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. Pivot Table
6. Topic Modeling - LDA, BERTopic and NMF

CWordTM Toolkit's Documentation: https://cwordtm.readthedocs.io

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

1. Meta Information Features

In [2]:
```
import cwordtm
from cwordtm import *
```

```
WARNING:tensorflow:From D:\Dev\Anaconda3\envs\rl\lib\site-packages\tf_keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [3]:
```
cwordtm.__version__
```

Out[3]:
```
'0.7.1'
```

In [4]:
```
# Show brief module information
print(meta.get_module_info())
```

```
The member information of the module 'cwordtm'
1. Submodule meta:
   addin (func, *, timing=False, code=0)
   addin_all (modname='cwordtm', *, timing=False, code=0)
   addin_all_functions (submod, *, timing=False, code=0)
   get_function (mod_name, submodules, func_name, *, timing=False, code=0)
   get_module_info (detailed=False, *, timing=False, code=0)
   get_submodule_info (submodname, detailed=False, *, timing=False, code=0)
   import_module (name, package=None, *, timing=False, code=0)
   wraps (wrapped, assigned=('__module__', '__name__', '__qualname__', '__doc__', '__annotations__'), updated=('__dic
t__',), *, timing=False, code=0)
2. Submodule pivot:
   pivot (df, column='Category', *, timing=False, code=0)
   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)
   split_chi_sentences (text, *, 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 (doc_file, num_topics, chi=False, embed=True)
   LDA (doc_file, num_topics, chi=False)
   NMF (doc_file, num_topics, chi=False)
   btm_process (doc_file, num_topics=10, source=0, text_col='text', doc_size=0, cat=0, chi=False, group=True, eval=Fa
lse, *, timing=False, code=0)
   lda_process (doc_file, num_topics=10, source=0, text_col='text', doc_size=0, cat=0, chi=False, group=True, eval=Fa
lse, *, timing=False, code=0)
   load_bible (textfile, cat=0, group=True, *, timing=False, code=0)
   load_text (textfile, nr=0, text_col='text', *, timing=False, code=0)
   ngrams (sequence, n, *, timing=False, code=0, **kwargs)
   nmf_process (doc_file, num_topics=10, source=0, text_col='text', doc_size=0, cat=0, chi=False, group=True, eval=Fa
lse, *, timing=False, code=0)
   pprint (object, stream=None, indent=1, width=80, depth=None, *, compact=False, sort_dicts=True, underscore_numbers
=False, timing=False, code=0)
   process_text (doc, *, timing=False, code=0)
6. Submodule util:
   add_chi_vocab (*, timing=False, code=0)
   bible_cat_info (lang='en', *, 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_csv (file_obj, nr=0, info=False, *, 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)
   reset_rows (*, timing=False, code=0)
   set_lang (lang='en', *, timing=False, code=0)
   set_rows (n=None, *, 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, web_app=False, *, timing=False, code=0)
   plot_cloud (wordcloud, figsize, web_app=False, *, timing=False, code=0)
   show_wordcloud (docs, clean=False, figsize=(12, 8), bg='white', image=0, web_app=False, *, timing=False, code=0)

In [5]:
```
# 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, web_app=False):
    """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
    :param web_app: The flag indicating the function is initiated from a web
        application, default to False
    :type web_app: bool
    :return: The wordcloud figure
    :rtype: matplotlib.pyplot.figure
    """

    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',
    wordcloud = WordCloud(background_color=bg, colormap='rainbow',
                          mask=mask, font_path=str(font_file)) \
                    .generate_from_frequencies(frequencies=diction)

    return plot_cloud(wordcloud, figsize=figsize, web_app=web_app)

def plot_cloud(wordcloud, figsize, web_app=False):
    """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
    :param web_app: The flag indicating the function is initiated from a web
        application, default to False
    :type web_app: bool
    :return: The wordcloud figure
    :rtype: matplotlib.pyplot.figure
    """

    fig = plt.figure(figsize=figsize)
    plt.axis("off");
    plt.imshow(wordcloud)
    if web_app: return fig

def show_wordcloud(docs, clean=False, figsize=(12, 8), bg='white', image=0, web_app=False):
    """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
    :param web_app: The flag indicating the function is initiated from a web
        application, default to False
    :type web_app: bool
    :return: The wordcloud figure
    :rtype: matplotlib.pyplot.figure
    """

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

    return plot_cloud(wordcloud, figsize=figsize, web_app=web_app)
```

In [6]:
```
# Show execution time
bbc_news = "BBC/BBC News Train.csv"
df = util.load_text(bbc_news, timing=True)
```

```
Finished 'load_text' in 0.0941 secs
```

In [7]:
```
# Execute and show code
df = util.load_text(bbc_news, 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()]
        lines = list(filter(None, lines))

        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
```

In [8]:
```
# Show code without execution
df = util.load_text(bbc_news, 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()]
        lines = list(filter(None, lines))

        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
```

In [9]:
```
# 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 [10]:
```
df = util.load_text(bbc_news, 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 [12]:
```
import nltk
nltk.download('punkt_tab', quiet=True)
```

Out[12]: True

In [13]:
```
text_list = util.get_text_list(df.iloc[:500], text_col='Text')
text = util.preprocess_text(text_list)
```

3. Text Visualization - Word Cloud

In [14]:
```
# White background with no image mask
viz.show_wordcloud(text)
```

In [15]:
```
# Black background with the prescribed image as the mask
viz.show_wordcloud(text, bg='black', image=2)  # Disc
```

4. Text Summarization

In [17]:
```
import nltk
nltk.download('averaged_perceptron_tagger_eng', quiet=True)
```

Out[17]: True

In [18]:
```
news = df.iloc[:5]['Text']  # "df" stores previously loaded text
ta.summary_en(news, sent_len=5)
```

