In [2]:
```python
# Visualization
import matplotlib.pyplot as plt

#  Dataframe
import pandas as pd

#  NLP processing
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from rake_nltk import Rake

# Statistical Analysis
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import CountVectorizer

# File saving
import pickle

# Used for text extraction
from io import StringIO

# Text extraction from a PDF
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfdocument import PDFDocument
from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from pdfminer.pdfparser import PDFParser

# Open files
import urllib

# Preparing file for processing
from io import BytesIO
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sambe\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\sambe\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [3]:
```python
# Extracts and parses text data from submitted document

def document_to_text(url):
    rsrcmgr = PDFResourceManager()
    retstr = StringIO()
    codec = 'utf-8'
    laparams = LAParams()
    device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams)
    f = urllib.request.urlopen(url).read()
    fp = BytesIO(f)
    interpreter = PDFPageInterpreter(rsrcmgr, device)
    password = ""
    maxpages = 0
    caching = True
    pagenos = set()
    for page in PDFPage.get_pages(fp,
                                  pagenos,
                                  maxpages=maxpages,
                                  password=password,
                                  caching=caching,
                                  check_extractable=True):
        interpreter.process_page(page)
    fp.close()
    device.close()
    parsed = retstr.getvalue()
    retstr.close()

# Test if no data can be parsed from the document
    if parsed == None:
        print("The submitted document cannot be read.")

# Initial cleaning of text
    try:
        parsed = parsed.replace('\n', '')
        parsed = parsed.replace('\u200b', '')
    except:
        pass
    return parsed
```

In [4]:
```python
# Example of what this function produces

document = 'https://rookieplay-uploads.s3.amazonaws.com/media/Bell-Resume_Apr2021_1NJ3wOt.pdf'
document_text = document_to_text(document)
document_to_text(document)
```

Out[4]:
```
'SAMUEL LOGAN BELLmedium.com/@sambellsoup | github.com/sambellsoup | linkedin.com/in/sam--bell/sambellsoup@gmai
l.com | 757-652-7433 | Brooklyn, NY 11216Data Scientist and machine learning engineer with a penchant for creat
ing data pipelines while prioritizing cleanlinessand data validation. Constantly seeking new skills to add to m
y toolbox with a hunger to work and eagerness to learn.TECHNICAL SKILLSAWS (S3, EC2), Bokeh, CSS, Django, Githu
b, Heroku, JavaScript, Linux, NumPy, Python, Pandas,scikit-learn, SQL, TableauRELEVANT EXPERIENCERookiePlayChie
f Technical OfficerNew York, NYMay 2020-Present● Scripting ETL pipeline incorporating NLP techniques for topic
modeling and data analysis● Developing applications for users to upload documents stored in an S3 database● Eng
ineering job recommendation system backed by PostgreSQL● Utilizing APIs to funnel in data for live job ads and
company information● Building and developing the backend of the application with Django● Creating and tailoring
data models with a focus on purpose● Implementing Luigi triggered by crontab for automated tasks● Ensuring data
consistency with PandasTECHNICAL PROJECTSVideo Game Recommendation System - https://github.com/sambellsoup/VGR2
Asystem that recommends users new games based on their previous ratings● Used Singular Value Decomposition(SVD)
algorithm and clustering to predict user ratings and generaterecommendation● Deployed as a web application writ
ten in Flask backed by a Postgres databaseGlobal Voter Turnout - https://github.com/sambellsoup/Voter_TurnoutEv
aluating the most influential predictor of voter turnout by combining UN and IDEA datasets● K fold cross valida
tion and modeling with Gradient Boost and Random ForestTree Coverage - https://github.com/sambellsoup/TreesPred
icting tree species growth in Colorado with Kaggle dataset.● Utilize XGBoost, Decision Trees, and Random Forest
to create the best model for prediction● Manipulate features with Principal Component Analysis and then Lasso f
or feature selectionOTHER EXPERIENCEBoy Scouts of AmericaExploring ExecutivePeace CorpsEnglish Volunteer● Manag
ed website updates for new programs and events● Taught English and computer skills to grades 6 through 9EDUCATI
ONFlatiron SchoolNew York University, Speech-Language PathologyGeorge Mason University, B.S. PsychologyNew Yor
k, NYMay 2014 - Aug 2017AlbaniaMay 2014 - Aug 2017Mar 2019 - Aug 2019, New York, NYSep 2017-Dec 2018, New York,
NYAug 2007 - May 2010, Fairfax, VA\x0c'
```

In [5]:
```python
# Creates a dataframe of the text and label it resume

