```
from urllib.request import urlopen, Request
from bs4 import BeautifulSoup
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import pandas as pd
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
! python -m pip install nltk
```

```
finviz_url = 'https://finviz.com/quote.ashx?t='
```

```
tickers = ['AAPL']
news_tables = {}
```

```
from types import new_class
for ticker in tickers:
    url = finviz_url + ticker

    req = Request(url=url, headers={'user-agent': 'my-app'})
    response = urlopen(req)
    print(response)
    html = BeautifulSoup(response,'html')
    news_table = html.find(id='news-table')
    news_tables[ticker] = news_table
    break

print(news_table)
```

```
<http.client.HTTPResponse object at 0x7b0855f50880>
<table border="0" cellpadding="1" cellspacing="0" class="fullview-news-outer news-table" id="news-table" width="100%">
<tr class="cursor-pointer has-label" onclick="trackAndOpenNews(event, 'Investor\u0027s Business Daily', 'https://finance.yahoo.co
<td align="right" width="130">
            Dec-04-24 10:35PM
        </td>
<td align="left">
<div class="news-link-container">
<div class="news-link-left">
<a class="tab-link-news" href="https://finance.yahoo.com/m/4205eaa9-f620-3a0b-a81a-0e82c7c9fd0b/magnificent-seven-stocks%3A.html"
</div>
<div class="news-link-right">
<span>(Investor's Business Daily)</span></div></div></td></tr>
<tr class="cursor-pointer has-label" onclick="trackAndOpenNews(event, 'DigiTimes', 'https://www.digitimes.com/news/a20241205VL203
<td align="right" width="130">
            10:30PM
        </td>
<td align="left">
<div class="news-link-container">
<div class="news-link-left">
<a class="tab-link-news" href="https://www.digitimes.com/news/a20241205VL203/manufacturing-microsoft-hp-dell-apple.html" rel="nof
</div>
<div class="news-link-right">
<span>(DigiTimes)</span></div></div></td></tr>
<tr class="cursor-pointer has-label" onclick="trackAndOpenNews(event, 'Fortune', 'https://finance.yahoo.com/news/steve-jobs-convi
<td align="right" width="130">
            06:54PM
        </td>
<td align="left">
<div class="news-link-container">
<div class="news-link-left">
<a class="tab-link-news" href="https://finance.yahoo.com/news/steve-jobs-convinced-tim-cook-235400373.html" rel="nofollow" target
</div>
<div class="news-link-right">
<span>(Fortune)</span></div></div></td></tr>
<tr class="cursor-pointer has-label" onclick="trackAndOpenNews(event, 'InvestorPlace', 'https://investorplace.com/2024/12/four-re
<td align="right" width="130">
            05:52PM
        </td>
<td align="left">
<div class="news-link-container">
<div class="news-link-left">
<a class="tab-link-news" href="https://investorplace.com/2024/12/four-reasons-the-market-is-headed-higher/" rel="nofollow" target
</div>
<div class="news-link-right">
<span>(InvestorPlace)</span></div></div></td></tr>
<tr class="cursor-pointer has-label" onclick="trackAndOpenNews(event, 'Quartz', 'https://finance.yahoo.com/m/1f84807b-1ccc-3499-9
<td align="right" width="130">
            01:53PM
        </td>
<td align="left">
<div class="news-link-container">
<div class="news-link-left">
<a class="tab-link-news" href="https://finance.yahoo.com/m/1f84807b-1ccc-3499-92d0-53f784de822f/donald-trump-just-picked-who.html
</div>
```

```
<div class="news-link-right">
<span>(Quartz)</span></div></div></td></tr>
```

```python
parsed_data = []

# Loop through each row in the news table
for index, row in enumerate(news_table.find_all('tr')):  # Ensure all <tr> tags are iterated
    try:
        # Extract the title if the <a> tag exists
        title_tag = row.find('a')
        title = title_tag.text.strip() if title_tag else None

        # Extract the timestamp
        timestamp_tag = row.find('td')
        timestamp = timestamp_tag.text.strip() if timestamp_tag else None

        # Extract the source if the <span> tag exists
        source_tag = row.find('span')
        source = source_tag.text.strip() if source_tag else None

        # Append the parsed data only if a title exists
        if title:
            parsed_data.append({
                'title': title,
                'timestamp': timestamp,
                'source': source
            })
        else:
            print(f"Row {index}: Skipped as no title was found.")
    except Exception as e:
        print(f"Error parsing row {index}: {e}")

# Convert to a DataFrame for analysis
import pandas as pd

df = pd.DataFrame(parsed_data)
print(df.head())  # Display the first few rows of the extracted data
```

```
Row 5: Skipped as no title was found.
Row 16: Skipped as no title was found.
Row 32: Skipped as no title was found.
                                            title          timestamp  \
0  Magnificent Seven Stocks: Nvidia Stock Rallies...  Dec-04-24 10:35PM
1  Tata reportedly in talks with Microsoft, Dell,...            10:30PM
2  Steve Jobs convinced Tim Cook that Apple would...            06:54PM
3           Four Reasons the Market is Headed Higher            05:52PM
4  Donald Trump just picked who will take over th...            01:53PM

                      source
0  (Investor's Business Daily)
1                (DigiTimes)
2                  (Fortune)
3             (InvestorPlace)
4                   (Quartz)
```

```python
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk

# Download the VADER lexicon if not already downloaded
nltk.download('vader_lexicon')

# Initialize the VADER Sentiment Analyzer
vader = SentimentIntensityAnalyzer()
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```python
parsed_data = []
last_known_date = None  # Variable to store the last known date

# Loop through the news tables for each ticker
for ticker, news_table in news_tables.items():
    for row in news_table.findAll('tr'):
        try:
            # Safely extract the title
            title = row.find('a').text.strip() if row.find('a') else None

            # Safely extract date and time data
            date_data = row.find('td').text.strip().split(' ') if row.find('td') else None
            if date_data:
                if len(date_data) == 1:  # Only time is provided
                    time = date_data[0]
                    date = last_known_date  # Use the last known date
                elif len(date_data) >= 2:  # Both date and time are provided
                    date = date_data[0]
                    time = date_data[1]
                    last_known_date = date  # Update the last known date
                else:
                    date = last_known_date  # Use the last known date
                    time = None
            else:
                date = last_known_date  # Use the last known date
                time = None

            # Append the extracted data with ticker, date, time, and title
            parsed_data.append([ticker, date, time, title])

        except Exception as e:
            # Log any errors encountered during parsing
            print(f"Error parsing row for ticker {ticker}: {e}")

df = pd.DataFrame(parsed_data, columns=['ticker', 'date', 'time', 'title'])
# Fill missing titles with an empty string
df['title'] = df['title'].fillna("")

vader = SentimentIntensityAnalyzer()

f = lambda title: vader.polarity_scores(title)['compound']
df['compound'] = df['title'].apply(f)
df['date'] = pd.to_datetime(df['date'], format='%b-%d-%y')

# Display the result
#print(df)
plt.figure(figsize=(10, 8))
mean_df = df.groupby(['ticker', 'date'])['compound'].mean().reset_index()

print(mean_df)
```

```
  ticker       date  compound
0   AAPL 2024-11-27  0.113333
1   AAPL 2024-11-28  0.126337
2   AAPL 2024-11-29  0.014844
3   AAPL 2024-11-30  0.222986
4   AAPL 2024-12-01  0.182971
5   AAPL 2024-12-02 -0.060129
6   AAPL 2024-12-03  0.008300
7   AAPL 2024-12-04  0.045885
<Figure size 1000x800 with 0 Axes>
```