

W205 Final Project report

“Redefining the Job Search Process”

Saad Padela, Juanjo Carin & Lucas Dan

Problem Statement:

Recently, companies have started to prioritize hiring Data Scientists for a wide variety of projects and overarching job responsibilities. However, due to the relatively ambiguous and vague state of Data Science in the current job market, these roles can vary so much that projects tend to span three or more disciplines. In addition to this high variance in project scope, toolkits and programming language can also vary. For example, some companies use Python and Tableau whereas other companies prefer R and Spotfire. Some companies utilize graphical databases while others use relational databases. While it is extremely difficult to consider every single one of these factors when applying to a job, these subtle differences can actually play a critical role in determining the ultimate career path/specialization for a Data Scientist. Similarly, the ways in which a company applies Data Science techniques to its products are greatly determined by the Data Scientists that company chooses to hire. Of course, mismatches cause mutual unhappiness.

Since the current process for hiring a Data Scientist relies heavily upon technical recruiters who are well tapped into the field of Data Science (and able to distinguish between subtleties), we see this process in general as an area for improvement. Our mission is to address this problem by providing prospective Data Scientists with an interactive query and visualization tool. Users can use this tool not only query available Data Scientist jobs by geographical locations, but also to extract deeper insight and summarize large amounts of information through analytics and visualizations that automatically update based on user-generated queries/filters. Our application focuses on two data sources, and the complete description of our design can be found below.

Data Acquisition and Storage:

Our first data source is United States Census survey data, based on a survey conducted annually from 2010 – 2014. This data is extremely rich data contains 66,400 variables (including transformations), and is available by State, County, and Metropolitan Statistical Area (MSA). This allowed us to analyze the data at multiple

geographical levels and choose the most appropriate grouping. Ultimately we decided to use county-level data for the entire United States (i.e. 3,200+ counties) because we felt it was the best way to help users choose between different areas to apply for Data Scientist jobs. Within the survey data, we focused on the total population, as well as those between ages 20 – 34 because that is the target population for our application. The survey data shows jobs for employed civilians ages 16 and older, and we focused on the fields of Information Technology, Research, Finance, Education, Public Administration, and Retail. The data also shows cost of living; concretely speaking, we extracted monthly owner costs of housing units with a mortgage (median dollars), and gross rent of occupied units (median dollars). Specific fields include jobs_retail, jobs_finance, housing_cost_rent, housing_cost_own, pop_top, etc. Our second data source is the Indeed API, which is a popular employment API that allows us to interact with frequently updated data related to job postings. Using this API, we can extract relevant information related to job postings and employers grouped by geographic location, and merge this with our primary US Census data source. Critically important Indeed fields include jobtitle, county, and other geographical location (we focused on jobtitle of Data Scientist). By merging these two data sources, we will ultimately provide more comprehensive analytics and visualizations to assist the user in his or her job search.

The ultimate goal of this project is to help users find out where the most job opportunities exist for a given role (we have focused on data scientist positions), taking information from Indeed, and combine that information with a second data source (Census) to select a location not only based on the number of job opportunities but also on other aspects (we've chosen the average housing cost -- mortgage or rent -- based on the median for each county, but many other aspects could have been considered, thanks to the level of detail of the Census data; some of these other aspects are shown in some Tableau visualizations available at <http://juanjocarín.github.io/w205-viz/>).

W205 Final Project

By Juanjo Carin, Lucas Dan, and Saad Padela

[View project on GitHub](#)

Visualizations from our W205 final project

The following visualizations show insights from the Census data retrieved in our project. Their purpose is to help users select best locations when considering a new position, by the number of opportunities in each location (on a county level), the cost of living there, and the expected competition (of course, we have used proxies for all that variables). They allow the user to select one or more of the following parameters:

- the State ,
- the industry (IT, Finance, ... or all),
- the type of housing (own or rented),
- if the whole population has to be considered, or only those between 20 and 34 (the age range where there are more people interested in changing jobs or getting one).

The first visualization shows 2 maps by county. Changing the States to be shown automatically adapts the zoom level. The first map shows the number of jobs for a certain industry in a given county, per 1,000 people, and hence is a proxy of the number of opportunities in that industry. The user can also select if he or she wants to consider only young people (between 20 and 34), because that age range will be, in most cases, the likely competition. Counties are colored by number of jobs, from light red to light green, so green (which corresponds to higher values) is desirable.


