news_headline_sentiment_analysis

June 19, 2023

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
↪docker-python
!pip install NRCLex
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting NRCLex
  Downloading NRCLex-4.0-py3-none-any.whl (4.4 kB)
Requirement already satisfied: textblob in /usr/local/lib/python3.10/dist-
packages (from NRCLex) (0.17.1)
INFO: pip is looking at multiple versions of nrclex to determine which version
is compatible with other requirements. This could take a while.
  Downloading NRCLex-3.0.0.tar.gz (396 kB)
                           396.4/396.4 kB
12.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.10/dist-
packages (from textblob->NRCLex) (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob->NRCLex) (8.1.3)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob->NRCLex) (1.2.0)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk>=3.1->textblob->NRCLex)
(2022.10.31)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk>=3.1->textblob->NRCLex) (4.65.0)
Building wheels for collected packages: NRCLex
  Building wheel for NRCLex (setup.py) … done
  Created wheel for NRCLex: filename=NRCLex-3.0.0-py3-none-any.whl size=43308
sha256=1b60ce5214523181e9f9085b2004af1a11e96c15d32e368d6c179e091f268dac
  Stored in directory: /root/.cache/pip/wheels/d2/10/44/6abfb1234298806a145fd6bc
aec8cbc712e88dd1cd6cb242fa
Successfully built NRCLex
Installing collected packages: NRCLex
Successfully installed NRCLex-3.0.0
```

[2]:
```
#load all required packages
#general data handling and processing
import pandas as pd
import numpy as np
#text data processing and sentiment analysis tools
import nltk
nltk.download('vader_lexicon')
nltk.download('punkt')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sia = SentimentIntensityAnalyzer()
from nrclex import NRCLex

#visualization
import matplotlib.pyplot as plt

from google.colab import files
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data…
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[3]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[4]:
```
#import data
abc = pd.read_csv('drive/MyDrive/data.csv')
```

[5]:
```
#take a look at the structure and basic information of this dataset
abc.info()
#identify missing values from the dataset
abc.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1244184 entries, 0 to 1244183
Data columns (total 2 columns):
 #   Column         Non-Null Count    Dtype
---  ------         --------------    -----
 0   publish_date   1244184 non-null  int64
 1   headline_text  1244184 non-null  object
dtypes: int64(1), object(1)
memory usage: 19.0+ MB
```

[5]:
```
publish_date     0
headline_text    0
dtype: int64
```

```
[6]: #change publish_date column to datetime as it is currently integer
     abc['publish_date'] = pd.to_datetime(abc['publish_date'], format='%Y%m%d')
```

0.0.1 NLTK VADER

Firstly, I'using NLTK VADER package to perform a sentiment analysis and plot it to see the change over the years. According to Schumacher (2019), "VADER, has different ratings depending on the form of the word and therefore the input should not be stemmed or lemmatized." I decided not to stem or lemmatize the words for this sentiment analysis.

I will be using thresholds values of -0.05 and 0.05, based on "About the Scoring" section on this github page. If compound score is larger than 0.05, the headline will be classified as positive; if compound score is smaller than -0.05 it will be negative.

```
[7]: #create polarity scores
     abc['senti_score'] = abc['headline_text'].apply(lambda headline: sia.
      ↪polarity_scores(headline))

     #extract compound scores to a new column
     abc['compound']  = abc['senti_score'].apply(lambda score_dict:␣
      ↪score_dict['compound'])

     #create a new column for sentiment labels
     abc['senti_label'] = abc['compound'].apply(lambda c: 'positive' if c >=0.05␣
      ↪else 'neutral' if c>-0.05 else 'negative')

     #counts of sentiment labels
     abc['senti_label'].value_counts()
```

```
[7]: neutral     564887
     negative    426479
     positive    252818
     Name: senti_label, dtype: int64
```

The above result shows that nearly half of the news headlines are neutral. While in the other half, there are more negative headlines identified than positive ones.

