CWordTM Toolkit 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 *
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\johnnyc\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\johnnyc\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\johnnyc\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\johnnyc\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

In [2]:
```
cwordtm.__version__
```

Out[2]:
```
'0.6.2'
```

In [3]:
```
# Show brief module information
print(meta.get_module_info())
```

```
The member information of the module 'cwordtm'
1. Submodule meta:
   addin (func)
   addin_all (modname='cwordtm')
   addin_all_functions (submod)
   get_function (mod_name, submodules, func_name)
   get_module_info (detailed=False)
   get_submodule_info (submodname, detailed=False)
   import_module (name, package=None)
   wraps (wrapped, assigned=('__module__', '__name__', '__qualname__', '__doc__', '__annotations__'), updated=('__di
ct__',))
2. Submodule pivot:
   stat (df, chi=False, *, timing=False, code=0)
3. Submodule quot:
   extract_quotation (text, quot_marks, *, timing=False, code=0)
   match_text (target, sent_tokens, lang, threshold, n=5, *, timing=False, code=0)
   match_verse (i, ot_list, otdf, df, book, chap, verse, lang, threshold, *, timing=False, code=0)
   show_quot (target, source='ot', lang='en', threshold=0.5, *, timing=False, code=0)
   tokenize (sentence, *, timing=False, code=0)
4. Submodule ta:
   get_sent_scores (sentences, diction, sent_len, *, timing=False, code=0) -> dict
   get_sentences (docs, lang='en', *, timing=False, code=0)
   get_summary (sentences, sent_weight, threshold, sent_len, *, timing=False, code=0)
   pos_tag (tokens, tagset=None, lang='eng', *, timing=False, code=0)
   preprocess_sent (text, *, timing=False, code=0)
   sent_tokenize (text, language='english', *, timing=False, code=0)
   summary_chi (docs, weight=1.5, sent_len=8, *, timing=False, code=0)
   summary_en (docs, sent_len=8, *, timing=False, code=0)
   word_tokenize (text, language='english', preserve_line=False, *, timing=False, code=0)
5. Submodule tm:
   BTM (textfile, chi=False, num_topics=15, embed=True)
   LDA (textfile, chi=False, num_topics=15)
   NMF (textfile, chi=False, num_topics=15)
   btm_process (doc_file, source=0, text_col='text', cat=0, chi=False, group=True, eval=False, *, timing=False, code
=0)
   lda_process (doc_file, source=0, text_col='text', cat=0, chi=False, group=True, eval=False, *, timing=False, code
=0)
   load_bible (textfile, cat=0, group=True, *, timing=False, code=0)
   load_text (textfile, text_col='text', *, timing=False, code=0)
   nmf_process (doc_file, source=0, text_col='text', cat=0, chi=False, group=True, eval=False, *, timing=False, code
=0)
   pprint (object, stream=None, indent=1, width=80, depth=None, *, compact=False, sort_dicts=True, underscore_number
s=False, timing=False, code=0)
   process_text (doc, *, timing=False, code=0)
6. Submodule util:
   add_chi_vocab (*, timing=False, code=0)
   chi_sent_terms (text, *, timing=False, code=0)
   chi_stops (*, timing=False, code=0)
   clean_sentences (sentences, *, timing=False, code=0)
   clean_text (df, text_col='text', *, timing=False, code=0)
   extract (df, testament=-1, category='', book=0, chapter=0, verse=0, *, timing=False, code=0)
   extract2 (df, filter='', *, timing=False, code=0)
   get_diction (docs, *, timing=False, code=0)
   get_diction_chi (docs, *, timing=False, code=0)
   get_diction_en (docs, *, timing=False, code=0)
   get_list (df, column='book', *, timing=False, code=0)
   get_sent_terms (text, *, timing=False, code=0)
   get_text (df, text_col='text', *, timing=False, code=0)
   get_text_list (df, text_col='text', *, timing=False, code=0)
   group_text (df, column='chapter', *, timing=False, code=0)
   is_chi (*, timing=False, code=0)
   load_text (filepath, nr=0, info=False, *, timing=False, code=0)
   load_word (ver='web.csv', nr=0, info=False, *, timing=False, code=0)
   preprocess_text (text, *, timing=False, code=0)
   remove_noise (text, noise_list, *, timing=False, code=0)
   set_lang (lang='en', *, timing=False, code=0)
   word_tokenize (text, language='english', preserve_line=False, *, timing=False, code=0)
7. Submodule version:
8. Submodule viz:
   chi_wordcloud (docs, figsize=(15, 10), bg='white', image=0, *, timing=False, code=0)
   plot_cloud (wordcloud, figsize, *, timing=False, code=0)
   show_wordcloud (docs, clean=False, figsize=(12, 8), bg='white', image=0, *, timing=False, code=0)
```

