Text Analysis - Companies description

In this notebook, we will do some text analysis for the companies existing in the WRDS and Harvard Business School, Impact-Weighted Accounts Project report.

We will apply some Nature Language Processing (NLP) using the pre-trained DistilBERT. First, we need to merge the datasets and obtain the companies description from Yahoo Finance.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import warnings
from sklearn import linear_model
from sklearn import metrics
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import requests
from bs4 import BeautifulSoup
warnings.filterwarnings('ignore')

## Useful for the DistilBERT model
import torch
import transformers as ppb # pytorch transformers
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
```

In [2]:

```
df_ishares = pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Da
df_ei = pd.read_csv('/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Dataset
stocks = pd.read_csv("/Users/maralinetorres/Documents/GitHub/Predicting-Environmental-and-Social-Actions/Datase
```

For the ishares, we don't need the tickers in the sector 'Cash and devirates' and we are also going to filter by the necessary columns

In [3]:

```
df_ishares = df_ishares.loc[df_ishares.Sector != 'Cash and/or Derivatives',['Ticker','Name','Sector','CUSIP','I
df_ishares.drop_duplicates(inplace=True)
df_ishares.head()
```

Out[3]:

	Ticker	Name	Sector	CUSIP	ISIN	Location
0	AAPL	APPLE INC	Information Technology	37833100	US0378331005	United States
1	MSFT	MICROSOFT CORP	Information Technology	594918104	US5949181045	United States
2	AMZN	AMAZON COM INC	Consumer Discretionary	23135106	US0231351067	United States
3	TSLA	TESLA INC	Consumer Discretionary	88160R101	US88160R1014	United States
4	FB	FACEBOOK CLASS A INC	Communication	30303M102	US30303M1027	United States

In [4]:

```
len(df_ishares.Ticker.unique())
```

Out[4]:

```
855
```

In [5]:

```
df_ei2 = df_ei.copy()
df_ei2 = df_ei2.loc[:,['ISIN','CompanyName','Country']]
df_ei2.drop_duplicates(inplace=True)
df_ei2.head()
```

Out[5]:

	ISIN	CompanyName	Country
0	DE0005545503	1&1 DRILLISCH AG	Germany
1	GB00B1YW4409	3I GROUP PLC	United Kingdom
4	US88579Y1010	3M COMPANY	United States
14	KYG8875G1029	3SBIO INC	China
17	GI000A0F6407	888 HOLDINGS PLC	Gibraltar

In [6]:

```
len(df_ei2.ISIN.unique())
```

Out[6]:
```
2628
```

In [7]:
```
stocks = stocks.iloc[:,0:2]
stocks.head()
```

Out[7]:

	Year	Ticker
0	2005	AEE
1	2006	AEE
2	2007	AEE
3	2008	AEE
4	2009	AEE

Let's start merging the datasets. First, we are going to merge iShares and Environmental impact by ISIN.

In [8]:
```
df = pd.merge(df_ishares, df_ei2, on='ISIN', how='outer')
df.head()
```

Out[8]:

	Ticker	Name	Sector	CUSIP	ISIN	Location	CompanyName	Country
0	AAPL	APPLE INC	Information Technology	37833100	US0378331005	United States	NaN	NaN
1	MSFT	MICROSOFT CORP	Information Technology	594918104	US5949181045	United States	MICROSOFT CORPORATION	United States
2	AMZN	AMAZON COM INC	Consumer Discretionary	23135106	US0231351067	United States	AMAZON.COM, INC.	United States
3	TSLA	TESLA INC	Consumer Discretionary	88160R101	US88160R1014	United States	NaN	NaN
4	FB	FACEBOOK CLASS A INC	Communication	30303M102	US30303M1027	United States	FACEBOOK INCORPORATION	United States

In [9]:
```
print(df.shape)
print(len(df.Ticker.unique()))
```

```
(3005, 8)
856
```

In [10]:
```
mising_Ticker = df.loc[df.Ticker.isna(),'ISIN']
df_missing = df_ei.loc[df_ei.ISIN.isin(mising_Ticker), ['ISIN','CompanyName','Country']]
df_missing.drop_duplicates(inplace=True)
print(f'We are unable to match {df_missing.shape[0]} ISIN')
df_missing.head()
```