Visualizations of sentiments over the years

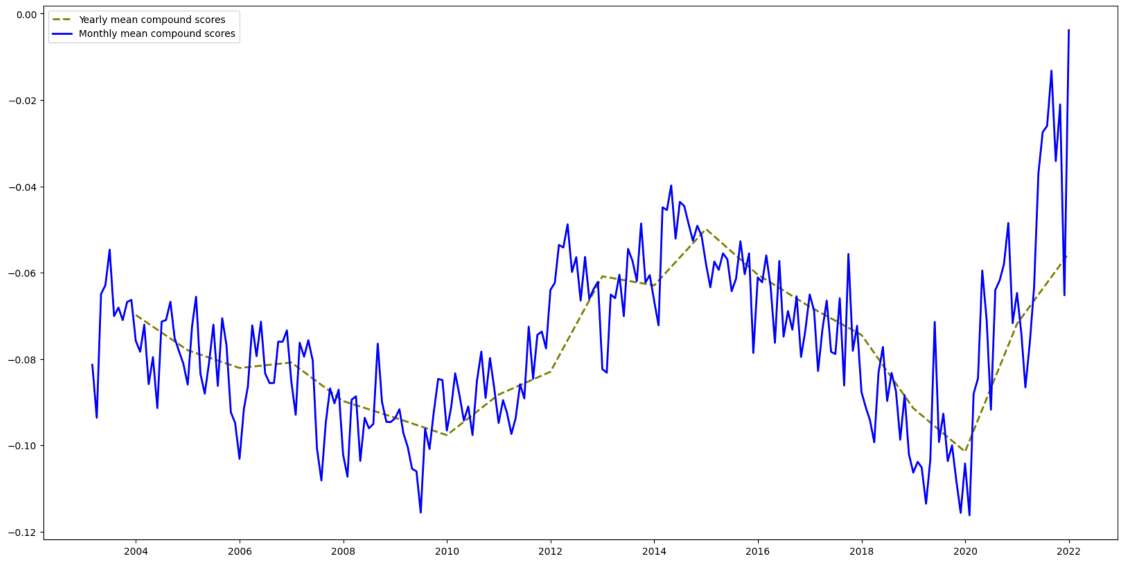
```
[8]: #calculate the average compound scores per month and per year respectively
     yearly_averages = abc.resample('A',on='publish_date').mean()
     monthly_averages = abc.resample('M',on='publish_date').mean()
```

```
<ipython-input-8-4176f8c4a919>:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  yearly_averages = abc.resample('A',on='publish_date').mean()
<ipython-input-8-4176f8c4a919>:3: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  monthly_averages = abc.resample('M',on='publish_date').mean()
```

[9]:
```python
monthly_averages.head(5)
```

[9]:
```
                 compound
publish_date
2003-02-28   -0.081339
2003-03-31   -0.093607
2003-04-30   -0.064970
2003-05-31   -0.062888
2003-06-30   -0.054663
```

[10]:
```python
#visualization of vader sentiment scores
plt.figure(figsize=(20,10))
plt.plot(yearly_averages.index,yearly_averages['compound'],  color='olive',␣
 ↪linewidth=2, linestyle='dashed', label='Yearly mean compound scores')
plt.plot(monthly_averages.index, monthly_averages['compound'], color = 'blue',␣
 ↪linewidth=2, label='Monthly mean compound scores')
plt.legend()
plt.show()
```



This graph indicates that there might be a cycle in the sentiment of news headlines as there are two peaks-around 2003 and 2015, and two troughs - in 2010 and end of 2019. It’s worth looking into the topics in those years to understand the peaks and troughs in sentiment.There could be major events happening around those times which had made the media change their sentiment.

0.0.2 NRC Emotion Lexicon

Next, I will be using NRCLex package to further detect emotions in news headlines.

[11]:
```
#function to retrieve nrc affect frequencies
def emotion_freq(headline):
    res1 = {'anger': 0.0, 'fear': 0.0, 'negative': 0.0, 'positive': 0.0,␣
 ↪'sadness': 0.0, 'trust': 0.0, 'anticipation': 0.0, 'joy': 0.0, 'disgust': 0.
 ↪0, 'surprise': 0.0}
    headline = NRCLex(headline)
    freq = headline.affect_frequencies
    for k, fq in freq.items():
      res1[k] = res1.get(k, 0.0) + fq
    return res1

#function to calculate word count in each headline
def word_count(row):
    row = nltk.word_tokenize(row)
    cnt = len(row)
    return cnt
```

[12]:
```
#create a new dataset without vader analysis
abc_nrc = abc.iloc[:,0:2].copy()
```

[13]:
```
#retrieve affect frequencies in each headline
abc_nrc['emo_freq']=abc_nrc['headline_text'].apply(emotion_freq)
```

[14]:
```
#take a look at our new column with affected frequencies
abc_nrc.head()
```

[14]:
```
  publish_date                                     headline_text  \
0   2003-02-19  aba decides against community broadcasting lic…
1   2003-02-19     act fire witnesses must be aware of defamation
2   2003-02-19     a g calls for infrastructure protection summit
3   2003-02-19           air nz staff in aust strike for pay rise
4   2003-02-19      air nz strike to affect australian travellers

                                            emo_freq
0  {'anger': 0.0, 'fear': 0.0, 'negative': 0.0, '…
1  {'anger': 0.0, 'fear': 0.5, 'negative': 0.25, …
2  {'anger': 0.0, 'fear': 0.0, 'negative': 0.3333…
3  {'anger': 0.16666666666666666, 'fear': 0.0, 'n…
4  {'anger': 0.5, 'fear': 0.0, 'negative': 0.5, '…
```

[15]:
```
#extract out the emotions to new columns for further analysis
abc_nrc = pd.concat((abc_nrc.drop(['emo_freq'],axis=1), abc_nrc['emo_freq'].
 ↪apply(pd.Series)), axis=1)
```