Out[18]: ['but ms cooper who now runs her own consulting business told a jury in new york on wednesday that external auditors arthur andersen had approved worldcom s accounting in early 2001 and 2002. she said andersen had given a green light to the procedures and practices used by worldcom.',
 'we re surprised that the ifo index has taken such a knock   said dz bank economist bernd weidensteiner.',
 'the poll found that a majority or plurality of people in 13 countries believed the economy was going downhill compared with respondents in nine countries who believed it was improving.',
 '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.',
 'we have to stop saying that these technologies will change their lives   he said.',
 'ericsson s research has shown that consumers divide into different tribes that use phones in different ways.',
 'dr bjorn said groups dubbed pioneers and materialists were most interested in trying new things and were behind the start of many trends in phone use.',
 'by contrast he said camera phones were being used much more to capture a moment and were being woven into everyday life.',
 'leading plaintiff the university of california announced the news adding that 10 of the former directors will pay $13m from their own pockets.',
 'the university of california said the trial in the case is scheduled to begin in october 2006. it joined the lawsuit in december 2001alleging massive insider trading and fraud claiming it had lost $145m on its investments in the company.',
 'mr ebbers has pleaded not guilty to charges of fraud and conspiracy.',
 'prosecution lawyers have argued that mr ebbers orchestrated a series of accounting tricks at worldcom ordering employees to hide expenses and inflate revenues to meet wall street earnings estimates.',
 'ms cooper also said that during shareholder meetings mr ebbers often passed over technical questions to the company s finance chief giving only brief answers himself.',
 'the prosecution s star witness former worldcom financial chief scott sullivan has said that mr ebbers ordered accounting adjustments at the firm telling him to hit our books .',
 'worldcom emerged from bankruptcy protection in 2004 and is now known as mci.',
 'munich-based research institute ifo said that its confidence index fell to 95.5 in february from 97.5 in january its first decline in three months.',
 'the study found that the outlook in both the manufacturing and retail sectors had worsened.',
 'economy and labour minister wolfgang clement called the dip in february s ifo confidence figure a very mild decline .',
 'he said that despite the retreat the index remained at a relatively high level and that he expected a modest economic upswing to continue.',
 'exports had kept things going during the first half of 2004 but demand for exports was then hit as the value of the euro hit record levels making german products less competitive overseas.',
 'analysts said that the ifo figures and germany s continuing problems may delay an interest rate rise by the european central bank.',
 'bbc poll indicates economic gloom citizens in a majority of nations surveyed in a bbc world service poll believe the world economy is worsening.',
 'but when asked about their own family s financial outlook a majority in 14 countries said they were positive about the future.',
 'while the world economy has picked up from difficult times just a few years ago people seem to not have fully absorbed this development though they are personally experiencing its effects   said pipa director steven kull.',
 'the countries where people were most optimistic both for the world and for their own families were two fast-growing developing economies china and india followed by indonesia.',
 'china has seen two decades of blistering economic growth which has led to wealth creation on a huge scale says the bbc s louisa lim in beijing.',
 'pipa conducted the poll from 15 november 2004 to 3 january 2005 across 22 countries in face-to-face or telephone interviews.',
 'the interviews took place between 15 november 2004 and 5 january 2005. the margin of error is between 2.5 and 4 points depending on the country.',
 'instead phone firms keen to get more out of their customers should not just be pushing the technology for its own sake.',
 'we should try to speak to consumers in their own language and help them see how it fits in with what they are doing   he told the bbc news website.',
 'people s habits remain the same   said dr bjorn.',
 'dr bjorn said that although consumers do what they always did but use a phone to do it the sheer variety of what the new handset technologies make possible does gradually drive new habits and lifestyles.',
 'for instance   he said   older people are using sms much more than they did five years ago.',
 'another factor governing the speed of change in mobile phone use was the simple speed with which new devices are bought by pioneers and materialists.',
 'dr bjorn said that early reports of camera phone usage in japan seemed to imply that the innovation was going to be a flop.',
 'dr bjorn said that people also used their camera phones in very different ways to film and even digital cameras.',
 'the settlement will be put to the courts for approval next week.',
 'its demise sent shockwaves through financial markets and dented investor confidence in corporate america.',
 'both men are facing criminal charges for their alleged misconduct in the run up to the firm s collapse.',
 'enron s shareholders are still seeking damages from a long list of other big name defendants including the financial institutions jp morgan chase citigroup merrill lynch and credit suisse first boston.',
 'worldcom ex-boss launches defence lawyers defending former worldcom chief bernie ebbers against a battery of fraud charges have called a company whistleblower as their first witness.',
 '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.',
 'mr ebber s lawyers have said he was unaware of the fraud arguing that auditors did not alert him to any problem
s.',
 'mr ebbers could face a jail sentence of 85 years if convicted of all the charges he is facing.',
 'last week mci agreed to a buyout by verizon communications in a deal valued at $6.75bn.',
 'observers had been hoping that a more confident business sector would signal that economic activity was picking u
p.',
 'germany s economy grew 1.6% last year after shrinking in 2003. however the economy contracted by 0.2% during the
last three months of 2004 mainly due to the reluctance of consumers to spend.',
 'latest indications are that growth is still proving elusive and ifo president hans-werner sinn said any improvemen
t in german domestic demand was sluggish.',
 'on top of that the unemployment rate has been stuck at close to 10% and manufacturing firms including daimlerchr
ysler siemens and volkswagen have been negotiating with unions over cost cutting measures.',
 'eurozone interest rates are at 2% but comments from senior officials have recently focused on the threat of infla
tion prompting fears that interest rates may rise.',
 'most respondents also said their national economy was getting worse.',
 'those surveyed in three countries were split.',
 'in percentage terms an average of 44% of respondents in each country said the world economy was getting worse co
mpared to 34% who said it was improving.',
 'and 47% saw their family s economic conditions improving as against 36% who said they were getting worse.',
 'the poll of 22 953 people was conducted by the international polling firm globescan together with the program on
international policy attitudes (pipa) at the university of maryland.',
 'people around the world are saying: i m ok but the world isn t .',
 'there may be a perception that war terrorism and religious and political divisions are making the world a worse p
lace even though that has not so far been reflected in global economic performance says the bbc s elizabeth blun
t.',
 'the philippines was among the most upbeat countries on prospects for respondents families but one of the most pe
ssimistic about the world economy.',
 'in eight of the countries the sample was limited to major metropolitan areas.',
 'historically in the industry there has been too much focus on using technology   said dr michael bjorn senior adv
isor on mobile media at ericsson s consumer and enterprise lab.',
 'for the study ericsson interviewed 14 000 mobile phone owners on the ways they use their phone.',
 'they just move the activity into the mobile phone as it s a much more convenient way to do it.',
 'youngsters use of text messages also reflects their desire to chat and keep in contact with friends and again jus
t lets them do it in a slightly changed way.',
 'only when about 25% of people have handsets with new innovations on them such as cameras can consumers stop worr
ying that if they send a picture message the person at the other end will be able to see it.',
 'once this significant number of users is passed use of new innovations tends to take off.',
 'digital cameras tend to be used on significant events such as weddings holidays and birthdays.',
 'enron went bankrupt in 2001 after it emerged it had hidden hundreds of millions of dollars in debt.',
 'the settlement is very significant in holding these outside directors at least partially personally responsible
william lerach the lawyer leading the class action suit against enron said.',
 'hopefully this will help send a message to corporate boardrooms of the importance of directors performing their l
egal duties   he added.',
 'the deal is the fourth major settlement negotiated by lawyers who filed a class action on behalf of enron s shareh
olders almost three years ago.',
 'neither does it cover andrew fastow who has pleaded guilty to taking part in an illegal conspiracy while he was c
hief financial officer at the group.',
 'however ms cooper said mr sullivan had not mentioned anything uncomfortable about worldcom s accounting during
a 2001 audit committee meeting.',
 'german business confidence slides german business confidence fell in february knocking hopes of a speedy recovery
in europe s largest economy.',
 'the main reason is probably that the domestic economy is still weak particularly in the retail trade.',
 'but the results also may reflect the untrammelled confidence of people who are subject to endless government propa
ganda about their country s rosy economic future our correspondent says.',
 'south korea was the most pessimistic while respondents in italy and mexico were also quite gloomy.',
 'lifestyle governs mobile choice faster better or funkier hardware alone is not going to help phone firms sell m
ore handsets research suggests.',
 'consumers are far more interested in how handsets fit in with their lifestyle than they are in screen size onboar
d memory or the chip inside shows an in-depth study by handset maker ericsson.',
 'one good example of this was diary-writing among younger people he said.',
 'however he said now 45% of the japanese people ericsson questioned use their camera phone at least once a mont
h.',
 'in 2003 the figure was 29%.',
 'similarly across europe the numbers of people taking snaps with cameras is starting to rise.',
 'in 2003 only 4% of the people in the uk took a phonecam snap at least once a month.',
 'now the figure is 14%.',
 'similar rises have been seen in many other european nations.',
 'usage patterns for digital cameras are almost exactly replacing usage patterns for analogue cameras   he said.',
 'enron bosses in $168m payout eighteen former enron directors have agreed a $168m (£89m) settlement deal in a share
holder lawsuit over the collapse of the energy firm.',
 'before its collapse the firm was the seventh biggest public us company by revenue.',
 'under the terms of the $168m settlement - $155m of which will be covered by insurance - none of the 18 former dire
ctors will admit any wrongdoing.',
 'so far including the latest deal just under $500m (£378.8m) has been retrieved for investors.',

```
 'however  the latest deal does not include former enron chief executives ken lay and jeff skilling.',
 'almost 23 000 people in 22 countries were questioned for the poll  which was mostly conducted before the asian tsu
nami disaster.',
 'similarly  48% were pessimistic about their national economy  while 41% were optimistic.',
 'while diaries have always been popular  a mobile phone -- especially one equipped with a camera -- helps them keep
it in a different form.',
 'this was because younger users  often the children of ageing mobile owners  encouraged older people to try it so t
hey could keep in touch.']
```

