CWordTM Usage on BBC News

This Jupyter notebook demonstrates how to use the package "CWordTM" on the BBC News:

1. Meta Information Features
2. Utility Features
3. Text Visualization - Word Cloud
4. Text Summarization
5. Topic Modeling - LDA and BERTopic

1. Meta Information Features

In [1]:
```
import cwordtm
from cwordtm import *
```

In [2]:
```
# Show execution time
df = util.load_text("BBC/BBC News Train.csv", timing=True)
```

```
Loading file 'BBC/BBC News Train.csv' ...
Finished 'load_text' in 0.1035 secs
```

In [3]:
```
# Execute and show code
df = util.load_text("BBC/BBC News Train.csv", code=1)
```

```
Loading file 'BBC/BBC News Train.csv' ...

def load_text(filepath, nr=0, info=False):
    """Loads and returns the text from the prescribed file path ('filepath').

    :param filepath: The prescribed filepath from which the text is loaded,
        default to None
    :type filepath: str
    :param nr: The number of rows of text to be loaded; 0 represents all rows,
        default to 0
    :type nr: int, optional
    :param info: The flag whether the dataset information is shown,
        default to False
    :type info: bool, optional
    :return: The collection of text with the prescribed number of rows loaded
    :rtype: pandas.DataFrame
    """

    print("Loading file '%s' ..." %filepath)
    df = pd.read_csv(filepath)
    if nr > 0:
       print("Initial Records:")
       print(df.head(int(nr)))
    if info:
        print("\nDataset Information:")
        df.info()
    return df
```

In [4]:
```
# Show code without execution
df = util.load_text("BBC/BBC News Train.csv", code=2)
```

```
def load_text(filepath, nr=0, info=False):
    """Loads and returns the text from the prescribed file path ('filepath').

    :param filepath: The prescribed filepath from which the text is loaded,
        default to None
    :type filepath: str
    :param nr: The number of rows of text to be loaded; 0 represents all rows,
        default to 0
    :type nr: int, optional
    :param info: The flag whether the dataset information is shown,
        default to False
    :type info: bool, optional
    :return: The collection of text with the prescribed number of rows loaded
    :rtype: pandas.DataFrame
    """

    print("Loading file '%s' ..." %filepath)
    df = pd.read_csv(filepath)
    if nr > 0:
       print("Initial Records:")
       print(df.head(int(nr)))
    if info:
        print("\nDataset Information:")
        df.info()
    return df
```

In [5]:
```
# Add timing and code reveal features to some other function
from importlib_resources import files
files = meta.addin(files)
files(code=2)
```

```
@package_to_anchor
def files(anchor: Optional[Anchor] = None) -> Traversable:
    """
    Get a Traversable resource for an anchor.
    """
    return from_package(resolve(anchor))
```

2. Utility Features

Load BBC News

In [6]:
```
bbc_file = "BBC/BBC News Train.csv"
df = util.load_text(bbc_file, info=True)
```

```
Loading file 'BBC/BBC News Train.csv' ...

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1490 entries, 0 to 1489
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ArticleId  1490 non-null   int64
 1   Text       1490 non-null   object
 2   Category   1490 non-null   object
dtypes: int64(1), object(2)
memory usage: 35.0+ KB
```

Preprocessing Text

In [7]:
```
text_list = util.get_text_list(df.iloc[:500], text_col='Text')
text = util.preprocess_text(text_list)
```

