Profile Matcher

2021F BIA 660-A: Web Mining

Instructor: Jingyi Sun

Introduction

It's hard to land your first job offer and harder to search for specific positions. Keeping up with all the open positions can be time-consuming and tedious. Looking for a specific position can be even more challenging. Our project aims to assist students in ranking job opportunities based on new graduate and internship positions and matching resumes. Software Engineer, Data Engineer, and Data Scientist positions in technology companies are our major focus. To achieve this objective, we are crawling LinkedIn data and applying machine learning to recommend a candidate for appropriate positions.

Literature review

In order to progress past the resume screening stage and onto the interview stage, we have to adapt to the new reality of candidate screening.

Candidate selection software helps employers match resumes to the required jobs. The skills and keywords on the employer's list are matched with the resumes to determine your potential value to the organization. A resume with the highest score relevant to the employer's keywords, phrases, and years of experience will be given priority for further review. Ultimately, the software simply scores resumes in order to determine which candidates are most qualified for moving up the ladder for a human supervisor to review.

From the candidate's perspective, even applying to good job matches will increase their chances of getting to the next round. Most of the existing job recommendation websites offer multiple job recommendations based on a range of features. Oftentimes it is a time-consuming task to apply for all the recommended jobs at once and by procrastinating there is a risk of missing out on the deadlines. To know which jobs to apply for first is a difficult task.

The system described in the paper "*A Machine Learning approach for automating Resume Recommendation System*"[1] expedites the selection process by identifying candidates whose CVs are closest to the job description. This suggests that if we build a matching based on job descriptions and resume contents, we would have a better chance of getting a match.

Based on job recommender systems, a research paper [4] surveys the e-recruitment process and recommender approaches for creating personalized recommender systems for matching candidates to jobs. A paper named "*Matching Resumes to Jobs via Deep Siamese Network*"[3] shows the importance and challenges of recommending appropriate jobs for job seeking candidates by matching semi structured resumes of candidates to job descriptions. From these papers, it becomes evident we will need to develop an own recommendation system that will give a good match over job descriptions and identify the features to get higher relevancy matches for positions of Software Engineer, Data Engineer, and Data Scientist.

Why LinkedIn?

In the study “Branding in Social Media”[2], a survey was sent to a pool of professionals and students via social media (LinkedIn and Facebook). The results of the study reveal that LinkedIn ranks first among the most common platforms used to find a job by the people surveyed. 71% use this network frequently or even always. LinkedIn also ranks first from a candidate’s perspective to develop a personal brand as well.

Due to the popularity of LinkedIn, we have chosen it as a source of data, as we are examining how recommending a post based on a student's profile can save them time.

Research question

The project aims to aid students in their job hunt by reducing their efforts in tracking and applying to internships and new grad positions, by recommending them a prioritized list of jobs they can apply for. Our hope is that the system helps the students save time and not miss out on the deadlines provided by the companies for the application.

Methodology

We are crawling data from LinkedIn to gather the job descriptions made available by tech companies for various positions for the year 2021 using web scraping methodologies. The data pulled includes positions for Software Engineers, Data Engineers, and Data Scientists. This data will be used to develop a recommendation system that suggests the best position a student can apply for. Using machine learning algorithms, we will make recommendations based on the user's input. We will investigate which feature sets will provide the most effective recommendations in this project.

During Web Scraping lectures, we explored different approaches to crawling and scraping web pages. The NLP II Lecture covered a number of topics, including Cosine Similarity and vectorization methods. In our project, we applied these concepts.

Web scraping

We took the below approaches to find the best way to crawl data from LinkedIn.

I. Using Web Scraping (Beautiful Soup)
   - Due to LinkedIn's added security, remote scraping of websites is restricted; thus, Beautiful Soup was unable to return the desired results.
   - We tried to scrape the LinkedIn website by collecting the list of LinkedIn job URLs.
   - We have collected data from individual job description urls. We planned on using this text data to diversify our training set. We have saved the text in a text file, and the file naming

is done using regex, where we have matched the name from the url links. The code snippet of the same can be found below.

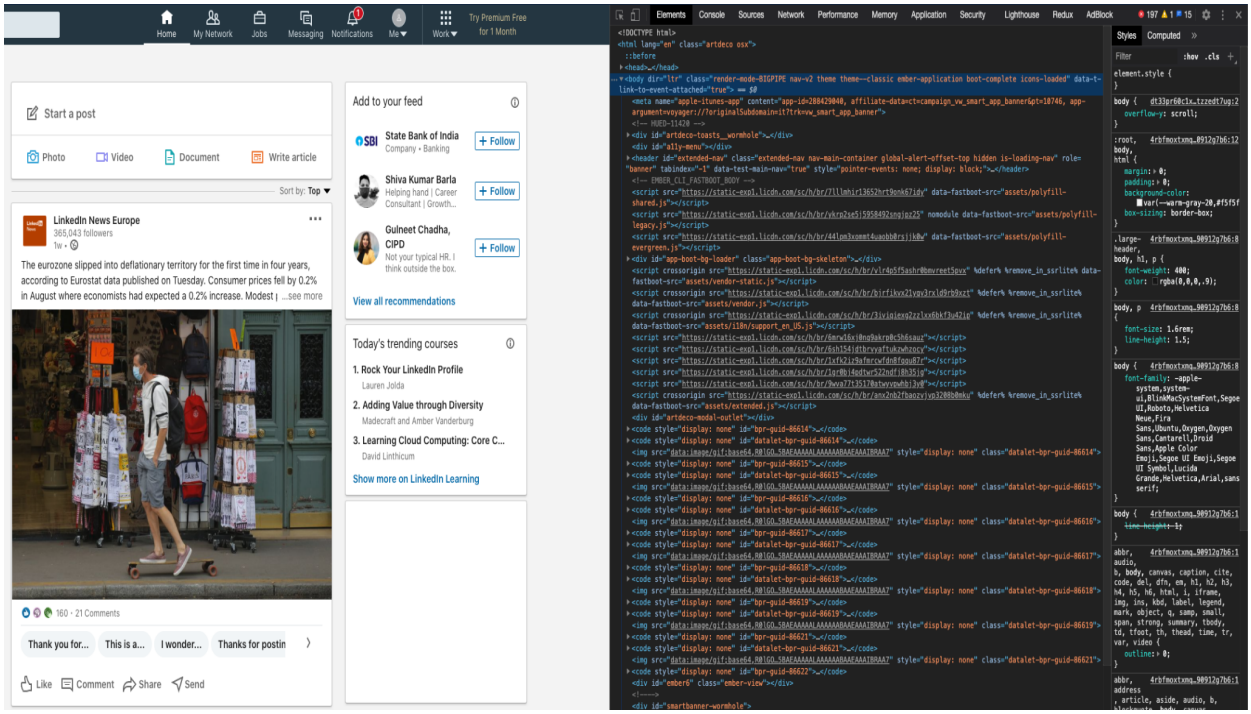
```
import re
import os
import json
import trafilatura

def scrape_text(link):
    '''
    read link
    create filename using regex
    open file
    write all text to the newly created file
    '''
    if link != None:
        path = '/Users/archanakalburgi/Downloads/project_660/descriptions'
        m = re.match(r'https:\/\/(.*?)\/(.*)(\/|\?gh_jid=)(.*)$',link)
        if m:
            downloaded = trafilatura.fetch_url(link)
            result = trafilatura.extract(downloaded)
            file_name = m.group(1)+m.group(4)+'.txt'
            if result:
                with open(os.path.join(path,file_name),'a') as the_file:
                    the_file.write(result)


if __name__ == '__main__':
    '''
    reading links from the links.txt
    passing all links one by one to scrape_text function to get job descriptions
    '''
    with open('links.txt') as f:
        for line in f:
            scrape_text(line)
```