5. Pivot Table

In [19]:
```
pivot.pivot(df, column='Category')
```

Out[19]:

Category	Text
business	336
entertainment	273
politics	274
sport	346
tech	261
Total	1490

6. Topic Modeling

In [20]:
```
import warnings
warnings.filterwarnings('ignore')
```

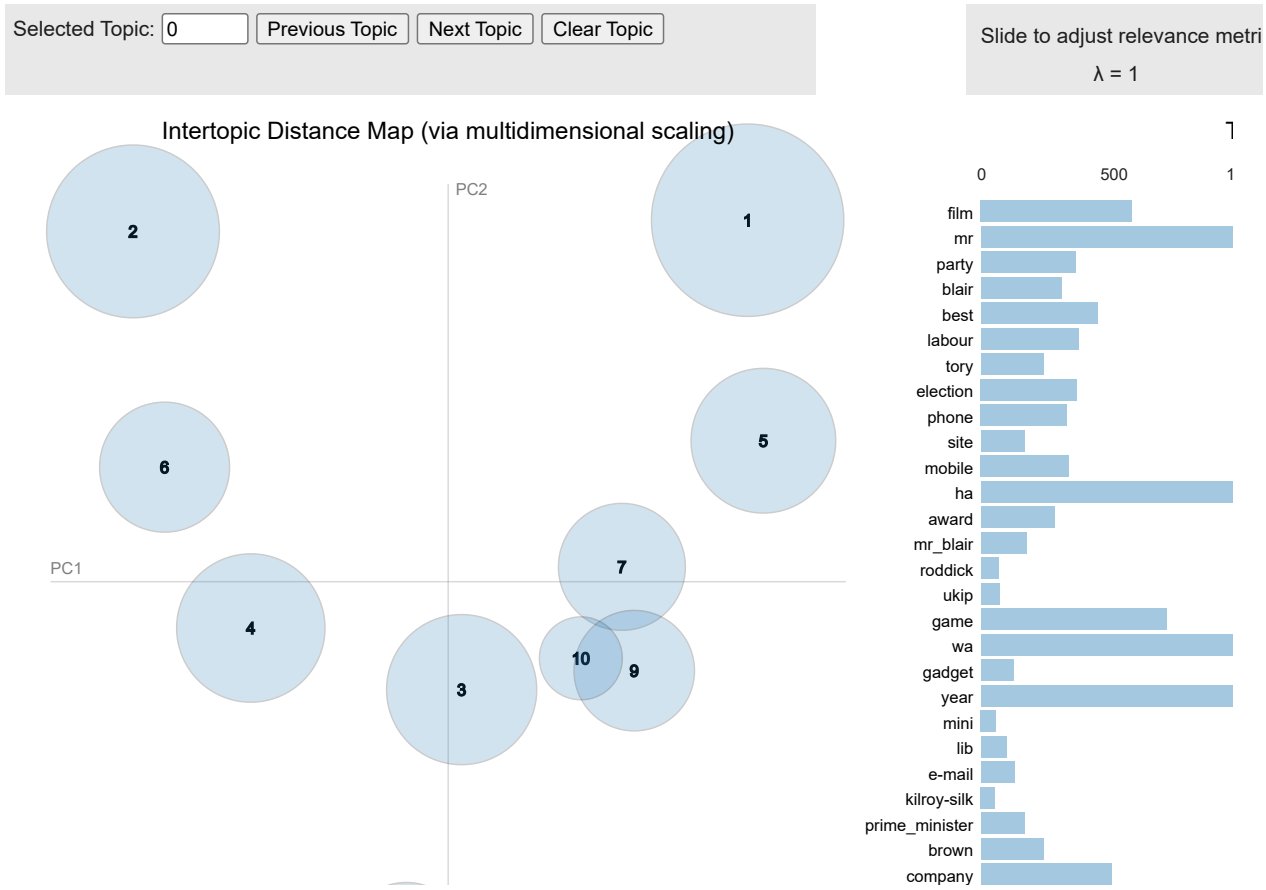
LDA Modeling

In [21]:
```
lda = tm.lda_process(bbc_news, source=1, text_col='Text', eval=True, timing=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
Corpus loaded!
Text preprocessed!
Text trained!
You may execute the following commands to show the visualization:
    import pyLDAvis
    pyLDAvis.display(lda.vis_data)

Visualization prepared!

Topics from LDA Model:
[(0,
  '0.006*"said" + 0.004*"ha" + 0.004*"wa" + 0.003*"best" + 0.002*"mr" + '
  '0.002*"film" + 0.002*"year" + 0.002*"new" + 0.002*"people" + '
  '0.002*"company"'),
 (1,
  '0.008*"said" + 0.005*"wa" + 0.004*"ha" + 0.003*"mr" + 0.002*"people" + '
  '0.002*"year" + 0.002*"new" + 0.002*"uk" + 0.002*"government" + '
  '0.002*"country"'),
 (2,
  '0.005*"wa" + 0.004*"said" + 0.003*"ha" + 0.002*"game" + 0.002*"time" + '
  '0.001*"win" + 0.001*"site" + 0.001*"england" + 0.001*"mr" + 0.001*"year"'),
 (3,
  '0.004*"wa" + 0.004*"ha" + 0.004*"said" + 0.003*"film" + 0.002*"year" + '
  '0.002*"mr" + 0.001*"new" + 0.001*"time" + 0.001*"world" + 0.001*"people"'),
 (4,
  '0.007*"said" + 0.006*"wa" + 0.005*"ha" + 0.003*"year" + 0.002*"new" + '
  '0.002*"people" + 0.002*"service" + 0.002*"mr" + 0.001*"game" + '
  '0.001*"government"'),
 (5,
  '0.005*"said" + 0.003*"wa" + 0.003*"ha" + 0.003*"year" + 0.002*"people" + '
  '0.002*"new" + 0.002*"game" + 0.001*"time" + 0.001*"phone" + '
  '0.001*"technology"'),
 (6,
  '0.003*"ha" + 0.003*"said" + 0.002*"wa" + 0.002*"year" + 0.001*"film" + '
  '0.001*"mr" + 0.001*"company" + 0.001*"new" + 0.001*"say" + 0.001*"people"'),
 (7,
  '0.006*"wa" + 0.006*"said" + 0.004*"ha" + 0.003*"mr" + 0.002*"year" + '
  '0.002*"people" + 0.002*"new" + 0.001*"music" + 0.001*"firm" + 0.001*"film"'),
 (8,
  '0.006*"said" + 0.005*"mr" + 0.004*"wa" + 0.004*"ha" + 0.002*"party" + '
  '0.002*"year" + 0.002*"labour" + 0.002*"say" + 0.002*"new" + 0.002*"tory"'),
 (9,
  '0.007*"said" + 0.007*"wa" + 0.004*"ha" + 0.003*"mr" + 0.002*"year" + '
  '0.002*"people" + 0.001*"new" + 0.001*"time" + 0.001*"party" + 0.001*"say"')]