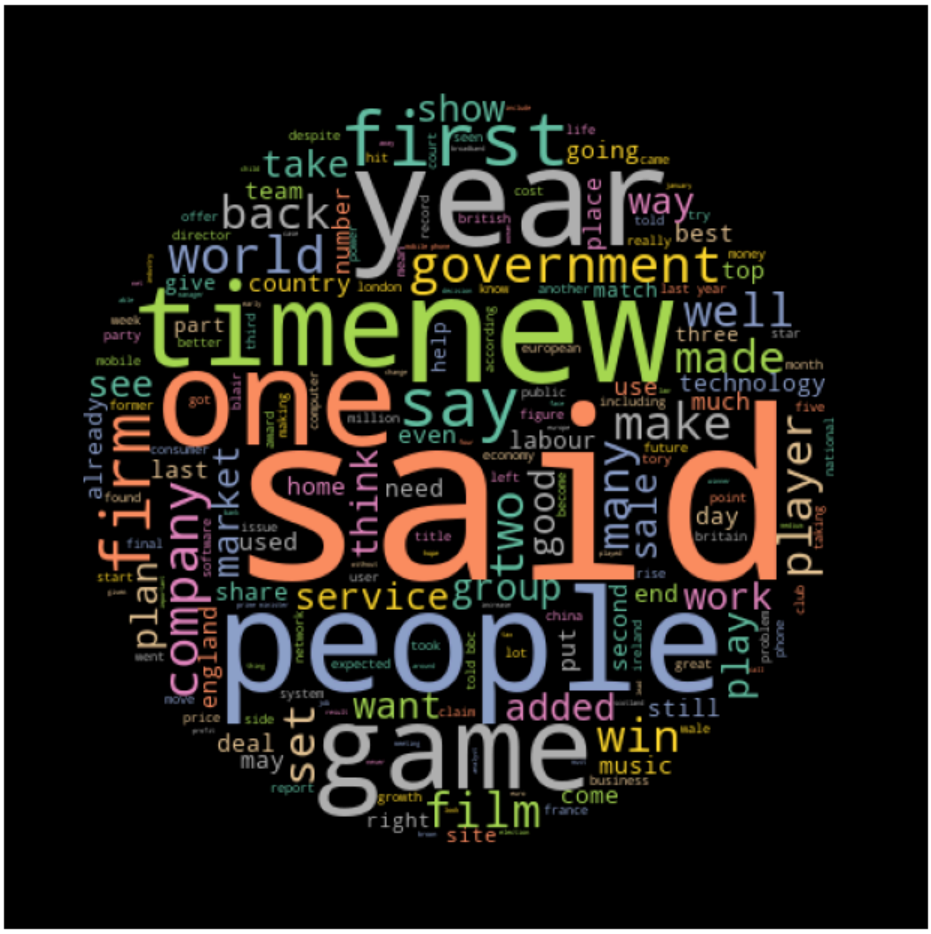
3. Text Visualization - Word Cloud

In [8]:
```
# White background with no image mask
viz.show_wordcloud(text)
```

```
C:\Dev\Anaconda3\envs\aiml\lib\site-packages\wordcloud\wordcloud.py:106: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
  self.colormap = plt.cm.get_cmap(colormap)
```

In [9]:
```
# Black background with the prescribed image as the mask
viz.show_wordcloud(text, bg='black', image='images/disc.png')
```

```
C:\Dev\Anaconda3\envs\aiml\lib\site-packages\wordcloud\wordcloud.py:106: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releases later. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instead.
  self.colormap = plt.cm.get_cmap(colormap)
```



4. Text Summarization

In [10]:
```
news = df.iloc[:50]['Text']  # "df" stores previously loaded text
ta.summary(news, weight=3)
```

Out[10]:
```
['in  the figure was %',
 'they also have to want it',
 'in a way  they were doing exactly what they do now']
```

