Binoy Dutt ([**binoyshekhar.dutt@mavs.uta.edu**](mailto:binoyshekhar.dutt@mavs.uta.edu))

Priya Sanodia(**priya.sanodia@mavs.uta.edu**)

Group 12

Resume-Job Description Matching

Table of Contents

[1 Overview 2](#_Toc481886829)

[2 Defeating Application Tracking Systems 4](#_Toc481886830)

[2.1 Step 1: Getting Job Description 4](#_Toc481886831)

[2.2 Step 2: Getting Resume 6](#_Toc481886832)

[2.3 Step 3: Similarity 6](#_Toc481886833)

[2.4 Step 4: Improving Score 10](#_Toc481886834)

[3 References: 11](#_Toc481886835)

# Overview

The entire process of reviewing and evaluating resumes is evolving fast. Being a job seeker today, we have to adapt to the new reality of candidate screening if you hope to move past resume screening software and onto the interview.

**These software applications are called many things** — applicant tracking systems (ATS), resume screening software, resume robots, soulless automated resume rejecters, etc.

But whatever we prefer to call them, it behooves you to learn to speak their “language” fluently in order to make your resume stand out enough to even be glimpsed by human eyes.

**History of Applicant Tracking Systems**

Old resume tracking systems weren’t nearly as sophisticated as modern versions. In the past, you could simply employ critical key words (and tons of articles were written with the goal of advising you on how to do this) at various points in your resume in order to attract attention and be recognized as a potential star hire worth interviewing.

Unfortunately, many people in the past went overboard with the resume keyword stuffing process and created resumes that were pure nonsense. Considering that human eyes will eventually view the resume (if you’re lucky), this is a bad idea and one that shouldn’t be considered in an effort to game the system.

Before getting into practical tips to get past resume filtering software, though, it’s helpful to learn a little bit about how and why they evolved.

**Why Applicant Screening Software Evolved**

The prevalence of applicant tracking systems has grown substantially since the time of the great recession.

Companies advertising open positions often have hundreds, if not thousands, of applications to sift through in order to find a few “cream of the crop” candidates to interview for the job. Jobs continue to attract far more applicants than busy hiring departments have the time to interview.

It’s often too much for any one person — or even a small group of HR staff members — to deal with on their own. That’s how these resume robot programs came into existence for the most part — as an **aid to overwhelmed hiring managers.**

**How Widespread are Resume Screening Programs Among Employers?**

The *Wall Street Journal* reports that resume screening software use is widespread among larger companies to the tune of the “high 90 percent range.” It indicates that finding a Fortune 500 company that doesn’t employ application tracking systems would be exceptionally rare today.

**How Great Candidates Are Missed by Resume Screening Robots**

While this practice of electronically screening resumes saves time for busy HR executives, it also means that many highly qualified candidates are slipping through the cracks because they didn’t use specific language or, in some cases, formatting on their resumes. Don’t let this happen to you!

For job seekers in today’s era, it’s imperative to learn how to move past the algorithms. This means that job seekers must become more creative in order to make the right impression on both the robots that initially scan your resumes **AND** the people who will ultimately read them and need to be impressed enough to invite you to interview.

**The bottom line is that applicants must learn how to optimize resumes to make it through the screening process so you can ultimately get the job you seek.**

**How Do the Resume Screening Robots Work?**

Before you can figure out how to craft a resume that **wows the robot**, it’s helpful to first learn what the robot is looking for.

First, the software removes all formatting from the resume and scans for specific recognized keywords and key phrases.

Next, it sorts the content of your resume into individual categories:

1. Education
2. Contact Information
3. Skills
4. Work Experience

Then, the employer’s list of desired skills and keywords are matched against the results of the resume to determine your potential value to the organization.

Resumes with the highest scores relevant to the employer’s specified keywords and phrases combined with your years of experience will be moved up for further review.