Model Evaluation Scores:
  Coherence: 0.6729429469526675
  Perplexity: -11.232006171889378
  Topic diversity: 0.0007284423351101676
  Topic size distribution: 0.0018270401948842874

Finished 'lda_process' in 61.7707 secs
```

In [22]:
```
# LDA Model Visualization
import pyLDAvis
pyLDAvis.display(lda.vis_data)
```

Out[22]:



Save LDA Model

In [23]:
```
lda.save("models/lda_bbc.gensim")
```

```
LDA model has been stored in 'models/lda_bbc.gensim'.
```

Load LDA Model

In [24]:
```
lda2 = tm.LDA("", lda.num_topics)
lda2.model = lda2.load("models/lda_bbc.gensim")
lda2.show_topics()
```

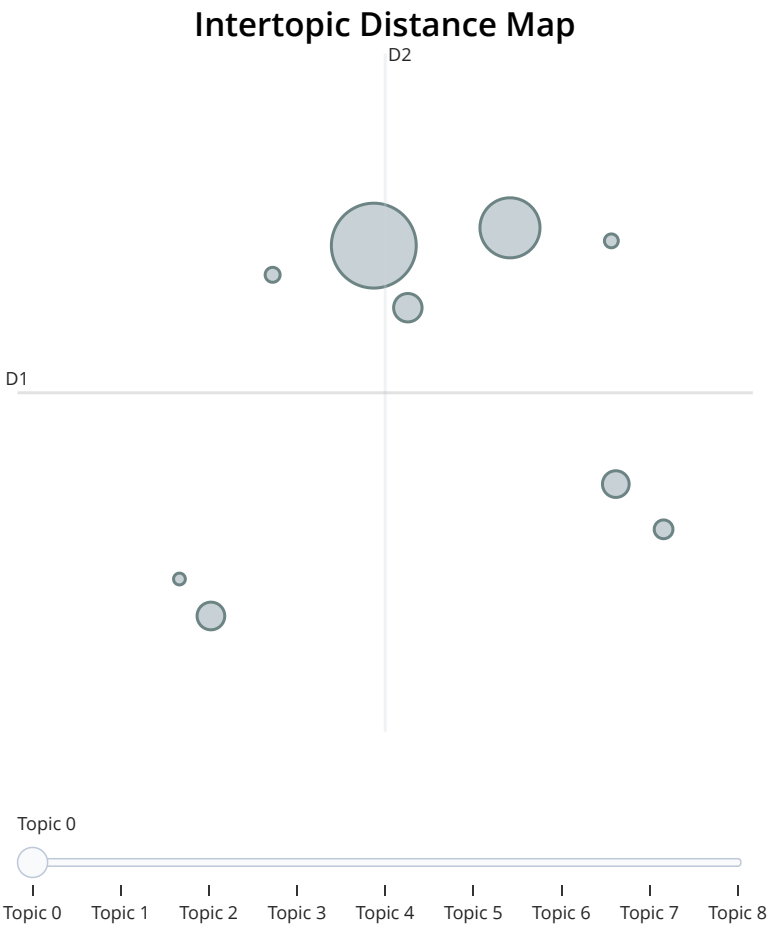
```
Topics from LDA Model:
[(0,
  '0.006*"said" + 0.004*"ha" + 0.004*"wa" + 0.003*"best" + 0.002*"mr" + '
  '0.002*"film" + 0.002*"year" + 0.002*"new" + 0.002*"people" + '
  '0.002*"company"'),
 (1,
  '0.008*"said" + 0.005*"wa" + 0.004*"ha" + 0.003*"mr" + 0.002*"people" + '
  '0.002*"year" + 0.002*"new" + 0.002*"uk" + 0.002*"government" + '
  '0.002*"country"'),
 (2,
  '0.005*"wa" + 0.004*"said" + 0.003*"ha" + 0.002*"game" + 0.002*"time" + '
  '0.001*"win" + 0.001*"site" + 0.001*"england" + 0.001*"mr" + 0.001*"year"'),
 (3,
  '0.004*"wa" + 0.004*"ha" + 0.004*"said" + 0.003*"film" + 0.002*"year" + '
  '0.002*"mr" + 0.001*"new" + 0.001*"time" + 0.001*"world" + 0.001*"people"'),
 (4,
  '0.007*"said" + 0.006*"wa" + 0.005*"ha" + 0.003*"year" + 0.002*"new" + '
  '0.002*"people" + 0.002*"service" + 0.002*"mr" + 0.001*"game" + '
  '0.001*"government"'),
 (5,
  '0.005*"said" + 0.003*"wa" + 0.003*"ha" + 0.003*"year" + 0.002*"people" + '
  '0.002*"new" + 0.002*"game" + 0.001*"time" + 0.001*"phone" + '
  '0.001*"technology"'),
 (6,
  '0.003*"ha" + 0.003*"said" + 0.002*"wa" + 0.002*"year" + 0.001*"film" + '
  '0.001*"mr" + 0.001*"company" + 0.001*"new" + 0.001*"say" + 0.001*"people"'),
 (7,
  '0.006*"wa" + 0.006*"said" + 0.004*"ha" + 0.003*"mr" + 0.002*"year" + '
  '0.002*"people" + 0.002*"new" + 0.001*"music" + 0.001*"firm" + 0.001*"film"'),
 (8,
  '0.006*"said" + 0.005*"mr" + 0.004*"wa" + 0.004*"ha" + 0.002*"party" + '
  '0.002*"year" + 0.002*"labour" + 0.002*"say" + 0.002*"new" + 0.002*"tory"'),
 (9,
  '0.007*"said" + 0.007*"wa" + 0.004*"ha" + 0.003*"mr" + 0.002*"year" + '
  '0.002*"people" + 0.001*"new" + 0.001*"time" + 0.001*"party" + 0.001*"say"')]
```

