tweetTopicModeling

August 5, 2023

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import warnings
warnings.filterwarnings("ignore")
```

[2]:
```python
data = pd.read_csv('final_twitterDatanew.csv')
data = data[~data.duplicated(subset = ['text'])]
data
```

[2]:
```
                       user_location   latitude  longitude  \
0                    London, England  51.507336  -0.127650
1                       East Cheshire  53.089516  -2.432569
2                  Kensington, London  51.500842  -0.179150
3                                  UK  54.702354  -3.276575
4                 Birmingham, England  52.479699  -1.902691
…                                 …          …          …
7678                         Auvergne  54.421697  -1.234967
7679  Boulogne-Billancourt, France        NaN        NaN
7680                    Paris, France        NaN        NaN
7681        La Roche-sur-Yon, France        NaN        NaN
7682                         Auvergne  54.421697  -1.234967

                     created_at                   id  \
0     2023-06-24 21:16:45+00:00  1672715409433190400
1     2023-06-24 21:08:08+00:00  1672713238255992834
2     2023-06-24 20:39:56+00:00  1672706144815415296
3     2023-06-24 20:38:54+00:00  1672705883921326081
4     2023-06-24 20:33:47+00:00  1672704594655191047
…                           …                    …
7678  2023-06-19 02:02:15+00:00  1670612929178116097
7679  2023-06-18 22:05:34+00:00  1670553363966984196
7680  2023-06-18 19:13:07+00:00  1670509969148354560
7681  2023-06-18 18:35:13+00:00  1670500430122561542
7682  2023-06-18 17:02:07+00:00  1670477001902243841

                                                    text               source  \
0     @HothfieldPlace All that pollution what "appar…   Twitter for iPhone
1     @PetenShirl Means a lower gear and mor polluti…      Twitter Web App
2     @toryboypierce @mailplus Londoners want ULEZ\n…   Twitter for iPhone
3     #LTN have reduced road space redundancy in the…  Twitter for Android
4     @YBcabbie @suemitch2017 @BBC @Keir_Starmer @Co…  Twitter for Android
…                                                  …                    …
7678  "@JackyBerland @BonGrosDodo This concerns the …      Twitter Web App
7679  Well, that's great, I must say. So, in additio…   Twitter for iPhone
7680  "@pascalCenteam @f_philippot I'm responding to…   Twitter for iPhone
7681  '@Bruno_Attal_ And pollution, what a big mess,…  Twitter for Android
7682  '@Fabien_Bagnon @brunobernard_fr @Gregorydouce…  Twitter for Android

      truncated  in_reply_to_status_id  in_reply_to_user_id  …  \
0         False           1.672699e+18         1.406968e+18  …
1         False           1.671498e+18         7.188028e+07  …
2         False           1.672705e+18         1.944467e+09  …
3         False                    NaN                  NaN  …
4         False           1.672692e+18         1.849338e+09  …
…           …                    …                    …   …
7678      False           1.670613e+18         1.466109e+18  …
7679      False                    NaN                  NaN  …
7680      False           1.670505e+18         8.114870e+08  …
7681      False           1.670438e+18         1.433049e+18  …
7682      False           1.670463e+18         8.805067e+17  …

     user_listed_count  user_favourites_count user_statuses_count  \
0                    0                    312                 704
1                    1                    134                3104
2                    0                   9438                4711
3                    3                  62642               54919
4                    4                  34524               28163
…                  …                    …                   …
7678                 8                   9420                6602
7679                 0                   1939                3378
7680                 1                  27016               54241
7681                 0                    182                 405
7682                 4                    550                1606

               user_created_at coordinates          place  is_quote_status  \
0     2022-12-20 15:52:12+00:00         NaN            NaN            False
1     2022-08-31 18:58:21+00:00         NaN            NaN            False
2     2022-04-07 15:05:13+00:00         NaN            NaN            False
3     2020-10-25 15:08:34+00:00         NaN            NaN             True
4     2020-03-13 11:47:00+00:00         NaN            NaN            False
…                           …         …            …              …
7678  2021-12-01 18:15:45+00:00         NaN            NaN            False
7679  2022-10-01 17:19:03+00:00         NaN  Paris, France             True
```

```
7680  2021-10-04 19:00:22+00:00        NaN            NaN             True
7681  2023-03-26 20:27:23+00:00        NaN            NaN            False
7682  2023-01-25 12:54:14+00:00        NaN            NaN            False

      favorite_count  lang                                      image_url
0                  0    en                                            NaN
1                  0    en                                            NaN
2                  2    en  https://pbs.twimg.com/tweet_video_thumb/FzalXs…
3                  3    en                                            NaN
4                  0    en    https://pbs.twimg.com/media/Fzaj9tpWIAEcwJa.jpg
…                ...   …                                              …
7678               1    fr                                            NaN
7679               3    fr                                            NaN
7680               0    fr                                            NaN
7681               1    fr                                            NaN
7682               4    fr                                            NaN