In the end, the software simply scores the resume in order to determine which candidates are most qualified to move up the ladder for an actual human within the organization to review.

# Defeating Application Tracking Systems

In order to find a way to increase matches with the Application tracking systems we would need to under take the following steps

Step 1: Get some Job Descriptions based on the search keywords

Step 2: Parse a PDF resume (Generally the Resumes are in PDF format)

Step 3: Build a status summary that gives a summarized match result

Step 4: Provide possible solutions to improve the match

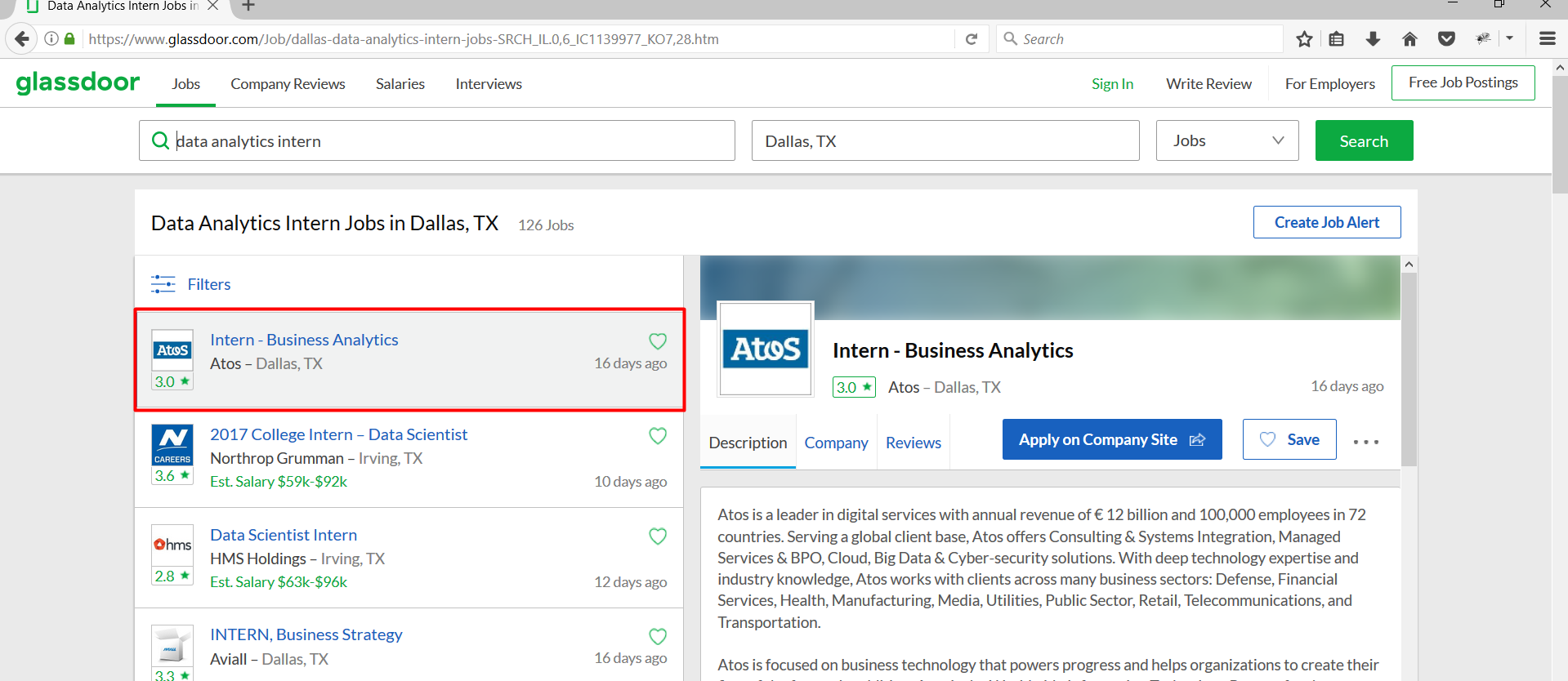
## Step 1: Getting Job Description

For getting the Job Description we first zeroed in on a source i.e. GlassDoor and undertook the following approaches

