Project Journal

Creating a Systematic ESG (Environmental Social Governance) Scoring System using Social Network Analysis and Machine Learning to Influence Company Practices to be More Sustainable

Aarav Patel

11/6 - Identifying a List of Relevant ESG Keywords to use for Data Collection

- In order to collect data, I will need to create a list of keywords to search for relevant information pertaining to the company
- I read over how current firms like MSCI, S&P Global, etc go about determining their ESG scores
    - They are all very systematic and have categorical outlines
    - I could potentially use these category names, but I might have to omit certain ones that are seldom discussed on social networks
- To make it so my sample of the ESG tweets are balanced, I can get the total number of tweets under each keyword; calculate the percent frequency of each keyword, and then take sample sizes based on this
    - I will make the balance between environmental, social, and governance equal
    - I could also simply just do separate searches for each keyword, and get the respective average of each one individually
- I will use the sumup.ai list of keywords for my keyword search algo
    - I still need to find more sources to make it more comprehensive; maybe I can use a semantic web search where it searches for similar keywords as well
- To make it so my sample of the ESG tweets are balanced, I can get the total number of tweets under each keyword; calculate the percent frequency of each keyword, and then take sample sizes based on this
    - I will make the balance between environmental, social, and governance equal
- I read over how current firms like MSCI, S&P Global, etc go about determining their ESG scores
    - They are all very systematic and have categorical outlines
    - However, they focus more on risk, but are still “synonymous”
- I also thought of some ways I can make the data mining part more accurate and relevant to the company

11/7 - Gathering ESG scores

- I looked for different public ESG ratings that I could potentially use to calibrate my ESG ratings to
- After some searching, I saw that MSCI did not have free public ratings. Sustainalytics has public ratings (though they are a bit odd since they specifically measure risk). S&P Global has free ratings on 9000 companies
- I will likely use S&P Global ESG ratings

11/11 - Creating a List of Companies

- Create a yahoo finance stock screener
- Realized would be more efficient to have an algo check which stocks have ESG scores on S&P Global rather than do it manually

11/11 - Fixing the ESG Problem

- I began thinking of some obstacles that I might encounter during data collection
- Some problems I identified are:
    - Getting a balanced number of tweets for both positive and negative sentiment. This is because some keywords only give positive or negative posts/tweets (like “corruption” only gives negative results while “philanthropy” would likely give more positive ones)
    - Eliminating “misunderstandings” in tweet/post-selection. I have noticed that if I was scraping data for Amazon, for example, it might scrape data about the Amazon rainforest rather than the Amazon company
    - Sarcasm in tweets (minor)
- Saw can use ESG parser to see parts of speech, etc, and see if they are talking about the organization in order to limit “misunderstandings” in the data
    - Some parsers I can use to do so include:
        - Stanza.run (worked quite well) (by Stanford)
        - CoreNLP.run
        - https://nlp.stanford.edu/software/lex-parser.shtml
- I will work on the Wikipedia scraper and News scraper first and try to read more literature in the meantime to address these issues of balancing Tweets and LinkedIn posts so there is no keyword bias that misconstrues results

11/12 - Looking into Reddit, Other News, Wikipedia, Glassdoor, LinkedIn

- Today, I looked at how I might go about scraping data for Reddit, News, Wikipedia, etc
- I thought about also profiling the company CEO to get a better understanding of the company’s governance
- I looked at the viability of Glassdoor and LinkedIn to supplement my analysis
- It seems like most of these social networks seem viable. Reddit seems to have limited amounts of relevant company data, but Glassdoor, LinkedIn, and the news have larger amounts

11/15 - Creating an ESG Score Web Scraper

- I began creating an ESG score web scraper to get the most up-to-date data about a company's ESG
    - My web scraper would collect data from S&P Global
- Unfortunately, S&P Global's URL system was very abstract; I was able to get information for one company at a time, but looping over it was tough
    - They abstract the URLs by using a number for each company instead of a ticker
    - I investigated the viability of doing a "search" in the search bar for the website
        - However, this might get too complicated

11/16 - Creating a Rough Implementation of the Wikipedia Web Scraper

- Today I created a rough implementation of the Wikipedia web scraper. The Wikipedia scraper serves to give a basic overview of a company's ESG practices
- I created a function that can convert stock tickers to full company names using Yahoo Finance
- I created a function that gets a company's Wikipedia data using the Wikipedia library
- Finally, I created an analysis function that dissects the article content into various sections. It then checks the relevance of each section by using a keyword search algorithm on either the title or paragraph content. From there, it calculates the sentiment for that section by using the NLTK library (which is also very good for many other things as well
- However, the current filtering algorithm seems very inaccurate since it is omitting lots of essential information. It would likely be smart to include more keywords in the lists

11/18 - Refining the Wikipedia Web Scraper

- I began by re-running the Wikipedia algorithm, and I found there was an exception when I ran BAC that said wikipedia.page doesn't match id
    - In the exception, it was saying I looked up "band of america", even though that was technically not what I had searched
- I tried modifying my code slightly in various ways to see if there were any bugs causing this. I had limited success

- I stumbled on this GitHub forum - https://github.com/goldsmith/Wikipedia/issues/192
    - It said to set the auto_suggest parameter to false in order to ensure what is typed is strictly what is searched
    - I tried it and it worked
- I also thought there was also an error in the ticker to long-name conversion, but it was because I had entered one of the tickers incorrectly
- I ran the scraper again; the filtering algorithm is very inaccurate based on me eyeballing it, but the rough implementation is done; I will modify the program later on

11/21 - Automatically Retrieving ESG Scores for Every Company

- I continued working on my web scraper which could automatically retrieve S&P Global ESG scores for a company
- I stumbled upon this GitHub crawler - https://github.com/shweta-29/Companies_ESG_Scraper - someone had already made something that can retrieve ESG scores for a company
- I installed the various packages needed to run the program, debugged certain parts, etc
- I found that the crawler failed to retrieve data for most of the ESG rating agencies due to old and unmaintained code. However, it worked for S&P Global