def compile_document_text(text):
    # job_descriptions = pd.read_csv('data/job_descriptions.csv', index_col=0)
    with open('pkl/job_descriptions_dec2020.pkl', 'rb') as f:
        job_descriptions = pickle.load(f)
    data = [['resume', text]]
    basic_documentdf = pd.DataFrame(data, columns = ['title', 'description'])
    return basic_documentdf
```

In [6]:
```python
# Example

doc_df = compile_document_text(document_text)
doc_df
```

Out[6]:

	title	description
0	resume	SAMUEL LOGAN BELLmedium.com/@sambellsoup | git...

In [7]:
```python
# RAKE algorithm to determine key phrases in a body of text
# by analyzing the frequency of word appearance
# and its co-occurance with other words in the text.
# RAKE outputs a list, the words are joined into a single string

def text_to_bagofwords(df):
    df['rake_key_words'] = ''
    r = Rake()
    for index, row in df.iterrows():
        r.extract_keywords_from_text(row['description'])
        key_words_dict_scores = r.get_word_degrees()
        row['rake_key_words'] = list(key_words_dict_scores.keys())
# Transform key words into bag of words
    df['bag_of_words'] = ''
    for index, row in df.iterrows():
        words = ''
        words += ' '.join(row['rake_key_words']) + ' '
        row['bag_of_words'] = words
    verbose_documentdf = df
    return verbose_documentdf
```

In [8]:
```python
# Example

bowdf = text_to_bagofwords(doc_df)
text_to_bagofwords(doc_df)
```

Out[8]:

	title	description	rake_key_words	bag_of_words
0	resume	SAMUEL LOGAN BELLmedium.com/@sambellsoup | git...	[influential, predictor, tailoring, data, mode...	influential predictor tailoring data models de...

In [9]:
```python
# Combine resume keywords with the pre-processed job description data

def join_and_condense(df):
    with open('pkl/job_descriptions_dec2020.pkl', 'rb') as f:
        job_descriptions = pickle.load(f)
    job_descriptions = job_descriptions.append(df)
    recommend_df = job_descriptions[['title', 'bag_of_words']]
    recommend_df = recommend_df.reset_index(drop=True)
    return recommend_df
```

In [10]:
```python
# Example

fulldf = join_and_condense(bowdf)
fulldf
```

Out[10]:

	title	bag_of_words
0	training_manager	provide limited express interest additional tr...
1	training_manager	workshops including schedules integrity respon...
2	training_manager	oversee translate diverse audience advanced kn...
3	training_manager	public speaking • enjoy getting previous exper...
4	training_manager	luxury business pr teams powerpoint paced envi...
...	...	...
561	Data Entry Specialist	enter heavy volume northeastern us entry prepa...
562	Pathway Advisor	students chosen guided academic pathway debt c...
563	Director Of Food And Beverage	3 years jumped around forbes five star service...
564	Director of Operations	improvements post meeting follow 1 administrat...
565	resume	influential predictor tailoring data models de...

566 rows × 2 columns

In [11]:
```python
# Returns a matrix of values that represents how closely each document
# matches to each other on a scale of 0 to 1

def vectorize_text(df):
    count = CountVectorizer()
    count_matrix = count.fit_transform(df['bag_of_words'])
    cosine_sim = cosine_similarity(count_matrix, count_matrix)
    return cosine_sim
```

In [12]:
```python
# Example