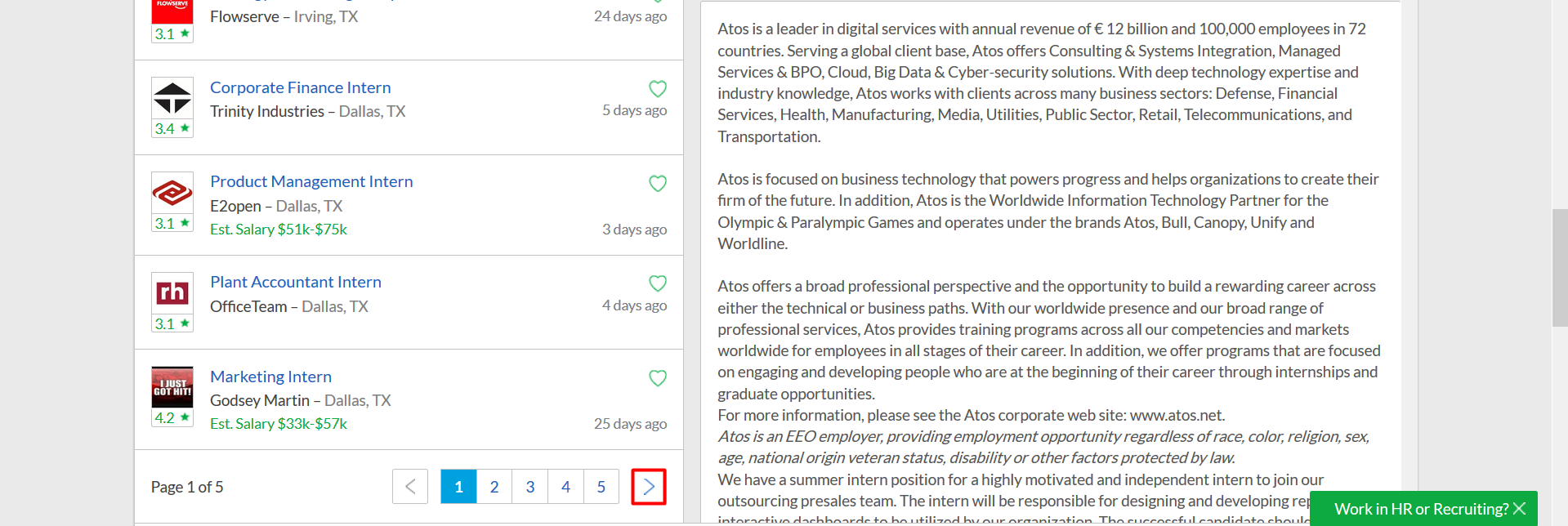
1. Getting Job Descriptions using Glass Door API
   * This approach failed as GlassDoor public is only shares information about various GlassDoor Stats
   * Glassdoor paid API provides the required information but we decided on taking up another route
2. Getting Job Descriptions using Web Scrapping (Beautiful Soup)
   * We tried to Scrape the Glass Door website by creating the Glass Door URL using the user inputted search Keyword and the user entered location ID
   * However due to added Glassdoor security remote scrapping of website is restricted and hence Beautiful Soup was unable to return the desired results.
3. Getting Job Descriptions using Selenium
   * As Glass Door had restriction on scrapping using remote scrappers we decided on using Selenium
   * The reason to use Selenium was the fact the Selenium mimics the human behavior by launching the web URL in real time and getting the desired information
   * One more advantage of Selenium was its ability to perform clicks like a real user

Now that we have narrowed down on Selenium we used a Mozilla Web Driver to launch the Glass Door URL with Keyword and location ID entered by the user

