David Forteguerre
Professor Yu
IST736
2/6/2018

HW2: What does it take to become a data scientist in 2018?

I. INTRODUCTION

In 2012, *Harvard Business Review* named data scientist "the sexiest job of the 21st century." According to Forbes, data scientist has been named the best job in America for three years running, with a median base salary of $110,000 and 4,524 job openings. When you realize that 90% of the world's data has been created in the last two years, it is no wonder that more and more companies are looking for talented people who can make sense of their big data in order to make more profit.

The big question remains: what does it take to become a data scientist nowadays? It is often said that the majority of today's data scientists acquired their extensive skillset "on the job." However, considering that more and more students and professionals seek to become data scientists, universities around the world try to meet the increasing demand by creating new degrees — often M.S.'s — in Data Science; although the curricula may vary dramatically depending on the school.

This job has only gained momentum recently, and even though lots of online resources emphasize the main areas and skillsets a data scientist needs to have in 2018, it is important to keep in mind that those resources are often biased—especially if they were written by people promoting specific school programs or companies for profit.[1]

Given that the field of data science is currently thriving and that the skills required by employers keep evolving, this analysis seeks to discover what the most common areas of expertise and skills are in order to become a data scientist in 2018.

II. DATA COLLECTION AND INITIAL CLEANSING

To get an overview of the top skills needed to become a data scientist in 2018, I decided to collect my own corpus and save it as a plain .txt document. In order to do so, I used Glassdoor.com to collect 100 job postings that were: (1) *not* older than 7 days, (2) all based in the United States, (3) had a full list of qualifications/education/skills needed for the job, and (4) contained the word "data scientist" in their title (possibly combined with any of the following words: junior, senior, intern, manager, lead.). Any other jobs related to data analysis (e.g. data analytics, business analytics, etc.) were not considered.

Note that for each job posting, *only* the qualifications/education/skills needed were copy-pasted into the .txt document. All other information (e.g. the information about the company

[1] Here's a typical article emphasizing the skills needed to be a data scientist today. It seems to be very complete (especially the downloadable PDF at the end), but how accurate is it? It is always important to take these articles with a grain of salt as they could be written in companies' own interests (e.g. 'this is what we think is important, so take our courses!')
https://blog.udacity.com/2014/11/data-science-job-skills.html

and information about the job duties) was disregarded. I also made sure that the postings were diverse, from all around the country (i.e. East Coast, West Coast), had been posted by companies from different sectors (e.g. education, technology, transportation, etc.) and different sizes (small and large companies). Some of the postings I collected came from industry leaders: iHeartMedia, AT&T, Walmart eCommerce, Quora, PepsiCo, Intel Sony Pictures Entertainment, McAfee, Amazon, eBay, Delta Airlines, Chegg, Google, Adobe, Airbnb, Dropbox, NBA, Microsoft, The Hershey Company, Uber, Lyft, Glassdoor, Foot Locker, IBM Corporation, J.PMorgan.

Before starting the analysis, I decided to use my text editor to do some initial data preprocessing in order to have better records. After going through the dataset ("PostingsDataScience.txt"), I noticed several inconsistencies that needed to be fixed for the analysis:

- I removed bullet points (bullet points or hyphens.)
- I removed all instances of the following words from the data: *basic qualifications/requirement/skills,* as well as *desirable/bonus/preferred qualifications/requirement/skills,* as they were more titles than actual skills.
- I replaced *master's degree* and *master degree* by *masters.*
- I replaced *bachelor's degree* and *bachelor degree* by *bachelors*.
- I replaced *B.S.* by *BS*, *M.S*. by *MS*, and *Ph.D* by *PhD*.
- I replaced *languages* by *language*.
- I replaced *data sets* by *datasets*.

III. METHOLOGY AND RESULTS

Below is a brief description of how I processed the data in R using the RStudio IDE. For the detailed analysis with extensive comments and explanations, please look at the code submitted with this document (R: HW2 Code.R, or Python: HW2 Code.ipynb). Note that I first explored the data using Python on a more surface level, and then replicated, continued, and finished the analysis in R on a deeper lever to practice more.