11/21 - Researching Named Entity Recognition with NLTK

- I began researching libraries that I could use for named entity recognition. This would help me separate out Tweets/posts that talk about the actual company rather than something similar
- https://towardsdatascience.com/named-entity-recognition-with-nltk-and-spacy-8c4a7d88e7da - this article said how to conduct named entity recognition with NLTK and spacy

11/21 - Researching How to Scrape News Data

- I began researching ways I could collect news data for my scraper
- I found an article (https://medium.com/analytics-vidhya/googlenews-api-live-news-from-google-news-using-python-b50272f0a8f0) which is about how to use the google news API
    - Google news seems like the most efficient way to scrape news data

    - However, I am encountering slight difficulties when adapting this article's code where the getpage request is throwing a bug

11/22 - Creating a Prototype for the News Algorithm

- I created a function to get URLs for relevant company news articles via Google News
    - Google News only can get 10 articles at a time, so I just iterated my program over the request multiple times to scale this up
    - I plan on using a keyword search algorithm and a named entity recognition algorithm to get the most relevant news
- I created a function that would retrieve the article's content by using this URL
- I used the NLTK sentiment analysis algorithm to calculate each news articles' sentiment
- After testing this rough implementation, I found that it gets some irrelevant news that does not pertain to ESG (for example, when I looked for "Amazon diversity," it got articles about new movies on Amazon Prime). Also, the sentiment analysis algorithm seems completely wrong for some articles (there was an article about AWS real estate managers embezzling millions, but the sentiment analysis scorer gave it positive sentiment)

11/27 - Fixing the Sentiment Analysis Algorithm for News Scraper

- I tried assessing the root cause of the issue of the sentiment analysis algorithm for Google News
- I tried 3 different sentiment analysis implementations: The first one would just analyze the article title, the second one would analyze the sentiment of the text as a whole, and the third one would analyze the sentiment of each paragraph individually and sum this together to get the average sentiment.
- Unfortunately, while implementing these algorithms, I got the error "HTTP Error 429: Too Many Requests." Google News temporarily restricted me since I made too many requests in a short amount of time.
- When I finish this meta-analysis, however, it can help me roughly determine which sentiment analysis approach would be optimal

11/27 - Looking Further into the Viability of Using LinkedIn and Glassdoor

- There is a wide range of company data available on LinkedIn. I looked to see if it would be viable to add to the algorithm

- After looking through posts pertaining to companies like Chevron, Amazon, Apple, and Kroger, I found that many posts speak positively about the company. However, the people who put these posts out are either employees at the company or the company themselves. So, if I try filtering out these “self-reported” posts, then there is a more balanced and holistic overview of the company’s practices
- LinkedIn is a goldmine for data since nearly all data on LinkedIn is about companies
- Methods to Gather Data:
    - The LinkedIn python API I found online does not have support for LinkedIn posts, so if I go forward with using LinkedIn, I will have to brute force it with selenium chrome driver
- I also looked into company data on Glassdoor
- There are many company reviews from employers (big companies like Amazon have over 100k reviews, while smaller ones like Sprouts Farmers Market have 3.5k). Each company has overall ratings for different categories, which is helpful too.
- Glassdoor does not have an API for reviews
- Methods to Gather Data:
    - https://github.com/MatthewChatham/glassdoor-review-scraper - Github repo for Glassdoor review scraper
    - https://vu-d.gitbook.io/journey/data-analytics/glassdoor-scrape - using python to scrape Glassdoor reviews
    - https://github.com/csharma91/GlassdoorData - Glassdoor ratings and trends script (incomplete)
- Some places online are saying that scraping Glassdoor data might be illegal; after looking at the Glassdoor terms of service, it says Glassdoor can’t be used for commercial use unless we enter in a separate agreement; However, they also say this does not apply to research institutes

11/28 - Debugging the Google News Sentiment Analysis Algorithm

- I tested my 3 implementations for sentiment analysis for Google News algorithms. I found that simply just analyzing the title does not give a full picture of an article’s ESG. Additionally, when I went to manually check, analyzing the whole article as one body of text yielded odd results (an article about Amazon workers being underpaid had a near-perfect sentiment score of 0.98). Analyzing each paragraph separately, and then summing the score up later seemed to be the most accurate, but even that it could be improved

    - I think the reason behind such odd results is the fact that news reporters try to unbiasedly report on the world around them. Even if they are talking about a negative topic, a sentiment analysis algorithm might detect neutral sentiment
- To fix these issues, I think I will have to supplement the algorithm with something else. I will likely do this with a bag-of-words approach with a unique ESG based lexicon
    - I will do this once I have rough implementations for all the scrapers, since then I would have a better understanding of what approach might work best
- I can also test out other sentiment analysis tools

11/28 - Creating a Rough LinkedIn Web Scraper Implementation

- There is no good LinkedIn API, so I must use the Selenium Chromedriver instead
- I installed the necessary packages to run Chromedriver on python
    - I ran into an error where Chromedriver won't part of my system's path, but I fixed it by putting the full path itself out as a parameter
- I was able to create code that would log in to LinkedIn, search up the desired parameters, and scroll through to get a certain amount of posts
- I then inspected the webpage to get the necessary class headers. I used these class headers to filter out information from each post. This step took a while since I had to test out various classes to find the right one, but eventually, I found it.
- Next time, I have to clean up the post data even more, since the content inside posts often has links embedded, which means I have to filter these out and store them somewhere else

11/29 - Filtering LinkedIn Post Data Out of Raw HTML

- Today, I made some code that can filter the raw text my web scraper retrieved into formatted text which I can use
- I programmed it using regex and was able to filter out the HTML text into text which I will use for analysis

12/1 - Getting Post Author Data on LinkedIn

- Oftentimes, employees or executives at companies self-report their practices in an unrealistically positive light; I want to filter out these self-reported posts