II. Using Selenium/Chrome Driver implemented through LinkedInScraper library

- LinkedIn had restrictions on scraping, so we decided to use Selenium instead
- Selenium was chosen due to its ability to simulate human behavior by launching web pages in real time and obtaining information as needed.
- The fact that Selenium can perform clicks like a real user was an added benefit
- Having narrowed down to Selenium, we used the already available LinkedInScraper Library , which needed a custom filter to obtain the required information for job related data.



(src :https://raw.githubusercontent.com/spinlud/py-linkedin-jobs-scraper/master/images/img3.png)

LinkedinScraper has been used to get job related information by applying the following filters. Data is stored in CSV files, then read as a data frame.

1. Columns of the data frame:
    a. title
    b. company
    c. location
    d. date
    e. link
    f. apply_link
    g. description
    h. seniority_level

2. Code snippet of extracting the data using LinkedinScraper

```python
import logging
from linkedin_jobs_scraper import LinkedinScraper
from linkedin_jobs_scraper.events import Events, EventData
from linkedin_jobs_scraper.query import Query, QueryOptions, QueryFilters
from linkedin_jobs_scraper.filters import RelevanceFilters, TimeFilters, TypeFilters, ExperienceLevelFilters, RemoteFilters
import csv
import pandas as pd

'''
change
slow_mo=3(increase it)
limit=3500
'''

# Change root logger level (default is WARN)
logging.basicConfig(level = logging.INFO)

linked_data = []
def on_data(data: EventData):
    linked_data.append([data.title, data.job_function, data.location,data.date, data.link ,data.apply_link, data.description, data.seniority_level])


def on_error(error):
    print('[ON_ERROR]', error)


def on_end():
    columns = ['title', 'company', 'location', 'date', 'link', 'apply_link', 'description', 'seniority_level']
    df = pd.DataFrame(linked_data, columns=columns)
    df.to_csv('linkedin_jobs.csv', index=False, encoding='utf-8')
    print('[ON_END]')


scraper = LinkedinScraper(
    chrome_executable_path="/Users/archanakalburgi/Downloads/project_660/chromedriver", # Custom Chrome executable path (e.g. /foo/bar/bin/chromedriver)
    chrome_options=None,  # Custom Chrome options here
    headless=True,  # Overrides headless mode only if chrome_options is None
    max_workers=1,  # How many threads will be spawned to run queries concurrently (one Chrome driver for each thread)
    slow_mo=2,  # Slow down the scraper to avoid 'Too many requests (429)' errors
)

# Add event listeners. call back
scraper.on(Events.DATA, on_data)
scraper.on(Events.ERROR, on_error)
scraper.on(Events.END, on_end)

# tasks
queries = [
    Query(
        query='Engineer',
        options=QueryOptions(
            locations=['United States'],
            optimize=True,
            limit=100,
            filters=QueryFilters(
                relevance=RelevanceFilters.RECENT,
                time=TimeFilters.ANY,
                type=[TypeFilters.INTERNSHIP],
                experience=None,
            )
        )
    )
]
```

Text Matching

Vectorizing and Embeddings

Computer models can only understand numbers, not words. Therefore, we must convert our documents into vectors. Vectors are numbers that represent each word as a feature.

There are several great strategies for turning a document into a vector.

- Countvectorizer
- TF- IDF Vectorizer
- Embeddings

Count-vectors describe documents based on how many words are in them, and TF-IDF vectorizations use floating point numbers to specify how specific each word is to a given document. These strategies retain the vocabulary used in each document, but throw out the order and relationships between the words.

In the case of word embeddings, however, words are embedded in n-dimensional space, where n is a developer-defined value. The dimension corresponding to each position in the vector array represents a relationship between words. Using this method, it is possible to identify the semantic relationship / context around the occurrence of the words rather than simply their frequency of appearance.

Cosine Similarity

We assumed matching two documents is equivalent to matching two sentences. In order to match job and resume documents, we treat each job and resume document as a long sentence. Upon converting the documents to vectors in the same dimensions, we use cosine similarity to match text.

According to the Cosine Similarity, if two vectors point roughly in the same direction in space, they are similar. Based on the cosine similarity, we can calculate the matching percentage between the two documents.

The steps for matching a resume to a job are as follows:

- Transform Resume Content into one single sentence
- Transform Job Description into one single sentence
- Extract features from the two documents by transforming them into vectors in defined dimensions
- Apply Cosine Similarity to find the match between these two sentences.

```
documents = (
"The sky is blue",
"The sun is bright"
)
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print(tfidf_matrix.shape)
```

```
(2, 6)
```

```
cosine_similarity(tfidf_matrix[0:1], tfidf_matrix)
# first is similar to itself
# first is 33% similar to second sentence
```

```
array([[1.        , 0.33609693]])
```

```
cosine_similarity(tfidf_matrix)[0][1]
# from the matrix compare ith to jth
# so that is compare 0th sentence to 1th sentence
```

```
0.3360969272762575
```

```
tfidf_matrix[0].shape
```

```
(1, 6)
```

```
cosine_similarity(tfidf_matrix)
```

```
array([[1.        , 0.33609693],
       [0.33609693, 1.        ]])
```

Fig. Code Snippet of how two sentences are matched using Cosine Similarity

Note: In order to test our text matching across different approaches, we have defined a common similarity comparison function to be called to keep some consistency.

