Company Perception Tool Using Sentiment Analysis

Team Name - Intelligent Analyzers

Team Members:

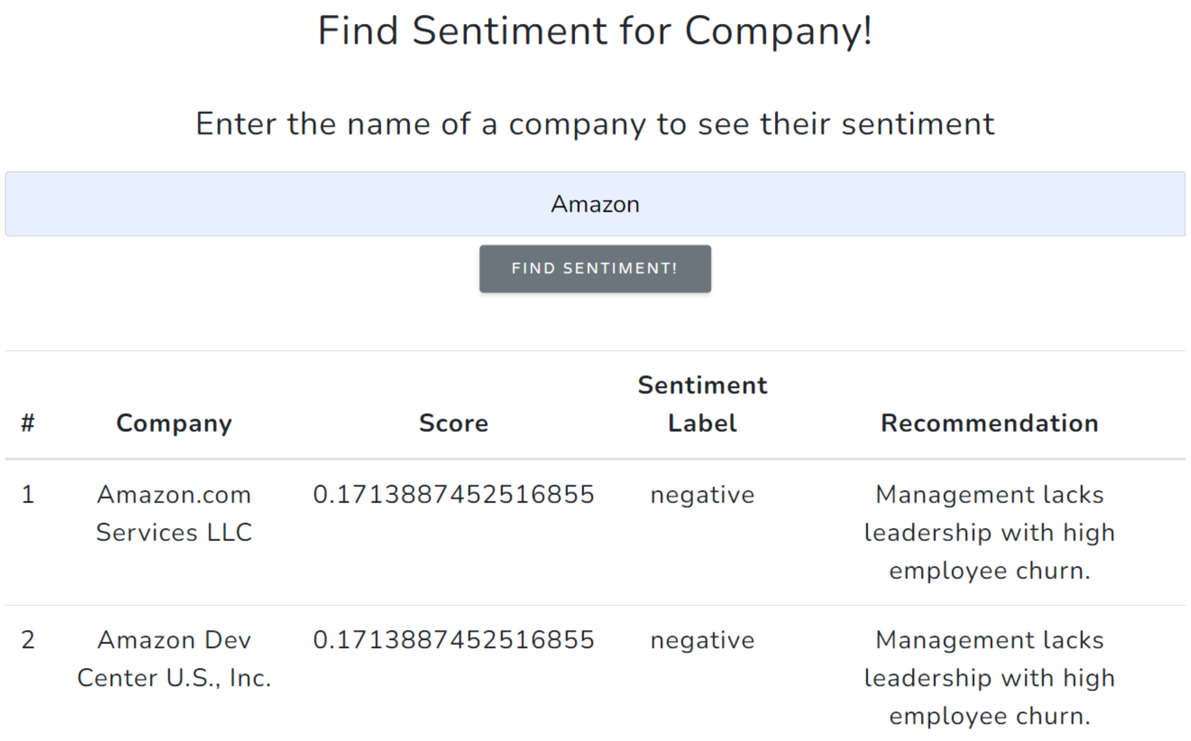
- Atharv Chandratre - atharvc2@illinois.edu (Captain)
- Uday Kanth Reddy Kakarla - uk3@illinois.edu
- Priyanka Awatramani - pma7@illinois.edu
- Ansh Bilimoria - amb20@illinois.edu

Table of Contents

Documentation

Overview Of The Function Of The Code

Sentiment analysis (or opinion mining) is a natural language processing (NLP) technique used to determine whether data is positive, negative, or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. We scrape the company reviews' data using our written scraper and then pre-process the data to tokenize the data and remove unnecessary tokens like stop words, punctuations, etc. We use NLTK (Natural Language Toolkit), a Python-based Natural Language Processing library for these tasks. We then use sentiment analysis on this data to help estimate employee satisfaction. We are using 'emotion detection' sentiment analysis, allowing us to go beyond polarity (i.e. very positive, positive, neutral, negative, very negative), to detect emotions, like happiness, frustration, anger, and sadness. Many emotion detection systems use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms. We would be using the lexicon-based approach. We even developed a UI where a user can input a company name and we provide the score, positive/negative sentiment, and some recommendations on how to improve that satisfaction score.


