Project Report

- Extracting Summarization Data from EDGAR Green Bond Documents using Python based on NLP

1. Introduction

SEC EDGAR database contains a wealth of information about Commission and the securities industry which is freely available to the public via the Internet. It has a bunch of documents filed electronically, with over 3,000 filings per day. For this reason, it is necessary to analyze these documents automatically by minimizing human effort in order to increase efficiency.

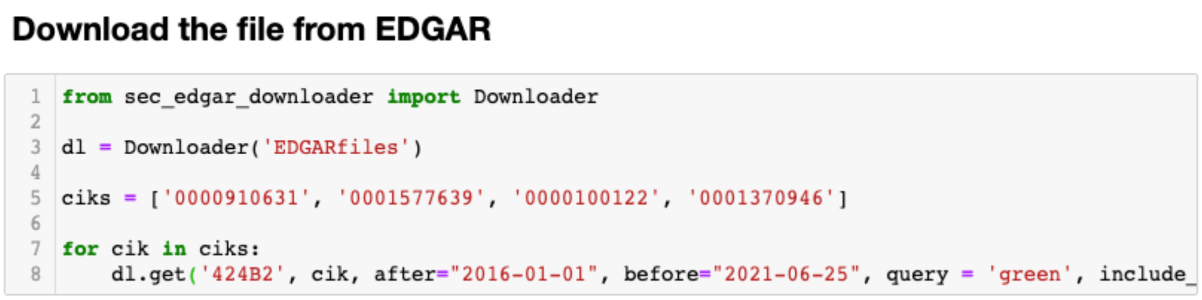
For the first simulation, I focused on the green bond document, and the document form I chose is 424B2 – the prospectus form that a company must file if it is making a primary offering of securities on a delayed basis. Basically, it is about 70–100-page PDF document, and the data that I tried to extract from 424B2 is 1–2-page summarization fact sheet similar to *Climate Bonds* (https://www.climatebonds.net/bond-library) green bond library data.

I used the programming language Python 3 to scrape the texts from PDF filings and extracted the information using NLP skills. I would like to show what I have done in the next section.

2. Data Preparation

2-1. Download the EDGAR file

'sec-edgar-downloader' is a Python package for downloading company filings from the SEC EDGAR database. Searches can be conducted either by stock ticker or Central Index Key (CIK). You can use the SEC CIK lookup tool if you cannot find an appropriate ticker. I downloaded four 424B2 documents using CIK number for this simulation from 4 different company including - *COCA COLA FEMSA SA DE CV, PIEDMONT OPERATING PARTNERSHIP, TUCSON ELECTRIC POWER CO, OWENS CORNING (REORGANIZED) INC.*



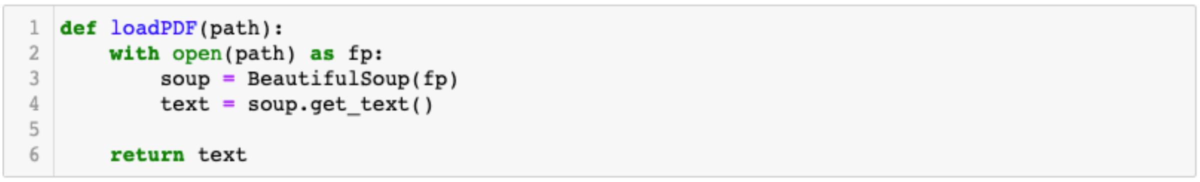
Download the file from EDGAR

```
from sec_edgar_downloader import Downloader

dl = Downloader('EDGARfiles')

ciks = ['0000910631', '0001577639', '0000100122', '0001370946']

for cik in ciks:
    dl.get('424B2', cik, after="2016-01-01", before="2021-06-25", query = 'green', include_
```

2-2. Load the filings and get texts

In order to scrape the text from pdf, I used ‘Beautiful Soup’, which is a Python library for pulling data out of PDF or HTML files. I also got text data only from the document which means I ignored the whole picture data like company logos.

