fox_segments

March 21, 2022

[5]:
```python
filename = "data/Analysis_123021_Colorado_Fire_Fox.docx"
```

[6]:
```python
import sys
sys.path.append('../')

from helpers.utils import read_docx_to_dict
```

[7]:
```python
data = read_docx_to_dict(filename)
```

[8]:
```python
import pandas as pd
pd.options.display.max_rows = 500

# create dataframe
df = pd.DataFrame.from_dict(data)
```

[11]:
```python
import sys
sys.path.append('../')

from helpers.utils import check_text_likeness
```

[12]:
```python
df['matches'] = df.apply(lambda row: check_text_likeness(df, row['text']),␣
  ↪axis=1)
```

[22]:
```python
from helpers.utils import fetch_biggest_text, mark_use_row
```

[23]:
```python
df['row_to_use'] = df.apply(lambda row: fetch_biggest_text(df, row['matches']),␣
  ↪axis=1)
```

[24]:
```python
mark_use_row(df)
```

[24]:
```
'done'
```

[26]:
```python
df['words'] = df['text'].str.lower().str.replace(',', '').str.replace('>', '').
  ↪str.replace('.', '').str.replace('\n', '').str.replace('’', "'").str.replace(
    '!', '').str.replace('?', '').str.replace('%', '').str.replace(')', '').str.
  ↪replace('(', '').str.replace('_', '').str.replace(':', '').str.strip().str.
  ↪split(' ')
```

```
/Users/loren/.pyenv/versions/3.7.4/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will
change from True to False in a future version. In addition, single character
regular expressions will *not* be treated as literal strings when regex=True.
  """Entry point for launching an IPython kernel.
/Users/loren/.pyenv/versions/3.7.4/lib/python3.7/site-
packages/ipykernel_launcher.py:2: FutureWarning: The default value of regex will
change from True to False in a future version. In addition, single character
regular expressions will *not* be treated as literal strings when regex=True.
```

[28]:
```
import sys
sys.path.append('../')
```

[29]:
```
from helpers.utils import parse_words
df['clean_words'] = df.apply(lambda row: parse_words(row['words']), axis=1)
```

[63]:
```
class Color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'

def highlight_word(word, text):
    highlighted_word = Color.BOLD + Color.RED + Color.UNDERLINE + word + Color.
 ↪END

    if word in text:
        return text.replace(word, highlighted_word)
    else:
        return ''

def highlight_words_found(climate_words, text):
    if not climate_words:
        return ''


    return [highlight_word(word, text) for word in climate_words][0]
```

[64]:
```
from helpers.utils import fetch_climate_words_in_words,␣
 ↪fetch_climate_phrases_in_text
```

```
df['climate_phrases_found'] = df.apply(lambda row:␣
 ↪fetch_climate_phrases_in_text(row['text']), axis=1)
df['climate_words_found'] = df.apply(lambda row:␣
 ↪fetch_climate_words_in_words(row['clean_words']), axis=1)

df['highlighted_words'] = df.apply(lambda row:␣
 ↪highlight_words_found(row['climate_words_found'], row['text']), axis=1)
df['highlighted_phrases'] = df.apply(lambda row:␣
 ↪highlight_words_found(row['climate_phrases_found'], row['text']), axis=1)
```

[65]:
```
# save data to csv
df.to_csv('reports/fox_all.csv', encoding='utf-8')
df.to_excel('reports/fox_all.xlsx', engine='xlsxwriter', encoding='utf-8')

# https://stackoverflow.com/questions/50495463/
 ↪unable-to-change-font-color-in-excel-using-python-xlsxwriter
```

[70]:
```
unique_df = df[df['use_row']]
```

[72]:
```
total_words = unique_df['clean_words'].str.len().sum()
total_words
```

[72]:
```
16506
```

[73]:
```
from helpers.utils import words_found_master_list

words_found = words_found_master_list(unique_df['clean_words'])
len(words_found)
```