Find Sentiment for Company!
Enter the name of a company to see their sentiment
Amazon
FIND SENTIMENT!

#	Company	Score	Sentiment Label	Recommendation
1	Amazon.com Services LLC	0.1713887452516855	negative	Management lacks leadership with high employee churn.
2	Amazon Dev Center U.S., Inc.	0.1713887452516855	negative	Management lacks leadership with high employee churn.

In today's time, employee satisfaction is of utmost importance. Having a satisfied and happy workforce strengthens the company by lowering employee turnover, increasing employee productivity, increasing customer satisfaction, and promoting loyalty, eventually leading to higher profits for the company. Our tool would help companies apply sentiment analysis on multiple forums and find out their employee satisfaction ratio. Based on the results, the tool also recommends some ideas to the managers like planning a team outing/dinner, etc. This would eventually lead to higher employee satisfaction and increased productivity.

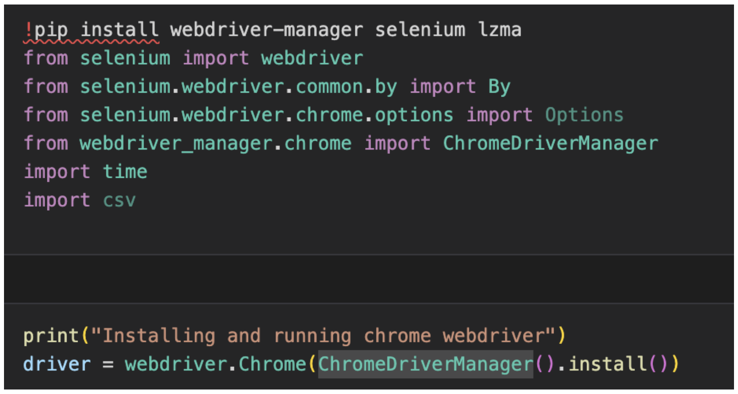
How The Software Is Implemented

The Indeed.com Web Scraper

The web scraper uses Selenium and Chrome Web Driver in order to do the following things:

1. The scraper goes to Indeed.com and scrapes the URLs of the first 150 companies it encounters. 150 has been set as a configurable parameter within the web scraping script and can be increased or decreased as per the user's requirements. More information about this configurable parameter can be found in the Software Usage section.
2. After the scraper has collected the 150 company URLs, it then proceeds to go to each URL and then scrape the first 100 reviews in text form from the website's review URL. Here, 100 is also a configurable parameter that can be modified according to the user's choice.
3. The scraped data is then saved to a csv file, which the sentiment analyzer can use for performing the remaining functions.

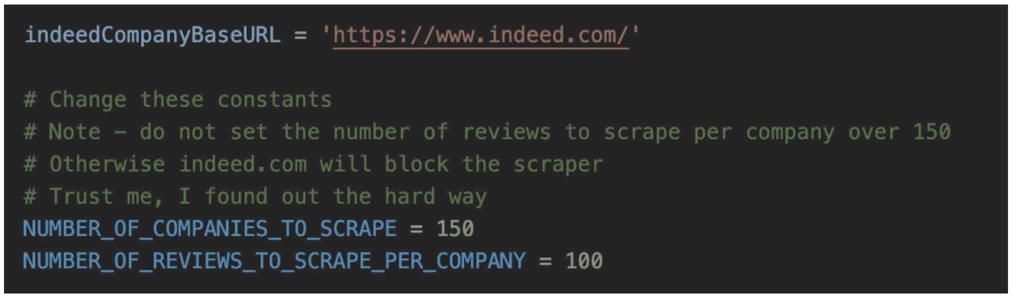
The code is explained below:



```
!pip install webdriver-manager selenium lzma
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.options import Options
from webdriver_manager.chrome import ChromeDriverManager
import time
import csv
```

```
print("Installing and running chrome webdriver")
driver = webdriver.Chrome(ChromeDriverManager().install())
```