The second map shows the housing cost in each county, filtered by mortgage or rent (in both cases the value represents the median amount in dollars for each county). This is an important detail when considering to accept a job offer/move to a new location. In this map the color range is reversed, so darker tones of green correspond to lower housing costs. This way, user can focus on green areas in both maps.

[Download .zip file](#)

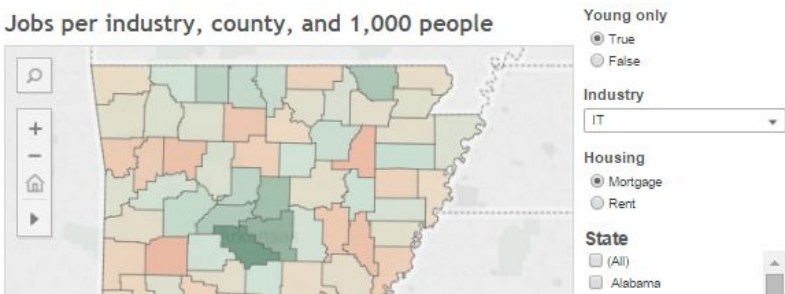
[Download .tar.gz file](#)

is maintained by [juanjocarín](#).

This page was generated by [GitHub Pages](#) using the [Architect](#) theme by [Jason Long](#).



Jobs per industry, county, and 1,000 people



The solution we propose could be further enhanced with many other details, such as letting the user consider other jobs, use more generic query terms, or results based on many other aspects. We have preferred to focus on the scope of this course, trying (with combined data sources) as many technologies as possible (Hadoop, Hive, and, Spark, all used in AWS environment).

Once establishing the problem we want to address and the data sources we will use to address this problem, we had to develop/configure the AWS environment with the necessary data science toolkit to achieve our goals. Of course we knew that python would be a necessary technology to leverage because there is an external python library (python-indeed) which allows us to interact with the Indeed API. In addition, we knew we could also leverage python to acquire the Census data, so we decided to write .py scripts (in the api directory of our github repository). These python scripts not only retrieve county-level Census and Indeed data regarding job postings and cost of living, but will also parse the data into text files that we will be able to load into a database.

After our .py scripts finish acquiring, parsing, and writing both data sources to text files, we use Bash shell scripting/Hadoop to create the necessary HDFS directories and move our data (using the put command and user w205) so that we will be able to load the data into a Hive database. Following this file movement, we use Bash to create our Hive database and a QA step to ensure that the database has been successfully created. Once we know this database has been successfully created, we are now ready to create specific schemas and load in our files to separate tables in our Hive database. Under the hive directory in our GitHub repository, there are four .sql scripts. The first two .sql scripts (census.sql and indeed.sql) are the ones we ended up actually using, as they will create the necessary schemas and load our text files into their respective tables in our Hive database. Note that we store the raw data as external tables. The other .sql script (census_ranks.sql) creates a new table (Staging Table in ORC format) with its own schema that performs back-end calculations to ultimately add metrics to the census data related to cost of living and available jobs in Finance/IT fields for young professionals. By utilizing SQL techniques in our schema (e.g. partitioning) we were able to optimize efficiency and vastly improve performance in our data pipeline. Our fourth .sql script (top_counties.sql) creates another schema that also utilizes back-end calculations and case when statements to optimize performance and ultimately provide another Staging Table (ORC format) that ranks each geographic location based on our outcome metrics (e.g. proportion of available jobs in user-specified industry for younger population). The final two scripts did not end up in our final application, but we wanted to include the scripts to show our work. It is important to note that the census text file has 3,220 records (as many as counties; all of which are loaded in Hadoop, as a text file as well as a Hive table), whereas the indeed text file (which of course changes with every execution) has about 3,000 records (the number of job postings is much lower, about 200, but many of them apply to neighbor counties), but once we aggregate them by county we are left with roughly 400 records (because there are no job opportunities for data scientists in many of the 3,220 counties). This will also be the number of records after joining with the census text file, which we filter to show only the top 20