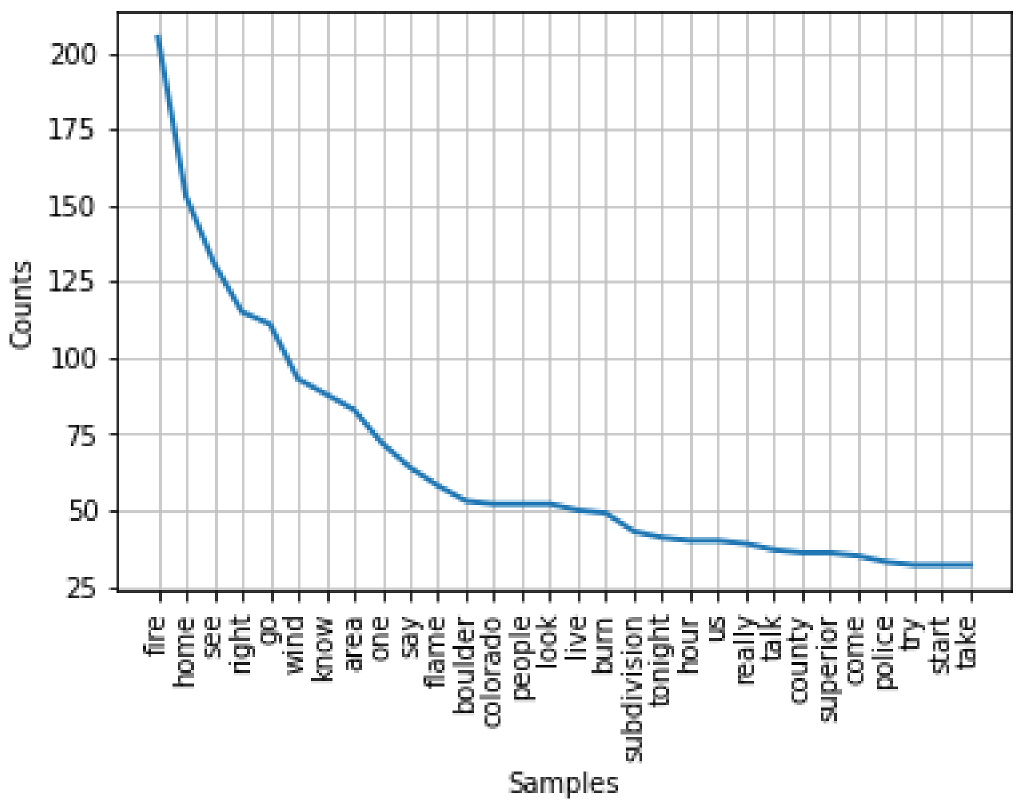
[73]:
```
16506
```

[75]:
```
from helpers.utils import clean_lemmatized_words, lemmatize_words
clean_lemma_words = clean_lemmatized_words(lemmatize_words(words_found))
```