[16]:
```
#calculate word count in each headline
abc_nrc['word_count']=abc_nrc['headline_text'].apply(word_count)
```

[17]:
```
abc_nrc.head()
```

[17]:
```
  publish_date                                      headline_text     anger  \
0   2003-02-19  aba decides against community broadcasting lic…  0.000000
1   2003-02-19     act fire witnesses must be aware of defamation  0.000000
2   2003-02-19     a g calls for infrastructure protection summit  0.000000
3   2003-02-19           air nz staff in aust strike for pay rise  0.166667
4   2003-02-19      air nz strike to affect australian travellers  0.500000

   fear  negative  positive  sadness     trust  anticipation       joy  \
0   0.0  0.000000  1.000000      0.0  0.000000      0.000000  0.000000
1   0.5  0.250000  0.000000      0.0  0.000000      0.000000  0.000000
2   0.0  0.333333  0.000000      0.0  0.333333      0.333333  0.000000
3   0.0  0.166667  0.166667      0.0  0.166667      0.166667  0.166667
4   0.0  0.500000  0.000000      0.0  0.000000      0.000000  0.000000

   disgust  surprise  anticip  word_count
0     0.00       0.0      0.0           6
1     0.25       0.0      0.0           8
2     0.00       0.0      0.0           7
3     0.00       0.0      0.0           9
4     0.00       0.0      0.0           7
```

[18]:
```
#normalize emotion frequencies by having it divided by word counts in each␣
 ↪headline
emotions =␣
 ↪['anger','fear','negative','positive','sadness','trust','anticipation','joy','disgust','sur
for emotion in emotions:
    abc_nrc[emotion] = abc_nrc[emotion]/abc_nrc['word_count']
```

[19]:
```
#now we have our dataframe as below
abc_nrc.head()
```

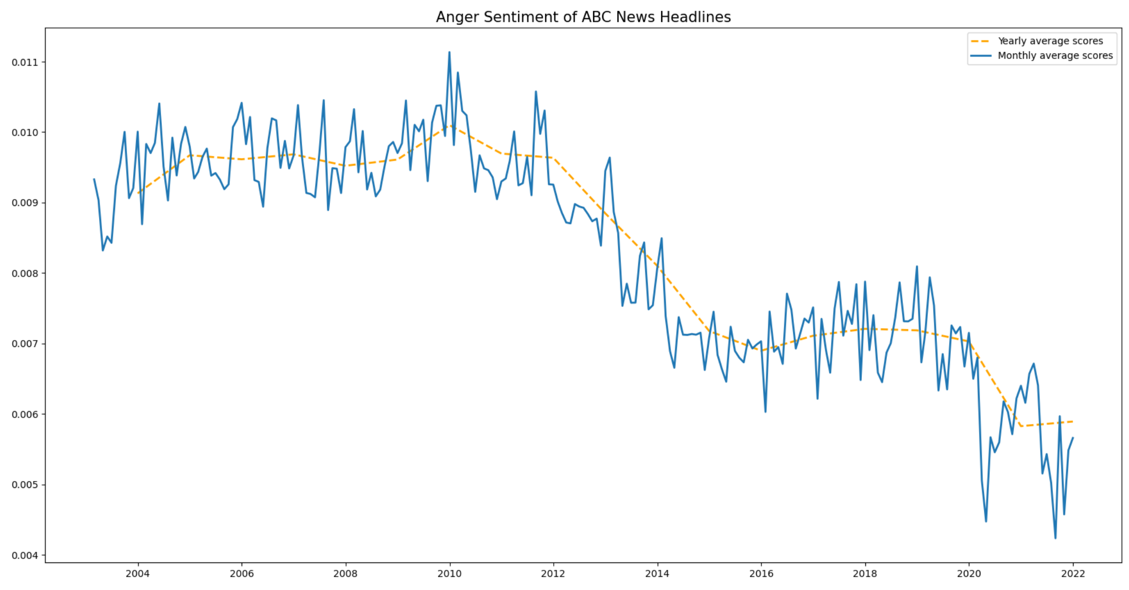
[19]:
```
  publish_date                                      headline_text     anger  \
0   2003-02-19  aba decides against community broadcasting lic…  0.000000
1   2003-02-19     act fire witnesses must be aware of defamation  0.000000
2   2003-02-19     a g calls for infrastructure protection summit  0.000000
3   2003-02-19           air nz staff in aust strike for pay rise  0.018519
4   2003-02-19      air nz strike to affect australian travellers  0.071429

     fear  negative  positive  sadness     trust  anticipation       joy  \
0  0.0000  0.000000  0.166667      0.0  0.000000      0.000000  0.000000
1  0.0625  0.031250  0.000000      0.0  0.000000      0.000000  0.000000
2  0.0000  0.047619  0.000000      0.0  0.047619      0.047619  0.000000
```