sim = vectorize_text(fulldf)
sim[-1]
```

Out[12]:
```
array([0.05156845, 0.05346523, 0.08437563, 0.08824484, 0.04750557,
       0.07299465, 0.05192616, 0.07882512, 0.0635963 , 0.06306009,
       0.0514302 , 0.06320751, 0.0649313 , 0.07768672, 0.07199194,
       0.06026831, 0.05647265, 0.0652884 , 0.08248703, 0.06078019,
       0.0234597 , 0.02349691, 0.01638244, 0.04421523, 0.05580924,
       0.0409917 , 0.03338903, 0.01337621, 0.04649626, 0.0426617 ,
       0.07281639, 0.06716127, 0.05648951, 0.05148525, 0.05519386,
       0.06255432, 0.04819291, 0.03283793, 0.05003065, 0.05488751,
       0.07790773, 0.06251526, 0.07566725, 0.05152039, 0.03272779,
       0.05812284, 0.04409672, 0.05605973, 0.07151985, 0.06590443,
       0.06019293, 0.06050702, 0.02197935, 0.01390096, 0.06047355,
       0.04055327, 0.07372098, 0.07207313, 0.07055637, 0.08819343,
       0.03883541, 0.03584437, 0.05434783, 0.05434783, 0.05529505,
       0.03683747, 0.07729469, 0.02027664, 0.        , 0.00897303,
       0.06668313, 0.05853937, 0.07795204, 0.10078925, 0.10384091,
       0.08891084, 0.09124486, 0.06216699, 0.07747984, 0.09710314,
       0.07061695, 0.06765101, 0.0555627 , 0.05734345, 0.05332   ,
       0.06656505, 0.05008752, 0.05571767, 0.0234597 , 0.02978777,
       0.07926529, 0.05277683, 0.07476672, 0.07602001, 0.0695048 ,
       0.04721922, 0.0441356 , 0.0437165 , 0.03971703, 0.00954722,
       0.03439596, 0.04914732, 0.05074516, 0.05339507, 0.01209923,
       0.04830553, 0.00992926, 0.01390096, 0.01751356, 0.04914732,
       0.05943622, 0.05507601, 0.02151377, 0.07350594, 0.0699912 ,
       0.05465397, 0.04681672, 0.06142751, 0.04478905, 0.09403659,
       0.05074516, 0.04256283, 0.01390096, 0.0695048 , 0.04480685,
       0.03575993, 0.01174846, 0.0477361 , 0.08025724, 0.02423931,
       0.06320751, 0.01418761, 0.05897678, 0.04769723, 0.03960404,
       0.02948839, 0.04170288, 0.04239753, 0.02119877, 0.05024484,
       0.04361119, 0.01516719, 0.02144965, 0.02864166, 0.00954722,
       0.02095649, 0.04783649, 0.        , 0.02358128, 0.01891681,
       0.02496686, 0.03529686, 0.03033438, 0.02730407, 0.06031084,
       0.05142939, 0.05478652, 0.0428993 , 0.08683861, 0.01449275,
       0.04126724, 0.03858463, 0.02958321, 0.01174846, 0.02419847,
       0.04732463, 0.02581344, 0.0426617 , 0.04796287, 0.01268978,
       0.02541347, 0.01313517, 0.02782419, 0.01363101, 0.07165716,
       0.068058  , 0.07038279, 0.01516719, 0.06444295, 0.05734345,
       0.06627023, 0.01418761, 0.09081144, 0.03371478, 0.0159455 ,
       0.04526272, 0.01685739, 0.02898551, 0.0539507 , 0.06478228,
       0.01626984, 0.01248343, 0.03033438, 0.0347524 , 0.0521286 ,
       0.02072233, 0.0082487 , 0.02963695, 0.04510067, 0.05797101,
       0.05194854, 0.05108384, 0.01337621, 0.02496686, 0.02675241,
       0.02496686, 0.01363101, 0.03015542, 0.04029478, 0.01751356,
       0.02780192, 0.02780192, 0.01751356, 0.05747886, 0.06865887,
       0.07710866, 0.06815507, 0.04597311, 0.07930257, 0.05277198,
       0.04420497, 0.05254069, 0.08670763, 0.04770855, 0.05172606,
       0.03603656, 0.05882989, 0.0639242 , 0.05110095, 0.0773302 ,
       0.06580657, 0.03524537, 0.0695048 , 0.04633654, 0.05698093,
       0.07439816, 0.06538462, 0.03240678, 0.03236771, 0.05988185,
       0.04821465, 0.02072233, 0.04404869, 0.02496686, 0.04751251,
       0.03858463, 0.02782419, 0.06704287, 0.04355229, 0.05066539,
       0.05982021, 0.03015542, 0.        , 0.01142652, 0.01965893,
       0.04161144, 0.04161144, 0.02366657, 0.        , 0.01481847,
       0.05419208, 0.02197935, 0.01337621, 0.04593007, 0.        ,
       0.02837522, 0.0362977 , 0.03870654, 0.08307413, 0.04351689,
       0.05020437, 0.05858749, 0.04267182, 0.04524377, 0.02837522,
       0.01751356, 0.04378391, 0.03791798, 0.01825286, 0.02197935,
       0.04012862, 0.02547409, 0.05254069, 0.01441463, 0.03015542,
