Data Acquisition Project

Kailey Hoo & Griffin Okamoto

10/10/2014

Problem Statement

Career choices are among the most important decisions we make in a lifetime. The decision process to select what job to pursue next is often guided by intuition or anecdotal evidence. We want to be able to model someone?s career trajectory based on their work history. Many people have a few jobs they are considering pursuing. If they were able to see what subsequent jobs each of those would lead to in the future, what salary they could expect, and how long it would likely spend in each position, they could make a more informed decision.

Executive Summary

This report summarizes the challenges of acquiring massive amounts of work history data from all of the public profiles on the LinkedIn social network. In addition, we will briefly describe the process of obtaining salary data from the Simply Hired job site to supplement our findings. These two data sources combined would serve as our training data in developing a predictive algorithm to forecast future job prospects. After encountering LinkedIn API restrictions, we switched to general web scraping using Python?s Beautiful Soup and urllib2 libraries. After overcoming complexity in the structure of LinkedIn?s member directory, we were able to scrape 50,000 profiles and obtain job histories.

Recommendations

With the computing resources we had available, we were only able to scrape a small subset of the LinkedIn public profiles. In order to have enough memory and perform the scraping in a reasonable amount of time, more machines would be needed to complete this project. In the meantime, it may be possible to begin developing the predictive model based upon what we have already gathered. For this to happen, we also need to merge the LinkedIn and Simply Hired data so that each job in LinkedIn users? work history will have an associated estimated salary. Lastly, further data filtering is required to target our population of interest: workers in the US.

Data Acquisition Experiments

We initially set out to use the LinkedIn API to access user profiles and get their job information through connections. However, the LinkedIn API was very restrictive and limited the amount of information that we could gather. Instead, we decided to use BeautifulSoup and urllib2 to scrape the public profiles. Using Python, we started in the member directory, went through all of the sub-directories of each of the names, until we got to a user's profile. Then the process required us to go through a user's profile and find the appropriate tables with the information that we wanted. Because we wanted to gather as much information as possible, we grabbed a user's name, first and last, and all information about their jobs. This included the job titles, companies, industries, start and end dates as well as the number of jobs that they had had.

We soon came to realize that we were including users that had no career trajectory to use in our model. After updating our code to exclude these users from the dataset, we were much more satisfied with the end result. Once we gathered all the information we needed, we saved the files as JSON files in order to make them easier to open and read.

Our next step was grabbing the salary data for job titles from Simply Hired. This process was very similar to LinkedIn, where we went into the job directory, grabbed all the job titles, and queried the salary estimator on the website with each of those titles to get the average salary amount.

Data Challenges

When we decided to use the LinkedIn Member Directory, our first challenge was to figure out how to scrape the profile URLs by traversing the tree of pages. Some links on each directory page would lead to profiles and some would lead to a subdirectory within that directory. In addition, for names that corresponded to more than one user, the link would lead to a differently formatted page to scrape. So we had to distinguish between the two in order to know what information we wanted to scrape. Lastly, there were links that would not work because the profile no longer existed or the URL had non-ASCII characters that urllib2 could not parse. These URL-related issues were programming obstacles, but not data quality problems.

In contrast, when parsing each profile, we had to consider the availability of information on the profile. Many people had missing work experience, location, company, time frame worked, or industry. Some of these fields are considered vital for our model, but eliminating an entire profile because of empty fields leads to a loss of information. There were also people with only one job listed, which is not entirely helpful when we need information about how different jobs lead to one another. Deciding how to handle records with missing fields was important in ensuring high-quality data to eventually use to train our model.

Lastly, the volume of data from LinkedIn was a concern because of the time and memory requirements to store all of the data. With upwards of 250 million profiles, LinkedIn?s vast amounts of data provides a wealth of valuable information, but it is difficult to manage. We could choose to keep all of the profiles possible, or perhaps scale down by some metric for more manageable data.

Solution Approaches

The handle URL differentiation, we looked at the URL structure for different types of pages (profile, directory, repeated name directory) and wrote regular expressions to classify them. Once we knew what type of page the URL was for, we would parse it with various functions we wrote to handle each case separately. We dove deeper into directories, scrape job history from profiles, and obtain repeated name URLs. If a URL couldn?t be reached, it was skipped over. Our modular approach made our mass scraping more robust and flexible.

We ultimately decided to eliminate profiles that did not have all of the necessary information. This included people with less than two jobs, missing time frames, missing companies, or missing industries.