values. Below there is a screenshot which show some counts from Indeed, for a given execution.

```
In [2]: indeed = sc.textFile('/HD/indeed_county.txt')
```

```
In [8]: print str(indeed.count())+'\n'
for x in indeed.take(5):
    for y in x.split('\t'):
        print y
    print
```

```
3053
```

```
1e6ff7c0a7700c2c
Data Scientist
Competent Systems, Inc
<b>Data</b> <b>Scientist</b> Birmingham, AL 6 Months Job Description:. They're going to be pulling <b>data</b> from v
arious marketing tools to make a singular tool to create a...
http://www.indeed.com/viewjob?jk=1e6ff7c0a7700c2c&qd=hCWQTHsi6RQ-F-nUjAy1PxorCELH72kgoakZk-o2op4vv9LH9eDbwcHIzC62Iu50
rqrU-I30xZbVdwSfm100oGcQBvodRHS7La7lk541b0soon7JYpMhk9QEPqAHZA8U&indpubnum=8924341972846274&atk=1a6l4r3cib8t9apj
Wed, 16 Dec 2015 00:21:58 GMT
Birmingham
Cullman County
Alabama
33.516483
-86.80769
```

```
df98646306bce8bb
E-commerce Data Scientist
SelectBlinds.com
SQL Ninja with experience extracting and analyzing <b>data</b> from web and mobile analytics collection hubs like Goo
gle Analytics ( *Previous Experience with Google...
http://www.indeed.com/viewjob?jk=df98646306bce8bb&qd=hCWQTHsi6RQ-F-nUjAy1PxorCELH72kgoakZk-o2op6xU3JTPlaB0pxegMN5D5Qf
hNLhnMB1qb-1ZkcUWC0k53Psmj3hfr5rFVlnlKoHNqmw_2FZ4z8yo7ESNtwM9PpNVbuGkvXR_7b87sG_geiaw&indpubnum=8924341972846274&atk
=1a6l4rlfrb8t98d1
Mon, 14 Dec 2015 23:31:26 GMT
Mesa
Maricopa County
Arizona
33.4011
-111.78571
```

```
In [10]: county_job = indeed.\
    map(lambda line: line.split('\t')).\
    map(lambda x: [x[7]+'', '+x[8],1]).\
    reduceByKey(lambda a, b: a + b).\
    cache()
print county_job.count()
print
print county_job.take(1)
```

```
429
```

```
[(u'Essex County, New Jersey', 5)]
```

Results & Scaling Considerations:

Once all up-to-date data is fully loaded in Hadoop HDFS, we are ready to produce the final deliverables/application. We serve the data from Production Tables to Tableau using hiveserver2. Users will be able to extract insights and meaning from the Census data source using Tableau. Note that a separate python script will run pyspark to output analytics based on the Indeed job postings data source (complete instructions for running the application can be found at the bottom of this document and in the Readme section of our GitHub repository). While acquiring and storing the data sources efficiently is of course a foundational component of our project design, we ultimately

chose to address our problem statement by providing visualizations and analytics to assist a user who is searching for a Data Scientist job. Please view our website which provides these analytic tools (<http://juanjocarín.github.io/w205-viz/>). The outcome metrics and associated visualizations that we provide to the user include jobs per industry/county/thousand individuals, housing cost per county, best jobs per state by jobs per 1,000 individuals, and distribution of jobs per 10,000 individuals/housing cost as well as % of young individuals by state.

In addition to our visualization tools, we also provide additional analytics related to Data Scientist jobs available by geographical location. Users can access these analytics when running the python script results.py. This script utilizes pyspark by initializing a sparkContext object and reading in our Indeed and Census text files to output a list of the top counties in terms of Data Scientist job availability. This script also utilizes key course concepts such as MapReduce. Below is an example screenshot of the output of this program:

County	Job offers	Average Housing Cost (Mortgage+Rent)
San Francisco County, California	73	97027
Alameda County, California	72	182824
Contra Costa County, California	71	121130
San Mateo County, California	71	88973
Charles County, Maryland	45	22023
Montgomery County, Maryland	45	146063
Prince George's County, Maryland	44	136650
Howard County, Maryland	44	47603
District of Columbia, District of Columbia	43	90609
Prince William County, Virginia	43	53940
Arlington County, Virginia	43	34196
New York County, New York	43	206016
Richmond County, New York	42	68442
Anne Arundel County, Maryland	42	74981
Hudson County, New Jersey	42	68929
Fairfax County, Virginia	42	141078
Santa Clara County, California	42	176415
Essex County, New Jersey	42	100008
Rockland County, New York	42	48459
Loudoun County, Virginia	42	37671