       0.02978777, 0.03584437, 0.04256283, 0.01909444, 0.02222771,
       0.05884616, 0.04170288, 0.01554175, 0.01418761, 0.04301324,
       0.04012862, 0.05339507, 0.05186466, 0.05284352, 0.04755917,
       0.05277198, 0.03619502, 0.02761841, 0.03502713, 0.03791171,
       0.05815189, 0.05429253, 0.04603068, 0.0439587 , 0.06371495,
       0.02864166, 0.05020437, 0.06865887, 0.03147948, 0.04486516,
       0.01765424, 0.05529505, 0.027528  , 0.0293058 , 0.0173762 ,
       0.04574425, 0.0485813 , 0.03298552, 0.03159309, 0.08758466,
       0.02867549, 0.06532016, 0.0374881 , 0.04880953, 0.06193801,
       0.05797101, 0.05057217, 0.0499173 , 0.08594742, 0.03575993,
       0.05378443, 0.02006431, 0.04445542, 0.04597311, 0.06220381,
       0.0409561 , 0.05427413, 0.04811871, 0.06241716, 0.0725439 ,
       0.04234286, 0.03108349, 0.07034224, 0.03818888, 0.06815507,
       0.065209  , 0.03958262, 0.00792081, 0.03855433, 0.03051657,
       0.02275079, 0.06384424, 0.03976214, 0.05410942, 0.02095649,
       0.06415985, 0.04603068, 0.04460046, 0.04783649, 0.05600857,
       0.01223911, 0.05982021, 0.03905833, 0.05140577, 0.03346958,
       0.04725632, 0.04445542, 0.04496938, 0.01313517, 0.04089304,
       0.0374881 , 0.04523313, 0.05550799, 0.04783649, 0.03814572,
       0.06286946, 0.05216188, 0.07232197, 0.05427413, 0.08462638,
       0.05324978, 0.02675241, 0.03589213, 0.01605145, 0.01895586,
       0.        , 0.05995917, 0.04964629, 0.02197935, 0.04633654,
       0.02737928, 0.07728058, 0.09020135, 0.0271319 , 0.03147948,
       0.00954722, 0.05675044, 0.04752485, 0.02914433, 0.04089304,
       0.05055731, 0.01337621, 0.04245682, 0.04347826, 0.03210289,
       0.01363101, 0.06178983, 0.03349656, 0.01903467, 0.06075813,
       0.01174846, 0.02407717, 0.03041495, 0.06322975, 0.04183698,
       0.08710458, 0.04993373, 0.06557475, 0.09172255, 0.0293058 ,
       0.04277725, 0.06848512, 0.04703498, 0.0443767 , 0.04471018,
       0.04984448, 0.07068294, 0.08080884, 0.08080884, 0.06148755,
       0.03074377, 0.04067461, 0.06078019, 0.01248343, 0.02650809,
       0.07430376, 0.04550158, 0.0084287 , 0.07143288, 0.06066877,
       0.04895645, 0.03074377, 0.04250864, 0.        , 0.02837522,
       0.12215819, 0.01356595, 0.04976555, 0.04953584, 0.06627023,
       0.01473499, 0.        , 0.0486631 , 0.04633654, 0.00577206,
       0.04069785, 0.05833221, 0.06593805, 0.0338255 , 0.04449589,
       0.05405929, 0.02510219, 0.05600857, 0.07356913, 0.0342218 ,
       0.06567587, 0.05938196, 0.06025521, 0.03704619, 0.02331262,
       0.06721028, 0.0732645 , 0.07129492, 0.04793958, 0.08882848,
       0.02376242, 0.01343857, 0.02054554, 0.04953584, 0.01248343,
       0.04149592, 0.04423033, 0.04039885, 0.02248469, 0.07722756,
       0.03269231, 0.06706528, 0.02891575, 0.09221946, 0.06679929,
       0.05752259, 0.04445542, 0.08216619, 0.06082456, 0.05308517,
       0.02197935, 0.01850266, 0.02991011, 0.06521739, 0.03783362,
       0.03524537, 0.01248343, 0.01751356, 0.06229425, 0.04324388,
       0.04660147, 0.06855912, 0.04716256, 0.02867549, 0.02285304,
       0.01724202, 0.07881104, 0.06719283, 0.01843024, 0.05172606,
       0.06417938, 0.07024033, 0.06286946, 0.02808418, 0.07160694,
       0.03986374, 0.04914732, 0.05461092, 0.07461272, 0.05694055,
       0.04953584, 0.04914732, 0.        , 0.07211213, 0.02714626,
       0.08683861, 0.02197935, 0.04267182, 0.03575993, 0.04012862,
       0.02510219, 0.1076321 , 0.03256448, 0.04617973, 0.07021046,
       0.02687713, 0.03074377, 0.05301618, 0.08244385, 0.02467022,
       0.05185085, 0.06607503, 0.06451986, 0.00684851, 0.04839693,
       0.02391824, 0.05246427, 0.05350482, 0.03893048, 0.03179815,
       0.05893996, 0.06216699, 0.05868409, 0.01003215, 0.03994412,
       1.        ])
```

In [13]:
```python
#  Returns a list of the top 100 job descriptions that are closest matches to resume document