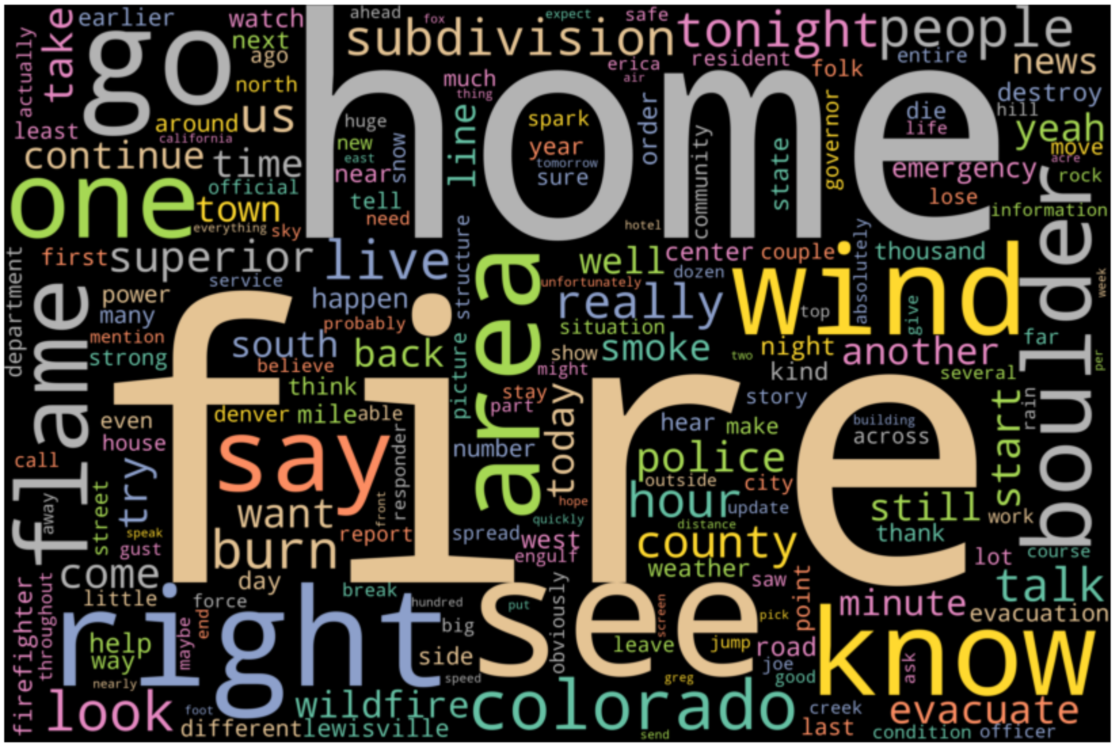
[76]:
```
from nltk.probability import FreqDist

lfdist = FreqDist(clean_lemma_words)
lfdist
```

[76]:
```
FreqDist({'fire': 205, 'home': 153, 'see': 131, 'right': 115, 'go': 111, 'wind':
93, 'know': 88, 'area': 83, 'one': 72, 'say': 64, …})
```

[77]:
```
import matplotlib.pyplot as plt
lfdist.plot(30,cumulative=False)
plt.show()
```



[79]:
```
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
from wordcloud import STOPWORDS
import matplotlib.pyplot as plt

from helpers.utils import master_stopwords_list

wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,␣
 ↪background_color='black', colormap='Set2', collocations=False, stopwords =␣
 ↪master_stopwords_list()).generate_from_frequencies(lfdist)

# Plot
plt.figure( figsize=(15,10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

#plt.savefig('word_cloud.png')
```