In [4]:
```
# Show detailed module information of a submodule
print(meta.get_submodule_info("viz", detailed=True))
```

```
The function(s) of the submodule 'cwordtm.viz':

def chi_wordcloud(docs, figsize=(15, 10), bg='white', image=0):
    """Prepare and show a Chinese wordcloud

    :param docs: The collection of Chinese documents for preparing a wordcloud,
        default to None
    :type docs: pandas.DataFrame
    :param figsize: Size (width, height) of word cloud, default to (15, 10)
    :type figsize: tuple, optional
    :param bg: The background color (name) of the wordcloud, default to 'white'
    :type bg: str, optional
    :param image: The filename of the presribed image as the mask of the wordcloud,
        or 1/2/3/4 for using an internal image (heart / disc / triangle / arrow),
        default to 0 (No image mask)
    :type image: int or str, optional
    """

    util.set_lang('chi')
    diction = util.get_diction(docs)

    masks = ['heart.jpg', 'disc.jpg', 'triangle.jpg', 'arrow.jpg']

    if image == 0:
        mask = None
    elif image in [1, 2, 3, 4]:  # Internal image file
        img_file = files('cwordtm.images').joinpath(masks[image-1])
        mask = np.array(Image.open(img_file))
    elif isinstance(image, str) and len(image) > 0:
        mask = np.array(Image.open(image))
    else:
        mask = None

    font_file = files('cwordtm.data').joinpath('msyh.ttc')
    wordcloud = WordCloud(background_color=bg, colormap='Set2',
                          mask=mask, font_path=str(font_file)) \
                    .generate_from_frequencies(frequencies=diction)

    plot_cloud(wordcloud, figsize=figsize)

def plot_cloud(wordcloud, figsize):
    """Plot the prepared 'wordcloud'
    :param wordcloud: The WordCloud object for plotting, default to None
    :type wordcloud: WordCloud object
    :param figsize: Size (width, height) of word cloud, default to None
    :type figsize: tuple
    """

    plt.figure(figsize=figsize)
    plt.imshow(wordcloud)
    plt.axis("off");

def show_wordcloud(docs, clean=False, figsize=(12, 8), bg='white', image=0):
    """Prepare and show a wordcloud

    :param docs: The collection of documents for preparing a wordcloud,
        default to None
    :type docs: pandas.DataFrame
    :param clean: The flag whether text preprocessing is needed,
        default to False
    :type clean: bool, optional
    :param figsize: Size (width, height) of word cloud, default to (12, 8)
    :type figsize: tuple, optional
    :param bg: The background color (name) of the wordcloud, default to 'white'
    :type bg: str, optional
    :param image: The filename of the presribed image as the mask of the wordcloud,
        or 1/2/3/4 for using an internal image (heart / disc / triangle / arrow),
        default to 0 (No image mask)
    :type image: int or str, optional
    """

    masks = ['heart.jpg', 'disc.jpg', 'triangle.jpg', 'arrow.jpg']

    if image == 0:
        mask = None
    elif image in [1, 2, 3, 4]:  # Internal image file
        img_file = files('cwordtm.images').joinpath(masks[image-1])
        mask = np.array(Image.open(img_file))
    elif isinstance(image, str) and len(image) > 0:
        mask = np.array(Image.open(image))
    else:
        mask = None

    if isinstance(docs, pd.DataFrame):
        docs = ' '.join(list(docs.text.astype(str)))
    elif isinstance(docs, pd.Series):
```

```
    docs = ' '.join(list(docs.astype(str)))
elif isinstance(docs, list) or isinstance(docs, np.ndarray):
    docs = ' '.join(str(doc) for doc in docs)

if clean:
    docs = util.preprocess_text(docs)

wordcloud = WordCloud(background_color=bg, colormap='Set2', mask=mask) \
                .generate(docs)

plot_cloud(wordcloud, figsize=figsize)
```

In [5]:
```
# Show execution time
df = util.load_text("BBC/BBC News Train.csv", timing=True)
```

```
Finished 'load_text' in 0.0668 secs
```

