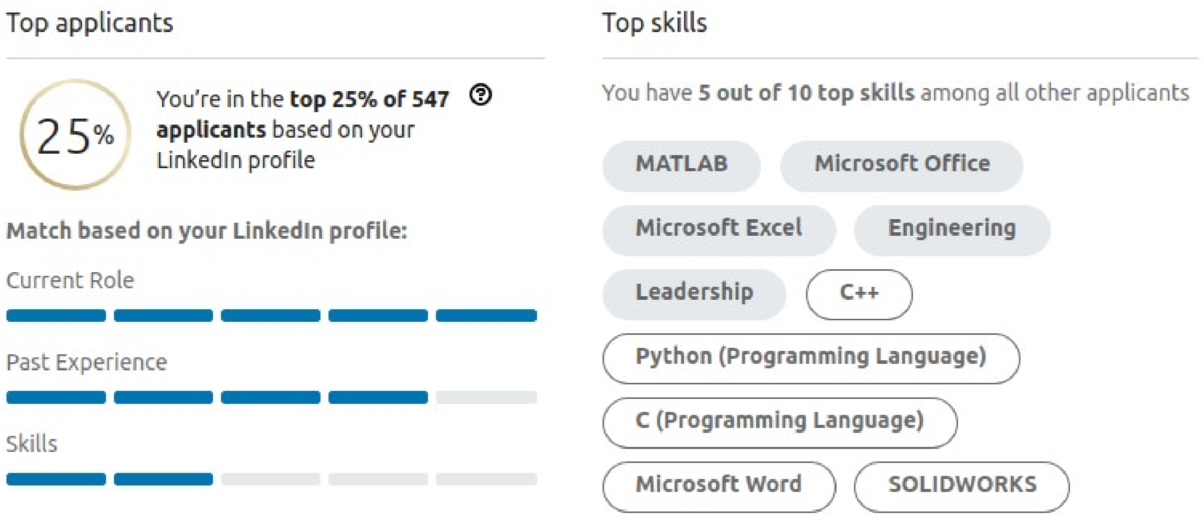
Introduction

A.1 Background on This Project

During this current job search there was an issue with almost instant rejection due to keyword matching on the Applicant Tracking System, or ATS. This system is the first barrier in job application as this tool scans a resume and will scan a resume, along with dozens more, to ensure that the most viable candidates get through the first filter of application. The ATS will then rank the submitted application in comparison to the other applications gathered.

This is done in many places, but in this example LinkedIn will be used as an ATS example:



Top applicants

25%

You're in the **top 25% of 547 applicants** based on your LinkedIn profile

Match based on your LinkedIn profile:

Current Role

Past Experience

Skills

Top skills

You have **5 out of 10 top skills** among all other applicants

MATLAB, Microsoft Office, Microsoft Excel, Engineering, Leadership, C++, Python (Programming Language), C (Programming Language), Microsoft Word, SOLIDWORKS

In this entry, a comparative ATS-like python script will be used to compare a resume to a job posting.

A.2 Data Selection

The datasets in this project will consist of the following:

- Job Description (not included)
- Resume used for the description (not included due to reference contact info inside)
- A stopwords file to remove irrelevant text from the search

The focus here will be to measure comparison between the resume and the job description.

B. Methodology

The method for comparison for this project will be the cosine similarity. This,, document matching method is performed by matching the total amount of common words between said documents. In this case job descriptions and resumes are being compared. At large scale, this is being used to filter many resumes at an application site, check for plagiarism, or find similar or matching documents in a location.

First, all libraries must be imported and *matplotlib* must be brought inline.

```
 import docx2txt
import collections
import pandas as pd
import matplotlib.pyplot as plt
import os
%matplotlib inline
```

The library *docx2txt* brings the docx files with all the resume and job information into a manageable text file to be used.

Importation and assignment to a variable of the job posting and the resume will be completed as follows:

```
 posting = docx2txt.process("posting")
resumeNAVAIR = docx2txt.process("resume")
```

The importation of the resume and the posting can be checked by printing the file.

```
print(posting)
print(resume)
```

After that is complete, the text will then be added to a list for further analysis.

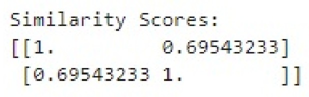
```
text=[resume,posting]
```

Next the text will be run through a *count vectorizer* to create a count vectorizor object. The text documents will then be converted to a matrix for token counts.

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
count_matrix = cv.fit_transform(text)
```

The cosine similarity scores will then be calculated.

```
from sklearn.metrics.pairwise import cosine_similarity

#Print the similarity scores
print("\nSimilarity Scores:")
print(cosine_similarity(count_matrix))
```



```
Similarity Scores:
[[1.         0.69543233]
 [0.69543233 1.        ]]
```

With the similarity matrix displayed, the percentage match can be now be shown.