5. Topic Modeling

LDA Model

In [11]:
```
doc_file = "BBC/BBC News Train.csv"
lda = tm.lda_process(doc_file, source=1, text_col='Text', eval=True)
```

```
Corpus loaded!
Text preprocessed!
Text trained!
If no visualization is shown,
  you may execute the following commands to show the visualization:
    > import pyLDAvis
    > pyLDAvis.display(lda.vis_data)
Visualization prepared!

Topics from LDA Model:
[(0,
  '0.029*"" + 0.025*"s" + 0.021*"m" + 0.013*"£" + 0.010*"bn" + 0.010*"said" + '
  '0.008*"share" + 0.007*"profit" + 0.007*"club" + 0.007*"firm"'),
 (1,
  '0.031*"" + 0.015*"said" + 0.014*"game" + 0.012*"s" + 0.010*"use" + '
  '0.009*"peopl" + 0.006*"technolog" + 0.006*"new" + 0.006*"comput" + '
  '0.005*"year"'),
 (2,
  '0.034*"" + 0.020*"s" + 0.014*"m" + 0.010*"said" + 0.010*"world" + '
  '0.008*"champion" + 0.008*"olymp" + 0.007*"year" + 0.006*"win" + '
  '0.006*"women"'),
 (3,
  '0.018*"mr" + 0.016*"s" + 0.016*"said" + 0.015*"parti" + 0.012*"elect" + '
  '0.009*"blair" + 0.008*"" + 0.006*"labour" + 0.006*"say" + 0.006*"ukip"'),
 (4,
  '0.014*"" + 0.011*"mobil" + 0.011*"v" + 0.011*"s" + 0.010*"music" + '
  '0.007*"said" + 0.005*"cont" + 0.005*"servic" + 0.004*"use" + 0.004*"jone"'),
 (5,
  '0.029*"" + 0.021*"said" + 0.018*"s" + 0.015*"mr" + 0.010*"govern" + '
  '0.007*"labour" + 0.006*"say" + 0.006*"tax" + 0.006*"year" + 0.006*"£"'),
 (6,
  '0.029*"s" + 0.025*"" + 0.015*"film" + 0.008*"said" + 0.007*"music" + '
  '0.006*"star" + 0.006*"play" + 0.005*"world" + 0.005*"new" + 0.005*"year"'),
 (7,
  '0.043*"" + 0.020*"s" + 0.018*"said" + 0.012*"year" + 0.007*"rate" + '
  '0.006*"bn" + 0.006*"growth" + 0.005*"sale" + 0.005*"market" + '
  '0.005*"month"'),
 (8,
  '0.027*"" + 0.020*"s" + 0.019*"said" + 0.008*"firm" + 0.007*"compani" + '
  '0.007*"year" + 0.007*"china" + 0.006*"new" + 0.005*"trade" + '
  '0.005*"dollar"'),
 (9,
  '0.020*"" + 0.012*"phone" + 0.011*"use" + 0.010*"s" + 0.010*"said" + '
  '0.009*"peopl" + 0.008*"mobil" + 0.007*"net" + 0.007*"gadget" + '
  '0.006*"make"'),
 (10,
  '0.026*"s" + 0.015*"" + 0.010*"said" + 0.010*"game" + 0.009*"play" + '
  '0.008*"win" + 0.008*"england" + 0.007*"player" + 0.006*"t" + 0.005*"time"'),
 (11,
  '0.023*"s" + 0.021*"film" + 0.021*"" + 0.020*"best" + 0.018*"award" + '
  '0.010*"star" + 0.009*"actor" + 0.008*"said" + 0.008*"nomin" + 0.008*"year"'),
 (12,
  '0.023*"said" + 0.013*"s" + 0.009*"peopl" + 0.008*"plan" + 0.007*"" + '
  '0.006*"site" + 0.005*"govern" + 0.005*"mr" + 0.004*"say" + 0.004*"id"'),
 (13,
  '0.017*"said" + 0.012*"yuko" + 0.011*"s" + 0.011*"" + 0.010*"e-mail" + '
  '0.008*"oil" + 0.008*"hunt" + 0.007*"russian" + 0.007*"virus" + '
  '0.007*"court"'),
 (14,
  '0.021*"" + 0.017*"said" + 0.012*"s" + 0.009*"peopl" + 0.009*"mr" + '
  '0.007*"servic" + 0.007*"use" + 0.006*"search" + 0.006*"broadband" + '
  '0.005*"tv"')]

Model Evaluation Scores:
  Coherence: 0.45912475892974697
  Perplexity: -7.8175556047677395
  Topic diversity: 0.0008193081154030254
  Topic size distribution: 0.002240896358543417
```

