Alumni Career Path Analysis

A Case Study of the Alumni of the Computer Science

Department at Mills College

Thesis work to be completed

to meet the requirements for the Degree of

Master of Arts in Interdisciplinary Computer Science

Mills College, June 2019

by

Hao Yu

Approved by

Acknowledgement

I would like to first thank the faculty members of Mills College Mathematic and Computer Science Department: prof. Susan Wang, prof. Ellen Spertus, prof. Almudena Konrad and prof. Barbara Li Santi, who taught and helped me during and after my study at Mills College. Another tremendous thanks to my academic and thesis advisor Susan, for helping me throughout this entire thesis process -- from big to small -- your impressive patience and professionalism has given me enormous confidence and encouragement.

I also extend my thanks to my interdisciplinary advisor prof. Jahan Ghofraniha for his thoughts, suggestions and comments from the starting of the initial proposal to the finish of this thesis.

I could also not have been able to complete this program and project, if it had not been for the endless love and support of my family. Thank you to my mom and dad and my brother, this one's for you.

Abstract

Alumni relations have long been considered key attributes in universities or programs evaluation. However, alumni relations at most universities and programs are narrowed into the evaluation of alumni's characteristics, lifestyles, types of behavior, and interests, as well as their potential in making contributions, features that are not as helpful to professors and students. In this thesis, I introduce a general programmatic methodology using data science and statistical machine learning techniques to be applied to alumni career path analysis through a relational alumni database. This work aims to bring more insights regarding the career paths of alumni, and in particular, the Mills Computer Science alumni and their careers, and sheds light on the outcomes and expectations of having a Mills computer science education.

Table of Contents

List of Figures

List of Tables

Code Snippets

1. Introduction

In most universities and programs, alumni relations are narrowed into the evaluation of alumni's characteristics, lifestyles, types of behavior, and interests, as well as their potential in making contributions, because the result will immediately help an institution to locate the most probable donors and thus economically benefit from them. However, these characteristics are not as helpful to current and prospective students as knowing the types of positions alumni hold and the growth or evolution of their careers, in other words, the knowledge gained after performing alumni career path analysis.

A good alumni career path analysis is necessary and beneficial for many reasons. For an institution, it provides supplemental material for advertising courses or program design. For current students, it gives them a clear understanding of what they can expect after graduation, further strengthens their confidence, and motivates them. For prospective students, it supplies the information needed to assist them in making their academic decisions.

In the case of the Mills College Computer Science Programs (Mills CS), as the population of alumni grows, the traditional manual analysis of alumni data becomes tedious, obsolete and error-prone. This project aims to apply data science and statistical machine learning techniques to the process of analyzing students' career paths before and after graduation and to answer the following questions:

1. What is the percentage of alumni working in technology vs. non-technology?
2. What skills should current students build to get on the path of technical or non-technical roles?

3. What are the career differences between undergraduate and graduate alumni?
4. What are the career differences of alumni before and after Mills CS?

These questions are especially important for Mills CS since it has two graduate programs, the Certificate of Computer Science and the Master of Arts in Interdisciplinary Computer Science, and these two programs are both designed for students with a non-CS background. Thus, looking at the difference between undergraduate and graduate alumni career provide some interesting insights.

2. Background

2.1. AlumDB structure and content

This project is built upon the 2018 version of the Mills CS alumni database (**AlumDB_2018**), which is comprised of several tables, each storing specific information about each alumna/alumnus (alum). The *person* table lists all alumni and associates each alum with a unique identifier ID (**pID**). The *linkedIn_profile* table lists the URL to each alum's LinkedIn profile and their LinkedIn header information along with respective pID. All the other tables are *one-to-many*, that is, one alum could have many different records in each table. Each of these tables includes all records available for all alumni, and each record is uniquely associated with one alum by their pID.

Table Name
person
education

experience
linkedIn_profile
name_table
skill

Table 1: AlumDB structures

The career path analysis mainly focused on the *person*, *education*, *experience* and *skill* tables. The *education* table lists education-related history from alumni's LinkedIn profiles. Each record in this table must include the alum's pID, a school name, a major, a degree, and the start and end years of the program. The *experience* table lists job experience history for all alumni. Each record in this table includes the alum's ID, a position title, a company name, and the start and end dates of their positions. The *skill* table lists all skills and number of endorsements for all alumni.

AlumDB_2018 *person* table contains 634 unique alumni starting from 1963 to 2018, while *education* table contains 821 records, *experience* table contains 2283 records and *skills* table contains 6169 records of 359 alumni, in other words, there are 634 - 359 = 275 alumni whose information was not available and hence they do not have records in *education*, *experience*, and *skill* tables.

2.2. AlumDB_2018 resources

The data content of AlumDB_2018 was merged from two resources: *person* table data was obtained from Mills College record office; *education, experience, linkedin_profile* and *skill* table data was extracted from alumni's LinkedIn profiles.

3. Related Work

While much of research on alumni analysis has been focused on alumni networks and alumni giving, in this section, I will be discussing research regarding alumni career path analysis, and some internal research at Mills College, more specifically, research related to Mills CS AlumDB.

3.1. Alumni career path analysis

Research on alumni career paths is often motivated by evolving training tracks, updating and designing curriculum, and alumni assessment [4][18]. Rudolf et al. [4] surveyed 27 alumni through email to explore the impact of fellowship training on their careers so as to continue to develop informatics fellowships to meet the future needs of trainees. Martin [6] analyzed survey data of 78 alumni to assess what percentage of senior IAC[1] alumni had addressed energy concerns over their careers. In their research, alumni jobs were categorized based on their relevance to application of energy efficiency knowledge gained in IAC. Case et al. [2], unlike the above researchers, focused on data collection and validation using LinkedIn resources instead of contacting each alum. Li. et al. [16] introduced an alumni database architecture that extracted data from 600 LinkedIn profiles of alumni who have graduated with BS and MS degree from an IT program and found them in general to be moving towards managerial positions as their career progressed. Massoni et al. [17] introduced a methodology inspired from biology and

[1] IAC: U.S. Department of Energy's Industrial Assessment Center

adapted it to career path analysis, combining optimal matching and self-organizing maps, a type of Artificial Neural Network. Their work used the “Generation 98” dataset from CEREQ, France (http://www.cereq.fr/) and concluded 8 career-path typologies of school-to-work transition. Despite all these research, a comprehensive methodology that takes advantage of modern statistical machine learning and data science techniques and also answers questions asked in the introduction section is missing.

3.2. Mills CS AlumDB related research

Over the past several years at Mills College, two computer science graduate students have focused on AlumDB projects for their Master thesis. Zaida Mejia [3] worked towards creating an AlumDB through the use of the LinkedIn alumni API, as well as a web view to explore the database, though the final database is now implemented differently than her proposal. Jennifer Diaz [5] focused her study on the security aspect of the web view of AlumDB. In her thesis, she introduced the general methodology of web security analysis. With my thesis, I will be adding one more block to the entire AlumDB mission.

4. Methodology

This project is developed in Python 3 as Python is increasingly popular in data science and machine learning.

4.1. Defining some keywords

TECH - Label keyword, used to annotate experiences and occupations that are in Tech.

NONTECH - Label keyword, used to annotate experiences and occupations that are not in Tech.

Moreover, since this project targets Mills CS alumni, a group of people who all have computer science education, ‘TECH’ and ‘NONTECH’ have been narrowed down to computer science or engineering related roles and experiences rather than all typical technical and non-technical positions.

Tech - In this project, an alum is considered in Tech if they have at least one experience being considered as ‘TECH’.

Non-tech - In this project, an alum is considered in Non-tech if they have no experience in Tech, i.e., all their experiences in the experience table are considered as ‘NONTECH’.

Career Path - Career Path sometimes refers to the growth of the employee in an organization, more specifically, it means the various positions an employee moves on one by one as s/he grows in an organization [10]. Another definition is that Career paths are routes that individuals take from their first foray into the job market to their final position before retirement [8]. However, due to the data deficiency of AlumDB_2018, I defined career path as simply all jobs that one alum has ever worked on and their changes between Tech and Non-tech fields.

4.2. Pulling data from AlumDB

The aforementioned AlumDB_2018 database is a MySQL database stored in the Linux server of the Mills CS; one must be connected to the Mills College network in order to access it. In this project, MySQL Connector, a Python package provided by Oracle, is used to access AlumDB_2018 with admin authentication. Each relevant table is pulled out by querying AlumDB and then transformed into a Pandas dataframe labeled person, edu, exp and skill.

```
# load mysql
import mysql.connector
import pandas as pd
alumDB = mysql.connector.connect(
   host="localhost",
   user="root",
   passwd="password"
)
person = pd.read_sql_query("SELECT * FROM alumDB_2018.person;", alumDB)
edu = pd.read_sql_query("SELECT * FROM alumDB_2018.education;", alumDB)
exp = pd.read_sql_query("SELECT * FROM alumDB_2018.experience;", alumDB)
skill = pd.read_sql_query("SELECT * FROM alumDB_2018.skill;", alumDB)
```

Code Snippet 1: Loading relevant tables into pandas dataframe

4.3. Data cleaning and tuning

Data cleaning and tuning involved four steps:

Cleaning up missing values

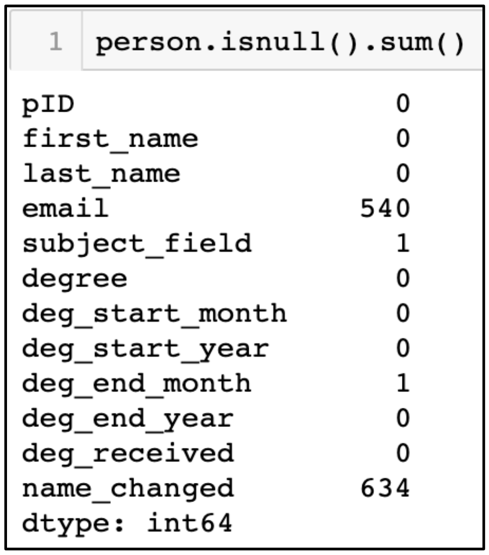
Splitting data into undergraduate and graduate

Data exploration

Classifying jobs into Tech and Non-tech

4.3.1. Cleaning up missing values

Identifying missing values and dealing with them is a vital part of data cleaning. Numerical values in Figure 1 to Figure 4 provide the column-wise total number of null observations.



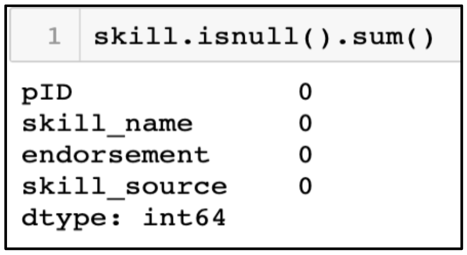
```
person.isnull().sum()
```

```
pID                 0
first_name          0
last_name           0
email             540
subject_field       1
degree              0
deg_start_month     0
deg_start_year      0
deg_end_month       1
deg_end_year        0
deg_received        0
name_changed      634
dtype: int64
```

Figure 1: Number of nulls in *person* table

For 'email' and 'name_changed' missing values, those are left as they are, since they are not relevant to this project.

For 'subject_field' and 'deg_end_month' columns, they are only used to fill in missing values in *education* table, thus, no action is needed if missing values in *person* do not overlap with missing values in *education* table. On the other hand, if overlap exists, it is necessary to contact Mills College record office for information in order to fill in these missing values in both *person* and *education* tables.

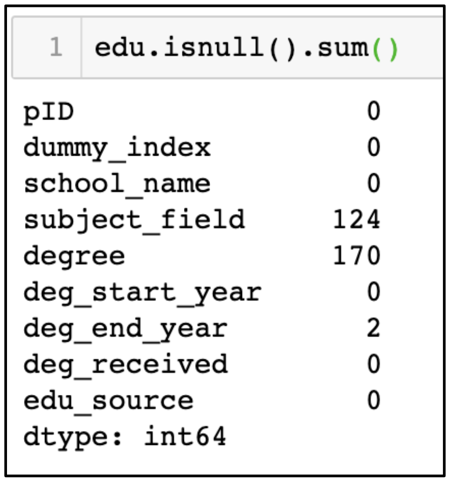


```
skill.isnull().sum()
```

```
pID             0
skill_name      0
endorsement     0
skill_source    0
dtype: int64
```

Figure 2: Number of nulls in *skill* table

Skill has no missing values. This is reasonable since LinkedIn does not accept skill endorsements with no name.



```
edu.isnull().sum()
```

```
pID                 0
dummy_index         0
school_name         0
subject_field     124
degree            170
deg_start_year      0
deg_end_year        2
deg_received        0
edu_source          0
dtype: int64
```

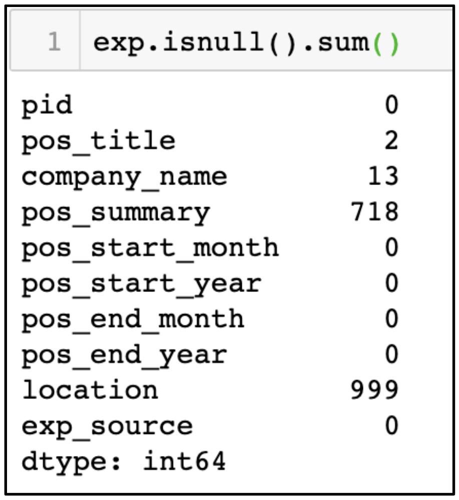
Figure 3: Number of nulls in *education* table

In the *education* table, I divided missing values into two categories:

1. Mills CS education
2. non-Mills CS education

For Mills CS education, missing values were filled by data from the *person* table and fortunately, missing values in *education* was successfully filled by data from *person* table, since these missing values did not overlap with missing value in *person* table.

For non-Mills CS education: numerical missing values were filled with 0 and non-numerical missing value filled with '0'.



```
exp.isnull().sum()
```

```
pid                  0
pos_title            2
company_name        13
pos_summary        718
pos_start_month      0
pos_start_year       0
pos_end_month        0
pos_end_year         0
location           999
exp_source           0
dtype: int64
```

Figure 4: Number of nulls in *experience* table

For company_name missing values, those were filled with 'Unknown, Co. Ltd.'

For pos_summary missing values, those are filled with ‘ ’.

For location missing values, they are left as is.

4.3.2. Splitting data into undergraduate and graduate

After handling missing values, the next step is to slice out undergraduate and graduate student information. Multiple programs of different levels are offered by the Mills CS. Undergraduate programs consist of a major and a minor in Computer Science and Data Science, and graduate programs consist of a Certificate in Computer Science and a Master of Arts in Interdisciplinary Computer Science. Mills CS accepts only people without a CS degree into graduate programs, hence there exists no alum who has received both undergraduate and graduate degrees from Mills CS. This makes it nontrivial and interesting to split the data to undergraduate and graduate portions and prepare it for further analysis.