The purpose of this analysis was to look at individual word frequencies and bigrams frequencies to see what the most common skills and education paths were required to land a job as a data scientist. Thus, I first imported my .txt file into R and vectorized the document using the tm package (a popular text mining package for R).
Then, I had to make several decisions about transforming my data. First, I made it lowercase so as to not have duplicates of words. This is a crucial step to not have words such as "data" and "Data" considered as separate entities. Secondly, I got rid of the punctuation present in the document as it was not relevant to this analysis either—I was looking at content, not style. Finally, I removed stop words by using the default English stop word list included in the tm package. The list is included in my code.
Note that I chose *not* to remove numbers from my data as many postings contained numeric values—especially when talking about how many years' experience a candidate should have for

a job. I also chose to get rid of a few other words that were not in the original stop word list (i.e. "etc", "e.g.", "eg") as they were irrelevant to my analysis. Finally, I created a term document matrix, counted the word frequencies for each word, and stored the results into a dataset (dataset1). I used the wordcloud package to plot the results. Below are the results.

	word	freq
1	experience	340
2	data	266
3	science	129
4	years	99
5	learning	93
6	skills	91
7	statistics	86
8	ability	83
9	python	82
10	computer	73
11	knowledge	71
12	machine	71
13	strong	69
14	r	64
15	analysis	56
16	language	56
17	statistical	55
18	programming	54
19	degree	54
20	quantitative	51
21	field	50
22	related	47
23	sql	47
24	techniques	47
25	phd	47

	word	freq
26	working	46
27	work	45
28	analytics	45
29	business	45
30	engineering	44
31	modeling	43
32	understanding	40
33	research	39
34	mathematics	38
35	using	36
36	tools	36
37	preferred	35
38	communication	35
39	plus	35
40	java	32
41	datasets	32
42	2	31
43	advanced	30
44	models	30
45	masters	30
46	one	29
47	large	28
48	software	27
49	problems	27
50	big	27

	word	freq
51	systems	26
52	mining	26
53	algorithms	25
54	spark	25
55	development	25
56	excellent	24
57	technical	24
58	analytical	24
59	hadoop	23
60	environment	23
61	applied	23
62	familiarity	22
63	predictive	22
64	including	22
65	3	22
66	deep	22
67	relevant	21
68	solutions	21
69	proficiency	21
70	equivalent	21
71	team	20
72	demonstrated	20
73	least	19
74	complex	19
75	ms	19

	word	freq
76	information	19
77	required	18
78	methods	18
79	industry	18
80	physics	18
81	operations	17
82	regression	17
83	technologies	17
84	processes	17
85	expertise	17
86	following	16
87	must	16
88	developing	16
89	visualization	16
90	like	16
91	scripting	16
92	able	16
93	written	16
94	discipline	16
95	decision	15
96	bachelors	15
97	c	15
98	distributed	15
99	databases	15
100	clustering	15

WordCloud:

The results turned out to be very interesting and already told a lot about what recruiters were looking for to hire data scientists in 2018. However, there was room for improvement as it is not always easy to make sense of single words. Even though they may help to get the gist of a document and visualize common patterns, it is important to remember that bigrams and trigrams are equally as important.
Consequently, I decided to take the analysis a step further and used the ngram package in R. I first looked at the top bigrams in the document, then at the top trigrams. Below are a few representative examples of common bigrams.

	word	freq
1	machine learning	69
2	computer science	68
3	years experience	42
4	data science	41
5	2 years	28
6	communication skills	25
7	experience data	23

	word	freq
8	science statistics	23
9	big data	22
10	experience working	22
11	programming language	21
12	data mining	21
13	3 years	20
14	related field	18

	word	freq
15	r python	17
16	operations research	16
17	python r	15
18	data analysis	14
19	predictive modeling	13
20	science related	12

We see in the results above that *machine learning, communication skills, big data, programming languages, data mining, python/r, data analysis,* and *predictive modeling* are very common patterns found in the job postings; we would not have known without analyzing bigrams! Similarly, trigrams showed the importance of *computer science statistics, machine learning techniques, degree [in] computer science, natural language processing, problem solving skills,* and *machine learning algorithms*.

As a final step, I created one last dataset combining (1) the top 100 word frequencies found previously in the analysis along with (2) the top 50 bigrams and (3) the top 50 trigrams. I cleaned the dataset and sorted it by frequency (from high to low)—the results were already much more conclusive when put all together! Below is a summary of the *top 30* entities in the final dataset:

	word	freq
1	experience	340
2	data	266
3	science	129
4	years	99
5	learning	93
6	skills	91
7	statistics	86
8	ability	83