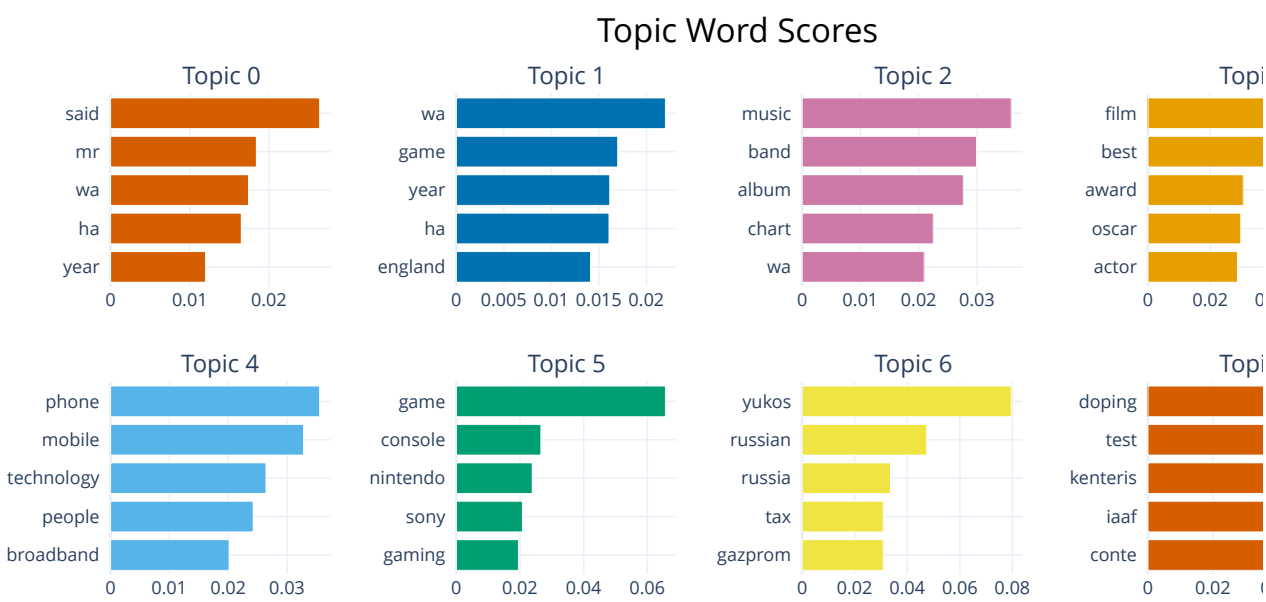
BERTopic Modeling

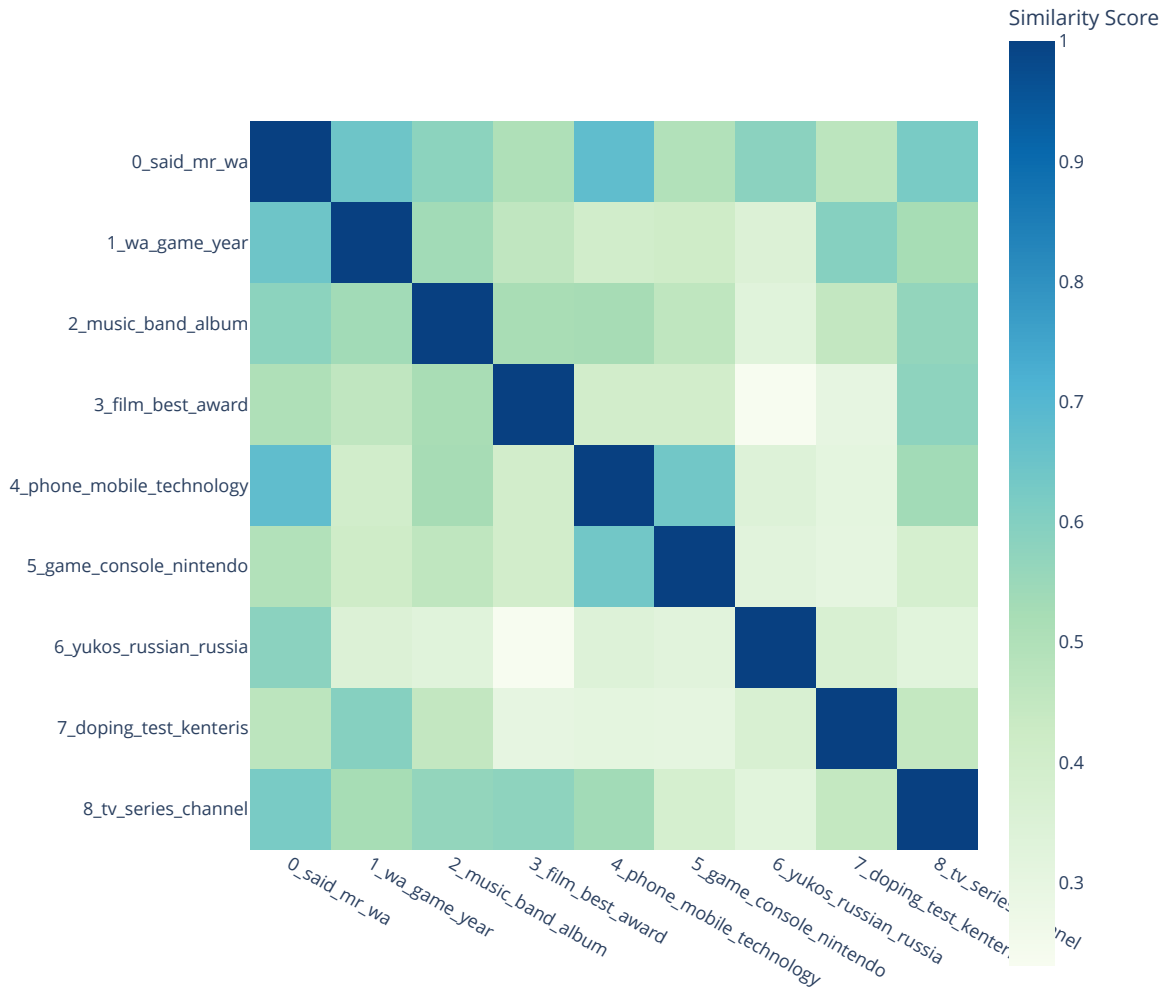
In [25]:
```
btm = tm.btm_process(bbc_news, source=1, text_col='Text', eval=True, timing=True)
```

```
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
Corpus loaded!
Text preprocessed!
modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]
config_sentence_transformers.json:   0%|          | 0.00/116 [00:00<?, ?B/s]
README.md:   0%|          | 0.00/10.7k [00:00<?, ?B/s]
sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]
config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]
tokenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
1_Pooling/config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]
```

```
Text trained!

Topics from BERTopic Model:
Topic 0: said | mr | wa | ha | year | government | people | election | bn | labour
Topic 1: wa | game | year | ha | england | said | win | time | half | player
Topic 2: music | band | album | chart | wa | song | year | single | said | rock
Topic 3: film | best | award | oscar | actor | star | wa | director | actress | aviator
Topic 4: phone | mobile | technology | people | broadband | service | said | digital | tv | camera
Topic 5: game | console | nintendo | sony | gaming | mobile | gadget | title | gamers | xbox
Topic 6: yukos | russian | russia | tax | gazprom | oil | company | ha | bn | court
Topic 7: doping | test | kenteris | iaaf | conte | greek | drug | thanou | sprinter | athens
Topic 8: tv | series | channel | bbc | audience | television | rating | said | fox | viewer

Model Evaluation Scores:
  Coherence: 0.6584994938126024

BERTopic Model Visualization:
```