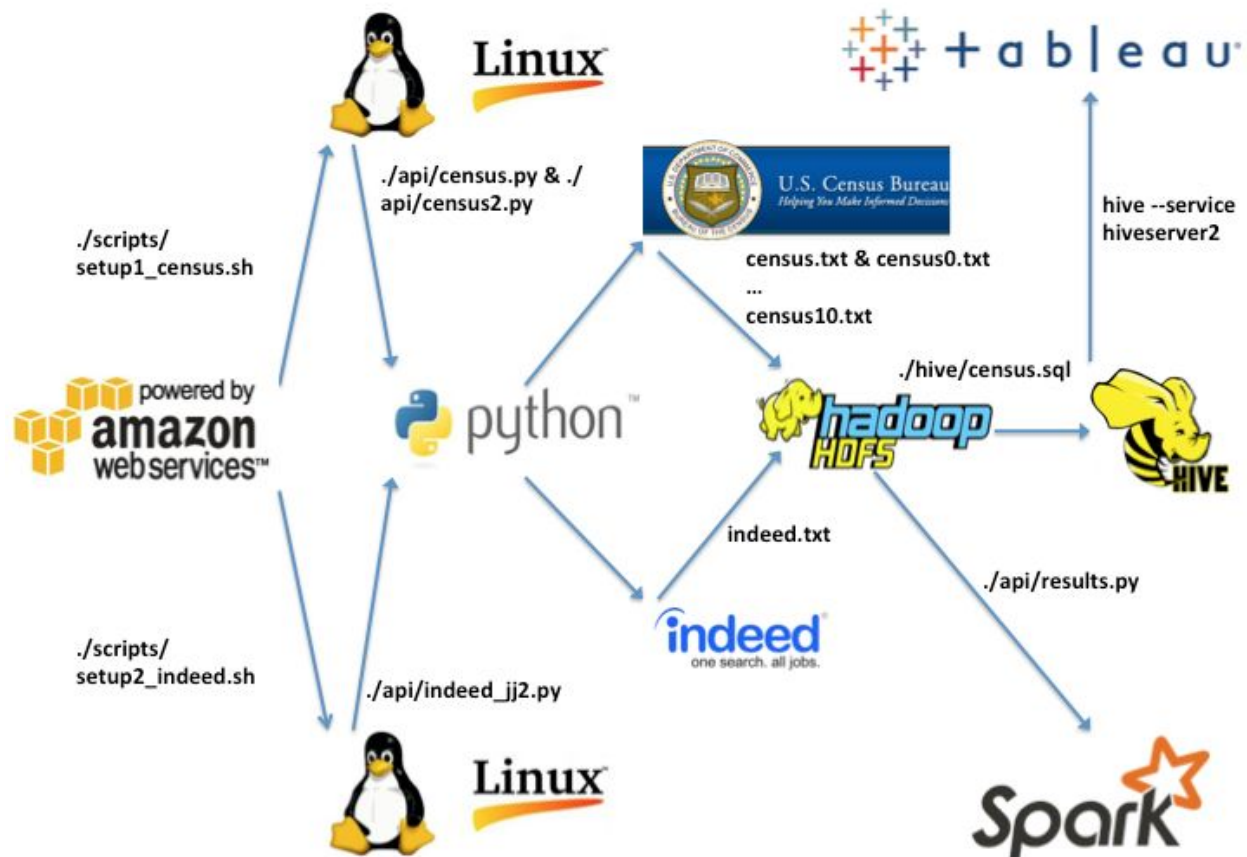
Using this information coupled with the visualizations described in the previous paragraph, users can make an informed decision about where to move/apply for Data Scientist jobs. Of course, certain factors are more important to some people than others, so users are able to focus on their priority factors, quickly digest all of this information related to average housing cost and job offers, and hopefully the user will experience a more seamless and informed decision-making process. For example, a single glance at the image above (which shows the result of running all this code Wednesday, 12/16/2015 5PM PST) lets the user know that most job opportunities are in the States of California, Maryland, DC, Virginia, New York... (in that order). Besides, by

combining information from the Census, the user can make an informed decision about where to move: for example, focusing on Top positions, all in CA State, San Francisco County not only has more opportunities than any other (at least for the last 30 days and this very specific position), but is also much cheaper than nearby counties with almost as many opportunities; similarly, moving to San Mateo County, though in the 4th position, seems more appropriate than to Alameda or Contra Costa Counties, which have the same or 1 more job opportunity, but are much more expensive. A similar analysis could be applied, for example, to New York, Richmond, and Rockland Counties, in NY State.