In AlumDB_2018, the two graduate programs are stored by their acronyms, ‘MA’ for Master of Arts in Interdisciplinary Computer Science, and ‘CSCrt’ for Certificate in

Computer Science. Programs that are labeled other than 'MA' or 'CSCrt' are all for Mills CS undergraduates.

```
grad = person[(person.degree == 'MA') | (person.degree == 'CSCrt')].pid.values
undergrad = list(set(person.pid) - set(grad))

gradExp = exp[exp.pid.isin(grad)]
underExp = exp[~exp.pid.isin(grad)]

gradEdu = edu[edu.pID.isin(grad)]
underEdu = edu[~edu.pID.isin(grad)]

gradSki = skill[skill.pID.isin(grad)]
underSki = skill[~skill.pID.isin(grad)]
```

Code Snippet 2: Separating undergraduate and graduate alumni

4.3.3. Preliminary data exploration

Prior to diving into alumni data and answering those ultimate questions asked in the introduction section, some preliminary data exploration would be beneficial in terms of gaining a better understanding of the data which in turn provide information regarding Mills students and alumni.

1) Person: Yearly student enrollments and total student population

As can be seen from Figure 5 below, the very first Mills College alum who received Mills CS Bachelor of Arts in Computer Science joined Mills in 1972. The first group of Mills CS graduate students joined Mills in 1982. In 2015, Mills CS had the largest number of new students, which is 29.

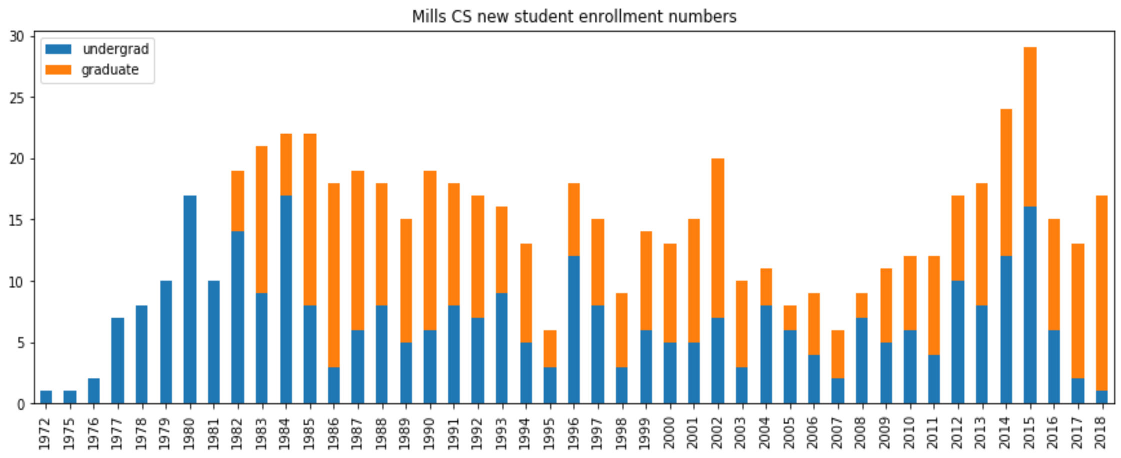


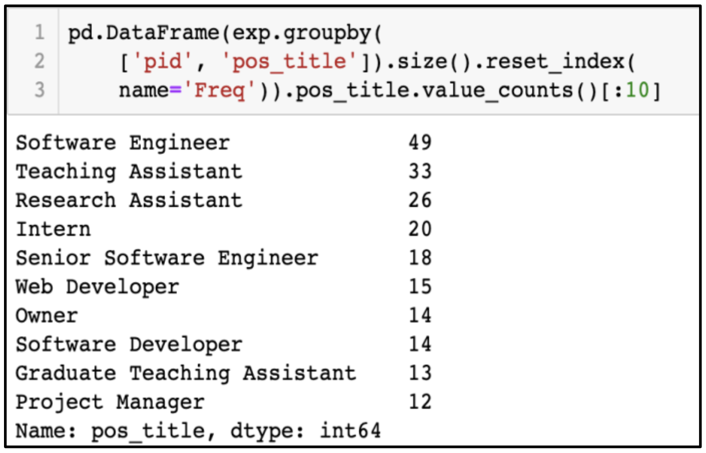
Figure 5: Mills CS yearly new student enrollment distribution

From 1972 to 2018, Mills CS has a total student population of **634**, consisting of **317** undergraduate students and **317** graduate students.

2) Experience: Common job titles

The experience table contains career information of 359 alumni, which stacked up to 2283 experience records. Finding common job titles within this data reveals some career patterns.

- overall common titles

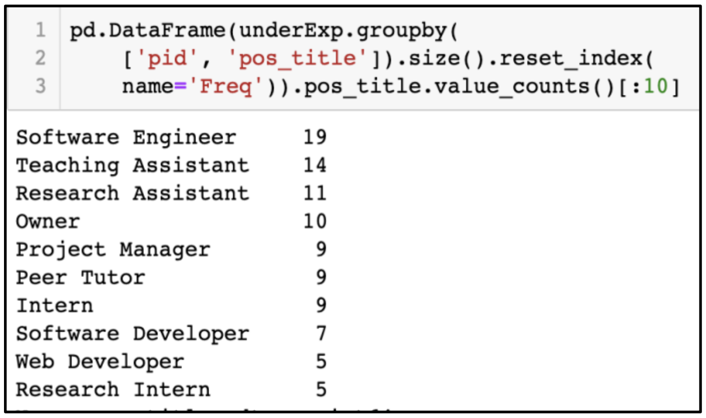


```
pd.DataFrame(exp.groupby(
    ['pid', 'pos_title']).size().reset_index(
    name='Freq')).pos_title.value_counts()[:10]
```

```
Software Engineer              49
Teaching Assistant             33
Research Assistant             26
Intern                         20
Senior Software Engineer       18
Web Developer                  15
Owner                          14
Software Developer             14
Graduate Teaching Assistant    13
Project Manager                12
Name: pos_title, dtype: int64
```

Figure 6: Top 10 common positions of all alumni

- common titles for undergraduates and graduates[2]

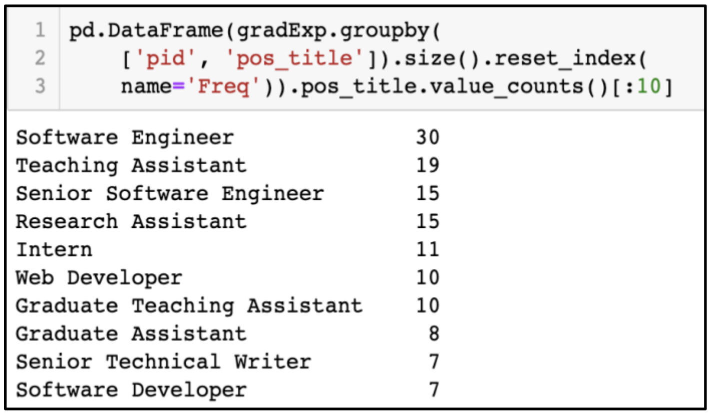


```
pd.DataFrame(underExp.groupby(
    ['pid', 'pos_title']).size().reset_index(
    name='Freq')).pos_title.value_counts()[:10]
```

```
Software Engineer     19
Teaching Assistant    14
Research Assistant    11
Owner                 10
Project Manager        9
Peer Tutor             9
Intern                 9
Software Developer     7
Web Developer          5
Research Intern        5
```

Figure 7: Top 10 common positions of undergraduate alumni

[2] Note that undergraduates and graduates represent degrees students received from Mills CS; they do not infer students' highest level of education.



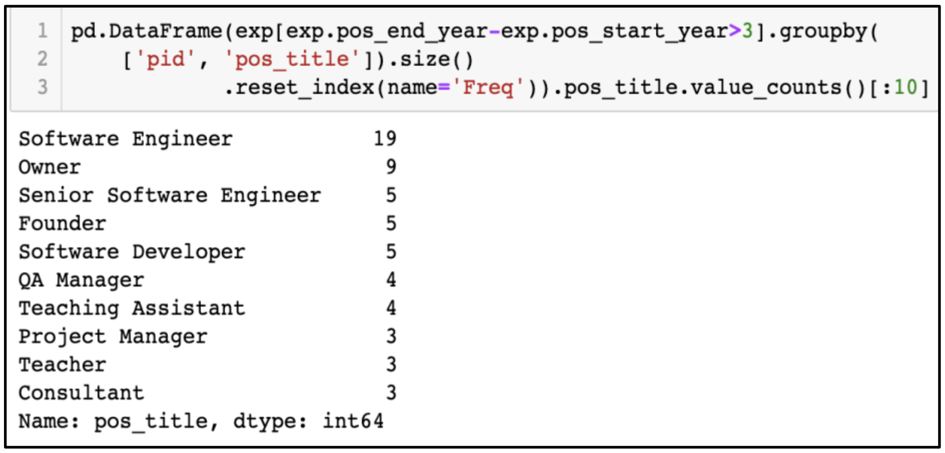
```
pd.DataFrame(gradExp.groupby(
    ['pid', 'pos_title']).size().reset_index(
    name='Freq')).pos_title.value_counts()[:10]
```

```
Software Engineer              30
Teaching Assistant             19
Senior Software Engineer       15
Research Assistant             15
Intern                         11
Web Developer                  10
Graduate Teaching Assistant    10
Graduate Assistant              8
Senior Technical Writer         7
Software Developer              7
```

Figure 8: Top 10 common positions of graduate alumni

Note that the most common job title among all alumni is Software Engineer, and there is no significant variance between undergraduate and graduate alumni job titles.

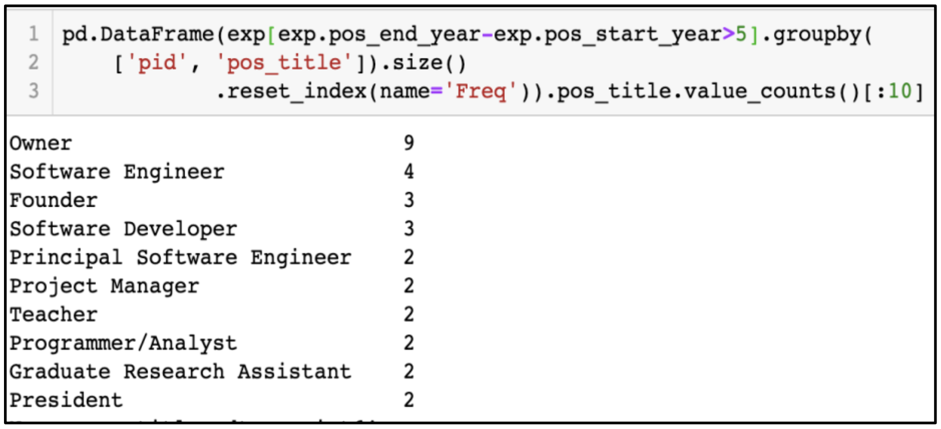
- common titles held by alumni for more than 3 years, 5 years, 10 year or 20 years.



```
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>3].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()[:10]
```

```
Software Engineer           19
Owner                        9
Senior Software Engineer     5
Founder                      5
Software Developer           5
QA Manager                   4
Teaching Assistant           4
Project Manager              3
Teacher                      3
Consultant                   3
Name: pos_title, dtype: int64
```

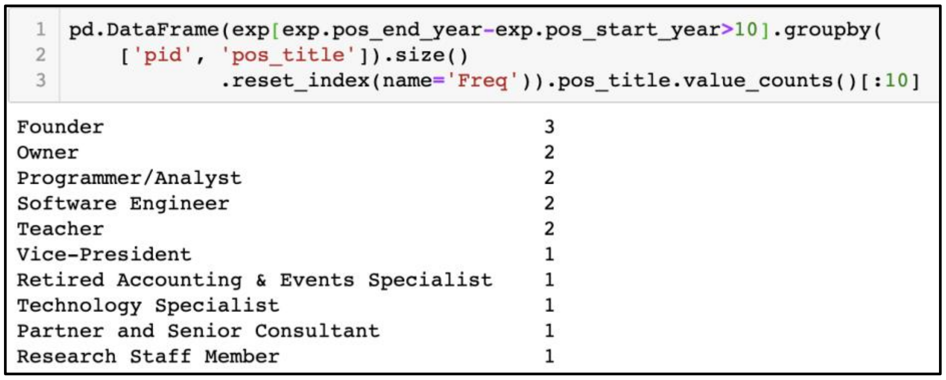
Figure 9: Top 10 common alumni positions held for more than 3 years



```
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>5].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()[:10]
```

```
Owner                             9
Software Engineer                 4
Founder                           3
Software Developer                3
Principal Software Engineer       2
Project Manager                   2
Teacher                           2
Programmer/Analyst                2
Graduate Research Assistant       2
President                         2
```

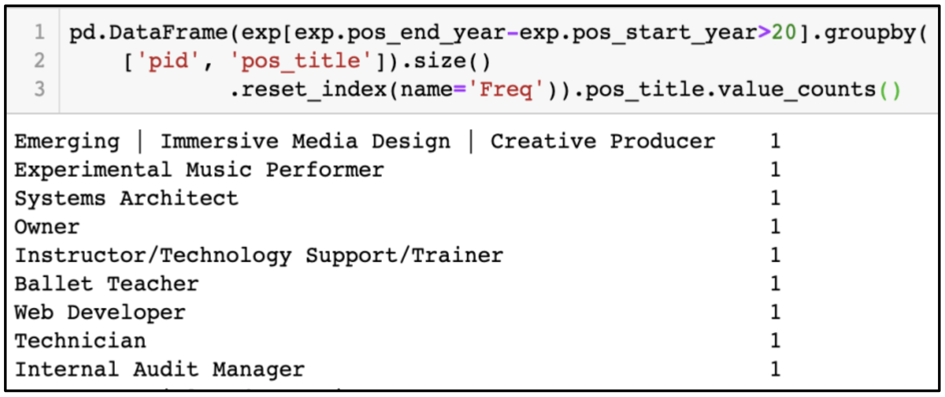
Figure 10: Top 10 common alumni positions held for more than 5 years



```
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>10].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()[:10]
```

```
Founder                                     3
Owner                                       2
Programmer/Analyst                          2
Software Engineer                           2
Teacher                                     2
Vice-President                              1
Retired Accounting & Events Specialist      1
Technology Specialist                       1
Partner and Senior Consultant               1
Research Staff Member                       1
```

Figure 11: Top 10 common alumni positions held for more than 10 years



```
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>20].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()
```

```
Emerging | Immersive Media Design | Creative Producer    1
Experimental Music Performer                             1
Systems Architect                                        1
Owner                                                    1
Instructor/Technology Support/Trainer                    1
Ballet Teacher                                           1
Web Developer                                            1
Technician                                               1
Internal Audit Manager                                   1
```

Figure 12: Common alumni positions held for more than 20 years

It is clear that while the number of years holding the same position goes up, the number of alumni goes down, with only a few alumni holding one position for as long as ten or twenty years. This is reasonable considering the age of Mills CS and the progression of career transitions. As for experiences in the 3-10 years range, Software Engineer still plays a significant role.