def recommend_100(df, matrix):
    recommended_jobs = []
    indices = pd.Series(df['title'])
    idx = indices[indices == 'resume'].index[0]
    score_series = pd.Series(matrix[idx]).sort_values(ascending = False)
    top_100_indices = list(score_series.iloc[1:101].index)

    for i in top_100_indices:
        recommended_jobs.append(list(df['title'])[i])

    return recommended_jobs
```

In [14]:
```python
# Exmaple

top_100 = recommend_100(fulldf, sim)
recommend_100(fulldf, sim)
```

Out[14]:
```
['Research Data Scientist',
 'eCommerce Category Manager',
 'actuary_analyst',
 'actuary_analyst',
 'actuary_analyst',
 'Program Coordinator',
 'Data Science Internship',
 'Research Associate/Associate Scientist- Immunology',
 'actuary_analyst',
 'Controls, Modeling, and Simulation Engineer',
 'Remote Mortgage Underwriter',
 'actuary_analyst',
 'Program Coordinator',
 'training_manager',
 'motion_graphics_designer',
 'Profit Growth Management Business Partner',
 'Manager, User Experience',
 'Proposal Manager',
 'Scientist II, Discovery Science',
 'PAC-3 Systems Engineer - Reliability',
 'Associate I Corporate Finance',
 'Quality Engineer',
 'training_manager',
 'Foreign Language Instructor',
 'hr_manager',
 'Data Analyst/Report Writer',
 'Project Analyst',
 'Project Manager',
 'Supply Chain Business Analyst',
 'Area Manager',
 'lead software engineer',
 'accounting_clerk',
 'training_manager',
 'Supply Chain Analyst',
 'actuary_analyst',
 'sales_rep',
 'hr_manager',
 'actuary_analyst',
 'Admissions Coordinator',
 'accounts_payable_specialist',
 'Data Scientist',
 'Marketing Operations Manager',
 'principal software engineer',
 'accounting_clerk',
 'sales_rep',
 'accounting_clerk',
 'Consulting Analyst',
 'document management analyst',
 'Molding Process Engineer',
 'motion_graphics_designer',
 'Data Analyst',
 'Health Program Manager',
 'Business Analyst',
 'training_manager',
 'account_manager',
 'RESG Project Analyst - Real Estate Services Group',
 'Product Designer',
 'Business Analyst - Financial Aid Experience',
 'motion_graphics_designer',
 'hr_manager',
 'Mechanical Engineering Visiting Scientist',
 'Customer Onboarding Specialist',
 'sales_rep',
 'Mechanical Engineer',
 'Leadership Development Coach',
 'Quality Control Engineer',
 'budget_analyst',
 'motion_graphics_designer',
 'Data Analyst',
 'Scientist',
 'Healthcare Access and Value Policy Analyst',
 'Financial Planning Analyst',
 'Executive Director',
 'accounting_clerk',
 'Director of Nursing',
 'Customer Service Representative',
 'data analyst',
 'Seasonal Sales & Service Professional ',
 'Analyst Finance I US',
 'Associate Portfolio Manager',
 'Missile assembly / test engineer associate',
 'Loan Processing Assistant',
 'Electrical Engineer',
 'budget_analyst',
 'Population Health Data Analyst',
 'Communications Specialist',
 'account_manager',
 'Project Management Intern',
 'manufacturing planner',
 'Sr Population Health Analyst',
 'actuary_analyst',
 'budget_analyst',
 'Gameplay Engineer - C++',
 'Mechanical Engineering Assistant Professor',
 'Presentation Designer',
 'Application Analyst',
 'sales_rep',
 'Gallery Client Services Associate',
 'REMOTE Product Design Co-op',
 'Aerospace Engineer']
```

In [15]:
```python
# Prepares the text for display

def format_recommendations(recommendations):
    jobs10 = []
    for job in recommendations:
        job = job.lower().replace("_", " ").title()
        job = job.replace('Hr Manager', 'HR Manager')
        job = job.replace('Care Giver / Hha / Cna', 'Care Giver')
        jobs10.append(job)
# Removes duplicates
        jobs10 = list(dict.fromkeys(jobs10))
    format_jobs = list(jobs10)
    return format_jobs
```

In [16]:
```python
# Example

