Zachery Macintyre

Nikhil Gopal

**Intro:**

No matter what point you are in your life you have had to apply for a job, or will be applying for a job in the future. It can be a long and tedious process, that involves reading through many roles and job descriptions that you ultimately are not interested in. On the flip side, all companies post job descriptions about open positions. Inevitable it leads to more applicants than they can possibly go through. In our project we create a matching tool from your resume to a job description, which also leads to insights about how you should write your resume if you are going after certain roles. However, it must be noted this is our best attempt to optimize your chances and find job matches that you are actually interested in. So, we are not guaranteeing you will get the job, but hopefully it will help.

**Data Description:**

For the data we used indeed.com job descriptions and info scrapped by our professor Dr. Wayne Lee. The data was broken down by Job, State, Employment type, Description, and Year,

in total we were dealing with a little over 500 jobs. The data clearly is not uniformly distributed, however for our methods it would not matter, and this is discussed later.

On to the actual data, and our cleaning techniques. The cleaning of the job descriptions was handled by Zac, and he noticed a few abnormalities that could be fixed. The first being that there were many instances of sentences that would end and a new began without any sort of spacing. For example, “engineer.The” and this would cause problems later down the line if not fixed. So, we fixed it by removing any periods and replacing them with a space. Outside of that, the data was relatively well structured for what we were trying to accomplish in the project. The only other things we did for data wrangling was creating different data frames for different applications. For example, we created a list that mapped job titles to the average number of words in total descriptions for the title. In other words, we got the frequency of words that appeared in every job title, such as data scientist or office manager. Then, we summed those frequencies and divided by total documents to get the average frequency for job titles. We additionally made a few more data frames for other instances in which it was helpful.

**Methodology:**

First, we must discuss the inherent problem and complications that arise in natural language processing (NLP) problems. In many machine learning techniques, the data used is purely numerical, and if not, there are clever ways to change it to a numerical form. However, in NLP problems, words and phrases are what is important. It makes methods like linear regression, or logistic regression more challenging to use. On top of that, we are not trying to predict a continuous variable, we trying to binarily decided if will you or if you will not like a job. We did this by trying out 2 different methods. The first was a comparison between a naïve method and a term frequency – inverse document frequency. This gave us job title matches based on your resume, and then we used a similar thing to get the job descriptions that best matched.

**Method 1.**

Now we are on to the actual way we solved this problem. Our first method used a comparison between a term frequency – inverse document frequency (tf-idf) method and a naïve method. The naïve method used the average frequency words appeared in each job title as discussed above. The argument for this was that in a job description you only have so many words to use and want to make them count. So, in a description for a statistician or data science the word data might come numerous times, and that indicates the importance. In the tf-idf method we used term frequency and the inverse document frequency to get important words. The argument here is this is a very common method to get important words in a document when dealing with NLP problems. Initially, we thought the naïve method would work best. Our reasoning being the fact that companies only have so many words to present the job and responsibilities and they want to highlight the most important things. Then, in your resume you should take a similar approach only having one page and emphasizing what you are good at. However, when running different applications, it seems the tf-idf method works best no matter what.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Job Rank | Normal Resume w/naïve method | Normal Resume w/tf-idf method | Spam Resume w/naïve method | Spam Resume w/tf-idf method | Changed Resume w/naïve method | Changed Resume w/tf-idf method |
| 1 | Data Architect | Data Scientist | Human Resources | Data Scientist | Data Architect | Ux designer |
| 2 | Data Scientist | Ux designer | Data Architect | Ux designer | Data Scientist | Data scientist |
| 3 | Machine learning engineer | Software developer | Data Scientist | Software Developer | Machine learning engineer | recruiter |
| 4 | statistician | Data architect | Machine learning engineer | Data Architect | statistician | Software developer |
| 5 | Deep learning | Deep learning | Statistician | Recruiter | Deep learning | marketing |

In this chart we have the normal resume of Zac, who is aspiring to be a data scientist after graduation. In his normal resume both methods appear to work about the same. All job titles have appeal. In the spam resume, we added spam words in that should map us to human resources. We did this by getting the most frequently used words in human resources descriptions, and added those to the bottom of the resume. For the naïve method human resources was the top ranked job, but in the tf-idf method it only helped include recruiter as the 5th ranked job. This happened because in the tf-idf method the word human and resource are downgraded because they appear in many documents. Finally, for changed resume we added other experiences Zac has had that deal with people and relationships, mentoring and being an orientation leader. In this situation the tf-idf better reflected the changes than the naïve method. For all the reasons above the tf-idf method does better as it accurately reflects the desire of the resume.

Next, gathering the top 5 job titles we output the top match in each position. This way the recommendation system most accurately fits your resume and gives out diverse recommendations. It must be noted, that this is more a human-in-the model algorithm than a traditional algorithm, and as such you can change many of the features. For example, you could take the top 3 titles that match and get the top 3 best fitting descriptions. The way we fit descriptions to the resume was using the tf-idf for each description. The rationale behind this, is that descriptions for a data scientist at many companies should be very similar to one another. Thus, the differences are what will make the matches the best.