

## Portfolio 2 – Textmining

### Assignment 1: The collection

For use in this assignment I have used the Guardian API to collect data on collections in the news category. For this instance, I have chosen the topic name 'Scotland', and expect it to find results based on the country and its involvement in Brexit. Firstly I enter the url <https://open-platform.theguardian.com/explore/> to open up the API. To use this service I need a developer key, which I have gotten.



The image shows a search interface with a text input field containing the word 'scotland' and a blue button labeled 'Search content'.

### Assignment 2: Pre-process and describe your collection

A document-term matrix describes the frequency of terms in a collection of documents. In this case I will do it with the articles around the topic I have chosen. Regarding pre-processing I will use stopwords. I download the nltk library as nk to use for the stopwords. Also sklearn where I use the count vectorizer to count the terms from the articles. The features can help me detain the many different terms and only find the most used words. In this case I have chosen the top 500 most used terms.

To transform the data into a document-term matrix use CountVectorizer from sklearn and stopwords from nltk:

```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import stopwords as sw
model_vect = CountVectorizer(stop_words=sw.words('english'), token_pattern=r'[a-zA-Z\-\_]{2,}')
data_vect = model_vect.fit_transform(texts)
data_vect
```

I apply the model, because I want to turn this model into my document term matrix. Stop words are regular used words like 'a' and 'in'. These, and other similar have been removed.

Finding info about data like uniqueness of a section, or the total number of characters. Number of articles in the most unique sections of the Guardian API.

```
import numpy as np
unique, counts = np.unique(texts, return_counts=True)
dict(zip(unique, counts))
```

The total number of characters in the data of the sections

```
total = 0
for text in texts:
    total = total + len(text)
total
```

total = 187195

Finding the texts length

```
all_lengths = list()
for text in texts:
    all_lengths.append(len(text))
print("Total sum: %i" % sum(all_lengths))
```

Total sum is 118100885

Next up is finding out how many words are in the data. I will use the word count function

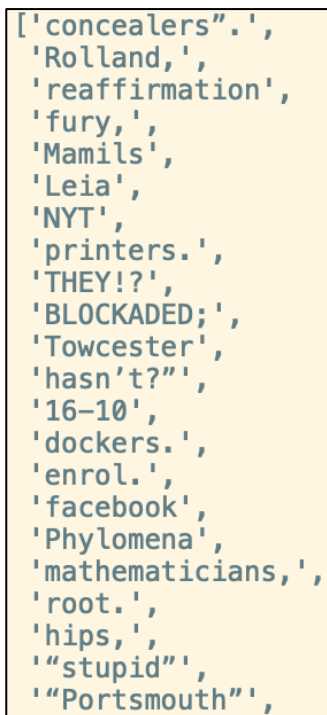
```
word_count = 0
for text in texts:
    words = text.split()
    word_count = word_count + len(words)
word_count
```

The total word count is 19999514

Then I am making all\_words into a list and finding the unique word count ("Notebook on nbviewer", n.d.)

```
all_words = list()
for text in texts:
    words = text.split()
    all_words.extend(words)
unique_words = set(all_words)
unique_word_count = len(unique_words)
print("Unique word count: %i" % unique_word_count)
```

I'm also using the random word sample and random.sample(unique\_words, 30). It will show me 30 random unique words, which I have listed some of in the next image:



```
['concealers.',
 'Rolland,',
 'reaffirmation',
 'fury,',
 'Mamils',
 'Leia',
 'NYT',
 'printers.',
 'THEY!?',
 'BLOCKADED;',
 'Towcester',
 'hasn't?',
 '16-10',
 'dockers.',
 'enrol.',
 'facebook',
 'Phylomena',
 'mathematicians,',
 'root.',
 'hips,',
 '"stupid"',
 '"Portsmouth"']
```

```
print("Previous word count: %i" % word_count)
print("Word count: %i" % len(tokens))
unique_tokens = set(tokens)
print("Previous unique word count: %i" % unique_word_count)
print("Unique word count: %i" % len(unique_tokens))
```

Previous word count: 19999514

Word count: 19713366

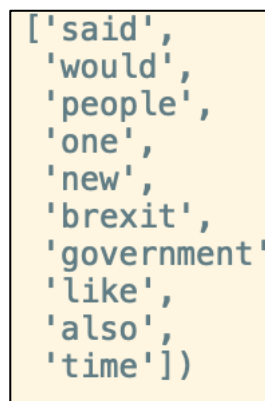
Previous unique word count: 557422

Unique word count: 147488

### Assignment 3: Select articles using a query

As the third assignment in this portfolio is to find select articles using a query. As earlier stated I have chosen 'scotland' as a topic and will further search to find a subset of articles.