3) Skill: Common skills

Skills of each alum were either entered by themselves into their LinkedIn profile or endorsed by other LinkedIn members. In this section, we look at skills in general, as well as the difference between skills with and without endorsement for undergraduate and graduate alumni.

- Top 10 common skills of all alumni

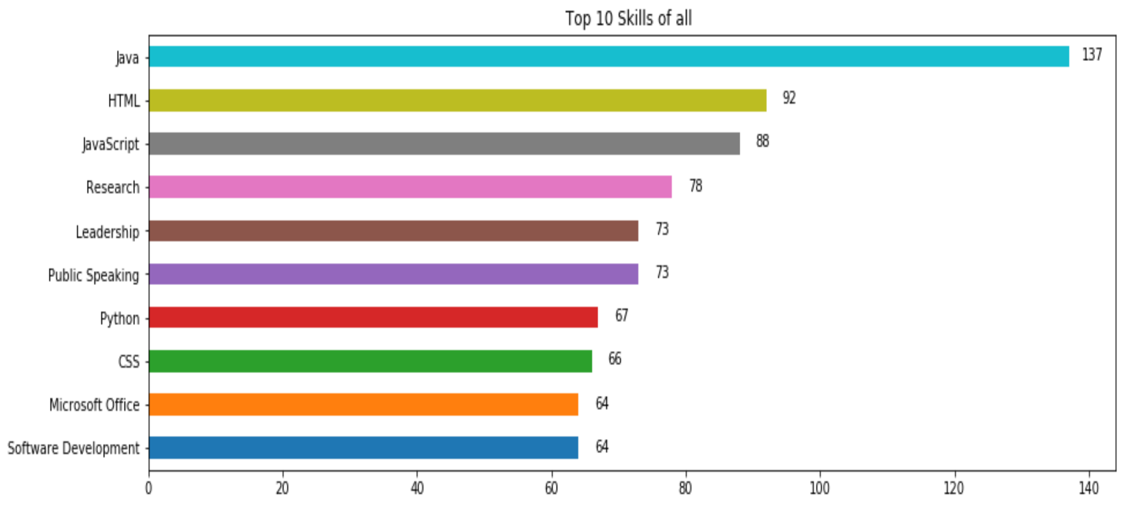


Figure 13: Top 10 common alumni skills

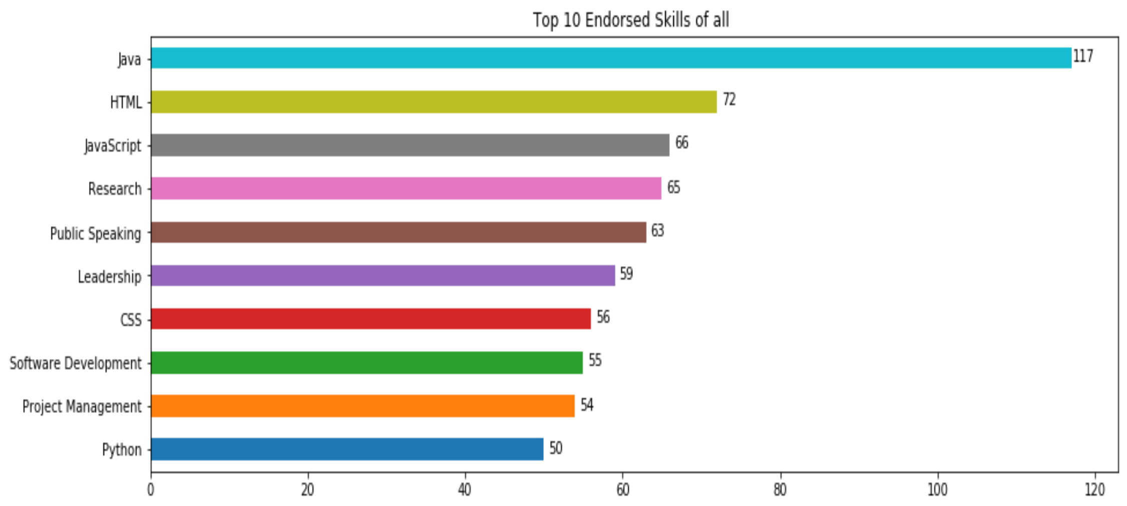


Figure 14: Top 10 common endorsed alumni skills

The top 10 common skills of all alumni are Java, HTML, Javascript, Research, Leadership, Public Speaking, Python, CSS, Microsoft Office, and Software Development. Common endorsed skills have a slightly different order: Microsoft Office is replaced by Project Management.

- Top 10 common skills of undergraduate alumni

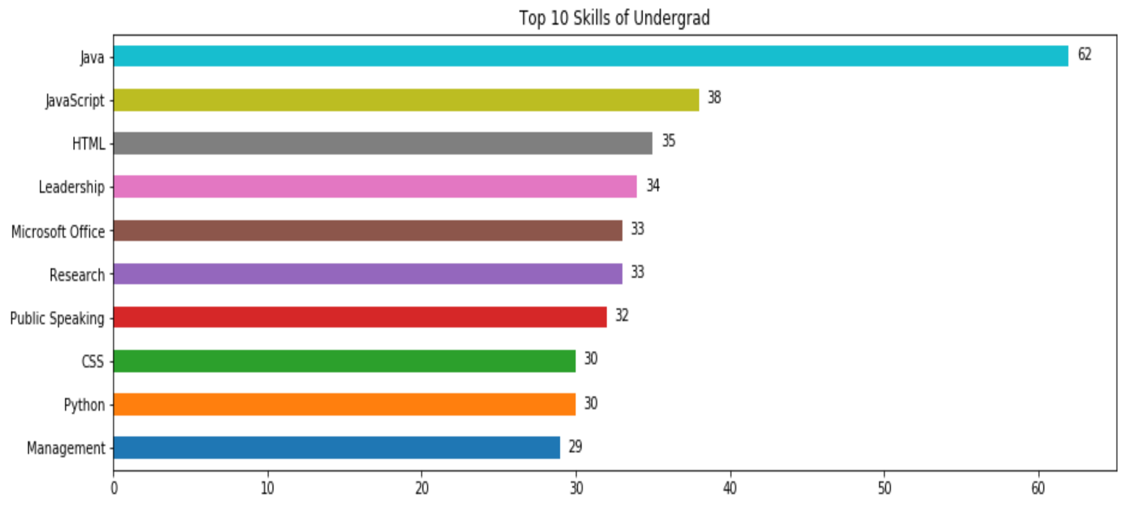


Figure 15: Top 10 common undergraduate alumni skills

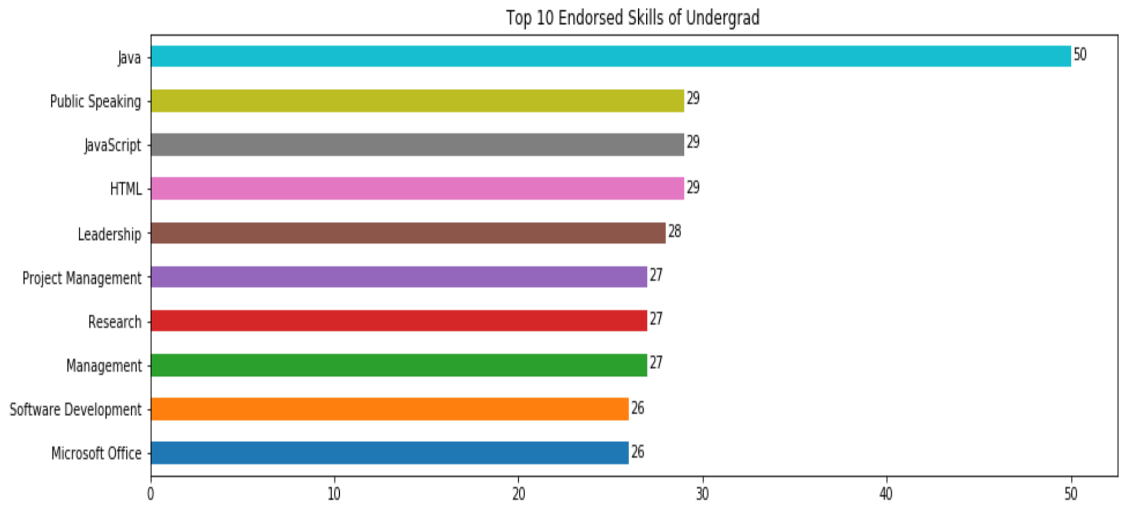


Figure 16: Top 10 common endorsed undergraduate alumni skills

The top 10 common skills of undergraduates are Java, JavaScript, HTML, Leadership, Microsoft Office, Research, Public Speaking, CSS, Python and Management. This list is very similar to the list of the overall top skills, except Software Development is replaced by Management. As for top 10 endorsed skills of undergraduates, there are eight skills in common, with CSS and Python replaced by Project Management and Software Development.

- Top 10 common skills of graduate alumni

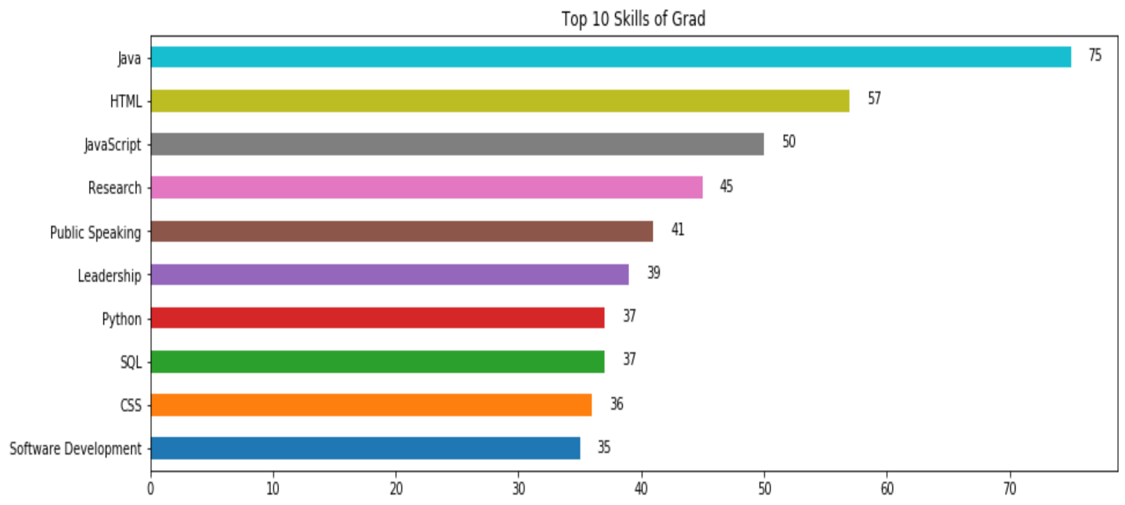


Figure 17: Top 10 common graduate alumni skills

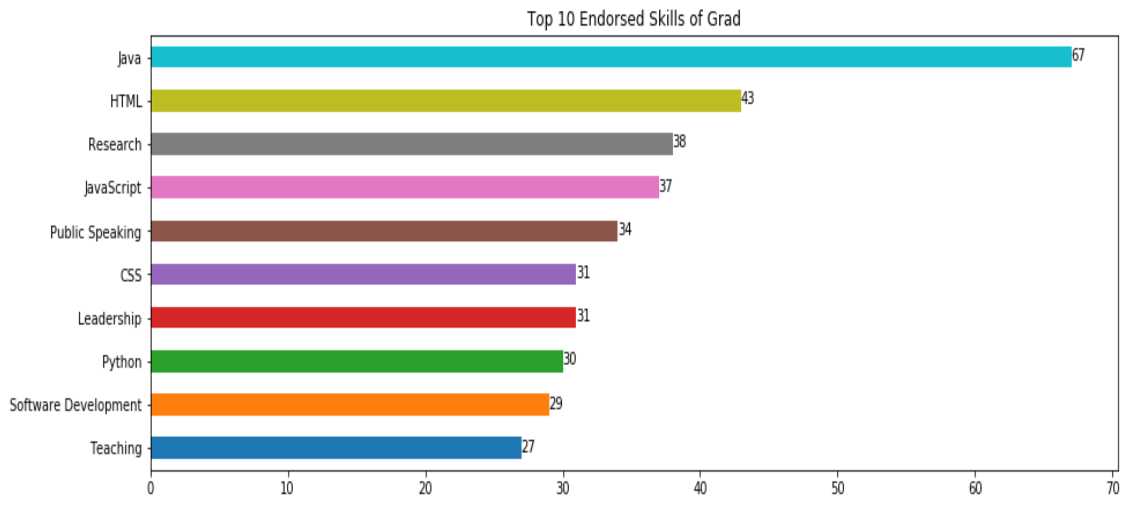


Figure 18: Top 10 common endorsed graduate alumni skills

The top 10 common skills of graduate alumni include skills other than aforementioned common skills. SQL is the 8th common skill of graduate alumni and teaching is the10th endorsed skill.

It is easily noticeable from the above six skill graphs that the top skill is always Java. This is reasonable since Java is the primary programming language taught at Mills CS. Comparison between skills with and without endorsement shows little difference.

4.3.4. Classifying jobs into TECH and NONTECH.

This section introduces a supervised machine learning approach in order to classify alumni job titles into 'TECH' and 'NONTECH'. Supervised Machine Learning is the task of learning a function that maps an input to an output based on example input-output pairs [13]. Furthermore, it classifies alumni into a Tech category if they had any experience in 'TECH' and Non-tech category if all their experiences were classified as 'NONTECH'.