```
We are unable to match 2136 ISIN
```

Out[10]:

	ISIN	CompanyName	Country
0	DE0005545503	1&1 DRILLISCH AG	Germany
14	KYG8875G1029	3SBIO INC	China
17	GI000A0F6407	888 HOLDINGS PLC	Gibraltar
20	PHY000221069	A Brown Company, Inc.	Philippines
21	GB00B6XZKY75	A.G. BARR P.L.C.	United Kingdom

With this merge, we were able to find a 492 tickers for the 2,628 companies in the Environmental Impact dataset. We need to figure out how to map their ISIN to a Ticker.

In [11]:
```
df_missing.to_csv('Companies_missing_Tickers.csv', index=False)
```

In the meantime, we are going to export the companies that we were unable to match and continue working with the ones that have the Ticker.

From the pilot stocks, we have 52 unique tickers. Let's see if all these tickers appear in the merged dataframe.

In [12]:
```
tickers = stocks.Ticker.unique()
x = df.loc[(df.Ticker.isin(tickers)), ['Ticker']]
print(x.shape)
print(f'{len(x.Ticker.unique())} Tickers')
x[x.duplicated()]
```

```
(55, 1)
52 Tickers
```

Out[12]:

	Ticker

540	DTE
726	MRO
802	ATO

We noticed that these three tickers are duplicated. Let's see why?

In [13]:

```python
df_ishares[df_ishares.Ticker == 'ATO']
```

Out[13]:

	Ticker	Name	Sector	CUSIP	ISIN	Location
417	ATO	ATMOS ENERGY CORP	Utilities	49560105	US0495601058	United States
809	ATO	ATOS	Information Technology	5654781З	FR0000051732	France

It seems the ticker is not globally unique. We are going to filter by United States because our pilot stocks belongs to the SP500 US Stocks.

In [14]:

```python
df_us = df.loc[(df.Ticker.isin(tickers)) & (df.Location == 'United States'), ['Ticker']]
print(f'Now, we have {len(df_us.Ticker.unique())} tickers.')
```

```
Now, we have 52 tickers.
```

We know that we weren't able to match the ISIN for all the companies in the Environmental Intensity dataset. However, let's see the ones that we were able too and validate they matched correctly.

In [15]:

```python
found = df.loc[(df.Ticker.notna()) & (df.CompanyName.notna()),]
print(len(found.Ticker.unique()))
print(found.shape)
```

```
487
(495, 8)
```

Could it be possible to have duplicate values? Let's subset for the records that have the same ticker.

In [16]:

```python
ticker_agg = found.groupby('Ticker')[['Name']].count().sort_values(by='Name', ascending=False).reset_index()
tickers2 = ticker_agg[ticker_agg.Name >= 2]['Ticker']
df3 = found[found.Ticker.isin(tickers2)].sort_values(by='Ticker')
df3.head(6)
```

Out[16]:

	Ticker	Name	Sector	CUSIP	ISIN	Location	CompanyName	Country
428	AAL	AMERICAN AIRLINES GROUP INC	Industrials	02376R102	US02376R1023	United States	AMERICAN AIRLINES GROUP INC	United States
557	AAL	ANGLO AMERICAN PLC	Materials	-	GB00B1XZS820	United Kingdom	ANGLO AMERICAN PLC	United Kingdom
99	ADP	AUTOMATIC DATA PROCESSING INC	Information Technology	53015103	US0530151036	United States	AUTOMATIC DATA PROCESSING, INC.	United States
866	ADP	AEROPORTS DE PARIS SA	Industrials	-	FR0010340141	France	AEROPORTS DE PARIS	France
271	DTE	DTE ENERGY	Utilities	233331107	US2333311072	United States	DTE ENERGY COMPANY	United States
540	DTE	DEUTSCHE TELEKOM N AG	Communication	5842359 1	DE0005557508	Germany	DEUTSCHE TELEKOM AG	Germany

It happened the same thing as before, it seems the Ticker is not globally unique. However, it seems it matches correctly by ISIN. We did a little research and found that AAL in UK actually is AAL.L in Yahoo Finance.

Another example, DTE- DEUTSCHE TELEKOM ticker in Yahoo Finance is DTE.DE. We decided to re-format the Ticker for the companies that are not from US. To some of them, we need to add '.' + 'the first two letters from the ISIN' to the current ticker.

Let's see.

In [17]:

```python
ticker_agg = df.groupby('Ticker')[['Name']].count().sort_values(by='Name', ascending=False).reset_index()
tickers2 = ticker_agg[ticker_agg.Name >= 2]['Ticker']
df3 = df[df.Ticker.isin(tickers2)].sort_values(by='Ticker')
df3.head(6)
```

Out[17]:

	Ticker	Name	Sector	CUSIP	ISIN	Location	CompanyName	Country
428	AAL	AMERICAN AIRLINES GROUP INC	Industrials	02376R102	US02376R1023	United States	AMERICAN AIRLINES GROUP INC	United States
557	AAL	ANGLO AMERICAN PLC	Materials	-	GB00B1XZS820	United Kingdom	ANGLO AMERICAN PLC	United Kingdom

735	ADM	ADMIRAL GROUP PLC	Financials	-	GB00B02J6398	United Kingdom	ADMIRAL GROUP PLC	United Kingdom
234	ADM	ARCHER DANIELS MIDLAND	Consumer Staples	39483102	US0394831020	United States	NaN	NaN
866	ADP	AEROPORTS DE PARIS SA	Industrials	-	FR0010340141	France	AEROPORTS DE PARIS	France
99	ADP	AUTOMATIC DATA PROCESSING INC	Information Technology	53015103	US0530151036	United States	AUTOMATIC DATA PROCESSING, INC.	United States

In [18]:

```
df4 = df3[df3.Location != 'United States']
df4
```

Out[18]:

	Ticker	Name	Sector	CUSIP	ISIN	Location	CompanyName	Country
557	AAL	ANGLO AMERICAN PLC	Materials	-	GB00B1XZS820	United Kingdom	ANGLO AMERICAN PLC	United Kingdom
735	ADM	ADMIRAL GROUP PLC	Financials	-	GB00B02J6398	United Kingdom	ADMIRAL GROUP PLC	United Kingdom
866	ADP	AEROPORTS DE PARIS SA	Industrials	-	FR0010340141	France	AEROPORTS DE PARIS	France
802	ATO	ATOS	Information Technology	S56547813	FR0000051732	France	ATOS SE	France
553	DG	VINCI SA	Industrials	-	FR0000125486	France	VINCI	France
540	DTE	DEUTSCHE TELEKOM N AG	Communication	S58423591	DE0005557508	Germany	DEUTSCHE TELEKOM AG	Germany
560	EL	ESSILORLUXOTTICA SA	Consumer Discretionary	S72124779	FR0000121667	France	ESSILORLUXOTTICA SA	France
731	LEG	LEG IMMOBILIEN N AG	Real Estate	-	DE000LEG1110	Germany	LEG IMMOBILIEN AG	Germany
631	MRK	MERCK	Health Care	S47418447	DE0006599905	Germany	MERCK KGAA	Germany
726	MRO	MELROSE INDUSTRIES PLC	Industrials	-	GB00BZ1G4322	United Kingdom	MELROSE INDUSTRIES PLC	United Kingdom
563	PRU	PRUDENTIAL PLC	Financials	S07099542	GB0007099541	United Kingdom	PRUDENTIAL PLC	United Kingdom
547	SAN	BANCO SANTANDER SA	Financials	S57059461	ES0113900J37	Spain	BANCO SANTANDER SA	Spain
516	SAN	SANOFI SA	Health Care	S56717358	FR0000120578	France	SANOFI S.A.	France
739	TEL	TELENOR	Communication	S47324959	NO0010063308	Norway	TELENOR GROUP ASA	Norway
586	TSCO	TESCO PLC	Consumer Staples	S08847097	GB0008847096	United Kingdom	TESCO PLC	United Kingdom

In [19]:

```
ticker = df4.Ticker.tolist()
yahoo_ticker = ['AAL.L','ADM.L','ADP.PA','ATO.PA','DG.PA','DTE.DE','EL.PA','LEG.DE','MRK.DE','MRO.L','PRU.L','S
```

In [20]:

```
df = df.sort_values(by='Ticker')
df.loc[(df.Ticker.isin(ticker)) & (df.Location != 'United States'), 'Ticker'] = yahoo_ticker
df.head()
```

Out[20]:

	Ticker	Name	Sector	CUSIP	ISIN	Location	CompanyName	Country
867	-	IBERDROLA SA	Utilities	-	ES0144583236	Spain	NaN	NaN
711	1COV	COVESTRO AG	Materials	-	DE0006062144	Germany	COVESTRO AG	Germany
170	A	AGILENT TECHNOLOGIES INC	Health Care	00846U101	US00846U1016	United States	AGILENT TECHNOLOGIES, INC.	United States
428	AAL	AMERICAN AIRLINES GROUP INC	Industrials	02376R102	US02376R1023	United States	AMERICAN AIRLINES GROUP INC	United States
557	AAL.L	ANGLO AMERICAN PLC	Materials	-	GB00B1XZS820	United Kingdom	ANGLO AMERICAN PLC	United Kingdom

In [21]:

```
print(len(df.Ticker.unique()))
```

```
870
```

Now, we have 870 unique tickers that we can send to yahoo finance and get the company description. Below, we will create a tickers list. Also, we want to export the merged dataset to do some analysis onwards.

In [22]:

```
tickers = df.Ticker.unique().tolist()
```

In [23]:

```
df.to_csv('df_merged.csv', index=False)
```

Collecting companies data in Reuters website

Reuters, the news and media division of Thomson Reuters, is the world's largest multimedia news provider, reaching billions of people worldwide every day.

First, we are going to add '.OQ' to each ticker. T

In [24]:

```
tickers_US = df.loc[(df.Location == 'United States') & (df.Ticker.notna()),'Ticker'].unique().tolist()
tickers_nonUS = df.loc[(df.Location != 'United States') & (df.Ticker.notna()),'Ticker'].unique().tolist()
```

In [25]:

```
reuter_ticker = []
for ticket in tickers_US:
    ticket = str(ticket) + '.OQ'
    reuter_ticker.append(ticket)
```

In [26]:

```
# Create a loop to store URLs of all stocks' description page
URL = []
DES = []
comp_desc = {}
for i in reuter_ticker:
    url ='https://www.reuters.com/companies/'+i
    URL.append(url)
    page = requests.get(url) # visits the URL
    htmldata = BeautifulSoup(page.content, 'html.parser')
    Business_Description = htmldata.find('p',{'class':'TextLabel__text-label___3oCVw TextLabel__black___2FN-Z T
    if Business_Description is not None:
        DES.append(Business_Description.text)
        comp_desc[i] = [Business_Description.text]
    else:
        comp_desc[i] = 'Not exists'
```

In [27]:

```
exists = 0
do_not = 0
companies_desc = []
for val in comp_desc.values():
    if val == 'Not exists':
        do_not+=1
    else:
        exists+=1
        companies_desc = []

print(do_not)
print(exists)
```

```
362
143
```

We were able to find 143 companies descriptions which we will use later on this notebook.

Collect companies descriptions from Yahoo Finance

In this section, we will grab the company description directly from the Yahoo Finance website for the 870 companies

We were unable to send 870 tickers to Yahoo Finance. It seems their page crashed and wasn't allowing web scraping

In [ ]:

```
# Create a loop to store URLs of all stocks' description page
URL = []
DES = []
comp_desc = {}
for i in tickers[:2]:
    i = str(i)
    url ='https://finance.yahoo.com/quote/'+i+'/profile'
    URL.append(url)
    page = requests.get(url) # visits the URL
    htmldata = BeautifulSoup(page.content, 'html.parser')
    Business_Description = htmldata.find('p',{'class':'Mt(15px) Lh(1.6)'}) # finds the business description pa
    DES.append(Business_Description.text)
    comp_desc[i] = [Business_Description.text]
```

In [ ]:

```
df_comp_desc = pd.DataFrame.from_dict(comp_desc, orient='index', columns = ['Description'])
df_comp_desc.reset_index(inplace = True)
df_comp_desc.rename(columns = {'index':'Ticker'}, inplace=True)
df_comp_desc.head()
```

In [ ]:

```
df_comp_desc.to_csv('Companydescription.csv',index=False)
```

Start working with the DistilBERT

First, we are going to create a new dataframe using the **202 companies_desc.csv**. This is a merged file from the companies descriptions that we collected in Sprint # 6 and the descriptions from Reuters collected earlier in the notebook.

In [28]:

```
stock_des=pd.read_csv('202 companies_des.csv') #Yahoo descriptions
df_ei_2019 = df_ei[df_ei.Year == 2019]
df_ei_merged = pd.merge(df,df_ei_2019, on='ISIN',how='inner')
df_ei_merged = df_ei_merged[['Ticker','Env_intensity']]
df_ei_merged.dropna(inplace=True)
df_ei_merged.head()
```

Out[28]:

	Ticker	Env_intensity
0	1COV	-0.1423
1	A	0.0037
2	AAL	-0.2853
3	AAL.L	-0.1605
4	AAP	-0.0145

Now, we will merge this dataset with the descriptions so we have the necessary fields to start running the DistilBERT

In [30]:

```
df_final = pd.merge(df_ei_merged, stock_des, on ='Ticker', how='inner')
df_final = df_final.iloc[1:,:] #Removing AAL Dupplicate
df2 = df_final.copy().reset_index()
print(f'We are going to be able to send {df2.shape[0]} companies descriptions to the DistilBERT model')
df2.head()
```