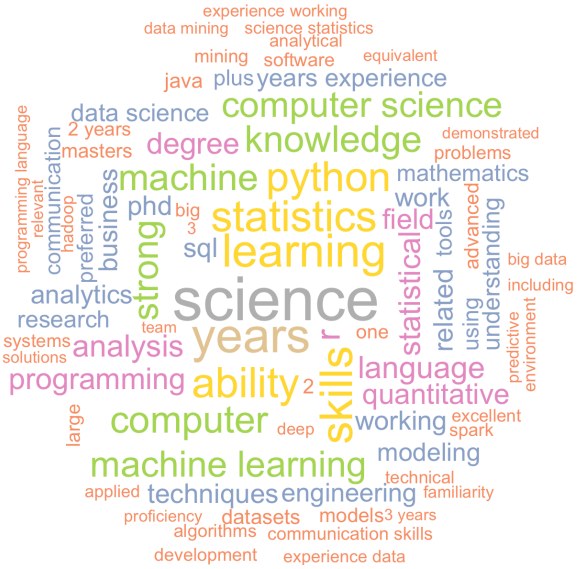
	word	freq
9	python	82
10	computer	73
11	knowledge	71
12	machine	71
13	strong	69
14	machine learning	69
15	computer science	68
16	r	64

	word	freq
17	analysis	56
18	language	56
19	statistical	55
20	programming	54
21	degree	54
22	quantitative	51
23	field	50
24	related	47

	word	freq
25	sql	47
26	techniques	47
27	phd	47
28	working	46
29	work	45
30	analytics	45

Finally, I decided to plot the results above using the wordcloud package again (seen in the picture on the left below), and then took out the most common words, “experience” and “data”, from the dataset (as it is obvious that a data scientist must have “experience” with

many “data” tools, those terms are a bit too general and lead us astray from what matters most.) to see if the word cloud would be improved. The second word cloud is below on the right.



As you can see, the second word cloud is much clearer – taking out those two words was very effective. Finally, I cleaned the final dataset one last time to make it ready to be processed in Wordle.net as well (as this tool requires very specific formatting to process data). Please see my R code attached with this document for more explanations. Below is the extensive word cloud generated on Wordle.net, including the *full* dataset (not *just* the top 30 words).

IV. CONCLUSION

To become a data scientist in early 2018, one must have a lot of experience and possess a large array of skills. Python, R, and SQL seem to be the most common languages that employers look for. Candidates must also be proficient in the fields of computer science (including machine learning, modeling/predictive techniques, visualization, engineering, data mining, algorithms, and big data), business, and mathematics (including statistics, and regression). Communication skills are also key. Other tools such as Spark and Hadoop seem to be much appreciated in the industry.[2] Finally, it seems that degrees (especially PhDs and master's degrees) are required most often. The applicant must have expertise, knowledge, and be comfortable with datasets. Let's check again in five or ten years and see how the profession and skills necessary evolved.

[2] **Apache Spark** is an open-source cluster-computing framework. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. **Apache Hadoop** is an open-source software framework used for distributed storage and processing of datasets of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