1) Model selection

   a) Naive Bayes (NB) vs. Others

In this project, a supervised machine learning model called Multinomial Naive Bayes (Multinomial NB) is chosen over other alternatives. This classification task is indeed a typical Natural Language Processing (NLP) problem since it is mainly dealing with job titles and job descriptions and these are all textual data. While coping with NLP problems, there are other alternatives such as Support Vector Machine (SVM), Artificial Neural Networks (ANN) and many more. However, the simple design of NB classifiers makes it attractive for the problem we face, especially since the amount of data we have is small. Moreover, NB has been demonstrated to be fast, reliable and accurate in a number of applications of NLP [11].

b) Simple NB vs. Multinomial NB

There are a number of different NB models. The major consideration in choosing the appropriate NB for this project was between simple NB and Multinomial NB. Simple NB refers to the strong independence assumptions in the model [12], i.e., it assumes every input feature is independent from each other and models the combination of all input features by checking the presence and absence of particular words and sums up the probability of the presences in order to determine the most probable category. Multinomial NB, as described by Manning et al. (2008) [9], estimates the conditional probability of a particular word/term/token as the relative frequency of term *t* in documents belonging to class *c*.

$$\hat{P}(t|c) = \frac{T_{ct}}{\sum_{t' \in V} T_{ct'}},$$

More specifically, Multinomial NB does not only simply look at whether a particular word exists in the example; it explicitly models the word count and adjusts the underlying calculations. In this project, Multinomial NB is eventually selected to make the best use of the content of *experience* table. In the job classification phase, the input data used in classification is the *pos_title* and *pos_summary* columns in *experience* table. They stand for ‘position title’ and ‘position summary’ and are both strings. Multinomial NB looks through the word groups of all TECH and NONTECH occupations and descriptions from the training dataset that are introduced in the next subsection and finds the pattern of each group. It then classifies each alumni experience into the most probable category based on how close words from alumni *pos_title* and *pos_summary* are to the patterns found.

2) Training data preparation

a) Training data source

As mentioned in the previous section, Multinomial NB is a supervised learning model, which means it needs labeled input. However, the alumni data is unlabeled, which is why another dataset, the 2018 version of Standard Occupational Classification (SOC2018) was brought in.

SOC2018 is a federal statistical standard used by federal agencies to classify workers into occupational categories for the purpose of collecting, calculating, or disseminating data. It is provided by the US Bureau of Labor Statistics and it contains 1447 job titles along with descriptions, as shown in Figure 19, and these job titles are classified into 23 major groups.

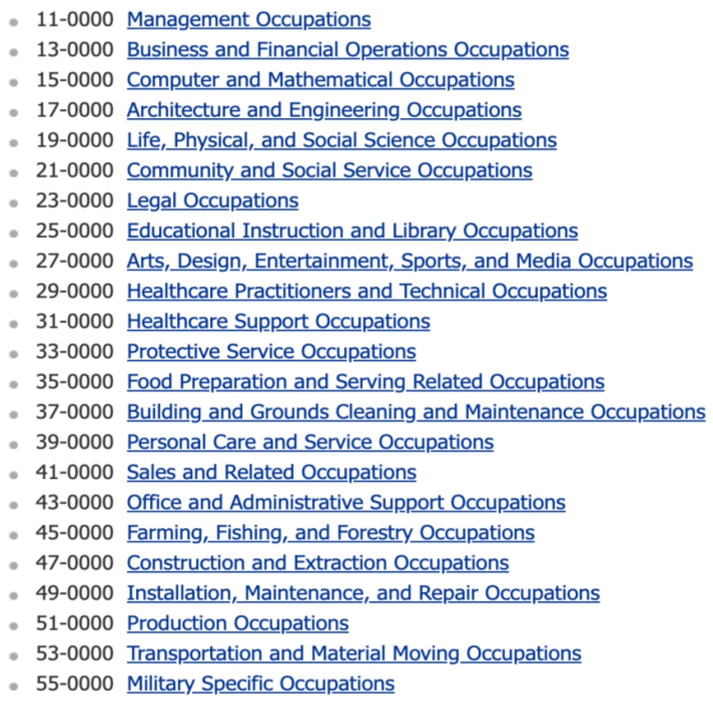

- 11-0000 Management Occupations
- 13-0000 Business and Financial Operations Occupations
- 15-0000 Computer and Mathematical Occupations
- 17-0000 Architecture and Engineering Occupations
- 19-0000 Life, Physical, and Social Science Occupations
- 21-0000 Community and Social Service Occupations
- 23-0000 Legal Occupations
- 25-0000 Educational Instruction and Library Occupations
- 27-0000 Arts, Design, Entertainment, Sports, and Media Occupations
- 29-0000 Healthcare Practitioners and Technical Occupations
- 31-0000 Healthcare Support Occupations
- 33-0000 Protective Service Occupations
- 35-0000 Food Preparation and Serving Related Occupations
- 37-0000 Building and Grounds Cleaning and Maintenance Occupations
- 39-0000 Personal Care and Service Occupations
- 41-0000 Sales and Related Occupations
- 43-0000 Office and Administrative Support Occupations
- 45-0000 Farming, Fishing, and Forestry Occupations
- 47-0000 Construction and Extraction Occupations
- 49-0000 Installation, Maintenance, and Repair Occupations
- 51-0000 Production Occupations
- 53-0000 Transportation and Material Moving Occupations
- 55-0000 Military Specific Occupations

Figure 19: Standard Occupational Classification

b) Labeling training data

While SOC2018 is official and authentic, it comes as unlabeled. However, it is certainly more organized and thus easier to label, compared to our alumni experience data. Since 'TECH' is considered to only indicate occupations that are related to computer science or engineering, the SOC2018 labeling task has been separated to 2 steps. First, skimming through all the 23 major categories. '15-0000 Computer and Mathematical Occupations' and '17-0000 Architecture and Engineering Occupations' are clearly two TECH categories. Second, searching through all other categories for each of these keywords ['Computer', 'engineer', 'engineering', 'data', 'software', 'developer', 'web', 'scientist'], we verify whether each is TECH or not. This step is necessary and important because it helps to find TECH occupations that have been put in other categories. For example, ‘Computer and Information Systems Managers’ and ‘Architectural and Engineering Managers’ have been put under ‘11-0000 Management Occupations’. Additionally, ‘Tutor’ is intentionally labeled as ‘TECH’ since it is one of top-10 common titles Mills CS alumni have during their education at Mills.

This labeling step yields 137 TECH and 1310 NONTECH out of 1447 SOC2018 occupations. In other words, NONTECH : TECH is roughly 10 : 1.

3) Modeling

a) Feature engineering

Feature engineering is a critical step prior to applying Multinomial NB, as well as all other Bayes’ theorem-based algorithms, to NLP problems. It is used to specify which text features to select and feed into the model. One simple text feature is Word Frequency (WF), which counts the occurrences of each word in the document, i.e., training dataset. Another more advanced text feature is Term Frequency - Inverse Document Frequency (TF-IDF), which not only cares about the frequency of each word, but also calculates the weight of rare words in the document. The words that occur rarely in the document have a high IDF score [14].

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \qquad idf(w) = log(\frac{N}{df_t}) \qquad w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

One benefit of choosing TF-IDF over WF is that TF-IDF can rule out the effect of stopwords. An intuitive thought is that word with high frequency in the document carries a higher value, but it is not always true. Words like ‘will’, ‘the’, ‘are’ may appear the most in a document, but are of very little significance. On the other hand, a word that appears less may carry more weight and can really help in classifying data.

In the next section, both text features were tested and eventually TF-IDF was chosen. While WF is obtained using the CountVectorizer function, TF-IDF is done using the TfidfVectorizer function. Both functions are provided by sklearn.

b) Training and testing model

The Multinomial NB model is trained on 80 percent of the SOC2018 dataset and tested with the remaining 20 percent, producing an average 0.93 accuracy and 0.6 F1 score over cross validation for TF-IDF text features, which outperformed WF. The model is ‘sklearn.naive_bayes.MultinomialNB’ and was configured with default settings.

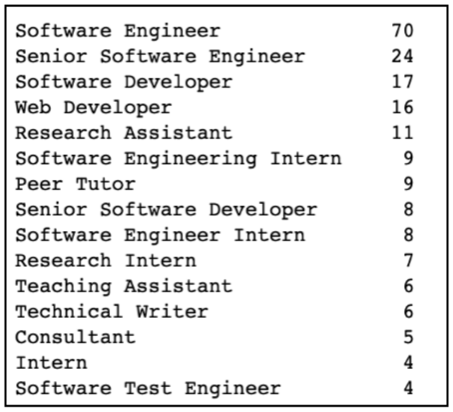
4) Classifying alumni experiences

a) Initial classification

With the trained model, it is plausible to classify alumni experiences, which is done by simply fitting *pos_title* and *pos_summary* for each alum experience into the model. However, some manual adjustments were still found to be needed.

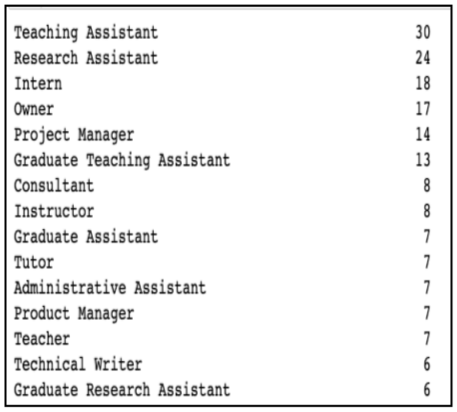
Mistakes were easily noticeable while looking at the top 15 TECH and NONTECH alumni job titles.

TECH



Software Engineer	70
Senior Software Engineer	24
Software Developer	17
Web Developer	16
Research Assistant	11
Software Engineering Intern	9
Peer Tutor	9
Senior Software Developer	8
Software Engineer Intern	8
Research Intern	7
Teaching Assistant	6
Technical Writer	6
Consultant	5
Intern	4
Software Test Engineer	4

NONTECH



Teaching Assistant	30
Research Assistant	24
Intern	18
Owner	17
Project Manager	14
Graduate Teaching Assistant	13
Consultant	8
Instructor	8
Graduate Assistant	7
Tutor	7
Administrative Assistant	7
Product Manager	7
Teacher	7
Technical Writer	6
Graduate Research Assistant	6

Figure 20: Top 15 common jobs classified as TECH and NONTECH

As shown in Figure 20, titles like ‘Project Manager’ and ‘Product Manager’ are most likely in TECH, considering the education background of alumni, and should be classified as such. Though this model took management roles as more similar to the major management category in SOC2018, and therefore, classified these roles as NONTECH. ‘Teaching Assistant’, ‘Research Assistant’ and ‘Consultant’ showed up for both TECH and NONTECH, which seemed confusing, but was a result of the various descriptions written by different alumni for positions with the same names.

This classification classified 781 TECH experiences and 1502 NONTECH experiences as shown in below Figure 21.

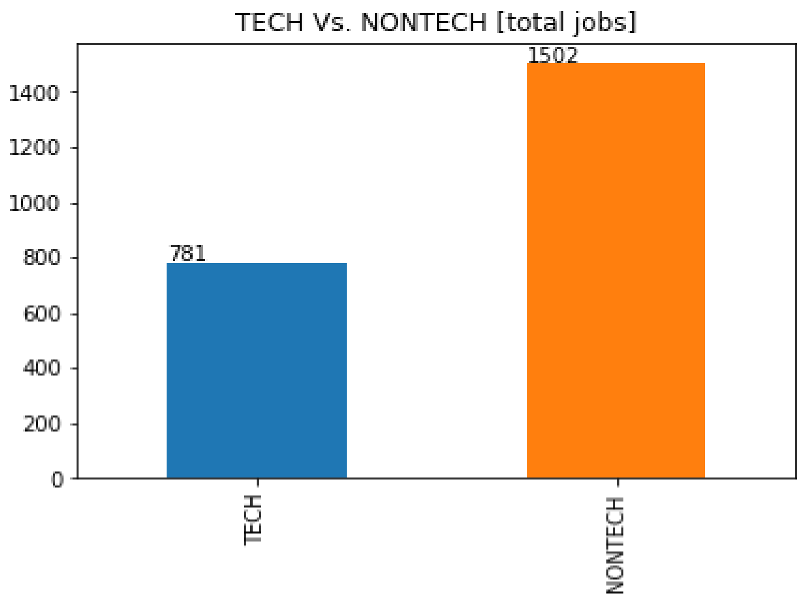


Figure 21: TECH and NONTECH experiences distribution

b) Handling mis-classifications

After noticing this problem, I figured the possible reasons for these mis-classifications are:

(1) Incorrect training data labeling

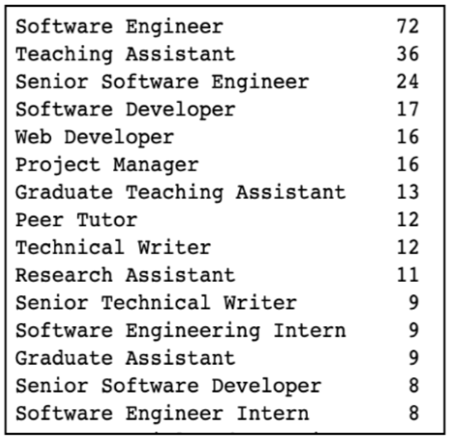
(2) Deficient or incomplete alumni data

(3) Model that was trained on imbalanced data yielding to biased classification

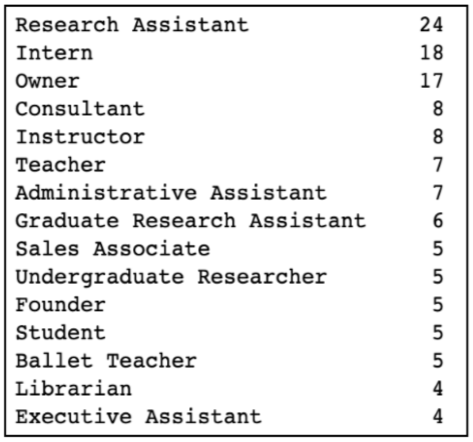
The training data SOC2018 was labeled with careful consideration and we did not have a better solution at this stage. Some alumni experiences have only job titles but with no descriptions which could cause potential mis-classification, since titles are always short and therefore provided less information for the model to make classifications on. This is also not currently solvable since we do not have other resources of alumni data. The Multinomial NB model was trained on the imbalanced SOC2018 dataset, which could possibly yield mis-classifications since the model could classify all experiences into NONTECH and still have good performance score. Solutions tried for this problem were to undersample the majority class, i.e., NONTECH, or oversample the minority class, i.e., TECH. Both strategies were eventually abandoned; undersampling the NONTECH class produced a small number of both classes and oversampling the textual data of TECH class was error-prone and could yield to overfitting, considering that the SOC2018 is not tremendously imbalanced.

