In [ ]:
```
# -*- coding: utf-8 -*-
"""
Created on Fri May 22 16:13:14 2020

@author: victo
"""

###necessary libraries
import nltk
import flair
import pandas as pd
import glob
import os
from datetime import datetime
import re

# file where csv files lies
path = r'C:\Users\victo\Master_Thesis\scraperproject\audi\audi_scraper\spiders\news
'
all_files = glob.glob(os.path.join(path, "*.csv"))

# read files to pandas frame
list_of_files = []

for filename in all_files:
    list_of_files.append(pd.read_csv(filename,
                                     sep=',',
                                     encoding='cp1252',
                                     header=None,
                                     names=["url", "header", "release time", "artic
le content"]
                                     )
                         )

# Concatenate all content of files into one DataFrames
concatenate_list_of_files = pd.concat(list_of_files,
                                      ignore_index=True,
                                      axis=0,
                                      )

# removing duplicates
cleaned_dataframe = concatenate_list_of_files.sort_values(by='url', ascending=Fals
e)
cleaned_dataframe = cleaned_dataframe.drop_duplicates(subset=["url"], keep='first',
ignore_index=True)

print(cleaned_dataframe)

##formatting date column
dates = []
times = []
regex = r'(.*)(((1[0-2]|0?[1-9])\/(3[01]|[12][0-9]|0?[1-9])\/(?:[0-9]{2})?[0-
9]{2})|((Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(ust)?|Sep
(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?)\s+\d{1,2},\s+\d{4}))'
regex2 = r'((1[0-2]|0?[1-9]):([0-5][0-9]) ?([AaPp][Mm]))'

for date in cleaned_dataframe['release time']:
    matches = re.finditer(regex, date)
    for m in matches:
        date = m.group()
        date_formatted = date.replace(date[:2], '')
        convert_date = datetime.strptime(date_formatted, '%B %d, %Y')
```

```
        final_date = datetime.strftime(convert_date, "%Y-%m-%d")
        #print(final_date)
        dates.append(final_date)

for time in cleaned_dataframe['release time']:
    matches = re.finditer(regex2, time)
    for t in matches:
        time = t.group()
        convert_time = datetime.strptime(time, '%I:%M %p')
        time_formatted = datetime.strftime(convert_time, '%H:%M:%S')
        #print(time_formatted)
        times.append(time_formatted)

## adding modified date to data frame
cleaned_dataframe['date'] = dates
cleaned_dataframe['time'] = times
cleaned_dataframe['formatted date'] = cleaned_dataframe['date'] + str(' ') + cleane
d_dataframe['time']

## dropping unnecessary columns
del cleaned_dataframe['date']
del cleaned_dataframe['time']

# New words and values
#new_words = {'crushes': 10,
#             'beats': 5,
#             'misses': -5,
#             'trouble': -10,
#             'falls': -100,
#             }

#print('Start!')
# Instantiate the sentiment intensity analyzer with the existing lexicon
#vader = SentimentIntensityAnalyzer()
# Update the lexicon
#vader.lexicon.update(new_words)

#print('ok!')


flair_sentiment = flair.models.TextClassifier.load('en-sentiment')

## analysis on header
score_header = []
score_header_label = []
for header in cleaned_dataframe['header']:
    score = flair.data.Sentence(header)
    pre = flair_sentiment.predict(score)
    total_sentiment = score.labels[0]
    labscore = (total_sentiment.score)
    response = {'result': total_sentiment.value, 'score': "%.4f" % labscore}
    if response.get('result') == 'NEGATIVE':
        neg = response.get('score')
        neg = float(neg) * (-1)
        neg_label = response.get('result')
        score_header.append(neg)
        score_header_label.append(neg_label)
    else:
        pos = response.get('score')
        pos_label = response.get('result')
        score_header.append(pos)

## analysis on article content
score_content = []
```

```
score_content_label = []
for articlecontent in cleaned_dataframe['article content']:
    score = flair.data.Sentence(articlecontent)
    pre = flair_sentiment.predict(score)
    total_sentiment = score.labels[0]
    labscore = (total_sentiment.score)
    response = {'result': total_sentiment.value, 'score': "%.4f" % labscore}
    if response.get('result') == 'NEGATIVE':
        neg = response.get('score')
        neg = float(neg) * (-1)
        neg_label = response.get('result')
        score_content.append(neg)
        score_content_label.append(neg_label)
    else:
        pos = response.get('score')
        pos_label = response.get('result')
        score_content.append(pos)
        score_content_label.append(pos_label)

#print(type(score_content))

# Join the DataFrames
cleaned_dataframe['flair_sentiment_header_label'] = pd.DataFrame(score_header_labe
l)
cleaned_dataframe['flair_sentiment_header_score'] = pd.DataFrame(score_header)
cleaned_dataframe['flair_sentiment_content_label'] = pd.DataFrame(score_content_lab
el)
cleaned_dataframe['flair_sentiment_content_score'] = pd.DataFrame(score_content)


#print(cleaned_dataframe)

## saving outcome of flair to csv
current_date = datetime.today().strftime('%Y-%m-%d')
cleaned_dataframe.to_csv(r'C:\Users\victo\Master_Thesis\semanticanalysis\analysis_w
ith_flair\audi\outcome_using_flair\outcome_of_flair_on_audi_news_' + str(current_da
te) + '.csv', index=False)
#cleaned_dataframe.to_csv('test.csv', index=False)
```