Eve-GPT

Group 6

Yang Fan (Blithe)	3035771646
Li Xinran (Anna)	3035767437
Zhu Jiarui (Janet)	3035638791
Fung Ho Kit (Justin)	3035779105

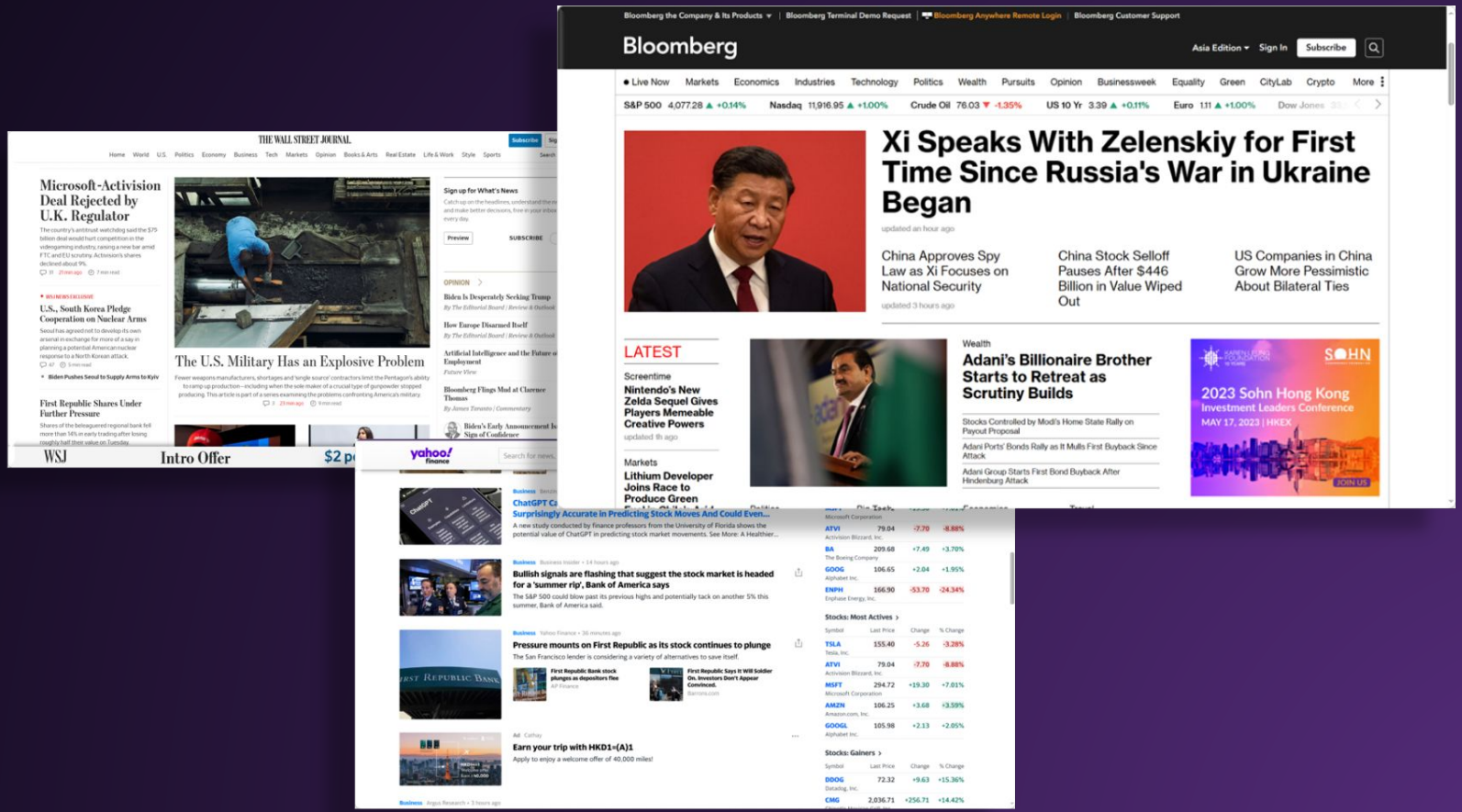

THE WALL STREET JOURNAL.
Microsoft-Activision Deal Rejected by U.K. Regulator
The U.S. Military Has an Explosive Problem
U.S., South Korea Pledge Cooperation on Nuclear Arms
First Republic Shares Under Further Pressure
WSJ
Intro Offer
$2 p
yahoo! finance
ChatGPT Ca
Surprisingly Accurate in Predicting Stock Moves And Could Even...
Bullish signals are flashing that suggest the stock market is headed for a 'summer rip', Bank of America says
Pressure mounts on First Republic as its stock continues to plunge
Earn your trip with HKD1=(A)1
Bloomberg
Xi Speaks With Zelenskiy for First Time Since Russia's War in Ukraine Began
China Approves Spy Law as Xi Focuses on National Security
China Stock Selloff Pauses After $446 Billion in Value Wiped Out
US Companies in China Grow More Pessimistic About Bilateral Ties
LATEST
Nintendo's New Zelda Sequel Gives Players Memeable Creative Powers
Lithium Developer Joins Race to Produce Green
Wealth
Adani's Billionaire Brother Starts to Retreat as Scrutiny Builds
Stocks Controlled by Modi's Home State Rally on Payout Proposal
Adani Ports' Bonds Rally as It Mulls First Buyback Since Attack
Adani Group Starts First Bond Buyback After Hindenburg Attack
2023 Sohn Hong Kong
Investment Leaders Conference
MAY 17, 2023 | HKEX
JOIN US
Stocks: Most Actives
Stocks: Gainers