format_recommendations(top_100)
```

Out[16]:
```
['Research Data Scientist',
 'Ecommerce Category Manager',
 'Actuary Analyst',
 'Program Coordinator',
 'Data Science Internship',
 'Research Associate/Associate Scientist- Immunology',
 'Controls, Modeling, And Simulation Engineer',
 'Remote Mortgage Underwriter',
 'Training Manager',
 'Motion Graphics Designer',
 'Profit Growth Management Business Partner',
 'Manager, User Experience',
 'Proposal Manager',
 'Scientist Ii, Discovery Science',
 'Pac-3 Systems Engineer - Reliability',
 'Associate I Corporate Finance',
 'Quality Engineer',
 'Foreign Language Instructor',
 'HR Manager',
 'Data Analyst/Report Writer',
 'Project Analyst',
 'Project Manager',
 'Supply Chain Business Analyst',
 'Area Manager',
 'Lead Software Engineer',
 'Accounting Clerk',
 'Supply Chain Analyst',
 'Sales Rep',
 'Admissions Coordinator',
 'Accounts Payable Specialist',
 'Data Scientist',
 'Marketing Operations Manager',
 'Principal Software Engineer',
 'Consulting Analyst',
 'Document Management Analyst',
 'Molding Process Engineer',
 'Data Analyst',
 'Health Program Manager',
 'Business Analyst',
 'Account Manager',
 'Resg Project Analyst - Real Estate Services Group',
 'Product Designer',
 'Business Analyst - Financial Aid Experience',
 'Mechanical Engineering Visiting Scientist',
 'Customer Onboarding Specialist',
 'Mechanical Engineer',
 'Leadership Development Coach',
 'Quality Control Engineer',
 'Budget Analyst',
 'Scientist',
 'Healthcare Access And Value Policy Analyst',
 'Financial Planning Analyst',
 'Executive Director',
 'Director Of Nursing',
 'Customer Service Representative',
 'Seasonal Sales & Service Professional ',
 'Analyst Finance I Us',
 'Associate Portfolio Manager',
 'Missile Assembly / Test Engineer Associate',
 'Loan Processing Assistant',
 'Electrical Engineer',
 'Population Health Data Analyst',
 'Communications Specialist',
 'Project Management Intern',
 'Manufacturing Planner',
 'Sr Population Health Analyst',
 'Gameplay Engineer - C++',
 'Mechanical Engineering Assistant Professor',
 'Presentation Designer',
 'Application Analyst',
 'Gallery Client Services Associate',
 'Remote Product Design Co-Op',
 'Aerospace Engineer']
```

In [17]:
```python
# Matches each job title to a job category
# Done by joining job titles to original job description dataframe

def top_100_categories(recommendations):
    with open('pkl/job_descriptions_dec2020.pkl', 'rb') as f:
        df = pickle.load(f)
    user_titles = df[df.title.isin(recommendations)]
    user_titles = user_titles[['title', 'category']]
    category_list = list(user_titles.category)
    return category_list
```

In [19]:
```python
# Example

cat_list = top_100_categories(top_100)
cat_list
```

Out[19]:
```
['management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'sales & retail',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'art, fashion & design',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'accounting & finance',
 'management',
 'management',
 'management',
 'management',
 'management',
 'management',
 'sales & retail',
 'science & engineering',
 'science & engineering',
 'science & engineering',
 'science & engineering',
 'science & engineering',
 'computer & it',
 'computer & it',
 'computer & it',
 'computer & it',
 'computer & it',
 'customer service',
 'customer service',
 'customer service',
 'legal',
 'manufacturing & warehouse',
 'Education',
 'Sales & Retail',
 'Sales & Retail',
 'Accounting & Finance',
 'Accounting & Finance',
 'Science & Engineering',
 'Science & Engineering',
 'Accounting & Finance',
 'Manufacturing & Warehouse',
 'Computer & IT',
 'Science & Engineering',
 'Science & Engineering',
 'Science & Engineering',
 'Accounting & Finance',
 'Management ',
 'Management ',
 'Computer & IT',
 'Science & Engineering',
 'Science & Engineering',
 'Accounting & Finance',
 'Manufacturing & Warehouse',
 'Management ',
 'Sales & Retail',
 'Manufacturing & Warehouse',
 'Science & Engineering',
 'Science & Engineering',
 'Manufacturing & Warehouse',
 'Computer & IT',
 'Computer & IT',
 'Computer & IT',
 'Education',
 'Science & Engineering',
 'Media, Communications & Writing',
 'Human Resources',
 'Accounting & Finance',
 'Management ',
 'Computer & IT',
 'Computer & IT',
 'Computer & IT',
 'Real Estate',
 'Sales & Retail',
 'Accounting & Finance',
 'Management ',
 'Media, Communications & Writing',
 'Healthcare',
 'Computer & IT',
 'Computer & IT',
 'Computer & IT',
 'Science & Engineering',
 'Sales & Retail',
 'Accounting & Finance',
 'Computer & IT',
 'Art, Fashion & Design']
```

In [20]:
```python
# Counts up how many times each category appears

