**CE314**

**Web Scraper AI**

**Name:** Leong Jia Juin, Benjamin

**Kaplan ID:** CT0365371

**PRID:** LEONG41406

**Use Case**

A possible use-case for a web-scraper AI would be a program that scrapes financial news headlines and performs sentiment analysis to uncover the underlying sentiments in the market. Financial markets are highly influenced by news and events. Making and accurate analysis of financial headlines is crucial for investors and traders. Machine learning techniques can extract valuable insights from vast amounts of textual data.

The program will first leverage web scraping techniques to collect financial headlines from various sources, such as news websites, financial blogs, and social media platforms. The headlines are then preprocessed such as to removing punctuation, stop words, and irrelevant information. Next, the headlines can be transformed into numerical representations, using natural language processing (NLP) techniques. Finally, a trained model will be employed to predict the sentiments of the financial headlines.

To capture the sentiments expressed in the headlines, the program employs a machine learning model trained specifically for sentiment analysis. Various machine learning models, such as support vector machines (SVM) or recurrent neural networks (RNN) can be employed for this task. The model is trained using a labeled dataset, where financial headlines are annotated with sentiment labels (e.g., positive, negative, or neutral). The program utilizes a combination of supervised learning and feature engineering to enhance the model's predictive accuracy. Once the sentiment analysis model is trained, it is applied to the collected financial headlines to classify their sentiments.

Overall, this web scraping AI can potentially be a powerful tool for investors to gain valuable insights into market sentiments, enabling them to make informed decisions based on sentiment trends and themes.

**Proposed Wireframe**

Financial News Website

Web Scraper

Text Processing (Converting Words to Vectors)

Labelled Financial News Dataset

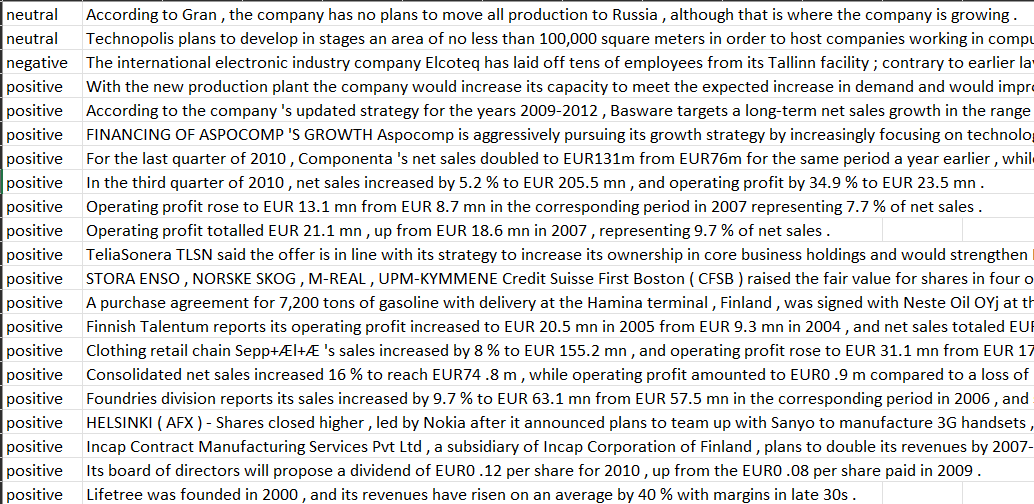
LSTM Model Training

LSTM Sentiment Classifier

Sentiment Output

1. **Financial News Website:** This is the source from which the financial news headlines will be scraped from. A specific website that provide financial news will be chosen.
2. **Web Scraper:** This module will be responsible for scraping the headlines from the financial news website. It will fetch the HTML content, parse it, and extract the relevant headlines. The extracted headlines may contain noise or unwanted elements. This web scraper will perform necessary preprocessing tasks such as removing HTML tags, special characters, or stop words. It will also convert the text into a suitable format for the LSTM model.
3. **LSTM Sentiment Classifier:** An LSTM model is employed to classify the sentiment of each headline. It will take preprocessed headlines as input and output the sentiment classification.
4. **Sentiment Output:** This represents the final output of the system, where the sentiment (positive, negative, neutral) for each headline will be displayed or stored for further analysis.

**Pseudo Code**

**Financial Dataset (Snapshot) -** This is the dataset that will be used to train the LSTM Classifier

**LSTM Model Training**

1. **Import Dependencies**

import numpy as np

import pandas as pd

…

from keras.layers import LSTM, Dense, Embedding, Dropout

from keras.preprocessing.text import Tokenizer

from keras\_preprocessing.sequence import pad\_sequences

1. **Load Financial Dataset**

df = pd.read\_csv('Financial\_News\_Dataset.csv', encoding = "ISO-8859-1")

1. **Dataset Preprocessing**

df['news'] = df['news'].apply(str.lower)

tokenizer = Tokenizer(num\_words=5000, split=" ")

tokenizer.fit\_on\_texts(df['news'].values)

X = pad\_sequences(X)

1. **Train Test Split**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.2)

1. **Build LSTM Model**

model = Sequential()

model.add(Embedding(5000, 256, input\_length = X.shape[1]))

1. **Train the Model**

model.fit(X\_train, y\_train, epochs = 10, batch\_size = 32, verbose=2)

model.save(r'SentimentModel.h5')

1. **Test the Model**

model = load\_model(r'SentimentModel.h5')

predictions = model.predict(X\_test)

**Web Scraper**

1. **Import Dependencies**

from urllib.request import urlopen, Request

from bs4 import BeautifulSoup

import pandas as pd

1. **Load LSTM model**

model = load\_model(r'SentimentModel.h5')

1. **Define website to scrape from**

finviz\_url = 'https://finviz.com/quote.ashx?t='

tickers = ['AAPL']

1. **Scrape news headlines**

news\_tables = {}

for ticker in tickers:

url = finviz\_url + ticker

req = Request(url=url, headers={'user-agent': 'my-app'})

response = urlopen(req)

html = BeautifulSoup(response, features='html.parser')

1. **Parse headlines into dataframe**

parsed\_data = []

for ticker, news\_table in news\_tables.items():

for row in news\_table.findAll('tr'):

title = row.a.text

1. **Process text to numerical vectors**

df = df['title'].apply(str.lower)

df = df.apply(lambda x: re.sub('[^a-zA-Z0-9\s]',"",x))

tokenizer = Tokenizer(num\_words=5000, split=" ")

tokenizer.fit\_on\_texts(df.values)

1. **Pad the vectors to fit the model**

padding\_size = 50 - len(X[0])

padding\_vector = np.zeros(padding\_size)

1. **Convert prediction back to text format**

def format\_predictions(predictions)

1. **Output a .csv file**

combined\_df.to\_csv('sentiments.csv', index=False)

**Functional Code**

1. Run ‘SentimentAnalysis\_TrainModel.py’ to train the model (Please note that model has been trained and saved under the file ‘SentimentModel.h5’)
2. Run ‘WebScraper.py’ to scrape financial news headlines and analyze sentiments

A screen shot of a computer

Description automatically generated with low confidence

You may edit the tickers to scrape your choice of stock news

1. The output of WebScraper.py is a .csv file with the headlines and sentiments.