Topic Word Scores
Topic 0
said
mr
wa
ha
year
0
0.01
0.02
Topic 1
wa
game
year
ha
england
0
0.005
0.01
0.015
0.02
Topic 2
music
band
album
chart
wa
0
0.01
0.02
0.03
Topi
film
best
award
oscar
actor
0
0.02
Topic 4
phone
mobile
technology
people
broadband
0
0.01
0.02
0.03
Topic 5
game
console
nintendo
sony
gaming
0
0.02
0.04
0.06
Topic 6
yukos
russian
russia
tax
gazprom
0
0.02
0.04
0.06
0.08
Topi
doping
test
kenteris
iaaf
conte
0
0.02




```
Finished 'btm_process' in 153.8618 secs
```

Save BERTopic Model

In [26]:
```
btm.save("models/bertopic_bbc.pickle")
```

```
2024-11-25 17:50:51,190 - BERTopic - WARNING: When you use `pickle` to save/load a BERTopic model,please make sure th
at the environments in which you saveand load the model are **exactly** the same. The version of BERTopic,its depende
ncies, and python need to remain the same.
BERTopic model has been stored in 'models/bertopic_bbc.pickle'.
```

Load BERTopic Model

In [27]:
```
btm2 = tm.BTM("", btm.num_topics)
btm2.model = btm2.load("models/bertopic_bbc.pickle")
btm2.show_topics()
```

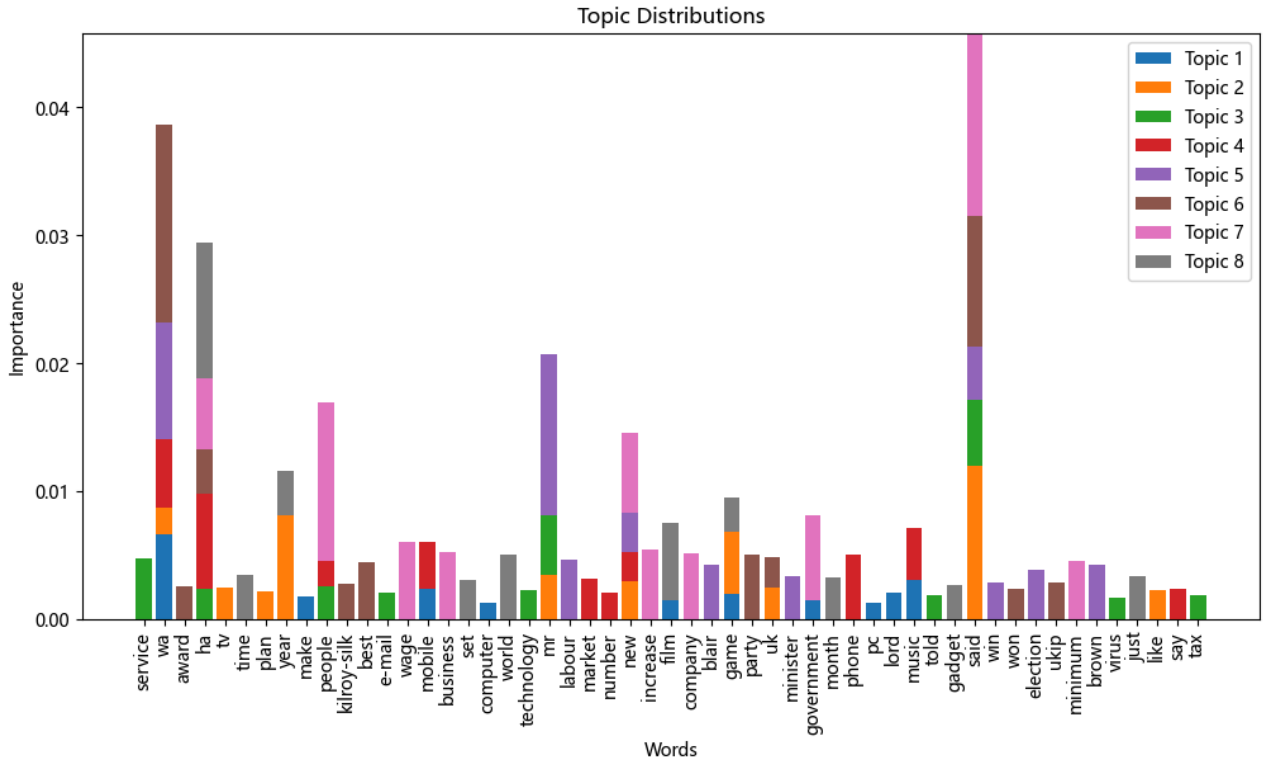
```
Topics from BERTopic Model:
Topic 0: said | mr | wa | ha | year | government | people | election | bn | labour
Topic 1: wa | game | year | ha | england | said | win | time | half | player
Topic 2: music | band | album | chart | wa | song | year | single | said | rock
Topic 3: film | best | award | oscar | actor | star | wa | director | actress | aviator
Topic 4: phone | mobile | technology | people | broadband | service | said | digital | tv | camera
Topic 5: game | console | nintendo | sony | gaming | mobile | gadget | title | gamers | xbox
Topic 6: yukos | russian | russia | tax | gazprom | oil | company | ha | bn | court
Topic 7: doping | test | kenteris | iaaf | conte | greek | drug | thanou | sprinter | athens
Topic 8: tv | series | channel | bbc | audience | television | rating | said | fox | viewer
```

NMF Modeling

In [28]:
```
nmf = tm.nmf_process(bbc_news, num_topics=8, source=1, text_col='Text', eval=True, timing=True, code=1)
```

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

Topics-Words from NMF Model:
Topic 1:
wa (0.006664)
music (0.003104)
mobile (0.002354)
lord (0.002106)
game (0.001928)
make (0.001744)
government (0.001472)
film (0.001448)
pc (0.001240)
computer (0.001224)

Topic 2:
said (0.011999)
year (0.008132)
game (0.004856)
mr (0.003420)
new (0.002976)
uk (0.002454)
tv (0.002431)
like (0.002299)
plan (0.002165)
wa (0.001994)