The vector shape input for the cosine function thus requires shape: (2, dimension)

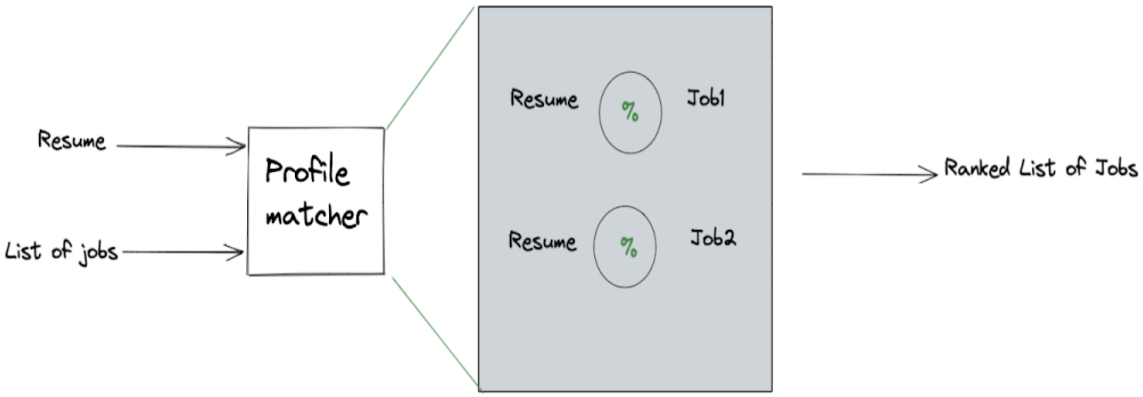
1. Job (dimension,)
2. Resume (dimension,)

Defined Common Similarity Comparison Across Approached Explored

> matrixx shape : (2, dim)

```
[ ]
def calculate_similarity(matrixx):
  ## from section for "Cosine Similarity"
  ## [0][1] position gives the matching value for two documents being compared
  matchPercentage = cosine_similarity(matrixx)[0][1] * 100
  matchPercentage = round(matchPercentage, 2) # round to two decimal
  print("Your resume matches about "+ str(matchPercentage)+ "% of the job description.")
  return matchPercentage
```

Initial Plan of Action



Steps:

- Scrape LinkedIn
- Analyze the resumes
- Leverage Natural Language Processing and Text Analysis
- Detect various skill sets
- Compare the dataset to skill sets
- Analyze the number of matches
- Generated match percentage
- Provide a prioritized list of jobs

We originally planned on building a recurrent neural network to match up the resumes and the job descriptions we have collected. However, we found that using the **cosine similarities** between a job and a resume, as well as some feature engineering, met our objectives.

Analysis

Data Cleaning and Processing

Data before preprocessing:

	title	company	location	date	link	apply_link	description	seniority_level
0	Process Manufacturing Engineer (Entry Level)	Engineering and Information Technology	United States	2021-10-21	https://www.linkedin.com/jobs/view/process-man...	https://www.linkedin.com/jobs/view/externalApp...	What Makes Honda The Best?\n\nAre you an innov...	Not Applicable
1	Entry Level Engineer / Engineering Development...	Engineering and Information Technology	United States	2021-09-25	https://www.linkedin.com/jobs/view/entry-level...	https://www.linkedin.com/jobs/view/externalApp...	What Makes Honda The Best?\n\nAre you an innov...	Not Applicable
2	Quality Engineer	Quality Assurance	United States	2021-10-29	https://www.linkedin.com/jobs/view/quality-eng...	https://www.linkedin.com/jobs/view/externalApp...	What Makes Honda The Best?\n\nAre you an innov...	Not Applicable
3	Process Manufacturing Engineer (Entry Level)	Engineering and Information Technology	United States	2021-09-25	https://www.linkedin.com/jobs/view/process-man...	https://www.linkedin.com/jobs/view/externalApp...	What Makes Honda The Best?\n\nAre you an innov...	Not Applicable
4	Entry Level Global Product Development - Hardw...	Engineering and Information Technology	United States	2021-09-04	https://www.linkedin.com/jobs/view/entry-level...	https://www.linkedin.com/jobs/view/externalApp...	About GM\n\nThere's never been a more exciting...	Not Applicable

We preprocessed the scraped data using NLTK. Below are the preprocessing steps:

1. Transformed the data to lowercase
2. Removed punctuations
3. Removes stopwords
4. Tokenized the words
5. Lemmatized all the tokens

Data after preprocessing:

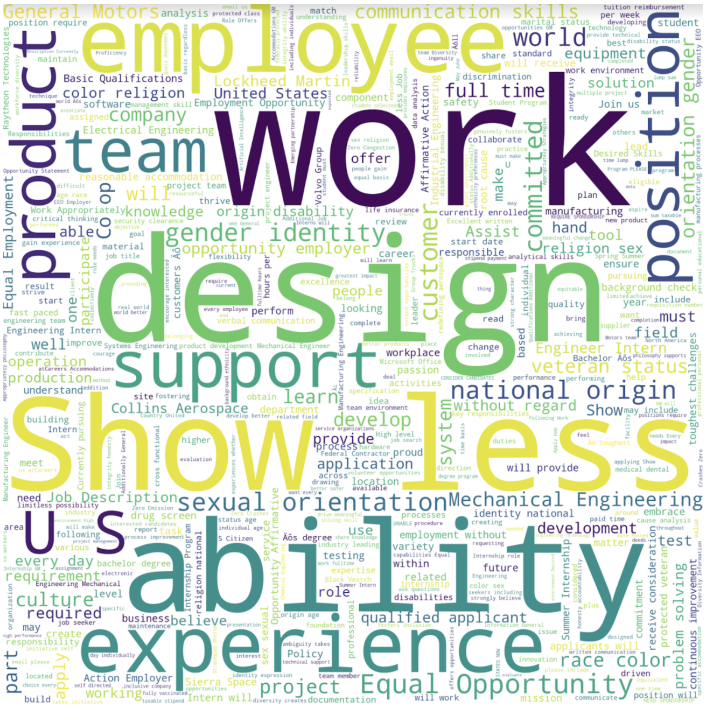
	description	clean_msg	msg_lower	msg_tokenied	no_stopwords
0	What Makes Honda The Best?\n\nAre you an innov...	What Makes Honda The Best Are you an innovato...	what makes honda the best are you an innovato...	[what makes honda the best are you an innovat...	[what makes honda the best are you an innovat...
1	What Makes Honda The Best?\n\nAre you an innov...	What Makes Honda The Best Are you an innovato...	what makes honda the best are you an innovato...	[what makes honda the best are you an innovat...	[what makes honda the best are you an innovat...
2	What Makes Honda The Best?\n\nAre you an innov...	What Makes Honda The Best Are you an innovato...	what makes honda the best are you an innovato...	[what makes honda the best are you an innovat...	[what makes honda the best are you an innovat...
3	What Makes Honda The Best?\n\nAre you an innov...	What Makes Honda The Best Are you an innovato...	what makes honda the best are you an innovato...	[what makes honda the best are you an innovat...	[what makes honda the best are you an innovat...
4	About GM\n\nThere's never been a more exciting...	About GM Theres never been a more exciting ti...	about gm theres never been a more exciting ti...	[about gm theres never been a more exciting t...	[about gm theres never been a more exciting t...
...	...	...	...	...	...
2864	Who We Are\n\nAddepar,Ãôs mission is to bring ...	Who We Are Addepar,Ãôs mission is to bring da...	who we are addepar,ãôs mission is to bring da...	[who we are addepar,ãôs mission is to bring d...	[who we are addepar,ãôs mission is to bring d...
2865	We,Ãôre looking for problem solvers, innovator...	We,Ãôre looking for problem solvers innovators...	we,ãôre looking for problem solvers innovators...	[we,ãôre looking for problem solvers innovator...	[we,ãôre looking for problem solvers innovator...
2866	Requisition ID: 123599\n\nJob Level: Internshi...	Requisition ID 123599 Job Level Internship D...	requisition id 123599 job level internship d...	[requisition id 123599 job level internship ...	[requisition id 123599 job level internship ...
2867	Requisition ID: 123639\n\nJob Level: Internshi...	Requisition ID 123639 Job Level Internship D...	requisition id 123639 job level internship d...	[requisition id 123639 job level internship ...	[requisition id 123639 job level internship ...
2868	Join us as we pursue our vision to make machin...	Join us as we pursue our vision to make machin...	join us as we pursue our vision to make machin...	[join us as we pursue our vision to make machi...	[join us as we pursue our vision to make machi...