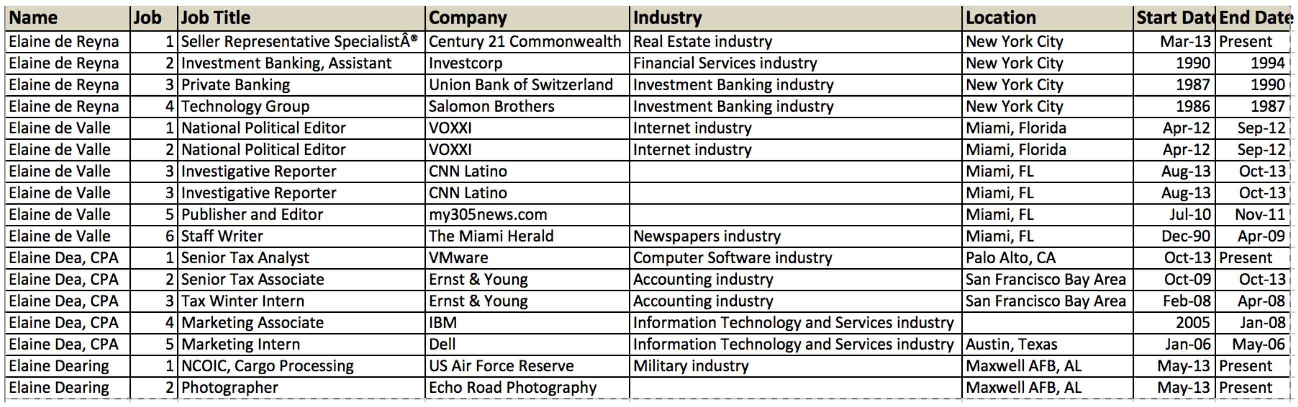
The massive amount of data we still retained seems more than sufficient to train our model. With this data reduction, we were able to process more of the member directory. However, we still only scraped a subset of the data because memory and time. Scraping over the course of the week, we were able to acquire URLs and corresponding profile data for one of several primary member directories.

Assumptions & Limitations

Because we are using BeautifulSoup and urllib2 to scrape the pages, our first assumption is that the page sources of both LinkedIn and Simply Hired stay the same. We are also making many assumptions that the information people put on their profile is accurate and consistent. Lastly, through our filtering of profiles that can't be accessed for one reason or another, we hope that we are not missing important information. With the large amounts of information we are gathering from LinkedIn's million+ profiles, we have limitations on the duration of how long it takes to get all of the information. We had to put a wait time in our code in order to avoid detection which caused the queries to take days to run, even on just a couple thousand profiles. Because the dataset was incredibly large, we were not able to go through every profile and have not combined our two datasets to create our model. Additionally, once we are able to canonicalize the job titles in LinkedIn, and match them to the job and average salary information from Simply Hired, we will be limited in the depth of the job titles.

Results Summary

We have obtained work history from a large number of LinkedIn profiles using web scraping in Python, but the majority of profiles still remained unscraped. The data we have acquired is relatively clean, but some processing remains. This includes connecting our data from our two sources: LinkedIn and Simply Hired. Expanded computing capabilities will allow us to finish scraping the necessary data and build a model that will help people make career decisions.



Name	Job	Job Title	Company	Industry	Location	Start Dat	End Date
Elaine de Reyna	1	Seller Representative SpecialistÂ®	Century 21 Commonwealth	Real Estate industry	New York City	Mar-13	Present
Elaine de Reyna	2	Investment Banking, Assistant	Investcorp	Financial Services industry	New York City	1990	1994
Elaine de Reyna	3	Private Banking	Union Bank of Switzerland	Investment Banking industry	New York City	1987	1990
Elaine de Reyna	4	Technology Group	Salomon Brothers	Investment Banking industry	New York City	1986	1987
Elaine de Valle	1	National Political Editor	VOXXI	Internet industry	Miami, Florida	Apr-12	Sep-12
Elaine de Valle	2	National Political Editor	VOXXI	Internet industry	Miami, Florida	Apr-12	Sep-12
Elaine de Valle	3	Investigative Reporter	CNN Latino		Miami, FL	Aug-13	Oct-13
Elaine de Valle	3	Investigative Reporter	CNN Latino		Miami, FL	Aug-13	Oct-13
Elaine de Valle	5	Publisher and Editor	my305news.com		Miami, FL	Jul-10	Nov-11
Elaine de Valle	6	Staff Writer	The Miami Herald	Newspapers industry	Miami, FL	Dec-90	Apr-09
Elaine Dea, CPA	1	Senior Tax Analyst	VMware	Computer Software industry	Palo Alto, CA	Oct-13	Present
Elaine Dea, CPA	2	Senior Tax Associate	Ernst & Young	Accounting industry	San Francisco Bay Area	Oct-09	Oct-13
Elaine Dea, CPA	3	Tax Winter Intern	Ernst & Young	Accounting industry	San Francisco Bay Area	Feb-08	Apr-08
Elaine Dea, CPA	4	Marketing Associate	IBM	Information Technology and Services industry		2005	Jan-08
Elaine Dea, CPA	5	Marketing Intern	Dell	Information Technology and Services industry	Austin, Texas	Jan-06	May-06
Elaine Dearing	1	NCOIC, Cargo Processing	US Air Force Reserve	Military industry	Maxwell AFB, AL	May-13	Present
Elaine Dearing	2	Photographer	Echo Road Photography		Maxwell AFB, AL	May-13	Present