[4982 rows x 27 columns]
```

[3]:
```
df= data[['text']]
df
```

[3]:
```
                                                   text
0     @HothfieldPlace All that pollution what "appar…
1     @PetenShirl Means a lower gear and mor polluti…
2     @toryboypierce @mailplus Londoners want ULEZ\n…
3     #LTN have reduced road space redundancy in the…
4     @YBcabbie @suemitch2017 @BBC @Keir_Starmer @Co…
…                                                  …
7678  "@JackyBerland @BonGrosDodo This concerns the …
7679  Well, that's great, I must say. So, in additio…
7680  "@pascalCenteam @f_philippot I'm responding to…
7681  '@Bruno_Attal_ And pollution, what a big mess,…
7682  '@Fabien_Bagnon @brunobernard_fr @Gregorydouce…

[4982 rows x 1 columns]
```

0.1 Data Cleaning

- Removing Hashtags and username mentions
- Data Cleaning: We'll preprocess the tweet data to remove noise and irrelevant information, perform tokenization, and remove stop words and special characters.

[4]:
```
import re
import pandas as pd

def clean_tweet(tweet):
    # Remove hashtags
    tweet = re.sub(r'#\w+', '', tweet)
    # Remove words starting with '@'
    tweet = re.sub(r'@\w+', '', tweet)
    return tweet



# Apply clean_tweet function to the 'text' column in the DataFrame
df['cleaned_text'] = df['text'].apply(clean_tweet)

df
```

[4]:

```
                                                   text  \
0     @HothfieldPlace All that pollution what "appar…
1     @PetenShirl Means a lower gear and mor polluti…
2     @toryboypierce @mailplus Londoners want ULEZ\n…
3     #LTN have reduced road space redundancy in the…
4     @YBcabbie @suemitch2017 @BBC @Keir_Starmer @Co…
…                                                 …
7678  "@JackyBerland @BonGrosDodo This concerns the …
7679  Well, that's great, I must say. So, in additio…
7680  "@pascalCenteam @f_philippot I'm responding to…
7681  '@Bruno_Attal_ And pollution, what a big mess,…
7682  '@Fabien_Bagnon @brunobernard_fr @Gregorydouce…

                                           cleaned_text
0      All that pollution what "apparently" is a pri…
1      Means a lower gear and mor pollution.   Good …
2       Londoners want ULEZ\nWe are fed up with chil…
3      have reduced road space redundancy in the hea…
4          I watched the man who bought his own poll…
…                                                 …
7678  "  This concerns the decline in soil fertility…
7679  Well, that's great, I must say. So, in additio…
7680  "  I'm responding to react to all this mental …
7681  ' And pollution, what a big mess, all for a vi…
7682  '             Creating traffic jams \n  Adding…

[4982 rows x 2 columns]
```

[5]:

```
# Load the regular expression library
import re
# Remove punctuation
papers = pd.DataFrame()
papers['paper_text_processed'] = df['cleaned_text'].map(lambda x: re.sub('[,\.!?
 ↪]', '', x))
```

```
# Convert the titles to lowercase
papers['paper_text_processed'] = papers['paper_text_processed'].map(lambda x: x.
 ↪lower())
# Print out the first rows of papers
papers['paper_text_processed'].head()
```

[5]:
```
0     all that pollution what "apparently" is a pri…
1     means a lower gear and mor pollution   good d…
2      londoners want ulez\nwe are fed up with chil…
3     have reduced road space redundancy in the hea…
4         i watched the man who bought his own poll…
Name: paper_text_processed, dtype: object
```

[6]:
```
import gensim
from gensim.utils import simple_preprocess
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
stop_words.extend(['rT', 'im', 'rt', 'hes', 'Rt', 'ye', 'one', 'nm', 'shit',␣
 ↪'yeah', 'bb', 'https',
                   'tco', 'amp', 'sa', 'but', 'in', 'my', 'your','gt', 'water',␣
 ↪'waste','ur', 'youu', 'bb'," "])
def sent_to_words(sentences):
    for sentence in sentences:
        # deacc=True removes punctuations
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc))
             if word not in stop_words] for doc in texts]
data = papers.paper_text_processed.values.tolist()
data_words = list(sent_to_words(data))
# remove stop words
data_words = remove_stopwords(data_words)
print(data_words[:1][0][:30])
```