The top words of the query 'Scotland':



```
['said',  
'would',  
'people',  
'one',  
'new',  
'brexit',  
'government',  
'like',  
'also',  
'time']
```

Top words in sections: ['said', 'one', 'would', 'people', 'new', 'also', 'time', 'like', 'first', 'says', 'last', 'could', 'two', 'years', 'government', 'year', 'back', 'may', 'get', 'world', 'way', 'even', 'many', 'brexit', 'trump', 'make', 'made', 'much', 'still', 'work', 'party', 'well', 'going', 'good', 'think', 'three', 'take', 'day', 'see', 'since', 'deal', 'week', 'women', 'another', 'told', 'right', 'around', 'want', 'say', 'need']

### Assignment 4: Model and visualize the topics in your subset

For the last bit of this portfolio 2 I will visualize my findings with a topic modeling and a word cloud. With my articles and results I have found the most commonly used terms as stated above. That can be described even more clearer, with the use of a word cloud. Word clouds make it easier to visualize a specific topic or a group of words. This will show my findings and present it in a more manageable way. First I will do some topic modeling:

```

from sklearn.decomposition import LatentDirichletAllocation
model_lda = LatentDirichletAllocation(n_components=6, random_state=0)
data_lda = model_lda.fit_transform(data_vect)
np.shape(data_lda)

for i, term_weights in enumerate(model_lda.components_):
    top_idx = (-term_weights).argsort()[0:10]
top_words = ["%s (%.3f)" % (model_vect.get_feature_names()[idx], term_weights[idx]) for idx in
              top_idx]
print("Topic %d: %s" % (i, ".join(top_words)))

```

Shows the top words in each topic.

```

Topic 0: game (8544.550), team (6949.551), one (6634.026), first (6457.470),
league (6325.767), min (6314.563), players (6111.863), last (5729.163), season
(5652.682), back (5537.819)
Topic 1: one (5720.235), two (4196.732), first (3694.353), back (3630.502),
england (3460.289), australia (3320.233), new (3210.084), four (2879.385), day
(2671.905), time (2594.106)
Topic 2: one (13246.753), like (10395.425), people (9829.506), time (8229.945),
first (7399.164), says (7100.684), life (6741.593), would (6108.534), years
(5701.787), even (5452.130)
Topic 3: said (52888.819), would (27466.267), people (22247.037), government
(21110.459), trump (17271.311), brexit (15108.609), party (14814.749), one
(13971.327), may (13641.185), also (12760.542)
Topic 4: said (10848.184), year (8667.648), also (6306.195), would (5909.141),
new (5710.385), company (5128.936), years (4729.887), last (4506.823), could
(4299.811), people (4293.144)
Topic 5: women (8322.206), one (7205.734), like (7074.405), says (5108.601),
people (4972.915), new (4783.714), also (4312.723), said (3910.067), show
(3531.472), time (3339.093)

```

Using word cloud to visualize most commonly used words of the query 'scotland'.

```

from wordcloud import WordCloud
import matplotlib.pyplot as plt

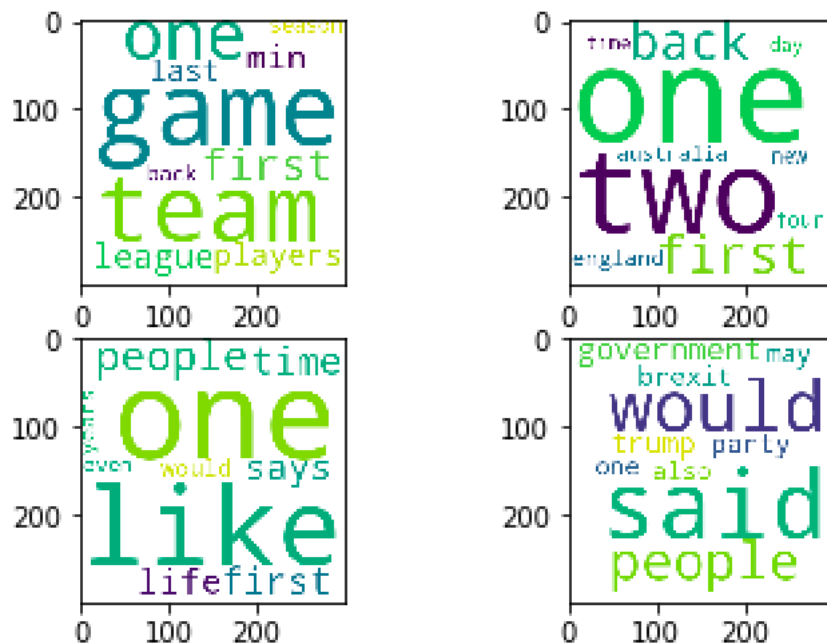
for i, term_weights in enumerate(model_lda.components_):
    top_idx = (-term_weights).argsort()[0:10]

```

```

top_words = [model_vect.get_feature_names()[idx] for idx in top_idxxs]
word_freqs = dict(zip(top_words, term_weights[top_idxxs]))
wc = WordCloud(background_color="white",width=300,height=300,
               max_words=10).generate_from_frequencies(word_freqs)
plt.subplot(2, 2, i+1)
plt.imshow(wc)

```



### Conclusion:

The word cloud has shown that the top used words for the query 'Scotland' is what I imagined it would be. Words like game and team shows how sports like football is popular in the European country, just like government and Brexit which is still undergoing over there as of now.

### Reference list:

Notebook on nbviewer. (n.d.).

Retrieved from <https://nbviewer.jupyter.org/github/fbkarsdorp/python-course/blob/master/Chapter%20-%20-%20First%20steps.ipynb>

The Guardian Open Platform. (n.d.).

Retrieved from <https://open-platform.theguardian.com/explore/>.

Working With Text Data¶. (n.d.).

Retrieved from [https://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html).