Load the pdf and extraxt text

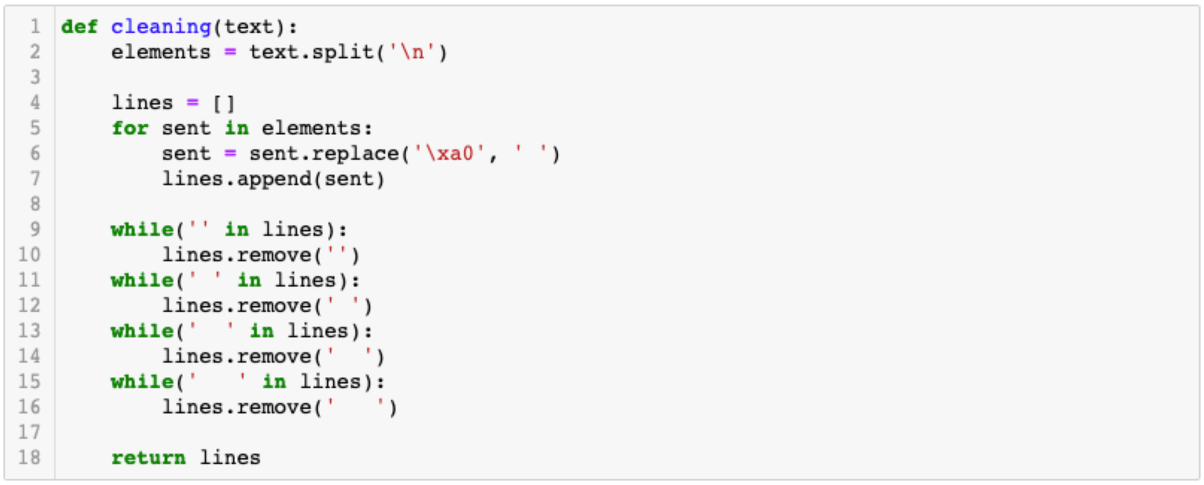


```python
def loadPDF(path):
    with open(path) as fp:
        soup = BeautifulSoup(fp)
        text = soup.get_text()

    return text
```

2-3. Cleaning

At first, when I printed out the text data from whole filings, they aren’t totally organized and dirty. So I needed to clean this data by splitting every sentence and removing unnecessary marks and blanks in the text.

Splitting and cleaning



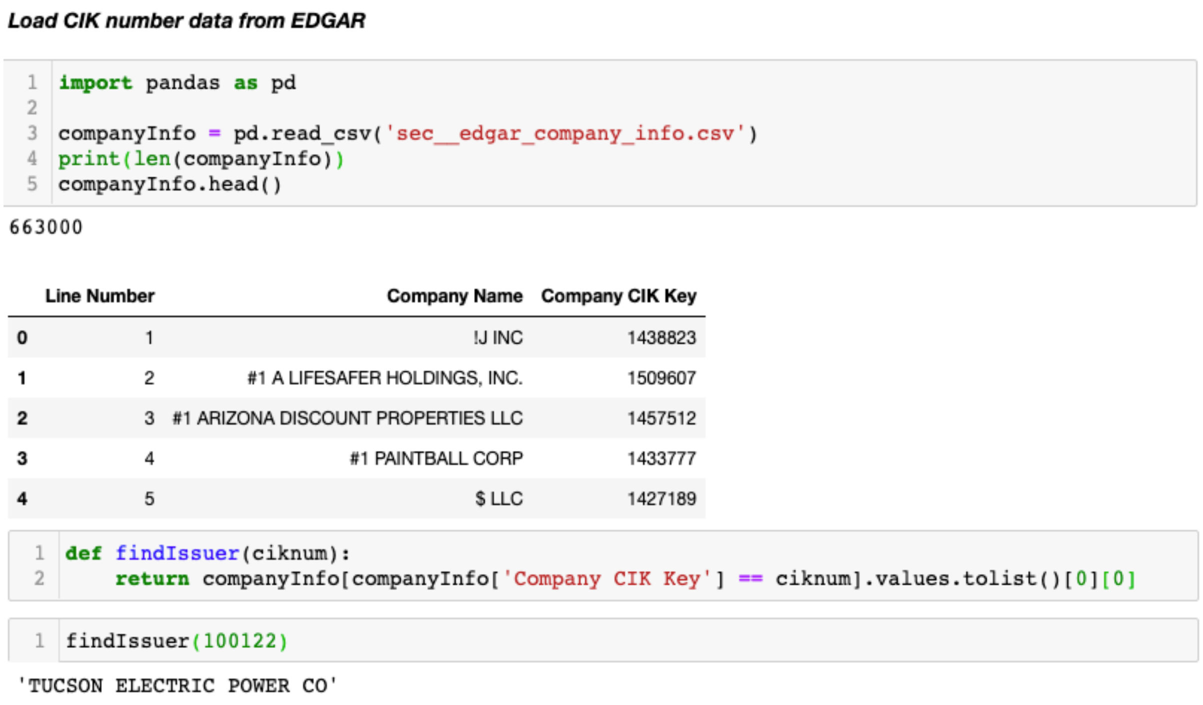
```python
def cleaning(text):
    elements = text.split('\n')

    lines = []
    for sent in elements:
        sent = sent.replace('\xa0', ' ')
        lines.append(sent)

    while('' in lines):
        lines.remove('')
    while(' ' in lines):
        lines.remove(' ')
    while('  ' in lines):
        lines.remove('  ')
    while('   ' in lines):
        lines.remove('   ')

    return lines
```

