Team 2 - Project 3

Data Science Skills

DATA 607
March 26, 2017

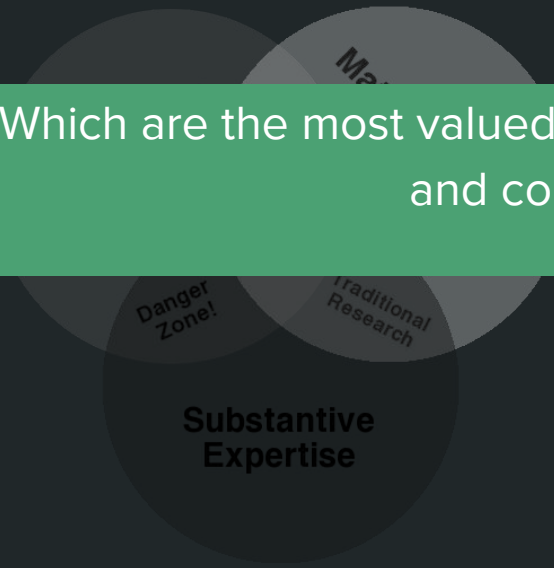
Members

Jaan Bernberg
Cesar Espitia
Georgia Galanopoulos
Michael Muller
Nkasi Nedd
Nnaemezue Obi-Eyisi
Matheesha Thambeliyagoda
Ilya Kats

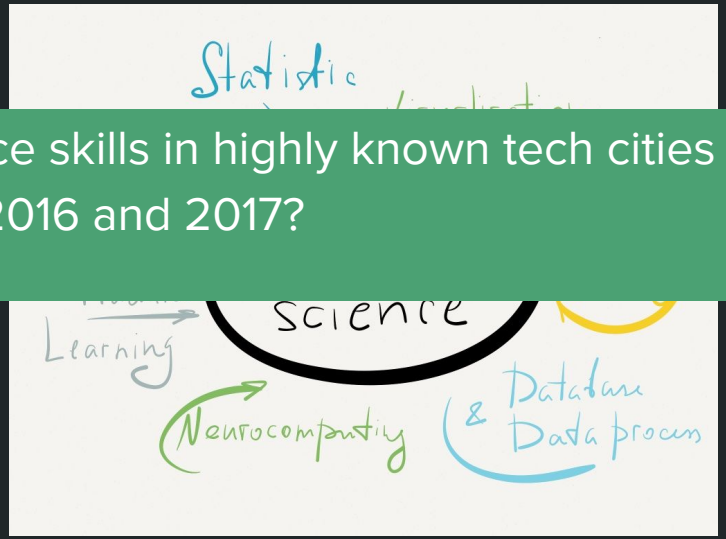
Contents

Problem

Which are the most valued data science skills?



Which are the most valued data science skills in highly known tech cities and countries for 2016 and 2017?



Organization

How We Collaborated

Email was immediately excluded due to team member locations

Communication	Data Collection	Analysis	Presentation Documentation
Daily Conversations slack Formal Updates Cisco webex	Database Google Cloud Platform Webscraping	Processing, Graphs and Figures	Google

Methodology

Workflow

Waterfall approach chosen over other methods due to geographic constraints

Project

Indeed was the source of our job postings for this analysis as it's currently one of the most used job search engines

Vast number of skills led our team to aggregate skills into key groups

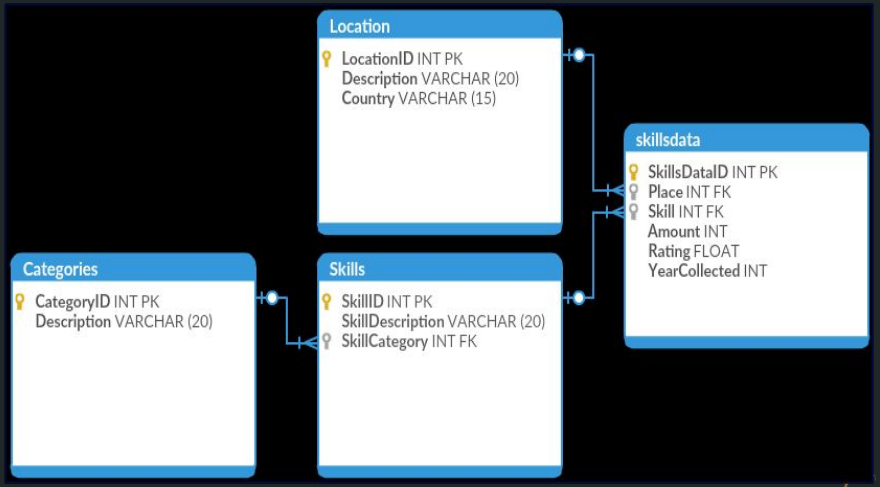
program_languages=['bash','r','python','java','c++','ruby','perl','matlab','javascript','scala','php']

Major US Cities with known tech structures chosen as well as countries that are high tech for 2016 and 2017

Austin, NY, SV, Germany, etc.