After all of the above considerations, the strategy we eventually used to handle these mis-classifications was to manually adjust and correct the classification results. Basically, we skimmed through titles and classification, then extracted the pattern of obvious classification mistakes and flipped the category based on the patterns found, which produced the results below in Figure 22 and Figure 23:

TECH *NONTECH*



```
Software Engineer                 72
Teaching Assistant                36
Senior Software Engineer          24
Software Developer                17
Web Developer                     16
Project Manager                   16
Graduate Teaching Assistant       13
Peer Tutor                        12
Technical Writer                  12
Research Assistant                11
Senior Technical Writer            9
Software Engineering Intern        9
Graduate Assistant                 9
Senior Software Developer          8
Software Engineer Intern           8
```



```
Research Assistant                24
Intern                            18
Owner                             17
Consultant                         8
Instructor                         8
Teacher                            7
Administrative Assistant           7
Graduate Research Assistant        6
Sales Associate                    5
Undergraduate Researcher           5
Founder                            5
Student                            5
Ballet Teacher                     5
Librarian                          4
Executive Assistant                4
```

Figure 22: Top 15 common TECH and NONTECH roles after manual adjusting

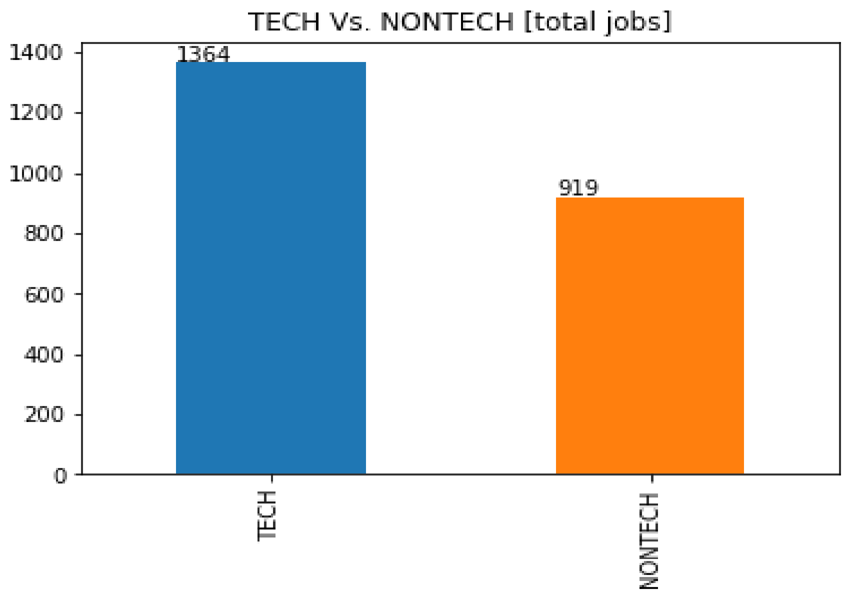


Figure 23: TECH and NONTECH roles distribution after adjusting

This adjusting strategy eventually led to a distribution of 1364 TECH experiences and 919 NONTECH experiences. This is more reasonable compared to the distribution before this adjustment since all these experiences are jobs of Mills CS alumni.

5. Determining Tech and Non-tech alumni

After experiences classification, we now are able to determine the number of Tech and Non-tech alumni. Experience table shows the total number of alumni is 359. The size of *gradExp* and *underExp* pandas dataframes generated in Section 4.3.2 reveal that there are 175 undergraduate alumni and 184 graduate alumni in the experience table (as shown in Table 2). Based on the experience classification results, the number of TECH pids and NONTECH pids are 320 and 301, i.e., 320 alumni have TECH roles and 301 alumni have NONTECH roles. These add up to 621 and is greater than the total number of alumni -- 359, which indicates that some alumni have both TECH and NONTECH experiences.

Recalling the definition of Tech and Non-tech in this project, Non-tech stands for alumni who have no experience being labelled as 'TECH', it is easy to calculate that there are **320** Tech alumni **39** Non-tech alumni, as shown in below Table 3.

Undergrad	Graduate	Total
175	184	359

Table 2: Undergraduate and graduate headcounts

Tech pid	Non-tech pid	Total
320	39	359

Table 3: Tech and Non-tech headcounts in the *experience* table

6. More Experience Analysis

Aside from experiences explorations done previously, it is also useful to take a look at alumni experiences after Mills CS and compare it with their experiences before Mills CS. This can effectively help in the evaluation of the outcomes of Mills CS educations. It may not be beneficial in analyzing undergraduate alumni since most of them only started their career after Mills CS. But for graduate alumni, it is an effective approach to assess whether they achieved their goal of transitioning into Tech field after Mills CS. In this section, we first investigated the percentage of alumni having a job before and after Mills CS, as well as the differences of the percentages between undergraduate and graduate students. From all alumni experiences before and after Mills CS, we then calculated the TECH and NONTECH ratios. Lastly, we focused on only the graduate alumni, and summarized how graduate alumni transitioned between Non-tech and Tech before and after Mills CS.

6.1. Ratios of having a job before and after Mills CS

This was done by conditionally merging *person*, *exp* by matching pids that meet the selecting criteria. Two dataframes were created in this step, "jobsBeforeMills" contains all available experiences before Mills and 'jobsAfterMills' contains all experiences at or after Mills CS.

```
# Merge experience and person
mergedJobs = pd.merge(exp[['pid', 'pos_title',  'pos_start_year','pos_end_year', 'job_category_adjusted' ]],
        person[['pid', 'deg_start_year', 'deg_end_year']], on=['pid'])

# Let out experiences and alumni with uncertain start year
mergedJobs = mergedJobs[(mergedJobs.deg_start_year!=0)]
mergedJobs = mergedJobs[(mergedJobs.pos_start_year!=0)]

# alumni in mergedJobs
total_number = mergedJobs.pid.nunique()
grad_number = len(set(mergedJobs.pid.unique()).intersection(grad))
undergrad_number = total_number - grad_number

# Slicing jobs before and after Mills CS
jobsBeforeMills = mergedJobs[mergedJobs.pos_start_year < mergedJobs.deg_start_year]
jobsAfterMills = mergedJobs[mergedJobs.pos_start_year >= mergedJobs.deg_start_year]

# Before Mills CS
before = jobsBeforeMills.pid.nunique()
grad_before = len(set(jobsBeforeMills.pid).intersection(grad))
undergrad_before = before - grad_before

# After Mills CS
after = jobsAfterMills.pid.nunique()
grad_after = len(set(jobsAfterMills.pid).intersection(grad))
undergrad_after = after - grad_after
```

Code Snippet 3: jobsBeforeMills and jobsAfterMills

This step required some filtering work since we filled in ‘pos_start_year’, ‘pos_end_year’, ‘deg_start_year’ and ‘deg_end_year’ when we lacked this data at the creation of AlumDB. With these uncertainties, it is not possible to determine whether certain experience is before or after Mills CS. The filtering step reduced the number of alumni from 359 to 346, which comprised of 178 graduate alumni and 168 undergraduate alumni. Also, the reason for using ‘deg_start_year’ as the threshold of jobsAfterMills rather than ‘deg_end_year’ is because 9999 was used for ‘deg_end_year’ in the situation when an alum has not finished required coursework or finished all coursework but did not receive a final degree/certificate, thus using ‘deg_end_year’ may exclude many experiences that are indeed after Mills CS.

The analysis showed that the number of alumni who had experience before Mills is 148 (116 graduates and 32 undergraduates) and the number of alumni who have jobs after Mills is 335 (169 graduates and 166 undergraduates), including current students who hold jobs.

	Before Mills CS	After Mills CS
Total	42.8%	96.8%
Graduate	65.2%[3]	94.9%
Undergraduate	19%	98.8%

Table 4: Ratio of alumni having jobs before and after Mills CS

Above Table 4 showed high employment rate for both graduate and undergraduate alumni after Mills CS; while 65.2% of the graduate alumni joined Mills CS having previous experiences, that percentage increased by 45.6% [(94.9% - 65.2%) / 65.2%] after Mills CS.

6.2. Alumni experiences TECH and NONTECH ratios before and after Mills CS

Results of running below Code Snippet 4 generated Table 4 below.

[3] Computation: number of graduate alumni having experience before Mills CS divided by the total number of graduate alumni remained in this section, i.e., 116 /178

	jobsBeforeMills	jobsAfterMills
TECH	186	1131
NONTECH	233	631

Table5: TECH and NONTECH distribution of experiences before and after Mills

```
jobsAfterMills.job_category_adjusted.value_counts()
jobsBeforeMills.job_category_adjusted.value_counts()
```

Code Snippet 4: TECH and NONTECH distribution of experiences before and after Mills CS.

We can conclude from Table 4 that the TECH position rate before Mills is 186 / (186 + 233) = 0.44 and this ratio after Mills is 1131 / (1131 + 631) = 0.64, which implied a 45.5% increase ((0.64 - 0.44) / 0.44).

6.3. Graduate alumni career changes before and after Mills

In this step, we aimed to find the changes of graduate alumni career before and after Mills CS, especially how they transitioned between Non-tech and Tech and. Among the 178 graduate alumni selected in Section 6.1 based on their experiences in the *experiences* table, some only had experiences before entering Mills CS, some only have experiences after entering Mills CS, and some of them have experiences in both periods. The first step in this subsection is to slice out graduate experience before and after Mills CS and then separate the 178 graduate alumni into these three categories, namely ‘beforeOnly’, ‘both’ and ‘afterOnly’. This gives 9 graduate alumni in beforeOnly, 107 graduate alumni in both, and 62 graduate alumni in afterOnly (results from Code Snippet 5).

```
gradJobsBeforeMills = jobsBeforeMills[jobsBeforeMills.pid.isin(grad)]
gradJobsAfterMills = jobsAfterMills[jobsAfterMills.pid.isin(grad)]

# Separating graduate alumni and experiences

# 1: alumni with experience both before and after Mills
both = set(gradJobsBeforeMills.pid).intersection(gradJobsAeforeMills.pid)

# 2: alumni with no experience after Mills yet
beforeOnly = list(set(gradJobsBeforeMills.pid)-both)

# 3: alumni with no experience before Mills
afterOnly = list(set(gradJobsAfterMills.pid)-both)
```

Code Snippet 5: Categorizing graduate alumni

For alumni in both, we categorize their career changes before and after Mills into Non-tech to Tech, Tech to Tech, Tech to Non-tech and Non-tech to Non-tech; For alumni in after, we summarize the Tech vs. Non-tech ratio; For alumni in before, we try to find a reason for them not having experience after Mills CS with the data we have.

For the 107 graduate alumni in both, the distribution, by running below Code Snippet 6, is:

Non-tech → Tech:	22
Tech → Tech:	68
Tech → Non-Tech:	12
Non-tech → Non-tech:	5

```
# Non-tech to Tech
nTot= 0
for id in both:
    if "TECH" not in gradJobsBeforeMills[gradJobsBeforeMills.pid == id].job_category_adjusted.values:
        if "TECH" in gradJobsAfterMills[gradJobsAfterMills.pid == id].job_category_adjusted.values:
            nTot+=1
```

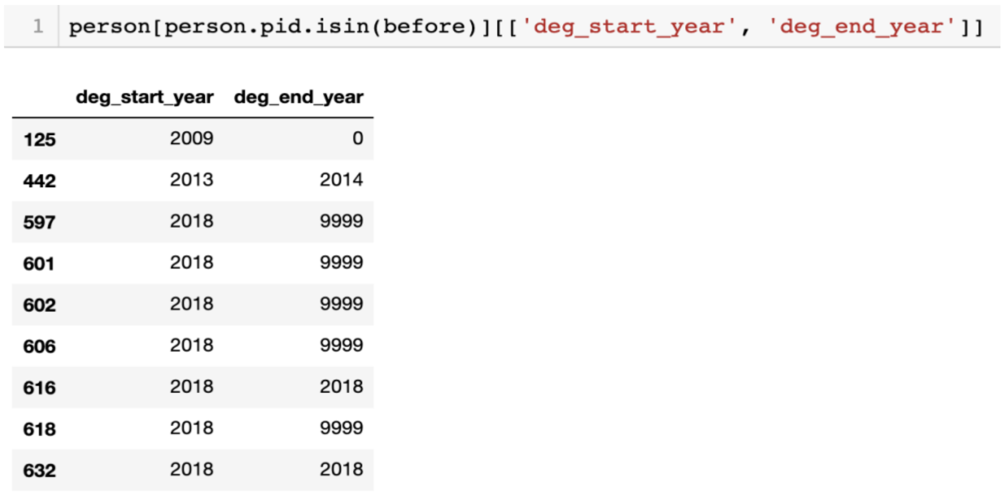
```
# Tech to Tech
tTot= 0
for id in both:
    if "TECH" in gradJobsBeforeMills[gradJobsBeforeMills.pid == id].job_category_adjusted.values:
        if "TECH" in gradJobsAfterMills[gradJobsAfterMills.pid == id].job_category_adjusted.values:
            tTot+=1

# Tech to Non-tech
tTon= 0
for id in both:
    if "TECH" in gradJobsBeforeMills[gradJobsBeforeMills.pid == id].job_category_adjusted.values:
        if "TECH" not in gradJobsAfterMills[gradJobsAfterMills.pid == id].job_category_adjusted.values:
            tTon+=1

# Non-tech to Non-tech
nTon= 0
for id in both:
    if "TECH" not in gradJobsBeforeMills[gradJobsBeforeMills.pid == id].job_category_adjusted.values:
        if "TECH" not in gradJobsAfterMills[gradJobsAfterMills.pid == id].job_category_adjusted.values:
            nTon+=1
```

Code Snippet 6: Categorizing graduate alumni career changes before and after Mills CS

For the 62 alumni in afterOnly, 55 of them are in Tech and 7 of them are in Non-tech. And for the 9 alumni in beforeOnly, we investigated their deg_start_year and found that 7 of them joined Mills in 2018 and are still current students or recent graduates.



```
person[person.pid.isin(before)][['deg_start_year', 'deg_end_year']]
```

	deg_start_year	deg_end_year
125	2009	0
442	2013	2014
597	2018	9999
601	2018	9999
602	2018	9999
606	2018	9999
616	2018	2018
618	2018	9999
632	2018	2018

Figure 24: Degree info of graduate alumni in beforeOnly

All in all, of all 178 graduate alumni whose deg_start_year were known, 145 are in Tech after Mills CS. Among the 33 (178 - 145) graduate alumni who are in Non-tech, some of them are current students or recent graduates whose information has not been updated.

7. Understanding Biases

Throughout the project life cycle, there may exist a few potential biases:

- Survival Bias

  Survival Bias is a type of selection bias, which may occur when data sample is chosen over some selection process. In this project, alumni data is obtained from their LinkedIn profiles. Only 359 profiles were available, while the rest 634 - 359 = 275 of the alumni profiles were missing from the sample. Looking at the history of LinkedIn, it was founded in 2003, and started to be popular worldwide in 2011 when it had 90 million users [19]. In 2013, a decade after it was found, LinkedIn became truly dominant with 225 million members [20]. It is also a time when a group of our older alumni had already retired or were at a stable stage of a long career. It would not be appealing for them to create a LinkedIn profile, and some may also use professional network other than LinkedIn. All in all, since the sample data does not represent all alumni, survival bias may exist.