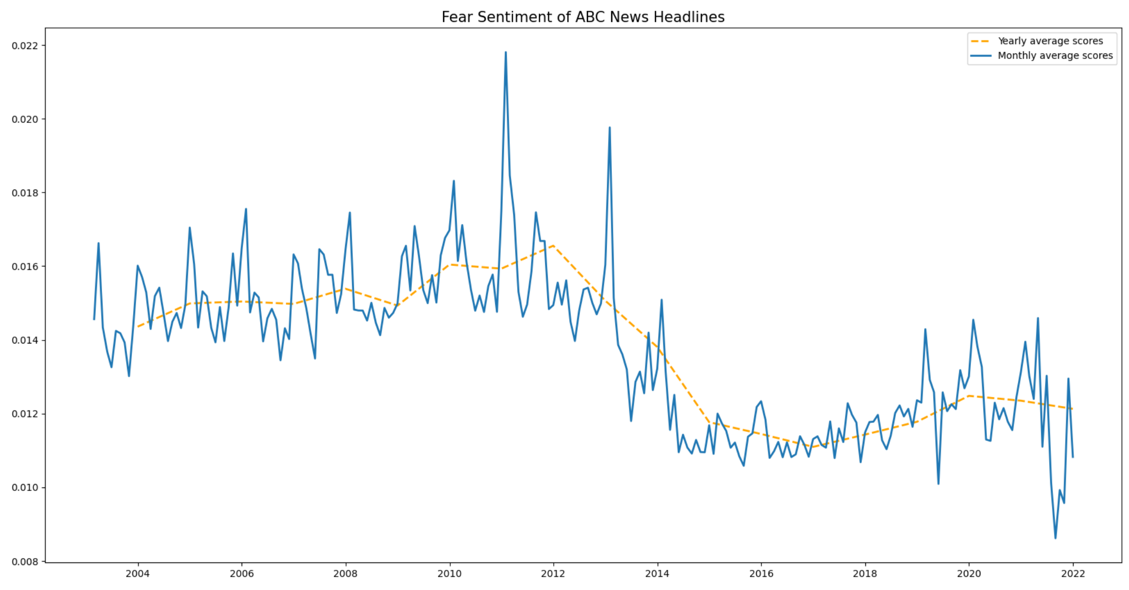
```
3  0.0000  0.018519  0.018519      0.0  0.018519      0.018519  0.018519
4  0.0000  0.071429  0.000000      0.0  0.000000      0.000000  0.000000

   disgust  surprise  anticip  word_count
0  0.00000       0.0      0.0           6
1  0.03125       0.0      0.0           8
2  0.00000       0.0      0.0           7
3  0.00000       0.0      0.0           9
4  0.00000       0.0      0.0           7
```

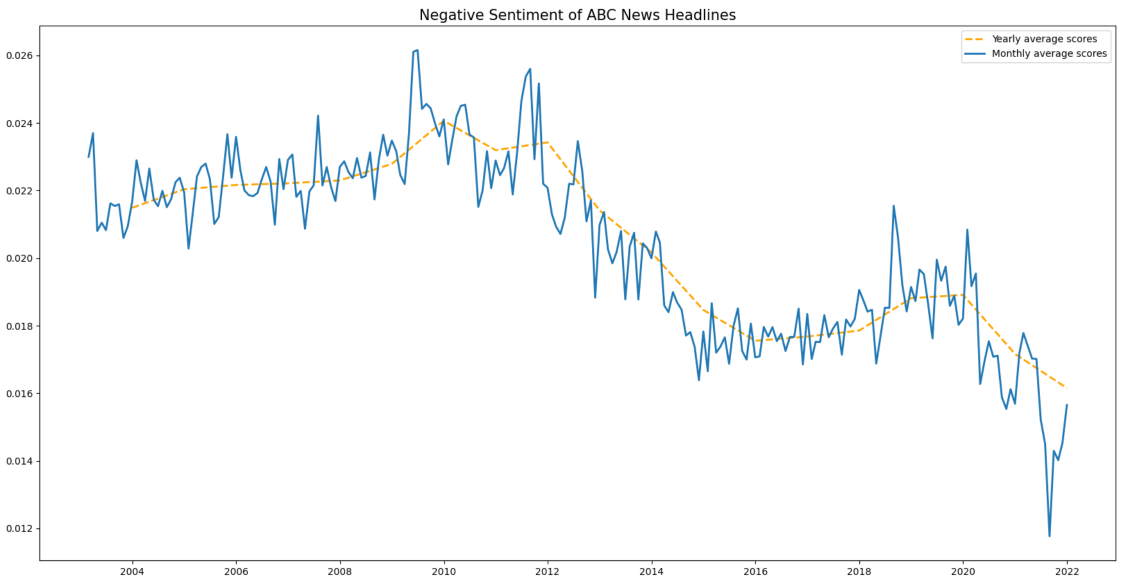
[20]:
```
nrc_yearly_averages = abc_nrc.resample('A',on='publish_date').mean()
nrc_monthly_averages = abc_nrc.resample('M',on='publish_date').mean()
```