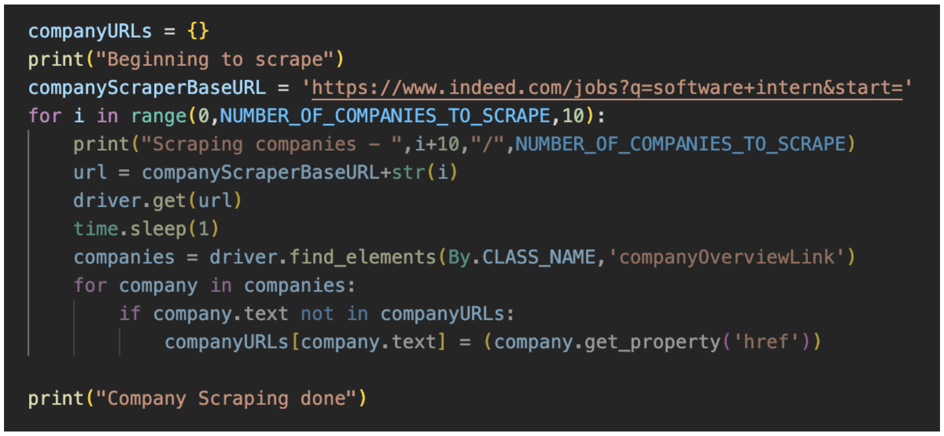
These lines first install webdriver-manager, selenium and lzma, the three libraries needed by the scraper to work, and imports the functionality needed from each of those libraries. Then, it installs and runs Chrome Web Driver, which we will use to access the Indeed website.



```python
indeedCompanyBaseURL = 'https://www.indeed.com/'

# Change these constants
# Note - do not set the number of reviews to scrape per company over 150
# Otherwise indeed.com will block the scraper
# Trust me, I found out the hard way
NUMBER_OF_COMPANIES_TO_SCRAPE = 150
NUMBER_OF_REVIEWS_TO_SCRAPE_PER_COMPANY = 100
```

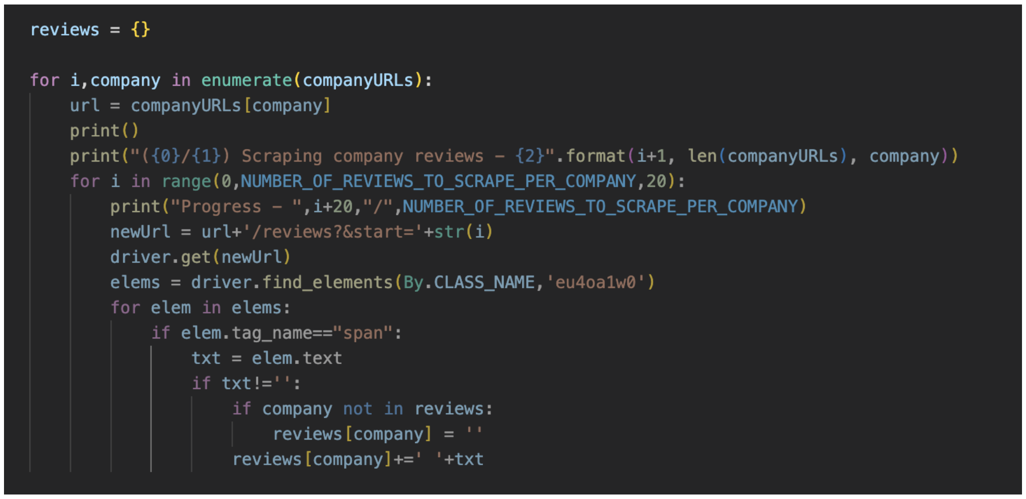
These lines set up the configurable variables, for the number of companies to scrape and the number of reviews to scrape per company. Please refer to the warning given in the comments while setting the values of these variables.



```python
companyURLs = {}
print("Beginning to scrape")
companyScraperBaseURL = 'https://www.indeed.com/jobs?q=software+intern&start='
for i in range(0,NUMBER_OF_COMPANIES_TO_SCRAPE,10):
    print("Scraping companies - ",i+10,"/",NUMBER_OF_COMPANIES_TO_SCRAPE)
    url = companyScraperBaseURL+str(i)
    driver.get(url)
    time.sleep(1)
    companies = driver.find_elements(By.CLASS_NAME,'companyOverviewLink')
    for company in companies:
        if company.text not in companyURLs:
            companyURLs[company.text] = (company.get_property('href'))

print("Company Scraping done")
```