2869 rows × 5 columns

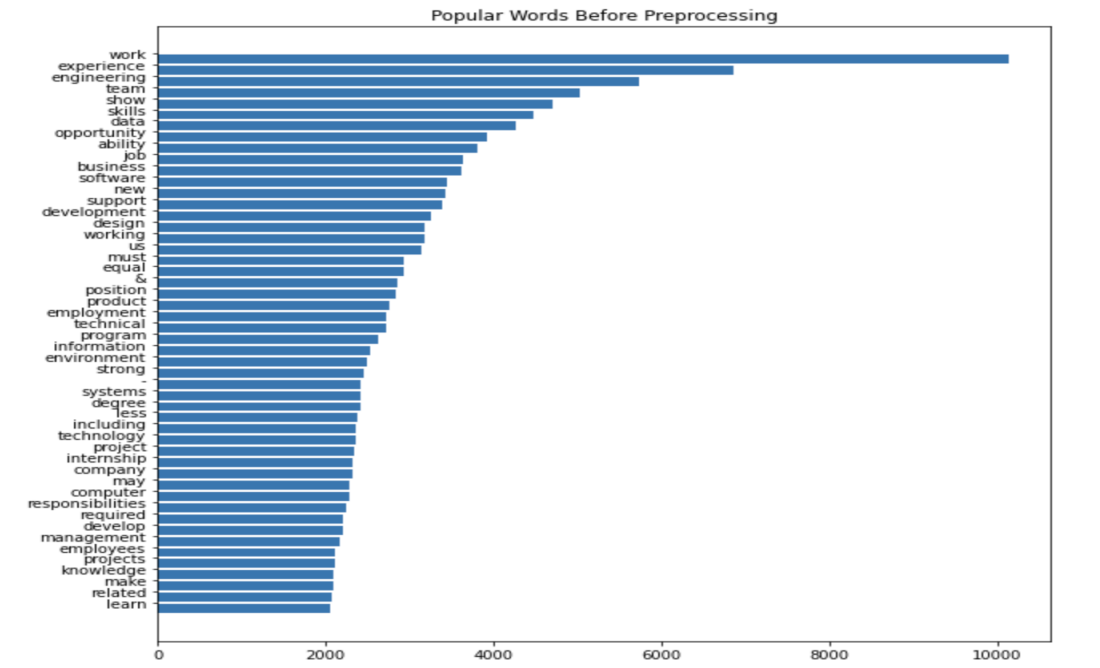
Exploratory Data Analysis on the data

Before preprocessing

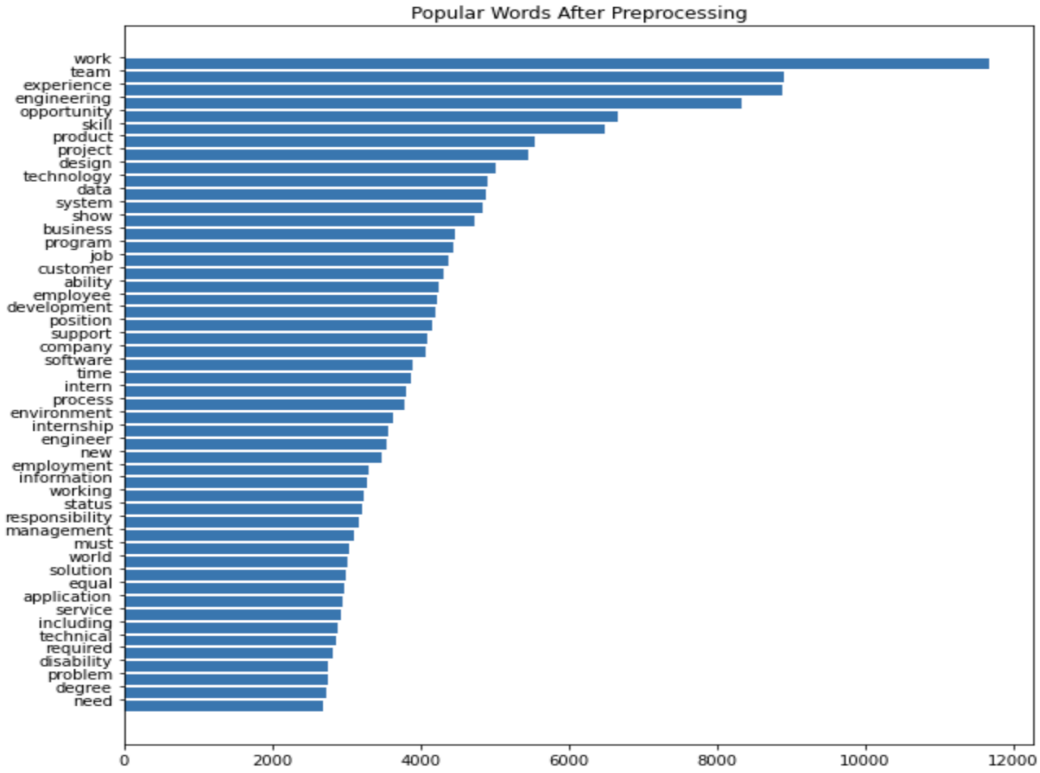
After preprocessing



Popular words before preprocessing



Polar words after preprocessing

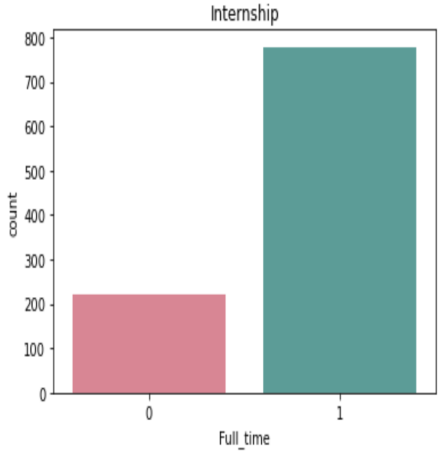


Dataset details

Number of rows (job descriptions) : 2869
Number of columns : 10

Columns of the dataset:
'title', 'Company','location', 'date', 'link', 'apply_link', 'description', 'seniority_level', 'job_function', 'place'

