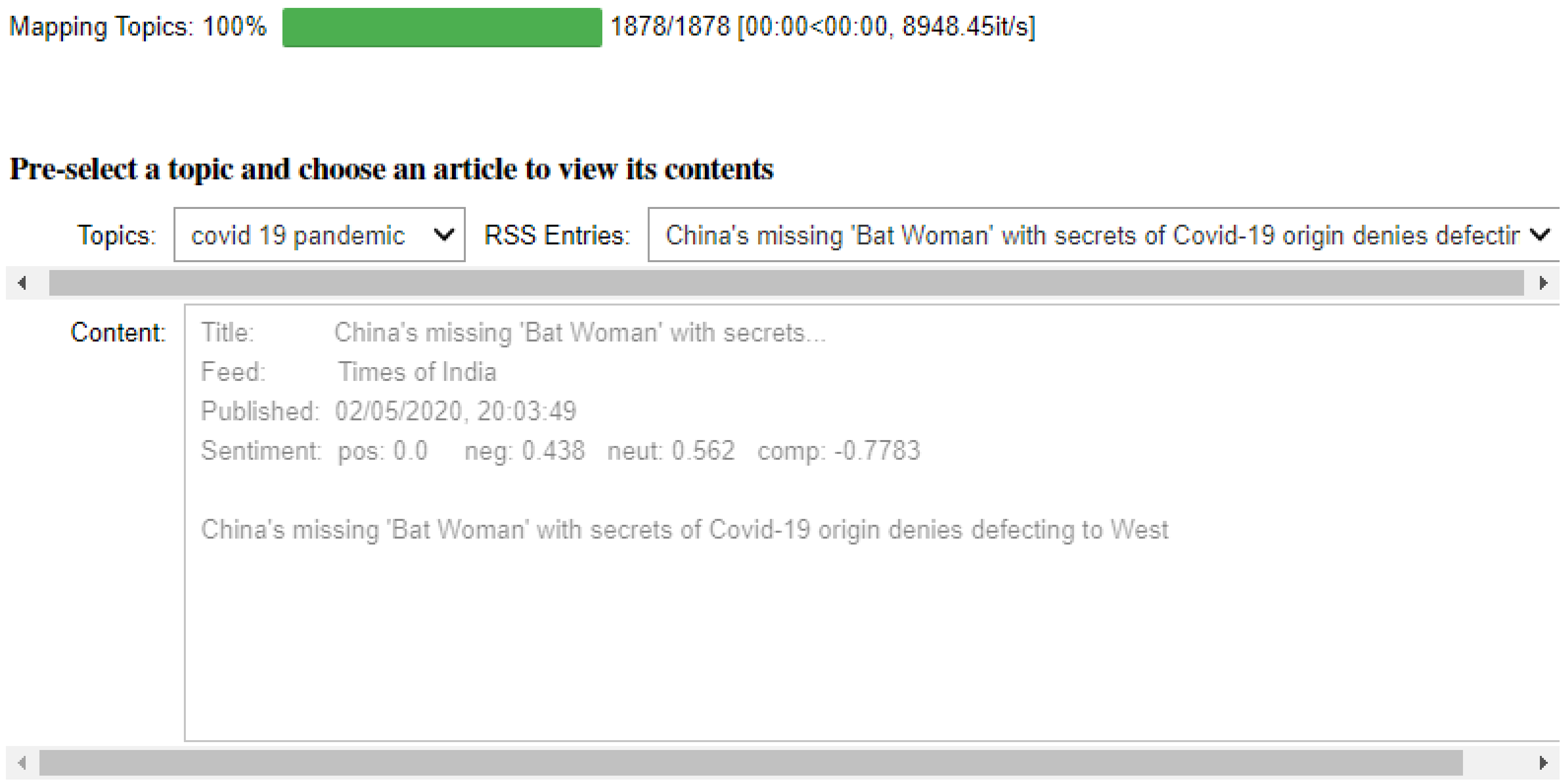
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Sentiment Analysis (3d interactive plot)



Content Viewer

```
%aimport sentiment3d
importlib.reload(sentiment3d)
from sentiment3d import conductSentimentAnalysis, contentsViewer
contentsViewer(allDict, topics)
```



```
Mapping Topics: 100% 1878/1878 [00:00<00:00, 8948.45it/s]
```

Pre-select a topic and choose an article to view its contents

Topics: covid 19 pandemic   RSS Entries: China's missing 'Bat Woman' with secrets of Covid-19 origin denies defectir

Content:

Title: China's missing 'Bat Woman' with secrets...
Feed: Times of India
Published: 02/05/2020, 20:03:49
Sentiment: pos: 0.0 neg: 0.438 neut: 0.562 comp: -0.7783

China's missing 'Bat Woman' with secrets of Covid-19 origin denies defecting to West

Versioning

All code is versioned in Github in this repository.

Authors

- **Janice Butler** - *Initial work* - Big-Data Project

License

Acknowledgments

- Many thanks to Damian Trilling for the ideas on using cosine similarity and Levenshtein Distance