Table of **contents**

Step 1:
Web Scraping

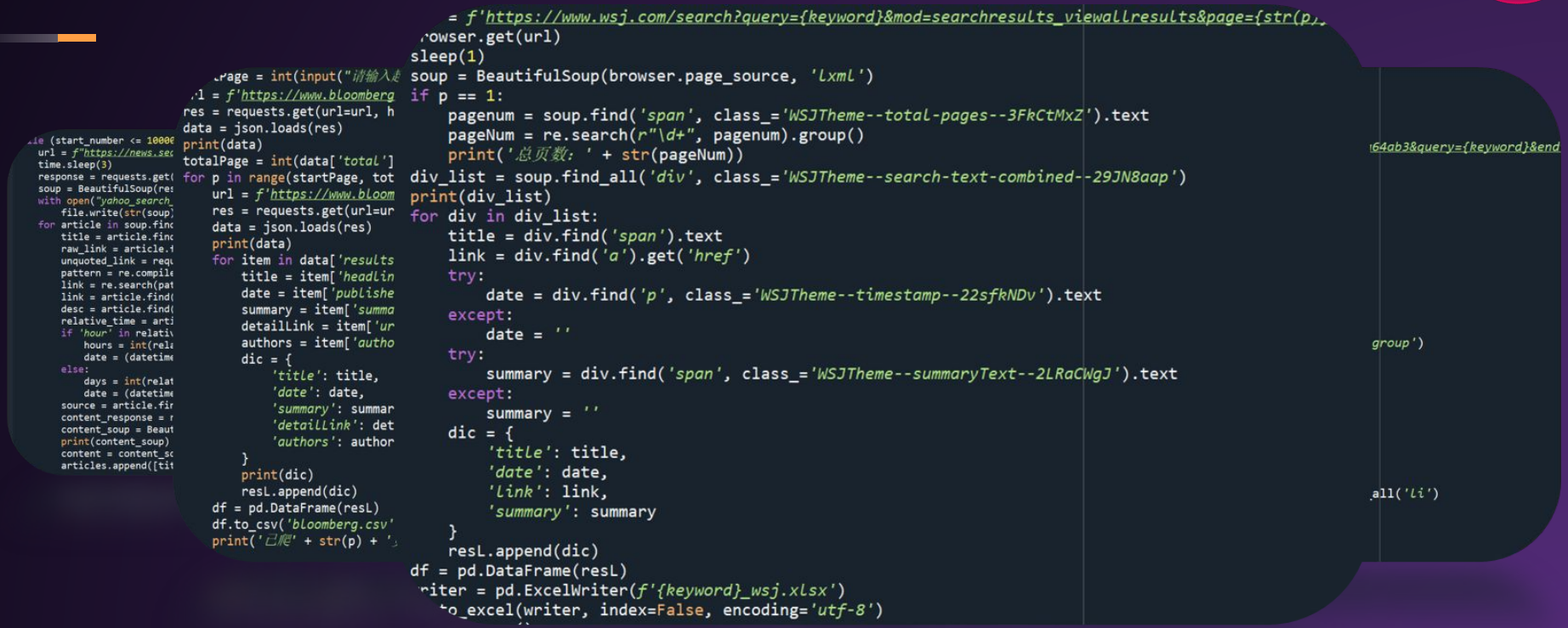
CNBC

01

```
main():
keyword = input("PLease Input Your Keyword: ")
endindex = 0
while 1:
    url = f'https://api.queryly.com/cnbc/json.aspx?queryly_key=31a35d40a9a64ab3&query={keyword}&end
    res = requests.get(url=url, headers=headers).json()
    data = res['results']
    for item in data:
        title = item['cn:title']
        date = item['_pubDate']
        summary = item['description']
        link = item['url']
        section = item['section']
        resp = requests.get(url=link, headers=headers).text
        soup = BeautifulSoup(resp, 'lxml')
        try:
            result=''
            first_group = soup.select_one('.ArticleBody-articleBody > div.group')
            text = [first_group] + first_group.find_next_siblings()
            for elem in text:
                result += elem.text.strip()
        except:
            result = ''
        # print(resp)
        if soup.find('div', class_='RenderKeyPoints-list') is None:
            summary_ = summary
        else:
            content_li = soup.find('div', class_='group').find('ul').find_all('li')
            content = ''
            for li in content_li:
                content += li.text
            summary_ = content
```