Database Structure

- 4 tables created to store different pieces of information
  - *Categories* stored the family core skills (i.e. apple is a fruit -> ruby is a programming skill)
  - *Location* stored where the job posting were located
  - *Skills* stored the detailed skills within a category found on a job description
  - *Skillsdata* is the job aggregation and analysis
- Aggregation of skills for the Categories table was done for easiers analysis



Data Collection

Based upon Yuanyuan Shi's data skills code*

Python script that takes in URLs to scrape data from them

- Opens each listing and searches for keywords
- Returns three variables - skill name, count and rating **(calculated by script)**
- Higher the rating the more often the skill is included in the job requirements

Script limited to first 10 pages with 15 posts per page

** https://github.com/yuanyuanshi/Data_Skills*

Analysis

- All analysis was done using R
- All skills were ranked based upon the occurrence of a particular skill divided by the total number of job postings scraped
- Data from the year 2016 were available only for the cities within the US, so US cities were analyzed separately from countries abroad
- Sample size differences between ‘16 and ‘17 required normalization prior to comparison

```
Result['Ranking'] =
float(len(job_keywords)) /
Result['Count']
```

Results

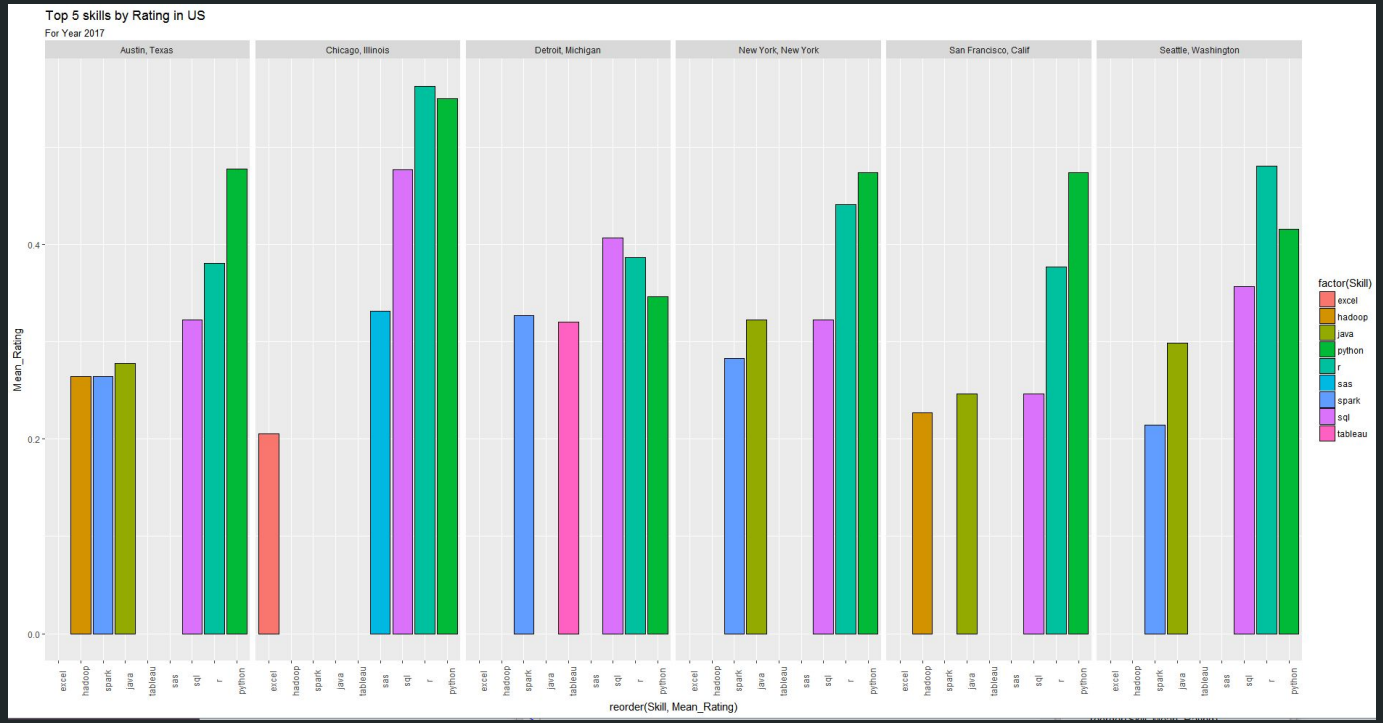
The job market for data science skills is like fashion because it is a sector still in its infancy and ever changing. Can be susceptible to trends (i.e. hadoop/pig)