After I cleaned the whole text, I tried to find out how to get the information I wanted based on Climate Bond fact sheet. In the next chapter, you can see how it worked.

3. Data Extraction

3-1. Find Issuer

It was hard to find the issuer name from the text file because the way of writing the company name is completely diverse by human or electronically such as punctuation, numbers, etc. For this reason, I searched another EDGAR data containing every company name (663K rows) based on CIK keys that I used to download the EDGAR data before.



Load CIK number data from EDGAR

```python
import pandas as pd

companyInfo = pd.read_csv('sec__edgar_company_info.csv')
print(len(companyInfo))
companyInfo.head()
```

```
663000
```

	Line Number	Company Name	Company CIK Key
0	1	!J INC	1438823
1	2	#1 A LIFESAFER HOLDINGS, INC.	1509607
2	3	#1 ARIZONA DISCOUNT PROPERTIES LLC	1457512
3	4	#1 PAINTBALL CORP	1433777
4	5	$ LLC	1427189

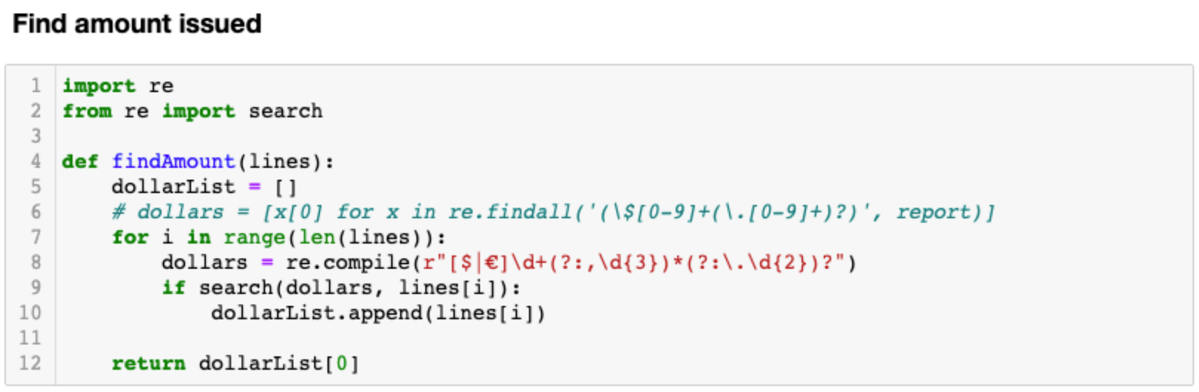
```python
def findIssuer(ciknum):
    return companyInfo[companyInfo['Company CIK Key'] == ciknum].values.tolist()[0][0]
```

```python
findIssuer(100122)
```

```
'TUCSON ELECTRIC POWER CO'
```

3-2. Find the amount issued

I analyzed the every text data and found out that the issue amount comes from the first sentence which has 'dollar amount' mark such as 'U.S.$705,000,000' and '$450,000,000'. In order to find out dollar expression, I used 'regex (regular expression)' which searches the specific pattern in the text.