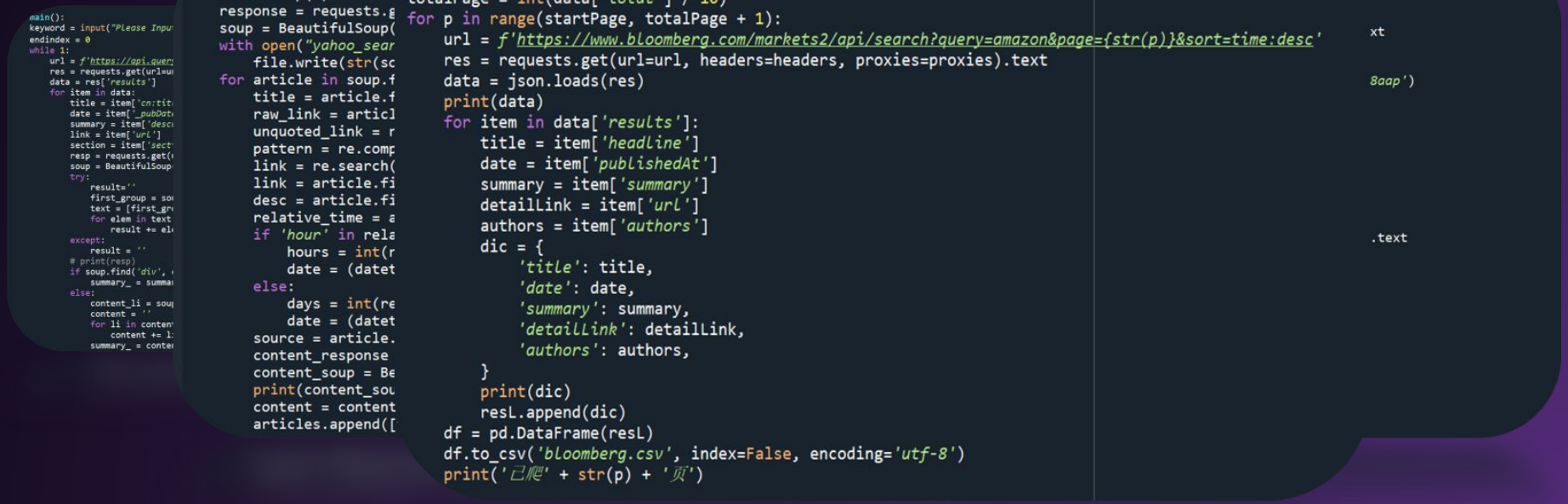
The Wall Street Journal

01



Bloomberg

01



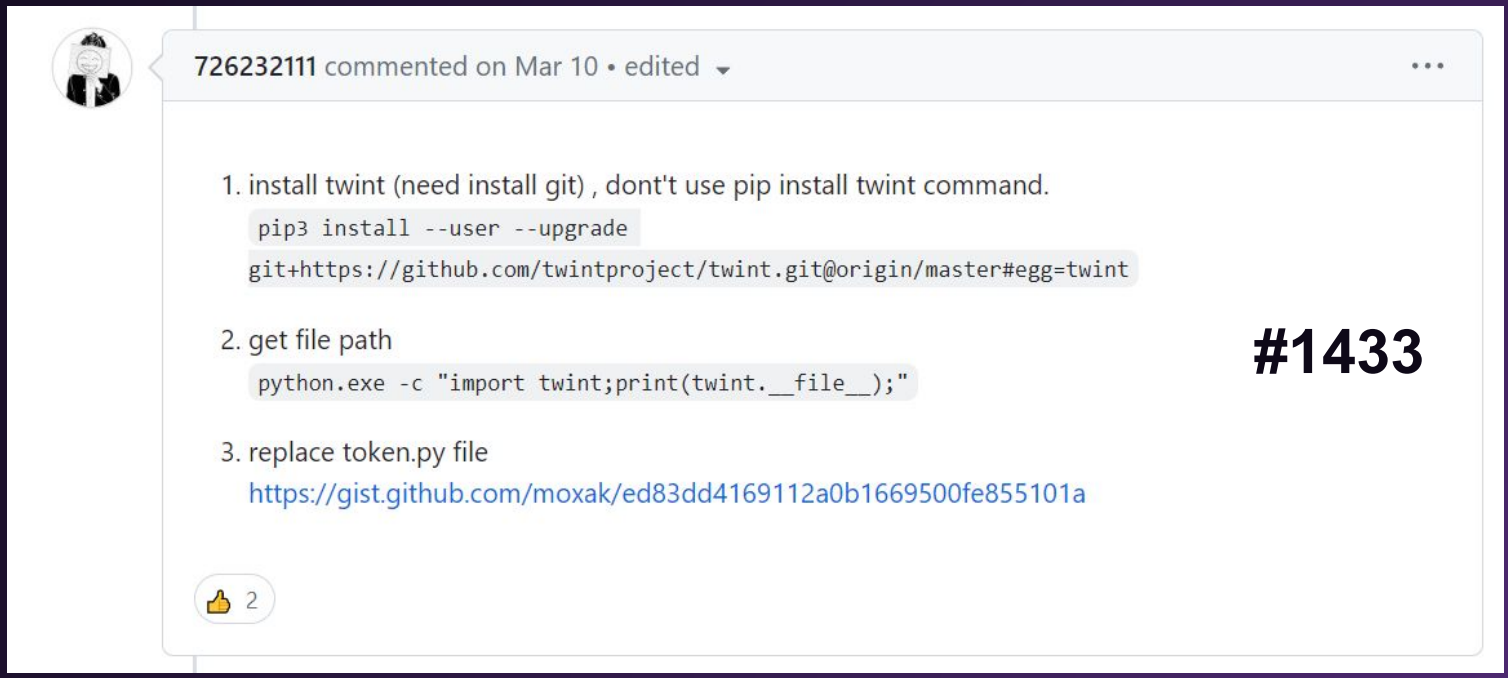
Yahoo Finance

01

```
le (start_number <= 10000):
  url = f"https://news.search.yahoo.com/search?p={keyword}&b={start_number}"
  time.sleep(3)
  response = requests.get(url, headers = headers)
  soup = BeautifulSoup(response.content, "html.parser")
  with open("yahoo_search_results.txt", "w", encoding="utf-8") as file:
      file.write(str(soup))
  for article in soup.find_all("div", class_="NewsArticle"):
      title = article.find("h4", class_="s-title").text.strip()
      raw_link = article.find('a').get('href')
      unquoted_link = requests.utils.unquote(raw_link)
      pattern = re.compile(r'RU=(.+)\/RK')
      link = re.search(pattern,unquoted_link).group(1)
      link = article.find('a').get('href')
      desc = article.find('p','s-desc').text.strip()
      relative_time = article.find('span','s-time').text.strip()[2:]
      if 'hour' in relative_time:
          hours = int(relative_time.split()[0])
          date = (datetime.utcnow() - timedelta(hours=hours)).strftime("%Y%m%d")
      else:
          days = int(relative_time.split()[0])
          date = (datetime.utcnow() - timedelta(days=days)).strftime("%Y%m%d")
      source = article.find('span','s-source').text
      content_response = requests.get(link, headers=headers)
      content_soup = BeautifulSoup(content_response.content, "html.parser")
      print(content_soup)
      content = content_soup.select_one(".caas-body").get_text()
      articles.append([title, source, link, desc, date])
```

Twitter——Twint

01



This repository has been archived by the owner on Mar 30, 2023. It is now read-only.