- To do this, I tried to find the link to an author's profile based on their post
    - I tried using class tag "app-aware-link feed-shared-actor__container-link relative display-flex flex-grow-1"; it didn't work
    - I tried using class tag "feed-shared-actor__name t-14 t-bold hoverable-link-text t-black"; it didn't work
    - I analyzed the website's html Soup to better understand which class calls what; if found that I can use the "feed-shared-actor display-flex feed-shared-actor--with-control-menu" to get relevant information
- I parsed this information to filter out the person's name and link to their profile
- I tried using beautiful soup to navigate to the person's LinkedIn profile, but I got an error saying the html5lib is unable to parse the page, and that I need to install a parser library
- I reformatted my code and used another parser; I got the websites "soup," but it did not include the info I was looking for
    - In normal html, their company is under "pv-entity__secondary-title t-14 t-black t-normal"
    - However, it wasn't present
- https://stackoverflow.com/questions/61192281/linkedin-profile-name-scraping - this stack overflow said the solution to my problem is simply to just use Selenium since there is no easy way to make it work with beautiful Soup
    - This will make data collection much slower. Unfortunately, it might be the only way

12/2 - Looking How to Scrape Data from Glassdoor

- I found that Glassdoor has an API that can get the overall company ratings that I desire
- Glassdoor has a bad interface where it is hard to make an account (since we have to leave a review to do so); the documentation on their site also does not have much information; finally, I need to give them access to all of my cookies for the site to work properly and for me to get access to the API
- Even after I enabled cookies, Glassdoor gave an error where I could not access the API
- I looked for some tutorials/past implementations online but could not find anything worthwhile
    - https://pretagteam.com/question/python-glassdoor-api
    - https://www.reddit.com/r/learnpython/comments/jy9ykz/trying_to_build_a_glassdoor_scraper_not_sure_how/ - This person had a similar problem as I did; he used Beautiful Soup to solve it, but then got bot detection errors

    - https://vu-d.gitbook.io/journey/data-analytics/glassdoor-scrape - get ratings of individual users
    - https://github.com/ceewick/glassdoorScraper/blob/master/2.0/README.md - this can get sub-ratings

12/3 - Setting up a Twitter Developer Account in Order to Easily Collect Tweets

- Twitter has a host of built-in API's for scraping data
- I created a Twitter account for my research project email apcoin2021@gmail.com so I could use this email rather than my personal one for the project
- I created a basic-level developer account; I should be able to get access to an academic account (which gives more functionality), but the application process takes extra time, so I will likely do this later on
    - The basic developer account only gives access to a million tweets. This might not be enough, but it should suffice for prototyping purposes
- I created a "TwitterESGProject" name for my developer account; I got a confidential API key and bearer token
- I tried testing the key and bearer token on a test endpoint; I ended up getting an error that said I was forbidden access to the page
    - I tried modifying this, but it still didn't work
- I created a rough program that utilizes the Twitter API in python
    - I was running into an authentication error where even though I put a bearer token, I was still getting an error that said I needed an API key
    - I looked online for solutions, but I found nothing easily
    - https://developer.twitter.com/en/docs/authentication/overview - methods of Twitter API authentication
        - https://developer.twitter.com/en/docs/authentication/oauth-2-0 - auth 2.0
            - When I tried the *curl "https://api.twitter.com/2/tweets?ids=1261326399320715264,1278347468690915330" -H "Authorization: Bearer ***bearer token***"* it successfully was able to retrieve data
- https://towardsdatascience.com/an-extensive-guide-to-collecting-tweets-from-twitter-api-v2-for-academic-research-using-python-3-518fcb71df2a - gives an overview on how I can implement the Twitter API with Python

12/4 - Looking Further into Glassdoor Scraping

- I tried looking for more solutions with using Glassdoor, but I still could not find anything
- I looked into potentially using indeed.com as a supplement

12/6 - Emailing Glassdoor

- I tried emailing glassdoor about the API issue
- For some reason, when I sent the message, the server crashed. I tried this on different computers, and later on in the day as well, but I kept getting the same error.

12/9 - Creating a Working Implementation of the Twitter Scraper using Snscrape

- Since I am using a "Twitter Essential Developer" account, I am also restricted from accessing tweets between certain time ranges
- I tried using the GetOldTweet3 library to get old tweets
    - When I ran the code, I got an "HTTP Error 404: Not Found"
    - However, when I checked the link myself, the site existed
    - Based on this forum (https://github.com/Mottl/GetOldTweets3/issues/98), I found many people have the same problem, but can't find any solution; I think there are just outdated bugs with the library
- I also tried using my Mentor's social network analysis tool named Griffin, but unfortunately, it can scrape tweets only 7 days back
- I tried snscrape, an alternative to GetOldTweet
    - https://betterprogramming.pub/how-to-scrape-tweets-with-snscrape-90124ed006af
    - I installed git to get the necessary packages
    - The program successfully can retrieve tweets from a certain date range
    - I tried converting the data frame into a CSV which I could store on my computer, but I was first getting errors since my program did not have permission to access my file directory
    - I resolved it by importing os via python and using that to open the file rather than the package itself
    - I began polishing the Twitter program

12/10 - Finishing up the Twitter Algorithm

- At this point, I wanted to finish up the Twitter algorithm so I could test out all the data-collectors simultaneously

- I re-ran the Twitter algorithm and was getting an error that said “"Unable to find guest token"; this error was very random, since sometimes when I run the code, I don’t get it, while other times I do
    - Online, it said there was no easy and definite fix
    - I looked for other Twitter scraper alternatives, but nothing matched that of snscrape
    - I played around with the code, debugged, and tried looking for workarounds. I found that adding a time delay significantly decreased the likelihood of the error, so I added a time delay
        - I also added error trapping where If I got this error in the future, my code would automatically pause and calibrate
- I attempted to contact Glassdoor for an API partnership so I could scrape data on their site. The contact form did not work. Unfortunately, I don’t think there are any workarounds, so I will have to type in these scores manually