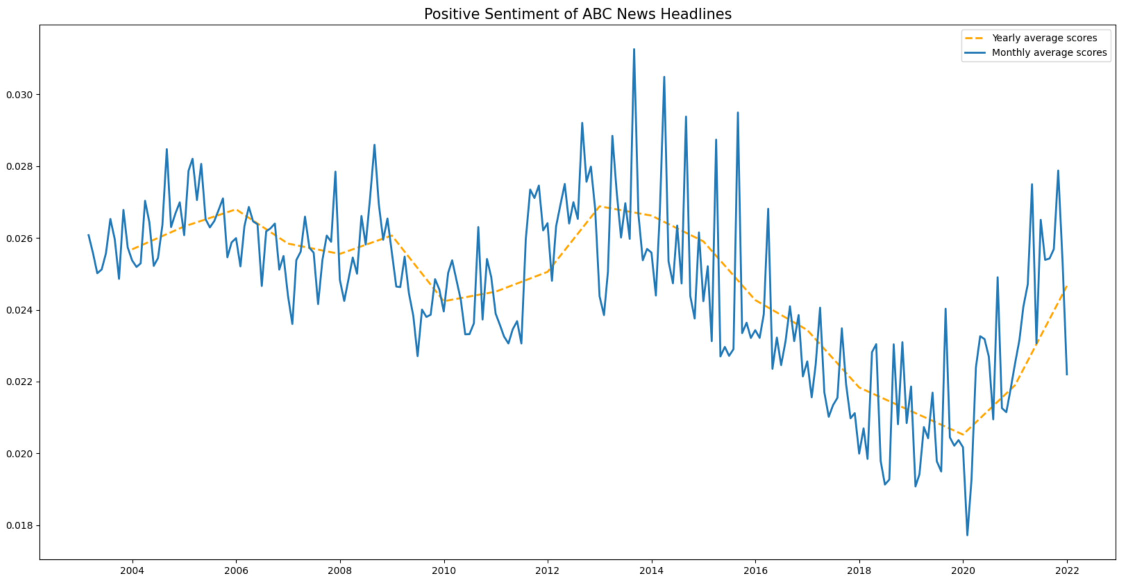
```
<ipython-input-20-d8d04474e0d3>:1: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  nrc_yearly_averages = abc_nrc.resample('A',on='publish_date').mean()
<ipython-input-20-d8d04474e0d3>:2: FutureWarning: The default value of
numeric_only in DataFrameGroupBy.mean is deprecated. In a future version,
numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  nrc_monthly_averages = abc_nrc.resample('M',on='publish_date').mean()
```

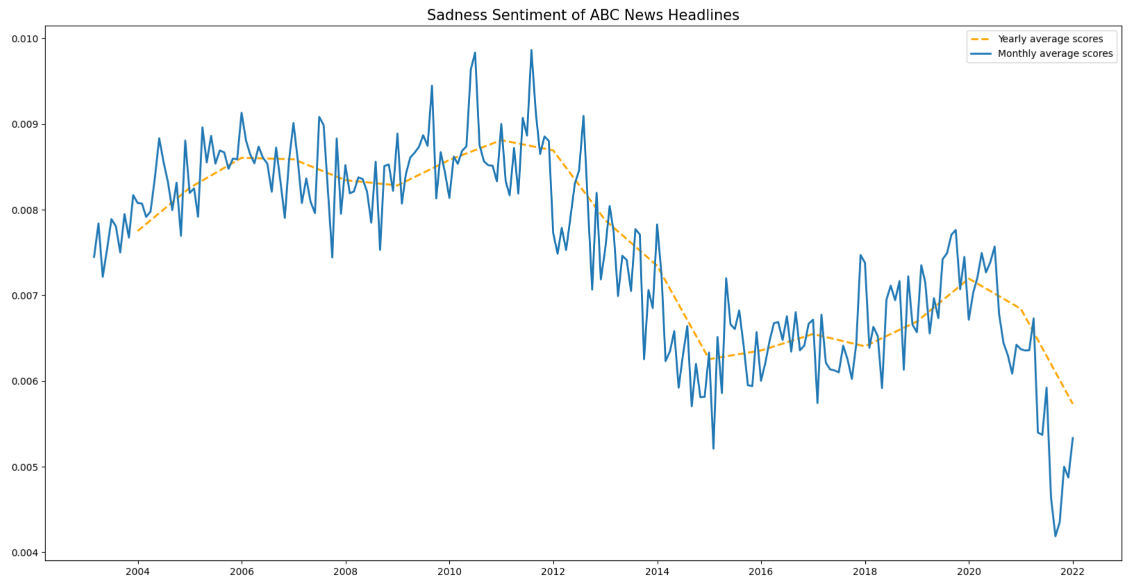
[21]:
```
for emotion in emotions:
    plt.figure(figsize=(20,10))
    plt.plot(nrc_yearly_averages.index,nrc_yearly_averages[emotion],␣
 ↪color='orange', linewidth=2, linestyle='dashed', label='Yearly average␣
 ↪scores')
    plt.plot(nrc_monthly_averages.index, nrc_monthly_averages[emotion], color =␣
 ↪'tab:blue', linewidth=2, label='Monthly average scores')
    plt.title('{} Sentiment of ABC News Headlines'.format(emotion.title()),␣
 ↪fontsize=15)
    plt.legend()
    plt.show()
```