Twitter——Selenium

01

```
while driver.execute_script(js1) > old_scroll_height:  # compare the height
    div_list = driver.find_elements(By.XPATH,
                                    '//*[@id="react-root"]/div/div/div[2]/main/div/div/div/div[1]
    for div in div_list:
        try:
            content = div.find_element(By.XPATH, './/div[@data-testid="tweetText"]')
        except Exception as e:
            print(e)
            continue
        if content.get_attribute('id') in content_l:
            continue
        cnt += 1
        print(content.text)
        try:
            time_element = div.find_element(By.XPATH, './/time').text
        except Exception as e:
            print(e)
            continue
        link_div = div.find_element(By.XPATH, './/div[@class="css-1dbjc4n r-18u37iz r-1q142lx"]')
        link_element = link_div.find_element(By.XPATH, './/a').get_attribute('href')
        content_l.append(content.get_attribute('id'))
        dic = {
            'Content': content.text,
            'Time': time_element,
            'Link': link_element
        }
        print(dic)
        resL.append(dic)
        df = pd.DataFrame(resL)
        writer = pd.ExcelWriter(f'{key}_twi.xlsx')
        df.to_excel(writer, index=False, encoding='utf-8')
        writer.save()
    old_scroll_height = driver.execute_script(js1)  # get current height
    driver.execute_script(js2)  # pull the weber
    for j in range(5):
        actions.send_keys(Keys.PAGE_DOWN).perform()
        time.sleep(1)
    time.sleep(4)  #leave load time
    if (driver.execute_script(js1) <= old_scroll_height): # deal with the crush of web pages
        cnt = 0
        winsound.Beep(3000, 3000) #sound alert
        input('Enter:')
        old_scroll_height = 0  #
        js1 = 'return document.body.scrollHeight'
```

02

Step 2:
Data Processing

Text Data: Overview

	Bloomberg	CNBC	WSJ	Yahoo
Apple	10000 rows link, date, title, summary, author	7400 rows section, date time, title, summary...	1883 rows link, date time, title, summary	1000 rows link, source, date, title, description
Google	9890 rows link, date, title, summary, author	3080 rows section, date time, title, summary...	2384 rows link, date time, title, summary	1000 rows link, source, date, title, description
⋮	⋮	⋮	⋮	⋮
Nvidia	2980 rows link, date, title, summary, author	2090 rows section, date time, title, summary...	147 rows link, date time, title, summary	1000 rows link, source, date, title, description

Text Data: What Our Model Need

FinBERT

- supports many common English punctuations
- can split a paragraph into sentences

date	platform	title	summary
2023-04-17	bloomberg	text	text
⋮	⋮	⋮	⋮
2020-07-04	wsj	text	text

Text Data: Ready for Model Input

	A	B	C	D	E	F
1	date	platform	title	summary		
2	2023-04-17	bloomberg	Transcript: Sc	Over time, we expect the world to get r		
3	2023-04-17	bloomberg	Wells Fargo F	Wells Fargo & Co. leaders are privately		
4	2023-04-17	bloomberg	Apple's Buyb	One of the most important numbers in		

Stock Data: Where We Get It

```
>>> import yfinance as yf
```

```
In [229]: #df = yf.download(tickers = 'TSLA', start = '2022-03-11',end = '2023-03-11')
          company = "TSLA"
          Starting = "2020-07-20"
          Ending = "2023-04-14"

          tickers = yf.Tickers(company)
          StockPrice = tickers.tickers[company].history(start=Starting,end=Ending)
          StockPrice.index = pd.to_datetime(StockPrice.index).date
          StockPrice
```

executed in 168ms, finished 22:42:44 2023-04-26

	Open	High	Low	Close	Volume	Dividends	Stock Splits
2020-07-20	101.267334	110.000000	99.199997	109.533333	256821000	0.0	0.0
2020-07-21	109.328667	111.666664	103.866669	104.557335	241608000	0.0	0.0
2020-07-22	106.599998	108.428001	104.133331	106.155334	212416500	0.0	0.0
2020-07-23	111.930000	112.599998	98.718002	100.871330	364927500	0.0	0.0
2020-07-24	94.400665	97.666664	91.102669	94.466667	290949000	0.0	0.0
...	...	...	...	...	...	...	...
2023-04-06	183.080002	186.389999	179.740005	185.059998	123857900	0.0	0.0

Stock Data: Calculate Technical Analysis Indicators

```
>>> import pandas_ta as ta
```

```
In [28]: Full_data['RSI']=ta.rsi(Full_data.Close, length=30)
         Full_data['EMA']=ta.ema(Full_data.Close, length=30)
```

```
In [29]: data_set = Full_data.iloc[30:-1, 6:14]
         data_set
```

executed in 14ms, finished 03:39:03 2023-04-27

	score_title	score_summary	RSI	EMA	TargetNextClose
2020-08-31	0.064944	0.178783	78.587101	115.660970	158.350006
2020-09-01	0.139595	0.208545	72.223916	118.415101	149.123337
2020-09-02	-0.175306	0.031887	65.679839	120.396277	135.666672
2020-09-03	-0.072719	-0.187810	57.780923	121.381464	139.440002
2020-09-04	-0.066646	-0.387875	59.204111	122.546531	110.070000
...	...	...	...	...	...
2023-04-05	0.159216	0.101471	50.021287	189.355496	185.059998
2023-04-06	-0.080997	0.019024	49.882707	189.078367	184.509995
2023-04-10	-0.066649	-0.001051	49.712360	188.783633	186.789993
2023-04-11	-0.112037	-0.080313	50.438172	188.655011	180.539993
2023-04-12	0.051657	0.028311	48.454957	188.131462	185.899994