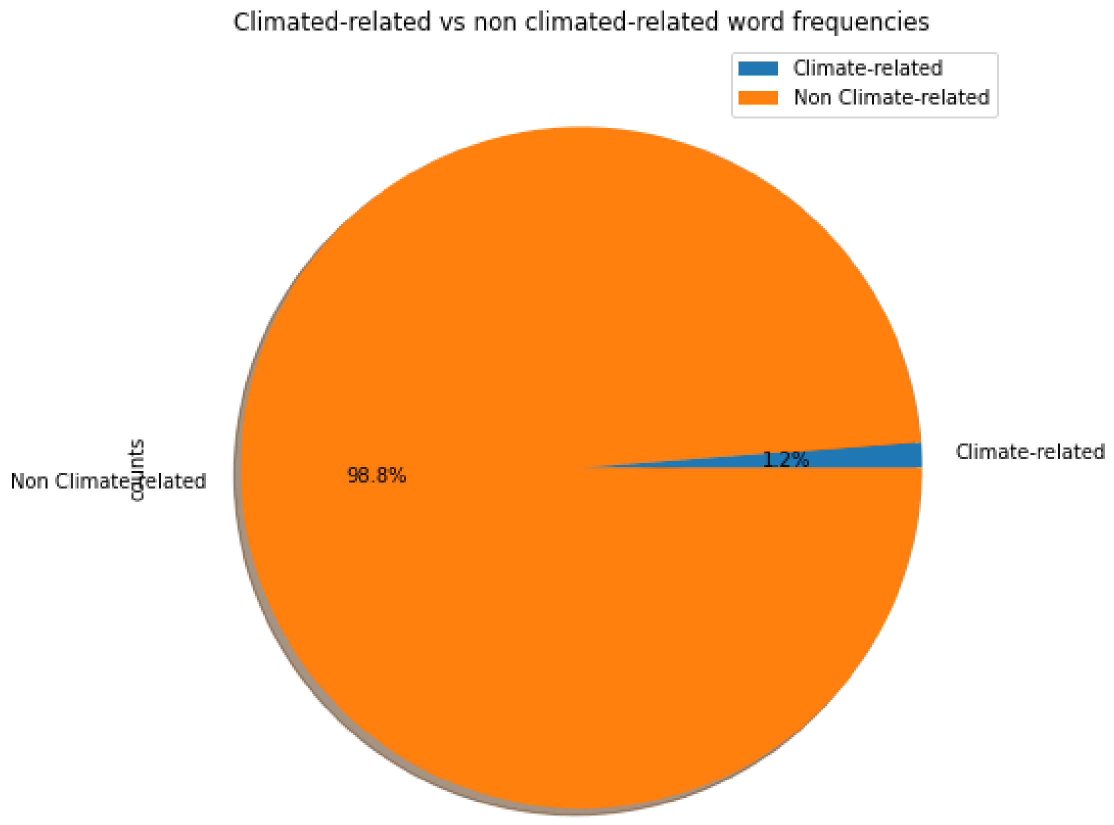
[80]:
```
import pandas as pd
pd.options.display.max_rows = 500
words_df = pd.DataFrame(lfdist.items(), columns=['Word', 'Count'])

words_df.sort_values(by=['Count'], ascending=False, inplace=True)
len(words_df)
# 1374 total words

words_df['Count'].sum()
```