```
[nltk_data] Downloading package stopwords to /home/c4leb/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
['pollution', 'apparently', 'priority', 'southwark', 'council']
```

[7]:
```
import gensim.corpora as corpora
# Create Dictionary
id2word = corpora.Dictionary(data_words)
# Create Corpus
texts = data_words
# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
```

```
# View
print(corpus[:1][0][:30])
```

```
[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1)]
```

[8]:
```
from pprint import pprint
import gensim

# Number of topics
num_topics = 15

# Build LDA model with LdaModel
lda_model = gensim.models.LdaModel(corpus=corpus,
                                   id2word=id2word,
                                   num_topics=num_topics,
                                   alpha='auto',
                                   passes=20)

# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

```
[(0,
  '0.012*"pollution" + 0.010*"need" + 0.008*"fucking" + 0.008*"like" + '
  '0.007*"left" + 0.007*"bed" + 0.007*"person" + 0.006*"drank" + 0.005*"fish" '
  '+ 0.005*"hate"'),
 (1,
  '0.023*"money" + 0.020*"drinking" + 0.018*"time" + 0.010*"clean" + '
  '0.008*"power" + 0.007*"know" + 0.007*"talk" + 0.006*"electricity" + '
  '0.006*"dear" + 0.006*"much"'),
 (2,
  '0.012*"pollution" + 0.009*"else" + 0.007*"part" + 0.007*"clean" + '
  '0.007*"dust" + 0.007*"need" + 0.007*"system" + 0.006*"may" + 0.005*"work" + '
  '0.005*"healthcare"'),
 (3,
  '0.017*"pollution" + 0.011*"stop" + 0.009*"life" + 0.009*"people" + '
  '0.009*"time" + 0.009*"climate" + 0.009*"change" + 0.008*"let" + 0.007*"god" '
  '+ 0.006*"think"'),
 (4,
  '0.046*"pollution" + 0.010*"people" + 0.009*"plastic" + 0.008*"river" + '
  '0.008*"let" + 0.008*"want" + 0.007*"cars" + 0.006*"less" + 0.006*"also" + '
  '0.006*"air"'),
 (5,
  '0.025*"like" + 0.023*"pollution" + 0.022*"time" + 0.018*"people" + '
  '0.013*"air" + 0.011*"even" + 0.010*"much" + 0.009*"food" + 0.009*"good" + '
  '0.008*"day"'),
 (6,
  '0.009*"find" + 0.007*"people" + 0.007*"dont" + 0.007*"person" + '
  '0.007*"video" + 0.007*"hair" + 0.006*"point" + 0.006*"nothing" + '
  '0.006*"understand" + 0.005*"air"'),
 (7,
  '0.046*"drink" + 0.023*"day" + 0.019*"today" + 0.011*"lot" + 0.011*"eat" + '
  '0.010*"go" + 0.010*"forget" + 0.009*"good" + 0.009*"get" + 0.009*"back"'),
 (8,
  '0.012*"coffee" + 0.009*"states" + 0.009*"tea" + 0.008*"nuclear" + '
  '0.007*"ice" + 0.007*"black" + 0.006*"thousands" + 0.006*"war" + '
  '0.005*"cover" + 0.005*"indian"'),
 (9,
  '0.022*"time" + 0.010*"tap" + 0.008*"looking" + 0.008*"na" + 0.007*"dey" + '
  '0.007*"bills" + 0.006*"story" + 0.006*"high" + 0.006*"go" + 0.006*"let"'),
 (10,
  '0.014*"time" + 0.010*"block" + 0.008*"said" + 0.008*"glass" + 0.008*"half" '
  '+ 0.006*"fact" + 0.005*"radioactive" + 0.005*"arguing" + 0.005*"us" + '
  '0.005*"pollution"'),
 (11,
  '0.009*"women" + 0.009*"sea" + 0.008*"cannot" + 0.007*"made" + 0.006*"cup" + '
  '0.006*"us" + 0.006*"bright" + 0.005*"providing" + 0.005*"wait" + '
  '0.005*"seeing"'),
 (12,
  '0.015*"body" + 0.012*"cold" + 0.010*"got" + 0.010*"pot" + 0.008*"new" + '
  '0.008*"kind" + 0.007*"put" + 0.007*"signs" + 0.007*"time" + 0.007*"follow"'),
 (13,
  '0.012*"bottle" + 0.008*"region" + 0.008*"always" + 0.008*"children" + '
  '0.008*"already" + 0.008*"going" + 0.007*"best" + 0.007*"taking" + '
  '0.007*"according" + 0.006*"everything"'),
 (14,
  '0.022*"like" + 0.015*"get" + 0.012*"time" + 0.012*"would" + 0.010*"money" + '
  '0.008*"feel" + 0.008*"hot" + 0.007*"say" + 0.006*"much" + 0.006*"free"')]
```