Results

Skills most sought by US Cities (Year 2017)

Top 3 Skills:

- Python
- R
- SQL

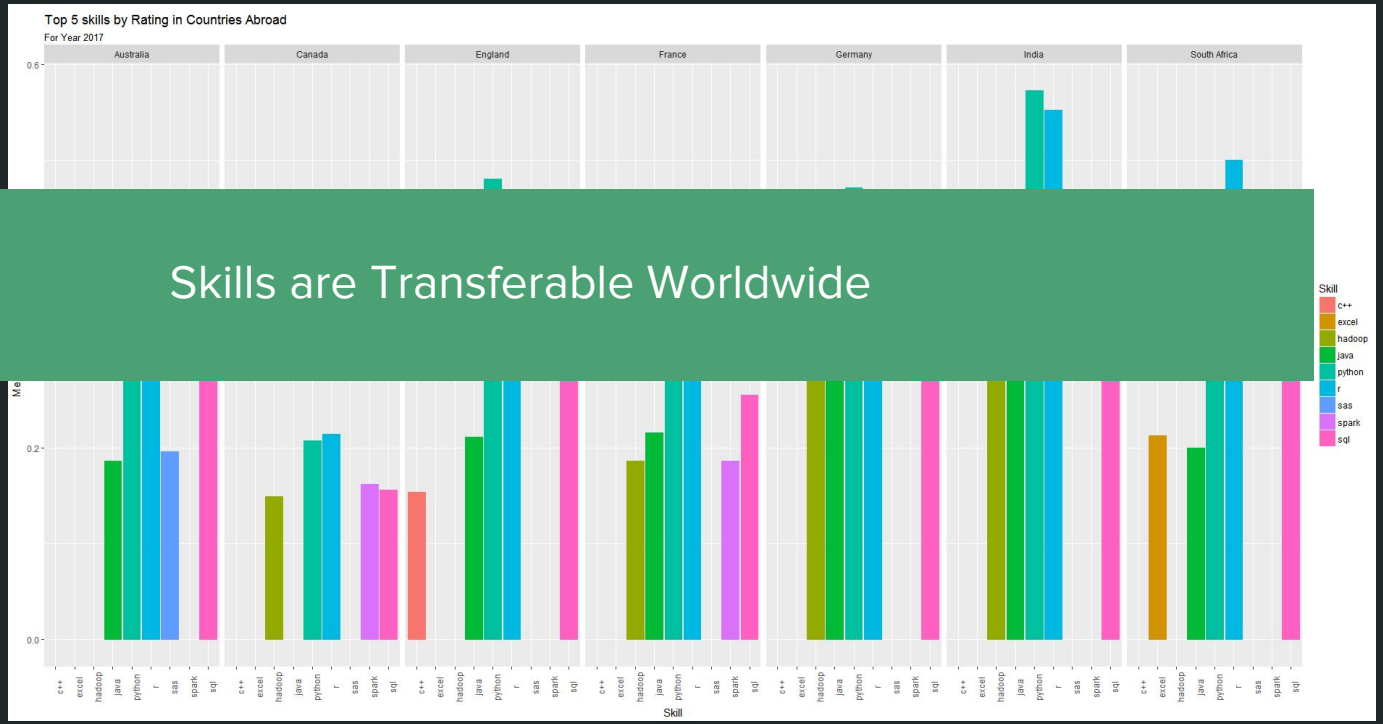


Results

Skills most sought by non-US Countries

Top 3 Skill

- P
- R
- SQL



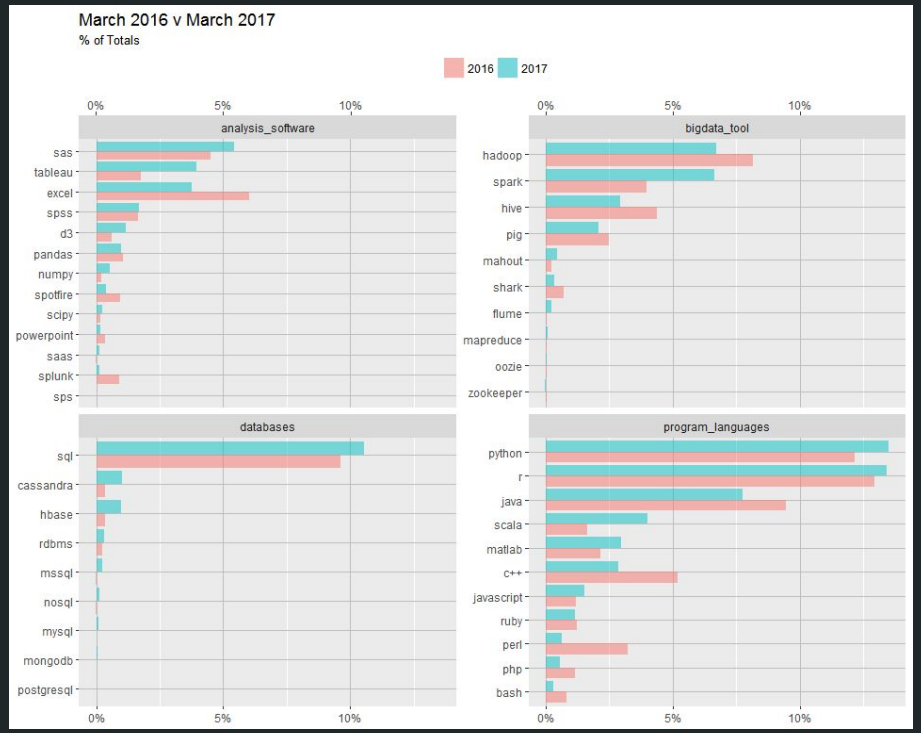
Skills are Transferable Worldwide

Results

Job Market Changes by Skill March 2016 vs. March 2017

Although the job market continues to show preference for the top 3. Each grouping shows variations:

- Big Data - seeking Spark as a new skill
- Analysis - Excel diminishing vs Tableau (marketing hype)
- Databases - SQL flavors dominate
- Programming - Perl and C++ dropped off



Lessons Learned

- Setting up environments (Python)
- Keyword selection
- Slack made it easy to communicate and parse tasks

APPENDIX

Team Roles and Responsibilities

Member	**Main Responsibilities**
Jaan Bernberg	Analysis
Cesar Espitia	Presentation Lead
Georgia Galanopoulos	Analysis
Ilya Kats	Project Lead
Michael Muller	Data Collection
Nkasi Nedd	Data Collection, Database Storage/Structure
Nnaemezue Obi-Eyisi	Database Load, Data Collection
Matheesha Thambeliyagoda	Presentation

Sources Considered

- CrowdFlower 2016 Data Science Report. http://visit.crowdflower.com/rs/416-ZBE-142/images/CrowdFlower_DataScienceReport_2016.pdf.
- Data Scientist Core Skills blog post by Mitchell A. Sanders at Data Science Central. http://www.datasciencecentral.com/profiles/blogs/data-scientist-core-skills
- NYC Jobs data set containing current job postings available on the City of New York's official jobs site. https://data.cityofnewyork.us/City-Government/NYC-Jobs/kpav-sd4t
- Data Skills analysis by Yuanyuan Shi using Indeed data. Contains historical data from April 2016 for several US cities. https://github.com/yuanyuanshi/Data_Skills
- R Package Rlinkedin - a series of functions that allow users to access the 'LinkedIn' API to get information about connections, search for people and jobs, share updates with their network, and create group discussions. https://cran.r-project.org/web/packages/Rlinkedin/
- Web Scraping Indeed for Key Data Science Job Skills by Jesse Steinweg-Woods, Ph.D. https://jessesw.com/Data-Science-Skills/