In [6]:
```
# Execute and show code
df = util.load_text("BBC/BBC News Train.csv", code=1)
```

```
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)
    if filepath.lower().endswith('csv'):
        nrows = None
        if nr > 0: nrows = nr
        df = pd.read_csv(filepath, nrows=nrows, encoding='utf-8')
    else:
        noise_list = ['\u3000', '— ', '•']
        tf =  open(filepath, encoding='utf-8')
        lines = [remove_noise(line, noise_list) for line in tf.readlines() \
                    if len(remove_noise(line, noise_list)) > 0]

        df = pd.DataFrame({'text': lines})
        if nr > 0: df = df.iloc[:nr]

    if info:
        print("\nDataset Information:")
        df.info()

    return df

>> cwordtm.util.remove_noise
def remove_noise(text, noise_list):
    """Removes a list of substrings in noise_list from the input text.

    :param text: The input text, default to None
    :type text: str
    :param noise_list: The list of substrings to be removed, default to ""
    :type noise_list: list, optional
    :return: The text with the prescribed substrings removed
    :rtype: str
    """

    text = text.rstrip()
    for noise in noise_list:
        text = text.replace(noise, '')
    return text

>> cwordtm.util.remove_noise
def remove_noise(text, noise_list):
    """Removes a list of substrings in noise_list from the input text.

    :param text: The input text, default to None
    :type text: str
    :param noise_list: The list of substrings to be removed, default to ""
    :type noise_list: list, optional
    :return: The text with the prescribed substrings removed
    :rtype: str
    """

    text = text.rstrip()
    for noise in noise_list:
        text = text.replace(noise, '')
    return text
```

In [7]:
```
# 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)
    if filepath.lower().endswith('csv'):
        nrows = None
        if nr > 0: nrows = nr
        df = pd.read_csv(filepath, nrows=nrows, encoding='utf-8')
    else:
        noise_list = ['\u3000', '— ', '•']
        tf =  open(filepath, encoding='utf-8')
        lines = [remove_noise(line, noise_list) for line in tf.readlines() \
                    if len(remove_noise(line, noise_list)) > 0]

        df = pd.DataFrame({'text': lines})
        if nr > 0: df = df.iloc[:nr]

    if info:
        print("\nDataset Information:")
        df.info()

    return df


>> cwordtm.util.remove_noise
def remove_noise(text, noise_list):
    """Removes a list of substrings in noise_list from the input text.

    :param text: The input text, default to None
    :type text: str
    :param noise_list: The list of substrings to be removed, default to ""
    :type noise_list: list, optional
    :return: The text with the prescribed substrings removed
    :rtype: str
    """

    text = text.rstrip()
    for noise in noise_list:
        text = text.replace(noise, '')
    return text

>> cwordtm.util.remove_noise
def remove_noise(text, noise_list):
    """Removes a list of substrings in noise_list from the input text.

    :param text: The input text, default to None
    :type text: str
    :param noise_list: The list of substrings to be removed, default to ""
    :type noise_list: list, optional
    :return: The text with the prescribed substrings removed
    :rtype: str
    """

    text = text.rstrip()
    for noise in noise_list:
        text = text.replace(noise, '')
    return text
```

In [8]:

```
# 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 [9]:
```
bbc_file = "BBC/BBC News Train.csv"
df = util.load_text(bbc_file, info=True)
```

```
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 [10]:
```
text_list = util.get_text_list(df.iloc[:500], text_col='Text')
text = util.preprocess_text(text_list)
```