[9]:
```python
# Print the topics in the desired format
for idx, topic in lda_model.print_topics(-1):
    topic_words = [word for word, _ in lda_model.show_topic(idx)]
    topic_words_str = ", ".join(topic_words)
    print(f"Topic {idx}: {topic_words_str}")
```

```
Topic 0: pollution, need, fucking, like, left, bed, person, drank, fish, hate
Topic 1: money, drinking, time, clean, power, know, talk, electricity, dear,
much
Topic 2: pollution, else, part, clean, dust, need, system, may, work, healthcare
Topic 3: pollution, stop, life, people, time, climate, change, let, god, think
Topic 4: pollution, people, plastic, river, let, want, cars, less, also, air
Topic 5: like, pollution, time, people, air, even, much, food, good, day
Topic 6: find, people, dont, person, video, hair, point, nothing, understand,
air
Topic 7: drink, day, today, lot, eat, go, forget, good, get, back
Topic 8: coffee, states, tea, nuclear, ice, black, thousands, war, cover, indian
Topic 9: time, tap, looking, na, dey, bills, story, high, go, let
Topic 10: time, block, said, glass, half, fact, radioactive, arguing, us,
pollution
Topic 11: women, sea, cannot, made, cup, us, bright, providing, wait, seeing
Topic 12: body, cold, got, pot, new, kind, put, signs, time, follow
Topic 13: bottle, region, always, children, already, going, best, taking,
according, everything
Topic 14: like, get, time, would, money, feel, hot, say, much, free
```

[10]:
```
# !pip install pyLDAvis
```

[11]:
```
import os
import pyLDAvis.gensim
import pickle
import pyLDAvis

# Visualize the topics
pyLDAvis.enable_notebook()
LDAvis_data_filepath = os.path.join('./results/ldavis_prepared_' +␣
 ↪str(num_topics))

# Create the 'results' directory if it doesn't exist
os.makedirs('./results/', exist_ok=True)

# Perform the visualization preparation and save the data
if 1 == 1:
    LDAvis_prepared = pyLDAvis.gensim.prepare(lda_model, corpus, id2word,␣
 ↪n_jobs=1)
    with open(LDAvis_data_filepath, 'wb') as f:
        pickle.dump(LDAvis_prepared, f)

# Load the pre-prepared pyLDAvis data from disk
with open(LDAvis_data_filepath, 'rb') as f:
    LDAvis_prepared = pickle.load(f)

# Save the pyLDAvis visualization as an HTML file
pyLDAvis.save_html(LDAvis_prepared, './results/ldavis_prepared_' +␣
 ↪str(num_topics) + '.html')

LDAvis_prepared
```

[11]:
```
PreparedData(topic_coordinates=              x         y  topics  cluster
Freq
topic
5      0.202586  0.023444       1        1  14.727999
7      0.104944  0.226564       2        1  10.344398
4      0.154940 -0.124788       3        1  10.106452
14      0.094788  0.055152       4        1   9.155590
3       0.104122 -0.119231       5        1   8.850198
2      -0.043164 -0.065266       6        1   5.725264
1      -0.024032 -0.019511       7        1   5.518443
0      -0.026948 -0.003504       8        1   5.409873
13     -0.062457  0.106636       9        1   5.299141
11     -0.038892 -0.020111      10        1   5.248532
6      -0.035042 -0.075964      11        1   4.910390
8      -0.133290  0.006213      12        1   3.949662
10     -0.066724  0.000469      13        1   3.711358
12     -0.135589  0.009100      14        1   3.569219
9      -0.095241  0.000796      15        1   3.473480, topic_info=
Term         Freq        Total Category  logprob  loglift
1538      drink  283.000000  283.000000  Default  30.0000  30.0000
474        time  497.000000  497.000000  Default  29.0000  29.0000
909         day  224.000000  224.000000  Default  28.0000  28.0000
640       money  170.000000  170.000000  Default  27.0000  27.0000
2     pollution  748.000000  748.000000  Default  26.0000  26.0000
…          …          …          …       …       …       …
345         let   11.357597  130.044569  Topic15  -5.1866   0.9220
169        well   10.997659  113.403676  Topic15  -5.2188   1.0267
329          go   11.553995  168.816362  Topic15  -5.1694   0.6782
202         say    9.205110   86.153316  Topic15  -5.3967   1.1236
847       going    7.577082  111.384821  Topic15  -5.5914   0.6722

[918 rows x 6 columns], token_table=      Topic      Freq        Term
term
5262      4  0.894562    absolute
1175      3  0.270851  absolutely
1175      8  0.631985  absolutely
1175     14  0.045142  absolutely
6840     14  0.916714         abt
…       …         …          …
2528      3  0.152148         yet
2528      4  0.786100         yet
2528      5  0.025358         yet
2528     13  0.025358         yet
212       9  0.878265        york

[1944 rows x 3 columns], R=30, lambda_step=0.01, plot_opts={'xlab': 'PC1',
'ylab': 'PC2'}, topic_order=[6, 8, 5, 15, 4, 3, 2, 1, 14, 12, 7, 9, 11, 13, 10])
```