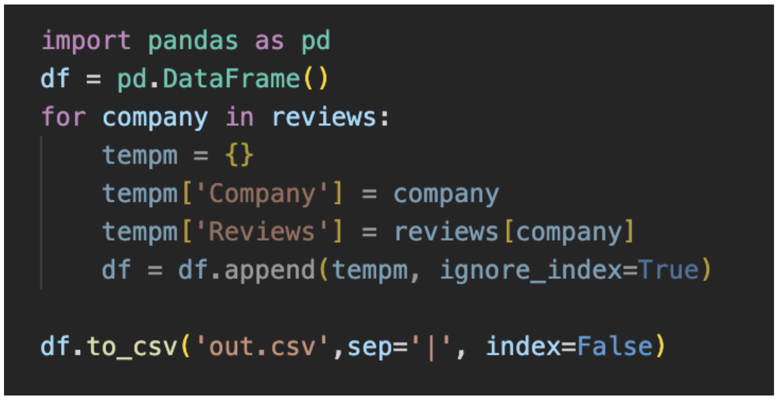
These lines scrape the company URLs for each company. It does this by first searching for a job (in our case, “software intern”), and then traversing through each page of the results. While doing so, it stores the URL of each company it displayed on the website among the search results. This is done by accessing the CSS selector containing this information.



```python
reviews = {}

for i,company in enumerate(companyURLs):
    url = companyURLs[company]
    print()
    print("({0}/{1}) Scraping company reviews - {2}".format(i+1, len(companyURLs), company))
    for i in range(0,NUMBER_OF_REVIEWS_TO_SCRAPE_PER_COMPANY,20):
        print("Progress - ",i+20,"/",NUMBER_OF_REVIEWS_TO_SCRAPE_PER_COMPANY)
        newUrl = url+'/reviews?&start='+str(i)
        driver.get(newUrl)
        elems = driver.find_elements(By.CLASS_NAME,'eu4oa1w0')
        for elem in elems:
            if elem.tag_name=="span":
                txt = elem.text
                if txt!='':
                    if company not in reviews:
                        reviews[company] = ''
                    reviews[company]+=' '+txt
```

After getting the company URLs, it accesses each website's URL and then goes to the reviews page. There, it goes through the reviews up to the limit specified by the user, and stores each scraped review in an array. This is done by accessing the CSS selector containing this information, just like we did for the company URLs. Once each company's reviews have been scraped, they are added to a dictionary called reviews.



```python
import pandas as pd
df = pd.DataFrame()
for company in reviews:
    tempm = {}
    tempm['Company'] = company
    tempm['Reviews'] = reviews[company]
    df = df.append(tempm, ignore_index=True)

df.to_csv('out.csv',sep='|', index=False)
```

Finally, the dictionary called reviews is converted into a data frame using Pandas. This data frame is exported as a CSV file, with the '|' character as the separator. The reason we chose to use that character as the separator (as opposed to the comma used in CSV files), is that many reviews contained commas in them, leading to the CSV file being unable to store the data in the format we wanted it to be stored in. Once the data has been stored in the CSV file, the Sentiment Analyzer can perform further operations on it.

Sentiment Analyzer

We have implemented two approaches for performing the sentiment analysis for the reviews obtained from Indeed. The first approach is the Lexicon-Based Approach and the second approach is the Machine Learning Based Approach. Before performing these steps, we preprocess the data.