Seniority_level unique values:
'Not Applicable', 'Entry level', 'Internship', 'Associate', 'Mid-Senior level','Director'



Initial Analysis

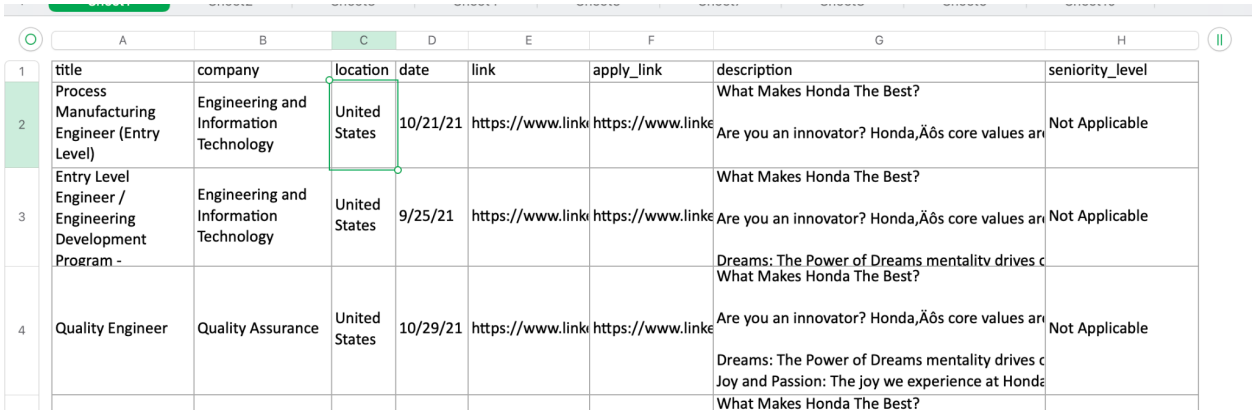
We have analyzed the cosine similarity over count frequency(CV), term frequency(TF), term frequency inverse document frequency(TF-IDF), pretrained embedding and trained embeddings using data scrapped from Linkedin.

	CV	TF	TF-IDF	PreTrained-Embeddings	Title
0	48.69	48.69	35.58	85.79	Security Lead - Prime Air
1	53.18	53.18	38.7	80.16	Learning and Development Business Partner, Tech
2	56.03	56.03	42.69	85.72	X-Ray Technologist
3	45.19	45.19	30.94	79.82	AR/VR Systems Frameworks Engineer
4	57.36	57.36	43.92	87.99	Data Engineer, Infrastructure Strategy
5	62.12	62.12	48.86	81.83	Data Engineer, Analytics (Domain Specialist)
6	44.21	44.21	29.76	90.69	Transportation Planner
7	55.83	55.83	42.52	85.34	X-Ray Technologist
8	47.25	47.25	34.18	78.31	Sourcing Lead - Software Engineering
9	46.92	46.92	31.73	89.72	Line Producer, The Choice
10	30.66	30.66	20.04	87.45	Microsoft Search Consultant (100% Remote)

Implementation Details

Step 1: LinkedIn Scraper

As a first step, we accessed LinkedIn as a source for job descriptions, and got the data as csv files. We will therefore create a Data Frame with the columns analyzed in the EDA. Example,



	A	B	C	D	E	F	G	H
1	title	company	location	date	link	apply_link	description	seniority_level
2	Process Manufacturing Engineer (Entry Level)	Engineering and Information Technology	United States	10/21/21	https://www.linke	https://www.linke	What Makes Honda The Best? Are you an innovator? Honda‚Äôs core values ar	Not Applicable
3	Entry Level Engineer / Engineering Development Program -	Engineering and Information Technology	United States	9/25/21	https://www.linke	https://www.linke	What Makes Honda The Best? Are you an innovator? Honda‚Äôs core values ar Dreams: The Power of Dreams mentality drives c	Not Applicable
4	Quality Engineer	Quality Assurance	United States	10/29/21	https://www.linke	https://www.linke	What Makes Honda The Best? Are you an innovator? Honda‚Äôs core values ar Dreams: The Power of Dreams mentality drives c Joy and Passion: The joy we experience at Honda	Not Applicable
							What Makes Honda The Best?	

Step 2: Getting a Resume

The resume here is the one of the applicants. In this case, the resume that we are matching to various jobs is my own. Suppose an applicant receives a low match result. He/she can re-word the information or add / delete information and re-run the screening to see if his/her edits helped.

Considering most resumes are in Adobe PDF format, we wanted to convert the resume to text format so that we could process natural language and vectorize to generate feature sets.

During the Web Scraping lecture, we learned how to parse PDF files using the Tika parser. Given a set of resumes with different file types, such as doc, docx, and pdf, those files will be processed by Tika to get the raw text, which will be stripped of table layouts, font type, and font colors.

We also added below pre-processing step for cleaning resume after it passes through parser:

```
# 1. remove all \n{3} with \n\n
# 2. keep \n\n as separation of headings
parsed_resume = re.sub(r'\n{3,}', '\n\n', parsed["content"]).strip()
```

Step 3: Analysis of Similarity Matching

The search for similarities was undertaken using various approaches, which initially failed to yield decent results.