658 rows × 5 columns

03

Step 3: Sentiment Analysis

FinBERT

FinBERT: Financial Sentiment Analysis with Pre-trained Language Models

- Araci, D. (2019).
- Improve BERT with further pretraining on
  - TRC2-financial.
  - Financial PhraseBank.*

	All data			Data with 100% agreement		
(lr)2-4 (lr)5-7 Model	Loss	Accuracy	F1 Score	Loss	Accuracy	F1 Score
LSTM	0.81	0.71	0.64	0.57	0.81	0.74
LSTM with ELMo	0.72	0.75	0.7	0.50	0.84	0.77
ULMFit	0.41	0.83	0.79	0.20	0.93	0.91
LPS	-	0.71	0.71	-	0.79	0.80
HSC	-	0.71	0.76	-	0.83	0.86
FinSSLX	-	-	-	-	0.91	0.88
FinBERT	**0.37**	**0.86**	**0.84**	**0.13**	**0.97**	**0.95**

FinBERT

FinBERT: Financial Sentiment Analysis with Pre-trained Language Models

- Araci, D. (2019).
- Improve BERT with further pretraining on
  - TRC2-financial.
  - Financial PhraseBank.*

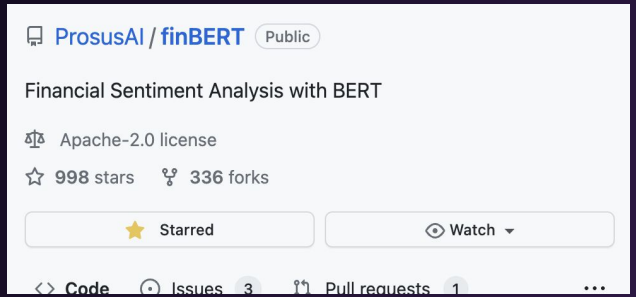
	All data			Data with 100% agreement		
(lr)2-4 (lr)5-7 Model	Loss	Accuracy	F1 Score	Loss	Accuracy	F1 Score
LSTM	0.81	0.71	0.64	0.57	0.81	0.74
LSTM with ELMo	0.72	0.75	0.7	0.50	0.84	0.77
ULMFit	0.41	0.83	0.79	0.20	0.93	0.91
LPS	-	0.71	0.71	-	0.79	0.80
HSC	-	0.71	0.76	-	0.83	0.86
FinSSLX	-	-	-	-	0.91	0.88
FinBERT	**0.37**	**0.86**	**0.84**	**0.13**	**0.97**	**0.95**