In terms of scaling considerations, we have designed our architecture in such a way that metrics are calculated based on a “point in time,” so the database will refresh each time the data is extracted from the APIs. However, since the Census data is only updated once per year, there is no need to stream data from this data source. Meanwhile, the Indeed API has many restrictions that make it very difficult to stream hourly or daily job postings without designing the API calls very carefully. The way in which our architecture is designed, the only way we would be able to scale without disobeying the API restrictions is by increasing either the time span of job postings or the radius in miles. Since neither of these expansions add value to our application, and it wouldn't increase the “size” of the data regardless, we found this to be unnecessary.

Overall Architecture & Implementation Details/Instructions:

Please see the diagram below which visualizes the Architecture for our application:



Please see complete documentation and instructions for running our application below:

Instructions

All the scripts (and data) in **this repository** are already cloned in the EBS of the **AMI instance**:

W205-juanjo_lucas_saad-final (public; search by its name of by **Owner: 298522642522**)

The screenshot shows the AWS Management Console interface for an Amazon Machine Image (AMI). At the top, there are buttons for 'Launch' and 'Actions'. Below this is a search bar with 'Public images' selected and a filter for 'Owner: 298522642522'. A table lists the AMI with columns: Name, AMI Name, AMI ID, Source, Owner, Visibility, Status, and Creation Date. The AMI 'W205-juanjo_lucas_saad-final' with ID 'ami-ec85d086' is highlighted. Below the table, the 'Details' tab is active, showing a list of attributes for the AMI.

Attribute	Value
AMI ID	ami-ec85d086
Owner	298522642522
Status	available
Creation date	December 16, 2015 at 6:31:22 PM UTC-8
Architecture	x86_64
Virtualization type	paravirtual
Root Device Name	/dev/sda1
RAM disk ID	-
Product Codes	-
AMI Name	W205-juanjo_lucas_saad-final
Source	298522642522/W205-juanjo_lucas_saad-final
State Reason	-
Platform	Other Linux
Image Type	machine
Description	MIDS W205 Final Project Carin,Dan,Padela
Root Device Type	ebs
Kernel ID	aki-919dcaf8
Block Devices	/dev/sda1=snap-e76599a7:10:true:gp2, /dev/sdb=snap-4f525b4d:50:false:gp2

The scripts (only the final versions; all of them, including some preliminary versions, are in the corresponding folders of this repository) can also be found in the [CODE.md file](#).

The instance also contains some of the data we stored (because the EBS those data are in is persistent). This way it is not mandatory to run all parts of the .sh scripts: you could comment some of their lines, and the others (for example, the one that runs the Spark code to present results) will still work (though with non-updated results).

We've also built a **website** with the most relevant visualizations from **Tableau** (data extracted in Python from the **CensusAPI**, uploaded to **Hadoop**, converted into a **Hive** table, and accessed by Tableau via a Hive server): <http://juanjocarín.github.io/w205-viz/>

Census data

All the variables accessible from the **Census API** (for each county in the U.S.) are described [here](#). The variables we've retrieved (apart from the county and the state) are:

- Jobs -- Civilian employed population 16 years and over in:
 - Retail trade (jobs_retail)

- Information (jobs_it)
- Finance and insurance, and real estate and rental and leasing (jobs_finance)
- Professional, scientific, and management, and administrative and waste management services (jobs_research)
- Educational services, and health care and social assistance (jobs_education)
- Public administration (jobs_public)
- Monthly costs:
 - Selected Monthly Owner Costs (SMOC) of housing units with a mortgage: median (dollars) (housing_cost_own)
 - Gross rent of occupied units (paying rent): median (dollars) (housing_cost_rent)
- Population:
 - Total (pop_tot)
 - 20 to 24 years (pop_20_24)
 - 25 to 34 years (pop_25_34)