def freq(list_of_categories):
    frequency = []
    unique_words = set(list_of_categories)
    for words in unique_words :
        frequency.append(list_of_categories.count(words))
    return frequency
```

In [21]:
```python
# Example

freq_list = freq(cat_list)
freq_list
```

Out[21]:
```
[40, 11, 5, 2, 1, 3, 5, 11, 4, 1, 8, 5, 1, 12, 1, 1, 5, 36, 10, 1, 2]
```

In [27]:
```python
# Combines the categories with their count values
# This is done in preparation for sample visualization

def viz_data(list_of_categories, frequency_of_categories):
    unique_words = set(list_of_categories)
    unique_words = list(unique_words)
    category_values = dict(zip(unique_words, frequency_of_categories))
    category_dict = {key:val for key, val in category_values.items() if val >= 10}
    names=category_dict.keys()
    size=category_dict.values()

#   Formats category names for display
    format_names = []
    for name in names:
        name = name.title()
        format_names.append(name)
    return format_names, size
```

In [31]:
```python
# Inputting the data into the function

format_names, size = viz_data(cat_list, freq_list)
format_names
```

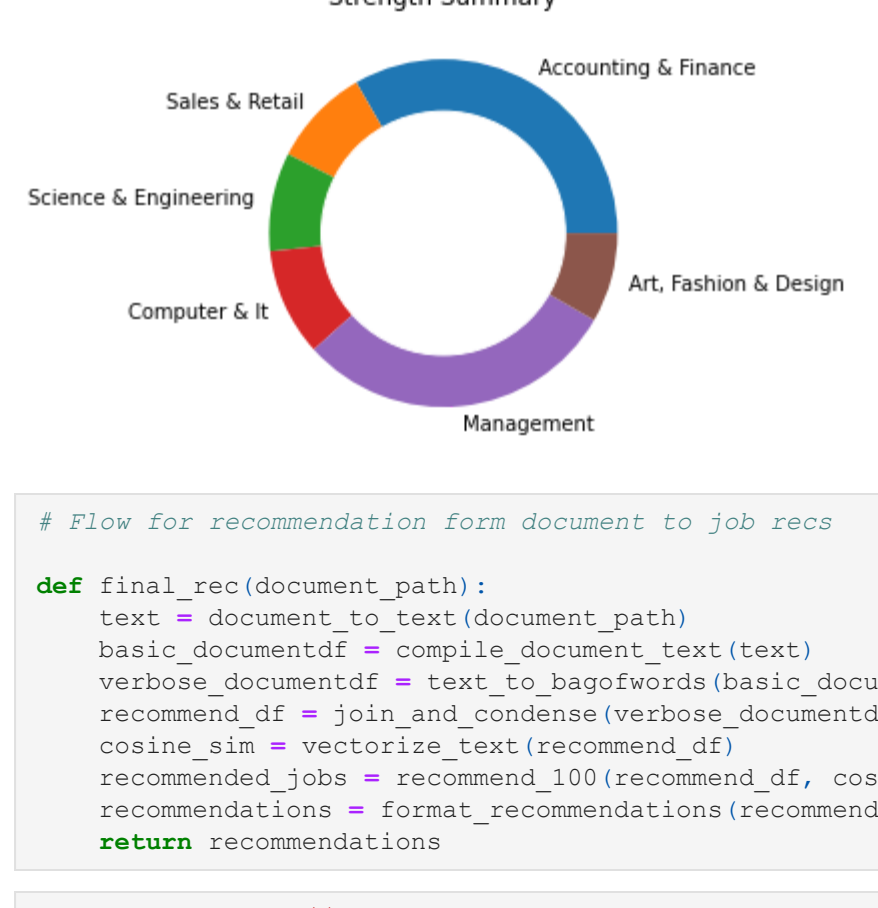
Out[31]:
```
['Accounting & Finance',
 'Sales & Retail',
 'Science & Engineering',
 'Computer & It',
 'Management',
 'Art, Fashion & Design']
```

In [32]:
```python
# Example

size
```

Out[32]:
```
dict_values([40, 11, 11, 12, 36, 10])
```

In [29]:
```python
# Creates a visualization showing the distribution of categories

def make_viz(names_of_categories, size_of_categories):