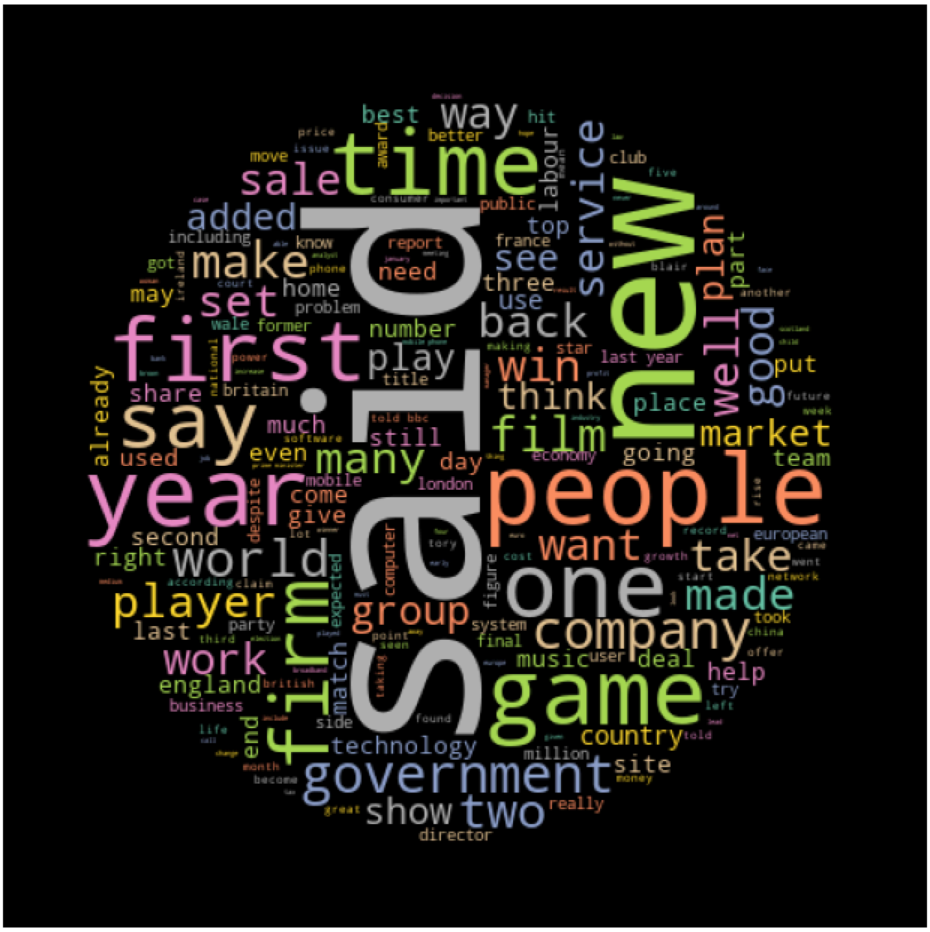
3. Text Visualization - Word Cloud

In [11]:
```
# White background with no image mask
viz.show_wordcloud(text)
```

```
D:\Dev\Anaconda3\lib\site-packages\wordcloud\wordcloud.py:106: MatplotlibDeprecationWarning: The get_cmap function w
as 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 [12]:
```
# Black background with the prescribed image as the mask
viz.show_wordcloud(text, bg='black', image='images/disc.png')
```

```
D:\Dev\Anaconda3\lib\site-packages\wordcloud\wordcloud.py:106: MatplotlibDeprecationWarning: The get_cmap function w
as 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 [13]:
```
news = df.iloc[:5]['Text']  # "df" stores previously loaded text
ta.summary_en(news, sent_len=5)
```

Out[13]:
```
['but ms cooper  who now runs her own consulting business  told a jury in new york on wednesday that external audito
rs arthur andersen had approved worldcom s accounting in early 2001 and 2002. she said andersen had given a  green l
ight  to the procedures and practices used by worldcom.',
 'cynthia cooper  worldcom s ex-head of internal accounting  alerted directors to irregular accounting practices at
the us telecoms giant in 2002. her warnings led to the collapse of the firm following the discovery of an $11bn (£5.
7bn) accounting fraud.',
 'prosecution lawyers have argued that mr ebbers orchestrated a series of accounting tricks at worldcom  ordering em
ployees to hide expenses and inflate revenues to meet wall street earnings estimates.',
 'the university of california said the trial in the case is scheduled to begin in october 2006. it joined the lawsu
it in december 2001alleging  massive insider trading  and fraud  claiming it had lost $145m on its investments in th
e company.',
 'the bbc s david willey in rome says one reason for that result is the changeover from the lira to the euro in 2001
which is widely viewed as the biggest reason why their wages and salaries are worth less than they used to be.']
```