- Questionable data integrity

  Since alumni education, experience and skill data was solely obtained from LinkedIn profiles, data integrity may be questionable. A LinkedIn user can literally write anything on their own profile at their own will, faking or exaggerating their experiences and skills. Because of this, alumni data used in this project may not be authentic. Additionally, information on LinkedIn profiles may be incomplete, which impacted the job title classification and may also lead to incorrect analysis results.

- Imbalanced training dataset

  The training dataset SOC2018 is unevenly labeled as NONTECH : TECH = 10:1. It's a classification system that is created in a broad manner as a general taxonomy of US occupations. It is not a perfect fit for Mills CS alumni occupations which are more focused in computer science and engineering fields. The imbalanced dataset could possibly lead to extreme cases where the model decides to predict all positions as NONTECH and yet still gets high accuracy. This problem was not as significant and obvious in this project since the level of imbalance was not too high.

8. Results and Conclusions

After performing the above analysis, the answers to the four questions are now clear:

1. What is the percentage of alumni working in technology vs. non-technology?

According to Section 5, the number of Tech alumni is 320 and the number of Non-tech alumni is 39. Therefore, the percentage of alumni working in technology is 320 / 359 = **89.1%** and the percentage of alumni not working in technology is 39 / 359 = **10.9%**.

2. What skills should current students build to get on the path of technical or non-technical roles?

- Overall top-10 Tech and Non-tech skills

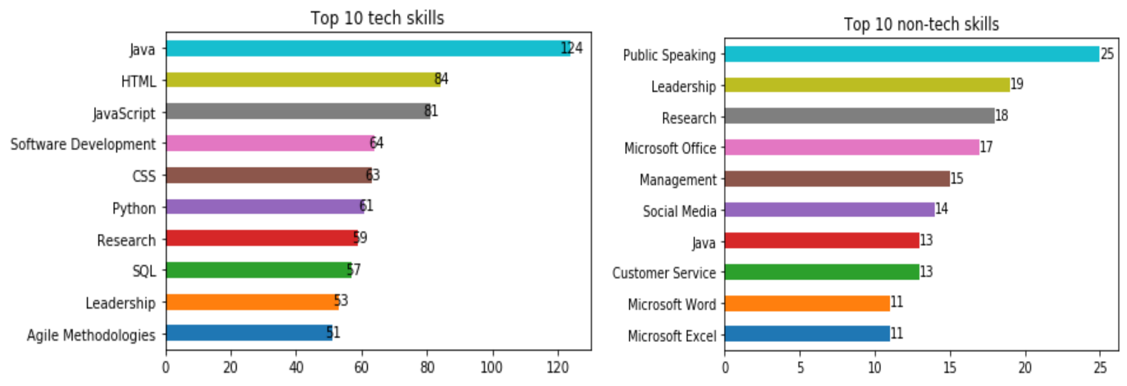


Figure 25: Overall top-10 Tech and Non-tech skills

The Tech skills mostly coincide with the overall top 10 skills discovered in Section 4.3.3. As for Non-tech, top 10 skills are totally different from those in Tech. Public Speaking, Leadership and Research are the top 3 most common skills possessed by Non-tech Mills CS alumni.

- Top Tech skills for undergraduates and graduates

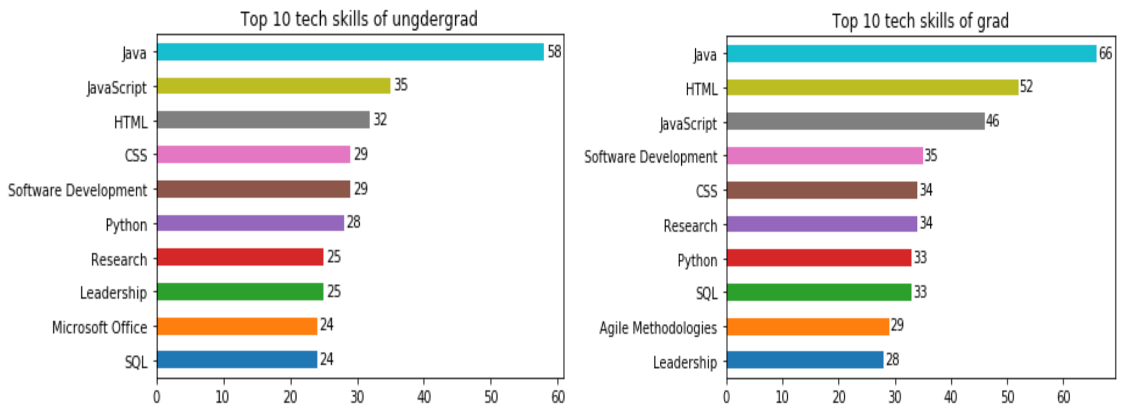


Figure 26: Top Tech skills for undergraduates and graduates

- Top Non-tech skills for undergraduate and graduate

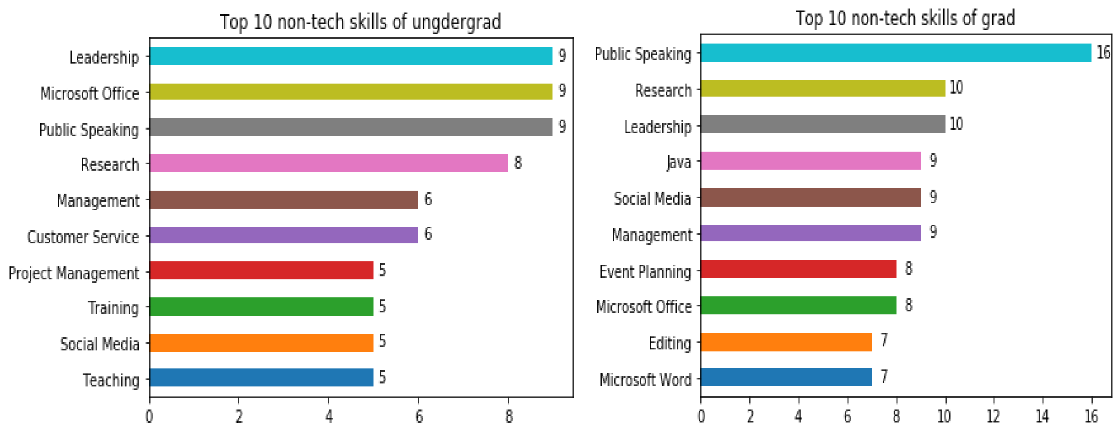
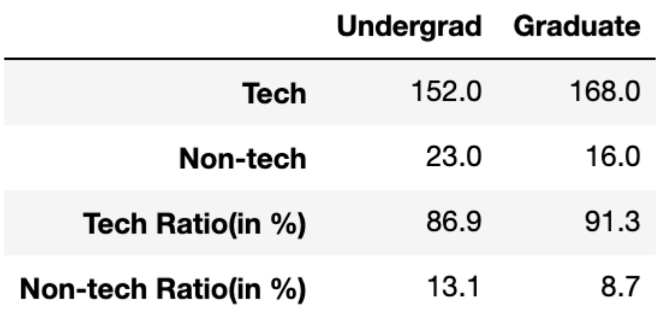


Figure 27: Top Non-tech skills for undergraduate and graduate

These skills distributions are reasonably understandable, although there were no timestamps associated with skills, so it is not possible to determine whether these skills were built during their time at Mills. It is definitely viable to build these skills through the Mills CS education.

3. What are the career differences between undergraduate and graduate alumni?



	Undergrad	Graduate
Tech	152.0	168.0
Non-tech	23.0	16.0
Tech Ratio(in %)	86.9	91.3
Non-tech Ratio(in %)	13.1	8.7

Figure 28: Undergrad and Graduate Tech and Non-tech headcount and ratios

According to above Figure 27, **91.3%** of the graduate alumni are in Tech and that is also higher than the overall Tech ratio of 89.1%. For graduate alumni, a ratio of more than ninety percent is indeed delightful considering they all came from non-CS background. As for the undergraduate alumni, **86.9%** are in Tech, which is slightly below average.

4. What are the career differences of alumni before and after Mills CS?

As can be concluded from Section 6 More Experience Analysis:

	Before Mills CS	After Mills CS
Rate of having a job [Total]	42.8%	96.8%
Rate of having a job [Graduate]	65.2%	94.9%
Rate of having a job [Undergraduate]	19%	98.8%
TECH Experiences	186	1131
TECH Ratio	44%	64%

NONTECH Experiences	233	631
NONTECH Ratio	56%	36%
Graduate Non-tech to Tech Rate	20.5%	
Graduate Tech to Tech Rate	63.6%	
Graduate Tech to Non-tech Rate	11.2%	
Graduate Non-tech to Non-tech Rate	4.7%	

Table 6: Summary of Career difference before and after Mills CS

9. Future Work

Due to time and resource limitations, this project was not able to explore a number of optimizations and extensions. One potential optimization is to find a better way to collect alumni data that is more trustworthy and authentic. It could be done by comparing and merging data from different resources besides LinkedIn, or by soliciting information directly from alumni, if possible. Moreover, if we could have skills data that is associated with individual jobs has related timestamp, then we could be able to determine and analyze the type of skills needed for the type of jobs that alumni currently have, as well as when do alumni build such type of skills -- whether or not during their Mills CS education. Another meaningful direction for future work is to find better job classification techniques, from better training dataset to better classification model. Furthermore, adding other attributes and metrics, such as salary, position level, may provide relevant information in order to stratify occupation and evaluate career path and success.

The initial goal of this project aims to evaluate the impact a Mills CS education on alumni career success. This project has definitely shed light on the outcomes and expectations of having a Mills CS education. What would strengthen this project further is having verified, reliable data, and how to do so is another area for future work.

It is also my hope to disseminate this analysis and make it useable for other institutions or programs in the future.

References

[1] N. Rubens, M. Russell, R. Perez, J. Huhtamaki, K. Still, D. Kaplan, and T. Okamoto, "Alumni network analysis," *2011 IEEE Global Engineering Education Conference (EDUCON)*, 2011.

[2] T. Case, A. Gardiner, P. Rutner, and J. Dyer, "A LinkedIn Analysis of Career Paths of Information Systems Alumni," *Journal of the Southern Association for Information Systems*, vol. 1, no. 1, Jan. 2013.

[3] Z. J. Mejia, "Alumnae/i database and website," Master's thesis, Department of Math and Computer Science, Mills College, Oakland, CA, 2014.

[4] J. Gilbertson, J. Rudolf, C. Garcia, M. Hanna, C. Williams, U. Balis, L. Pantanowitz, and J. Tuthill, "Career paths of pathology informatics fellowship alumni," *Journal of Pathology Informatics*, vol. 9, no. 1, p. 14, 2018.

[5] J. Diaz, "Security Analysis Methodology for Student Web Applications: A Case Study of the Mills College Computer Science Department Alumni Website", Master's thesis, Department of Math and Computer Science, Mills College, Oakland, CA, 2018.

[6] M. Martin, "An Assessment of Energy-Related Career Paths of Senior Industrial Assessment Center Program Alumni," 2003. University of North Texas Libraries, Digital Library, https://digital.library.unt.edu; crediting UNT Libraries Government Documents Department. (https://digital.library.unt.edu/ark:/67531/metadc876851/. [Accessed: 04-Jun-2019].

[7] J. Meer and H. Rosen, "The Impact of Athletic Performance on Alumni Giving: An Analysis of Micro Data," Economics of Education Review, 28(3), pp. 287–294, 2008.

[8] HRZone. (2019). What is a Career Path?. [online] Available at: https://www.hrzone.com/hr-glossary/what-is-a-career-path. [Accessed: 04-Jun-2019].

[9] Nlp.stanford.edu. (2019). Naive Bayes text classification. [online] Available at: https://nlp.stanford.edu/IR-book/html/htmledition/naive-bayes-text-classification-1.html. [Accessed: 04-Jun-2019].

[10] MBA Skool-Study.Learn.Share. (2019). Career Path Definition | Human Resources (HR) Dictionary | MBA Skool-Study.Learn.Share.. [online] Available at: https://www.mbaskool.com/business-concepts/human-resources-hr-terms/1782-career-path.html. [Accessed: 04-Jun-2019].

[11] "Applying Multinomial Naive Bayes to NLP Problems: A Practical Explanation," *Synced*, 17-Jul-2017. [Online]. Available: https://syncedreview.com/2017/07/17/applying-multinomial-naive-bayes-to-nlp-problems-a-practical-explanation/. [Accessed: 04-Jun-2019].

[12] garakgarak 91531728, jlund3jlund3 858713, and sjm.majewskisjm.majewski 2, "Difference between naive Bayes & multinomial naive Bayes," *Cross Validated*. [Online]. Available: https://stats.stackexchange.com/questions/33185/difference-between-naive-bayes-multinomial-naive-bayes. [Accessed: 04-Jun-2019].

[13] S. J. Russell and P. Norvig, Artificial intelligence a modern approach. Boston: Pearson, 2018.

[14] freeCodeCamp.org, "How to process textual data using TF-IDF in Python," *freeCodeCamp.org News*, 27-May-2018. [Online]. Available: https://www.freecodecamp.org/news/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3/. [Accessed: 04-Jun-2019].

[15] "A Brief Overview and History of LinkedIn," *CandyBitSocial*. [Online]. Available: https://candybitsocial.com/news/history-of-linkedin. [Accessed: 04-Jun-2019].

[16] L. Li, G. Zheng, S. Peltsverger, and C. Zhang, "Career Trajectory Analysis of Information Technology Alumni," *Proceedings of the 17th Annual Conference on Information Technology Education - SIGITE 16*, 2016.

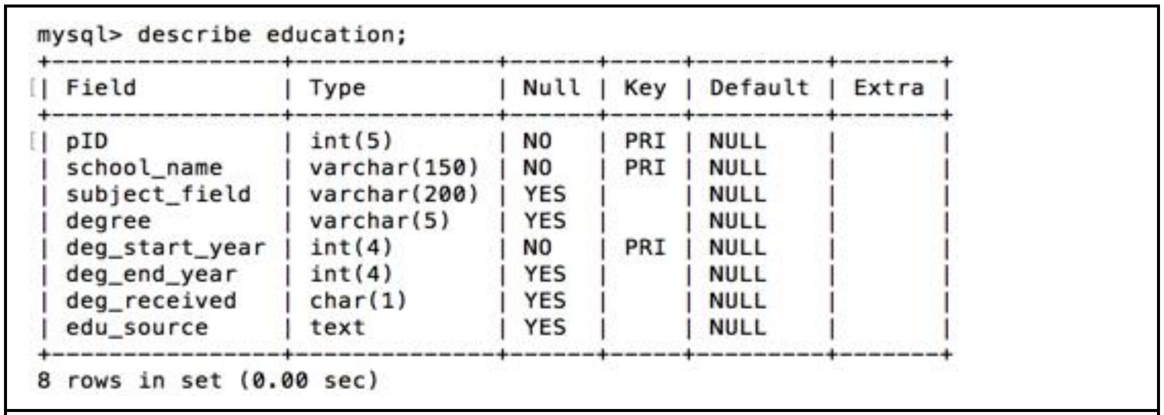
[17] S. Massoni, M. Olteanu, and P. Rousset, "Career-Path Analysis Using Optimal Matching and Self-Organizing Maps," *Advances in Self-Organizing Maps Lecture Notes in Computer Science*, pp. 154–162, 2009.