In [12]:
```
# LDA Model Visualization
import pyLDAvis
pyLDAvis.display(lda.vis_data)
```

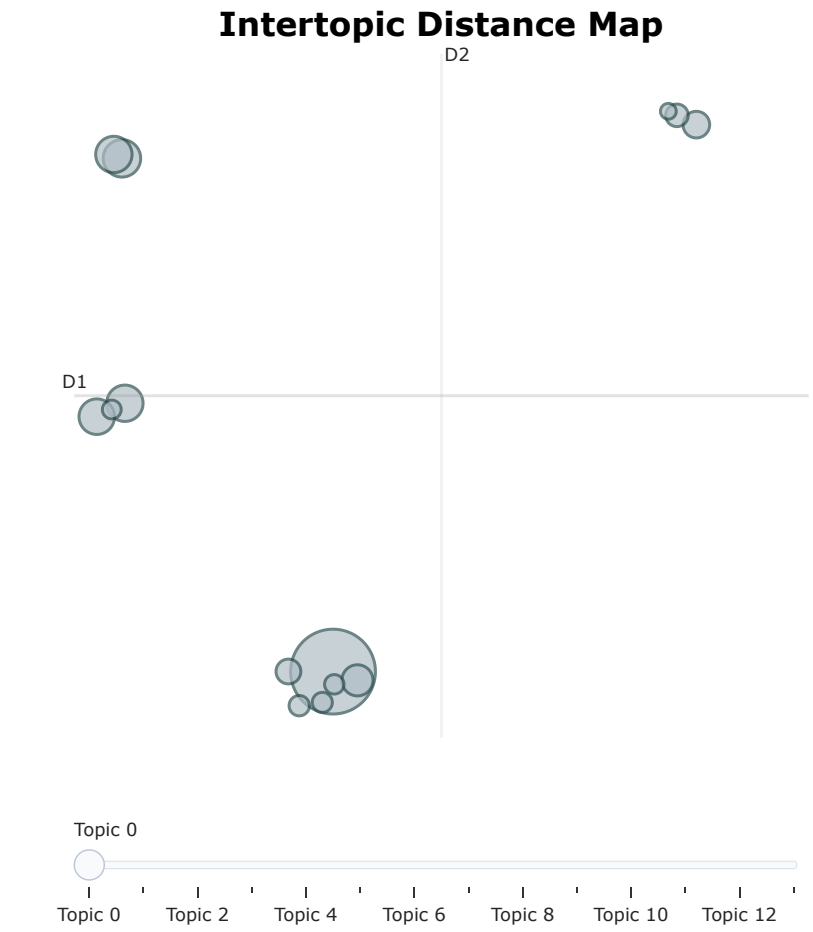
Out[12]:

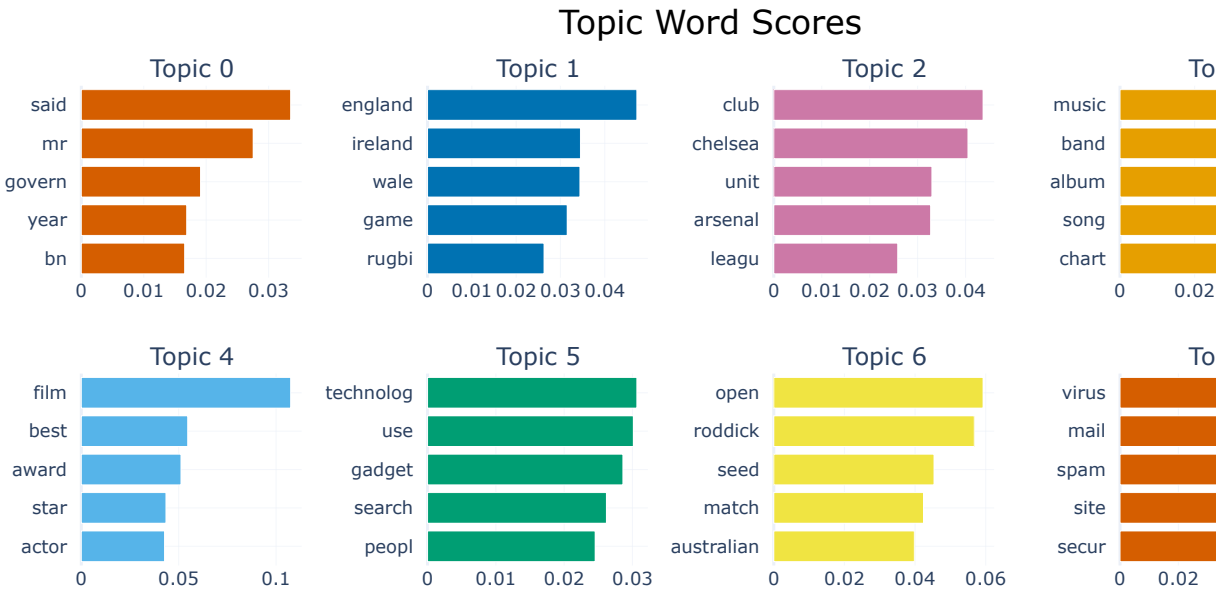
BERTopic Model

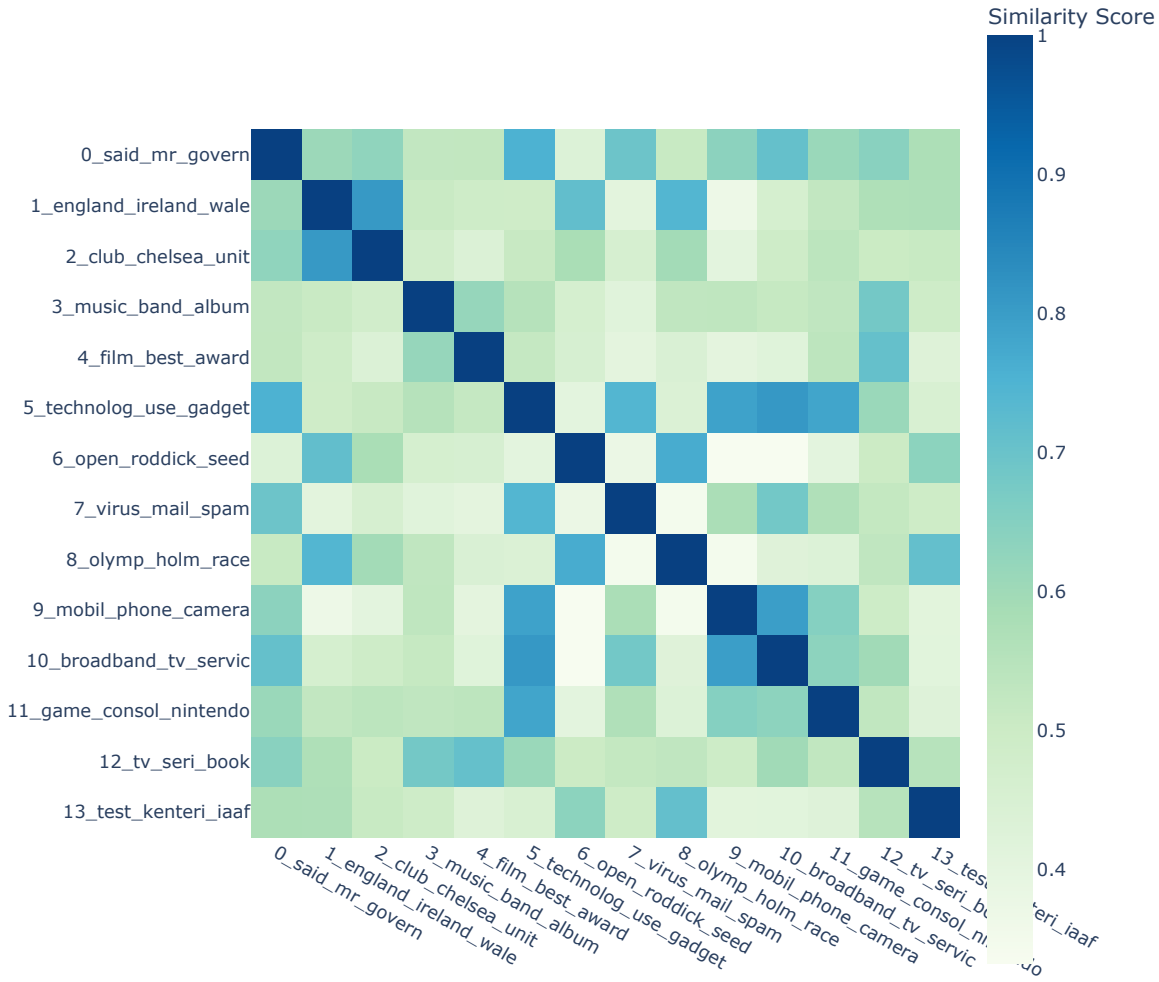
In [13]:

```
btm = tm.btm_process(doc_file, source=1, text_col='Text', eval=True)
```

```
Corpus loaded!
Text preprocessed!
Text trained!

Topics from BERTopic Model:
Topic 0: said | mr | govern | year | bn | elect | say | labour | parti | minist
Topic 1: england | ireland | wale | game | rugbi | win | play | half | franc | player
Topic 2: club | chelsea | unit | arsenal | leagu | goal | game | play | liverpool | player
Topic 3: music | band | album | song | chart | record | singl | singer | year | perform
Topic 4: film | best | award | star | actor | oscar | nomin | director | actress | year
Topic 5: technolog | use | gadget | search | peopl | said | comput | blog | digit | devic
Topic 6: open | roddick | seed | match | australian | play | nadal | set | win | final
Topic 7: virus | mail | spam | site | secur | user | program | attack | use | softwar
Topic 8: olymp | holm | race | world | indoor | champion | radcliff | championship | marathon | athlet
Topic 9: mobil | phone | camera | use | handset | peopl | servic | music | technolog | said
