

# CUNY 607 - Project 3

Team 6 Presentation

October 17, 2021

## Team Members

- Donald Butler
  - Nick Oliver
  - Jeffrey Parks
  - Mark Schamfield
- 

## Project Objective

Use data to answer the question, “Which are the most valued data science skills?”

- Identify viable data sources.
  - Load data into a relational database with normalized tables.
  - Perform tidying, transformaiton and exploratory data analytis in R.
  - Analyze data and show findings.
- 

## Project Approach

We considered potential data sources for this project, including:

- Scraping “Data Scientist” job postings from websites like Indeed and LinkedIn.
- Examine Reddit data science-related subreddits for questions related to “skills” or “careers”
- Looking at Python, NPM or R packages that mention specific skills
- Reviewing Youtube videos that mention learning some data science skill
- Interviewing question/interview study guides that are targeted at data scientists
- Finding LinkedIn public profiles of employed data scientists and what skills they list
- Analyzing StackOverflow data science-related pages

After initial discussion, we decided to attempt the first option, scraping ‘Data Scientist’ job listings.

Our initial idea was to leverage an API endpoint from one of the popular services, such as Indeed.com. However, we found that while Indeed does produce an API, it restricts access to verified Publishers of job postings. In the interests of time, and since we had some team members with experience in web scraping, we pulled job listings directly from the from LinkedIn website using a popular browser automation tool, **Selenium** using Python.

Once we had the data in hand, we identified a multi-step approach to analysis:

- Use a text mining package such **tidytext** to tokenize keywords in the job descriptions
- Manually scan the tokenized keywords for terms describing discrete job skills, and filter out non-skill-related terms.
- Built a relational database with individual tables for Job Listings, Companies and Skills
- Query the database to analyze the counts, frequency and trends.

---

## Data Acquisition

wfoerfgoewrhiugwergoh

---

## Data Transformation

Our first step was to transform the raw job descriptions into tokenized fields for analysis. We decided to create one table with all words (except for “stop words”) to analyze overall frequency at a high level, and a second table that would identify common multi-word combinations called “ngrams.”

This second approach would correctly return the frequency of phrases such as “Machine Learning”, instead of separate results for “Machine” and “Learning”. We further filtered this second table manually to identify actual job skills, and not extraneous information (such as company names and other filler text.)

1. Load the job descriptions dataset

```
urlfile<-"https://raw.githubusercontent.com/nolivercuny/data607-team-6-project-3/master/data/job_listings.csv"
jobdat <- read_csv(url(urlfile))
jobdat<-data_frame(jobdat)
```

2. Create **jobdata\_words.csv** file from dataset for analysis of overall term frequency. Used **tidytext** to un-nest the job descriptions into individual words, removing “stop words” (the, of, to, etc)

```
jobdat_word<- unnest_tokens(
  jobdat,
  word,
  description,
  token= "words",
  format=c("text"),
  to_lower=TRUE,
  drop=TRUE,
  collapse=NULL,
)

jobdat_word <-jobdat_word %>%
  anti_join(stop_words)

write_csv(jobdat_word,'data/jobdata_words.csv')
```

3. Create **jobdat\_ngrams.csv** file from raw dataset for analysis of tokenized terms/phrases that meet criteria for “skills.” Using **tidytext**, created ngrams of 1, 2, and 3 words, filtered out the common **stop\_words**, and then did a count of each and filtered down to the rows that occurred at least 10 times within the raw dataset. The resulting 6015 rows were exported to a f.csv ile which I loaded into Excel to manually determine which terms described actual job skills and which did not.

```
jobdat_1gram <- jobdat %>%
  unnest_tokens(ngram,description,token='ngrams',n=1,format='text',
                drop=TRUE,to_lower=TRUE) %>%
  filter(!ngram %in% stop_words$word) %>%
  count(ngram,sort = TRUE) %>%
  filter(n >= 10)

jobdat_2gram <- jobdat %>%
  unnest_tokens(ngram,description,token='ngrams',n=2,format='text',
                drop=TRUE,to_lower=TRUE) %>%
  separate(ngram,c('word1','word2'),sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  unite(ngram, c('word1','word2'), sep = " ") %>%
  count(ngram,sort = TRUE) %>%
  filter(n >= 10)

jobdat_3gram <- jobdat %>%
  unnest_tokens(ngram,description,token='ngrams',n=3,format='text',
                drop=TRUE,to_lower=TRUE) %>%
  separate(ngram,c('word1','word2','word3'),sep = " ") %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word3 %in% stop_words$word) %>%
  unite(ngram, c('word1','word2','word3'), sep = " ") %>%
  count(ngram,sort = TRUE) %>%
  filter(n >= 10)

jobdat_ngrams <- jobdat_1gram %>%
  rbind(jobdat_2gram) %>%
  rbind(jobdat_3gram) %>%
  arrange(desc(n))

jobdat_ngrams %>%
  write.table(file = './jobdat_ngrams.csv',quote = FALSE,
             sep = '\t', row.names = FALSE)
```

## Loading into Database

To support the data analysis, we transformed the raw dataset into a relational database using **Sqlite**, a file-based format that is both fast and portable, so that all team members could access the database without having to manage any kind of server.