# Creates a circle in the center of the plot, turning a pie into a donut
    my_circle=plt.Circle( (0,0), 0.7, color='white')

# Labels and creates visualization
    plt.title('Strength Summary')
    plt.pie(size_of_categories, labels=names_of_categories)
    p=plt.gcf()
    p.gca().add_artist(my_circle)
    plt.show()
```

In [30]:
```python
# Example

make_viz(format_names, size)
```



In [34]:
```python
# Flow for visuzlization from document to viz

def analyze(document_path):
    resume_text = document_to_text(document_path)
    basic_documentdf = compile_document_text(resume_text)
    verbose_documentdf = text_to_bagofwords(basic_documentdf)
    recommend_df = join_and_condense(verbose_documentdf)
    cosine_sim = vectorize_text(recommend_df)
    recommended_jobs = recommend_100(recommend_df, cosine_sim)
    final_jobs10 = format_recommendations(recommended_jobs)
    category_list = top_100_categories(recommended_jobs)
    frequency = freq(category_list)
    names, size = viz_data(category_list, frequency)
    strength_summary = make_viz(names, size)
```

In [35]:
```python
analyze('https://rookieplay-uploads.s3.amazonaws.com/media/Bell-Resume_Apr2021_1NJ3wOt.pdf')
```


In [38]:
```python
# Flow for recommendation form document to job recs

def final_rec(document_path):
    text = document_to_text(document_path)
    basic_documentdf = compile_document_text(text)
    verbose_documentdf = text_to_bagofwords(basic_documentdf)
    recommend_df = join_and_condense(verbose_documentdf)
    cosine_sim = vectorize_text(recommend_df)
    recommended_jobs = recommend_100(recommend_df, cosine_sim)
    recommendations = format_recommendations(recommended_jobs)
    return recommendations
```

In [39]:
```python
final_rec('https://rookieplay-uploads.s3.amazonaws.com/media/Bell-Resume_Apr2021_1NJ3wOt.pdf')
```

```
Out[39]: ['Research Data Scientist',
 'Ecommerce Category Manager',
 'Actuary Analyst',
 'Program Coordinator',
 'Data Science Internship',
 'Research Associate/Associate Scientist– Immunology',
 'Controls, Modeling, And Simulation Engineer',
 'Remote Mortgage Underwriter',
 'Training Manager',
 'Motion Graphics Designer',
 'Profit Growth Management Business Partner',
 'Manager, User Experience',
 'Proposal Manager',
 'Scientist Ii, Discovery Science',
 'Pac-3 Systems Engineer - Reliability',
 'Associate I Corporate Finance',
 'Quality Engineer',
 'Foreign Language Instructor',
 'HR Manager',
 'Data Analyst/Report Writer',
 'Project Analyst',
 'Project Manager',
 'Supply Chain Business Analyst',
 'Area Manager',
 'Lead Software Engineer',
 'Accounting Clerk',
 'Supply Chain Analyst',
 'Sales Rep',
 'Admissions Coordinator',
 'Accounts Payable Specialist',
 'Data Scientist',
 'Marketing Operations Manager',
 'Principal Software Engineer',
 'Consulting Analyst',
 'Document Management Analyst',
 'Molding Process Engineer',
 'Data Analyst',
 'Health Program Manager',
 'Business Analyst',
 'Account Manager',
 'Resg Project Analyst - Real Estate Services Group',
 'Product Designer',
 'Business Analyst - Financial Aid Experience',
 'Mechanical Engineering Visiting Scientist',
 'Customer Onboarding Specialist',
 'Mechanical Engineer',
 'Leadership Development Coach',
 'Quality Control Engineer',
 'Budget Analyst',
 'Scientist',
 'Healthcare Access And Value Policy Analyst',
 'Financial Planning Analyst',
 'Executive Director',
 'Director Of Nursing',
 'Customer Service Representative',
 'Seasonal Sales & Service Professional ',
 'Analyst Finance I Us',
 'Associate Portfolio Manager',
 'Missile Assembly / Test Engineer Associate',
 'Loan Processing Assistant',
 'Electrical Engineer',
 'Population Health Data Analyst',
 'Communications Specialist',
 'Project Management Intern',
 'Manufacturing Planner',
 'Sr Population Health Analyst',
 'Gameplay Engineer - C++',
 'Mechanical Engineering Assistant Professor',
 'Presentation Designer',
 'Application Analyst',
 'Gallery Client Services Associate',
 'Remote Product Design Co-Op',
 'Aerospace Engineer']

In [ ]:
```