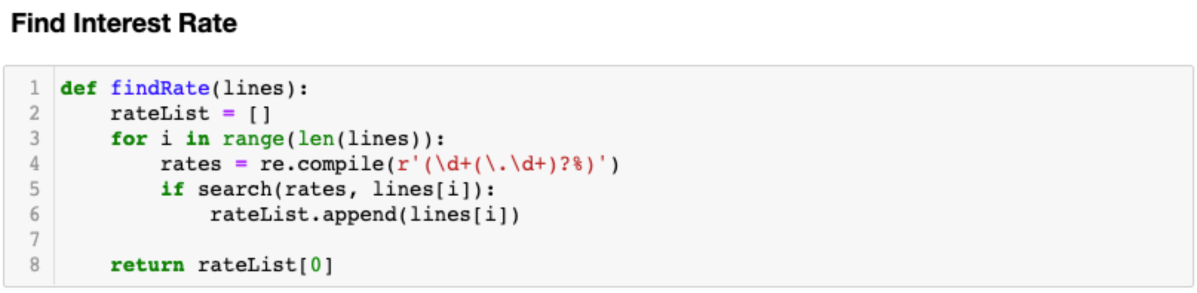


Find amount issued

```python
import re
from re import search

def findAmount(lines):
    dollarList = []
    # dollars = [x[0] for x in re.findall('(\$[0-9]+(\.[0-9]+)?)', report)]
    for i in range(len(lines)):
        dollars = re.compile(r"[$|€]\d+(?:,\d{3})*(?:\.\d{2})?")
        if search(dollars, lines[i]):
            dollarList.append(lines[i])

    return dollarList[0]
```

3-3. Find Interest Rate

Similar to the issue amount, I could find out the interest rate from the first sentence which has '%' expression like '1.850%' or '3.950%'.

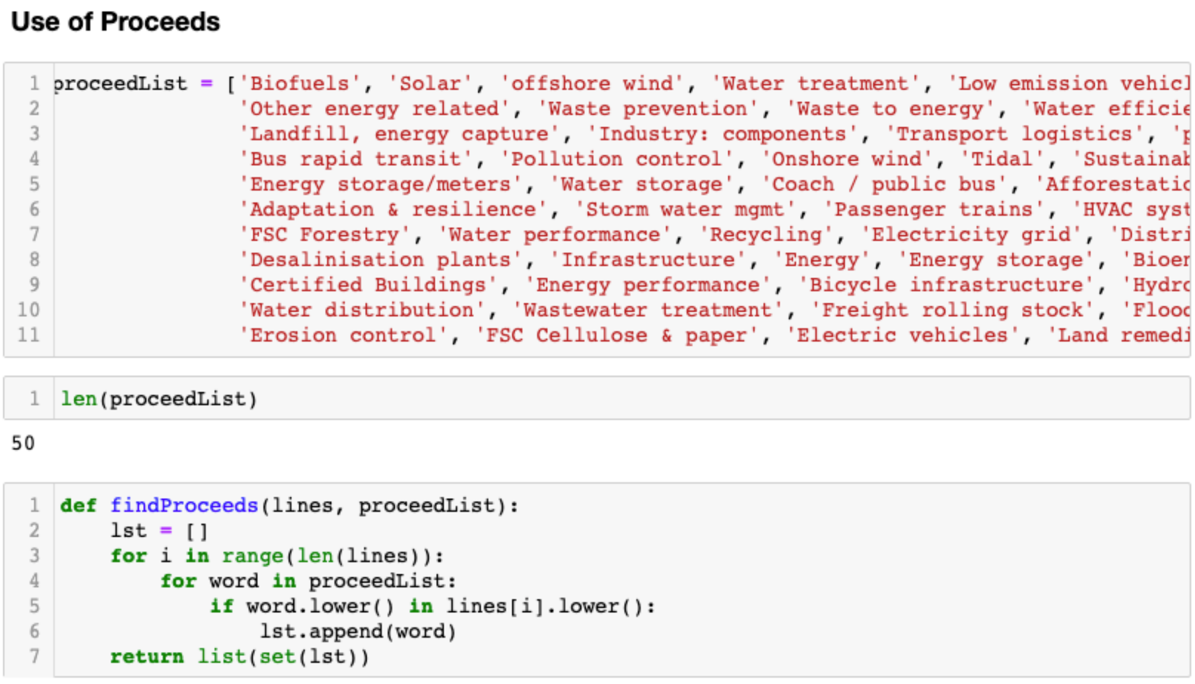


Find Interest Rate

```python
def findRate(lines):
    rateList = []
    for i in range(len(lines)):
        rates = re.compile(r'(\d+(\.\d+)?%)')
        if search(rates, lines[i]):
            rateList.append(lines[i])

    return rateList[0]
```

3-4. Find Use of Proceeds

The number of use of proceed list from *Climate Bond* is 50, so what I did to find the list of use of proceeds is that I tried to find the exact same word in the document. If any sentence contains the word in the 50 use of proceed list, I added them on my proceed list.