Topic 10: broadband | tv | servic | net | bt | peopl | uk | user | use | connect
Topic 11: game | consol | nintendo | gamer | xbox | high | soni | dvd | titl | definit
Topic 12: tv | seri | book | brother | said | novel | channel | prize | televis | evict
Topic 13: test | kenteri | iaaf | cont | greek | olymp | drug | thanou | athlet | ban

Model Evaluation Scores:
  Coherence: 0.5884881731425379

BERTopic Model Visualization:
```


Intertopic Distance Map
D2
D1
Topic 0
Topic 0
Topic 2
Topic 4
Topic 6
Topic 8
Topic 10
Topic 12


Topic Word Scores
Topic 0
said
mr
govern
year
bn
0
0.01
0.02
0.03
Topic 1
england
ireland
wale
game
rugbi
0
0.01
0.02
0.03
0.04
Topic 2
club
chelsea
unit
arsenal
leagu
0
0.01
0.02
0.03
0.04
To
music
band
album
song
chart
0
0.02
Topic 4
film
best
award
star
actor
0
0.05
0.1
Topic 5
technolog
use
gadget
search
peopl
0
0.01
0.02
0.03
Topic 6
open
roddick
seed
match
australian
0
0.02
0.04
0.06
To
virus
mail
spam
site
secur
0
0.02




```
If no visualization is shown,
  you may execute the following commands one-by-one:
    btm.model.visualize_topics()
    btm.model.visualize_barchart()
    btm.model.visualize_heatmap()
```