1. Preprocessing the data

- Reading the dataset
- Converting all text to lowercase and removing punctuations

```
data = pd.read_csv('dataset/dataset.csv', sep='|')
data.Reviews=data.Reviews.astype(str)

#Transform text to lowercase
data['Reviews'] = data['Reviews'].apply(lambda x: x.lower())
#Removing all punctuations and special characters
data['Reviews'] = data['Reviews'].apply(lambda x: re.sub('[,.]', '', x))
```

- Removing all stopwords from the data. These stop words are received from the ntlk package by using the command nltk.download('stopwords')

```
stopwords_list = stopwords.words('english')
data['Reviews'] = data['Reviews'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stopwords_list)]))
```

2. Sentiment Analysis using Textblob - a lexicon based approach

```
score_list=[]

for text in data['Reviews']:

  sentence = TextBlob(text)

  score = sentence.sentiment.polarity
  score_list.append(score)
```

Textblob has an inbuilt dictionary which calculates the sentiment of an input that is given to it. Thus, it calculates the polarity of a sentence which ranges from -1 to +1. Negative values indicate that the sentiment is negative and on the other hand, positive values indicate that the sentiment is positive.

```
sentiment_label = []
for score in score_list:
  if score > median:
    sentiment_label.append('positive')
  elif score < median:
    sentiment_label.append('negative')
  else:
    sentiment_label.append('neutral')
print(sentiment_label)
```

The scores are then categorized into three labels - positive, negative and neutral.

```
dataset = data
dataset['SCORE']=score_list
dataset['SENTIMENT_LABEL']=sentiment_label
print(dataset)
dataset.to_csv("../dataset/TextBlobResultswithMedian.csv", sep='|')
```

The results are added to a CSV file with the respective scores and sentiment labels for each company.

3. Training the Machine Learning Models

We trained the algorithm based on labeled data which we labeled manually. Following is how the labeled data looks like-

```
data.head()
```

	Column1	Company	Reviews	HUMAN_LABEL
0	0	Netflix	high performance culture challenge netflix ask...	positive
1	1	Uber	pay gone years first started pay great gas pri...	negative
2	2	Yext	excellent ceo lucky worked given opportunity f...	negative
3	3	CACI	nice place work great company work benefits gr...	negative
4	4	Covenant Eyes	best company ive ever worked covenant eyes car...	positive

We select the particular column, i.e here we select Reviews as the data we want to train. Here, we split the dataset into two parts - the training set and the testing set. The training set consists of 75% of the data while the testing set is the other 25%.

```
X = data.iloc[:, 2]
y = data.iloc[:, -1]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 42)
```

Following shows how the training data now looks-

```
X_train
```

```
15     low paced environment day consisted learning c...
72     good place work fun caring people guess starte...
22     good wlb terrible culture never work later sch...
116    overall great place intern first internship co...
91     fun workplace ive worked almost two years hone...
                              ...
106    good people challenges sort great product visi...
14     excellent employer microsoft excellent employe...
92     great place work great worklife balance welcom...
51     flexible flexible hours management help sucked...
102    disconnected benefits mediocre high turnaround...
Name: Reviews, Length: 91, dtype: object
```

Following shows how the testing data now looks-

```
X_test
```

```
18     productive fun place work would definitely rec...
45     good work ethics working help client stay clea...
47     productive strong work environment good workli...
89     much game culture dedicated product ip making ...
4      best company ive ever worked covenant eyes car...
40     good place work people passionate products tak...
62     great place work great place work never job tr...
107    small company feel good people small company f...
31     overall great place work overall great place w...
55     okay often left manage difficult accounts cand...
53     feel well cared working gdms nature work fulfi...
119    best place work managers jobs correctly always...
10     productive fun environment overall kasisto fun...
90     electrical repair technician great place work ...
109    great wonderful work environment everyone real...
11     productive fun workplace learned new things co...
76     good pay terrible management management terrib...
56     fast moving environment clear mission great pl...
115    best job ever working apple great experience w...
0      high performance culture challenge netflix ask...
26     good place overall treat employees well greet ...
44     2 stars best part working company good lower l...
66     great experience work fun would definitely wor...
98     good job money good good job time money good c...
24     productive fun place amazing culture innovatio...
42     ok place work good coworkers schedule place di...
105    pays extremely well pay almost unbeatable skil...
93     intern experience overall lot learning opportu...
36     great company salary good great benefits oppor...
100    productive workplace fun table tennis outdoor ...
12     good place work good perks employee engagement...
Name: Reviews, dtype: object
```