Use of Proceeds

```python
proceedList = ['Biofuels', 'Solar', 'offshore wind', 'Water treatment', 'Low emission vehicl
               'Other energy related', 'Waste prevention', 'Waste to energy', 'Water efficie
               'Landfill, energy capture', 'Industry: components', 'Transport logistics', 'p
               'Bus rapid transit', 'Pollution control', 'Onshore wind', 'Tidal', 'Sustainab
               'Energy storage/meters', 'Water storage', 'Coach / public bus', 'Afforestatio
               'Adaptation & resilience', 'Storm water mgmt', 'Passenger trains', 'HVAC syst
               'FSC Forestry', 'Water performance', 'Recycling', 'Electricity grid', 'Distri
               'Desalinisation plants', 'Infrastructure', 'Energy', 'Energy storage', 'Bioer
               'Certified Buildings', 'Energy performance', 'Bicycle infrastructure', 'Hydro
               'Water distribution', 'Wastewater treatment', 'Freight rolling stock', 'Flood
               'Erosion control', 'FSC Cellulose & paper', 'Electric vehicles', 'Land remedi
```

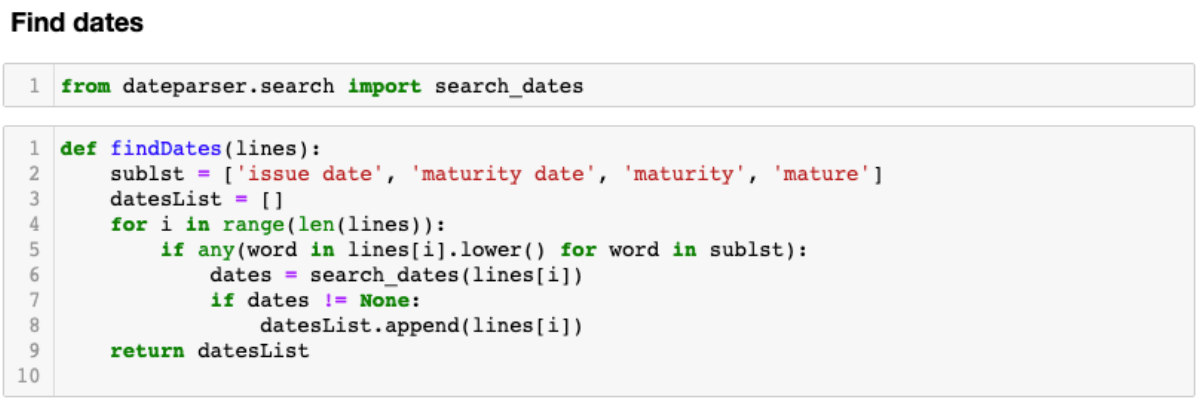
```python
len(proceedList)
```

```
50
```

```python
def findProceeds(lines, proceedList):
    lst = []
    for i in range(len(lines)):
        for word in proceedList:
            if word.lower() in lines[i].lower():
                lst.append(word)
    return list(set(lst))
```

3-5. Find Dates

In this part, I used another Python library ‘dateparser.search’ to find maturity date and issue date for the bond. The way I did is that I firstly tried to find the sentences that contain ‘issue date’, ‘maturity date, ‘maturity’, ‘mature’, and picked the sentences which have ‘date’ expression like ‘August 15, 2029’ or ‘May 15, 2029’.