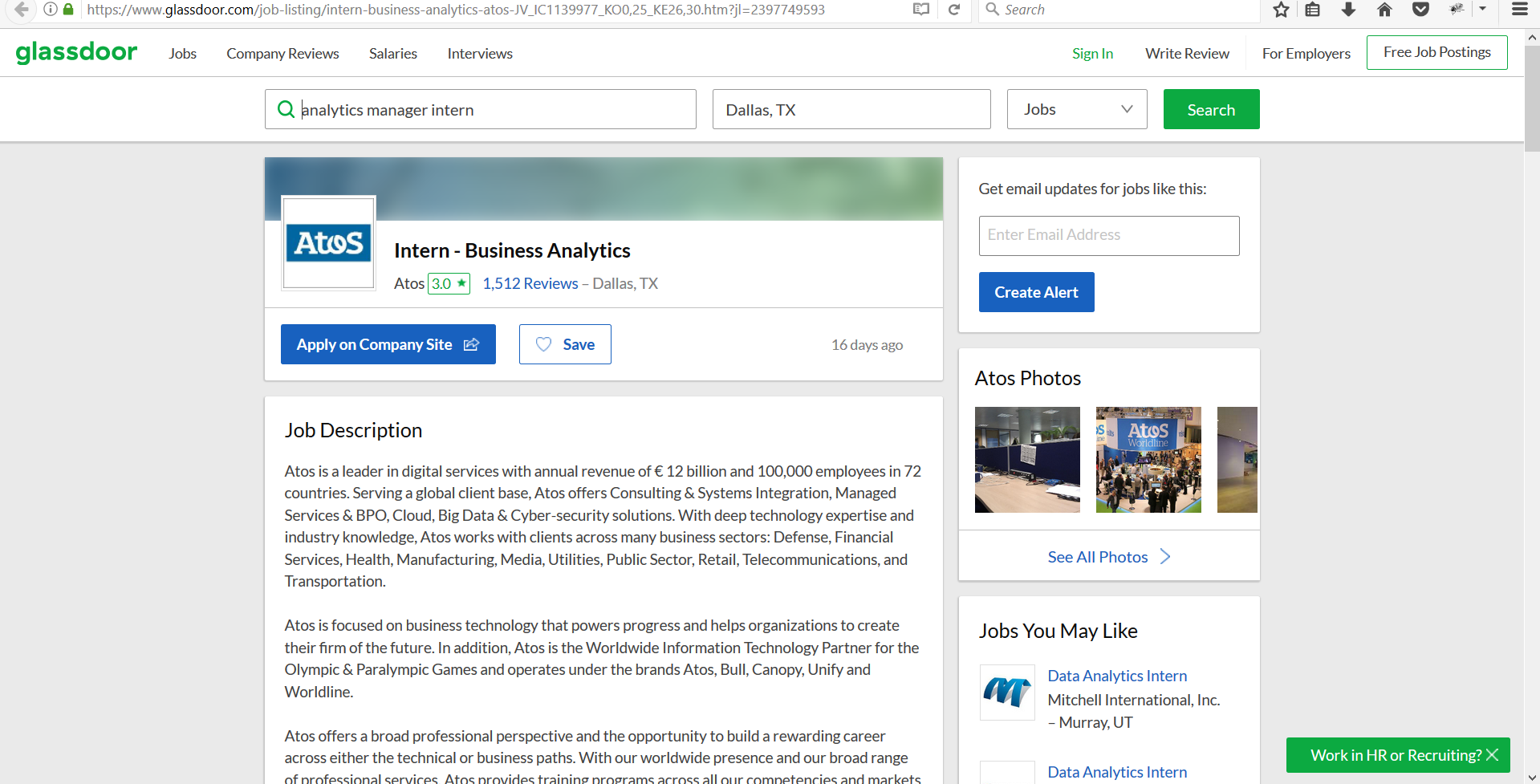
Selenium would then open a Mozilla Browser instance and would fetch each Job URL for each job that is listed for that location and keyword.