1. Create Database

```
# For testing - use in-memory DB
con <- dbConnect(RSQLite::SQLite(), dbname = ":memory:")

# Uncomment to manipulate real DB
#con <- dbConnect(RSQLite::SQLite(), dbname = "project3_job_skills.db")
```

## 2. Create Tables

```
baseUrl <- "https://raw.githubusercontent.com/nolivercuny/data607-team-6-project-3/master/sql/"
tables <- c("company", "job_listing", "job_listing_skill", "job_listing_word")
for (table in tables) {
  fileUrl <- paste(baseUrl, table, ".sql", sep = "")
  createTableStatement <- getURL(fileUrl)
  print(paste("Creating ", table, " table"))
  dbSendQuery(con, createTableStatement)
}
```

## 3. Load Data From CSV

```
urlfile<-"https://raw.githubusercontent.com/nolivercuny/data607-team-6-project-3/master/data/job_listing.csv"
jobdat <- read_csv(url(urlfile))
```

## 4. Populate **company** table: unique row per company (420)

```
companiesDf <- jobdat %>%
  select(company_name, company_size, industry) %>%
  distinct()

companiesDf$company_name <- companiesDf$company_name %>% replace_na("unknown")
dbWriteTable(con, "company", companiesDf, append=TRUE)
```

column_name	data_type	attrs
id	int	pk
company_name	int	not null
company_size	int	null
industry	int	null

## 5. Populate **job\_listing** table: unique row per job listing (849)

```
# Remove columns that are in the company table
jobListingDf <- jobdat %>% select(-c("company_size", "industry"))

# Fix the one company name that is NA
jobListingDf$company_name <- jobListingDf$company_name %>% replace_na("unknown")

# Read companies to get company ID for joining
companiesWithId <- dbReadTable(con, "company")

# Join job listing with company to populate id, then drop company-specific cols
```

```
# rename id column to company id
joined <- left_join(jobListingDf, companiesWithId, by="company_name") %>%
  select(-c("company_size", "industry", "company_name")) %>%
  rename(company_id = id)

#write dataframe to job_listing table
dbWriteTable(con, "job_listing", joined, append=TRUE)
```

column_name	data_type	attrs
id	int	pk
search_rank	int	not null
job_title	text	not null
region	text	not null
applicant_count	int	null
salary	text	null
employment_type	text	not null
career_level	text	null
description	text	null
date_queried	text	null
date_posted	text	null
company_id	int	fk

6. Populate **job\_listing\_word** table: unique row per word (18,000)

```
wordsDataUrl <- "https://raw.githubusercontent.com/nolivercuny/data607-team-6-project-3/master/data/job_listing_words.csv"
wordsJobListingData <- read_csv(url(wordsDataUrl))
wordsJobListingData<-data_frame(wordsJobListingData)

wordsDf <- wordsJobListingData %>% select(search_rank, word)
jobListingWithId <- dbReadTable(con, "job_listing")
joinedWords<-left_join(wordsDf, jobListingWithId, by="search_rank") %>%
  select(id, word) %>%
  rename(job_listing_id = id, skill = word)
dbWriteTable(con, "job_listing_word", joinedWords, append=TRUE)
```

column_name	data_type	attrs
id	int	pk
skill	text	not null

7. Populate **job\_listing\_skill** table: unique row per tokenized skill (53)

```
skillsDataUrl <- "https://raw.githubusercontent.com/nolivercuny/data607-team-6-project-3/master/JobSkillTokenized.csv"
skillsData <- read_csv(url(skillsDataUrl))
skillsDf <-data_frame(skillsData)

jobListingWithId <- dbReadTable(con, "job_listing")
skillsTableDf <- data.frame(job_listing_id=integer(), skill=character())
for (i in 1:nrow(jobListingWithId)) {
  listing <- jobListingWithId[i,]
```

```

for (j in 1:nrow(skillsDf)) {
  skill <- skillsDf[j,]
  detected <- str_detect(listing$description,
                        regex(paste('^A-Z0-9',skill,'^A-Z0-9'),sep = ''),
                        ignore_case = TRUE))

  if(detected==TRUE){
    skillsTableDf <- skillsTableDf %>%
      add_row(job_listing_id = listing$id, skill = skill$JobSkill)
  }
}

dbWriteTable(con,"job_listing_skill", skillsTableDf, append=TRUE)
dbDisconnect(con)

```

---

column_name	data_type	attrs
id	int	pk
job_listing_id	int	fk
skill	text	not null

---

## Exploratory Data Analysis

For our initial EDA, we wanted to look at the raw frequency of terms among all 849 job listings, and identify some of the most commonly-used words.

---

## Conclusions

- Split between soft skills vs hard skills?
- Different types of skills per company / industry
- Companies like Amazon – using a general template?
- Importance of ‘trilingual’ Python and R, SQL
- Plus general Stats, Math