FinBERT: Financial Sentiment Analysis with BERT

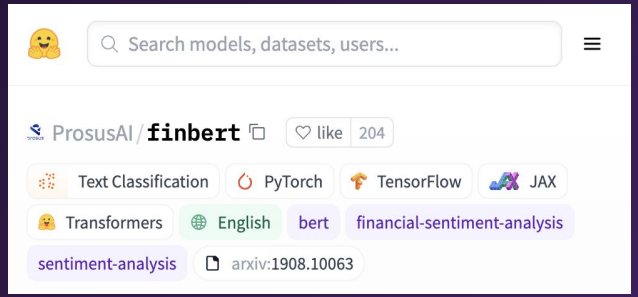
- Prosus AI Team

03

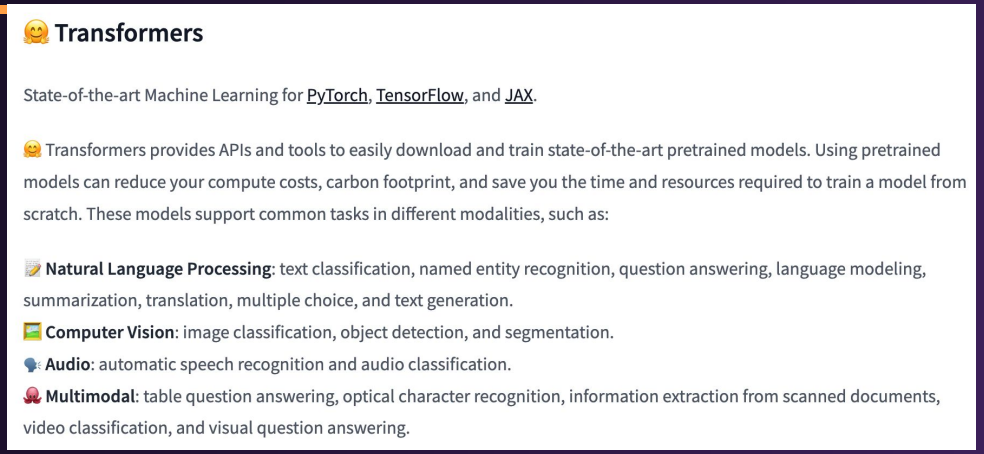
Code base



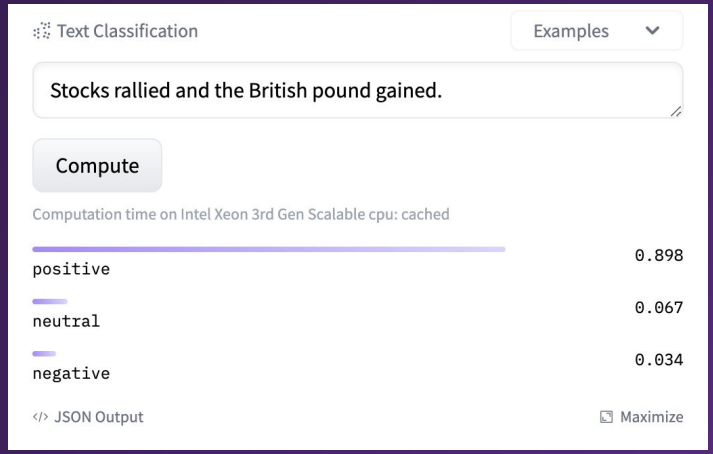
Pre-trained parameters



Model import API



Demo



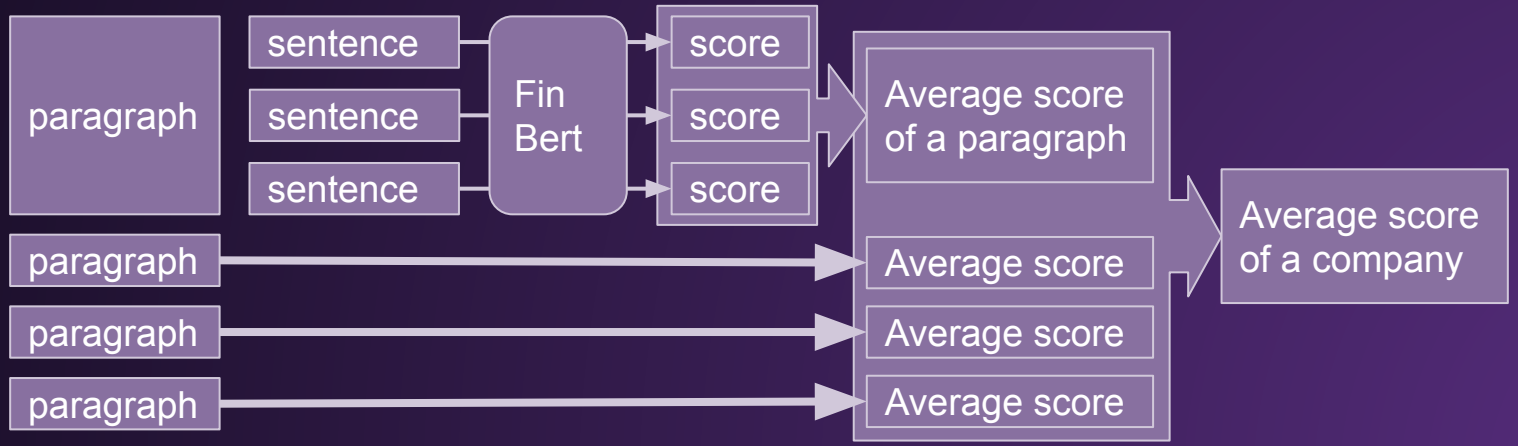
Changes and Innovations

- Add text validity checking before process

Changes and Innovations

- Add text validity checking before process
- One sentiment score per sentence

  ⇒ Average score of the company per day



Changes and Innovations

- Add text validity checking before process
- One sentiment score per sentence

  ⇒ Average score of the company per day
- Output baseline scores for dates without textual information

  ⇒ Ensure data have consecutive dates for time series analysis in next step

Processing Results

- 12000 records / hour with HKU GPU farm
- Daily news semitation analysis score for:

  6 companies:

  - Amazon, Apple, Google, Microsoft, Nvidia, Tesla

  1000+ days:

  - 2020-07-04 ~ 2023-04-17

Step 4: EveGPT

Workflow

News articles
Blogs
Tweets

Scrape (Part 1) → Process (Part 2) → Finbert (Part 3)

Stock Price

Date	2023.03.11	2023.03.12	…	2023.03.24
Classification result Title_Score Summary_Score	0.1	-0.3	…	???
Indicators: RSI EMA	62.0372 206.8453	50.5564 209.2611	…	

Stock price prediction (Part 4)

EveGPT

Group 6: Yang Fan (Blithe) Li Xinran (Anna) Zhu Jiarui (Janet) Fung Ho Kit (Justin)
Instructor: Dr. Matthias Buehlmaier

Task Construction

```
Full_data['TargetNextClose'] = Full_data['Close'].shift(-1)
Full_data['TargetNextClose'] = Full_data['Close'].shift(-1)
```

Input Variables

	score_title	score_summary	RSI	EMA	TargetNextClose
2020-08-31	0.568099	0.671994	0.872132	0.000000	0.166462
2020-09-01	0.611594	0.690165	0.763835	0.011589	0.135897
2020-09-02	0.428118	0.582304	0.652460	0.019925	0.091320
2020-09-03	0.487890	0.448165	0.518026	0.024070	0.103820
2020-09-04	0.491428	0.326013	0.542247	0.028972	0.006526
...	...	...	...	...	...
2023-04-05	0.623026	0.624790	0.385962	0.310083	0.254944
2023-04-06	0.483067	0.574451	0.383604	0.308917	0.253122
2023-04-10	0.491427	0.562193	0.380705	0.307677	0.260675
2023-04-11	0.464982	0.513799	0.393057	0.307136	0.239971
2023-04-12	0.560357	0.580121	0.359304	0.304933	0.257727

658 rows × 5 columns

Output Target

Data Preparation

```
X = []
backcandles = 10 # Look back period

print(data_set_scaled.shape[0])
for j in range(4):
    X.append([])
    for i in range(backcandles, data_set_scaled.shape[0]):
        X[j].append(data_set_scaled[i-backcandles:i, j])

#move axis from 0 to position 2
X=np.moveaxis(X, [0], [2])

X, yi =np.array(X), np.array(data_set_scaled[backcandles:,-1])
y=np.reshape(yi,(len(yi),1))

print(X.shape) #658 data, 10 days loop back, 6 indicators
print(y.shape) #Start from 10 days since need 10 to process data
```