CODE

```
# IST736 TEXT MINING
# Homework #2
# David Forteguerre

# to import relevant packages
library(XML)
library(tm)
library(wordcloud)
library(ngram)

# To import the doc
docFile   <-   "/Users/davidforteguerre/Google   Drive/2-SCHOOL/3-M.S.   Data
Science/IST736 Text Mining/HW2/PostingsDataScience.txt"
# PLEASE MAKE SURE TO CHANGE THE PATH HERE TO THE DATASET

# to read the data using the scan function
doc <- scan(docFile, character(0),sep = "\n")
doc <- scan(docFile, character(0))
head(doc, 50) # to view

# To vectorize the document, i.e. to convert text into numbers
words.vec <- VectorSource(doc)
words.corpus <- Corpus(words.vec)
words.corpus

# To preprocess the data
words.corpus  <-  tm_map(words.corpus,  content_transformer(tolower))  #  to
lowercase the data
words.corpus   <-   tm_map(words.corpus,   removePunctuation)   #   to   remove
punctuation
words.corpus <- tm_map(words.corpus, removeWords, stopwords("english")) # to
remove stopwords
stopwords("english") # to view the stopwords included in the tm package.
words.corpus <- tm_map(words.corpus, removeWords, c("etc", "e.g.", "eg")) #
to remove extra words I personally chose


# To create a term doc matrix and count the words
tdm <- TermDocumentMatrix(words.corpus,control=list(wordLengths=c(1,Inf))) #
Had to do some research and add control=list(wordLengths=c(1,Inf)) to keep
one-leeter words (e.g. 'R') in the data
tdm # to visualize
m <- as.matrix(tdm)
wordCounts <- rowSums(m)
wordCounts <- sort(wordCounts, decreasing=TRUE)
head(wordCounts, 100) # to view the first 100 most common words
dataset1 <- data.frame(head(wordCounts, 100)) # to create a dataset with the
results
dataset1$names <- rownames(dataset1) # to duplicate the index column into a
real column
rownames(dataset1) <- NULL # to reindex the data with numbers
dataset1 <- dataset1[c(2,1)] # to reorder columns using column index
names(dataset1)[1]<-"word" # to rename the first column
names(dataset1)[2]<-"freq" # to rename the second column
View(dataset1)
```

```
# To visualize the top word frequencies with a WordCloud in R
cloudFrame <- data.frame(word = names(wordCounts), freq=wordCounts) # to
create a dataframe with the words and frequencies
wordcloud(cloudFrame$word, cloudFrame$freq, min.freq=15, max.words=100,
rot.per=0.35, colors=brewer.pal(8, "Dark2")) # to view
# display.brewer.all() # to view which colors to choose from




# LET'S NOW LOOK AT N-GRAMS!

# https://cran.r-project.org/web/packages/ngram/vignettes/ngram-guide.pdf #
Read more about the ngram package here

# To make the data ready (the ngram package cannot read data in a "corpus
object" format, which is needed for the tm package)
str <- concatenate(lapply(words.corpus, "[", 1))
string.summary(str)

# To get Bigrams (n-gram with n=2)
ngram2 <- ngram(str, n=2)
ngram2
dataset2 <- data.frame(head(get.phrasetable(ngram2), 50)) # to create a
dataset with the results
dataset2 <- dataset2[,1:2] # to only keep the first two columns
names(dataset2)[1]<-"word" # to rename the first column
names(dataset2)[2]<-"freq" # to rename the second column
View(dataset2)

# To get Trigrams (n-gram with n=3)
ngram3 <- ngram(str, n=3)
ngram3
dataset3 <- data.frame(head(get.phrasetable(ngram3), 50)) # to create a
dataset with the results
dataset3 <- dataset3[,1:2] # to only keep the first two columns
names(dataset3)[1]<-"word" # to rename the first column
names(dataset3)[2]<-"freq" # to rename the second column
View(dataset3)

# To create a final dataset combining all the data
dataset <- data.frame(c(dataset1$word, dataset2$word, dataset3$word),
c(dataset1$freq, dataset2$freq, dataset3$freq))
names(dataset)[1]<-"word" # to rename the first column
names(dataset)[2]<-"freq" # to rename the second column
dataset <- dataset[order(-dataset$freq),] # to sort the data by frequency
(high to low)
rownames(dataset) <- NULL # to reindex the data with numbers
View(dataset)

# # To visualize the final results of word/bigrams/trigrams frequencies with
a WordCloud in R
wordcloud(dataset$word, dataset$freq, min.freq=15, max.words=100,
random.order=FALSE, rot.per=0.2, colors=brewer.pal(8, "Set2"))

# To try removing the first two common words to see if we can get a better
WordCloud
```

```
datasetV2 <- dataset[-(1:2),] # to remove "experience" and "data"
wordcloud(datasetV2$word,   datasetV2$freq,      min.freq=20,   max.words=100,
random.order=FALSE, rot.per=0.2, colors=brewer.pal(8, "Set2"))


# Final WordCloud for Wordle (respecting the format [word]:frequency)
    # Full data
datasetWORDLE1 <- dataset
datasetWORDLE1$word <- paste0(datasetWORDLE1$word, ':') # to add a : after
each word
write.table(datasetWORDLE1, '/Users/davidforteguerre/Google Drive/2-SCHOOL/2-
M.A.   Linguistics/#4   IST736   Yu/HW2/FinalDatasetV1.csv',   row.names=FALSE,
col.names=FALSE) # to export the dataset
    # Full data without the two most common words "experience" and "data"
datasetWORDLE2 <- datasetV2
datasetWORDLE2$word <- paste0(datasetWORDLE2$word, ':') # to add a : after
each word
write.table(datasetWORDLE2, '/Users/davidforteguerre/Google Drive/2-SCHOOL/2-
M.A.   Linguistics/#4   IST736   Yu/HW2/FinalDatasetV2.csv',   row.names=FALSE,
col.names=FALSE) # to export the dataset
# Then open the documents, go to wordle.net, the "Advanced" tab, and copy
paste the data in the first box. Finally, generate the WordCloud.
```