Topic 3:
said (0.005131)
service (0.004791)
mr (0.004677)
people (0.002527)
ha (0.002387)
technology (0.002278)
e-mail (0.002035)
tax (0.001902)
told (0.001870)
virus (0.001691)

Topic 4:
ha (0.007368)
wa (0.005367)
phone (0.005010)
music (0.004010)
mobile (0.003701)
market (0.003178)
say (0.002403)
new (0.002257)
number (0.002056)
people (0.001990)

Topic 5:
mr (0.012616)
wa (0.009169)
labour (0.004658)
blair (0.004285)
brown (0.004268)
said (0.004147)
election (0.003837)
minister (0.003368)
new (0.003119)
win (0.002853)
```

```
Topic 6:
wa (0.015449)
said (0.010216)
party (0.005047)
best (0.004406)
ha (0.003555)
ukip (0.002891)
kilroy-silk (0.002738)
award (0.002586)
uk (0.002421)
won (0.002382)

Topic 7:
said (0.014298)
people (0.012374)
government (0.006646)
new (0.006233)
wage (0.006070)
ha (0.005542)
increase (0.005462)
business (0.005241)
company (0.005100)
minimum (0.004553)

Topic 8:
ha (0.010574)
film (0.006055)
world (0.005031)
time (0.003483)
year (0.003468)
just (0.003404)
month (0.003206)
set (0.003066)
game (0.002754)
gadget (0.002658)
```



```
Model Evaluation Scores:
  Coherence: 0.5921949460698812
  Topic diversity: 0.0007065822696027979
  Topic size distribution: 0.0010397712503249284

Finished 'nmf_process' in 37.5477 secs

def nmf_process(doc_file, num_topics=10, source=0, text_col='text', doc_size=0, cat=0, chi=False, group=True, eval=Fa
lse):
    """Pipelines the NMF modeling.

    :param doc_file: The filename of the prescribed text file to be loaded,
        or a BytesIO object from Streamlit's file_uploader, default to None
    :type doc_file: str or io.BytesIO
    :param num_topics: The number of topics to be modeled, default to 10
    :type num_topics: int, optional
    :param source: The source of the prescribed document file ('doc_file'),
        where 0 refers to internal store of the package and 1 to external file,
        default to 0
    :type source: int, optional
    :param text_col: The name of the text column to be extracted, default to 'text'
    :type text_col: str, optional
    :param doc_size: The number of documents for external text to be processed,
        0 represents all documents, default to 0
    :type doc_size: int, optional
    :param cat: The category indicating a subset of the Scripture to be loaded, where
        0 stands for the whole Bible, 1 for OT, 2 for NT, or one of the ten categories
        ['tor', 'oth', 'ket', 'map', 'mip', 'gos', 'nth', 'pau', 'epi', 'apo'] (See
        the package's internal file 'data/book_cat.csv'), default to 0
    :type cat: int or str, optional
    :param chi: The flag indicating whether the text is processed as Chinese (True)
        or English (False), default to False
    :type chi: bool, optional
    :param group: The flag indicating whether the loaded text is grouped by chapter,
        default to True
    :type group: bool, optional
    :param eval: The flag indicating whether the model evaluation results will be shown,
        default to False
    :type eval: bool, optional
    :return: The pipelined NMF
    :rtype: cwordtm.tm.NMF object
    """

    nmf = NMF(doc_file, num_topics, chi)
    if source == 0:
        nmf.docs = load_bible(nmf.doc_file, cat=cat, group=group)
    else:
        nmf.docs = load_text(nmf.doc_file, doc_size, text_col)

    print("Corpus loaded!")

    if chi:
        nmf.preprocess_chi()
    else:
        nmf.preprocess()
    print("Text preprocessed!")

    nmf.fit()
    print("Text trained!")
    nmf.show_topics_words()
    nmf.viz()

    if eval:
        print("\nModel Evaluation Scores:")
        nmf.evaluate()

    return nmf

>> cwordtm.tm.NMF
class NMF:
    """The NMF object for Non-negative Matrix Factorization (NMF) modeling.

    :cvar num_topics: The number of topics to be modeled, default to 10
    :vartype num_topics: int
    :ivar doc_file: The filename of the text file to be processed
```

```
:vartype doc_file: str
:ivar chi: The flag indicating whether the processed text is in Chinese or not,
    True stands for Traditional Chinese or False for English
:vartype chi: bool
:ivar num_topics: The number of topics set for the topic model
:vartype num_topics: int
:ivar docs: The collection of the original documents to be processed
:vartype docs: pandas.DataFrame or list
:ivar pro_docs: The collection of documents, in form of list of lists of words
    after text preprocessing
:vartype pro_docs: list
:ivar dictionary: The dictionary of word ids with their tokenized words
    from preprocessed documents ('pro_docs')
:vartype dictionary: gensim.corpora.Dictionary
:ivar corpus: The list of documents, where each document is a list of tuples
    (word id, word frequency in the particular document)
:vartype corpus: list
:ivar model: The NMF model object
:vartype model: gensim.models.Nmf
"""

def __init__(self, doc_file, num_topics, chi=False):
    """Constructor method.
    """

    self.doc_file = doc_file
    self.num_topics = num_topics
    self.chi = chi
    self.docs = None
    self.pro_docs = None
    self.dictionary = None
    self.corpus = None
    self.model = None


def preprocess(self):
    """Process the original English documents (cwordtm.tm.NMF.docs)
    by invoking cwordtm.tm.process_text, and build a dictionary
    and a corpus from the preprocessed documents for the NMF model.
    """

    self.pro_docs = [process_text(doc) for doc in self.docs]

    for i, doc in enumerate(self.pro_docs):
        self.pro_docs[i] += ["_".join(w) for w in ngrams(doc, 2)]
        # self.pro_docs[i] += ["_".join(w) for w in ngrams(doc, 3)]

    # Create a dictionary and corpus for the NMF model
    self.dictionary = corpora.Dictionary(self.pro_docs)
    self.corpus = [self.dictionary.doc2bow(doc) for doc in self.pro_docs]


def preprocess_chi(self):
    """Process the original Chinese documents (cwordtm.tm.NMF.docs)
    by tokenizing text, removing stopwords, and building a dictionary
    and a corpus from the preprocessed documents for the NMF model.
    """

    # Build stop words
    stop_file = files('cwordtm.data').joinpath("tc_stopwords_2.txt")
    stopwords = [k[:-1] for k in open(stop_file, encoding='utf-8')\
                 .readlines() if k != '']

    # Tokenize"the text using Jieba
    dict_file = files('cwordtm.data').joinpath("user_dict_4.txt")
    jieba.load_userdict(str(dict_file))
    docs = [jieba.cut(doc) for doc in self.docs]

    # Replace special characters
    docs = [[word.replace('\u3000', ' ') for word in doc] \
                                  for doc in docs]

    # Remove stop words
    self.pro_docs = [' '.join([word for word in doc if word not in stopwords]) \
                                     for doc in docs]
```