```
matchPercentage = cosine_similarity(count_matrix)[0][1] *100
matchPercentage = round(matchPercentage, 2)
print("Your resume matches about "+ str(matchPercentage)+ "% of the job description.")
```

Your resume matches about 69.54% of the job description.

It looks like there is some work to be done.

Before the next step can be completed, it needs to be known what stop words are. Stop words are words that are filtered out during the processing of natural language data. These words are words such as: *I, we, and, for, the, can* or any other words that are determined.

An entire file can be created to designate what stop words are needed to be filtered out, but a stop words file from GitHub will be used to expedite this process. Additional words will be added on to this list as seen fit.

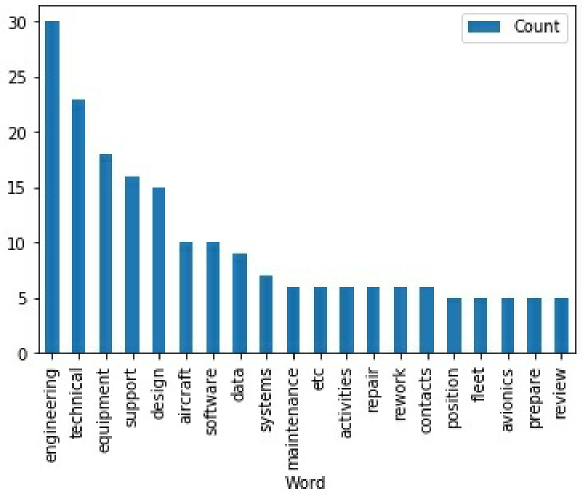
First, the word count of the job posting will be looked at.

```
import pandas as pd
```

```
 # Read input file, note the encoding is specified here

a= posting

# Stopwords
stopwords = set(line.strip() for line in open('stopwords.txt'))
stopwords = stopwords.union(set(['assigned','provide','incumbent','related','factor','include','changes','cited']))
# Instantiate a dictionary, and for every word in the file,
# Add to the dictionary if it doesn't exist. If it does, increase the count.
wordcount = {}
# To eliminate duplicates, remember to split by punctuation, and use case demiliters.
for word in a.lower().split():
    word = word.replace(".","")
    word = word.replace(",","")
    word = word.replace(":","")
    word = word.replace("\"","")
    word = word.replace("!","")
    word = word.replace("â€œ","")
    word = word.replace("â€˜","")
    word = word.replace("*","")
    if word not in stopwords:
        if word not in wordcount:
            wordcount[word] = 1
        else:
            wordcount[word] += 1
# Print most common word
n_print = int(input("How many most common words to print: "))
print("\nOK. The {} most common words are as follows\n".format(n_print))
word_counter = collections.Counter(wordcount)
for word, count in word_counter.most_common(n_print):
    print(word, ": ", count)

# Create a data frame of the most common words
# Draw a bar chart
lst = word_counter.most_common(n_print)
df = pd.DataFrame(lst, columns = ['Word', 'Count'])
df.plot.bar(x='Word',y='Count')
```

Results in bar graph visualization:

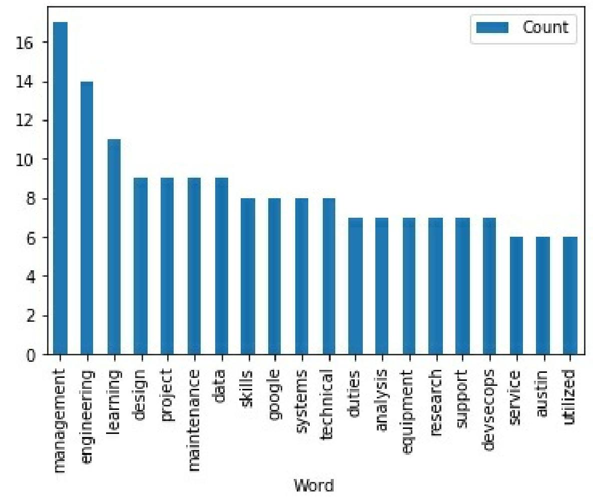
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c46fbb4e0>
```



The same technique will be used on the resume with the following stop words added instead:
'assigned','provide','incumbent','related','factor','include','changes','cited','–','2020','apr','ibm','linkedin','tx','-'.

Results in bar graph visualization:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c46ea3128>
```



C. Results

As previously stated, the initial resume vs job posting resulted in a cosine similarity matching of 69.54%. This leaves a lot of room to correct non matching terms in the resume vs the job posting.

The disconnect for the top 3 items within the list comparison is that the job posting lists *engineering, technical,* and *management* as the top three recurring words vs the resume listing *management, engineering,* and *learning* as the top three.

D. Discussion

These key words will need to be adjusted in order to match the ATS tracking metrics much closer and allow this resume to pass the initial screening filters.

However, this does not mean that the ATS is be be-all / know-all in the application process.

This resume was submitted to USAJobs for a position as a depot level engineer at a Marine base and was promptly rejected. When the resume was rejected, the applicant contacted the point of contact on the job listing, talked with them, and the resume was forwarded to another department. This lead to several interviews with Senior Engineering staff.

It can be said that tenacity does play a factor in the job search process, sometimes.

E. Conclusion

The ATS is a system that needs to be overcome in order to have a resume pass through the initial filters in the job search activity. However, it is not the only thing that is measured. If a resume is submitted and rejected, do not hesitate to follow back up and ask what the cause of the rejection was for. On the other hand, one should always prepare their resume to match each and every job description as closely as possible.

Further work on this project will include tabular word counts for direct display.