```
We are going to be able to send 74 companies descriptions to the DistilBERT model
```

Out[30]:

	index	Ticker	Env_intensity	description
0	1	AAL	-0.2853	American Airlines Group Inc. is a holding comp...
1	2	AC	-0.1211	Associated Capital Group, Inc. provides invest...
2	3	ADSK	0.0093	Autodesk, Inc. is a design software and servic...
3	4	AEE	-1.3412	Ameren Corporation, together with its subsidia...
4	5	AEP	-1.6381	American Electric Power Company, Inc., an elec...

We created a binary variable that is 1 if the Environmental Intensity (Sales) is above its median and 0 otherwise.

This is the dependent variable (label) that we'll try to predict.

In [31]:

```
df2['HIGH_EI'] = (df2['Env_intensity'].gt(df2['Env_intensity'].median())).astype(int)
```

Load a pre-trained BERT model.

In [32]:

```
model_class, tokenizer_class, pretrained_weights = (ppb.DistilBertModel, ppb.DistilBertTokenizer, 'distilbert-b

# Load pretrained model/tokenizer
tokenizer = tokenizer_class.from_pretrained(pretrained_weights)
model = model_class.from_pretrained(pretrained_weights)
```

```
Some weights of the model checkpoint at distilbert-base-uncased were not used when initializing DistilBertMode
l: ['vocab_projector.weight', 'vocab_layer_norm.weight', 'vocab_projector.bias', 'vocab_transform.bias', 'vocab
_layer_norm.bias', 'vocab_transform.weight']
- This IS expected if you are initializing DistilBertModel from the checkpoint of a model trained on another ta
sk or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTrain
ing model).
- This IS NOT expected if you are initializing DistilBertModel from the checkpoint of a model that you expect t
o be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification
model).
```

Tokenize the textual data for DistilBERT which in our case would be the companies description

In [33]:

```
tokenized = df2['description'].apply((lambda x: tokenizer.encode(x, add_special_tokens=True)))
```

Pad all lists of tokenized values to the same size

In [34]:

```
max_len = 0
for i in tokenized.values:
    if len(i) > max_len:
        max_len = len(i)

padded = np.array([i + [0]*(max_len-len(i)) for i in tokenized.values])
```

In [35]:

```
np.array(padded).shape
```

Out[35]:

```
(74, 463)
```

Create attention mask variable for BERT to ignore (mask) the padding when it's processing its input.

In [36]:

```
attention_mask = np.where(padded != 0, 1, 0)
attention_mask.shape
```

Out[36]:

```
(74, 463)
```

We run the pretrained DistilBERT model on the prepared predictor and keep the result in last_hidden_states variable.

In [37]:

```
input_ids = torch.tensor(padded)
attention_mask = torch.tensor(attention_mask)

with torch.no_grad():
    last_hidden_states = model(input_ids, attention_mask=attention_mask)
```

Keep the first layer of the hidden states and assign the outcome variable to labels.

In [38]:

```
features = last_hidden_states[0][:,0,:].numpy()
labels = df2['HIGH_EI']
```

Split the data in train and test subsets, train the Logistic Regression on train set and evaluate its accuracy on the test set.

In [39]:

```
train_features, test_features, train_labels, test_labels = train_test_split(features, labels,test_size=0.2,ranc
lr_clf = LogisticRegression(max_iter=5000)
lr_clf.fit(train_features, train_labels)
print(lr_clf.score(test_features, test_labels))
```

```
0.8
```

Our model can 80% accuratly capture whether the company is high or low environmental intensity.

In [40]:

```
test_labels
```

Out[40]:

```
4     0
63    1
18    1
0     0
28    0
73    0
10    0
34    1
12    1
55    1
65    0
31    0
9     1
45    1
5     0
Name: HIGH_EI, dtype: int64
```

Conclusion

We had difficulties trying to map the 4,270 companies to their corresponding Ticker. Also, we discovered that sometimes 'Tickers' are not globally unique. Therefore, it can be difficult to find an accurate way to map the values.

Other difficulty was the technical issue with Yahoo Finance page. Their page wasn't allowing web scraping. Hence, we were unable to gather the 870 companies descriptions from the companies Tickers that we were able to map. We decided to collect some descriptions from the Reuters.com and, together with other companies description collected in earlier sprints, we were able to run the DistilBERT model.

The dependent variable, the one that we want to predict/classify, is High Environmental Intensity and after training the model with 80% of the observations, it was able to accurate capture whether the company is high or low environmental intensity 80% of the times.

This is the end of our analysis.