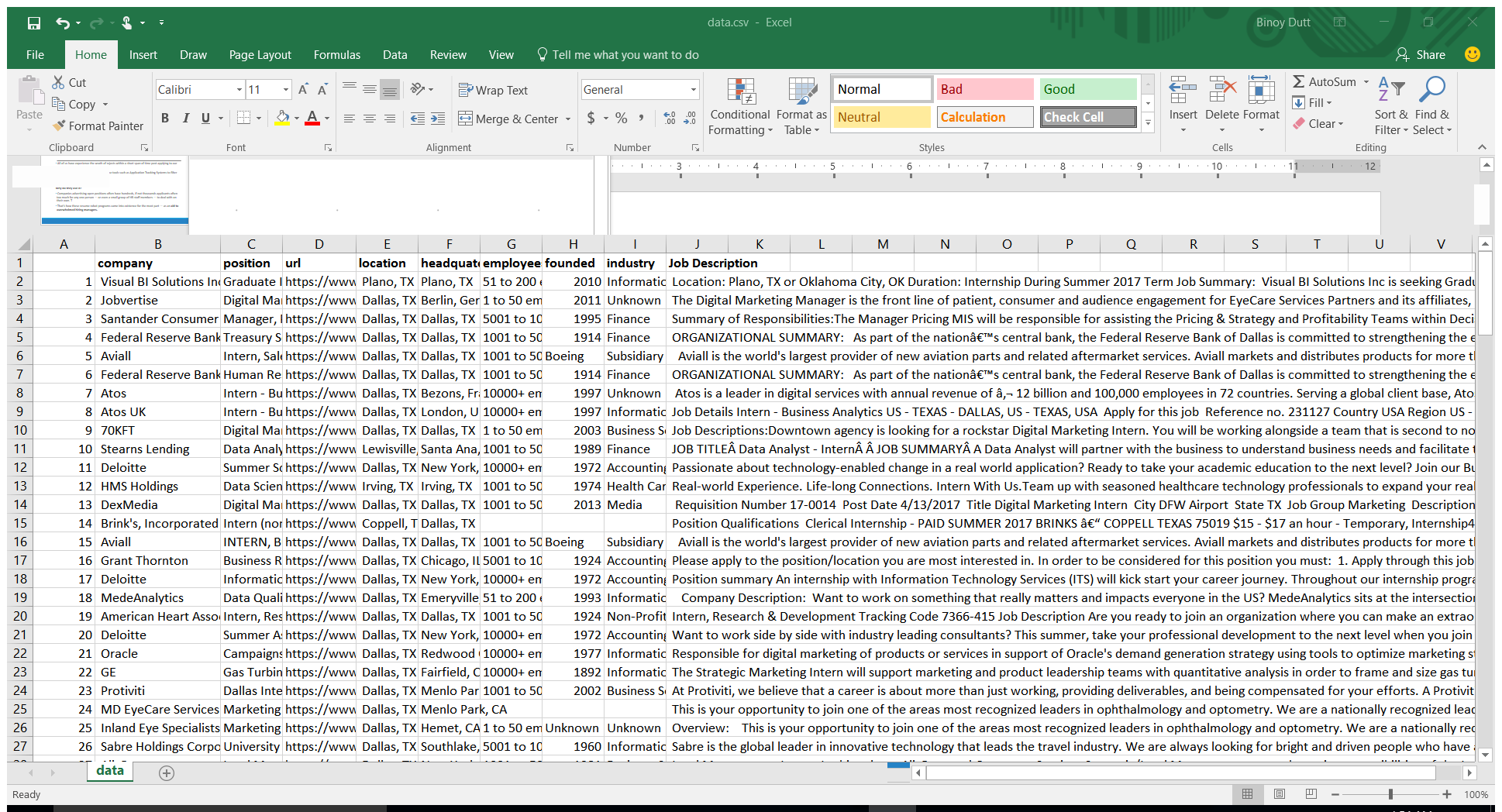
Selenium Webdriver would then move on to the next page and keep navigating to all the pages and fetch the Job URLs



The fetched URLs would again be launch in a Selenium Web Browser to get the page source and hence we would then be able to scrape the webpage getting the required information such as Job Description etc.