[ ]:

```
```

[12]:

```
import pandas as pd
import re
import nltk
```

```python
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize


# Data cleaning function
def clean_tweet(tweet):
    # Remove URLs
    tweet = re.sub(r"http\S+|www\S+|https\S+", "", tweet, flags=re.MULTILINE)

    # Remove special characters and numbers
    tweet = re.sub(r"[^a-zA-Z\s]", "", tweet)

    # Convert to lowercase
    tweet = tweet.lower()

    # Tokenization
    words = word_tokenize(tweet)

    # Remove stopwords
    stop_words = set(stopwords.words("english"))
    words = [word for word in words if word not in stop_words]

    # Join words back to form the cleaned tweet
    cleaned_tweet = " ".join(words)
    return cleaned_tweet

# Apply data cleaning to the 'text' column in the DataFrame
df['cleaned_text2'] = df['cleaned_text'].apply(clean_tweet)
df
```

[12]:

```
                                                   text  \
0     @HothfieldPlace All that pollution what "appar…
1     @PetenShirl Means a lower gear and mor polluti…
2     @toryboypierce @mailplus Londoners want ULEZ\n…
3     #LTN have reduced road space redundancy in the…
4     @YBcabbie @suemitch2017 @BBC @Keir_Starmer @Co…
…                                                   …
7678  "@JackyBerland @BonGrosDodo This concerns the …
7679  Well, that's great, I must say. So, in additio…
7680  "@pascalCenteam @f_philippot I'm responding to…
7681  '@Bruno_Attal_ And pollution, what a big mess,…
7682  '@Fabien_Bagnon @brunobernard_fr @Gregorydouce…

                                           cleaned_text  \
0      All that pollution what "apparently" is a pri…
1      Means a lower gear and mor pollution.   Good …
2       Londoners want ULEZ\nWe are fed up with chil…
```

```
3      have reduced road space redundancy in the hea…
4          I watched the man who bought his own poll…
…                                                  …
7678  "  This concerns the decline in soil fertility…
7679  Well, that's great, I must say. So, in additio…
7680  "  I'm responding to react to all this mental …
7681  ' And pollution, what a big mess, all for a vi…
7682  '             Creating traffic jams \n  Adding…

                                          cleaned_text2
0       pollution apparently priority southwark council
1     means lower gear mor pollution good decision w…
2     londoners want ulez fed children dying chronic…
3     reduced road space redundancy heart capital me…
4     watched man bought pollution detector gov used…
…                                                  …
7678  concerns decline soil fertility decrease yield…
7679  well thats great must say addition car terrori…
7680  im responding react mental pollution raoult al…
7681                pollution big mess video outrageous
7682  creating traffic jams adding pollution priorit…

[4982 rows x 3 columns]
```

[13]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/c4leb/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