After launching that **instance** (remember to select at least t3.medium, and to set ports 22 for SSH, and 50070, 10000, 4040, 8080, 8088, and 8020 for TCP):

1. Go to `cd /data/w205/W205_final` (where the repository was cloned and the scripts are located)
2. Go to `cd scripts` (where the .sh files scripts to replicate the project are)
3. **Run `./setup1_census.sh`**

This script:

- launches a **Python** program (`./api/census.py`) that retrieves data from the Census API and stores them in another folder in the EBS (`/data/w205/W205_final_storage/census/txt`), in some text files
 - these results will be used to construct Hive tables, to be used to build some exploratory visualizations in Tableau

- launches another **Python** program (`./api/census2.py`, that numpy and pandas -- and anaconda) that generates a single text file
 - this was the one finally used to combine results with the other data source, Indeed
- after that, the script moves those data (from both Python files) to **HDFS**, and
- then an SQL file (`./hive/census.sql`) queries the data and creates a **Hive** table (changing the schema, grouping population 20 to 34 years, which is more amenable to be interested in jobs),
 - We created a more complex query that creates several rankings among counties and states, using windows.
- that was accessed by Tableau (running `hive --service hiveserver2` in the folder where the table was stored: `/data/w205/W205_final_storage/census/hive`) to generate the visualizations.

Indeed data (and results)

To replicate this part of the project, run `./setup2_indeed.sh` (again, in `/data/w205/W205_final/scripts`). This script:

- launches a **Python** program (`./api/indeed_jj2.py`) that retrieves data from the Indeed API and stores them in another folder in the EBS (`/data/w205/W205_final_storage/indeed/txt`), in a single text file (`indeed.txt`)
- After that, the script moves those data to **HDFS**, and
- then another **Python** program (that calls **Spark**), `./api/results.py`):
 - extracts the data from the Indeed text file, getting counties (with States; this point is very important since many counties in different States have the same name, so including the latter is the only way to differentiate them) as keys and the number of job posts in that county (or a neighbor one) as values,
 - extracts the data from the Census text file (the one generated by `census2.py`), getting counties as keys and the average of the housing cost (mortgage + rent) as values,

- joins both RDDs (a left join, since not all counties appear in the Indeed text file, only those with job opportunities for data scientists) and order results by (descending) number of job posts, showing the Top 20.

Extracting data from Indeed

The most important part of this stage are the 2 Python scripts. The API from Indeed presented several changes, such as:

- The number of posts is limited to 25 by request. Luckily any request also contains information about the total number of posts returned by the query, so all we had to do was to make subsequent requests to the API, to get all the posts.
- Though the query terms are supposed to be connected by AND, that was not the case (e.g, a "data scientist" query returns jobs with both words or any of them), so we had to filter the results by looking for exact matches.
- Though Indeed accepts queries based on counties, it does not report those results, so we had to include the county we had used in the query.
- etc.

We retrieved the following parameters from all job posts, those who are of interest to any potential user (some of them may not be included in some posts):

- jobkey: a unique ID.
- jobtitle: the position demanded. We queried "**data scientist**," a future enhancement could be to broaden the terms, to include synonyms, but for the scope of this project we preferred to focus on a term that applies to us and is still not broadly used.
- company
- snippet: the description of the job.
- url
- date

- city
- county: as mentioned, we took this one not from the API itself but from the query we sent to it.
- state
- latitude
- longitude

The parameters for the query, apart from the term "data scientist" and the county, were the number of days back to search (set to 30) and the radius or distance from search location (set to 50 miles, which we thought was a reasonable commute).

The counties are taken from the Census API (some transformations were required because Indeed uses abbreviations). For each one of them, a first query informs of the total number of job posts in that area (some results may correspond to companies in another county, as long as the distance does not exceed the selected radius). Then, supposed N job posts are found, it is necessary to send N/25 requests to retrieve all data. The results are then filtered, discarding:

- discarded offers
- job titles that contain "data" or "scientist," but not both
- duplicated offers (on a county basis; as mentioned, the same offer might be retrieved again for a nearby county)