Code: Schema Creation

```
#Query for the creation of the database (schema) that will be
used for the project
CREATE SCHEMA IF NOT EXISTS datascienceskills;
use datascienceskills;
DROP TABLE IF EXISTS skillsdata;
DROP TABLE IF EXISTS Skills;
DROP TABLE IF EXISTS Categories;Location
DROP TABLE IF EXISTS Location;
#The first table to be created is the categories table.
#It will store information about the categories of data
science skills.
CREATE TABLE Categories
       (CategoryID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
     Description VARCHAR(20) NOT NULL);

#The second table to be creatd is the skills table.
#It will store the data science skills

CREATE TABLE Skills
       (SkillID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
     SkillDescription VARCHAR(15) NOT NULL,
     SkillCategory INT,
     CONSTRAINT FOREIGN KEY (SkillCategory)
             REFERENCES Categories (CategoryID)
             ON DELETE SET NULL);
```

```
#The third table to be created is the location.
#It will store the data about where jobs are found.

CREATE TABLE Location
       (LocationID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
     Description VARCHAR(20) NOT NULL,
     Country VARCHAR (15) NOT NULL);

#The fourth table to be created is the skillsdata.
#It will store the data about skills found.

CREATE TABLE skillsdata
       (SkillsDataID INT AUTO_INCREMENT PRIMARY KEY NOT NULL,
     Place INT,
     Skill INT,
     Amount INT,
     Rating FLOAT,
     CONSTRAINT FOREIGN KEY (Skill)
             REFERENCES Skills (SkillID)
        ON DELETE SET NULL,
        CONSTRAINT FOREIGN KEY (Place)
             REFERENCES Location (LocationID)
        ON DELETE SET NULL);
```

Code: Data Collection 1

```
import re
from nltk.corpus import stopwords
from goose import Goose
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.common.exceptions import TimeoutException
from selenium.webdriver.firefox.firefox_binary import
FirefoxBinary
import time
import requests
import random
import pandas as pd
import matplotlib.pyplot as plt
import os

print os.getcwd()
count = 0
count2= 0
fp = webdriver.FirefoxProfile()
fp.set_preference("http.response.timeout", 5)
fp.set_preference("dom.max_script_run_time", 5)

#I get these keywords from the first page search result of data
scientist at indeed; they're not whole but already tell a story.
program_languages=['visualization','communication',"data
driven","analysis","analytical","visual","statistics","mathematics
","leadership","senior","developer","programmer","firmware","softw
are","solving","critical
thinking","translate","translation","scientific","reasoning","quer
y","mastery","curious","creative","inquisitive","persuasive","comm
unicative","communication","practices","manage","derive","developm
ent","articulate","insight","decision","challenge","diverse","dive
rsity"]
analysis_software=['Bachelor','Master','B.sc','Phd','
MBA','Ph.D','MSc','associates']
bigdata_tool=[]
databases=['Mathematics', 'Statistics', 'Computer
Science','Engineering','Math','Comp Sci','Stats','Physics','Operations
Research','Data Science']
overall_dict = program_languages + analysis_software + bigdata_tool + databases
# the following two functions are for webpage text processing to extract the
skill keywords.
def keywords_extract(url):
    g = Goose()
    article = g.extract(url=url)
    text = article.cleaned_text
    text = re.sub("[^a-zA-Z+3]"," ", text) #get rid of things that aren't words;
3 for d3 and + for c++
    text = text.lower().split()
    stops = set(stopwords.words("english")) #filter out stop words in english
language
    text = [w for w in text if not w in stops]
    text = list(set(text))
    keywords = [str(word) for word in text if word in overall_dict]
    return keywords

#for this function, thanks to this blog:https://jessesw.com/Data-Science-Skills/
def keywords_f(soup_obj):
    for script in soup_obj(["script", "style"]):
        script.extract() # Remove these two elements from the BS4 object
    text = soup_obj.get_text()
    lines = (line.strip() for line in text.splitlines()) # break into line
    chunks = (phrase.strip() for line in lines for phrase in line.split("  ")) #
break multi-headlines into a line each
    text = ''.join(chunk for chunk in chunks if chunk).encode('utf-8') # Get rid
of all blank lines and ends of line
```