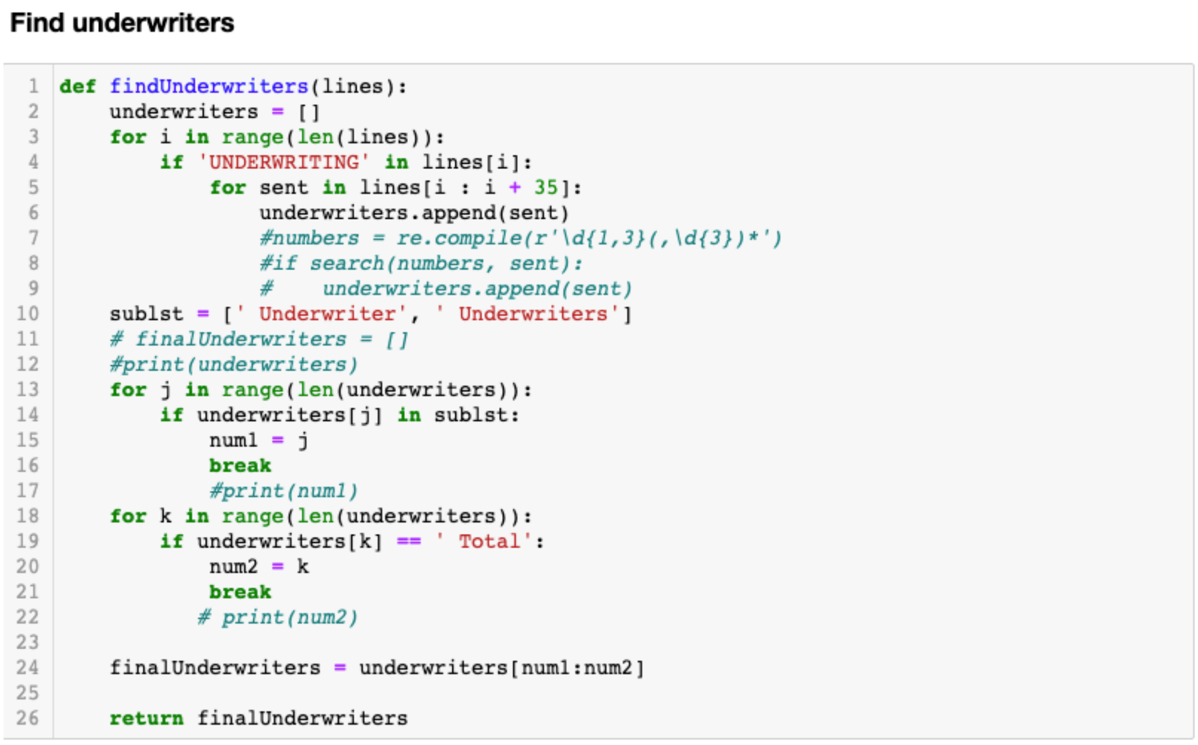
Find dates

```python
from dateparser.search import search_dates
```

```python
def findDates(lines):
    sublst = ['issue date', 'maturity date', 'maturity', 'mature']
    datesList = []
    for i in range(len(lines)):
        if any(word in lines[i].lower() for word in sublst):
            dates = search_dates(lines[i])
            if dates != None:
                datesList.append(lines[i])
    return datesList

```

3-6. Find Underwriters

To find the underwriters and the amount respectively, I analyzed the text data taking a lot of time. Finally, I found out that after ‘UNDERWRITING’ sentence, the document has the underwriters and the amount. So, I caught this part and extracted the underwriter data using NLP and basic Python skills.