Here, we use the TfidfVectorizer for better accuracy of the ML models by using sublinear transformation to not give very high frequency to a word which occurs frequently as well as IDF weighting to reward rare words seen in the dataset.

```python
from sklearn.feature_extraction import text
from sklearn.feature_extraction.text import TfidfVectorizer

my_stop_words = text.ENGLISH_STOP_WORDS
# Create feature vectors
vectorizer = TfidfVectorizer(min_df=5,
                             max_df=0.95,
                             stop_words=my_stop_words,
                             sublinear_tf = True,
                             use_idf = True,
                             ngram_range=(1,1))
train_vectors = vectorizer.fit_transform(X_train.values.astype('U'))
test_vectors = vectorizer.transform(X_test.values.astype('U'))
```

4. Random Forest - the first machine learning algorithm

Random Forest is a simple classifier that uses decision trees. Thus, we use the RandomForestClassifier() function which fits the training sets. We have calculated the training and prediction time that Random Forest took. The output_dict parameter is a boolean function which returns the output if it is set to true. The prediction parameter returns the estimated targets returned by the classifier.

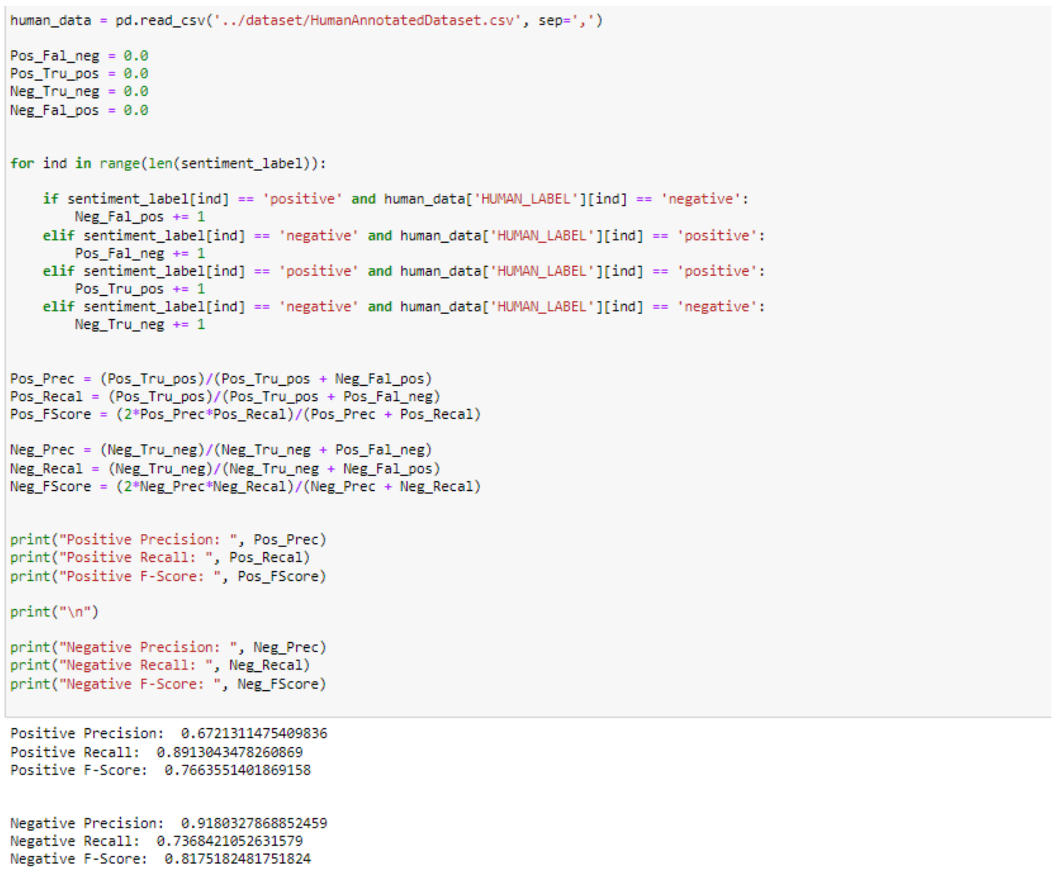
```python
classifier = RandomForestClassifier()
t0 = time.time()
classifier.fit(train_vectors, y_train)
t1 = time.time()
prediction = classifier.predict(test_vectors)
t2 = time.time()
time_linear_train = t1-t0
time_linear_predict = t2-t1
# results
print("Training time: %fs; Prediction time: %fs" % (time_linear_train, time_linear_predict))
report = classification_report(y_test, prediction, output_dict=True,labels=np.unique(prediction))
print('positive: ', report['positive'])
print('negative: ', report['negative'])
```