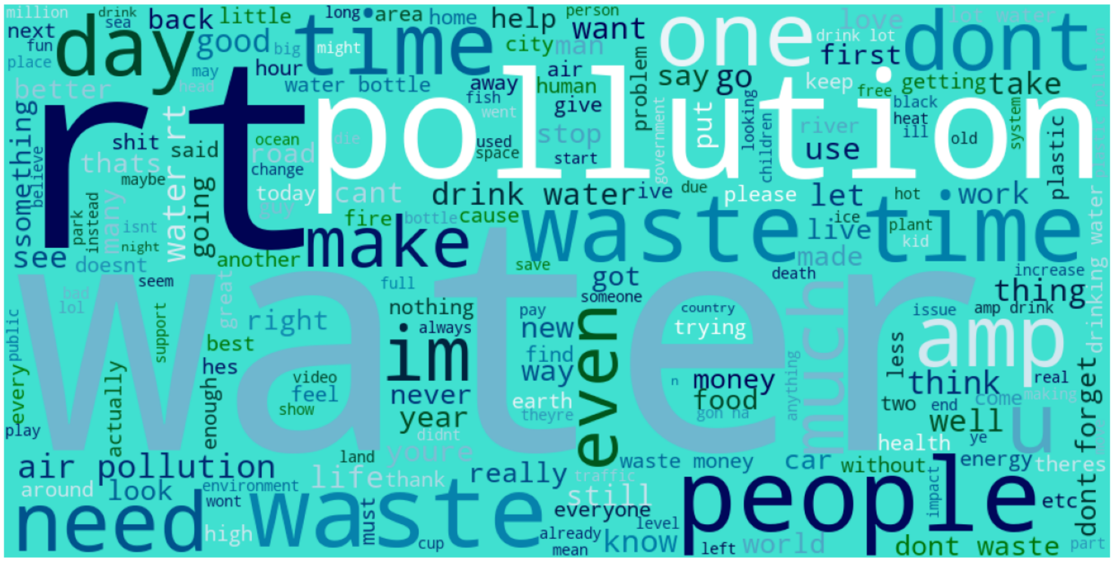
[13]:
```
True
```

[14]:
```
def text_data(df):
    df = df.cleaned_text2.values
    df = ','.join(str(x) for x in df)
    return df
def plot_cloud(wordcloud):
    plt.figure(figsize=(18, 8))
    stop_words = ['rT', 'im', 'rt', 'hes', 'Rt', 'ye', 'one', 'nm', 'shit',␣
 ↪'yeah', 'bb', 'https',
                'a', 'an', 'the', 'and', 'it', 'for', 'or', 'but', 'in', 'my',␣
 ↪'your',
                 'our', 'and' 'their', 'ur', 'youu', 'bb'," "]
    plt.imshow(wordcloud)
    plt.axis("off");

wordcloud = WordCloud(width = 1000, height = 500, background_color='#40E0D0',
                        colormap="ocean",  random_state=10).
 ↪generate(text_data(df))
plot_cloud(wordcloud)
```

```
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
```

```
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:499:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = Image.ROTATE_90
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
/home/c4leb/anaconda3/lib/python3.9/site-packages/wordcloud/wordcloud.py:519:
DeprecationWarning: ROTATE_90 is deprecated and will be removed in Pillow 10
(2023-07-01). Use Transpose.ROTATE_90 instead.
  orientation = (Image.ROTATE_90 if orientation is None else
```



```
[15]: df
```

```
[15]:                                                    text  \
      0     @HothfieldPlace All that pollution what "appar…
      1     @PetenShirl Means a lower gear and mor polluti…
      2     @toryboypierce @mailplus Londoners want ULEZ\n…
      3     #LTN have reduced road space redundancy in the…
      4     @YBcabbie @suemitch2017 @BBC @Keir_Starmer @Co…
      …                                                   …
      7678  "@JackyBerland @BonGrosDodo This concerns the …
      7679  Well, that's great, I must say. So, in additio…
      7680  "@pascalCenteam @f_philippot I'm responding to…
      7681  '@Bruno_Attal_ And pollution, what a big mess,…
      7682  '@Fabien_Bagnon @brunobernard_fr @Gregorydouce…
```

```
                                              cleaned_text  \
0      All that pollution what "apparently" is a pri…
1      Means a lower gear and mor pollution.   Good …
2       Londoners want ULEZ\nWe are fed up with chil…
3      have reduced road space redundancy in the hea…
4          I watched the man who bought his own poll…
…                                                 …
7678  "  This concerns the decline in soil fertility…
7679  Well, that's great, I must say. So, in additio…
7680  "  I'm responding to react to all this mental …
7681  ' And pollution, what a big mess, all for a vi…
7682  '             Creating traffic jams \n  Adding…

                                             cleaned_text2
0       pollution apparently priority southwark council
1     means lower gear mor pollution good decision w…
2     londoners want ulez fed children dying chronic…
3     reduced road space redundancy heart capital me…
4     watched man bought pollution detector gov used…
…                                                 …
7678  concerns decline soil fertility decrease yield…
7679  well thats great must say addition car terrori…
7680  im responding react mental pollution raoult al…
7681                 pollution big mess video outrageous
7682  creating traffic jams adding pollution priorit…

[4982 rows x 3 columns]
```