5. Topic Modeling

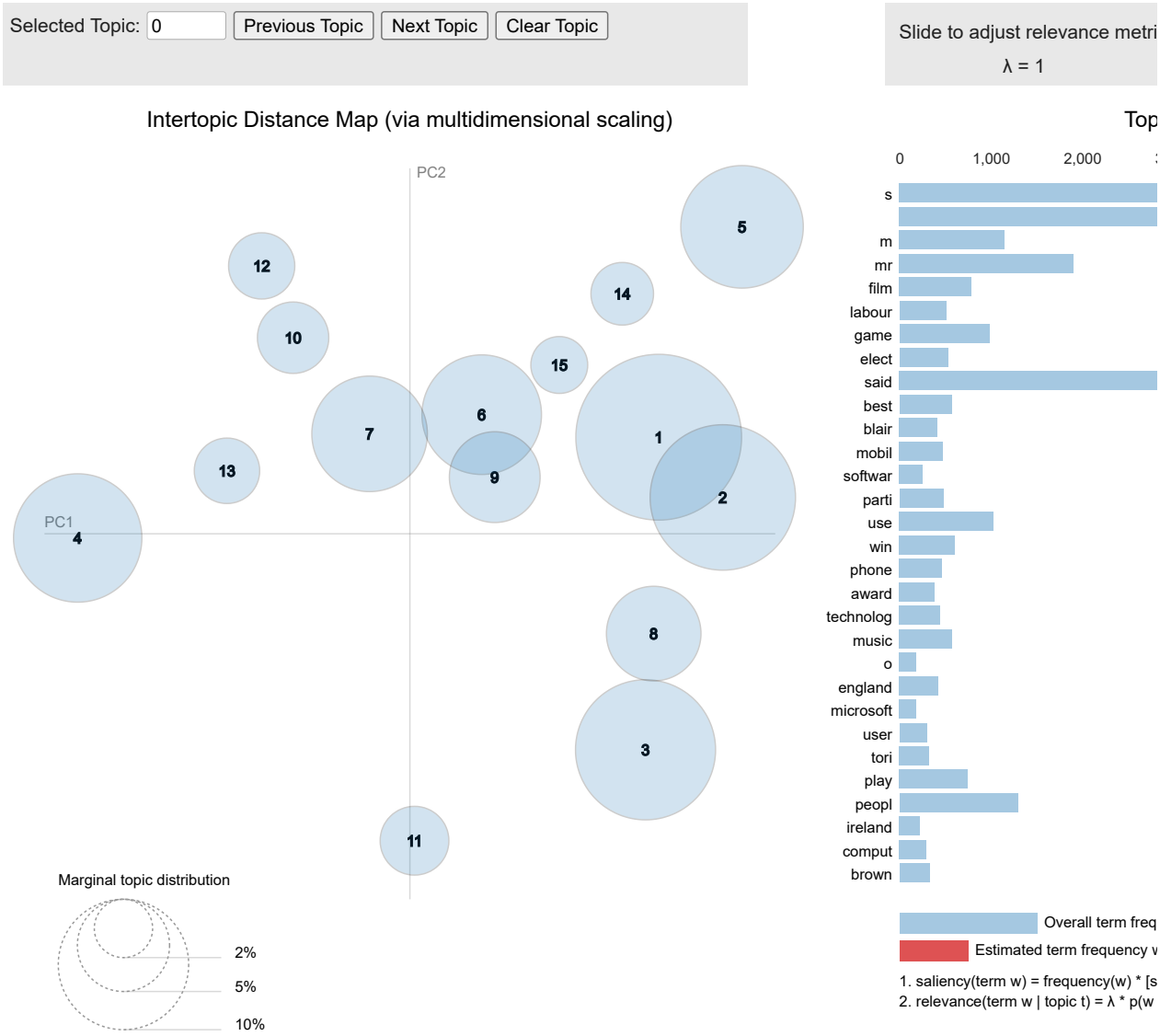
LDA Model

In [14]:
```
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,
  '0.019*"" + 0.013*"s" + 0.011*"game" + 0.009*"roddick" + 0.007*"said" + '
  '0.006*"nadal" + 0.006*"break" + 0.004*"point" + 0.004*"ball" + '
  '0.004*"minut"'),
 (1,
  '0.018*"s" + 0.015*"o" + 0.014*"said" + 0.009*"ireland" + 0.008*"" + '
  '0.007*"mr" + 0.006*"home" + 0.006*"arrest" + 0.006*"hous" + 0.005*"terror"'),
 (2,
  '0.016*"softwar" + 0.016*"s" + 0.013*"program" + 0.012*"microsoft" + '
  '0.009*"" + 0.009*"secur" + 0.009*"said" + 0.008*"user" + 0.007*"patent" + '
  '0.007*"spywar"'),
 (3,
  '0.028*"" + 0.022*"said" + 0.014*"s" + 0.006*"mr" + 0.006*"peopl" + '
  '0.005*"game" + 0.005*"year" + 0.005*"govern" + 0.005*"say" + 0.005*"use"'),
 (4,
  '0.029*"" + 0.024*"m" + 0.020*"s" + 0.009*"year" + 0.009*"world" + '
  '0.008*"best" + 0.007*"olymp" + 0.007*"holm" + 0.006*"win" + 0.006*"said"'),
 (5,
  '0.028*"" + 0.021*"s" + 0.012*"film" + 0.011*"best" + 0.010*"award" + '
  '0.010*"said" + 0.007*"star" + 0.006*"year" + 0.006*"actor" + 0.005*"mr"'),
 (6,
  '0.017*"said" + 0.017*"s" + 0.014*"" + 0.008*"m" + 0.006*"club" + '
  '0.006*"airlin" + 0.005*"new" + 0.005*"t" + 0.005*"arsenal" + '
  '0.004*"chelsea"'),
 (7,
  '0.046*"" + 0.019*"s" + 0.017*"said" + 0.013*"year" + 0.007*"bn" + '
  '0.006*"market" + 0.005*"growth" + 0.005*"£" + 0.005*"economi" + '
  '0.005*"price"'),