5. K-Nearest Neighbours - the second machine learning algorithm

The KNN algorithm considers the k nearest training samples in the data set. We have imported the KNeighborsClassifier() that fits the training vectors. As mentioned above, the output_dict parameter is a boolean function which returns the output if it is set to true. The prediction parameter returns the estimated targets returned by the classifier.

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier()
t0 = time.time()
classifier.fit(train_vectors, y_train)
t1 = time.time()
prediction = classifier.predict(test_vectors)
t2 = time.time()
time_linear_train = t1-t0
time_linear_predict = t2-t1
# results
print("Training time: %fs; Prediction time: %fs" % (time_linear_train, time_linear_predict))
report = classification_report(y_test, prediction, output_dict=True)
print('positive: ', report['positive'])
print('negative: ', report['negative'])
```

Further, we calculated the Precision, Recall, classification accuracy and F-Score in order to evaluate the performance of our models and compare it across models. In order to do this we had to assign sentiment labels by going through the reviews of various companies. This is because the data that we scraped is unlabelled data and for the calculation of these scores we need labeled data. Below is the code used for calculating these measure for the case of TextBlob:



```
human_data = pd.read_csv('../dataset/HumanAnnotatedDataset.csv', sep=',')

Pos_Fal_neg = 0.0
Pos_Tru_pos = 0.0
Neg_Tru_neg = 0.0
Neg_Fal_pos = 0.0


for ind in range(len(sentiment_label)):

    if sentiment_label[ind] == 'positive' and human_data['HUMAN_LABEL'][ind] == 'negative':
        Neg_Fal_pos += 1
    elif sentiment_label[ind] == 'negative' and human_data['HUMAN_LABEL'][ind] == 'positive':
        Pos_Fal_neg += 1
    elif sentiment_label[ind] == 'positive' and human_data['HUMAN_LABEL'][ind] == 'positive':
        Pos_Tru_pos += 1
    elif sentiment_label[ind] == 'negative' and human_data['HUMAN_LABEL'][ind] == 'negative':
        Neg_Tru_neg += 1


Pos_Prec = (Pos_Tru_pos)/(Pos_Tru_pos + Neg_Fal_pos)
Pos_Recal = (Pos_Tru_pos)/(Pos_Tru_pos + Pos_Fal_neg)
Pos_FScore = (2*Pos_Prec*Pos_Recal)/(Pos_Prec + Pos_Recal)

Neg_Prec = (Neg_Tru_neg)/(Neg_Tru_neg + Pos_Fal_neg)
Neg_Recal = (Neg_Tru_neg)/(Neg_Tru_neg + Neg_Fal_pos)
Neg_FScore = (2*Neg_Prec*Neg_Recal)/(Neg_Prec + Neg_Recal)


print("Positive Precision: ", Pos_Prec)
print("Positive Recall: ", Pos_Recal)
print("Positive F-Score: ", Pos_FScore)

print("\n")

print("Negative Precision: ", Neg_Prec)
print("Negative Recall: ", Neg_Recal)
print("Negative F-Score: ", Neg_FScore)
```

```
Positive Precision:  0.6721311475409836
Positive Recall:  0.8913043478260869
Positive F-Score:  0.7663551401869158