- Countvectorizer and TF-IDF Vectorizer
    - In light of the fact that these are traditional NLP processes that have proven to work in the past, they were the first ones to be tested against a job description and my resume
    - A limitation of TF-IDF and Countvectorizer is that they are a bag of words approach (**keywords and frequency of matching words**) that does not take into account the placement and similarity of the words.
    - In the case of Job Descriptions and Resumes, this approach did not prove to be successful, since the content can have different synonyms for the subject matter they describe.
- Pre Trained Word2Vec Embeddings
    - We can achieve better results if we **match the semantic meaning of words** in a pair of documents rather than just matching keywords, based on the understanding given above.
    - We then decided to go with Word2Vec. Because Word2Vec requires a big set of training data to work effectively, we decided to try out the model using pre-trained Spacy English Word2Vec.
    - The results of applying the model were still inconclusive. Compared with positive and negative samples, they all matched at higher rates.
- Custom Word2Vec Embeddings
    - We thought of using Gensim Word2Vec Deep Learning Algorithm training to own Job Descriptions embeddings.
    - We can use matching against the same word in the resume if we have a model that understands the semantic of the occurrence of the word in the job description.
    - Hence now we have each Job Description represented as a list of word vectors with 300 dimensions for each word. In order to compute the similarities, we required to get a single vector representation for each Job Description.
    - Taking the average to get a single vector representation
        - We used this method to take the average of all the word vectors for each of the 300 dimensions, so we ended up with one vector for a Job Description with 300 dimensions.

Step 4: Training Word2Vec Embeddings

We performed many iterations adjusting preprocessing steps and training parameters to train the embeddings over the data for job descriptions.

The steps involved were as follows:

- Pre-processing Data
- Train Model
- Test Model (Visualize Models)
- Save Model

Training Parameters

Parameter	Finalized	Iterations
PreProcessing Step	All included	* Remove email * Remove web addresses * Remove tokens which are digits * Remove website addresses "www." * Remove all non-ascii characters e.g ,äôs * Replace punctuations with space * only unigrams with 2 or more characters are taken * Check if the word is made up of English letters and is not alpha-numeric * strip punctuations and leading/trailing spaces from unigram * Remove alphanumeric tokens like gd17, 11693b3
min_count	1	{ 5, 2, 1}
window	5	{ 5, 10}
size	300	{ 100, 150, 300}

Model Performance / Evaluation Step

```
print("Top 5 words similar to word 'engineer'")
wv_model.wv.most_similar('engineer', topn=5)
```

```
Top 5 words similar to word 'engineer'
[('developer', 0.7451989650726318),
 ('analyst', 0.6401157975196838),
 ('designer', 0.6314972043037415),
 ('architect', 0.6287281513214111),
 ('engineering', 0.6092355251312256)]
```

```
print("Top 5 words similar to word 'intern'")
wv_model.wv.most_similar('intern', topn=5)
```

```
Top 5 words similar to word 'intern'
[('experienced', 0.612639307975769),
 ('internship', 0.5955079197883606),
 ('expect', 0.5773656964302063),
 ('summer', 0.5754691958427429),
 ('senior', 0.5600848197937012)]
```

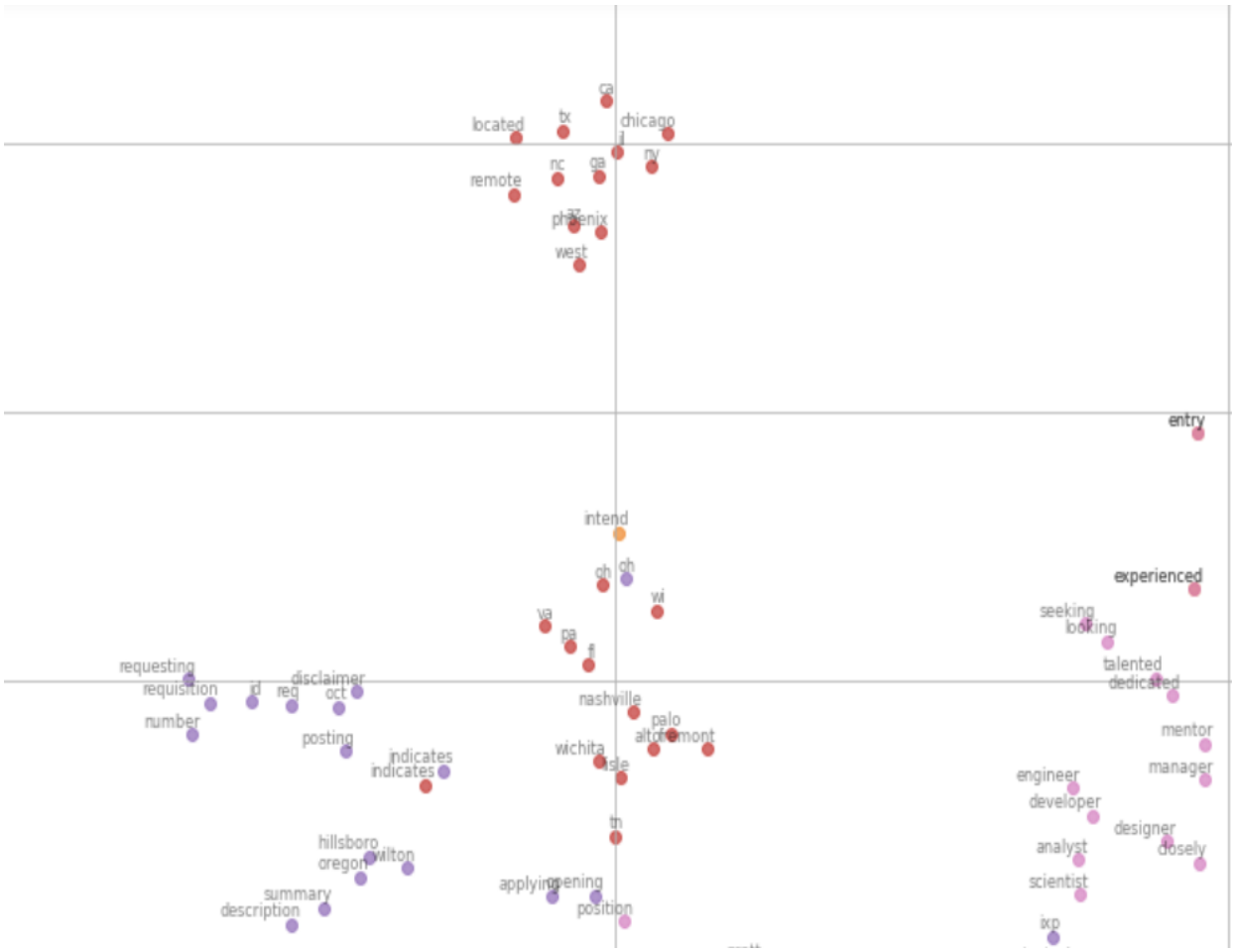
```
print("Similarity between 'manager' and 'engineer':")
wv_model.wv.similarity('manager', 'engineer')
```

```
Similarity between 'manager' and 'engineer':
0.4708756
```

```
print("Similarity between 'scientist' and 'engineer':")
wv_model.wv.similarity('scientist', 'engineer')
```

```
Similarity between 'scientist' and 'engineer':
0.5966589
```

Visualizing embeddings in 2 dimension space



Above figure is a 2 dimensions plot of the words that are similar in resume and job description. The model was successfully able to cluster the common words together. Which means we were able to train the model on our scrapped data and was able to learn the semantic and syntatic similarities.

