Key Hiring Trends in Fintech

Summary	In this codelab, you'll analyse the fintech hiring trends in the largest banks in the U.S.
URL	A1_HiringTrends
Category	Analysis
Environment	Jupyter, Tableau, Selenium Web Driver , MS Excel
Status	Draft
Feedback Link	https://github.com/phadkeraj/A1_HiringTrends
Author	Rishika , Akhilesh Tawde , Rohit Jain , Raj Phadke

Introduction

This analysis revolves around finding the job hiring trends in Northern Trust and M&T bank with respect to a specific domain called fintech. With changing demographics, automation efforts and demand for new products and services, large financial institutions are realizing the power of technologies like data science, AI, cloud technologies and machine learning and are heavily investing to upgrade their technological platforms to cater to the upcoming revolution. Technology has been a key player in helping drive this revolution. Things are fast evolving and as we enter 2019, it is interesting to understand the hiring trends in the top financial institutions in the US.

What is fintech?

Fintech is:

- Small , technology enabled new entrant to financial services.
- Application of new technology to automate financial services.

Fintech is the new technology and innovation that aims to compete with traditional financial methods in the delivery of financial services. It is an emerging industry that uses technology to improve activities in finance.The use of smartphones for mobile banking, investing services and cryptocurrency are examples of technologies aiming to make financial services more accessible to the general public.

This codelab will walk you through creating relevant datasets from pdf files, obtaining keywords from the pfds, as well scraping bank websites, to ensure that your analysis of hiring trends is correct .

What you will build

In this codelab, you're going to build analysis of hiring trends in Northern Trust and M&T bank. You'll :

- Extract relevant datasets from pdfs..
- Obtain word count , text rank and tf/idf from csv's.
- Classifying the words on basis of their occurence.
- Scrape data from bank websites
- Analyse the results.

Job No	Institution	URL	Position	Location	List Id	1	2	3	4	5	6
1	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Randolph/Senior-Service-Assoc	Senior Service Ass	New Jersey	1	8	0	0	0	7	8
2	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Randolph/Senior-Service-Assoc	Senior Service Ass	New Jersey	2	17	0	0	8	8	0
3	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Randolph/Senior-Service-Assoc	Senior Service Ass	New Jersey	3	0	8	8	0	7	0
4	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Irondequoit-Hudson-Avenue/Sr	Sr Service Associat	New York	1	12	0	0	0	8	9
5	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Irondequoit-Hudson-Avenue/Sr	Sr Service Associat	New York	2	18	0	0	9	12	0
6	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Irondequoit-Hudson-Avenue/Sr	Sr Service Associat	New York	3	0	12	9	0	8	0
7	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/One-MT-Plaza/Assistant-Senior-	Assistant Senior A	New York	1	1	1	0	0	0	1
8	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/One-MT-Plaza/Assistant-Senior-	Assistant Senior A	New York	2	0	1	0	1	1	0
9	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/One-MT-Plaza/Assistant-Senior-	Assistant Senior A	New York	3	1	1	1	0	0	0
10	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	1	2	2	1	0	0	3
11	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	2	0	2	0	3	2	1
12	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	3	2	2	3	1	0	0
13	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	1	2	2	1	0	0	3
14	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	2	0	2	0	3	2	1
15	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	3	2	2	3	1	0	0
16	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	1	2	2	1	0	0	3
17	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	2	0	2	0	3	2	1
18	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	3	2	2	3	1	0	0
19	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Ma	Technology Manag	New York	1	7	3	1	2	1	1
20	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Ma	Technology Manag	New York	2	1	3	0	1	7	1

What you'll learn

- How to use python to get relevant datasets from pdfs.

- How to clean raw data

- How to obtain word count , tf/idf and text rank from csv files

- How to build a scraper to scrape the data from the bank websites.

What you'll need

- A recent version of Jupyter or python 3.5 or any python coding platform of your choice
- Tableau
- The sample code
- Basic knowledge of Python, Tableau, JavaScript, and MS-Excel

This codelab is focused on analysis of hiring trends. Non-relevant concepts and code blocks are glossed over and are provided for you to simply copy and paste.

Getting set up

Download the Code

Click the following link to download all the code for this codelab:

Download source code

Unpack the downloaded zip file. This will unpack a root folder (A1_HiringTrends), which contains one folder for each step of this codelab, along with all of the resources you will need.

The step-NN folders contain the desired end state of each step of this codelab. They are there for reference. We'll be doing all our coding work in a directory called work.

Install and run Jupyter notebook

While you're free to use your own python platform, this codelab is designed to work well with the Jupyter platform.The best way to install Jupyter is to

download it from a scientific python distribution which also includes scientific python packages. Anaconda is the most common one. If you don't have this app yet ,you can install it from chrome using the following link.

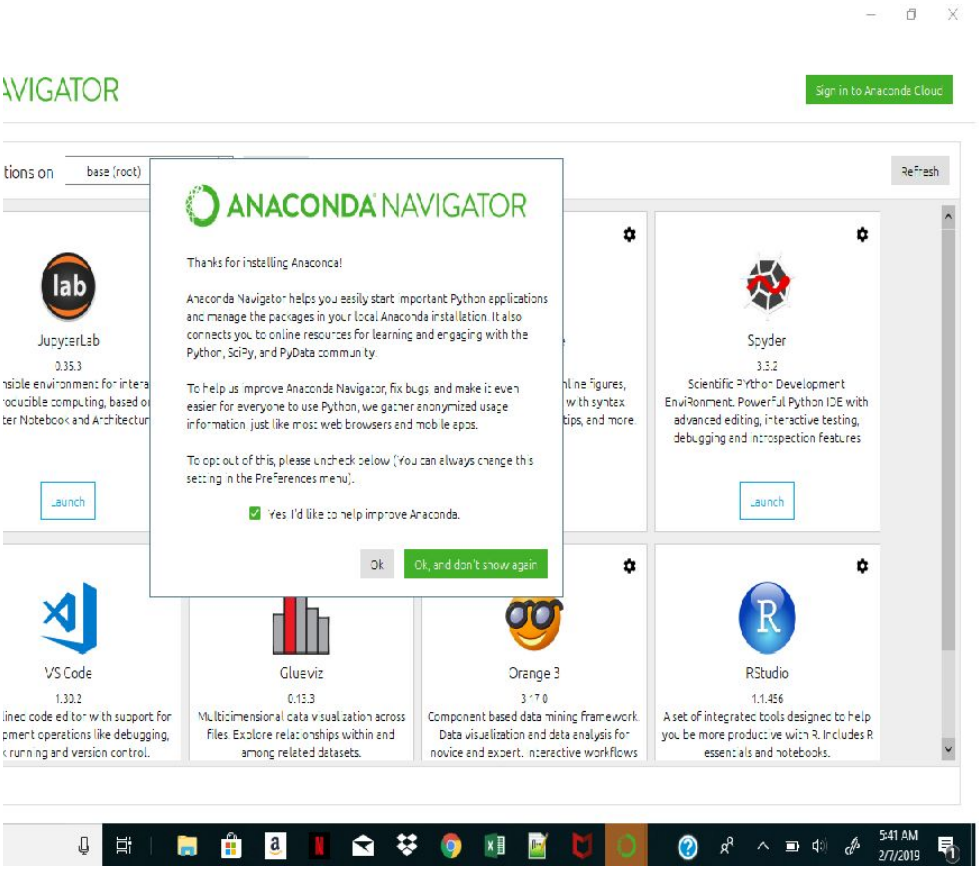
Install Anaconda for installing Jupyter

After installing the anaconda platform click shortcut on the desktop :

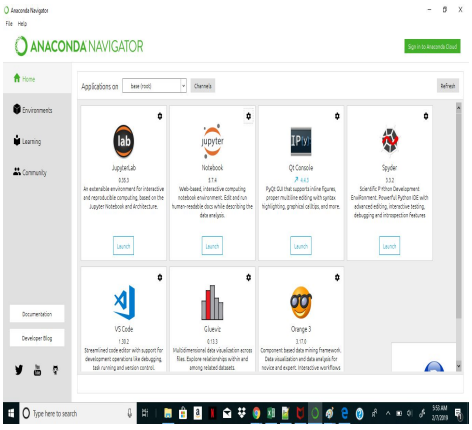
More help: Learn more about Jupyter

Running the Jupyter notebook:

You'll see this dialog next, which allows you to configure your Jupyter notebook:

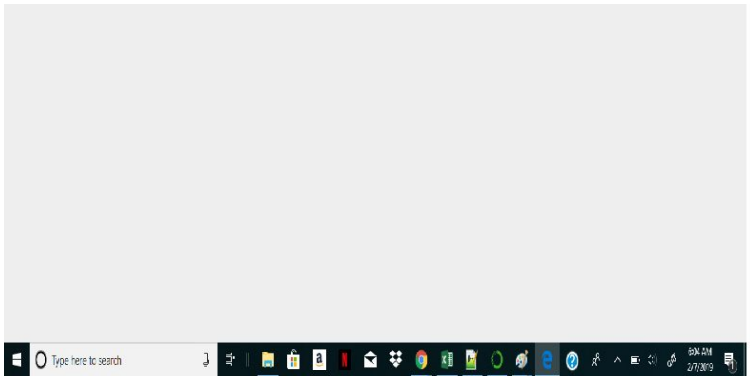


Click the **okay and dont show again** button, and click on the **launch** button under `jupyter` folder. This will enable you to open a jupyter window in your default set browser.



Once the jupyter is open click on :

Then click on the dropdown sign in the **new** button.Then click on Python 3 button.



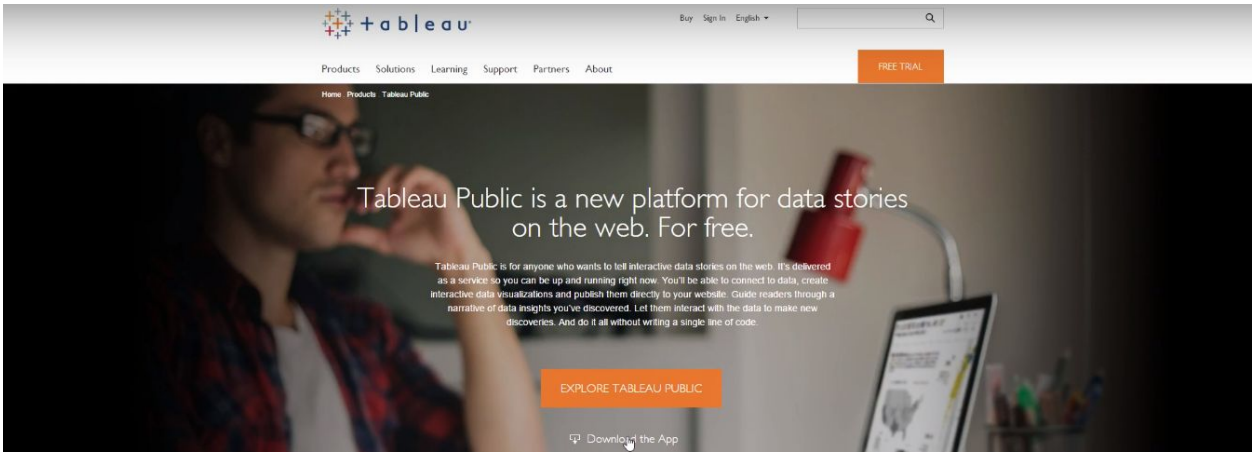
Now visit your work site in your jupyter file and copy the code given above:

Install and run Tableau

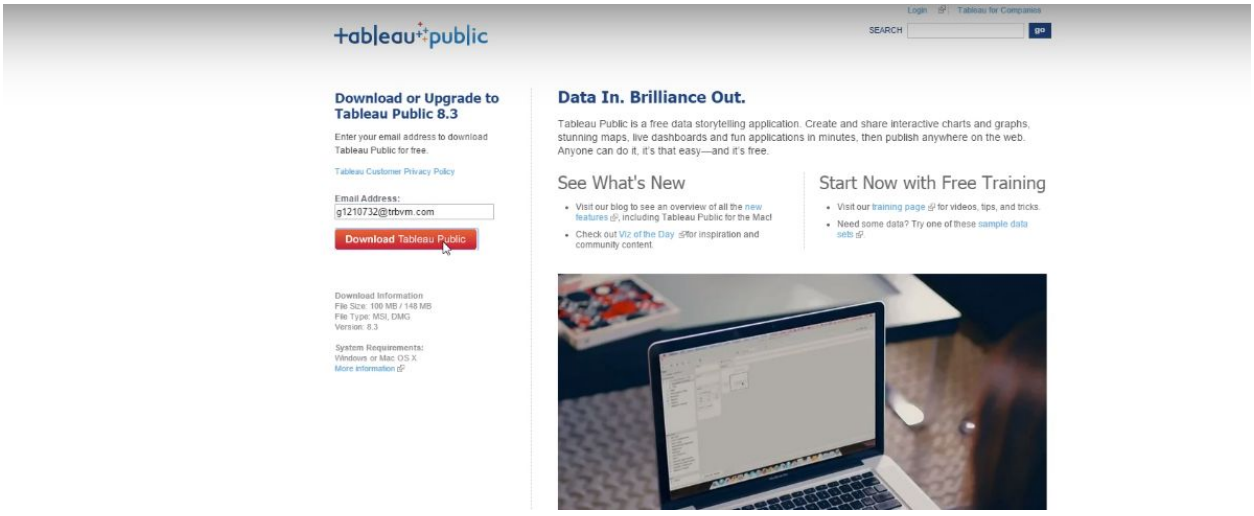
Go to the site whose link is given below and sign up for subscription for free.

Install Tableau

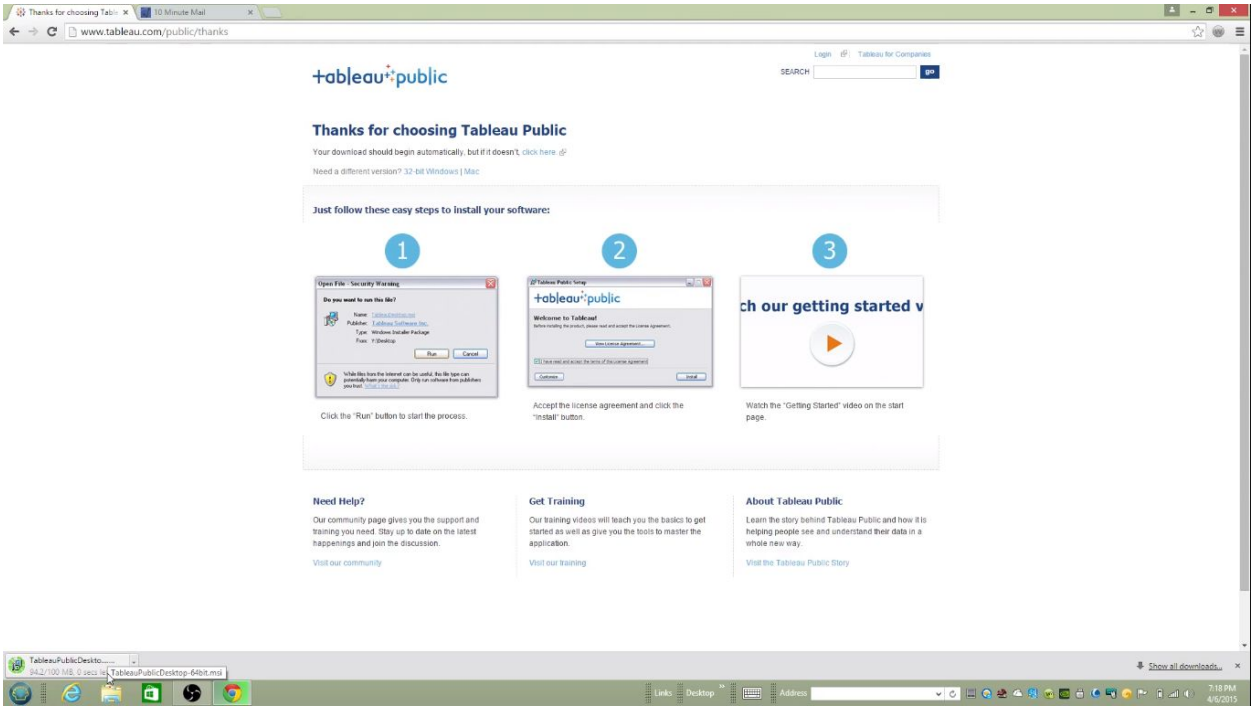
Click on **Tableau Public** button



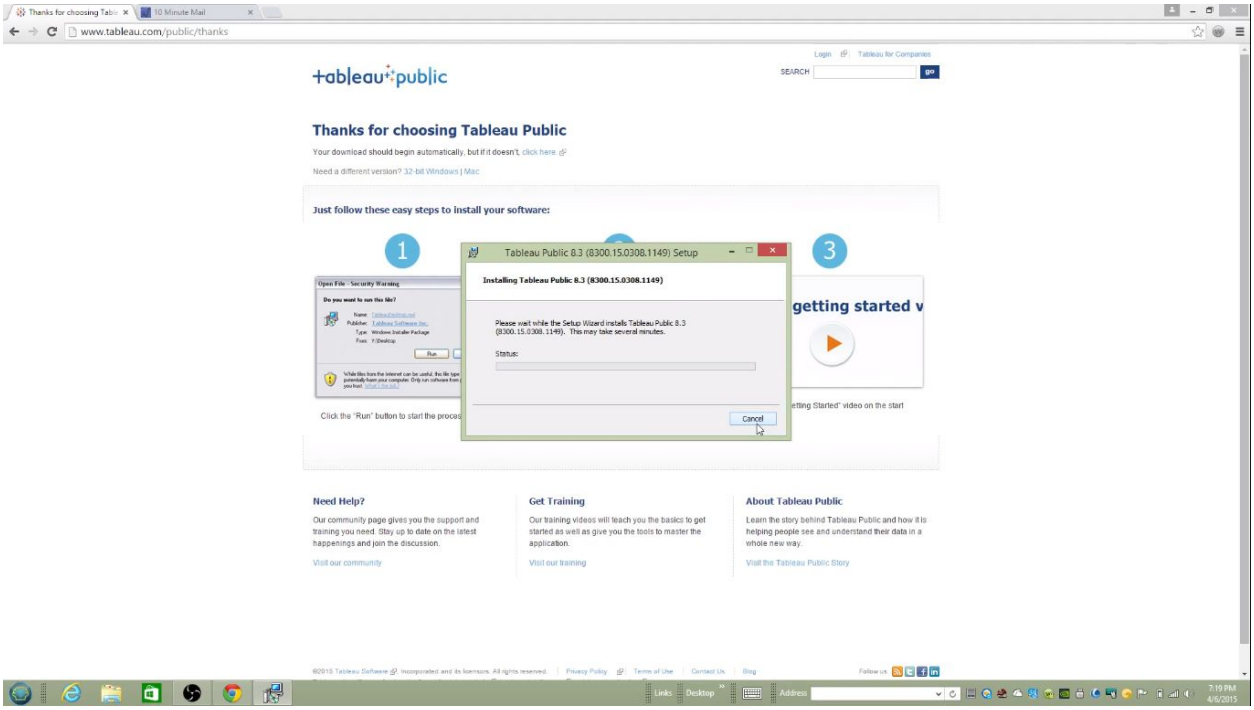
Click on **Download Tableau Public** button



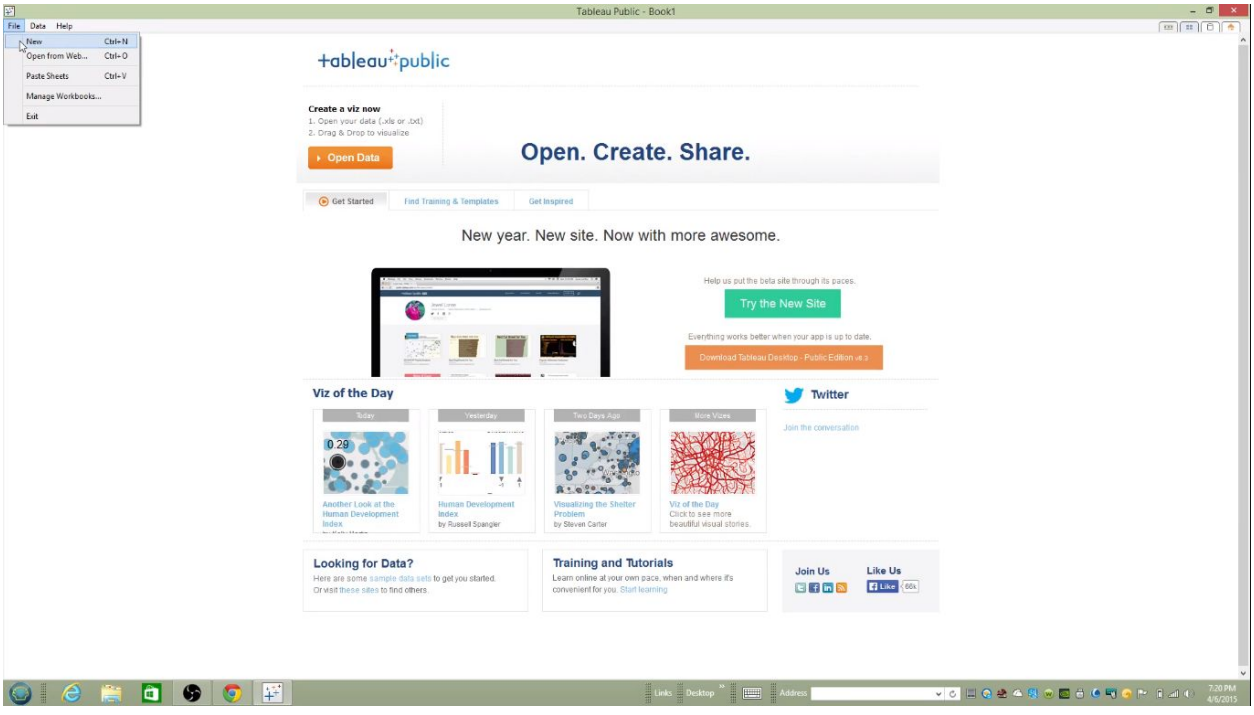
Once the setup is installed **launch the setup**

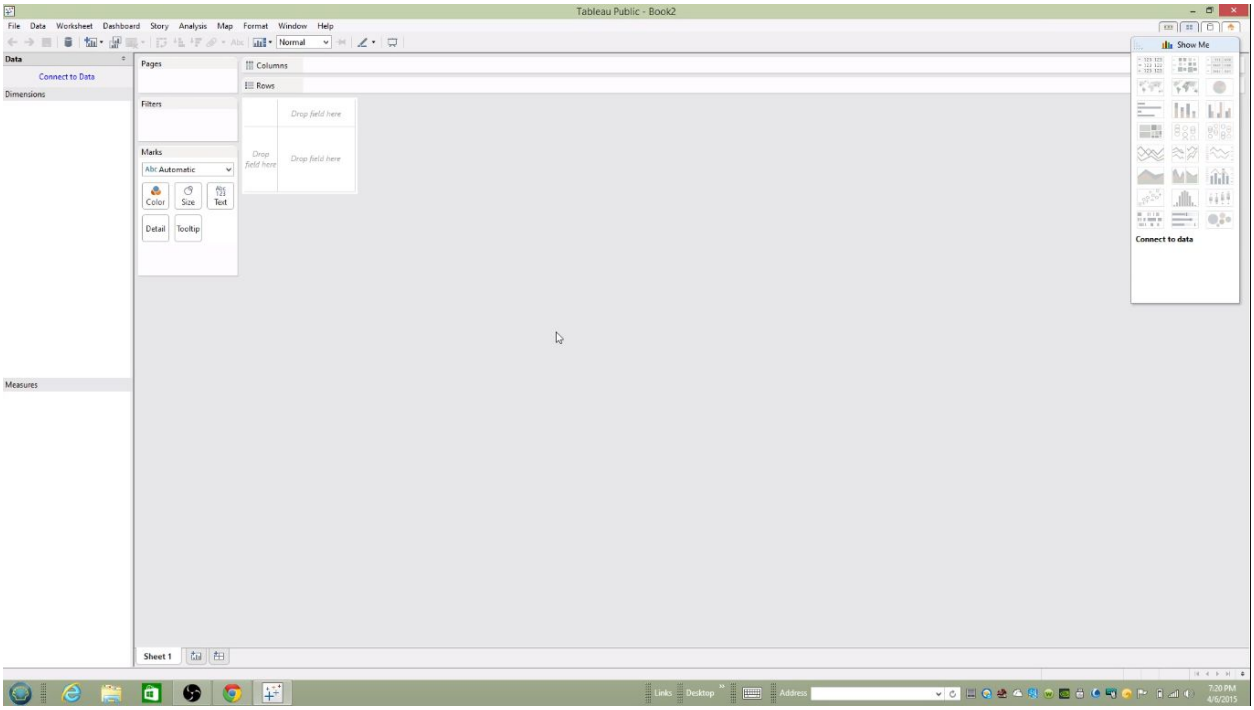


If you have Windows XP or Windows 7 or latest versions of Tableau then there should not be any problem installing it.



Click on **File > New**





From this point forward, all coding of program should be performed using this jupyter notebook unless stated otherwise.

Building a dictionary of top keywords used frequently in Fintech

Procedure to extract text from all the pdfs

Extracting of texts from pdfs using pdf miner

1. Read the pdf files in python using pdf miner.
2. What is a PDF Miner ?

PDFMiner is a tool for extracting information from PDF documents. Unlike other PDF-related tools, it focuses entirely on getting and analyzing text data. PDFMiner allows one to obtain the exact location of text in a page, as well as other information such as fonts or lines. It includes a PDF converter that can transform PDF files into other text formats (such as HTML). It has an extensible PDF parser that can be used for other purposes than text analysis.

3. Convert the pdf into text - The pdf is converted into text form and then the text is tokenized so that data can be worked upon.

What is Tokenization of data

Splitting of data into words is called tokenization.A sentence or data can be split into words using the method **word_tokenize():**

Cleaning the raw datasets

The datasets need to be cleaned as it is raw data and contains a lot of anomalies like stop words, special characters, repeated words, punctuation, tags. Tokenization with pos is done.This is done by lemmatizing the data

Converting text into lower case

The whole text is converted into lower case so that if there are two same words, one with first letter in upper case and the other with first letter in lower case , the algorithm should treat it the same way.

Lemmatizing data

Lemmatization is used to prepare words, texts and documents for further processing. This is done by importing nltk package . Natural Language Tool Kit (nltk) is to make programs that work with natural language.
You can install nltk using pip installer if it is not installed in your Python installation. To test the installation:

1. Open your Python IDE or the CLI interface (whichever you use normally)
2. Type import nltk and press enter if no message of missing nltk is shown then nltk is installed on your computer.

Now after installation, you can use the nltk library for Stemming and Lemmatization using Python.

```
def extract_text_from_pdf(pdf_path):
resource_manager = PDFResourceManager()
fake_file_handle = io.StringIO()
```

```
    codec = 'utf-8'
    laparams = LAParams()
    converter = TextConverter(resource_manager,
fake_file_handle,codec=codec, laparams=laparams)
    page_interpreter = PDFPageInterpreter(resource_manager,
converter)

    with open(pdf_path, 'rb') as fh:
        for page in PDFPage.get_pages(fh):
            page_interpreter.process_page(page)
        text = fake_file_handle.getvalue().lower()
        stop_words = set(stopwords.words("english"))
         #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)
         # remove special characters and digits
        text=re.sub("(\\d|\\W)+"," ",text)
         ##Convert to list from string
        text = text.split()
         #Lemmatisation
        lem = WordNetLemmatizer()
        text = [lem.lemmatize(word,pos="v") for word in
text if not word in stop_words]
        text = " ".join(text)
    # close open handles
    converter.close()
    fake_file_handle.close()
    if text:
        return text
```

Extracting top 100 keywords for different methodologies

Word Count

The count of occurance of every word in all the csv's is called word count. A word count csv file is generated to calculate the frequency of every word. Write the following code in the jupyter file:

```
import io
import nltk
import os
import math
import re
import csv
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfinterp import PDFResourceManager
from pdfminer.pdfpage import PDFPage
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from pdfminer.layout import LAParams

def extract_text_from_pdf(pdf_path):
    resource_manager = PDFResourceManager()
    fake_file_handle = io.StringIO()
    codec = 'utf-8'
    laparams = LAParams()
    converter = TextConverter(resource_manager,
fake_file_handle,codec=codec, laparams=laparams)
    page_interpreter = PDFPageInterpreter(resource_manager,
converter)

    with open(pdf_path, 'rb') as fh:
        for page in PDFPage.get_pages(fh):
            page_interpreter.process_page(page)
        text = fake_file_handle.getvalue().lower()
        stop_words = set(stopwords.words("english"))
         #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)
         # remove special characters and digits
        text=re.sub("(\\d|\\W)+"," ",text)
         ##Convert to list from string
        text = text.split()
         #Lemmatisation
        lem = WordNetLemmatizer()
```

```
        text = [lem.lemmatize(word,pos="v") for word in
text if not word in stop_words]
        text = " ".join(text)
    # close open handles
    converter.close()
    fake_file_handle.close()
    if text:
        return text
filePath=[]
bloblist = []
for file in
os.listdir("C:/ADS/A1_HiringTrends/Data/To_Parse"):
    if file.lower().endswith(".pdf"):

filePath.append(os.path.join("C:/ADS/A1_HiringTrends/Data/T
o_Parse", file))
datatext=''
lem = WordNetLemmatizer()
stem = PorterStemmer()
for path in filePath:
    datatext+='\n'+extract_text_from_pdf(path).lower()
stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(datatext)
filtered_sentence = []
for w in word_tokens:
    if w not in stop_words:
        #w=stem.stem(w)
        filtered_sentence.append(w)
fdist1 = nltk.FreqDist(filtered_sentence)
wtr = csv.writer(open
('C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/WordCount.csv', 'w'), delimiter=',',
lineterminator='\n')
    #writer = csv.writer(f)
x=1
for row in fdist1.most_common(500):
    wtr.writerow([str(x)] + [row[0]])
    x+=1
#f.close()
```

Tf/ Idf

Transform a count matrix to a normalized tf or tf-idf representation

Tf means term-frequency while tf-idf means term-frequency times inverse document-frequency. This is a common term weighting scheme in information retrieval, that has also found good use in document classification.

The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

The formula that is used to compute the tf-idf of term t is tf-idf(d, t) = tf(t) * idf(d, t), and the idf is computed as idf(d, t) = log [n / df(d, t)] + 1 (if `smooth_idf=False`), where n is the total number of documents and df(d, t) is the document frequency; the document frequency is the number of documents d that contain term t. The effect of adding “1” to the idf in the equation above is that terms with zero idf, i.e., terms that occur in all documents in a training set, will not be entirely ignored. (Note that the idf formula above differs from the standard textbook notation that defines the idf as idf(d, t) = log [n / (df(d, t) + 1)]).

If `smooth_idf=True` (the default), the constant “1” is added to the numerator and denominator of the idf as if an extra document was seen containing every term in the collection exactly once, which prevents zero divisions: idf(d, t) = log [(1 + n) / (1 + df(d, t))] + 1.

Furthermore, the formulas used to compute tf and idf depend on parameter settings that correspond to the SMART notation used in IR as follows:

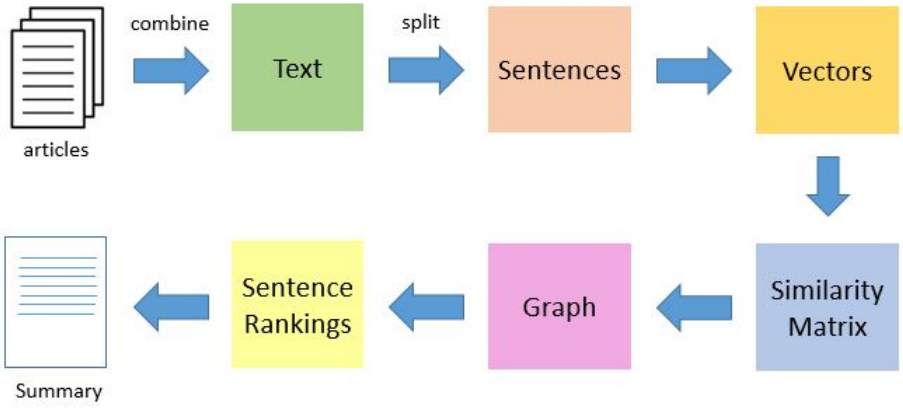
Tf is “n” (natural) by default, “l” (logarithmic) when `sublinear_tf=True`. Idf is “t” when use_idf is given, “n” (none) otherwise. Normalization is “c” (cosine) when `norm='l2'`, “n” (none) when `norm=None`.

```
import io
import nltk
import os

tfidf = TfidfVectorizer(analyzer='word',
stop_words='english', tokenizer=dummy_fun,
    preprocessor=dummy_fun,
    token_pattern=None, use_idf=True)
response = tfidf.fit_transform(tokenized_doc)
weights =
np.asarray(response.mean(axis=0)).ravel().tolist()
weights_df = pd.DataFrame({'Term':
tfidf.get_feature_names(), 'Weight': weights})
weights_df.sort_values(by='Weight',
ascending=False).head(500)
#print(weights_df.sort_values(by='Weight',
ascending=False).head(100))
```

Text Rank

- The first step would be to concatenate all the text contained in the articles
- Then split the text into individual sentences

- In the next step, we will find vector representation (word embeddings) for each and every sentence
- Similarities between sentence vectors are then calculated and stored in a matrix
- The similarity matrix is then converted into a graph, with sentences as vertices and similarity scores as edges, for sentence rank calculation
- Finally, a certain number of top-ranked sentences form the final summary



Reviewing the data manually

The data is reviewed manually as there are still some anomalies left in the lists so you manually select top 100 most relevant words out of all the 3 lists obtained in the previous section.Now we have 3 lists of top 100 keywords.

The code for Text Rank is given below

```
from __future__ import print_function
import io
import nltk
import os
import math
import re
import csv
import editdistance
import itertools
```

```
import networkx as nx
import numpy as np
from pdfminer.converter import TextConverter
from pdfminer.pdfinterp import PDFPageInterpreter
from pdfminer.pdfinterp import PDFResourceManager
from pdfminer.pdfpage import PDFPage
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from pdfminer.layout import LAParams
from summa import keywords
import scipy
import sys
try:
    reload(sys)
    sys.setdefaultencoding('utf-8')
except:
    pass

def extract_text_from_pdf(pdf_path):
    resource_manager = PDFResourceManager()
    fake_file_handle = io.StringIO()
    codec = 'utf-8'
    laparams = LAParams()
    converter = TextConverter(resource_manager,
fake_file_handle,codec=codec, laparams=laparams)
    page_interpreter = PDFPageInterpreter(resource_manager,
converter)

    with open(pdf_path, 'rb') as fh:
        for page in PDFPage.get_pages(fh):
            page_interpreter.process_page(page)
        text = fake_file_handle.getvalue()
        stop_words = set(stopwords.words("english"))
         #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)
         # remove special characters and digits
        text=re.sub("(\\d|\\W)+"," ",text)
         ##Convert to list from string
```

```
        text = text.split()
          #Lemmatisation
        lem = WordNetLemmatizer()
        text = [lem.lemmatize(word,pos="v") for word in
text if not word in stop_words]
        text = " ".join(text)
    if text:
        return text

filePath=[]
for file in
os.listdir("C:/ADS/A1_HiringTrends/Data/To_Parse"):
    if file.lower().endswith(".pdf"):

filePath.append(os.path.join("C:/ADS/A1_HiringTrends/Data/T
o_Parse", file))
datatext=''
for path in filePath:
    datatext+='\n'+extract_text_from_pdf(path).lower()

import codecs
from textrank4zh import TextRank4Keyword

text = datatext


tr4w = TextRank4Keyword()
tr4w.analyze(text=text,lower=True, window=3,
pagerank_config={'alpha':0.85})

#for item in tr4w.get_keywords(200, word_min_len=2):
#    print(item.word, item.weight)

wtr = csv.writer(open
('C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/Textrank.csv', 'w'), delimiter=',',
lineterminator='\n')
x = 1
for item in tr4w.get_keywords(500, word_min_len=2):
    wtr.writerow([str(x)] + [item.word] + [item.weight])
```

```
x+=1
```

Building a scraper to scrape the data from Northern Trust Bank

What is a scraper ?

Web scraping is a method of data mining from web sites that uses software to extract all the information available from the targeted site by simulating human behavior.

Step 1: Find the URL you want to scrape.

Go to Northern Trust Careers

Step 2 : Identify the structure of the sites HTML

Once you've found a site that you can scrape, you can use chrome's developer tools to inspect the site's HTML structure. This is important, because more than likely, you'll want to scrape data from certain HTML elements, or elements with specific classes or IDs. With the inspect tool, you can quickly identify which elements you need to target.

Step 3 : Install Beautiful Soup and Requests

There are other packages and frameworks, like Scrapy. But Beautiful Soup allows you to parse the HTML in a a beautiful way, so that's what I'm going to use. With Beautiful Soup, you'll also need to install a Request library, which will fetch the url content.

If you aren't familiar with it, the Beautiful Soup documentation has a lot of great examples to help get you started as well.

To install these two packages, you can simply use the pip installer.

```
$ pip install requests
```

and then…

```
$ pip install beautifulsoup4
```

Step 4: Web Scraping Code .

Here's how you structure the code:

```
import requests
import csv
from bs4 import BeautifulSoup
import re
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import lxml
import csv
from collections import defaultdict
import pandas as pd
import numpy as np
```

```
#from lxml import etree

#import urllib
from urllib.request import urlopen
pages = []
abcd = {}
qq=[]
count=0
testDict = defaultdict(int)
for i in range(1, 26):
    url =
'https://careers.northerntrust.com/jobs/search/8420211/page
' + str(i) + '.htm'
    #print(url)
    pages.append(url)
dataset=[]
counter = 1
link=[]
#print(pages)
count = 0
for item in pages:
    page = requests.get(item)
    soup = BeautifulSoup(page.text, 'html.parser')

#print(soup)

    hash_links = soup.find(class_='job_filters_toggle
jJobFiltersToggle')
    hash_links.decompose()

    job_name_list = soup.find(class_='info_listings
jJobResultsListHldr')
    job_name_list_items = job_name_list.find_all('a')

    for job_name in job_name_list_items:
        #print(job_name.prettify())
        links = job_name.get('href')
        link.append(links)
        page1 = requests.get(links)
        soup1 = BeautifulSoup(page1.text, 'html.parser')
```

```
        ignore =
soup.find("div",["flg_hldr","info_box","jFooter compact"])
        ignore.decompose()
        #ignore1 = soup.find(class_='info_box')
        #ignore1.decompose()
        #ignore2 = soup.find(class_='jFooter compact')
        #ignore2.decompose()
        content = soup1.find(class_='jBlock')
        data = content.get_text()
        delete = soup1.find("span","field_value
font_header_light")
        delete.decompose()
        idjob = soup1.find("span","field_value")
        job_id=idjob.get_text()
        qq.append(job_id)
        stop_words = set(stopwords.words("english"))
         #remove tags
        data=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",data)
         # remove special characters and digits
        data=re.sub("(\\d|\\W)+"," ",data)
         ##Convert to list from string
        data = data.split()
         #Lemmatisation
        lem = WordNetLemmatizer()
        data = [lem.lemmatize(word) for word in data if not
word in stop_words]
        data = " ".join(data)
        dataset.append(data)
```

Building a scraper to scrape the data from M&T Bank

Installation of selenium web driver

```
pip install -U selenium
```

Why selenium web driver is needed ?

The data on the website of M&T bank is generated dynamically by Java Script Code so Selenium Driver is used to automate and control a full fledged web browser.

Why is there a need for timer in this website?

There is dynamic loading of content in the M&T bank website due to which all the links can not be transversed in a single transversal so you have to use a timer , counter and a scroller to access all the links.

What is PhantomJS ?

PhantomJS is a scripted headless browser used for automating web page interaction. PhantomJS along with selenium is used for headless automation which enables you to run the test for different websites.

How to build the scraper ?

Visit M&T Careers by going to the link given below

Go to M&T Careers

- Use PhantomJS to fetch the inner content by running the java script code.
- Scroll down to the bottom of the page through selenium web driver.
- Traverse through each of the links generated through Step 3.
- Fetch content such as job description , job position , locations from every traversed links.

The code for scraping information for M&T bank is given below

```
from selenium import webdriver
from bs4 import BeautifulSoup as bs
import time
import sys
import csv
import re
import pandas as pd
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer

def
fetchJobData(link,fileNames,destFile,institutionName,counte
r):
    driver =
webdriver.PhantomJS('C:/ADS/A1_HiringTrends/Support/phantom
js/phantomjs')
    driver.get(link)
    counterVal=counter

    time.sleep(10)
    htmlJobContent = driver.execute_script("return
document.body.innerHTML;")
    soup = bs(htmlJobContent,'html.parser')

divClass=soup.find('div',attrs={'class':'GWTCKEditor-Disabl
ed','dir':'ltr'})
    headVal=divClass.text
    ulClass=soup.find_all('ul',attrs={'class':'WPFO
WGGO','role':'presentation'})[2]
    try:
        location=str(ulClass)
        loc=location.find('<b>Location</b>')
        loc=location.find('</h1>',loc)+5
        loc2=location.find('<p',loc)
        location=location[loc:loc2]
        data=''
        for x in ulClass:
            data+=' '+x.text
        data=data.lower()
        driver.close()
```

```
        driver.quit()
        stop_words = set(stopwords.words("english"))
        data=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",data)
          # remove special characters and digits
        data=re.sub("(\\d|\\W)+"," ",data)
          ##Convert to list from string
        data = data.split()
        #Lemmatisation
        stop_words = set(stopwords.words("english"))
        lem = WordNetLemmatizer()
        data = [lem.lemmatize(word,pos="v") for word in
data if not word in stop_words]
        data = " ".join(data)
        id=1

        for fileN in fileNames:

valToAppend=[counterVal,institutionName,link,headVal,locati
on,id]
            with open(fileN, newline='') as myFile:
                reader = csv.reader(myFile)
                for row in reader:
                    count=0
                    valtosearch=str(row[1])
                    count = sum(1 for match in
re.finditer(valtosearch.lower(), data.lower()))
                    valToAppend.append(str(count))
            with open(destFile, 'a') as f:
                writer = csv.writer(f)
                writer.writerow(valToAppend)
                counterVal+=1
            id+=1
    except:
        with open('C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/Unresolved.csv', 'a') as unresolved:
            writer = csv.writer(unresolved)
            writer.writerow([link])
        driver.close()
        driver.quit()
    return counterVal
```

```
driver =
webdriver.PhantomJS('C:/ADS/A1_HiringTrends/Support/phantom
js/phantomjs')
driver.get("https://mtb.wd5.myworkdayjobs.com/MTB")
# This will get the initial html - before javascript
# This will get the html after on-load javascript
time.sleep(10)
html2 = driver.execute_script("return
document.body.innerHTML;")

soup = bs(html2,'html.parser')
totalcount=0
for i,tag in
enumerate(soup.find_all('span',attrs={'class':'gwt-InlineLa
bel WF2N WG2N'})):
    totalcount=int(tag.text.replace("Results","").strip())
    break
print(totalcount)
count=0
totalcount+=50
while count<totalcount:
    driver.execute_script("window.scrollTo(0,
document.body.scrollHeight);")
    time.sleep(5)
    count+=50
html2 = driver.execute_script("return
document.body.innerHTML;")
driver.close()
soup = bs(html2,'html.parser')
driver.quit()

link=''
unsolved=[]
counter=1
for i,tag in
enumerate(soup.find_all(['div','span'],attrs={'class':['gwt
-Label WOTO WISO','gwt-InlineLabel WM-F WLYF']})):
    if(i%2==0):
        link=''
```

```
        if tag.has_attr('aria-label'):
            link=tag['aria-label']
        else:
            continue
    else:
        if(link==''):
            continue
        else:

link='https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/'+tag
.text.split('|')[0].strip().replace('
','-').replace('.','').replace('&','').replace('(','').repl
ace(')','').replace(',','').replace('/','').replace('\\',''
)+'/'+link.replace('- Bankruptcy','').replace('
','-').replace('.','').replace('&','-').replace('(','-').re
place(')','-').replace(',','-').replace('/','-').replace('\
\','-')+'_'+tag.text.split('|')[1].strip()

fileNames=['C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/Textrank_top100.csv','C:/ADS/A1_HiringTrends/Dat
a/Generated/By
Algorithms/TF-IDF_top100.csv','C:/ADS/A1_HiringTrends/Data/
Generated/By Algorithms/WordCount_top100.csv']

destFile='C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/M&T_Scraped.csv'
            instName='M&T Bank'
            try:
                print(link)

counter=fetchJobData(link,fileNames,destFile,instName,count
er)
            except:
                print(sys.exc_info()[0])
                try:
                    print(link+'-1')

counter=fetchJobData(link+'-1',fileNames,destFile,instName,
counter)
                except:
```

```
                    unsolved.append(link)
                    with
open('C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/Unresolved.csv', 'a') as unresolved:
                        writer = csv.writer(unresolved)
                        writer.writerow([link])

print('*'*50+'\nUnsolved:'+link+'\n'+'*'*50)
                    print(sys.exc_info()[0])
print('End')
```

Fetching requirements for analysis from the websites

Comparison of data collected from the bank websites to the lists obtained in former sections.

The data obtained from the bank websites is compared with the data obtained from the former lists of fintech and the count of occurence of each word is appended in a new csv file.

```
import requests
import csv
from bs4 import BeautifulSoup
import re
```

```
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
import lxml
import csv
from collections import defaultdict
import pandas as pd
import numpy as np

#from lxml import etree

#import urllib
from urllib.request import urlopen
pages = []
abcd = {}
qq=[]
count=0
testDict = defaultdict(int)
for i in range(1, 26):
    url =
'https://careers.northerntrust.com/jobs/search/8420211/page
' + str(i) + '.htm'
    #print(url)
    pages.append(url)
dataset=[]
counter = 1
link=[]
#print(pages)
count = 0
for item in pages:
    page = requests.get(item)
    soup = BeautifulSoup(page.text, 'html.parser')

#print(soup)

    hash_links = soup.find(class_='job_filters_toggle
jJobFiltersToggle')
    hash_links.decompose()

    job_name_list = soup.find(class_='info_listings
jJobResultsListHldr')
```

```
    job_name_list_items = job_name_list.find_all('a')

    for job_name in job_name_list_items:
        #print(job_name.prettify())
        links = job_name.get('href')
        link.append(links)
        page1 = requests.get(links)
        soup1 = BeautifulSoup(page1.text, 'html.parser')
        ignore =
soup.find("div",["flg_hldr","info_box","jFooter compact"])
        ignore.decompose()
        #ignore1 = soup.find(class_='info_box')
        #ignore1.decompose()
        #ignore2 = soup.find(class_='jFooter compact')
        #ignore2.decompose()
        content = soup1.find(class_='jBlock')
        data = content.get_text()
        delete = soup1.find("span","field_value
font_header_light")
        delete.decompose()
        idjob = soup1.find("span","field_value")
        job_id=idjob.get_text()
        qq.append(job_id)
        stop_words = set(stopwords.words("english"))
         #remove tags
        data=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",data)
         # remove special characters and digits
        data=re.sub("(\\d|\\W)+"," ",data)
         ##Convert to list from string
        data = data.split()
         #Lemmatisation
        lem = WordNetLemmatizer()
        data = [lem.lemmatize(word) for word in data if not
word in stop_words]
        data = " ".join(data)
        dataset.append(data)
        count+=1

print(count)
```

```
#dataset = ['hi my name is financial customer financial',
'hii hii hii nmy name is customer customer customer']
#print(dataset)
x = 1
abc=[]
df = pd.read_csv("C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/WordCount_top100.csv",header=None,names=["0","wo
rds"])
#df = pd.read_csv("Z:/ADS/a/textrank_final.csv")
with open('C:/ADS/A1_HiringTrends/Data/Generated/By
Algorithms/WordCount_top100.csv', newline='') as myFile:
    reader = csv.reader(myFile)
    for row in reader:
        abc.append(row[1])
    #for row in reader:
        #print(row[1])
#print(abc)
for text in dataset:
        #print(text)
    thisline = text.split(" ");
    #print(thisline)
    for q in abc:
        #print(row)
        for wd in thisline:
            #print(wd)
            if q == wd:
                if wd in abcd:
                    abcd[wd] +=1

                else:
                    abcd.update({wd:1})
                        #abcd[wd] = 1
    #print(abcd)
    #x = 0
    #df =
pd.read_csv("Z:/ADS/a/textrank_final.csv",header=None,names
=["ranking","words","rate"])
    df[x]= df['words'].map(abcd)
    x +=1
    abcd.clear()
```

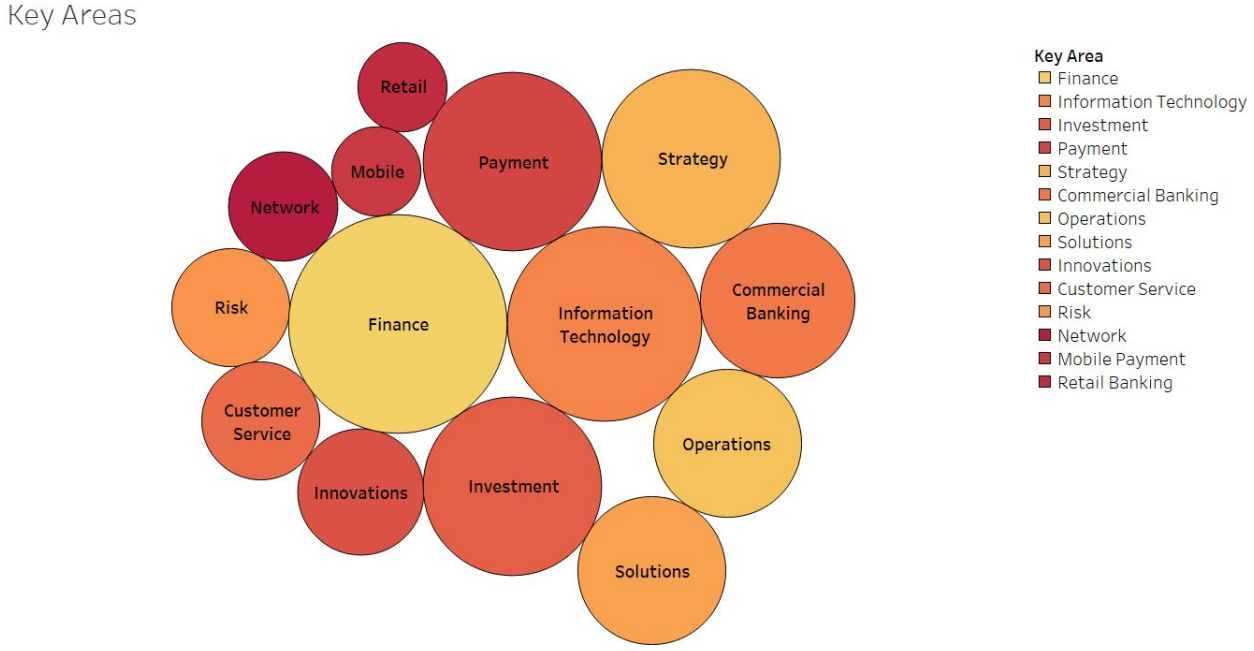
```
#df
#df2 = df.drop(df.column2)
df2_transposed = df.transpose() # or df2.transpose()
df2_transposed2 =
df2_transposed.drop(df2_transposed.index[1])

df2_transposed2.insert(loc=0, column='Job No',value='')
df2_transposed2.insert(loc=1,
column='Institutation',value='')
df2_transposed2.insert(loc=2, column='URL(url of job
posting)',value='')
df2_transposed2.insert(loc=3, column='List Id',value='')
df2_transposed2
q=1
m=1
o=1
for a in qq:
    df2_transposed2.iloc[m,0] = a
    m+=1
for l in link:
    df2_transposed2.iloc[q,2] = l
    q+=1
inst = 'northern trust'
for o in range(o,count+1):
    df2_transposed2.iloc[o,3] = '1'
    df2_transposed2.iloc[o,1] = inst
    o+=1
df2_transposed2 = df2_transposed2.replace(np.nan, 0)
df2_transposed2.to_csv("C:/ADS/A1_HiringTrends/Data/Generat
ed/By
Algorithms/final_files/WordCount_final_file.csv",index=Fals
e, encoding='utf8')
```

The CSV file generated should look like the following

Job No	Institution	URL	Position	Location	List Id	1	2	3	4	5	6
1	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Randolph/Senior-Service-Assoc	Senior Service Ass	New Jersey	1	8	0	0	0	7	8
2	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Randolph/Senior-Service-Assoc	Senior Service Ass	New Jersey	2	17	0	0	8	8	0
3	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Randolph/Senior-Service-Assoc	Senior Service Ass	New Jersey	3	0	8	8	0	7	0
4	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Irondequoit-Hudson-Avenue/Sr	Sr Service Associat	New York	1	12	0	0	0	8	9
5	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Irondequoit-Hudson-Avenue/Sr	Sr Service Associat	New York	2	18	0	0	9	12	0
6	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Irondequoit-Hudson-Avenue/Sr	Sr Service Associat	New York	3	0	12	9	0	8	0
7	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/One-MT-Plaza/Assistant-Senior-	Assistant Senior A	New York	1	1	1	0	0	0	1
8	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/One-MT-Plaza/Assistant-Senior-	Assistant Senior A	New York	2	0	1	0	1	1	0
9	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/One-MT-Plaza/Assistant-Senior-	Assistant Senior A	New York	3	1	1	1	0	0	0
10	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	1	2	2	1	0	0	3
11	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	2	0	2	0	3	2	1
12	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	3	2	2	3	1	0	0
13	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	1	2	2	1	0	0	3
14	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	2	0	2	0	3	2	1
15	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	3	2	2	3	1	0	0
16	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	1	2	2	1	0	0	3
17	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	2	0	2	0	3	2	1
18	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Tea	Technology Team	New York	3	2	2	3	1	0	0
19	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Ma	Technology Manag	New York	1	7	3	1	2	1	1
20	M&T Bank	https://mtb.wd5.myworkdayjobs.com/en-US/MTB/job/Lafayette-Court/Technology-Ma	Technology Manag	New York	2	1	3	0	1	7	1

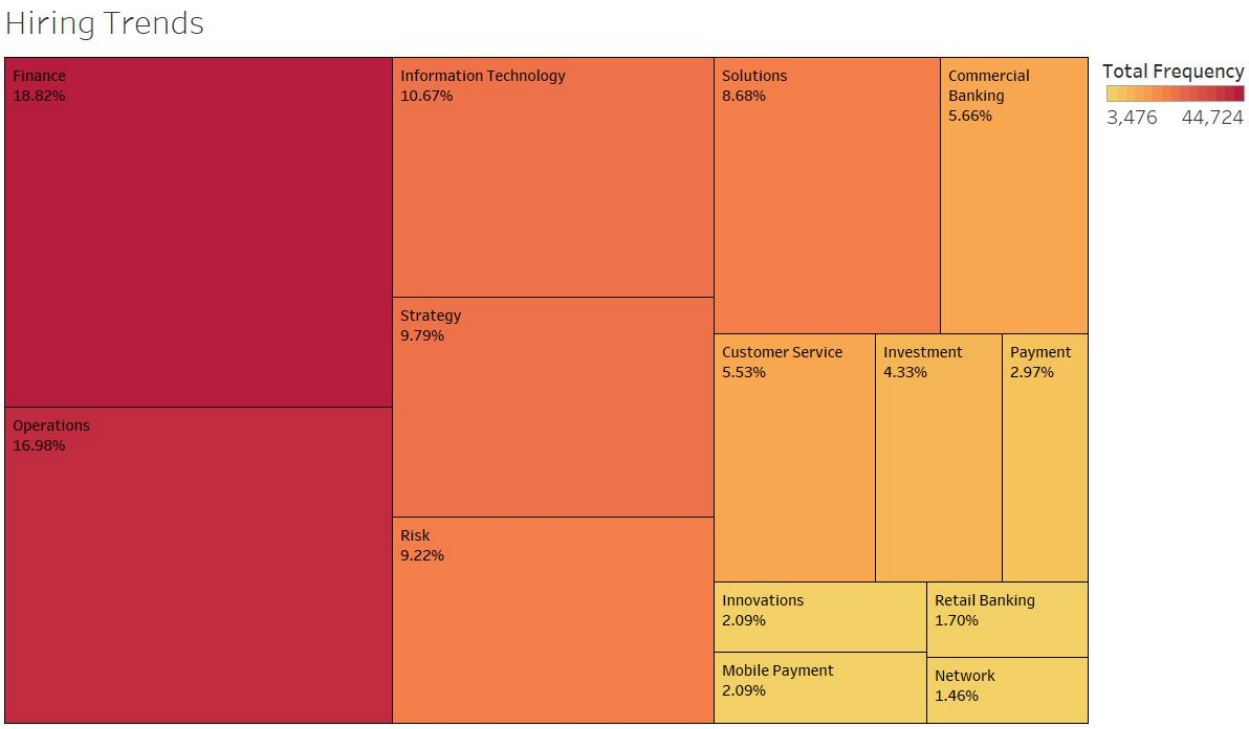
Key Areas

Ordered the key areas based on the number of words .This is done by slicing the csv files made in the previous sections. For this analysis we would need The bucket ID and the key areas.For Example finance had the highest count so the size of finance word would be largest and then the size of words keep decreasing according to their count.



Key Hiring Trends

After analysing the words on the basis of their count.Key areas can be calculated on the basis of the frequency of words. Compare the parsed data obtained from scraping the websites to our former csv's. Then obtain the frequency of every word.The percentage of the total of the count of all the words is calculated.



Word Clouds

The words with the most frequency are the largest in size.The data is filtered on the basis of frequency of the words occurrence in both the institutions.

WordClouds

Institution

M&T Bank

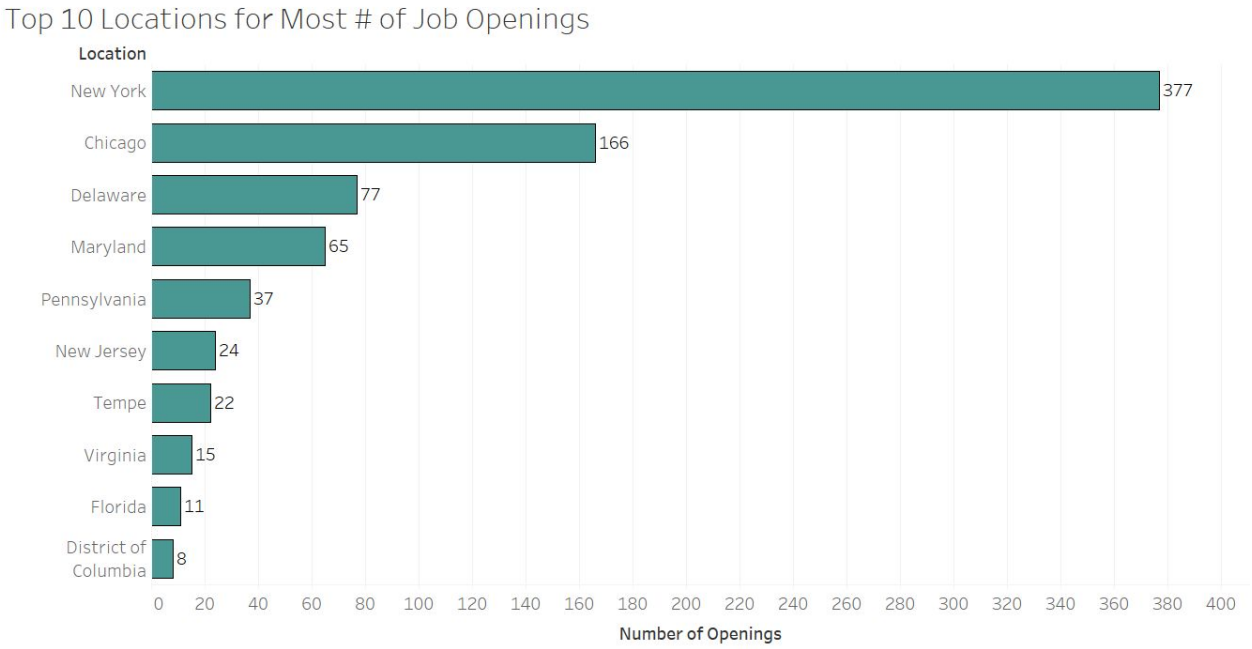
WordClouds

Institution

Northern Trust

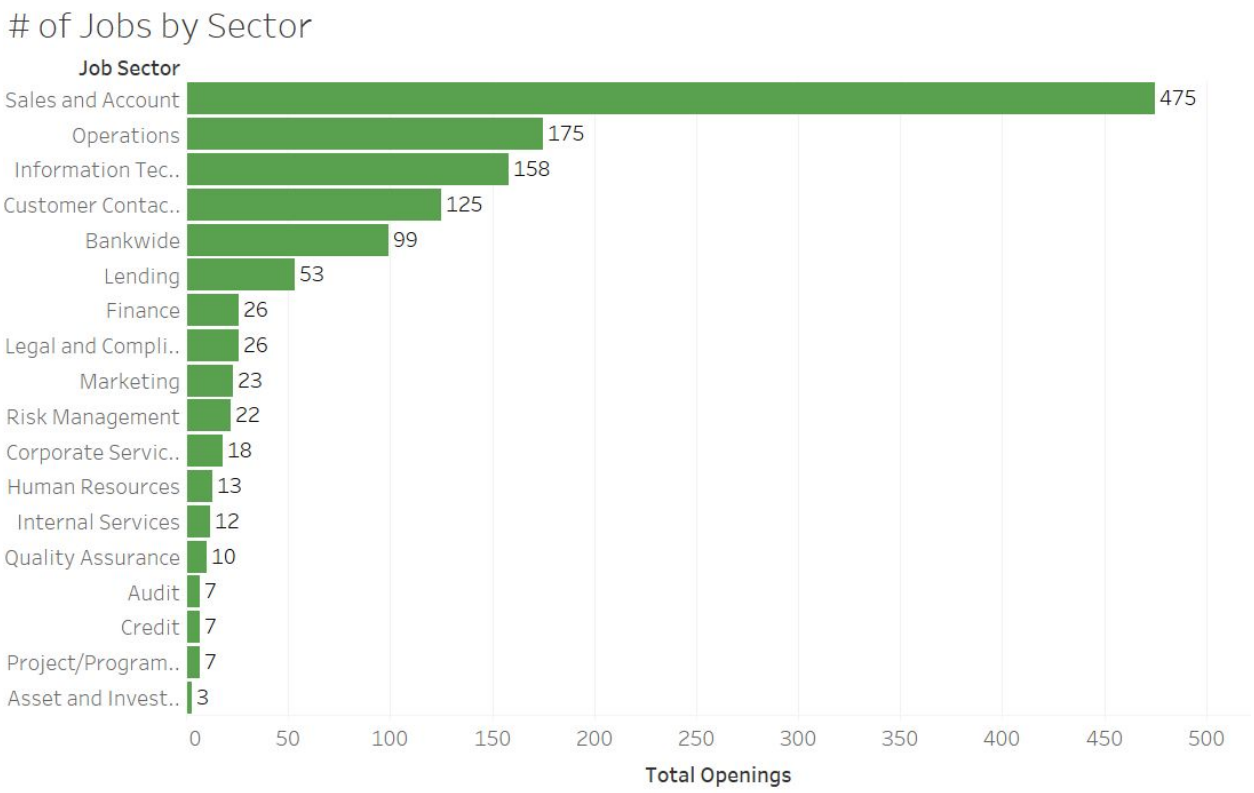
Top 10 location of job openings

Slice the scraped data observed from the websites into ID , institution , Position and location. So job openings are analysed on the basis of the sliced data. Top 10 positions with the job openings are displayed in the bar chart.



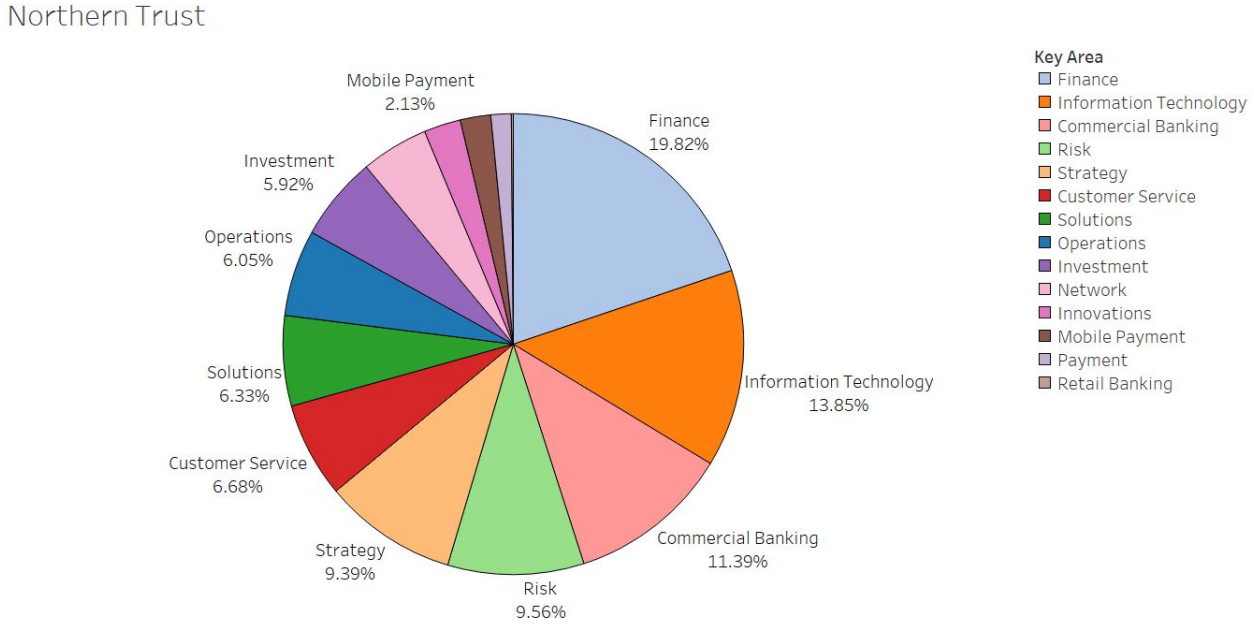
Classification of jobs by sector

Slice the scraped data observed from the websites into ID , institution , Position and location. So job openings are analysed on the basis of the sliced data.Then the classification of all the jobs is displayed in the bar graph.



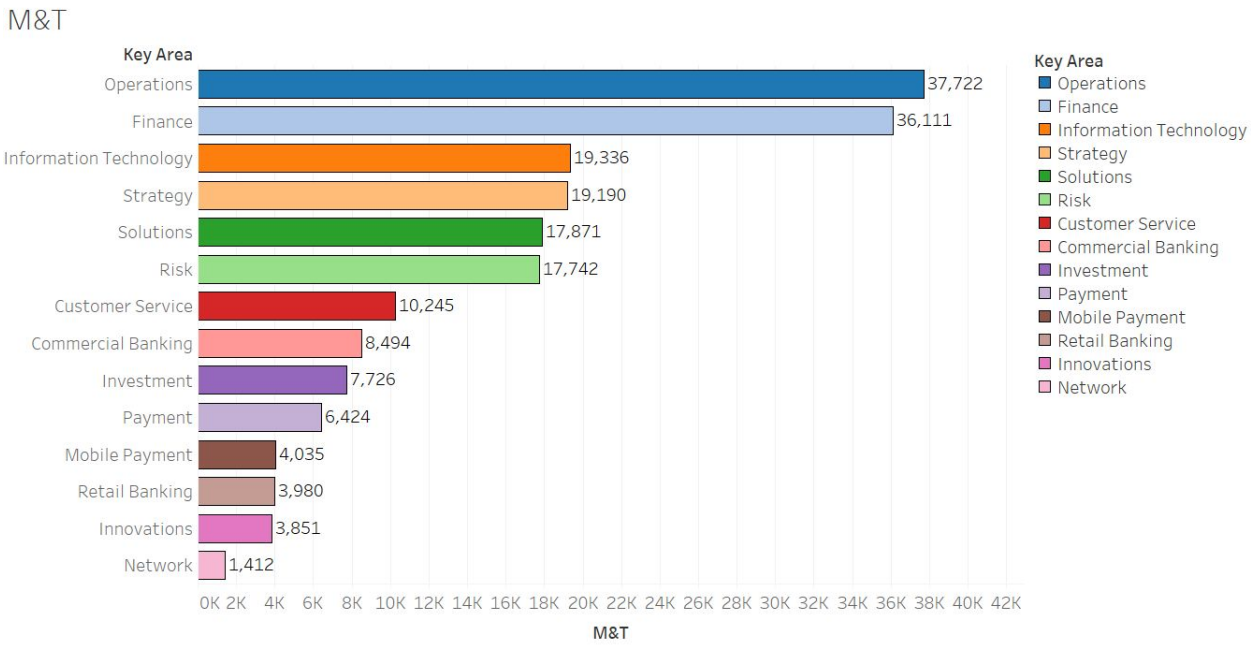
Northern Trust Key Areas

Contribution of a specific key area for Northern Trust Bank. This can be displayed by the following pie chart



M & T Bank Key Areas

Contribution of a specific key area for M & T Bank. This can be displayed by the following pie chart.



Interpretation of analysis

What we have used for our interpretation

1.Total Scores of each word based on their Number of Occurrence in each Link
2.These total scores have been segmented by Institutions
3.Also, we have taken into consideration, the most common words (164 in our case) from the pool of 300 words (Top 100 retrieved from each Algorithm)
4.We bucketed these 164 words based on the Most Relevant Categories (i.e Our Bucket Headers)
5.We scraped the websites of both banks and also retrieved a list of Job Openings, Job Categories and then we took this List and arranged them based on Job Categories (which is our another bucket of data)

FOR PART 3

KEY AREAS

The Key Areas drawn from our interpretation using Part 1 Are the Segmentation of Buckets based on the Keywords

KEY HIRING TRENDS

Based on the Keywords we retrieved in Part 1, when we ran them against the data retrieved by web scraping in Part 2, we could see that for both the banks, there are a certain set of Keywords that occurred the greatest number of times. It could be interpreted by the Word Clouds we have generated. But what's more interesting is that, when we visualized the data for the "Job Categories by the number of Openings" and compared it against "Key areas based on Word Frequency", we find out similar results which brings us to a conclusion that our analysis is accurate

OTHER INTERPRETATION

We have other visualizations where we are looking at the contribution of our key areas for both the Institutions individually. Also we can see that New York has the most number of Job openings for both institutions.