12/10 - Grouping all of the Programs Together

- I went through the code for all my web scrapers thus far. Since I have been working on each in a separate project, I created a new project where I put all of them together in one codebase
- I cleaned up and optimized the code for each program
- I gave the program access to my computer’s files so they can write data down onto my hard drive. This will be very useful later when I conduct an analysis
    - At first, I was getting a permission denied Error. So, I moved the file location out of my “C:\\” drive and onto my profile drive, since that does not require authentication
    - I created code where the LinkedIn scraper and GoogleNews scraper could write data onto a CSV file and store it in its respective directory

12/11 - Finishing up a Rough Implementation of the Data Scraper Part of my Project

- I created code where the Wikipedia and Twitter scrapers could write data onto a CSV file and store it on my computer
- I created code that could also attach the analyzed data scores onto the CSVs
    - I made a basic NLP analysis algorithm to test this out which relied solely on sentiment analysis
- I debugged the code for small bugs/errors; I also error trapped my entire program
- I optimized the code a b**it more to increase the speed of the web scrapers**

- **At this point, my code can successfully scrape and store all necessary data, so, I think** the data scraping portion of my project is complete (besides Glassdoor)

12/15 - Polishing up the Data Scraper

- I continued to optimize tiny parts of all of my data scraper programs
- I cleaned up the code and added comments to make it more readable
- I sent the code to my mentor for him to check over and give me feedback for
- He responded quite quickly and told me that there seemed to be no major issues

12/16 - Gathering Company News Volume

- I wanted to gather news/search volume surrounding a company to make the experiment itself more balanced
    - I also want to see how well the algorithm works when faced with little news volume to work with
    - However, I am not sure if there is a definite way to measure this
- I looked into these various approaches to see if I could find news volume this way:
    - Google Trends does not provide specific counts for monthly searches
    - Adtargeting.io can get the number of monthly searches surrounding a company
        - I made an account, however, the website requires a premium account in order to retrieve more than 20 searches
    - Google Ads has a google keyword planner (https://adwords.google.co.in/KeywordPlanner) which provides the data I need for free
    - Another possibility is just searching the company name normally, and see how many search results come up

12/18 - Creating a List of Companies

- I created a screener in yahoo finance to identify a list of 200-400 companies (both mid and large-cap) per sector
- I tried creating a python script to automatically execute the GitHub ESG crawler that I found previously; I ran into a bug where my os.system commands would not execute since some commands prompted the user for more input

- I re-tested the GitHub crawler (https://github.com/shweta-29/Companies_ESG_Scraper) manually; I found there was an error when scraping MSCI b/c of cookies; S&P was able to collect data, but it was not necessarily the companies I put in the spreadsheet
    - I still have to test CSRHub, yahoo finance, and sustainalytics. However, it will suffice if I simply just use S&P Global data as a baseline

12 /19 - ESG Company Score Collection

- I tested the ESG crawler for Yahoo Finance; there was a cookies error
- I tested the crawler on CSRHub; the program was completed, but no scores were stored
- I think these errors are because I am passing in the tickers, rather than the full names
- I re-tested the crawler but with full names rather than tickers
    - To get the full name, I googled the ticker and copy-pasted the name at the top
- After re-testing I found
    - CSRHUB - it is being blocked by a ReCaptcha; the program can finish without bugs, but it does not return any data
    - S&P - it works perfectly, and it collects data for the right companies
    - MSCI - runs into a cookie error
        - I tried solving it by automatically accepting all cookies for my Chromedriver account

1/5 - Creating and Debugging the LinkedIn Profile Filter

- Previously I tried to use beautiful soup to scrape LinkedIn profiles, but this did not yield the necessary company data I needed
- I tried basic Selenium approaches where I collected a page's source code, but for some reason, the person's company was not present
- After hours of trying different approaches via selenium (by playing around with different class and id tags), I still could not find a way to get the necessary data

1/5 - Finishing up ESG Score Collection

- I ran the GitHub crawler to collect S&P ESG scores for all companies I have compiled… it was a lengthy process since it retrieved scores for over 3000 companies
- I uploaded these files to google drive as a backup

1/5 - Researching the Creation of an ESG Relevance Measurement Algorithm

- I realized that lots of the data I have are not directly related to company ESG; I need to eliminate this excess data in order to significantly boost accuracy
- I could use a keyword approach where it measures the proportion of relevant keywords there are within a text
- I read a few research articles detailing approaches/solutions some have done in the past
    - https://arxiv.org/pdf/2010.08319.pdf - Detecting ESG topics using domain-specific language models and data augmentation approaches
        - They didn't go too much into the details, but it seems they created their classifier by using analyst labeled ESG articles
        - I can potentially contact them to see if they would be willing to send their classifier
    - https://web.p.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=0&sid=84308b93-37c7-47bf-8923-2c543b67916e%40redis - Uncovering hidden signals for sustainable investing using Big Data: Artificial intelligence, machine learning, and natural language processing - https://drive.google.com/file/d/1mEV0QdNl7N8sf9u71B-5J5xoDVaV5Ejr/view?usp=sharing
        - Did not include much information pertaining to ESG relevancy detection

1/5 - Refining the Web Scrapers and Analyzers

- I want to make sure everything is ready before I begin data collection for all companies
- I notice minor bugs in the Twitter and News algorithm where the code used the company ticker as a search term rather than the company name itself
- I made changes to the programs, but when I re-ran the twitter algorithm, I randomly ran into the "snscrape.base.ScraperException: Unable to find guest token" bug which I resolved a few weeks ago
- The bug persisted, despite reverting my code to what it previously was

1/7 - Fixing the Twitter Bug (again)