Look back method

Considering past 10 days data to do the prediction

```
658
Shape of X: (648, 10, 4)
Shape of y: (648, 1)
```

648 Data Volume
10 days lookback
4 Indicators

Model Training

```
In [48]: indicators = 4
         lstm_input = Input(shape=(backcandles, indicators), name='lstm_input')
         inputs = LSTM(150, name='first_layer')(lstm_input)
         inputs = Dense(1, name='dense_layer')(inputs)
         output = Activation('linear', name='output')(inputs)
         model = Model(inputs=lstm_input, outputs=output)
         adam = optimizers.Adam()
         model.compile(optimizer=adam, loss='mse')
         history = model.fit(x=X_train, y=y_train, batch_size=15, epochs=200, shuffle=True
         history

executed in 41.2s, finished 03:43:34 2023-04-27

s: 0.0026
Epoch 197/200
32/32 [==============================] - 0s 7ms/step - loss: 0.0020 - val_los
s: 0.0028
Epoch 198/200
32/32 [==============================] - 0s 7ms/step - loss: 0.0019 - val_los
s: 0.0029
Epoch 199/200
32/32 [==============================] - 0s 7ms/step - loss: 0.0019 - val_los
s: 0.0026
Epoch 200/200
32/32 [==============================] - 0s 8ms/step - loss: 0.0018 - val_los
s: 0.0028
```

Model Setting

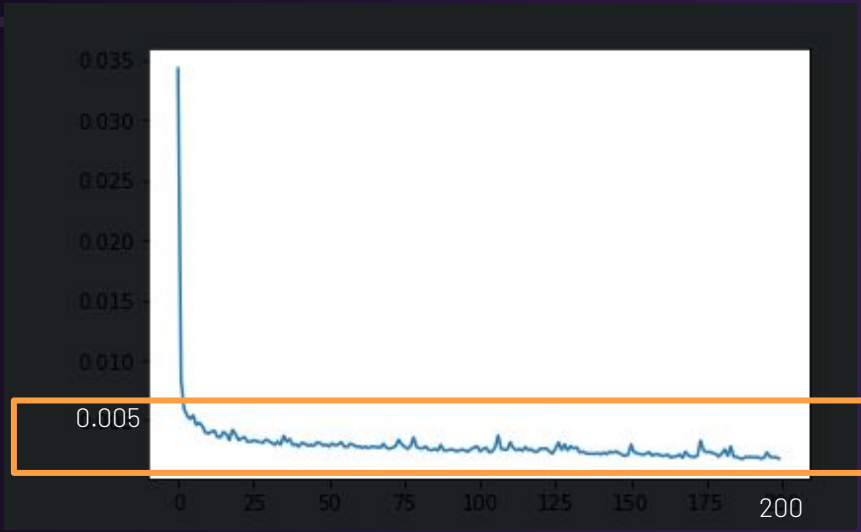
Prevent Overfitting

Model Evaluation - Loss

```
In [58]: loss_values = history.history['loss']

         # print out the loss values for each epoch
         for i in range(len(loss_values)):
             print("Epoch:", i + 1, "Loss:", loss_values[i])

         plt.plot(loss_values)
```



Loss below 0.005 and become stable

Model Demo & Storage

```
plt.figure(figsize=(16,8))
plt.plot(y_test, color = 'black', label = 'Test')
plt.plot(y_pred, color = 'green', label = 'pred')
plt.legend()
plt.show()
```

executed in 110ms, finished 03:53:35 2023-04-27


```
model.save('EveGPT')
```

```
loaded_model = keras.models.load_model('EveGPT')
loaded_model.summary()

y_pred = loaded_model.predict(X_test)
for i in range(10):
    print(y_pred[i], y_test[i])
```

executed in 1.14s, finished 03:54:44 2023-04-27

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_input (InputLayer)     [(None, 10, 6)]           0

 first_layer (LSTM)          (None, 150)               94200

 dense_layer (Dense)         (None, 1)                 151

 output (Activation)         (None, 1)                 0

=================================================================
Total params: 94,351
Trainable params: 94,351
Non-trainable params: 0
_________________________________________________________________
```

Web App Design

Instant Data retrieval

WebApp Setting
Html & CSS coding

```
##############################################
#
#              Setting
#
##############################################
```

```
def local_css(file_name):
    with open(file_name) as f:
        st.markdown(f"<style>{f.read()}</style>", unsafe_allow_html=True)
st.set_page_config(page_title="EveGPT", page_icon="🐲")
st.markdown('<div style="background-color:#F4D03F;padding:10px;border-radius:10px;">', unsaf
st.markdown("<h1 style='text-align: center; color: #FF9E44;'>EveGPT</h1>", unsafe_allow_html
Group = """
            <div style="background-color:#F4D03F;padding:10px;border-radius:10px;">
                <div style="color: black;">
                Group 6:
                Yang Fan (Blithe)
                Li Xinran (Anna)
                Zhu Jiarui (Janet)
                Fung Ho Kit (Justin)
                </br>
                Instructor: Dr. Matthias Buehlmaier
                </div>
            </div>
        """
st.markdown(Group, unsafe_allow_html=True)
local_css("style.css")
```

```
##############################################
#
#              Raw Data
#
##############################################
```