Code: Data Collection 2

```
    try:
        text = text.decode('unicode_escape').encode('ascii', 'ignore') # Need
this as some websites aren't formatted
    except:
        return
    text = re.sub("[^a-zA-Z+3]"," ", text)
    text = re.sub(r"([a-z])([A-Z])", r"\1 \2", text) # Fix spacing issue from
merged words
    text = text.lower().split()  # Go to lower case and split them apart
    stop_words = set(stopwords.words("english")) # Filter out any stop words
    text = [w for w in text if not w in stop_words]
    text = list(set(text)) #only care about if a word appears, don't care about
the frequency
    keywords = [str(word) for word in text if word in overall_dict] #if a skill
keyword is found, return it.
    return keywords


base_url = "http://www.indeed.com"
#change the start_url can scrape different cities.
start_url = "https://www.indeed.com/jobs?q=data+scientist&l=San+Francisco%2C+CA"
resp = requests.get(start_url)
start_soup = BeautifulSoup(resp.content)
urls = start_soup.findAll('a',{'rel':'nofollow','target':'_blank'}) #this are the
links of the job posts
urls = [link['href'] for link in urls]
num_found = start_soup.find(id = 'searchCount').string.encode('utf-8').split()
#this returns the total number of results
num_jobs = num_found[-1].split(',')
if len(num_jobs)>=2:
    num_jobs = int(num_jobs[0]) * 1000 + int(num_jobs[1])
else:
    num_jobs = int(num_jobs[0])
num_pages = num_jobs/10 #calculates how many pages needed to do
the scraping
job_keywords=[]
print 'There are %d jobs found and we need to extract %d
pages.'%(num_jobs,num_pages)
print 'extracting first page of job searching results'
# prevent the driver stopping due to the unexpectedAlertBehaviour.
webdriver.DesiredCapabilities.FIREFOX["unexpectedAlertBehaviour"]
= "accept"
get_info = True
driver=webdriver.Firefox(firefox_profile=fp)
# set a page load time limit so that don't have to wait forever if
the links are broken.

#driver.set_page_load_timeout(5)
for i in range(len(urls)):
    count += 1
    print count
    print 'break point a \n \n \n \n '
    get_info = True
    try:
        driver.get(base_url+urls[i])
    except TimeoutException:
        get_info = False
        continue
    j = random.randint(1000,1300)/1000.0
    time.sleep(j) #waits for a random time so that the website
don't consider you as a bot
```

Code: Data Collection 3

```
 if get_info:
        try:
            print count
            print 'berak point b \n \n \n \n'
            soup=BeautifulSoup(driver.page_source)
            print 'extracting %d job keywords...' % i
            single_job = keywords_f(soup)
            print single_job,len(soup)
            print driver.current_url
            job_keywords.append([driver.current_url,single_job])
        Except:   pass
#driver.set_page_load_timeout(35)
for k in range(1,10):
#this 5 pages reopen the browser is to prevent connection refused error.
    if k%5==0:
        print 'BREAKING CONNECTION FOR A SEC \n \n \n \n '
        driver.quit()
        driver=webdriver.Firefox()
        #driver.set_page_load_timeout(35)
    current_url = start_url + "&start=" + str(k*10)
    print 'extracting %d page of job searching results...' % k
    print count
    print 'break point C \n \n \n \n \n'
    resp = requests.get(current_url)
    current_soup = BeautifulSoup(resp.content)
    current_urls = current_soup.findAll('a',{'rel':'nofollow','target':'_blank'})
    current_urls = [link['href'] for link in current_urls]
    print len(current_urls)
 count2 = 0
    for i in range(len(current_urls)):
        get_info = True
        count2+=1
```

```
try:
            driver.get(base_url + current_urls[i])
        except TimeoutException:
            get_info = False
            continue
        j = random.randint(1500,2200)/1000.0
        time.sleep(j) #waits for a random time
        if get_info:
            try:
                soup=BeautifulSoup(driver.page_source)
                print count2
                print 'extracting %d job keywords...' % i
                print count2
                single_job = keywords_f(soup)
                print count2
                print single_job,len(soup)
                print count2
                print driver.current_url
                job_keywords.append([driver.current_url,single_job])
            except:
                pass
# use driver.quit() not driver.close() can get rid of the openning too many files
error.
driver.quit()
skills_dict = [w[1] for w in job_keywords]
dict={}
for words in skills_dict:
    for word in words:
        if not word in dict:
            dict[word]=1
        else:
            dict[word]+=1
Result = pd.DataFrame()
Result['Skill'] = dict.keys()
Result['Count'] = dict.values()
Result['Ranking'] = Result['Count']/float(len(job_keywords))

Result.to_csv('CA_SoftSkills_2017.csv',index=False)
```