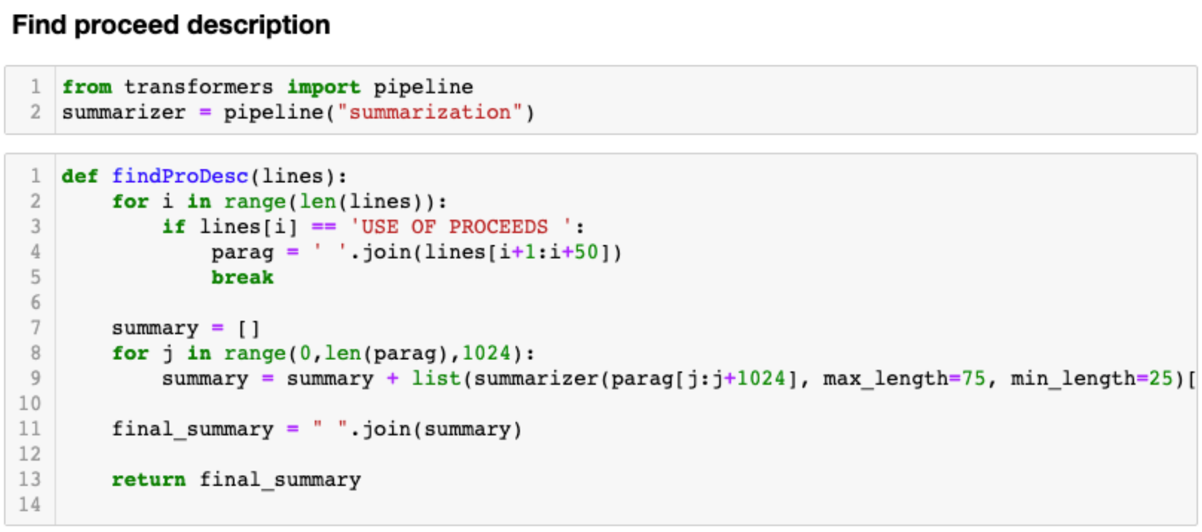


Find underwriters

```python
def findUnderwriters(lines):
    underwriters = []
    for i in range(len(lines)):
        if 'UNDERWRITING' in lines[i]:
            for sent in lines[i : i + 35]:
                underwriters.append(sent)
                #numbers = re.compile(r'\d{1,3}(,\d{3})*')
                #if search(numbers, sent):
                #    underwriters.append(sent)
    sublst = [' Underwriter', ' Underwriters']
    # finalUnderwriters = []
    #print(underwriters)
    for j in range(len(underwriters)):
        if underwriters[j] in sublst:
            num1 = j
            break
            #print(num1)
    for k in range(len(underwriters)):
        if underwriters[k] == ' Total':
            num2 = k
            break
        # print(num2)

    finalUnderwriters = underwriters[num1:num2]

    return finalUnderwriters
```

3-7. Find Proceed Description

The last part of data extraction was Proceed description. The description from original PDF file has 2-3 page - long and large text data. What I did is text summarization with Python. This is an approach that distills a document down to its most important sentences. The idea is very simple. The algorithm simply focuses on the essence of a document. (https://drive.google.com/file/d/1fAgr85WAQU8OXYkwifuF4Ep-LXfrwinv/view?usp=sharing)
I chose the first 50 sentences in Proceed description, split into 5 parts, summarized respectively, and merged them.



Find proceed description

```python
from transformers import pipeline
summarizer = pipeline("summarization")
```

```python
def findProDesc(lines):
    for i in range(len(lines)):
        if lines[i] == 'USE OF PROCEEDS ':
            parag = ' '.join(lines[i+1:i+50])
            break

    summary = []
    for j in range(0,len(parag),1024):
        summary = summary + list(summarizer(parag[j:j+1024], max_length=75, min_length=25)[

    final_summary = " ".join(summary)

    return final_summary

```

4. Results

One of four fact sheets from my code looks like this:

Issuer, CIK#	OWENS CORNING (REORGANIZED) INC.
Issue Amount	$450,000,000
Interest Rate	3.950% Senior Notes due 2029
Use of Proceeds	['Energy', 'Infrastructure', 'Solar']
Dates	['mature on August 15, 2029. We may redeem the notes at any time and from time to time prior to maturity, in whole or in part, for cash at the applicable redemption price, plus accrued and unpaid interest to, but not including, the redemption', 'The notes will be redeemable, in whole at any time or in part from time to time, at our option. If we elect to redeem the notes prior to May 15, 2029 (three months prior to the maturity date of the notes), the redemption price will include', 'a \x93make-whole\x94 premium, plus accrued and unpaid interest to, but not including, the redemption date. If we elect to redeem the notes on or after May 15, 2029 (three months prior to the maturity date of the notes), we will pay a', 'implementation of existing and new Eligible Projects, during the period that begins 24 months prior to the issuance of the notes and ends on the maturity date of the notes. ', 'our 2022 Notes was outstanding and $410 million aggregate principal amount of our 2036 Notes was outstanding. The 2022 Notes bear interest at a rate of 4.20% per annum and mature on December 15, 2022. The 2036 Notes bear interest at a rate', 'of 7.00% per annum and mature on October 31, 2036. Certain affiliates of some of the underwriters hold some of our 2022 Notes and 2036 Notes. In the event that any of the underwriters, together with their respective affiliates, receives at', 'The notes will mature on August 15, 2029. We may issue additional notes from time to time after this offering. See \x93\x97Issuance of Additional Notes.\x94 ', 'Interest on the notes will accrue at a rate per annum of 3.950% from the issue date or from the most recent interest payment date to which', 'accrue for that period from and after the applicable interest payment date, maturity date or redemption date. ', 'notes will be redeemable, in whole at any time or in part from time to time, at our option. If we elect to redeem the notes at any time prior to May 15, 2029 (three months prior to their maturity), we will pay a redemption price equal to the', 'after May 15, 2029 (three months prior to their maturity), the redemption price will equal 100% of the aggregate principal amount of the notes being redeemed, plus accrued and unpaid interest thereon to, but not including, the redemption date.', 'purpose, that the notes matured on May 15, 2029) that would be utilized, at the time of selection and in accordance with customary financial practice, in pricing new issues of corporate', '\x93Treasury Rate\x94 means, with respect to any redemption date, the rate per annum equal to the semi-annual equivalent yield to maturity', ' the original issue and stated maturity