 (8,
  '0.021*"" + 0.014*"s" + 0.011*"said" + 0.011*"blog" + 0.008*"site" + '
  '0.008*"chart" + 0.007*"m" + 0.006*"search" + 0.005*"mr" + 0.005*"download"'),
 (9,
  '0.027*"s" + 0.021*"" + 0.014*"said" + 0.012*"film" + 0.010*"music" + '
  '0.006*"year" + 0.006*"band" + 0.005*"mr" + 0.004*"new" + 0.004*"yuko"'),
 (10,
  '0.022*"said" + 0.012*"" + 0.011*"mr" + 0.010*"s" + 0.008*"parti" + '
  '0.007*"peopl" + 0.006*"polic" + 0.005*"ukip" + 0.005*"hunt" + 0.004*"new"'),
 (11,
  '0.028*"s" + 0.024*"" + 0.009*"unit" + 0.005*"said" + 0.005*"number" + '
  '0.005*"best" + 0.004*"game" + 0.004*"v" + 0.004*"ferguson" + 0.004*"new"'),
 (12,
  '0.019*"" + 0.016*"use" + 0.013*"peopl" + 0.012*"said" + 0.012*"mobil" + '
  '0.012*"technolog" + 0.011*"phone" + 0.011*"s" + 0.008*"digit" + '
  '0.007*"gadget"'),
 (13,
  '0.024*"mr" + 0.021*"s" + 0.018*"said" + 0.017*"" + 0.013*"labour" + '
  '0.012*"elect" + 0.011*"blair" + 0.008*"parti" + 0.008*"brown" + '
  '0.008*"govern"'),
 (14,
  '0.023*"s" + 0.016*"" + 0.012*"game" + 0.011*"play" + 0.011*"win" + '
  '0.011*"england" + 0.009*"said" + 0.008*"player" + 0.006*"wale" + 0.006*"t"')]

Model Evaluation Scores:
  Coherence: 0.39748179276562057
  Perplexity: -7.854038950434769
  Topic diversity: 0.0008062968325630886
  Topic size distribution: 0.0018746652383502945
```

In [15]:
```python
# LDA Model Visualization
import pyLDAvis
pyLDAvis.display(lda.vis_data)
```

Out[15]:



BERTopic Model

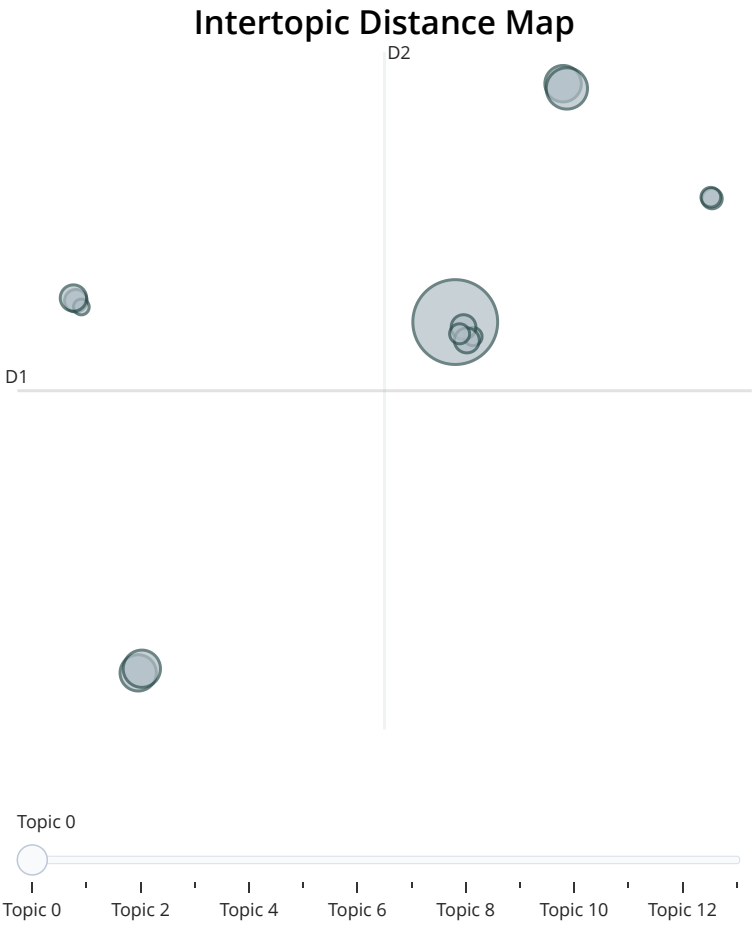
In [16]:

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

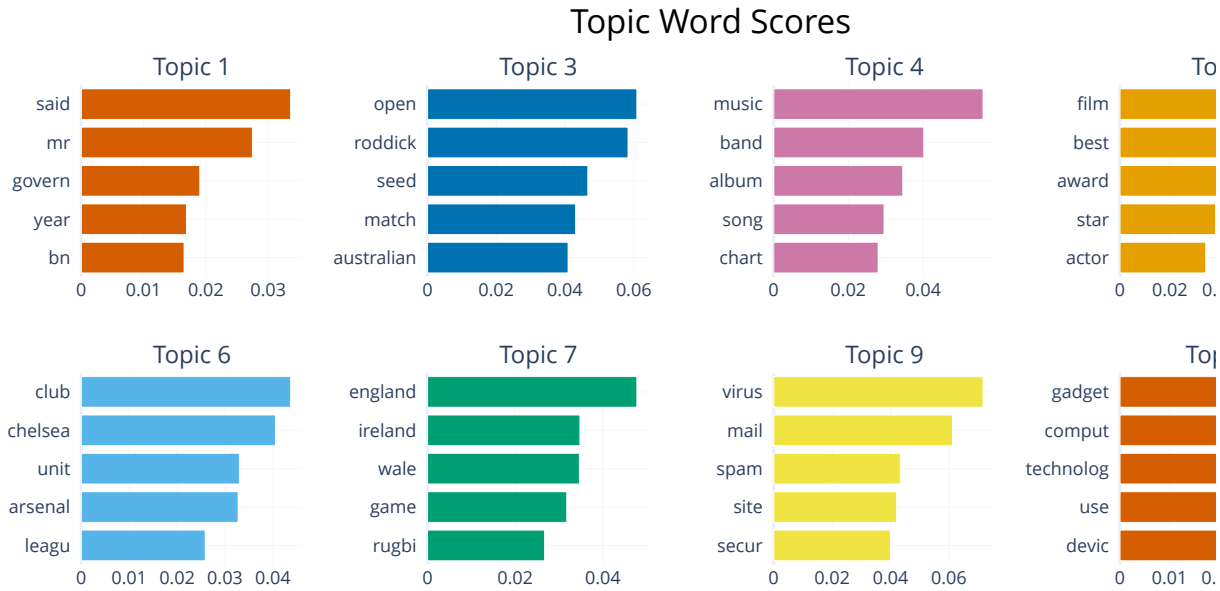
```
Corpus loaded!
Text preprocessed!
Some weights of the model checkpoint at bert-base-uncased were not used when initializing BertModel: ['cls.predictio
ns.transform.LayerNorm.weight', 'cls.predictions.transform.dense.weight', 'cls.predictions.bias', 'cls.predictions.d
ecoder.weight', 'cls.seq_relationship.weight', 'cls.predictions.transform.LayerNorm.bias', 'cls.seq_relationship.bia
s', 'cls.predictions.transform.dense.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with
another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactl
y identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
D:\Dev\Anaconda3\lib\site-packages\hdbscan\hdbscan_.py:1170: DeprecationWarning: `alltrue` is deprecated as of NumPy
1.25.0, and will be removed in NumPy 2.0. Please use `all` instead.
  self._all_finite = is_finite(X)
```

```
Text trained!

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

Model Evaluation Scores:
  Coherence: 0.5917232663101984

BERTopic Model Visualization:
```