```
    self.pro_docs = [doc.split() for doc in self.pro_docs]

    # Create a dictionary and corpus
    self.dictionary = corpora.Dictionary(self.pro_docs)
    self.corpus = [self.dictionary.doc2bow(doc) for doc in self.pro_docs]


def fit(self):
    """Build the NMF model with the created corpus and dictionary.
    """

    self.model = models.Nmf(self.corpus,
                            num_topics=self.num_topics)


def show_topics_words(self):
    """Shows the topics with their keywords from the built NMF model.
    """

    print("\nTopics-Words from NMF Model:")
    for topic_id in range(self.num_topics):
        topic_words = self.model.show_topic(topic_id, topn=10)
        print(f"Topic {topic_id+1}:")
        for word_id, prob in topic_words:
            word = self.dictionary[int(word_id)]
            print("%s (%.6f)" %(word, prob))
        print()


def viz(self):
    """Plot the topic distributions as a stacked bar chart for the built NMF model.
    """

    # Build a list of word ids from the built topics
    word_ids = []
    for topic_id in range(self.num_topics):
        topic_words = self.model.show_topic(topic_id, topn=10)
        for word_id, _ in topic_words:
            word_ids.append(int(word_id))

    word_ids = list(set(word_ids))

    # Create a topic distribution table
    topic_dist = np.zeros((self.num_topics, len(word_ids)))
    for topic_id in range(self.num_topics):
        topic_words = self.model.show_topic(topic_id, topn=10)
        for word_id, prob in topic_words:
            topic_dist[topic_id, word_ids.index(int(word_id))] = prob

    # Build a list of distinct words from the word id list
    word_list = []
    for i in range(len(word_ids)):
        word_list.append(self.dictionary[word_ids[i]])

    # Plot the topic distributions
    matplotlib.rcParams['font.family'] = ['Microsoft YaHei']
    plt.figure(figsize=(12, 6))

    bottom = np.zeros(len(word_list))
    for i, topic in enumerate(topic_dist):
        plt.bar(word_list, topic, width=0.8, bottom=bottom, label=f"Topic {i+1}")
        bottom += topic

    plt.xticks(range(len(word_list)), word_list, rotation=90)
    plt.title("Topic Distributions")
    plt.xlabel("Words")
    plt.ylabel("Importance")
    plt.legend(loc="best")
    plt.show()


def evaluate(self):
    """Computes and outputs the coherence score, topic diversity,
    and topic size distribution.
```

```
        """

        # Compute coherence score
        coherence_model = CoherenceModel(model=self.model,
                                         texts=self.pro_docs,
                                         dictionary=self.dictionary,
                                         coherence='c_v')
        print(f"  Coherence: {coherence_model.get_coherence()}")

        # Compute topic diversity
        topic_sizes = [len(self.model[self.corpus[i]]) for i in range(len(self.corpus))]
        total_docs = sum(topic_sizes)
        topic_diversity = sum([(size/total_docs)**2 for size in topic_sizes])
        print(f"  Topic diversity: {topic_diversity}")

        # Compute topic size distribution
        # topic_sizes = [len(self.model[self.corpus[i]]) for i in range(len(self.corpus))]
        topic_size_distribution = max(topic_sizes) / sum(topic_sizes)
        print(f"  Topic size distribution: {topic_size_distribution}\n")


    def save(self, file):
        """Saves the built NMF model to the specified file.

        :param file: The name of the file to store the built model, default to None
        :type file: str
        """

        if file is None or len(file.strip())==0:
            print("No valid filename has been specifid!")
            return

        base_name = file.split('.')[0]
        model_file = base_name + '_model.gensim'
        dict_file = base_name + '_dictionary.gensim'
        self.model.save(model_file)
        self.dictionary.save(dict_file)
        # corpora.MmCorpus.serialize(base_name+'_corpus.mm', self.corpus)
        print(f"NMF model has been saved: {model_file!r} and {dict_file!r}")


    def load(self, file):
        """Loads the stored NMF model from the specified file.

        :param file: The name of the file to be loaded, default to None
        :type file: str
        :return: The loaded NMF model and the loaded dictionary of the NMF's corpus
        :rtype: gensim.models.Nmf, gensim.corpora.Dictionary
        """

        if file is None or len(file.strip())==0:
            print("No valid filename has been specifid!")
            return

        base_name = file.split('.')[0]
        model_file = base_name + '_model.gensim'
        dict_file = base_name + '_dictionary.gensim'
        try:
            loaded_model = models.Nmf.load(model_file)
            loaded_dict = corpora.Dictionary.load(dict_file)
        except:
            print("Moldel file or dictionary file cannot be loaded!")
            return

        return loaded_model, loaded_dict

>> cwordtm.tm.load_bible
def load_bible(textfile, cat=0, group=True):
    """Loads and returns the Bible Scripture from the prescribed internal
    file ('textfile').

    :param textfile: The package's internal Bible text from which the text is loaded,
        either World English Bible ('web.csv') or Chinese Union Version (Traditional)
        ('cuv.csv'), default to None
    :type textfile: str
```

```
    :param cat: The category indicating a subset of the Scripture to be loaded, where
        0 stands for the whole Bible, 1 for OT, 2 for NT, or one of the ten categories
        ['tor', 'oth', 'ket', 'map', 'mip', 'gos', 'nth', 'pau', 'epi', 'apo'] (See
        the package's internal file 'data/book_cat.csv'), default to 0
    :type cat: int or str, optional
    :param group: The flag indicating whether the loaded text is grouped by chapter,
        default to True
    :type group: bool, optional
    :return: The collection of Scripture loaded
    :rtype: pandas.DataFrame
    """

    # textfile = "web.csv"
    scfile = files('cwordtm.data').joinpath(textfile)
    print("Loading Bible '%s' ..." %scfile)
    df = pd.read_csv(scfile)

    cat_list = ['tor', 'oth', 'ket', 'map', 'mip',\
                'gos', 'nth', 'pau', 'epi', 'apo']
    cat = str(cat)
    if cat == '1' or cat == 'ot':
        df = util.extract(df, testament=0)
    elif cat == '2' or cat == 'nt':
        df = util.extract(df, testament=1)
    elif cat in cat_list:
        df = util.extract(df, category=cat)

    if group:
        # Group verses into chapters
        df = df.groupby(['book_no', 'chapter'])\
                        .agg({'text': lambda x: ' '.join(x)})\
                .reset_index()

    df.text = df.text.str.replace('  ', '')
    return list(df.text)

>> cwordtm.tm.load_text
def load_text(textfile, nr=0, text_col='text'):
    """Loads and returns the list of documents from the prescribed file ('textfile').

    :param textfile: The prescribed text file from which the text is loaded,
        default to None
    :type textfile: str
    :param nr: The number of rows of text to be loaded; 0 represents all rows,
        default to 0
    :type nr: int, optional
    :param text_col: The name of the text column to be extracted, default to 'text'
    :type text_col: str, optional
    :return: The list of documents loaded
    :rtype: list
    """

    docs = util.load_text(textfile, nr, text_col)
    return list(docs[text_col])
```