**Natural Language Processing Trading Simulator by Hanaa Ratip**

Over the past few years, I have invested in bitcoin which has led to my recent interest in stocks and shares and wanting to understand how to analyse the markets. As I learnt more about the process of choosing which stocks to buy and at which prices and times, I thought ‘wouldn’t it be useful to have a piece of software that can analyse the status of a company for you?’. This was primarily because it would significantly speed up the time taken to choose which company to invest in.

Therefore, the software I have created saves time and energy as it takes away the need to research a company and read through endless articles to find the background knowledge needed before you invest. For example, recent scandals (or public successes) may impact the stock price. For example, Amazon’s shares fell about 47% throughout 2022 due to high inflation, increased interest rates and a lack of digital advertising. This information and background are essential when deciding whether to invest.

Therefore, throughout this essay, I will explain the nature in which I have created this piece of software to output efficiently and clearly what a user needs to know before investing. The program will output the date and time the stock was bought at along with the price, company name and symbol and the part of the news article signally positivity or negativity.

**Libraries**

import requests

import nltk

from nltk.sentiment.vader import SentimentIntensityAnalyzer

import yfinance as yf

from datetime import datetime

The beginning of the Python scripts involves importing any necessary libraries. A Python library is a public, reusable chunk of code that can be included in programs.

The ‘requests’ library is used to fetch data from a given URL.

The ‘nltk’ library is used to calculate the sentiment (how positive or negative) of a given piece of text. I utilised this library to serve as one of the two confirmations required to execute a trade.

‘yfinance’ is a library that accesses the Yahoo Finance API (application programming interface) and collects data and financial information on a given stock. It obtains real time stock quotes and information.

The ‘datetime’ library allows programmers to handle dates and time programmatically as well as define specific formats to display them.

**APIs**

APIs serve to provide data to whomever requests it. The diagram below depicts how data is transferred between a client (i.e. this Python script) and a server by means of an API.

Diagram

Description automatically generated

(Image source: Codecademy)

Marketaux is a service which provides news article data via an API. This program uses these news articles to generate a sentiment which determines a trade decision.

To retrieve news articles from the Marketaux API, I generated a private API key (crossed out for privacy concerns) which corresponds to my account. I appended this key to the APIs URL and obtained the data by passing it as a parameter into requests.get(). This returned the news article data in a text format, so I converted it into JSON format to allow for the data processing to follow.

page = requests.get('https://api.marketaux.com/v1/news/all?exchanges=NYSE&filter\_entities=true&limit=10&published\_after=2023-02 19T21:21&api\_token=668WuIXvMAeFigMNp6SftkC4nRrSG339SQbz4Qx7')

response = page.json()

**Classes**

A class is a code template with various attributes and behaviours for creating objects.

In this program, I created a ‘News’ class to represent each news article given by the API.

class News:

def \_\_init\_\_(self, url, highlights, sentiment, company\_symbol, company\_name):

self.url = url

self.highlights = highlights

self.sentiment = sentiment

self.company\_symbol = company\_symbol

self.company\_name = company\_name

As the code above shows, when a ‘News’ object is created, it is initialised with a URL of the online article, a list of highlights in the article, a sentiment, a company symbol (of the company in which the article writes about) and a company name.

**Natural Language Processing**

As mentioned above, ‘nltk’ is a library used to calculate the sentiment of a given piece of text. The following function, ‘analyze\_sentiment’, receives the piece of text as a parameter called ‘article’ and passes this through a function called ‘polarity\_scores()’ to generate a sentiment score. If the sentiment score is greater than 0, this indicates that the text is ‘positive’, if the sentiment is less than 0, it will return ‘negative’, and if it is exactly 0, it will return ‘neutral’.

nltk.download('vader\_lexicon')

def analyze\_sentiment(article):

sia = SentimentIntensityAnalyzer()

sentiment\_scores = sia.polarity\_scores(article)

if sentiment\_scores['compound'] > 0:

return 'positive'

elif sentiment\_scores['compound'] < 0:

return 'negative'

else:

return 'neutral'

**Data Processing**

The news article data obtained from the Marketaux API (consisting of the attributes defined in the ‘News’ class) and the ‘nltk’ library must now be processed.

The following code processes the news article data in JSON format by accessing it’s properties and storing the corresponding values in variables (temporary storage in memory).

The sentiment score attribute in the API data is the second confirmation required to execute a trade (the first being the sentiment outputted by the ‘nltk’ library). If the sentiment score ranges from 0 to 1; if it is greater than 0.65, this indicates that the article is positive, if it is less than 0.35, negative, and between those two values, neutral.

allNews = []

for data in response['data']:

url = data['url']

highlights = set()

sentiment = ''

company\_symbol = ''

company\_name = ''

for entity in data['entities']:

sentiment\_score = entity['sentiment\_score']

if sentiment\_score:

if sentiment\_score > 0.65:

sentiment = 'positive'

elif sentiment\_score < 0.35:

sentiment = 'negative'

else:

sentiment = 'neutral'

company\_symbol = entity['symbol']

company\_name = entity['name']

highlights.add(entity['highlights'][0]['highlight'])

After the data processing steps, the program compares the two sentiments obtained by the ‘nltk’ library and the Marketaux API. Two confirmations are required to execute a trade i.e. both sentiments have to be the same.

The following code checks if both sentiments match and if so, the ‘News’ object is appended to a list, ‘allNews’, which contains all the news articles which have matching sentiments.

match = True

for highlight in highlights:

new\_sentiment = analyze\_sentiment(highlight)

if sentiment != '':

if new\_sentiment != sentiment:

match = False

else:

sentiment = new\_sentiment

if match:

allNews.append(News(url, highlights, sentiment, company\_symbol, company\_name))

**Simulating Trades**

The final step of the program is to simulate a trade based on the sentiments calculated above. The code below does the following:

1. It sifts through each ‘News’ object in ‘allNews’.
2. It uses the ‘yfinance’ library to obtain the current stock price for the company in which the news article is about and rounds it to 4 decimal places for readability purposes.
3. If the sentiment is positive, it displays a ‘buy’ trade and vice versa. The time of the trade (i.e. the current time) is calculated and displayed too in ‘hh:mm:ss’ format.

for news in allNews:  
 ticker = yf.Ticker(news.company\_symbol)  
 price\_data = ticker.history(period='1d')  
 price = round(price\_data['Close'][0], 4)  
 print('Article URL:')  
 print(news.url)  
 if news.sentiment == 'positive':  
 print(f'The sentiment of this article is positive.')  
 print(f'Bought {news.company\_symbol} @ ${price} {datetime.now().strftime("%H:%M:%S")}')  
 elif news.sentiment == 'negative':  
 print(f'The sentiment of this article is negative.')  
 print(f'Sold {news.company\_symbol} @ ${price} {datetime.now().strftime("%H:%M:%S")}')  
 else:  
 print(f'The sentiment of this article is neutral - no trade will be executed.')