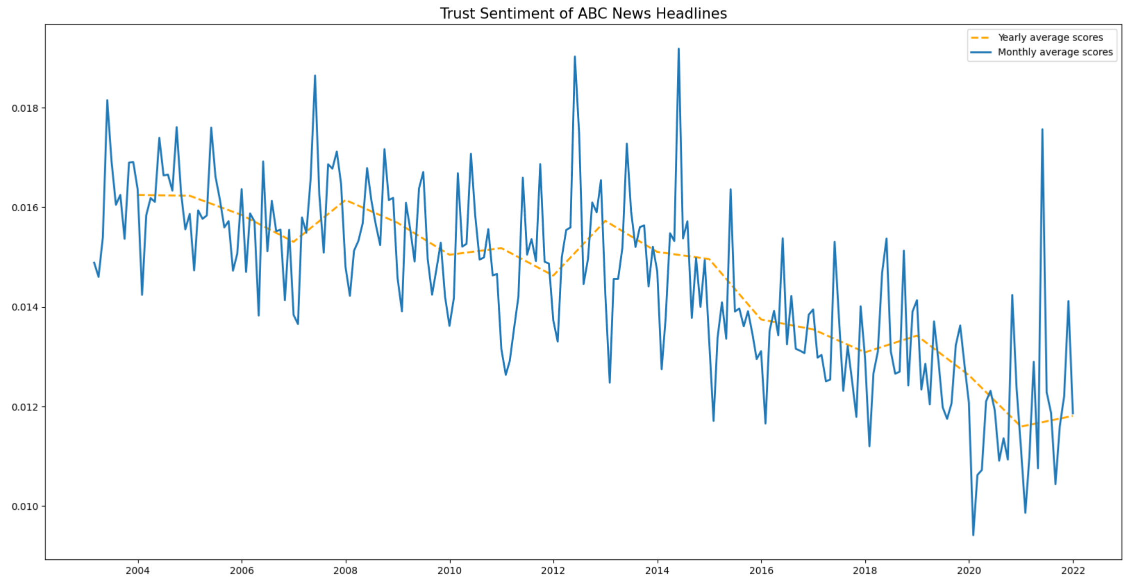

Anger Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.011
0.010
0.009
0.008
0.007
0.006
0.005
0.004
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022

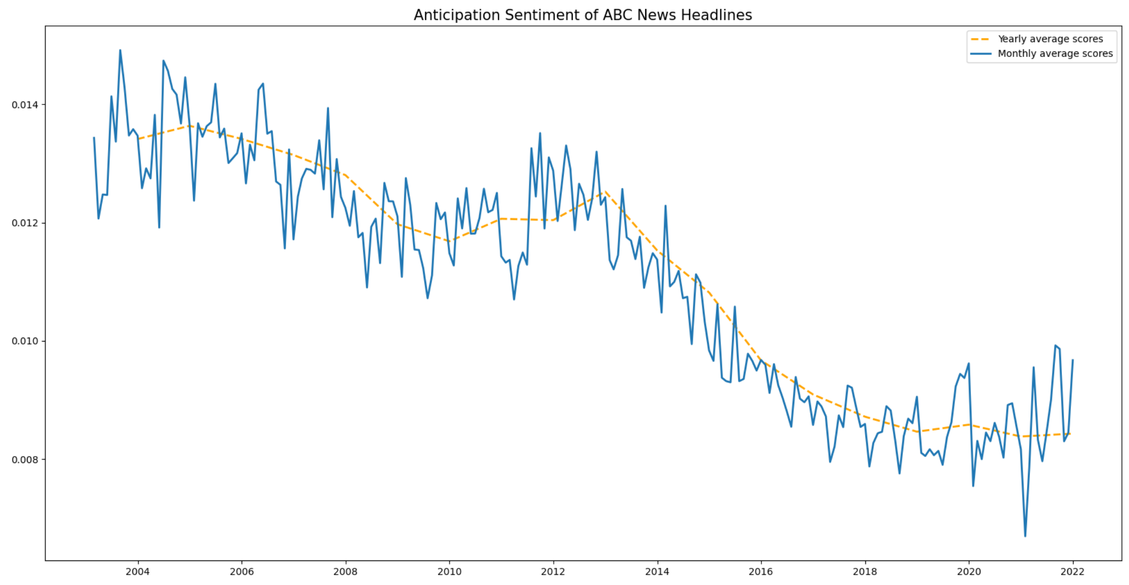

Fear Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.022
0.020
0.018
0.016
0.014
0.012
0.010
0.008
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022