- The Snscrape bug I faced last time came up again, and this prevented me from running the Twitter scrapers
- I tried using debugging approaches that I used last time I faced this snscrape bug, but it came to no avail
- I created a new project in a new directory to see if even a simple implementation of snscrape still worked
    - Luckily, it worked!
    - This means there is something somewhere in my large codebase that is corrupting the file
    - It might also be that I have made so many requests with my code that snscrape has blocked it
- I tried testing the same basic implementation again, only this time in the same file directory as my original code
- I incorporated this new code back into my old code, reformatted it, and somehow it worked. I'm surprised since fundamentally nothing changed with the program

1/7 - Checking over Collected Data

- I began collecting data for a small sample of hand-picked companies
- I looked at the tweets/news articles/LinkedIn posts collected. I found that oftentimes my sentiment analysis algorithm misclassified something that represented bad ESG into something that showed good ESG. As per my estimates, this happens for around 30% of the posts
- I also saw that sarcasm inside Tweets created a large number of false positives. I need to combat this by either using a sarcasm detecting algorithm to offset this or using a new sentiment analysis algorithm entirely just for Twitter (since tweets are typically written differently than posts on many other platforms)
- I looked into other sentiment analysis libraries I can use as well… I will run multiple algorithms and then choose the one that returned the best results
    - https://www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/ - some more about different sentiment analysis algorithms
    - There is TextBlob, VADER, SentiWordNet, etc
    - NLTK (the one I am currently using) is a rule-based classifier that assigns positive and negative scores to specific words, and then adjusts for negation
        - Does not work well with sarcasm

- https://towardsdatascience.com/the-best-python-sentiment-analysis-package-1-huge-common-mistake-d6da9ad6cdeb
  - It showed that Flair machine learning model significantly outperformed the VADER/NLTK one since it uses word embeddings, rather than simple word based classification
  - http://alanakbik.github.io/papers/coling2018.pdf - why Flair ML algo is best
- https://towardsdatascience.com/fine-grained-sentiment-analysis-in-python-part-1-2697bb111ed4
  - FastText is another word embedded NLP algorithm
  - https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf - Stanford paper where they used parsing to create efficient sentiment analysis
  - TextBlob had an F1 score of 24.7%
  - VADER was made specifically for analyzing social media
    - VADER had an f1 score of 31.3%
  - For FastText, a word embedding algo, it was 39.1%
  - For flair, it was higher
    - You can also stack word embeddings from other algorithms with flair to make it more accurate
    - Flair+BERT has an F1 of 40.2%
    - Flair+ELMo has an F1 or 44.2%
  - https://towardsdatascience.com/fine-grained-sentiment-analysis-part-3-fine-tuning-transformers-1ae6574f25a6 - optimized even more to have transformers
    - F1 is now 47.1%

1/7 - Creating Code to Organize Score Averages into One File

- At this point, my collected company data is scattered in various subfiles within my directory; I need the averages to be stored in one file to feed it to my machine learning algorithm
- My program iterates through all data files for a company
- Afterward, It reads the file contents and computes an average
  - I was running into a bug where the python CSV reader could not read my files since there was a unique non-decodable character (0x9d) present
    - After some research, I found this character: ” (also known as: "RIGHT DOUBLE QUOTATION MARK"), which has the bytes 0xE2 0x80 0x9D, which includes the problematic byte

    - I tried resolving this by specifying the encoder to ‘utf8’ and it fixed the problem
- These averages are then written down into a master CSV file

1/7 - Planning the Machine Learning Algorithm

- I began planning my machine learning algorithm which would be trained to automatically score ESG
- Data collection has been incredibly time-consuming (takes 15-20 mins per company). Even if I run my collectors on multiple computers simultaneously, data collection would still take an incredible amount of time for 500-1000 companies.
- I looked into some machine learning algorithms that work well on limited datasets
    - Simpler models tend to work better
        - Linear Regression seems like an obvious choice that comes to mind
        - Naive Bayes classification, or other classification algorithms can also be good
        - SVM under some conditions
        - k-nearest neighbor
        - Decision tree
- Another solution to combat my limited data issue is to use data augmentation, where I essentially artificially create more data
    - https://towardsdatascience.com/breaking-the-curse-of-small-datasets-in-machine-learning-part-1-36f28b0c044d - article discussing methods for dealing with limited data
    - I can use company ESG over various years (though, this would not reduce data collection time for my model)
    - Synthetic Minority Over-sampling Technique (SMOTE) - I can create synthetic data by generating random points in between two other points
    - I can generate more data by adding Gaussian noise to the data
    - Use ensembling techniques where I combine multiple algorithms together

1/9 - Testing out Different Sentiment Analysis Methods

- Today, I tested out various sentiment analysis libraries to see which one would be best for my project
- I reformatted my code to make it so that I can test multiple classifiers at the same time (this will be especially useful since it makes optimizing the code easier)
- I tried implementing the following libraries: Core NLP, Flair, Watson API, SciPy

    - https://towardsdatascience.com/natural-language-processing-using-stanfords-corenlp-d9e64c1e1024 - how to install core NLP for python
        - Bug with core NLP… needed specific keys to access server
        - Overall, the process became too tedious since core NLP is built for Java, so instead of simply pip installing a library, I must connect my computer with an external server
    - Nltk and TextBlob seemed to show similar results, but both were not as accurate as I would like them to be

        · I tested the flair algorithm, and I found it to be exceedingly slow
        o There is an alternative called “sentiment-fast” within the flair library (https://github.com/flairNLP/flair/blob/master/resources/docs/TUTORIAL_2_TAGGING.md#list-of-pre-trained-text-classification-models)
        o Even after running the sentiment fast algorithm, I still found that it wasn’t that fast
    - I tried using a batch classification algorithm for Flair, which means it would analyze multiple bodies of text at once… this reduces the number of requests needed and significantly reduces the time
    - The IBM Watson API unfortunately only allows for a few thousand free uses… after that, I would need special permission
- ScaPy required me to pass in data to train the algo from scratch, which I did not pursue since I was anticipating to get similar accuracy as the one from flair
    - I created a function to automatically add these scores to a csv if it was not there already
    - I found the flair algorithm to have the best accuracy; I will likely go forward with that