[18] K. Pina, *Career path analysis of professionals selected by MIT undergraduates*. MIT, Department of Mechanical Engineering, 2018.

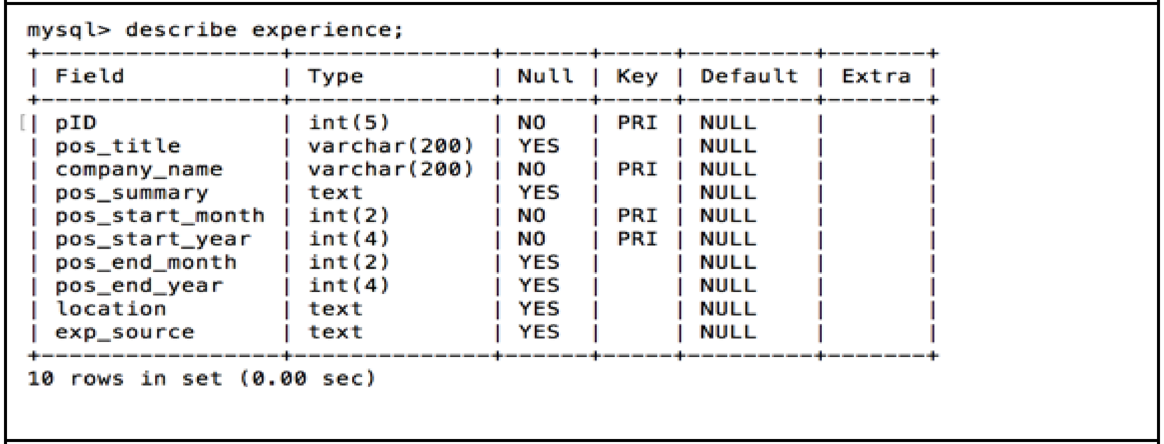
[19] "A Brief History of LinkedIn," *A Brief History of LinkedIn*. [Online]. Available: https://ourstory.linkedin.com/. [Accessed: 21-Jun-2019].

[20] S. Adams, "New Survey: LinkedIn More Dominant Than Ever Among Job Seekers And Recruiters, But Facebook Poised To Gain," *Forbes*, 05-Feb-2013. [Online]. Available: https://www.forbes.com/sites/susanadams/2013/02/05/new-survey-linked-in-more-dominant-than-ever-among-job-seekers-and-recruiters-but-facebook-poised-to-gain/#5cd03af7414b. [Accessed: 21-Jun-2019].

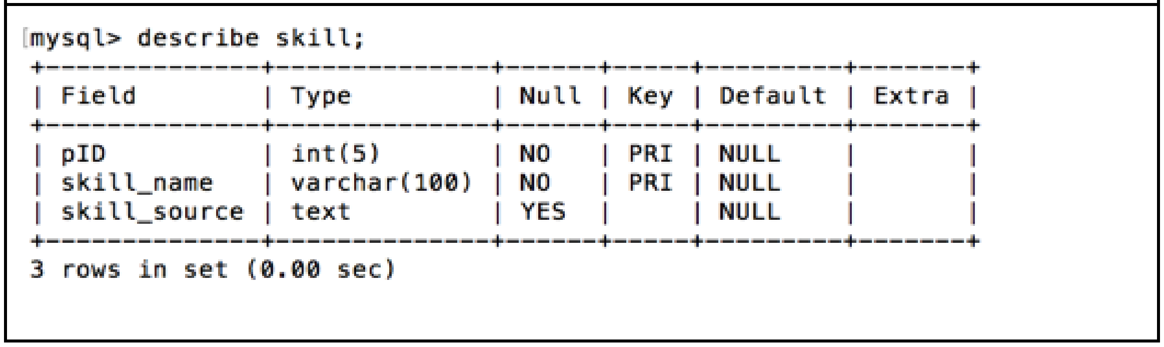
Appendix A: AlumDB_2018 structure



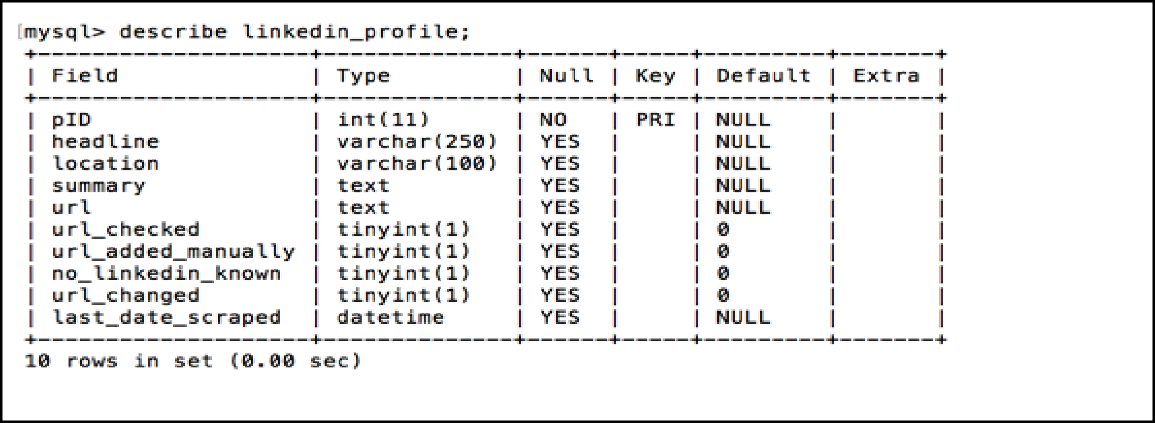
```
mysql> describe education;
+-----------------+--------------+------+-----+---------+-------+
[| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
[| pID             | int(5)       | NO   | PRI | NULL    |       |
 | school_name     | varchar(150) | NO   | PRI | NULL    |       |
 | subject_field   | varchar(200) | YES  |     | NULL    |       |
 | degree          | varchar(5)   | YES  |     | NULL    |       |
 | deg_start_year  | int(4)       | NO   | PRI | NULL    |       |
 | deg_end_year    | int(4)       | YES  |     | NULL    |       |
 | deg_received    | char(1)      | YES  |     | NULL    |       |
 | edu_source      | text         | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```



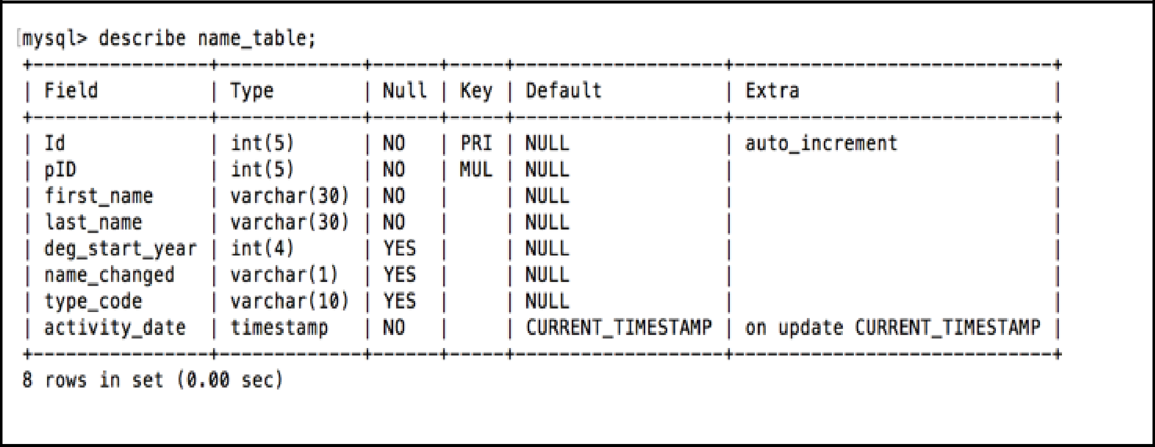
```
mysql> describe experience;
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
[| pID             | int(5)       | NO   | PRI | NULL    |       |
 | pos_title       | varchar(200) | YES  |     | NULL    |       |
 | company_name    | varchar(200) | NO   | PRI | NULL    |       |
 | pos_summary     | text         | YES  |     | NULL    |       |
 | pos_start_month | int(2)       | NO   | PRI | NULL    |       |
 | pos_start_year  | int(4)       | NO   | PRI | NULL    |       |
 | pos_end_month   | int(2)       | YES  |     | NULL    |       |
 | pos_end_year    | int(4)       | YES  |     | NULL    |       |
 | location        | text         | YES  |     | NULL    |       |
 | exp_source      | text         | YES  |     | NULL    |       |
+-----------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```



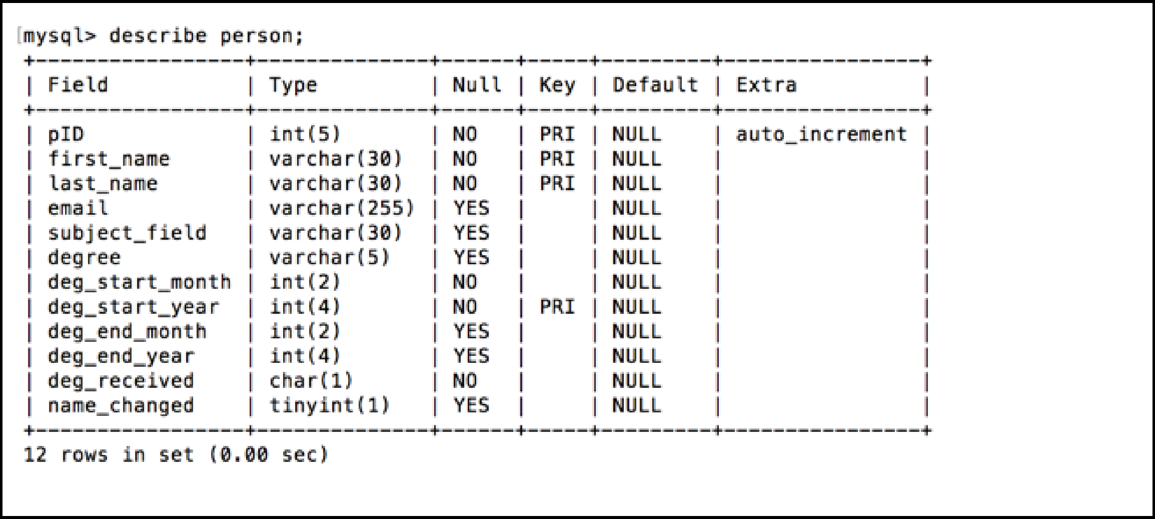
```
[mysql> describe skill;
+--------------+--------------+------+-----+---------+-------+
| Field        | Type         | Null | Key | Default | Extra |
+--------------+--------------+------+-----+---------+-------+
| pID          | int(5)       | NO   | PRI | NULL    |       |
| skill_name   | varchar(100) | NO   | PRI | NULL    |       |
| skill_source | text         | YES  |     | NULL    |       |
+--------------+--------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```



```
mysql> describe linkedin_profile;
+----------------------+--------------+------+-----+---------+-------+
| Field                | Type         | Null | Key | Default | Extra |
+----------------------+--------------+------+-----+---------+-------+
| pID                  | int(11)      | NO   | PRI | NULL    |       |
| headline             | varchar(250) | YES  |     | NULL    |       |
| location             | varchar(100) | YES  |     | NULL    |       |
| summary              | text         | YES  |     | NULL    |       |
| url                  | text         | YES  |     | NULL    |       |
| url_checked          | tinyint(1)   | YES  |     | 0       |       |
| url_added_manually   | tinyint(1)   | YES  |     | 0       |       |
| no_linkedin_known    | tinyint(1)   | YES  |     | 0       |       |
| url_changed          | tinyint(1)   | YES  |     | 0       |       |
| last_date_scraped    | datetime     | YES  |     | NULL    |       |
+----------------------+--------------+------+-----+---------+-------+
10 rows in set (0.00 sec)
```



```
mysql> describe name_table;
+----------------+-------------+------+-----+-------------------+-----------------------------+
| Field          | Type        | Null | Key | Default           | Extra                       |
+----------------+-------------+------+-----+-------------------+-----------------------------+
| Id             | int(5)      | NO   | PRI | NULL              | auto_increment              |
| pID            | int(5)      | NO   | MUL | NULL              |                             |
| first_name     | varchar(30) | NO   |     | NULL              |                             |
| last_name      | varchar(30) | NO   |     | NULL              |                             |
| deg_start_year | int(4)      | YES  |     | NULL              |                             |
| name_changed   | varchar(1)  | YES  |     | NULL              |                             |
| type_code      | varchar(10) | YES  |     | NULL              |                             |
| activity_date  | timestamp   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+----------------+-------------+------+-----+-------------------+-----------------------------+
8 rows in set (0.00 sec)
```



```
mysql> describe person;
+------------------+--------------+------+-----+---------+----------------+
| Field            | Type         | Null | Key | Default | Extra          |
+------------------+--------------+------+-----+---------+----------------+
| pID              | int(5)       | NO   | PRI | NULL    | auto_increment |
| first_name       | varchar(30)  | NO   | PRI | NULL    |                |
| last_name        | varchar(30)  | NO   | PRI | NULL    |                |
| email            | varchar(255) | YES  |     | NULL    |                |
| subject_field    | varchar(30)  | YES  |     | NULL    |                |
| degree           | varchar(5)   | YES  |     | NULL    |                |
| deg_start_month  | int(2)       | NO   |     | NULL    |                |
| deg_start_year   | int(4)       | NO   | PRI | NULL    |                |
| deg_end_month    | int(2)       | YES  |     | NULL    |                |
| deg_end_year     | int(4)       | YES  |     | NULL    |                |
| deg_received     | char(1)      | NO   |     | NULL    |                |
| name_changed     | tinyint(1)   | YES  |     | NULL    |                |
+------------------+--------------+------+-----+---------+----------------+
12 rows in set (0.00 sec)
```

Appendix B: Code of Figures

Loading Necessary Packages

In [ ]:
```python
import pandas as pd
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfVectorizer
import matplotlib.pyplot as plt
```

Loading AlumDB_18 data

In [ ]:
```python
# data downloaded in advance
exp = pd.read_csv('experiences.csv')
edu = pd.read_csv('educations.csv')
skill = pd.read_csv('skills.csv')
person = pd.read_csv('person.csv')
```

Data cleaning

Figure 1: Number of null in person table

In [ ]:
```python
person.isnull().sum()
```

Figure 2: Number of null in skill table

In [ ]:
```python
skill.isnull().sum()
```

Figure 3: Number of null in education table

In [ ]:
```python
edu.isnull().sum()
```

Figure 4: Number of null in experience table

In [ ]:
```python
exp.isnull().sum()
exp[exp.company_name.isnull()]
exp.company_name = exp.company_name.fillna('Unknown, Co. Ltd.')