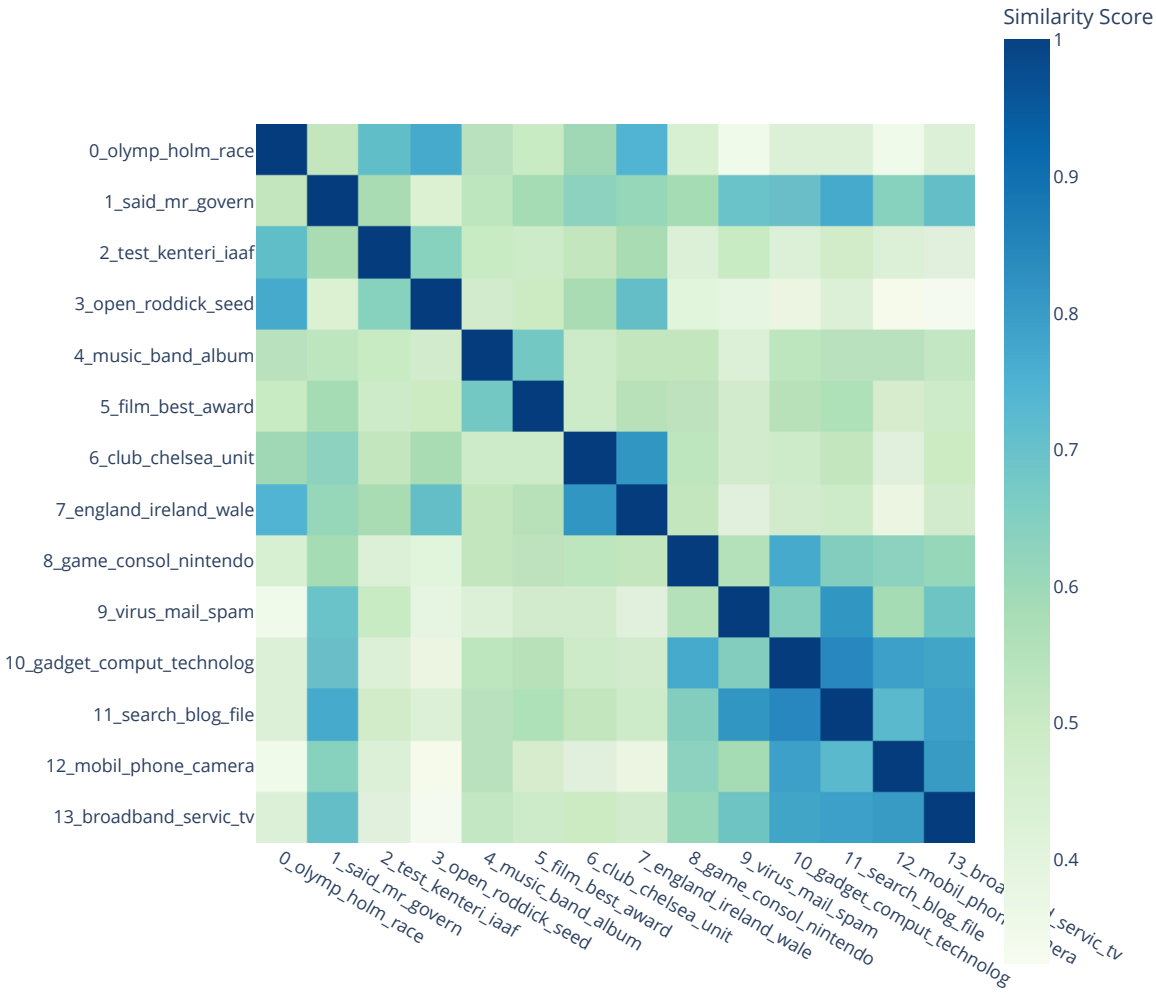
```
D:\Dev\Anaconda3\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

D:\Dev\Anaconda3\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.
```




```
D:\Dev\Anaconda3\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

D:\Dev\Anaconda3\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.
```




```
D:\Dev\Anaconda3\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

D:\Dev\Anaconda3\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.
```




```
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()
```