1/22 - Gathering Preliminary Data for ML Algorithms

- Since it has been a while (I did not work since it was mid-term week), I reran my program to check if there were any bugs; there were none
- I created a LinkedIn account for my apcoin2021@gmail.com email so I would not have to use my personal email for data collection
- I modified the flair algorithm with a variant called “flair sentiment fast” to batch classify tweets rather than singularly classifying them; this significantly sped up the algorithm, but it is still slower than NLTK
- I modified the long-article algorithm so it would build off the pre-coded functions of the short-post algorithm
- I added a feature so my scraper would not overwrite preexisting data

- I ran my scraper algorithm overnight to collect data for 20 companies; I used the NLTK classification algorithm for speed purposes

1/23 - Analyzing the Data with Machine Learning (Linear Regression)

- When I came to my computer, I saw that data collection finished without any errors
    - However, when I went to check the data itself, I saw there was some missing data in the Twitter scraper, and there was virtually no data from the news scraper
    - This is probably because I got rate limited, but I will investigate further
- I used the sklearn.impute library to replace any missing data with the mean of that column
    - However, if the amount of Nan variables were above 6, then I would just eliminate the column entirely
- Now that preliminary data was collected (albeit a very small amount), I wanted to see if I could use it to train a basic machine learning algorithm
- I created a prelim linear regression algorithm
    - I used ~10 companies as the training set, and ~5 as the testing set
    - When I graphed the predicted data against the actual data, it showed very little correlation
        - This obviously makes sense since I have 40-50 features being trained on 10 data samples… I will need much more data to get a better understanding

1/23 - Implementing SMOTE

- SMOTE is a data augmentation technique that is typically used to optimize results for regression problems. It helps a program adjust more for outliers
- How to implement SMOTE in Python
    - https://towardsdatascience.com/smote-synthetic-data-augmentation-for-tabular-data-1ce28090debc
    - https://rpmcruz.github.io/machine%20learning/2018/05/11/regression-data-augmentation.html - GitHub with some augmentation techniques
- I implemented a rough implementation of SMOTE with the aid of the KNN Regressor
- I also implemented a Gaussian noise function as another way to augment data

1/23 - Migrating Code onto Another PC

- To more efficiently collect data, I decided it would be smart to do it on my dad's PC since it has more computational power and can run overnight in the background
    - Also, this is somewhat necessary since data collection should take around 3-5 days for 500 total companies
- Today, I uploaded my code to a private GitHub repo
- I transferred it to my dad's PC, which has much more computational power that will allow me to run multiple programs simultaneously
- I redownloaded the necessary libraries, and I installed another Chromedriver
- I created a requirements.txt file so I could more efficiently install python packages

1/29 - Data Collection on my dad's PC

- I made changes to the data collectors where it would omit companies that do not have Wikipedia pages
- I created 4 new email addresses and LinkedIn accounts so I could run it 4x at once
- I ran the program on 4 different cores to speed up collection
- I waited for it to run overnight

1/30 - Checking the Newly Collected Data

- I saw that the programs were able to run smoothly without crashing
- When I looked at the specific data collected, I saw that after a while it completely stopped collecting data for Twitter and Google News
    - Maybe this is because it got rate limited
- I began looking around for some ways to not get rate limited
    - I could potentially pause my program for like 10-15 minutes periodically
    - https://github.com/Altimis/Scweet - this GitHub has another Twitter scraping API that can collect unlimited Tweets

2/3 - Conducting Further Testing on my Models

- I created a function that iterated over my models with different random train-test splits
- I tested the accuracy (mean absolute average error) of each algorithm and wanted to see how it varied depending on the amount of data

    - ***note that a regression algorithm that simply just guessed a random number would have a mean absolute error of 34
    - I tested with ~25 samples with a train-test split of 75%-25%, and I got the following results:
        - supportVectorRegressionAccuracy 25.250
        - linearRegressionAccuracy 26.830
        - decisionTreeRegressionAccuracy 30.536666666666665
        - randomForestRegressionAccuracy 27.390000000000005
        - XGBoostRegressionAccuracy 29.96333333333332
        - KNNRegressionAccuracy 23.84999999999999
    - I tested with ~50 samples with a train-test split of 75%-25%, and I got the following results:
        - supportVectorRegressionAccuracy 19.120000000000007
        - linearRegressionAccuracy 24.40833333333334
        - decisionTreeRegressionAccuracy 23.241666666666674
        - randomForestRegressionAccuracy 19.39166666666667
        - XGBoostRegressionAccuracy 21.46166666666667
        - KNNRegressionAccuracy 17.993333333333344
    - I tested with ~100 samples with a train-test split of 75%-25%, and I got the following results:
        - supportVectorRegressionAccuracy 15.717272727272727
        - linearRegressionAccuracy 48.60727272727272
        - decisionTreeRegressionAccuracy 23.333636363636356
        - randomForestRegressionAccuracy 17.92636363636364
        - XGBoostRegressionAccuracy 19.630909090909092
        - KNNRegressionAccuracy 15.422727272727273
- I added a feature where my allScores file would store the number of data points per sub-score so I can see how reliable that score might be. It can also allow me to disregard averages that have only a few relevant data points
- https://corpgov.law.harvard.edu/2017/07/27/esg-reports-and-ratings-what-they-are-why-they-matter/: Bloomberg, as well as many other ESG raters, penalizes companies who fail to provide certain data… maybe if I do that it can increase accuracy

2/4 - Researching the Development of an ESG Irrelevant Data Filtration Algorithm

- I began looking for ways to filter out irrelevant ESG data
- I looked at my raw data to check for patterns between relevant data and irrelevant data:
    - News data:

-

Company	Issues	Potential solutions
JNJ	One article got a report on Boris Johnson gray (miscategorized person instead of company) One article got a report on Najee Harris and Dionte “Johnson” One article talked about JNJ’s vaccine but didn’t even mention diversity When searching for different keywords, it was showing almost the same results (scores for different categories were almost identical)	Used named entity recognition to see if talking about the company instead of a person Use “” when searching to make sure it searches for full company name Check if the article has the keyword (anywhere really) Maybe search the keyword before the company name; check if the article has the keyword

BBY	One article talked about Russia-Amer relations; it mentioned the negotiation “climate”, and that the “best buy” date for materials was X/X/XXXX (just simply checked if contained keywords One article talked about climate change and did not even mention BestBuy Another article talks about best buy stock but not actually best buy There is a weird text title where it includes random “â€ ” characters When searching for NewsESG for BestBuy, returned many generic Zacks or MarketWatch articles that talk about many sustainable stocks in general	Use named entity recognition to check if best buy company is mentioned Check if it contains both keywords Use regex to clean and format the text

- LinkedIn Data
-

Company	Issues	Potential Solutions

RGLD	A lot of the LinkedIn posts do not contain the name Royal Gold, and instead, have the keyword carbon… a lot simply just have one keyword such as royal or gold A lot of posts did not pertain to royal gold, but instead gold in general	Have named entity recognition See if the company name is in the post Use “” to search
NFLX	Got relevant data pertaining to the Netflix community, but also had odd non-Unicode characters in the post A lot of relevant diversity LinkedIn posts for Netflix One of the Netflix diversity posts was in German… I can add a google translate algorithm	Use regex to parse out non-Unicode values I can add a google translate algorithm
AMZN	One of the LinkedIn discrimination posts talked about someone’s book on diversity that they published on Amazon, rather than talking about Amazon itself Lots of odd Unicode characters Some talked about the amazon rainforest	Use Named entity parsing

	Some people were advertising the product they were selling on amazon.com	

- o
- o Twitter

Company	Issues	Potential Solutions
KO	Some tweets do not have mentions of coke in KO Diversity There are a lot of random Unicode characters A lot of tweets are replying to other tweets (the @mentions might interfere with sentiment analysis) KO Diversity has mostly relevant tweets, but sometimes it would not output a sentiment score (maybe b/c of lots of noise) A tweet in the KO Pollution is sarcastic… my algo thinks it is positive, when it is negatively reviewing coke The sentiment analysis algorithm is classifying littering as positive ESG, when it is not	‘’
KR	for something like KR Human Rights, it is tough to evaluate whether it is ESG relevant or not since	‘’

	sometimes it is relevant when it only has the keyword “human”	
LMT	Seems to be getting relevant tweets on LMT’s corruption	‘’

- To incorporate this ESG relevancy algorithm, I can assign each article/tweet a relevancy score, and it can proportionally sum this up based on relevancy
- I can also program an ESG relevancy algorithm that checks if a company name and keyword is inside the text, and if the company name was classified as an “organization”
- I can also remove any .com extensions on company names
    - This .com issue affected the search of amazon.com, salesforce.com, etc

2/5 - Cleaning up Code

- Today, I cleaned up and debugged some minor issues within all my code
- I modified the longCompanyName variable, so it did not have extensions such as .com (this makes it better optimized for search)
- I also worked more on the ESG relevancy algorithm by making the criteria a little looser
- I also added link parsing in my Twitter and LinkedIn post algorithms so it would not interfere with sentiment analysis

2/7 - Finishing the ESG Relevancy Algorithm

- Last time, there was a bug in my program where it would say the majority of ESG posts, even the ones that contained the company name and keyword, were irrelevant. This was obviously incorrect
- I found that the error was from the cleanShortPost algorithm, which was deleting capital letters since I forgot to include it in my regex filter

2/8 - Extracting the LinkedIn Author and Articles

- I saw that many of the LinkedIn posts contained an article embedded within them, so I thought it might be helpful to extract it
- I went through the page's HTML to see which tags were there for each content type
- All of this was under class = "feed-shared-article feed-shared-update-v2__content"
    - #for picture first it says class = feed-shared-image feed-shared-image--single-image feed-shared-update-v2__content
    - #for pictures inside says class = "feed-shared-image__container"
    - #for videos it is class= feed-shared-linkedin-video feed-shared-update-v2__content
    - #for videos sub one is feed-shared-linkedin-video__container
    - #for article it class = is feed-shared-article feed-shared-update-v2__content
    - #for bloomber article it is feed-shared-article feed-shared-update-v2__content
- My beautiful soup extraction method does not collect the LinkedIn page's entire HTML
    - https://stackoverflow.com/questions/39379504/extract-entire-html-tag-including-child-tags-with-beautiful-soup - stack overflow which gives more information
    - Unfortunately, adding this beautiful soup extraction caused by code to bug out (it could not get past the login page
- I investigated confidence/certainty levels for sentiment analysis; there was no existing solution for this
    - Actually, Textblob has a subjectivity score which quantifies the amount of personal opinion and factual information contained in the text. The higher subjectivity means that the text contains opinion rather than factual information. TextBlob has one more parameter — intensity. TextBlob calculates subjectivity by looking at the 'intensity'
- I also looked more into Dr. Gloor's suggestion of using sentiment analysis to compute more emotions
    - https://towardsdatascience.com/text2emotion-python-package-to-detect-emotions-from-textual-data-b2e7b7ce1153 - about the textToEmotion package which calculates Happy, Sad, Fear, Surprised, Angry
    - https://glhuilli.github.io/limbic-package.html - limbic calculates sadness, joy, fear, and anger

2/12 - Dividing up Companies and Starting Data Collection

- Many of the companies I was using did not have adequate data, so I realized it would be smarter to train my model on the S&P 500 companies
- I downloaded a list of the S&P 500 companies, organized by market cap
- I divided them up by sector
- I ran my data collectors first for these 5 sectors: technology, energy, consumer discretionary, industrials, and healthcare
- I let my collectors run overnight