[16]:
```python
import gensim
from gensim import corpora
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

# Assuming 'df' is the DataFrame with a column named 'text' containing the␣
 ↪preprocessed tweets
tweets = df["cleaned_text2"].tolist()

# Remove outliers (you can inspect your entire dataset and remove irrelevant␣
 ↪tweets)

# Tokenize the tweets (assuming tweets are already preprocessed and␣
 ↪space-separated)
tweets_tokenized = [tweet.split() for tweet in tweets]

# Remove stopwords
nltk.download('stopwords')
```

```
stop_words = set(stopwords.words('english'))
tweets_filtered = [[word for word in tweet if word not in stop_words] for tweet␣
 ↪in tweets_tokenized]

# Porter stemming
stemmer = PorterStemmer()
tweets_stemmed = [[stemmer.stem(word) for word in tweet] for tweet in␣
 ↪tweets_filtered]

# Handle common typos and abbreviations (customize as needed)
# Example: Replace 'u' with 'you', 'rt' with 'retweet', etc.
custom_replacements = {
    'u': 'you',
    'rt': 'retweet',
    'im': 'I am',
    'k': 'ok',
    # Add more replacements as needed
}
tweets_corrected = [[custom_replacements.get(word, word) for word in tweet] for␣
 ↪tweet in tweets_stemmed]

# Create a dictionary and a corpus (word count representation) for LDA
dictionary = corpora.Dictionary(tweets_corrected)
corpus = [dictionary.doc2bow(tweet) for tweet in tweets_corrected]

# Train the LDA model
num_topics_lda = 10  # Set the number of topics for LDA
lda_model = gensim.models.LdaModel(corpus, num_topics=num_topics_lda,␣
 ↪id2word=dictionary, passes=10)

# Print the topics in the desired format
for idx, topic in lda_model.print_topics(-1):
    topic_words = [word for word, _ in lda_model.show_topic(idx)]
    topic_words_str = ", ".join(topic_words)
    print(f"Topic {idx}: {topic_words_str}")
```

```
[nltk_data] Downloading package stopwords to /home/c4leb/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
Topic 0: water, retweet, day, drank, I am, train, night, river, coffe, bridg
Topic 1: water, drink, dont, day, retweet, eat, lot, today, amp, forget
Topic 2: water, retweet, plant, amp, bomb, pure, nuclear, worri, sea, go
Topic 3: wast, money, time, fuck, month, water, retweet, product, link, type
Topic 4: pollut, water, need, retweet, amp, wast, one, take, us, air
Topic 5: water, retweet, pollut, time, wast, peopl, dont, park, want, even
Topic 6: water, retweet, morn, earth, you, pollut, dust, use, fire, good
Topic 7: pollut, air, peopl, wast, year, retweet, caus, state, water, amp
Topic 8: water, retweet, like, bottl, na, drink, get, one, love, look
Topic 9: water, wast, time, retweet, dont, go, like, get, I am, know
```

---

1. Non-negative Matrix Factorization (NMF)

[17]:
```python
import gensim
from gensim import corpora
# Handle common typos and abbreviations (customize as needed)
# Example: Replace 'u' with 'you', 'rt' with 'retweet', etc.
custom_replacements = {
    'u': 'you',
    'rt': 'retweet',
    'im': 'I am',
    'ur': 'your',
    'k': 'ok',
    'g':'',
    'f':'',
    # Add more replacements as needed
}
tweets_corrected = [[custom_replacements.get(word, word) for word in tweet] for␣
 ↪tweet in tweets_stemmed]

# Assuming 'df' is the DataFrame with a column named 'cleaned_text2' containing␣
 ↪the preprocessed tweets
tweets = tweets_corrected

# Tokenize the tweets (assuming tweets are already preprocessed and␣
 ↪space-separated)
# tweets_tokenized = [tweet.split() for tweet in tweets]


# Create a dictionary and a corpus (word count representation) for LDA
dictionary = corpora.Dictionary(tweets_corrected)
corpus = [dictionary.doc2bow(tweet) for tweet in tweets_tokenized]

# Train the LDA model
num_topics_lda = 10  # Set the number of topics for LDA
lda_model = gensim.models.LdaModel(corpus, num_topics=num_topics_lda,␣
 ↪id2word=dictionary, passes=10)

# Print the topics in the desired format
for idx, topic in lda_model.print_topics(-1):
    topic_words = [word for word, _ in lda_model.show_topic(idx)]
    topic_words_str = ", ".join(topic_words)
    print(f"Topic {idx}: {topic_words_str}")
```

```
Topic 0: women, water, need, start, want, cost, environment, half, fight, dr
Topic 1: money, water, light, time, car, good, park, new, total, tax
Topic 2: water, would, need, may, earth, nuclear, thank, block, made, seen
Topic 3: water, amp, children, power, fish, high, region, clean, c, hold
Topic 4: water, air, plastic, amp, make, health, clean, free, river, work
Topic 5: like, well, time, money, us, let, space, one, area, even
Topic 6: life, water, time, better, dont, like, even, left, men, need
Topic 7: time, state, man, less, pay, water, dust, reason, china, yet
Topic 8: water, like, drink, get, hot, take, use, one, na, cold
Topic 9: water, dont, time, day, drink, like, go, would, today, good
```

---

2. Hierarchical Dirichlet Process (HDP)

[18]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import NMF

# Assuming 'data' is a DataFrame with a column named 'text' containing the␣
↪preprocessed tweets
tweets = df["cleaned_text2"].tolist()