Step 5: Profile Matcher

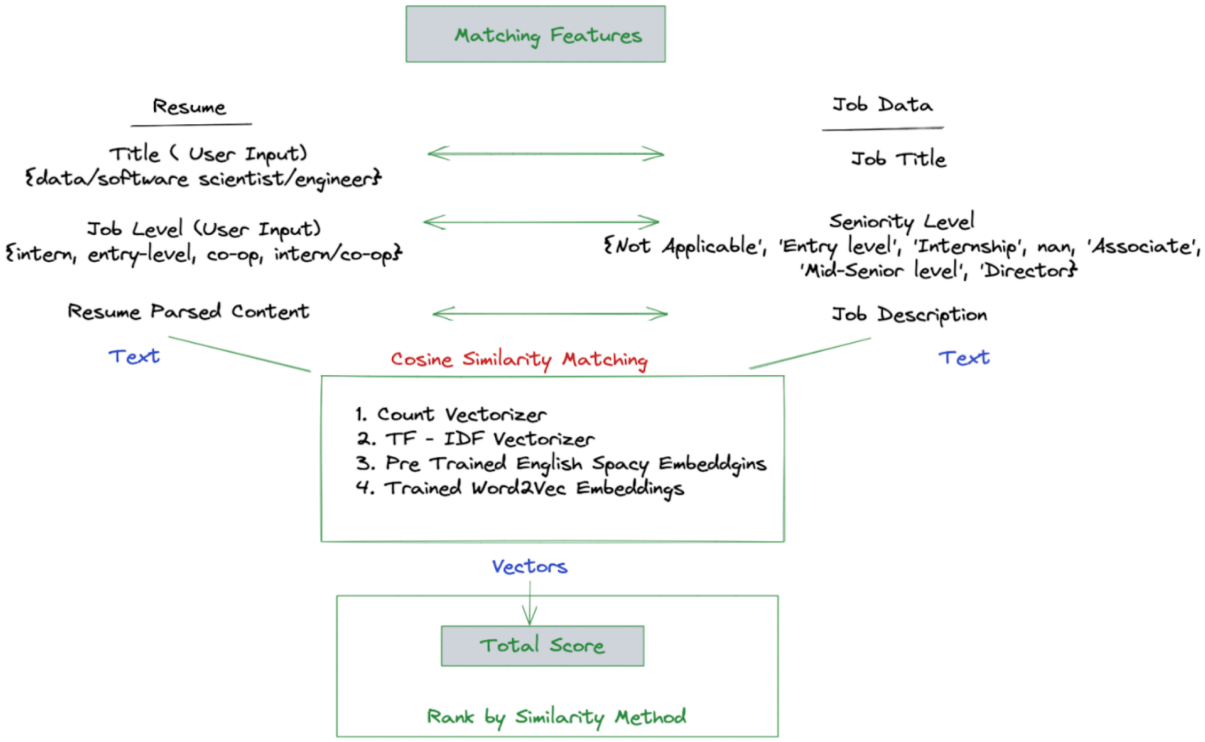
- We now have trained the word2vec model with **vocabulary of 14436 words**
- To put things together, the user inputs his/her resume and test_job data in the same format as LinkedIn and the profile matcher will rank these jobs.
- To rank we follow the below steps:
    - We now have a single vector representation of each document (job and resume) using all vectorization methods.
    - Now that we have the Job description vectors and Resume vector we have used cosine similarity to compute the distance against each method.
    - For each Job Description, the cosine distance from the resume was computed and stored in a list. Hence, now we can infer how close the Job description is from a particular resume.

Initial Results

	CV	TF	TF-IDF	PreTrained-Embeddings	Title
0	48.69	48.69	35.58	85.79	Security Lead - Prime Air
1	53.18	53.18	38.7	80.16	Learning and Development Business Partner, Tech
2	56.03	56.03	42.69	85.72	X-Ray Technologist
3	45.19	45.19	30.94	79.82	AR/VR Systems Frameworks Engineer
4	57.36	57.36	43.92	87.99	Data Engineer, Infrastructure Strategy
5	62.12	62.12	48.86	81.83	Data Engineer, Analytics (Domain Specialist)
6	44.21	44.21	29.76	90.69	Transportation Planner
7	55.83	55.83	42.52	85.34	X-Ray Technologist
8	47.25	47.25	34.18	78.31	Sourcing Lead - Software Engineering
9	46.92	46.92	31.73	89.72	Line Producer, The Choice
10	30.66	30.66	20.04	87.45	Microsoft Search Consultant (100% Remote)

We first undertook some experiments with the following approaches to get some insights into the words that can be added to improve our cosine similarity and therefore increase the match.

Feature Engineering To Improve Matches / Ranking



Calculation of score

It is not just about looking at the cosine similarity between the resume and the job description when we are giving a final score to a job. We are giving a weight of 100 if a job is for an entry level, internship or co-op. And then we are giving an additional weight of 100 if a particular position is for data science and it is either for a data scientist, a data engineer or a software engineer.

In other words we are giving more weight to data science, data engineer and software engineering internships or co-op or entry-level positions. This makes sure that data science, data engineer and software engineering internships or co-op or entry-level positions that have a high match with a candidate's resume will appear on top of the list compared to those with low matches

Once the jobs are sorted we are returning the sorted list of jobs to the user to consume and apply for the jobs. We are giving users matches over count frequency(CV), term frequency(TF), term frequency inverse document frequency(TF-IDF), pretrained embedding and trained embeddings.

Final results comparison

Total score based on custom trained embeddings

	CV	TF	TF-IDF	PreTrained-Embeddings	CustomTrained-Embeddings	Matched Level	Matched Title	Title	TotalScore
3	27.32	27.32	16.31	89.95	70.01	100	100	Data Science Intern	270.01
4	9.54	9.54	5.33	93.4	47.92	100	100	Information Technology / Software Engineer - 2022 Summer Internship	247.92
5	11.57	11.57	6.36	92.76	65.29	100	0	Design Engineer, Internship 1	165.29
0	36.15	36.15	24.29	83.51	72.87	0	100	Data Engineer, Analytics (Domain Specialist)	172.87
1	9.32	9.32	5.19	86.21	62.76	0	100	Senior Software Engineer	162.76
2	4.8	4.8	2.5	88.13	17.18	0	0	Microbiology Supervisor	17.18