Hence we would we creating a Data Frame with the details as shown below:



## Step 2: Getting Resume

## We are well aware that the companies use Application Tracking Systems to filter resume. The idea here is make the applicant have some tool to get pass the Application Tracking Systems and reach into human hands.

Hence, the resume here is the  applicant's resume. Thus, in this case its my own resume that we am trying to match to various jobs and see my similarity. In case of a low match an applicant can re-word or add delete information and re-run and check if the edits helped or not.

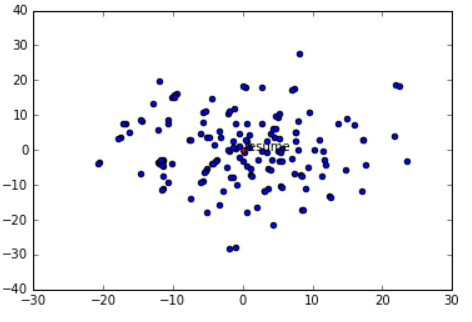
As we generally have the resume in PDF format we wanted the resume to be in Basic Text so that we can perform the required Natural Language Processing

In order to convert the PDF Resume to Text we have used the module PDFMiner which parses the resume going over each page of the PDF and getting the data in Text Format.

## Step 3: Similarity

In order to find similarities various approaches were undertaken which eventually failed to give decent results.

1. TF-IDF, Count Vectorizer
   * These are tradition NLP processes which have proved to give good result in the past and hence these were the first ones to be tried and tested
   * The limitation of TF-IDF and Count Vectorizer is that it’s a bag of word approach where in the placement and similarity of the words are not considered.
   * As Job Descriptions can have various synonymous words and same is the case of Resume this approached did not proved to give good enough results
2. Deep Learning Doc2Vec
   * We though of using Doc2Vec Deep Learning Algorithm training would own Job Descriptions
   * First attempt at Doc2Vec was made by tagging each job Description with the company name and generating the job
   * The next step was to plot the results



* + As can be seen from the results the resume sit right between the plot at 0,0
  + Under Doc2Vec we also attempted on creating our own model with all the job description tags as JD but again that was not a great idea as then would have a single vector representation of the Job Description.
  + The plot would then contain just 2 vectors i.e. a single vector representation of all the jobs and the resume vector representation

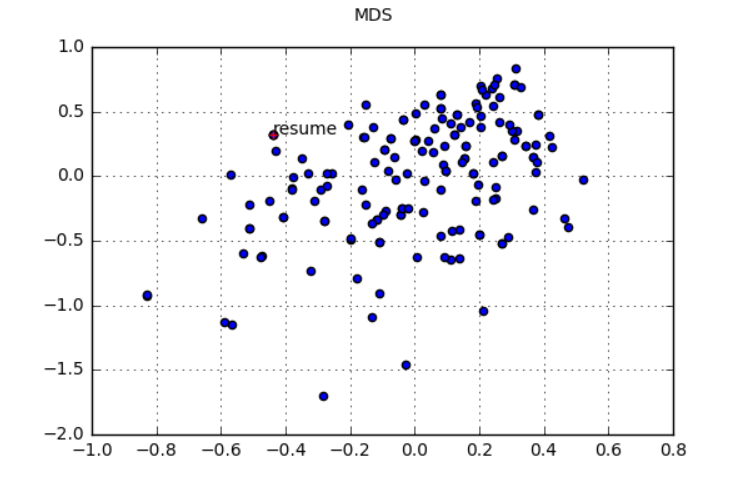
1. Deep Learning Word2Vec
   * Then we finally decided to go with Word2Vec. As for word2vec to work effectively we would require a huge training data set we decide to try out the model using pre-trained google word2vec model