2/13 - General Research about ESG Ratings

- http://apjfs.org/resource/global/cafm/2020-10-1.pdf - shows that there is little correlation between Glassdoor ratings and ESG
- https://www.institutionalinvestor.com/article/b1n706z8lqfscs/Where-ESG-Ratings-Fail-The-Case-for-New-Metrics - says correlation b/w MSCI, Sustainalytics, Bloomberg, etc, is only 0.3
- https://www.globalreporting.org/media/nmmnwfsm/gri-policymakers-guide.pdf - corporate accountability and sustainability is a primary concern of the UN
    - That is why the UN has SDGs
    - Almost all companies mention the UN SDGs in their sustainability reports
    - I can measure my algorithm sub-category scores against the UN SDGs

2/14 - Data Collection Continued

- I ran my data collectors for the remaining sectors
- I was able to finish collecting most of my data; however, the programs for the consumer staples sector crashed mid-way because one of the ticker symbols contained a “.”
- I uploaded my collected data to google drive
- I manually began collecting glassdoor ratings pertaining to the companies

2/15 - Beginning Analysis

- I began running my analysis code on the data I collected so far
- It seems to be taking ~15 minutes per company since flair sentiment analysis takes longer to run

2/16 - Running Another Analysis Algorithm Parallel

- I transferred my code and collected data back onto my laptop so I could run my sentiment analysis code from 2 different computer
- I will use a lightweight algorithm such as NLTK to quickly compute results, while the Flair one runs on my other computer
- I found there was a slight bug in the ESG relevancy algorithm, so I made some changes to rectify this
    - I added some word stemming
    - I also added a ticker symbol search option
- Changes to the codebase on my laptop:
    - I changed the main.py file and the esg_relevency_algo file
    - I manually disabled logging
    - I also disabled setting up the LinkedIn scraper
    -

2/16 - Adding Other sentiment Analysis Algorithms

- While the flair sentiment analysis is the most accurate python sentiment analysis library, I still feel it is misclassifying a somewhat large amount of posts
- https://fasttext.cc/docs/en/supervised-tutorial.html - I tried implementing the fast text sentiment analysis library which is fast and uses contextual word embeddings. I was not able to implement it due to the lengthy process for doing so
- I looked into potentially creating my own ESG sentiment analysis algorithm
    - https://link.springer.com/chapter/10.1007/978-3-030-66891-4_10 - a research paper which contains a lexicon for ESG sentiment analysis
    - https://www.paulweiss.com/insights/esg-thought-leadership/esg-lexicon - another lexicon of ESG words (It is like 50 total, so I can try to manually label them)
    - https://towardsdatascience.com/nlp-meets-sustainable-investing-d0542b3c264b - ESG-Bert is an NLP analysis algorithm I think someone has already made

2/18 - Installing ESG-BERT Analysis Repository

- I went to the ESG-BERT GitHub repo, and I began downloading the necessary files in order to run the code
- I also installed the oracle JDK 11 since that was one of the requirements
- While waiting for the download, I also looked at some other potential sentiment analysis approaches
    - I could make my own by employing BERT, ELmo, and contextual word embeddings

    - https://yale-lily.github.io/public/kathan_s2019.pdf - calculates ESG performance based on text
    - https://aws.amazon.com/marketplace/pp/prodview-4doy3qrqm3y6g#support - ESG news sentiment dataset
- Afterward, I converted the Pytorch model to a more easily runnable torchServe .mar file
- I finally started the Torchserve model, but this step was taking excessively long
- I was able to get it running, but unfortunately, I discovered that it served as a method of ESG topic classification, rather than actual sentiment analysis
    - I will likely have to make the sentiment analysis classifier myself
    - This package, however, could serve to be useful in the future

2/21 - Doing Some General Research as well as Formatting Averages into One Cohesive File

- I investigated some more ESG research to come up with more ways I could improve my results
- https://www.bloomberg.com/graphics/2021-what-is-esg-investing-msci-ratings-focus-on-corporate-bottom-line/ - MSCI ratings focus on corporate bottom line more than social responsibility
- https://docs.google.com/document/d/1I27N-djOpOo6KZECPOLSYpmMWK-IZp1-sp5MjWbqwVM/edit?usp=sharing - my lexicon for ESG classification
- I finished computing the averages for each company, and now I want to compile it into one cohesive file
- When I ran my “format data" program, the file did not compile due to a data type error. For some reason, whenever it parsed a file name, it was converted into a byte type rather than a string
- I resolved this issue. There was also a bug caused by a mismatch in title labels which I also resolved

2/22 - Running Machine Learning on my New Data

- I compiled all of my company averages into one cohesive spreadsheet; I ran it so the averages would include all data points, rather than just the ones that were classified as “ESG relevant”
- I ran my machine learning algorithm file to build my algorithm
- Unfortunately, the correlation was good but not phenomenal (between 0.1-0.25). However, if I looked at the difference between the predicted score and the actual

score, I found that this was relatively low (average was b/w 10 and 15 for all algorithms)
    - This means my algorithm was working
- I also added functionality so the predicted and actual scores would maintain the label for their respective company. This lets me see if the algorithm at least performs well for very notable companies like FB, APPL, AMZN, T, etc
    - I found that these large companies had the same relative accuracy as the other companies
- When looking at my graphed data, I found that much of the predicted scores were between 30-70, while the actual scores ranged from 10-80
    - Maybe this was caused by the specific cost function the algorithm tried to optimize
    - However, these results look a bit odd

2/25 - Statistical Analysis

- After meeting with my mentor Dr. Gloor, he gave me tips on some added statistical analysis I must compute
- He told me that I need to have statistically significant results, meaning if I am testing my data on 100 results, then I realistically need a correlation above 0.2 to get statistically significant results ($p<.05$)
- I began playing around with p-values in python and excel

2/26 - Computing the Testing Results

- I reran my machine learning algorithms and randomly selected a train test split
- I found that the XGBoost algorithm performed the best
- Other data is in my metadata spreadsheet