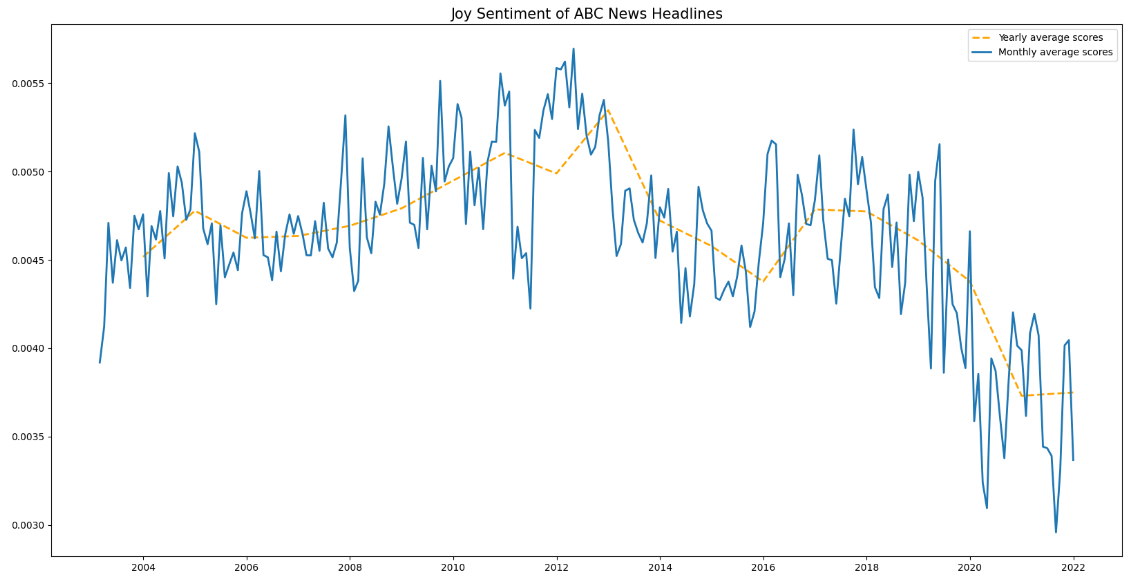

Negative Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.026
0.024
0.022
0.020
0.018
0.016
0.014
0.012
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022

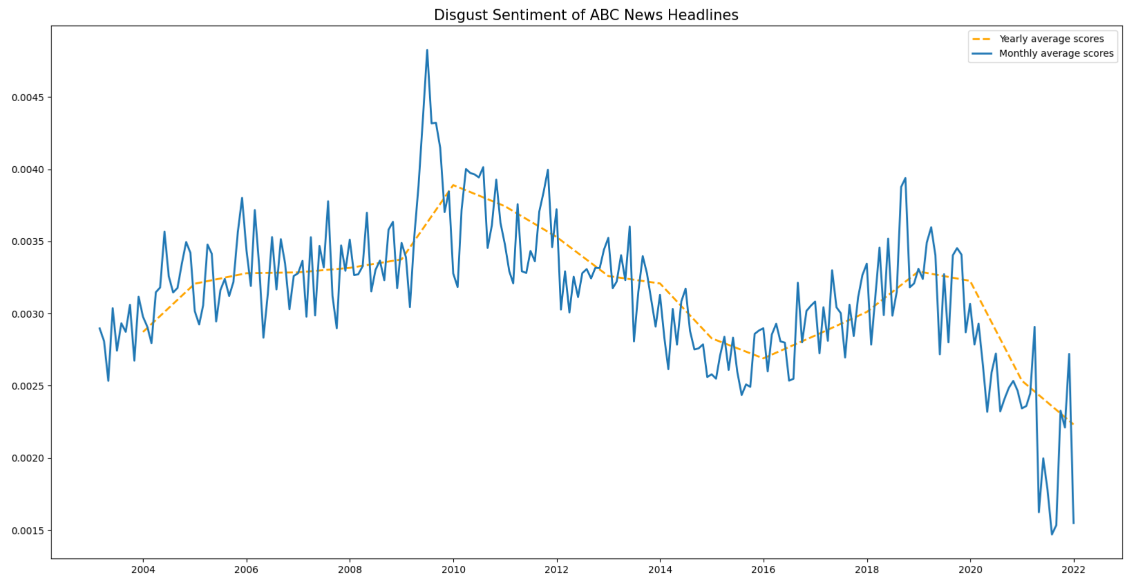

Positive Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.030
0.028
0.026
0.024
0.022
0.020
0.018
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022