exp[exp.isnull()]
exp.isnull().sum()
```

Spliting Undergraduate and Graduate

In [ ]:
```python
grad = person[(person.degree == 'MA') | (person.degree == 'CSCrt')].pid.values
undergrad = list(set(person.pid) - set(grad))

gradExp = exp[exp.pid.isin(grad)]
underExp = exp[~exp.pid.isin(grad)]

gradEdu = edu[edu.pID.isin(grad)]
underEdu = edu[~edu.pID.isin(grad)]

gradSki = skill[skill.pID.isin(grad)]
underSki = skill[~skill.pID.isin(grad)]
```

Preliminary data exploration

Person

In [ ]:
```python
# Person
person.head()

total=person.deg_start_year.value_counts().sort_index(ascending=True)[1:]
gr=person[person.pid.isin(grad)].deg_start_year.value_counts().sort_index(ascending=True)[1:]

under=person[person.pid.isin(undergrad)].deg_start_year.value_counts().sort_index(ascending=True)[1:]
df = pd.concat([total,under, gr], axis=1)
df.columns = ['total','undergrad', 'graduate']
```

Figure 5: Mills CS yearly new student enrolment number distribution

In [ ]:
```python
df2 = pd.concat([under, gr], axis=1)
df2.columns = ['undergrad', 'graduate']
df2.plot(kind='bar', stacked=True,title= "Mills CS new student enrollment numbers", figsize=(15, 5))
```

In [ ]:
```python
# first cs alum
person[person.deg_start_year==1972]
```

Experience

Figure 6: Top 10 common positions of all alumni

In [ ]:
```python
pd.DataFrame(exp.groupby(
    ['pid', 'pos_title']).size().reset_index(
    name='Freq')).pos_title.value_counts()[:10]
```

Figure 7: Top 10 common positions of undergraduate alumn

In [ ]:
```python
pd.DataFrame(underExp.groupby(
    ['pid', 'pos_title']).size().reset_index(
    name='Freq')).pos_title.value_counts()[:10]
```

Figure 8: Top 10 common positions of graduate alumni

In [ ]:
```python
pd.DataFrame(gradExp.groupby(
    ['pid', 'pos_title']).size().reset_index(
    name='Freq')).pos_title.value_counts()[:10]
```

Figure 9: Top 10 common alumni positions held for more than 3 years

In [ ]:
```python
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>3].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()
```

Figure 10: Top 10 common alumni positions held for more than 5 years

In [ ]:
```python
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>5].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()
```

Figure 11: Top 10 common alumni positions held for more than 10 years

In [ ]:
```python
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>10].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()
```

Figure 12: Top 10 common alumni positions held for more than 20 years

In [ ]:
```python
pd.DataFrame(exp[exp.pos_end_year-exp.pos_start_year>20].groupby(
    ['pid', 'pos_title']).size()
            .reset_index(name='Freq')).pos_title.value_counts()
```

Skill: Common skills

Figure 13: Top 10 common alumni skills

In [ ]:

```python
ax=skill.skill_name.value_counts()[:10].iloc[::-1].plot(kind="barh",
                                                        title="Top 10 Skills of all ",
                                                        figsize=(15, 5))

for p in ax.patches:
    width = p.get_width()
    plt.text(1.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Figure 14: Top 10 common endorsed alumni skills

In [ ]:

```python
ax=skill[skill.endorsement>0].skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title="Top 10 Endorsed Skills of all ",
                           figsize=(15, 5))

for p in ax.patches:
    width = p.get_width()
    plt.text(1.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Figure 15: Top 10 common undergraduate alumni skills

In [ ]:

```python
ax=underSki.skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title="Top 10 Skills of Undergrad",
                           figsize=(15, 5))

for p in ax.patches:
    width = p.get_width()
    plt.text(.5 +p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Figure 16: Top 10 common endorsed undergraduate alumni skills

In [ ]:

```python
ax=underSki[underSki.endorsement>0].skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title="Top 10 Endorsed Skills of Undergrad",
                           figsize=(15, 5))

for p in ax.patches:
    width = p.get_width()
    plt.text(.5 +p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Figure 17: Top 10 common graduate alumni skills

In [ ]:

```python
ax=gradSki.skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title="Top 10 Skills of Grad",
                           figsize=(15, 5) )

for p in ax.patches:
    width = p.get_width()
    plt.text(.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Figure 18: Top 10 common endorsed graduate alumni skills

In [ ]:

```python
ax=gradSki[gradSki.endorsement>0].skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title="Top 10 Endorsed Skills of Grad",
                           figsize=(15, 5))
for p in ax.patches:
    width = p.get_width()
    plt.text(.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Classifying Alumni Experiences

```
In [ ]: # SOC_2018 distribution
        job_titles['SOC_Category'] = job_titles['SOC_Category'].replace([1],'TECH')
        job_titles['SOC_Category'] = job_titles['SOC_Category'].replace([0],'NONTECH')
        job_titles["SOC_Category"].value_counts()
        job_titles["SOC_Category"].value_counts()
```

```
In [ ]: from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from nltk.tokenize import RegexpTokenizer
        from nltk.stem.snowball import SnowballStemmer
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix
        from sklearn.metrics import f1_score
        from sklearn.metrics import accuracy_score
        import pandas as pd
```

```
In [ ]: # Load the data SOC and Alum
        job_titles = pd.read_csv("soc_2018_definitions.csv")
        print("size of all jobs is: ", len(job_titles))  # 3730
        exp = pd.read_csv("experiences.csv")
        print("size of exp is: ", len(exp))    # 2283

        # # Convert the categories TECH, NONTECH to numbers 1, 0
        job_titles['SOC_Category'] = job_titles['SOC_Category'].replace(['TECH'],1)
        job_titles['SOC_Category'] = job_titles['SOC_Category'].replace(['NONTECH'],0)

        # deal with NaN
        job_titles["SOC_Title"].fillna(" ", inplace = True)
        job_titles["SOC_Definition"].fillna(" ", inplace = True)

        # concat job title and summary
        job_titles['job'] = job_titles[["SOC_Title",
                                        "SOC_Definition"]].apply(lambda x: ''.join(x), axis=1)

        # tokenize
        tokenizer = RegexpTokenizer(r'[A-Za-z]+')
        job_titles['job'] = job_titles.job.map(lambda x:tokenizer.tokenize(x))

        # stemming
        stemmer = SnowballStemmer("english")
        job_titles['job'] = job_titles.job.map(lambda l: [stemmer.stem(word) for word in l])
        job_titles.job = job_titles.job.str.join(sep=' ')
        cv = CountVectorizer(stop_words='english')
        job_titles_tf = cv.fit_transform(job_titles.job)
```

```
In [ ]: trainX,testX,trainY,testY = train_test_split(job_titles_tf, job_titles.SOC_Category)
        mnb = MultinomialNB()
        mnb.fit(job_titles_tf[:1446], job_titles.SOC_Category[:1446])
        MultinomialNB(alpha=1.0, fit_prior=True)
        exp['job_category'] = mnb.predict(job_titles_tf[1447:])

        # change category back to TECH and NONTECH
        exp['job_category'] = exp['job_category'].replace([1],'TECH')
        exp['job_category'] = exp['job_category'].replace([0],'NONTECH')
```

Figure 20: Top 15 common jobs classified as TECH and NONTECH

```
In [ ]: exp[exp.job_category=='TECH']['pos_title'].value_counts()[:15]
        exp[exp.job_category=='NONTECH']['pos_title'].value_counts()[:15]
```

Figure 21: TECH and NONTECH experiences distribution

```
In [ ]: ax = exp.job_category.value_counts().plot.bar(title="TECH vs. NONTECH")
        for p in ax.patches:
            ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```

Handling Job Mis-classifications

```
In [ ]: exp['job_category_adjusted'] = exp['job_category']
        words = ['product',
                 'project',
                 'analyst',
                 'lab',
                 'tutor',
                 'computer',
                 'scientist',
                 'engineer',
                 'software',
                 'hardware',
                 'it',
                 'developer',
                 'technical',
                 'data',
                 'machine',
                 'web',
                 'frontend',
                 'backend',
                 'database',
                 'technology',
                 'information',
                'database',

                ]


        for word in words:
            exp.loc[(exp.pos_title.str.contains(word, case=False)==True),
                    'job_category_adjusted'] = 'TECH'
```

Figure 22: Top 15 common TECH and NONTECH roles after manual adjusting

```
In [ ]: exp[exp.job_category_adjusted=='TECH']['pos_title'].value_counts()[:15]
        exp[exp.job_category_adjusted=='NONTECH']['pos_title'].value_counts()[:15]
```

Figure 23: TECH and NONTECH roles distribution after adjusting

```
In [ ]: ax = exp.job_category_adjusted.value_counts().plot.bar(title="TECH vs. NONTECH")
        for p in ax.patches:
            ax.annotate(str(p.get_height()), (p.get_x() * 1.005, p.get_height() * 1.005))
```

Determining Tech and Non-tech

```
In [ ]: # get tech and nontech pids
        def get_tech_pID(list1, list2):
            count = []
            for i in list1:
                if i not in list2:
                    count.append(i)
            return list(set(list1).intersection(list2)) + count

        techpID = get_tech_pID(exp[exp.job_category_adjusted=='TECH'].pid.unique().tolist(),
                   exp[~(exp.job_category_adjusted=='TECH')].pid.unique().tolist())
        nontechpID = set(exp.pid.unique()) - set(techpID)
        print(len(techpID), ", ", len(nontechpID))
```

```
In [ ]: print(len(set(techpID).intersection(undergrad)))
        print(len(set(techpID).intersection(grad)))

        print(len(set(nontechpID).intersection(undergrad)))
        print(len(set(nontechpID).intersection(grad)))
```

Figure 27: Undergrad and Graduate Tech and Non-tech headcount and ratios

```
In [ ]: headcount={'Undergrad': [152, 23, 86.9, 13.1],
                'Graduate': [168, 16, 91.3, 8.7]}

        data = pd.DataFrame(headcount, index=['Tech', 'Non-tech', 'Tech Ratio(in %)', 'Non-tech Ratio(in %)'])
        data
```

More in job analysis

```
In [ ]: exp2 = exp[(exp.pos_start_year!=0) & (exp.pos_end_year!=0)]
        person2 = person[(person.deg_start_year!=0) & (person.deg_end_year!=0)]
```

```
In [ ]: mergedJobs = pd.merge(exp2[['pid', 'pos_title',
                                    'pos_start_year', 'job_category_adjusted']],
                              person2[['pid', 'deg_start_year', 'deg_end_year']], on=['pid'])

        jobsBeforeMills = mergedJobs[mergedJobs.pos_start_year < mergedJobs.deg_start_year]
        jobsAfterMills = mergedJobs[mergedJobs.pos_start_year >= mergedJobs.deg_end_year]

        print("Number of alumni had job before Mills: ", jobsBeforeMills.pid.nunique())
        print("Number of alumni have job after Mills: ", jobsAfterMills.pid.nunique())
```

```
In [ ]: ids = pd.merge(jobsBeforeMills, jobsAfterMills, how="inner").pid.unique()
        len(ids)
```

```
In [ ]: print("Jobs after Mills CS\n", jobsAfterMills.job_category_adjusted.value_counts())
        print("\n\nJobs before Mills CS\n ",jobsBeforeMills.job_category_adjusted.value_counts())
```

```
In [ ]: ids = pd.merge(jobsBeforeMills, jobsAfterMills, how="inner").pid.unique()
        len(ids)

        noOfTran= 0
        for id in ids:
            if "NONTECH" in jobsBeforeMills[jobsBeforeMills.pid == id].job_category_adjusted.values:
                if "TECH" in jobsAfterMills[jobsAfterMills.pid == id].job_category_adjusted.values:
                    noOfTran+=1
        noOfTran
```

Tech skills and Non-tech skills

```
In [ ]: techSki = skill[skill.pID.isin(techpID)]
        nontechSki = skill[skill.pID.isin(nontechpID)]
```

Figure 24: Overall top-10 tech and non-tech skills

```
In [ ]: ax=techSki.skill_name.value_counts(
            )[:10].iloc[::-1].plot(kind="barh",
                                   title='Top 10 tech skills')
        for p in ax.patches:
            width = p.get_width()
            plt.text(.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
                     '{:1.0f}'.format(width),
                     ha='center', va='center')
```

```
In [ ]: ax=nontechSki.skill_name.value_counts(
            )[:10].iloc[::-1].plot(kind="barh",
                                   title='Top 10 Non-tech skills')
        for p in ax.patches:
            width = p.get_width()
            plt.text(.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
                     '{:1.0f}'.format(width),
                     ha='center', va='center')
```

Non-tech / Tech and undergrad / grad

```
In [ ]: utSki = techSki[techSki.pID.isin(undergrad)]
        gtSki = techSki[techSki.pID.isin(grad)]

        untSki = nontechSki[nontechSki.pID.isin(undergrad)]
        gntSki = nontechSki[nontechSki.pID.isin(grad)]

```

Figure 25: Top tech skills for undergraduates and graduates

```
In [ ]: ax = utSki.skill_name.value_counts(
            )[:10].iloc[::-1].plot(kind="barh",
                                   title='Top 10 tech skills of ungdergrad')
        for p in ax.patches:
            width = p.get_width()
            plt.text(1.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
                     '{:1.0f}'.format(width),
                     ha='center', va='center')
```

In [ ]:

```python
ax = gtSki.skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title='Top 10 tech skills of grad')
for p in ax.patches:
    width = p.get_width()
    plt.text(1.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

Figure 26: Top non-tech skills for undergraduate and graduate ¶

In [ ]:

```python
ax=untSki.skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title='Top 10 non-tech skills of ungdergrad')
for p in ax.patches:
    width = p.get_width()
    plt.text(.2+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```

In [ ]:

```python
ax=gntSki.skill_name.value_counts(
    )[:10].iloc[::-1].plot(kind="barh",
                           title='Top 10 non-tech skills of grad')
for p in ax.patches:
    width = p.get_width()
    plt.text(.5+p.get_width(), p.get_y()+ 0.55*p.get_height(),
             '{:1.0f}'.format(width),
             ha='center', va='center')
```