	date or dates; ', 'of a series at the same time, and debt securities of the same series may vary as to interest rate, maturity and other provisions. Unless otherwise provided in the applicable prospectus supplement, the aggregate principal amount of a series may be', 'securities of the same series in the same aggregate principal amount and having the same stated maturity date and other terms and conditions. If so provided in the applicable prospectus supplement, to the extent permitted by law, debt securities of', 'same stated maturity date and other terms and conditions. We may not impose any service charge, other than any required tax or other governmental charge, on the transfer or exchange of debt securities. ', 'Indebtedness under which there has been a Payment Default or the maturity of which has been so accelerated, aggregates $75 million or more; ']
Underwriters, Amount	[' Underwriter', 'Principal Amountof Notes', ' BofA Securities, Inc.', '$', '90,000,000', ' Citigroup Global Markets Inc.', '90,000,000', ' Wells Fargo Securities, LLC', '90,000,000', ' Credit Agricole Securities (USA) Inc.', '27,000,000', ' Goldman Sachs & Co. LLC', '27,000,000', ' BNP Paribas Securities Corp.', '18,000,000', ' Fifth Third Securities, Inc.', '18,000,000', ' HSBC Securities (USA) Inc.', '18,000,000', ' J.P. Morgan Securities LLC', '18,000,000', ' Loop Capital Markets LLC', '18,000,000', ' PNC Capital Markets LLC', '18,000,000', ' Scotia Capital (USA) Inc.', '18,000,000']
Proceed Description	We estimate the net proceeds to us from the sale of the notes will be approximately $445 million . We intend to allocate an amount equal to net proceeds from this offering to one or more Eligible Green Projects . See �Summary�Recent Developments�Tender Offers� for more information regarding the tender offers . A. ty Criteria� means any of the following: Renewable Energy, Energy Efficiency, Glass fiber, and Glass fabric manufacturing of components, for use in wind energy blades, as well as associated research and development . Owens Corning's sustainability team, along with our treasury management team, will identify, evaluate and select Eligible Green Projects based on the Eligibility C-Efficient and Circular Economy Adapted Products, Production Technologies and Processes . Products that save or conserve resources include products that incorporate recycled and/or bio-based/renewable content We intend to allocate an amount equal to the net proceeds from this offering to Eligible Green Projects . As long as the notes are outstanding, we will allocate the proceeds to eligible green projects .

5. Challenge and Conclusion

After comparing the fact sheet from Climate Bonds data, I found out that the extracted information from my code is not 100% the same and should be more improved. Also, I could not find out how to extract the data in the fact sheet such as Country of risk, Bond type, Green bond rating, and so on.

Additionally, Climate Bonds fact sheet has many human reporting sections, but it was impossible to extract the human opinion part from the documents only using Python and NLP skills.

The last challenging point is that I am stick to EDGAR 424B2 – prospectus form only. The code might not work if the document form changes. It is necessary to build up the program which works for every kind of document to improve generalized efficiency.

In order to develop this project, I thought that I need to split the pdf data by its bookmarks, which makes easier to categorize the document and extract the data. Plus, I should find out how to minimize the difference between human opinion and code output by training a huge amount of document data using Machine Learning and Deep Learning.