Total score based on pretrained embeddings

	CV	TF	TF-IDF	PreTrained-Embeddings	CustomTrained-Embeddings	Matched Level	Matched Title	Title	TotalScore
4	9.54	9.54	5.33	93.4	47.92	100	100	Information Technology / Software Engineer - 2022 Summer Internship	293.4
3	27.32	27.32	16.31	89.95	70.01	100	100	Data Science Intern	289.95
5	11.57	11.57	6.36	92.76	65.29	100	0	Design Engineer, Internship 1	192.76
1	9.32	9.32	5.19	86.21	62.76	0	100	Senior Software Engineer	186.21
0	36.15	36.15	24.29	83.51	72.87	0	100	Data Engineer, Analytics (Domain Specialist)	183.51
2	4.8	4.8	2.5	88.13	17.18	0	0	Microbiology Supervisor	88.13

table 1

Total score based on TF-IDF

	CV	TF	TF-IDF	PreTrained-Embeddings	CustomTrained-Embeddings	Matched Level	Matched Title	Title	TotalScore
3	27.32	27.32	16.31	89.95	70.01	100	100	Data Science Intern	216.31
4	9.54	9.54	5.33	93.4	47.92	100	100	Information Technology / Software Engineer - 2022 Summer Internship	205.33
5	11.57	11.57	6.36	92.76	65.29	100	0	Design Engineer, Internship 1	106.36
0	36.15	36.15	24.29	83.51	72.87	0	100	Data Engineer, Analytics (Domain Specialist)	124.29
1	9.32	9.32	5.19	86.21	62.76	0	100	Senior Software Engineer	105.19
2	4.8	4.8	2.5	88.13	17.18	0	0	Microbiology Supervisor	2.5

table 2

Total score based on TF

	CV	TF	TF-IDF	PreTrained-Embeddings	CustomTrained-Embeddings	Matched Level	Matched Title	Title	TotalScore
3	27.32	27.32	16.31	89.95	70.01	100	100	Data Science Intern	227.32
4	9.54	9.54	5.33	93.4	47.92	100	100	Information Technology / Software Engineer - 2022 Summer Internship	209.54
5	11.57	11.57	6.36	92.76	65.29	100	0	Design Engineer, Internship 1	111.57
0	36.15	36.15	24.29	83.51	72.87	0	100	Data Engineer, Analytics (Domain Specialist)	136.15
1	9.32	9.32	5.19	86.21	62.76	0	100	Senior Software Engineer	109.32
2	4.8	4.8	2.5	88.13	17.18	0	0	Microbiology Supervisor	4.8

table 3

Total score based on CV

	CV	TF	TF-IDF	PreTrained-Embeddings	CustomTrained-Embeddings	Matched Level	Matched Title	Title	TotalScore
3	27.32	27.32	16.31	89.95	70.01	100	100	Data Science Intern	227.32
4	9.54	9.54	5.33	93.4	47.92	100	100	Information Technology / Software Engineer - 2022 Summer Internship	209.54
5	11.57	11.57	6.36	92.76	65.29	100	0	Design Engineer, Internship 1	111.57
0	36.15	36.15	24.29	83.51	72.87	0	100	Data Engineer, Analytics (Domain Specialist)	136.15
1	9.32	9.32	5.19	86.21	62.76	0	100	Senior Software Engineer	109.32
2	4.8	4.8	2.5	88.13	17.18	0	0	Microbiology Supervisor	4.8

table 4

Conclusion

For analysis purposes, we have tested our Profile matcher on a resume of a computer science graduate student looking for any of data science, data engineer or software engineer internship positions against Data Science Intern, Information Technology /Software Engineer 2022 Summer Internship, Design Engineer Internship 1, Data Engineer Analytics (Domain Specialist), Senior Software Engineer and Microbiology Supervisor job descriptions

As a result, we have tabulated the results of all keyword matching techniques. We can see in table 1, which is a score based on trained embeddings, we have been able to score highly for the position of Data Science Intern as it meets the user's requirement. A system is also able to score low on those jobs that are not in line with the candidate's interests.

Discussion / Future Scope

We scraped the LinkedIn website for this project. The pool can be expanded by scraping websites like eJobs, Bestjobs, Glassdoor, Hipo, Facebook, and CareerBuilder. Our efforts can save students a considerable amount of time by recommending suitable job positions.

As part of our future work and enhancements, we are considering the following:

- Improving Custom Training with more data
    - Using more job data scraped, our model can be further refined.
- Professional title matching (regex-based currently; we can further build title embeddings)
- Skills matching section-wise
    - Match specific sections from resume to Job sections, for example:
        - Resume: skills, courses sections
        - Job: qualifications, preferred skills sections
    - In addition, we could identify sections and construct specific embeddings over each section for matching the same words in the same context.
- Embeddings Phrases (* Bigrams/n-grams)
    - We can further explore embeddings over phrases.
- Taking the vector average for Resume and Job document using TF-IDF
    - Using the TF-IDF scores for each word, multiply its vector by the same value and then take the average. In this way, we would be able to get the weighted average.
- Answer another set of the research question
    - We can extend the scope of the project to answer questions like what are some skills candidates can master/add to their resume in order to meet the company's hiring requirements for their dream job?

References

1. Pradeep Kumar Roy, Sarabjeet Singh Chowdhary, Rocky Bhatia, et al. “A Machine Learning Approach for Automation of a Resume Recommendation System.” *Procedia Computer Science*, Elsevier, 16 Apr. 2020, https://www.sciencedirect.com/science/article/pii/S187705092030750X#!

2. Marin, Georgiana Diana, and Constantin Nilă. “Branding in Social Media. Using Linkedin in Personal Brand Communication: A Study on Communications/Marketing and Recruitment/Human Resources Specialists Perception.” *Social Sciences & Humanities Open*, Elsevier, 10 June 2021, https://www.sciencedirect.com/science/article/pii/S259029112100070X.

3. Saket Maheshwary and Hemant Misra. 2018. Matching Resumes to Jobs via Deep Siamese Network. In *WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France. ACM, New York, NY, USA* 3 Pages. https://doi.org/10.1145/3184558.3186942

4. Alotaibi, Shaha. (2012). A survey of job recommender systems. International Journal of the Physical Sciences. 7. 10.5897/IJPS12.482.

Appendix

Appendix A : List of Python Dependencies / Libraries

- LinkedinScraper (https://pypi.org/project/linkedin-jobs-scraper/)
- Tika Parser (https://tika.apache.org/)
- Scikit (https://scikit-learn.org/)
- Gensim (https://pypi.org/project/gensim/)
- NLTK (https://www.nltk.org/)
- Spacy (https://spacy.io/)
- Wordcloud (https://amueller.github.io/word_cloud/)
- Matplotlib (https://matplotlib.org/)
- Numpy (https://numpy.org/)
- Pandas (https://pandas.pydata.org/)
- Regular Expression - re (https://docs.python.org/3/library/re.html)