```
start = "2020-07-20"
today = date.today().strftime("%Y-%m-%d")
st.write(f"Today is {today}")

stocks = ["AAPL",
            "TSLA",
            "GOOG",
            "AMZN",
            "MSFT",
            "NVDA"]
selected_Stocks = st.selectbox("Select stock", stocks)
#n_years = st.slider("Year of prediction",1,4)
#period = n_years*365

def load_data(ticker):
    tickers = yf.Tickers(ticker)
    data = tickers.tickers[ticker].history(start=start)
    data.index = pd.to_datetime(data.index).date
    #data.reset_index(inplace = True)
    return data

data_load_state = st.text("load data...")
StockPrice = load_data(selected_Stocks)
data_load_state.text("load data...done!")

st.write("Raw data")
st.write(StockPrice.tail())

def plot_raw_data(Data):
    fig = go.Figure()
    fig.add_trace(go.Scatter(x=Data.index, y= Data["Open"], name = "stock_open"))
    fig.add_trace(go.Scatter(x=Data.index, y= Data["Close"], name = "stock_Close"))
    fig.layout.update(title_text="Time series data", xaxis_rangeslider_visible=True)
    st.plotly_chart(fig)
plot_raw_data(StockPrice)


st.markdown('<div style="background-color:#F4D03F;padding:10px;border-radius:10px;">', unsafe_allow_htm
```

Web App Design

Loading Scraping Results and Sentiment Scores

```
##############################################
#
#              Scraping
#
##############################################


def load_scrapped_text(ticker):
    Path = f"data/{ticker}.xlsx"
    scraped_text = pd.read_excel(Path).iloc[::-1].set_index("date")
    scraped_text.index = pd.to_datetime(scraped_text.index).date
    return scraped_text
def load_sentiment_score(ticker):
    Path = f"../Part3_FinBert/output/predictions_{ticker}.csv"
    Sentiment = pd.read_csv(Path).iloc[::-1]\
                            .set_index("date")\
                            .drop(['Unnamed: 0'],axis=1)
    #Sentiment.index = pd.to_datetime(Sentiment["date"]).date
    Sentiment.column = ["Scores for title", "Score for summary"]
    return Sentiment

data_load_state = st.text("load data...")
scraped_text = load_scrapped_text(selected_Stocks)
Sentiment = load_sentiment_score(selected_Stocks)
data_load_state.text("load data...done!")

st.write("Scraped Text")
st.write(scraped_text.tail(10))

st.write("Sentiment score retrival")
st.write(Sentiment.tail(10))
```

Loading Pretrained Model For Prediction Of Stock Price

```
##############################################
#
#              Run prediction
#
##############################################

company = selected_Stocks
Starting = "2020-07-20"
Ending = "2023-04-14"

tickers = yf.Tickers(company)
StockPrice = tickers.tickers[company].history(start=Starting,end=Ending)
StockPrice.index = pd.to_datetime(StockPrice.index).date

Path = f"../Part3_FinBert/output/predictions_{company}.csv"
Sentiment = pd.read_csv(Path).iloc[::-1]\
                            .reset_index()\
                            .drop(['Unnamed: 0', "index"],axis=1)

Sentiment = Sentiment[Sentiment["date"] >= Starting]
Sentiment['date'] = pd.to_datetime(Sentiment['date'])
Sentiment.set_index('date', inplace=True)
Sentiment.index = pd.to_datetime(Sentiment.index).date

missing_rows_index = StockPrice.index.difference(Sentiment.index)
new_data = {'score_title': 0, 'score_summary': 0}
new_row = pd.DataFrame(new_data, index=missing_rows_index)
Sentiment = pd.concat([new_row,Sentiment],axis=0,sort=False)
index_to_keep = pd.Series(True, index=StockPrice.index)
mask = Sentiment.index.isin(index_to_keep.index)
Sentiment = Sentiment[mask]
Full_data = pd.concat([StockPrice,Sentiment],axis=1)

Full_data['RSI']=ta.rsi(Full_data.Close, length=30)
Full_data['EMAF']=ta.ema(Full_data.Close, length=30)
Full_data['EMAM']=ta.ema(Full_data.Close, length=30)
Full_data['EMAS']=ta.ema(Full_data.Close, length=30)

Full_data['TargetNextClose'] = Full_data['Close'].shift(-1)
Full_data['TargetNextClose'] = Full_data['Close'].shift(-1)
Full_data.iloc[:, 0:-1].dropna(inplace=True)
Full_data.drop(['Volume'], axis=1, inplace=True)
data_set = Full_data.iloc[30:-1, 6:12]
st.write(data_set)

sc = MinMaxScaler(feature_range=(0,1))
data_set_scaled = sc.fit_transform(data_set)
X = []
backcandles = 10 # Look back period

print(data_set_scaled.shape[0])
for j in range(6):
    X.append([])
    for i in range(backcandles, data_set_scaled.shape[0]):
        X[j].append(data_set_scaled[i-backcandles:i, j])

#move axis from 0 to position 2
X=np.moveaxis(X, [0], [2])
print(X.shape)

loaded_model = keras.models.load_model('EveGPT')
loaded_model.summary()
y_pred = loaded_model.predict(X)

y = Full_data.iloc[40:-1, 12:]
print(y)
sc = MinMaxScaler(feature_range=(0,1))
y_test = sc.fit_transform(y)

def plot_result():
    fig = go.Figure()
    fig.add_trace(go.Scatter(x = y.index,y = y_test.ravel(), name = 'original', line=dict(color='white')))
    fig.add_trace(go.Scatter(x = y.index,y = y_pred.ravel(), name = 'predition',  line=dict(color='#FF0F4A')))
    fig.layout.update(title_text="Prediction", xaxis_rangeslider_visible=True)
    st.plotly_chart(fig)
plot_result()
```

Web App Demo

240 lines
to build the web app

https://justinfungi-fina4350-nlp-part4-evegptwebapp-fxtphb.streamlit.app/

Thank you!