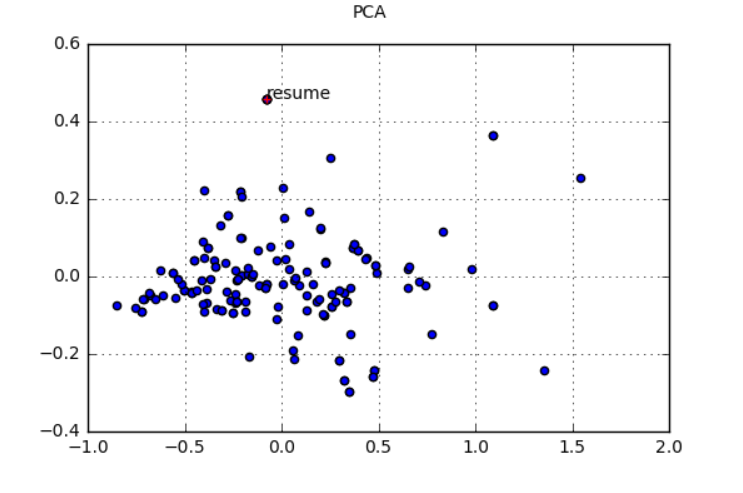
While using Google Word2Vec the document was first stripped off the punctuation and digits. Then for each word in the document a vector representation was obtaining skipping the words not present in the Google Word2Vec model.

Hence now we have each Job Description represented as a list of word vectors with 300 dimensions for each word. In order to compute the similarities, we required to get a single vector representation for each Job Description

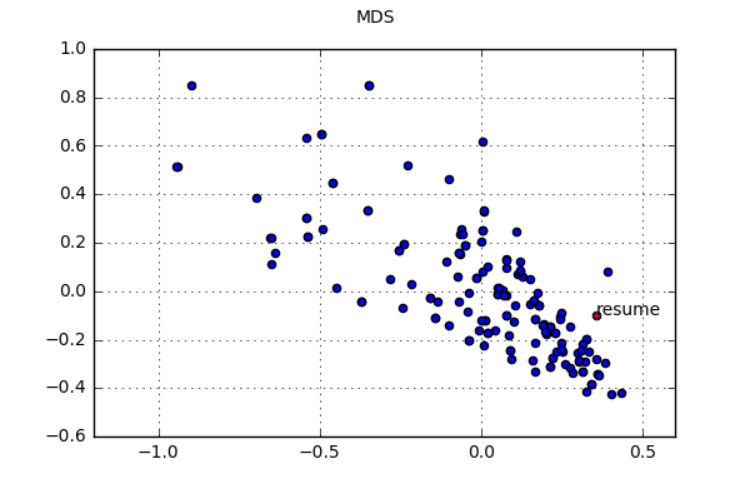
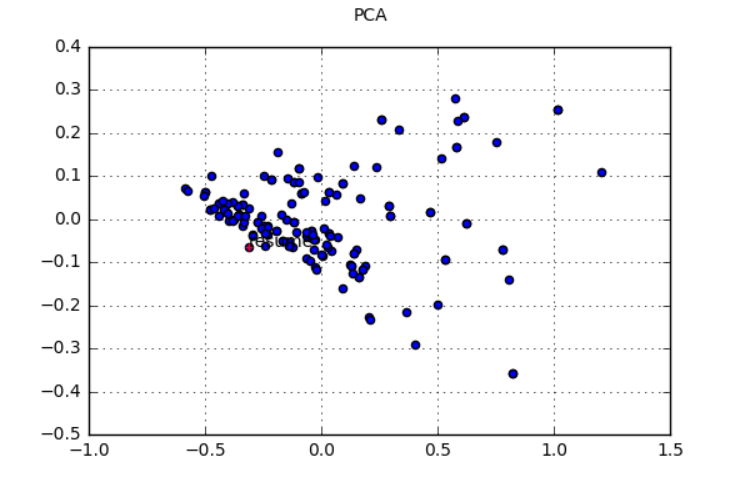
The following 2 approached were identified to get a single vector representation:

1. Taking the Average
   * In this method, we took the average of all the word vectors for each of the 300 dimensions and hence ultimately we had a single vector for a Job Description with 300 dimension in them
   * This approach was not suitable in our case as there were many words which were abundant in the Job Descriptions but would never appear in a Resume for example words like ‘Job Description’, ‘Requirements’ etc.
   * Also, the weights of all the words are same we do know that for a job in any domain there are always few highlight skills that are important.





1. Taking the Average using TF-IDF
   * To tackle with the weights issue used the TF- IDF scores for each word and multiplied its vector to the same and then took the average
   * In this way, we would get the weighted averages
   * The problem with this approach is that words like ‘Python’ which is a key skill and can appear in multiple Job Description would lose its weights because of the IDF factor.
   * Hence, we created a list of key skill and weighted them higher manually.
   * We would require extensive domain knowledge to come up with the key skill and the manual weights would require to be optimized by running the code with different values

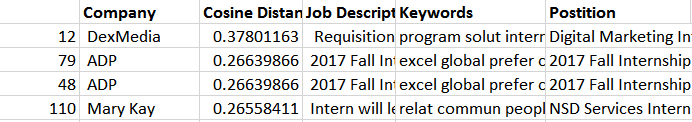


Now as we have a single vector representation of each document.

Similar steps were performed for the Resume where in the resume was first represented as a list of word vectors with 300 dimensions and then the mean was taking using the TF-IDF scores resulting a single vector representation of each word

Now that we have the Job description vectors and Resume vector we have used cosine similarity to compute the distance.

For each Job Description cosine distance from the resume was computed and stored in a list. Hence, now we can infer about how close is the Job description from a particular resume.



## Step 4: Improving Score

In order to get some insights on to what words can be added to improve out cosine similarity and hence increase the match we under took the following approached

1. Keywords – Gensim
   * For each Job description the Keyword method from Gensim module was used to extract the top words.
   * However the results obtained were to generic
2. Topic Modelling
   * We also clustered all the job description into a corpus and performed topic modelling using Non Negative Matrix Factorization.
   * The results as expected were not as promising as each job description would be holding a separate topic.
   * Even though all jobs are related to a specific keyword but the required would vary a lot and hence considering all the jobs together will not give us good result
   * Hence a better approach would be to cluster the similar docs together and then perform topic modelling on them which would more relevant results. This again is a challenge in it self.
3. Word2Vec Most Similar
   * As each Job Description is hosted on a 300 dimensional vector space which also contain individual words the best approach to increase the cosine similarity would be to find the words which are close to the Job Description vector
   * If we obtain such words and incorporate them into our resume, our resume would then tend to come closer to the Job Description
   * Implementation of this method is still under research
4. Dealing with more structured Data
   * Most of the companies makes the applicant fill out details separately rather than just ask for the resume which more or less contains the same information
   * This in turn helps getting the data in a more structured manner.
   * More structure here mean that we can easy filter out the details which are un wanted such as name, address, Education etc. and focus more on the required skill set

Notes:

* We regret the fact that without having positive and negative resumes we wont be able to set threshold value for similarity.
* If we have such data most of the work is done and we can easily created a very strong model that can by pass the application tracking systems used by companies.
* We would need some major support probably of some Job board or few companies to help provide the data incase we want to create a universal model for this.
* We will try something over the summer to take this ahead if possible.

# References:

https://www.glassdoor.com/developer/jobsApiActions.htm

http://workplacediversity.com/articles/How-to-Get-the-Applicant-Tracking-System-to-Pick-You

Stack overflow for code snippets