FingerPrint API

### **class ArticleAnalyser(article)**

Instance through which all methods are accessed. To instantiate, call the class and pass in the text of the article you wish to analyse

The article must be surrounded by three single quotation marks. Eg…

* article1=ArticleAnalyser(‘’’This is the article I want to analyse’’’)

Each instance represents one article to be analysed

# Class methods

### **Article.opinion\_news\_score()**

### Returns a rating between 0 and 100 of how closely the article resembles an opinion piece, with 0 being the minimum and 100 the maximum

**Article.generate\_headline()**

Returns a headline for the article passed into the class. Also generates a visualisation of the model’s attention heads, showing which words in the article attend to which words in the headline at each head, and at each layer of the model.

The model is assumed to become more abstractive the higher the level

**Article.generate\_tweet()**

Returns a tweet for the article passed into the class. Also generates a visualisation of the model’s attention heads, showing which words in the tweet attend to which words in the headline at each head, and at each layer of the model.

**Article.predict\_tweet\_virality()**

Returns the predicted ‘virality’ of the generated tweet (*Article.generate\_tweet()* must be run first)

Virality score is one of five categories. In ascending order of virality; ‘no play’, ‘below par’, ‘average’, ‘above par’ and ‘viral’

**Article.viz\_opinion()**

Returns a visualisation of SHAP values explaining which words the model weights most heavily in predicting the ‘opinion score’ of the article. (*Article.opinion\_news\_score()* must be run first) It can help show us words we are using that may be inappropriate for a news article

**Article.viz\_virality()**

Returns a visualisation of LIME values explaining which words the model weights most heavily in predicting the ’virality’ of a tweet. It can help us in deciding how to words tweets in order to get maximum retweet

**Article.show\_vocab\_vectors(outlet)**

parameters:

* outlet: outlet whose vocabulary vectors you would like to be displayed. Must be one of; ‘guardian’, ‘bbc’, ‘cnn’, ‘fox’, ‘reuters’, ‘all’

outlet name must be surrounded by quotation marks

Returns a 3D representation showing the semantic and similarity of words in each outlet’s tweet vocabulary, as learned by the model which predicts the ‘virality’ of tweets.

You can search by word, and compare with other outlets to see how it varies across outlets.

The word vectors are generated by their lexical similarities, but also by the number of retweets generated by tweets they appear in, so words that generate high numbers of retweets will be drawn together.

**Article.make\_word\_time\_fig(word, outlet, tweet=**False**)**

parameters:

* word - the word whose usage you want to track. The word must be lower case and not include any punctuation marks. For instance, ‘Four-year’ should be passed in as ‘fouryear’. The word must be surrounded by quotation marks.
* outlet: outlet for which you want to track a word’s use over time. Must be one of; ‘guardian’, ‘bbc’, ‘cnn’, ‘fox’, ‘reuters’, ‘all’

Outlet name must be surrounded by quotation marks.

* tweet=False: Returns results for news articles

tweet=True: Returns results for tweets

Returns a graph showing how frequently a word has been used by an outlet from 2015 onwards. The score on the y-axis represents how many times the word appears per article over that year.

**Article.make\_virality\_wordcloud(outlet, virality)**

parameters:

* outlet: outlet whose tweets you want to turn into a wordcloud. Must be one of; ‘guardian’, ‘bbc’, ‘cnn’, ‘fox’, ‘reuters’, ‘all’

Outlet name must be surrounded by quotation marks.

* virality: The virality score of the tweets that you want to turn into a wordcloud. Must be one of ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘any’ with ‘0’ being the fewest retweets and ‘4’ the highest.

Returns a wordcloud showing the most common words appearing in tweets. The search can be filtered by outlet and by ‘virality’ score.

**Article.make\_opinion\_news\_wordcloud(news=True)**

parameters:

* news=True: displays wordcloud of news stories across all outlets.
* news=False: displays wordcloud of opinion pieces across all outlets

Returns a wordcloud of all news piece in the corpus, or all opinion pieces

**Article.headline\_package()**

Runs *generate\_headline*, *opinion\_news\_score* and *viz\_opinion* with one click

**Article.tweet\_package()**

Runs *generate\_tweet*, *predict\_tweet\_virality* and *viz\_virality* with one click