[80]:
```
7538
```

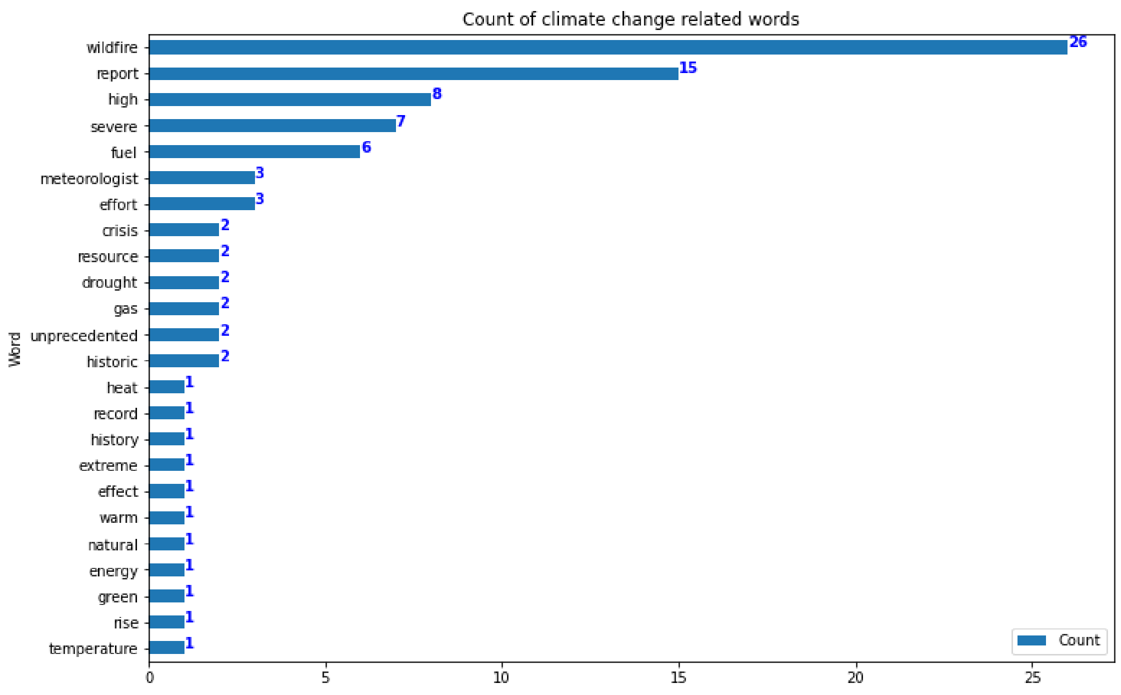
[81]:
```
import sys
sys.path.append('../')
from helpers.words import CLIMATE_CHANGE_RELATED_WORDS,␣
  ↪CLIMATE_CHANGE_RELATED_PHRASES

# create data
climate_change_words_df = words_df.loc[words_df['Word'].
  ↪isin(CLIMATE_CHANGE_RELATED_WORDS)]

climate_words_count = climate_change_words_df['Count'].sum()
non_climate_words_count = words_df['Count'].sum() - climate_words_count
```

```
comparison_df = pd.DataFrame({'Words': ['Climate-related', 'Non␣
 ↪Climate-related'],
                              'counts': [climate_words_count,␣
 ↪non_climate_words_count]})
comparison_df.set_index('Words', inplace=True)
print(comparison_df)

plot = comparison_df.plot.pie(y='counts', title="Climated-related vs non␣
 ↪climated-related word frequencies", legend=True, autopct='%1.1f%%',␣
 ↪shadow=True, figsize=(8, 8))

fig = plot.get_figure()
#fig.savefig("comparison.png")
```

```
                     counts
Words
Climate-related          91
Non Climate-related    7447
```



[85]:
```
# find climate related word frequencies

# set figure size
fig, ax = plt.subplots(figsize=(12, 8))
# plot horizontal bar plot
climate_change_words_df.sort_values(by='Count').plot.barh(x="Word", y="Count",␣
↪ax=ax)
# set the title
plt.title("Count of climate change related words")

for i, v in enumerate(climate_change_words_df['Count'].sort_values()):
    ax.text(v, i , str(v),
            color = 'blue', fontweight = 'bold')

plt.show()
# plt.savefig('climate-related-words-breakdown.png', transparent=False)
```



[86]:
```
# find segments
climate_change_words_found = list(climate_change_words_df['Word'].unique())
climate_change_words_found
```

[86]:
```
['wildfire',
 'report',
 'high',
 'severe',
 'fuel',
 'meteorologist',
 'effort',
 'resource',
 'drought',
 'gas',
 'crisis',
 'unprecedented',
 'historic',
 'record',
 'history',
 'extreme',
 'effect',
 'warm',
 'natural',
 'energy',
 'green',
 'rise',
 'heat',
 'temperature']
```

[87]:
```
unique_df[unique_df["climate_words_found"].str.len() != 0].to_csv('reports/
 ↪abc_final.csv', encoding='utf-8')
```

[88]:
```
# total segments
total_segments = len(df)
total_segments
```

[88]:
```
98
```

[89]:
```
# unique segments
unique_segments = len(df[df['use_row'] == True])
unique_segments
```

[89]:
```
96
```

[90]:
```
# how many segments had climate related words/phrases - %
possible_climate_related_segments = len(df[(df["climate_words_found"].str.len()␣
 ↪!= 0) & (df["use_row"] == True)])
possible_climate_related_segments


f'{possible_climate_related_segments / unique_segments * 100.0} %'
```

[90]:
```
'42.708333333333 %'
```