Negative Precision:  0.9180327868852459
Negative Recall:  0.7368421052631579
Negative F-Score:  0.8175182481751824
```

Below is a table with a comparison of all the above approaches -

	Precision	**Recall**	**F1**
Textblob (Positive)	0.67	0.89	0.77
Textblob (Negative)	0.91	0.74	0.81
Random Forest (Positive)	0.82	0.69	0.75
Random Forest (Negative)	0.80	0.89	0.84
KNN (Positive)	0.70	0.54	0.61
KNN (Negative)	0.71	0.83	0.77

The precision of Random Forest and KNN appears to be higher than that of Textblob for the positive case. But the Precision of Textblob appears to be the highest for the negative case.
However, this is not to be confused that these models are better than the lexicon based approach just on the inference of positive cases. This is because these traditional ML models are first trained and are then tested, whereas in the case of Textblob it uses NLP and doesn't need any training phase. Further, these traditional ML models might not perform well for new test dataset or as good as the Text blob. This is evident by the fact that the Textblob F-score is comparable to the Random Forest and KNN F-score.

Recommendation System

The recommendation system is a simple web app written in HTML, CSS and JavaScript. It has three major files:

1. Index.html - contains the main scaffolding of the web app, and is used to invoke the javascript for interactivity.
2. Styles.css - contains the stylesheet used for the web app, used to style the components of the webpage like the button, and ensuring the layout and alignment of various components on the webpage.
3. App.js - This file contains the code written to search through the csv file created in the two steps above. The function in the code will return the sentiment values and the recommendations for the company searched for. It is written using d3.js to open and parse the csv file. Then, it performs some user input validation, like making sure that the string the user has entered to search for is greater than 3 characters. Once these checks pass, it searches the csv file for the company, and if found, returns the sentiment value and the recommendation for it. If not, it returns a message saying that the company was not found.

Usage Of The Software

You will need Python 3 in order to run the software. To install the requirements, run the following command in your terminal:

pip install webdriver-manager selenium lzma numpy pandas nltk scikit-learn textblob statistics

Then you can run the “Sentiment Analysis using Textblob with PreProcessing.ipynb” and “ML_Based_Approach.ipynb” notebooks for running the respective Sentiment Analyzers.

Scraper Usage:

1. Change the configuration variables which govern the number of companies the scraper will scrape for, and the number of reviews for each company which the scraper will extract. This can be found in the third cell of the Jupyter Notebook. Note - do not set the number of reviews to scrape per company to over 150. Otherwise, indeed.com will block the scraper (by blocking your IP address for a cooldown period).
2. Run all the cells in the Jupyter Notebook. There should be a run all cells button at the top of the IDE.
3. Once all the cells have run, it will create your requested dataset called ‘out.csv’, which will be in your present working directory.

Sentiment Analyzer Usage:

1. Textblob is required to run the sentiment analyzer. For this, you need to run -
   pip3 install textblob
2. You might also need to install some other required packages that are mentioned in the .ipynb file.
3. After all dependencies are correctly installed, run every block of the Python notebook for both the lexicon-based as well as the machine learning models.

Recommendation System Usage:

1. For running the web application which includes the recommendation system please install the live-server extension in VS code. This is demonstrated in the video. If you are using live-server, once you install the extension and open app.js as shown in the demo, you will have an option which says to “Go Live” at the bottom right of your VS code editor.

2. Once you click Go Live the application is deployed and you can enter the name of the company of which you want to find the Sentiment score, label and the recommendation for that company.

Contribution

Overall there was an equitable contribution by all members of the team. The following table breaks down the contributions of each member.

Team Member	Contribution
Atharv	Created the data scraping framework to extract the reviews from Indeed.com using Webdriver, data purification, and made the video.
Priyanka	Wrote how to perform the sentiment analysis using the lexicon-based approach and machine learning-based approaches
Uday	Worked on the data pre-processing for the sentiment analyzers to use and on a portion of the sentiment analysis using lexicon-based approach. Further, developed the website and worked on half of the Recommendation system.
Ansh	Worked on the remaining part of the Recommendation system and did human labeling of the dataset to perform statistical analysis.