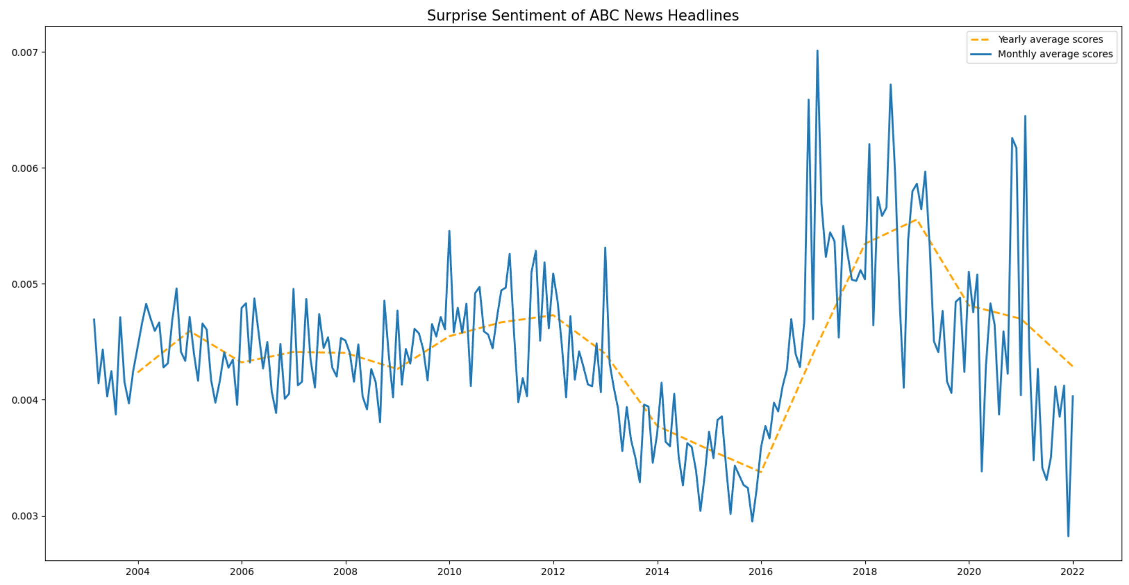

Sadness Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.010
0.009
0.008
0.007
0.006
0.005
0.004
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022


Trust Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.018
0.016
0.014
0.012
0.010
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022


Anticipation Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.014
0.012
0.010
0.008
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022

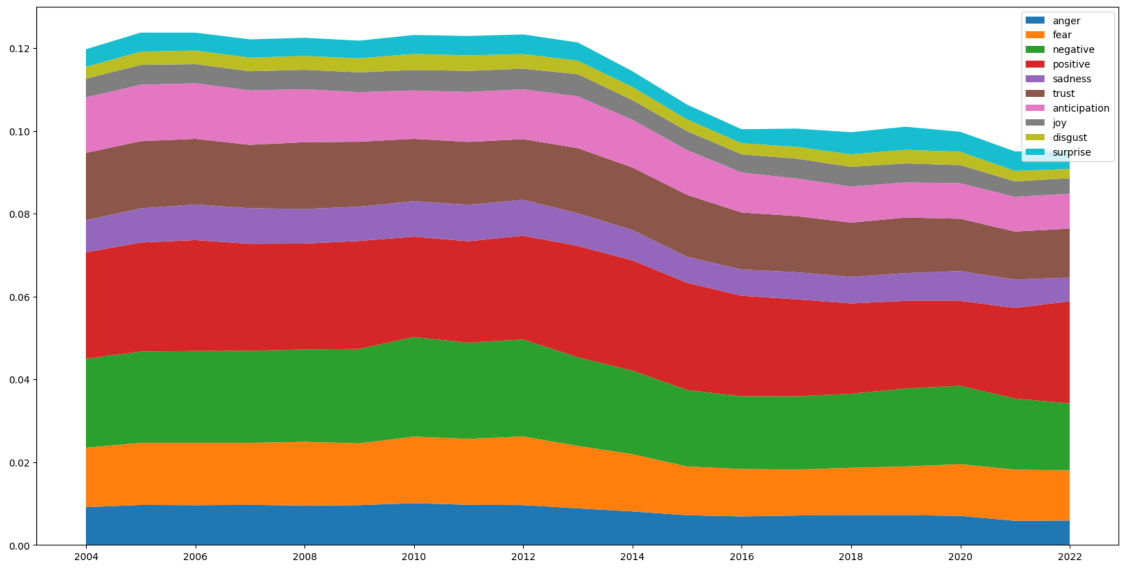

Joy Sentiment of ABC News Headlines
Yearly average scores
Monthly average scores
0.0055
0.0050
0.0045
0.0040
0.0035
0.0030
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022





Anger, fear and negative are going downward.

[22]:
```
x= nrc_yearly_averages.index
y= [nrc_yearly_averages[emotion].tolist() for emotion in emotions]
plt.figure(figsize=(20,10))
plt.stackplot(x,y, colors=('#1f77b4',
                           '#ff7f0e',
                           '#2ca02c',
                           '#d62728',
                                '#9467bd',
                                '#8c564b',
                                '#e377c2',
                                '#7f7f7f',
                                '#bcbd22',
                                '#17becf'), labels=emotions)
plt.legend()
plt.show()
```



```
[ ]:
```