# Convert the preprocessed tweets to a TF-IDF matrix
vectorizer = TfidfVectorizer(max_features=1000)  # Set the max_features to␣
↪control the number of features
tfidf_matrix = vectorizer.fit_transform(tweets)

# Train the NMF model
num_topics_nmf = 10  # Set the number of topics for NMF
nmf_model = NMF(n_components=num_topics_nmf, random_state=42)
nmf_model.fit(tfidf_matrix)

# Print the top words for each topic
for topic_idx, topic in enumerate(nmf_model.components_):
    top_words = [vectorizer.get_feature_names()[i] for i in topic.argsort()[:
↪-10 - 1:-1]]
    print(f"Topic {topic_idx}: {', '.join(top_words)}")
```

```
Topic 0: water, bottle, drinking, cold, hot, love, tap, put, bomb, pure
Topic 1: time, waste, dont, life, well, someone, precious, got, youre, spend
Topic 2: rt, bomb, bottle, new, life, region, first, splashing, children, today
Topic 3: pollution, air, noise, plastic, light, traffic, climate, world, levels,
reduce
Topic 4: dont, ur, forget, today, lot, day, beautiful, wish, cheer, bfast
Topic 5: like, look, looks, feel, sound, well, seems, actually, always, running
Topic 6: waste, money, space, energy, taxpayer, let, total, years, fucking, go
Topic 7: drink, eat, day, water, warm, good, take, forget, plenty, happy
Topic 8: people, get, im, one, go, need, would, think, even, know
Topic 9: amp, wish, day, bright, warm, forget, food, drop, follow, going
```

---

3. Latent Semantic Analysis (LSA)

[19]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import TruncatedSVD

# Assuming 'data' is a DataFrame with a column named 'text' containing the␣
 ↪preprocessed tweets
tweets = df["cleaned_text2"].tolist()

# Convert the preprocessed tweets to a TF-IDF matrix
vectorizer_lsa = TfidfVectorizer(max_features=1000)  # Set the max_features to␣
 ↪control the number of features
tfidf_matrix_lsa = vectorizer_lsa.fit_transform(tweets)

# Train the LSA model
num_topics_lsa = 10  # Set the number of topics for LSA
lsa_model = TruncatedSVD(n_components=num_topics_lsa, random_state=42)
lsa_topic_matrix = lsa_model.fit_transform(tfidf_matrix_lsa)

# Print the top words for each topic
for topic_idx, topic in enumerate(lsa_model.components_):
    top_words = [vectorizer_lsa.get_feature_names()[i] for i in topic.
 ↪argsort()[:-10 - 1:-1]]
    print(f"Topic {topic_idx}: {', '.join(top_words)}")
```

```
Topic 0: water, rt, drink, waste, like, dont, time, amp, drinking, get
Topic 1: waste, time, rt, dont, money, pollution, people, like, life, want
Topic 2: rt, bomb, region, pollution, new, children, bottle, first, according,
splashing
Topic 3: pollution, air, amp, dont, drink, day, people, forget, like, wish
Topic 4: drink, dont, forget, day, amp, wish, today, ur, cheer, lot
Topic 5: like, im, get, dont, people, one, would, know, think, drinking
Topic 6: like, money, waste, drink, amp, day, one, wish, beautiful, cheer
Topic 7: drink, time, eat, good, take, pollution, plenty, sleep, hope, morning
Topic 8: time, like, amp, pollution, beautiful, air, today, cheer, bfast, lot
Topic 9: dont, like, pollution, waste, ur, skip, bb, gratefull, youu, sweetie
```

---

4. Latent Dirichlet Allocation (LDA)

[20]:
```
# import gensim
# from gensim import corpora

# # Assuming 'data' is a DataFrame with a column named 'text' containing the␣
 ↪preprocessed tweets
# tweets = data["text"].tolist()
# # Tokenize the tweets (assuming tweets are already preprocessed and␣
↪space-separated)
# tweets_tokenized = [tweet.split() for tweet in tweets]

# # Create a dictionary and a corpus (word count representation) for LDA
# dictionary = corpora.Dictionary(tweets_tokenized)
# corpus = [dictionary.doc2bow(tweet) for tweet in tweets_tokenized]

# # Train the LDA model
# num_topics_lda = 10  # Set the number of topics for LDA
# lda_model = gensim.models.LdaModel(corpus, num_topics=num_topics_lda,␣
↪id2word=dictionary, passes=10)

# # Print the topics in the desired format
# for idx, topic in lda_model.print_topics(-1):
#     topic_words = [word for word, _ in lda_model.show_topic(idx)]
#     topic_words_str = ", ".join(topic_words)
#     print(f"Topic {idx}: {topic_words_str}")
```

[ ]: