# Assignment 1 Report

## Topic:

Fintech Hiring trends in the largest banks in the US

## Authors:

- Dawna Grace Raj, Sathia
- Krutika kothawale
- Aayush Jain
- Keshav Vyas

## Objective:

Financial institutions in the US are changing rapidly. Changing business models and the technological revolution has fueled the growth of a new breed of financial products and services collectively known as 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. In a 2017 report by CB Insights more than 46% of the job openings were in the technology section. 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. Our goal in this case study is to conduct a study on the job openings in the top US Banks in the United States and analyze trends in the industry particularly in the area of Fintech.

## Dataset:

We have to extract Fintech keywords from these PDFs.

1. http://www3.weforum.org/docs/Beyond_Fintech__A_Pragmatic_Assessment_of_Disruptive_Potential_in_Financial_Services.pdf
2. http://www3.weforum.org/docs/WEF_The_future__of_financial_services.pdf
3. http://www3.weforum.org/docs/WEF_The_future_of_financial_infrastructure.pdf
4. http://www3.weforum.org/docs/WEF_A_Blueprint_for_Digital_Identity.pdf

We have to Scrape data from these links.

- https://usbank.taleo.net/careersection/10000/jobsearch.ftl?lang=en&keyword=campus
- https://jobs.americanexpress.com/jobs?page=1

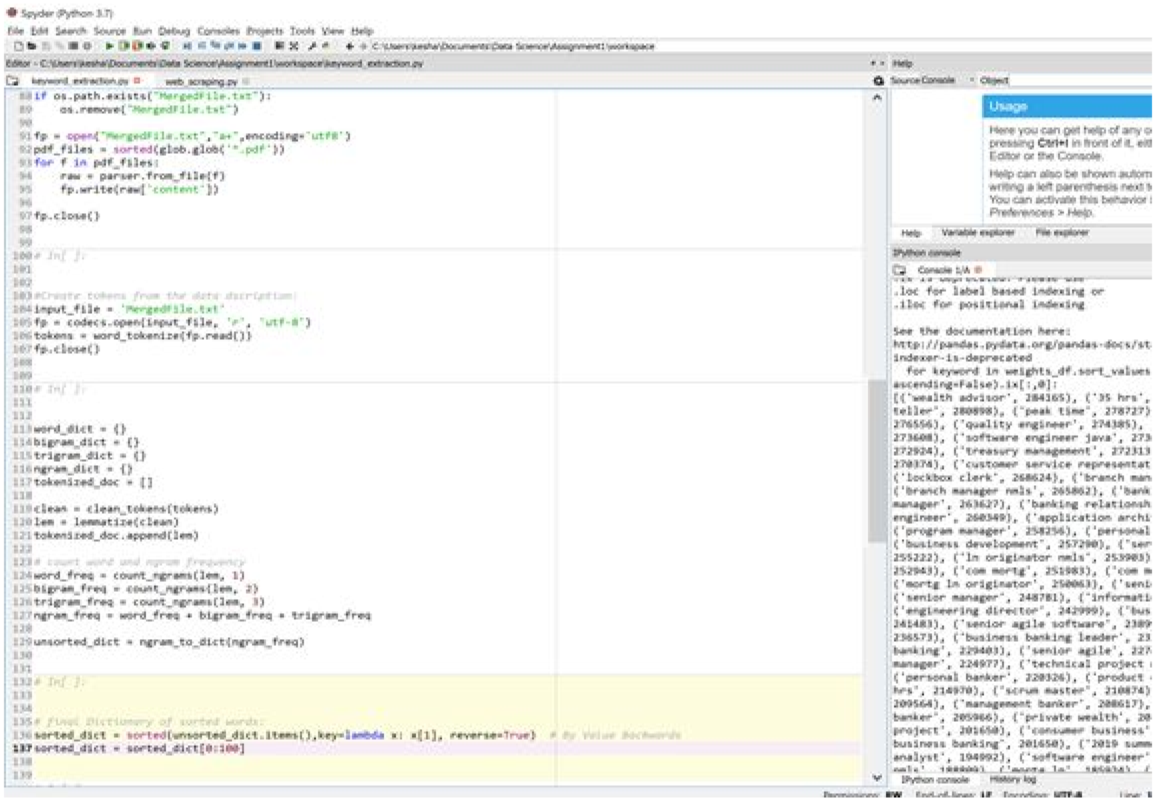
## Data Preparation:

### Data Merge:

- Parsed Fintech documents from World Economic Forum using tika library.
- Merged all documents into single text file.
- Read merged text and tokenize into words using nltk library

**Data Cleaning:**

- Clean Tokens: Ignoring case and punctuation. Remove stopwords provided by nltk library and custom stopwords to manually clear junk words or text.
- Lemmatize: Sort words by grouping inflected or variant forms of the same word.



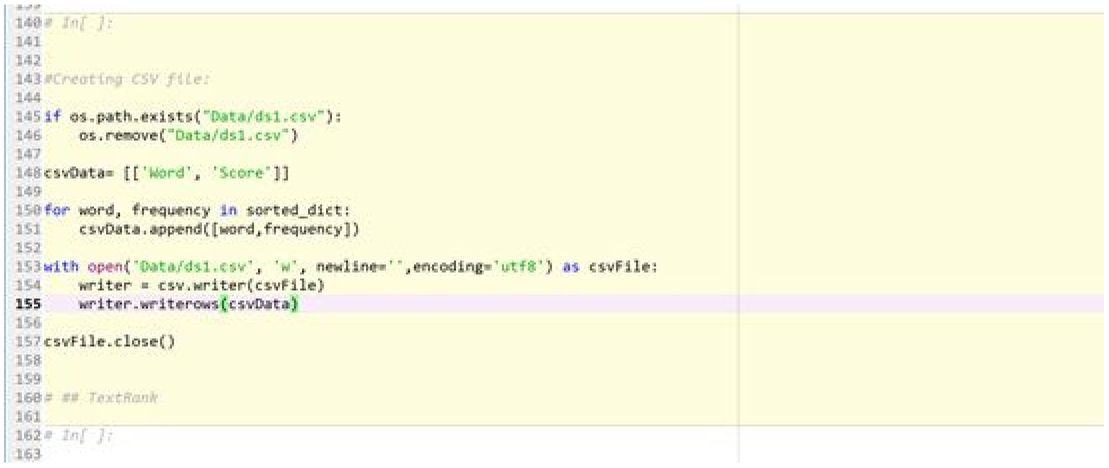
```python
if os.path.exists("MergedFile.txt"):
    os.remove("MergedFile.txt")

fp = open("MergedFile.txt","a+",encoding='utf8')
pdf_files = sorted(glob.glob('*.pdf'))
for f in pdf_files:
    raw = parser.from_file(f)
    fp.write(raw['content'])

fp.close()


# In[ ]:


#Create tokens from the data description
input_file = 'MergedFile.txt'
fp = codecs.open(input_file, 'r', 'utf-8')
tokens = word_tokenize(fp.read())
fp.close()


# In[ ]:


word_dict = {}
bigram_dict = {}
trigram_dict = {}
ngram_dict = {}
tokenized_doc = []

clean = clean_tokens(tokens)
lem = lemmatize(clean)
tokenized_doc.append(lem)

# count word and ngram frequency
word_freq = count_ngrams(lem, 1)
bigram_freq = count_ngrams(lem, 2)
trigram_freq = count_ngrams(lem, 3)
ngram_freq = word_freq + bigram_freq + trigram_freq

unsorted_dict = ngram_to_dict(ngram_freq)


# In[ ]:


# Final Dictionary of sorted words:
sorted_dict = sorted(unsorted_dict.items(),key=lambda x: x[1], reverse=True)  # By Value Backwards
sorted_dict = sorted_dict[0:100]
```

```
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/st
indexer-is-deprecated
  for keyword in weights_df.sort_values
ascending=False).ix[:,0]:
[('wealth advisor', 284165), ('35 hrs',
teller', 280898), ('peak time', 278727)
276556), ('quality engineer', 274385),
273608), ('software engineer java', 273
272924), ('treasury management', 272313
270374), ('customer service representat
('lockbox clerk', 268624), ('branch man
('branch manager nmls', 265862), ('bank
manager', 263627), ('banking relationsh
engineer', 260349), ('application archi
('program manager', 258256), ('personal
('business development', 257290), ('ser
255222), ('ln originator nmls', 253903)
252943), ('com mortg', 251983), ('com m
('mortg ln originator', 250063), ('seni
('senior manager', 248781), ('informati
('engineering director', 242999), ('bus
241483), ('senior agile software', 2389
236573), ('business banking leader', 23
banking', 229403), ('senior agile', 227
manager', 224977), ('technical project
('personal banker', 220326), ('product
hrs', 214970), ('scrum master', 210874)
209564), ('management banker', 208617),
banker', 205966), ('private wealth', 20
project', 201650), ('consumer business'
business banking', 201650), ('2019 summ
analyst', 194092), ('software engineer'
```

**Keyword extraction and ranking:**

**Word Frequency:**

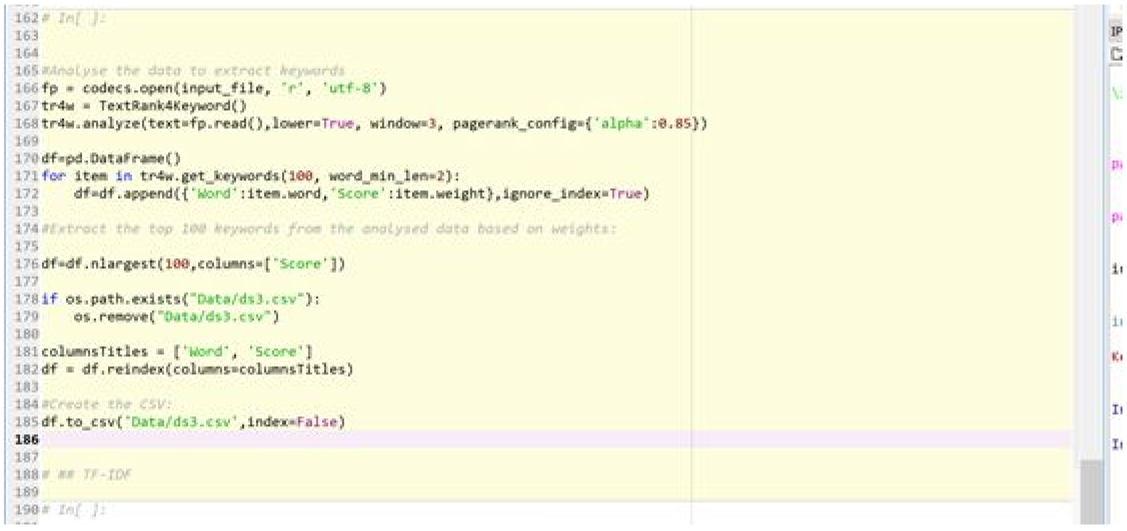
- Use clean tokens to create a list of words, bigrams, trigrams.
- Use nltk find most common 100 for each of the n-grams.
- Manually filter top 100 relevant using 300 n-grams.



```
# In[ ]:


#Creating CSV file:

if os.path.exists("Data/ds1.csv"):
    os.remove("Data/ds1.csv")

csvData= [['Word', 'Score']]

for word, frequency in sorted_dict:
    csvData.append([word,frequency])

with open('Data/ds1.csv', 'w', newline='',encoding='utf8') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(csvData)

csvFile.close()


# ## TextRank

# In[ ]:
```

**TF-IDF Score:**

- Using stop_words and ngram_range execute TfidfVectorizor from sklearn and get tfidf score.
- Convert to dataframe and sort by score.

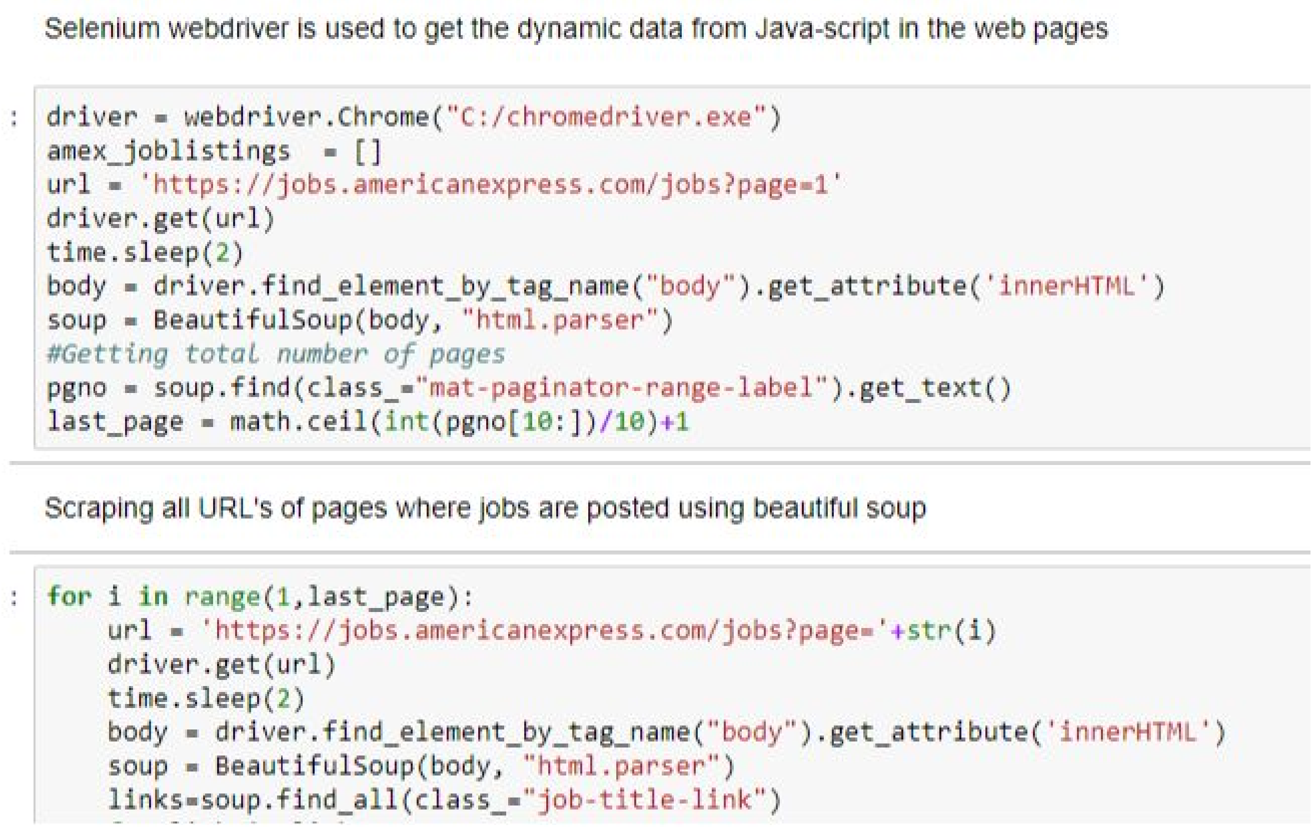


```
# In[ ]:


#Analyse the data to extract keywords
fp = codecs.open(input_file, 'r', 'utf-8')
tr4w = TextRank4Keyword()
tr4w.analyze(text=fp.read(),lower=True, window=3, pagerank_config={'alpha':0.85})

df=pd.DataFrame()
for item in tr4w.get_keywords(100, word_min_len=2):
    df=df.append({'Word':item.word,'Score':item.weight},ignore_index=True)

#Extract the top 100 keywords from the analysed data based on weights:

df=df.nlargest(100,columns=['Score'])

if os.path.exists("Data/ds3.csv"):
    os.remove("Data/ds3.csv")

columnsTitles = ['Word', 'Score']
df = df.reindex(columns=columnsTitles)

#Create the CSV:
df.to_csv('Data/ds3.csv',index=False)


# ## TF-IDF

# In[ ]:
```

**TextRank:**

- In order to find relevant keywords, the textrank algorithm constructs a word network. This network is constructed by looking which words follow one another.
- A link is set up between two words if they follow one another, the link gets a higher weight if these 2 words occur more frequently next to each other in the text.

## Web Scraping:


Selenium webdriver is used to get the dynamic data from Java-script in the web pages

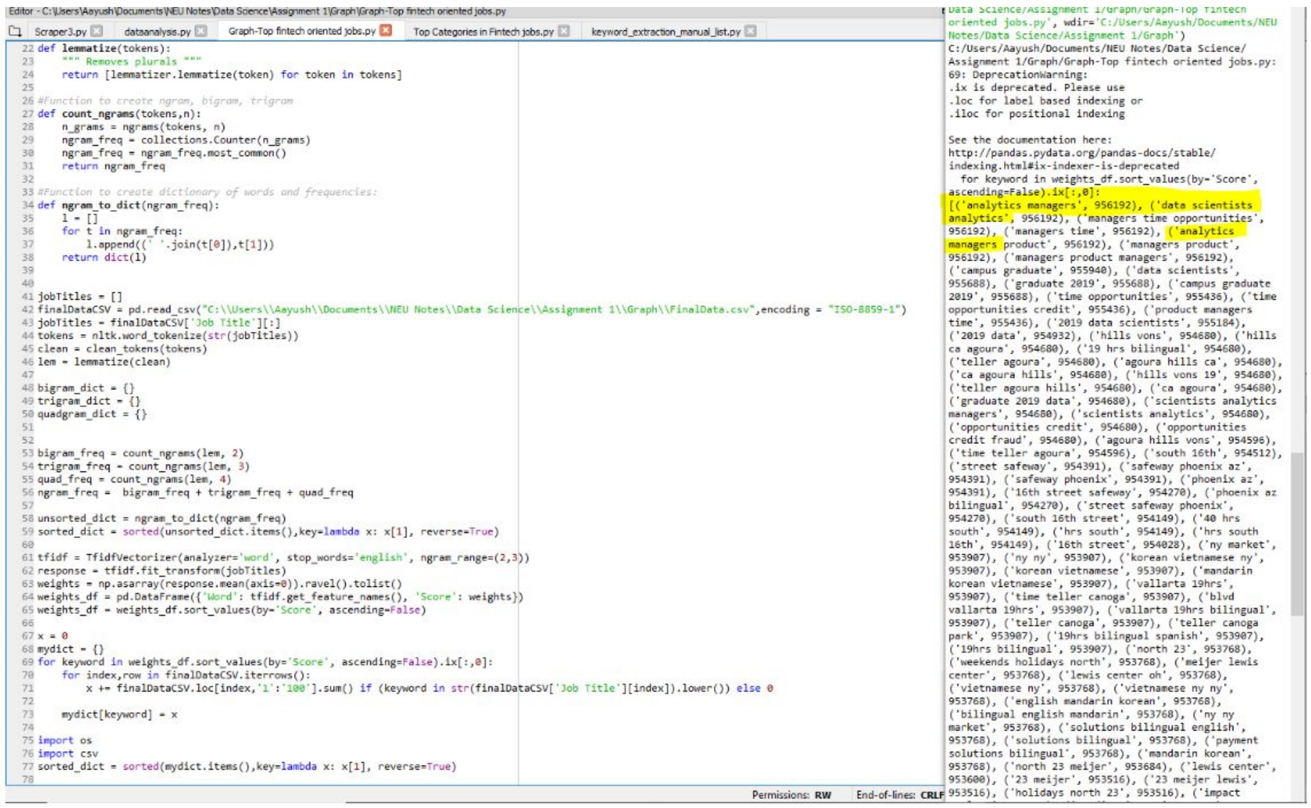
```
driver = webdriver.Chrome("C:/chromedriver.exe")
amex_joblistings  = []
url = 'https://jobs.americanexpress.com/jobs?page=1'
driver.get(url)
time.sleep(2)
body = driver.find_element_by_tag_name("body").get_attribute('innerHTML')
soup = BeautifulSoup(body, "html.parser")
#Getting total number of pages
pgno = soup.find(class_="mat-paginator-range-label").get_text()
last_page = math.ceil(int(pgno[10:])/10)+1
```

Scraping all URL's of pages where jobs are posted using beautiful soup

```
for i in range(1,last_page):
    url = 'https://jobs.americanexpress.com/jobs?page='+str(i)
    driver.get(url)
    time.sleep(2)
    body = driver.find_element_by_tag_name("body").get_attribute('innerHTML')
    soup = BeautifulSoup(body, "html.parser")
    links=soup.find_all(class_="job-title-link")
```

- Web Scraping done for US Bank and American Express
- Data extracted from webpage

## Top Job Titles in Fintech Industry:


Editor - C:\Users\Aayush\Documents\NEU Notes\Data Science\Assignment 1\Graph\Graph-Top fintech oriented jobs.py
Scraper3.py
dataanalysis.py
Graph-Top fintech oriented jobs.py
Top Categories in Fintech jobs.py
keyword_extraction_manual_list.py
Permissions: RW
End-of-lines: CRLF

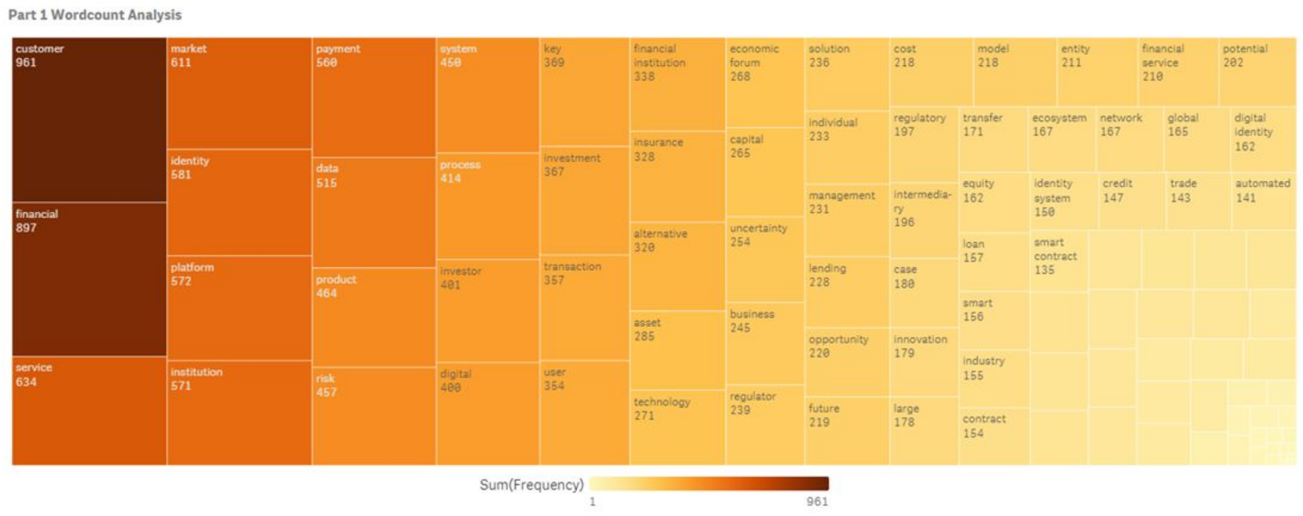
```python
def lemmatize(tokens):
    """ Removes plurals """
    return [lemmatizer.lemmatize(token) for token in tokens]

#Function to create ngram, bigram, trigram
def count_ngrams(tokens,n):
    n_grams = ngrams(tokens, n)
    ngram_freq = collections.Counter(n_grams)
    ngram_freq = ngram_freq.most_common()
    return ngram_freq

#Function to create dictionary of words and frequencies:
def ngram_to_dict(ngram_freq):
    l = []
    for t in ngram_freq:
        l.append((' '.join(t[0]),t[1]))
    return dict(l)


jobTitles = []
finalDataCSV = pd.read_csv("C:\\Users\\Aayush\\Documents\\NEU Notes\\Data Science\\Assignment 1\\Graph\\FinalData.csv",encoding = "ISO-8859-1")
jobTitles = finalDataCSV['Job Title'][:]
tokens = nltk.word_tokenize(str(jobTitles))
clean = clean_tokens(tokens)
lem = lemmatize(clean)

bigram_dict = {}
trigram_dict = {}
quadgram_dict = {}


bigram_freq = count_ngrams(lem, 2)
trigram_freq = count_ngrams(lem, 3)
quad_freq = count_ngrams(lem, 4)
ngram_freq =  bigram_freq + trigram_freq + quad_freq

unsorted_dict = ngram_to_dict(ngram_freq)
sorted_dict = sorted(unsorted_dict.items(),key=lambda x: x[1], reverse=True)

tfidf = TfidfVectorizer(analyzer='word', stop_words='english', ngram_range=(2,3))
response = tfidf.fit_transform(jobTitles)
weights = np.asarray(response.mean(axis=0)).ravel().tolist()
weights_df = pd.DataFrame({'Word': tfidf.get_feature_names(), 'Score': weights})
weights_df = weights_df.sort_values(by='Score', ascending=False)

x = 0
mydict = {}
for keyword in weights_df.sort_values(by='Score', ascending=False).ix[:,0]:
    for index,row in finalDataCSV.iterrows():
        x += finalDataCSV.loc[index,'1':'100'].sum() if (keyword in str(finalDataCSV['Job Title'][index]).lower()) else 0

    mydict[keyword] = x

import os
import csv
sorted_dict = sorted(mydict.items(),key=lambda x: x[1], reverse=True)

```

```
Data Science/Assignment 1/Graph/Graph-Top fintech
oriented jobs.py', wdir='C:/Users/Aayush/Documents/NEU
Notes/Data Science/Assignment 1/Graph')
C:/Users/Aayush/Documents/NEU Notes/Data Science/
Assignment 1/Graph/Graph-Top fintech oriented jobs.py:
69: DeprecationWarning:
.ix is deprecated. Please use
.loc for label based indexing or
.iloc for positional indexing

See the documentation here:
http://pandas.pydata.org/pandas-docs/stable/
indexing.html#ix-indexer-is-deprecated
  for keyword in weights_df.sort_values(by='Score',
ascending=False).ix[:,0]:
[('analytics managers', 956192), ('data scientists
analytics', 956192), ('managers time opportunities',
956192), ('managers time', 956192), ('analytics
managers product', 956192), ('managers product',
956192), ('managers product managers', 956192),
('campus graduate', 955940), ('data scientists',
955688), ('graduate 2019', 955688), ('campus graduate
2019', 955688), ('time opportunities', 955436), ('time
opportunities credit', 955436), ('product managers
time', 955436), ('2019 data scientists', 955184),
('2019 data', 954932), ('hills vons', 954680), ('hills
ca agoura', 954680), ('19 hrs bilingual', 954680),
('teller agoura', 954680), ('agoura hills ca', 954680),
('ca agoura hills', 954680), ('hills vons 19', 954680),
('teller agoura hills', 954680), ('ca agoura', 954680),
('graduate 2019 data', 954680), ('scientists analytics
managers', 954680), ('scientists analytics', 954680),
('opportunities credit', 954680), ('opportunities
credit fraud', 954680), ('agoura hills vons', 954596),
('time teller agoura', 954596), ('south 16th', 954512),
('street safeway', 954391), ('safeway phoenix az',
954391), ('safeway phoenix', 954391), ('phoenix az',
954391), ('16th street safeway', 954270), ('phoenix az
bilingual', 954270), ('street safeway phoenix',
954270), ('south 16th street', 954149), ('40 hrs
south', 954149), ('hrs south', 954149), ('hrs south
16th', 954149), ('16th street', 954028), ('ny market',
953907), ('ny ny', 953907), ('korean vietnamese ny',
953907), ('korean vietnamese', 953907), ('mandarin
korean vietnamese', 953907), ('vallarta 19hrs',
953907), ('time teller canoga', 953907), ('blvd
vallarta 19hrs', 953907), ('vallarta 19hrs bilingual',
953907), ('teller canoga', 953907), ('teller canoga
park', 953907), ('19hrs bilingual spanish', 953907),
('19hrs bilingual', 953907), ('north 23', 953768),
('weekends holidays north', 953768), ('meijer lewis
center', 953768), ('lewis center oh', 953768),
('vietnamese ny', 953768), ('vietnamese ny ny',
953768), ('english mandarin korean', 953768),
('bilingual english mandarin', 953768), ('ny ny
market', 953768), ('solutions bilingual english',
953768), ('solutions bilingual', 953768), ('payment
solutions bilingual', 953768), ('mandarin korean',
953768), ('north 23 meijer', 953684), ('lewis center',
953600), ('23 meijer', 953516), ('23 meijer lewis',
953516), ('holidays north 23', 953516), ('impact
```

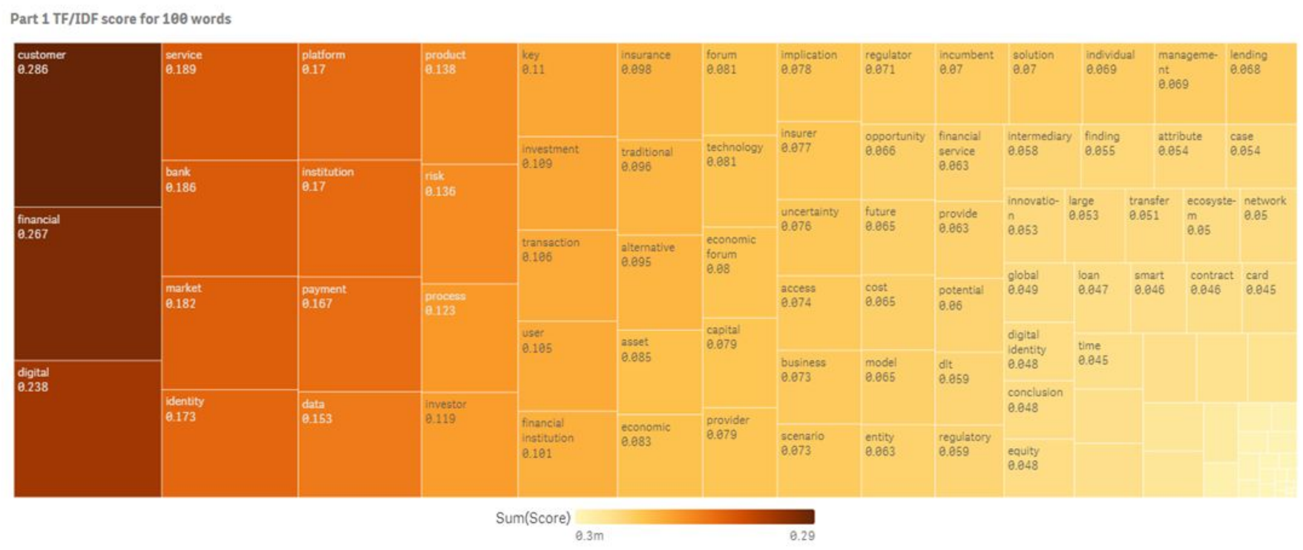
- Compiled a list of every bigrams, trigrams in job titles in the final data
- Combined similar job titles and compared them with each job listing and deducted the final fintech score for each job title.
- Hence extracted top fintech oriented job titles.

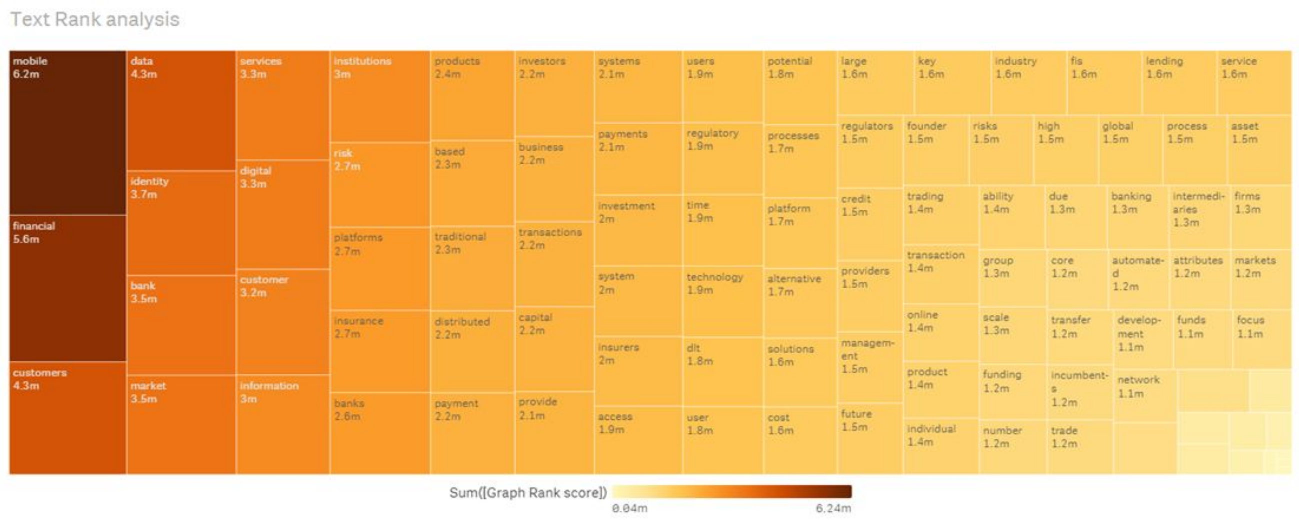
## Insights:

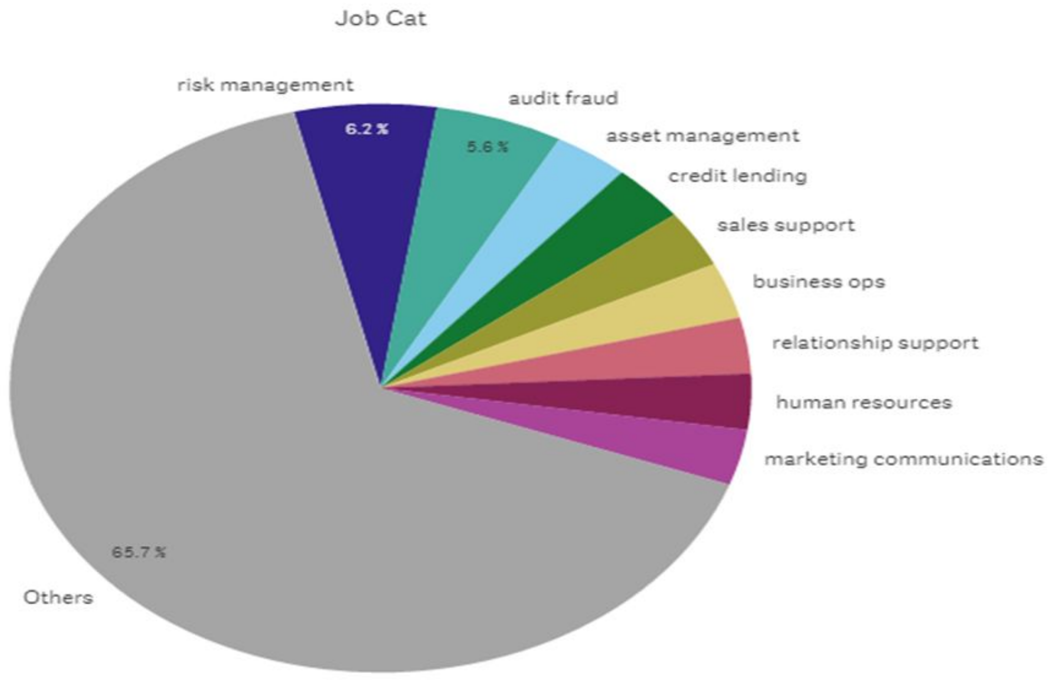
Part 1 A Wordcount Analysis for 100 words

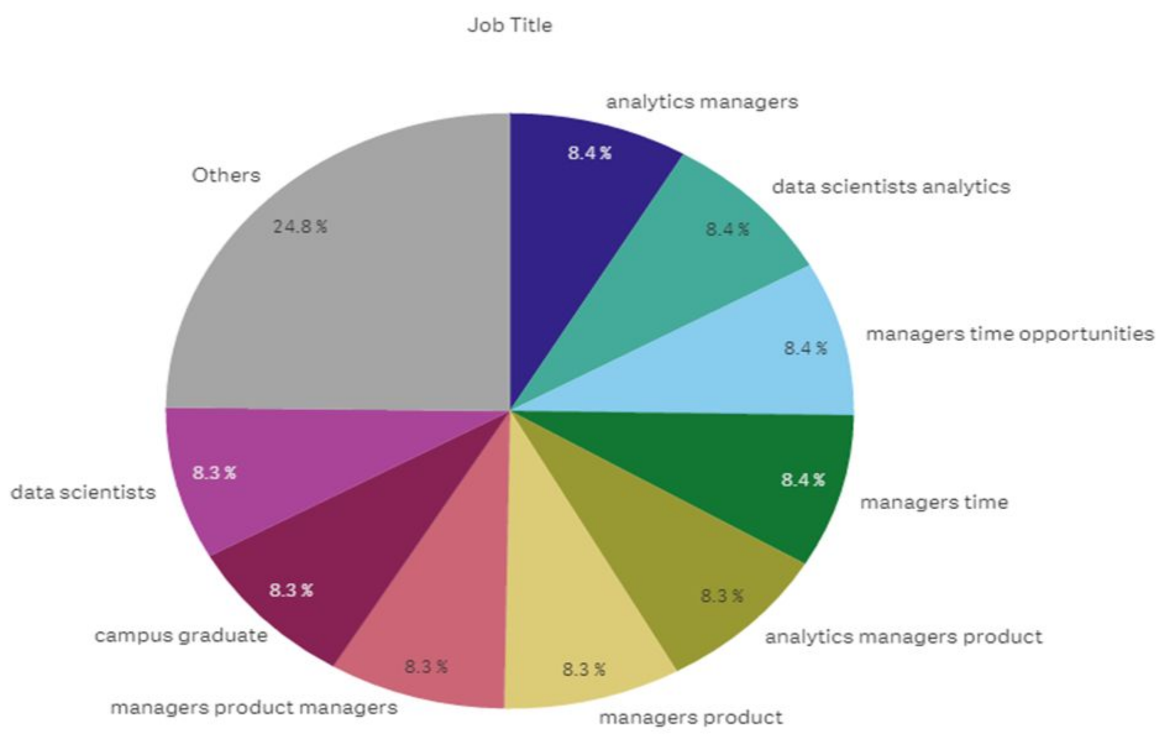


Part 1B TF/IDF score for 100 words

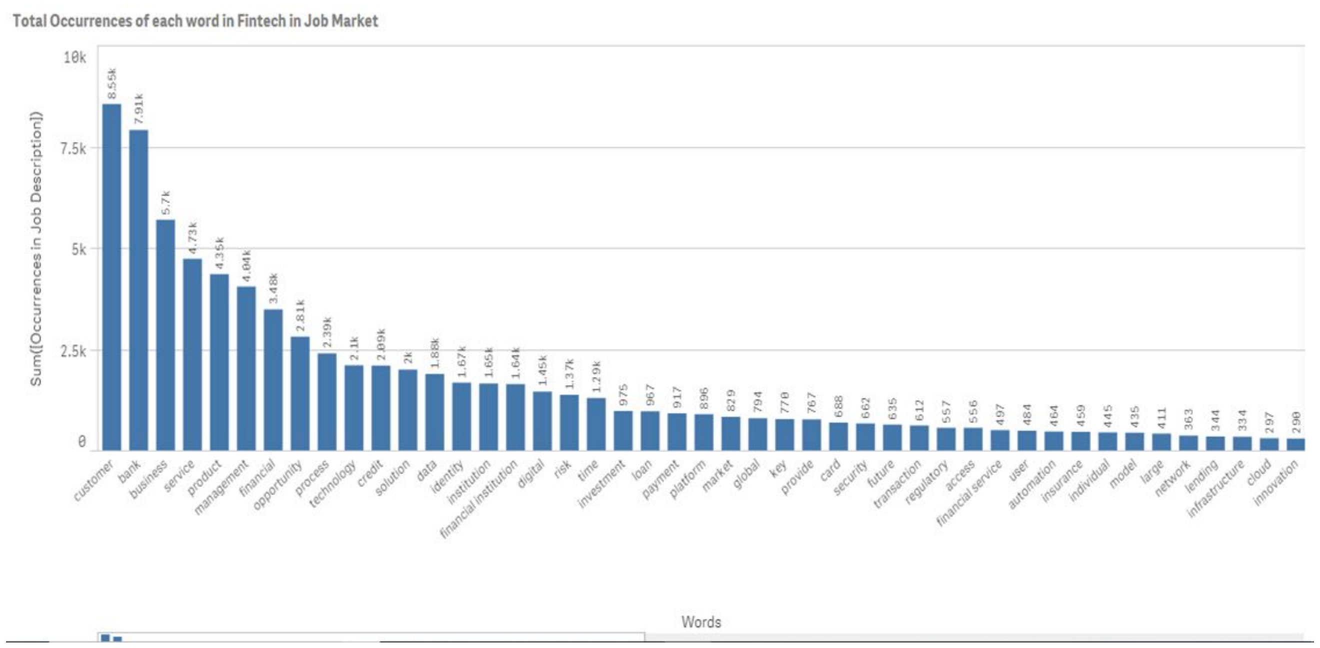



Text Rank analysis
mobile 6.2m
financial 5.6m
customers 4.3m
data 4.3m
identity 3.7m
bank 3.5m
market 3.5m
services 3.3m
digital 3.3m
customer 3.2m
information 3m
institutions 3m
risk 2.7m
platforms 2.7m
insurance 2.7m
banks 2.6m
products 2.4m
based 2.3m
traditional 2.3m
distributed 2.2m
payment 2.2m
investors 2.2m
business 2.2m
transactions 2.2m
capital 2.2m
provide 2.1m
systems 2.1m
payments 2.1m
investment 2m
system 2m
insurers 2m
access 1.9m
users 1.9m
regulatory 1.9m
time 1.9m
technology 1.9m
dlt 1.8m
user 1.8m
potential 1.8m
processes 1.7m
platform 1.7m
alternative 1.7m
solutions 1.6m
cost 1.6m
large 1.6m
key 1.6m
industry 1.6m
fis 1.6m
lending 1.6m
service 1.6m
regulators 1.5m
founder 1.5m
risks 1.5m
high 1.5m
global 1.5m
process 1.5m
asset 1.5m
credit 1.5m
providers 1.5m
managem-ent 1.5m
future 1.5m
trading 1.4m
transaction 1.4m
online 1.4m
product 1.4m
individual 1.4m
ability 1.4m
group 1.3m
scale 1.3m
funding 1.2m
number 1.2m
due 1.3m
core 1.2m
transfer 1.2m
incumbent-s 1.2m
trade 1.2m
banking 1.3m
automate-d 1.2m
develop-ment 1.1m
network 1.1m
intermedi-aries 1.3m
attributes 1.2m
funds 1.1m
firms 1.3m
markets 1.2m
focus 1.1m
Sum([Graph Rank score])
0.04m
6.24m


Job Cat
risk management
6.2 %
audit fraud
5.6 %
asset management
credit lending
sales support
business ops
relationship support
human resources
marketing communications
65.7 %
Others



Total Occurrences of each word in Fintech in Job Market


Total Occurrences of each word in Fintech in Job Market
Sum([Occurrences in Job Description])
10k
7.5k
5k
2.5k
customer 8.55k
bank 7.91k
business 5.7k
service 4.73k
product 4.35k
management 4.04k
financial 3.48k
opportunity 2.81k
process 2.39k
technology 2.1k
credit 2.09k
solution 2k
data 1.88k
identity 1.67k
institution 1.65k
financial institution 1.64k
digital 1.45k
risk 1.37k
time 1.29k
investment 975
loan 967
payment 917
platform 896
market 829
global 794
key 770
provide 767
card 688
security 662
future 635
transaction 612
regulatory 557
access 556
financial service 497
user 484
automation 464
insurance 459
individual 445
model 435
large 411
network 363
lending 344
infrastructure 334
cloud 297
innovation 290
Words