ANLY-533-01

Fall Term – Group 3

Adam Sampson, Jack Letcher, Nathan Thomas, Taylor Maxson

## Business Purpose

The business purpose defined for this project is to develop a method, using text classification, to produce a list of the top fitting jobs for a client of a career recruiting firm, with the intended goal of getting the client selected for an interview. Initially, this concept would review a client’s resume, scan for the key words and pair it with jobs matching the highest number of key terms. This model provides a solution to the problem of spending several hours of work time sifting through job postings and determining which jobs accurately match the skills of the client.

During the brainstorming sessions, several secondary applications to this model were developed and discussed. One such method would include reverse matching, or determining which key words or skills should be attained or highlighted on a resume in order to more closely match the skills needed in the job posting. Another potential use xxxxx

## Data

The data used for this project was scraped from the individual companies’ websites. The original strategy for obtaining data was to do an API call to job sites such as Monster.com and Indeed.com. These sites though did not grant permission to use the data, therefore the group decided to scrape individual job data from company websites. The parameters chosen were to perform a search on the “careers” section of the company website using only keyword “Data”. The simplicity here would provide consistency between each company. The extracted raw files were in XML and HTML formats and were formatted differently for each company. The five companies who we were able to obtain job data for were Apple, Dice.com, Humana, Microsoft, and Walmart.

## Data Preparation

Obtaining the data was a tedious process as each website was built a bit differently and code had to be tweaked to scrape the different sites. The data sets themselves presented some real challenges as well. The first method to getting it to a useable corpus was to parse the xml and html data into a data frame. It was determined to keep only a few pieces of identifying information from the documents and load most of the remaining data as the description. The information kept was the job identification number, the job title, and finally, the job description. The job description consisted of text about the job, the requirements for the job, as well as any company information that was included in the file. A combination of R functions and regular expressions were used to extract the information from the documents. There were a few issues with the Dice.com data though. The description data was duplicated on several sub-nodes, causing confusion on what sub-node to include. The simple solution was to pull in sub-nodes 18-20, however some of the descriptions ended at node 18. The data was able to be extracted to a text file and the data string from the text file was read into the description field. One other key issue with the Dice.com data was that the job title was not able to be parsed from the data. On the other data sets, the job title was embedded as a separate node, while in the Dice.com data, it was embedded in the general text and not able to be parsed out. The data was eventually cleansed to include only alpha text, numeric text as well as the “+” and “-“ symbols. The final transformation was done in the model files, where the text was transformed to lower case, stop words removed and whitespace stripped out. The final data was then loaded into a corpus used for the modeling process.

## Modeling

In an attempt to produce a model that would accurately match a resume with a job description, several methods were chosen to accomplish this task. K-Means was used to determine what key terms were present and how the jobs could be grouped. xxxxx

### k-Means Clustering

A screenshot of a cell phone

Description generated with high confidenceOne method chosen to analyze the data was k-Means Clustering. This method was chosen to group jobs by types so that a client could match key terms from their resume to the terms in the job descriptions. Initially, a histogram of the top words was generated to see the highest key term frequency. Intuitively, data was the highest frequency term since our job search was based exclusively on this term. Other high frequency terms were not helpful in determining groups associated with “data” job types. There were some key qualities or traits that were more frequent, such as “management”, “engineering”, “technical”, and “software”.

A screenshot of a cell phone

Description generated with very high confidence

A simple word cloud was produced to graphically display the frequency of words in the corpus. The larger words indicate the highest frequency. The word cloud indicates several key qualities or general business terms such as “experience”, “business” and “team”. Again, you have to dig pretty deep to see the real key job terms.

The expectation of this model was to gain insight into what categories may surface from the key terms. With the initial model, it was determined that the total variance in our data set explained by the clustering was 12.4%, which is not a very high value. To improve the model, some terms which were obviously not key to the analysis were removed. This only improved the model to accounting for 36.3% of the variation in the data. Using this information, we attempted to determine how many groups were applicable to the data obtained.

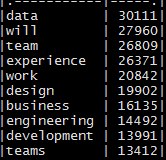
A screenshot of a cell phone

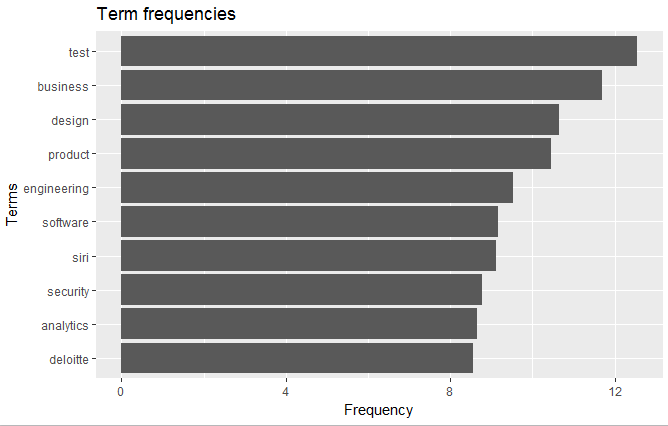
Description generated with very high confidence

### TDF-IDF

TF-IDF, short for term frequency-inverse document frequency, is a numeric measure that is used to score the importance of a word in a document based on how often it appears in a document or a collection of documents. The perception is: If a word appears frequently in a document, then it should be important and we should score the word higher, but if the word appears in too many other documents, it’s most likely not a unique identifier and therefore scores less. For this reason, TF-IDF is a better modeling technique because it doesn’t matter if a term appears once or 100 times in a document, it will still be counted as 1 appearance and therefore represents our business purpose more accurately.

TF-IDF was selected as a useful text mining model for our data because it parallels our business objective by finding the most frequently used words in a job description/ job posting. Term frequency by itself is less meaningful to us, but by isolating both term frequency and inverse document frequency, we can identify words that were less frequent, with how many documents the words are associated with. This is particularly important by identifying words that appear only once or twice in a job description, but appear in most documents. Think about IDF as a measure of uniqueness. It helps search engines, or in our case job postings, identify what it is that makes a given document special. This is much more sophisticated than keyword density. What IDF teaches us is the importance of uniqueness in the content we create. If you can pick out a smaller number of terms with much less competition and create content around those needs, you can start to rank for these terms because of the uniqueness of the content and combinations of rarer terms.

The expectation we had going into this project was a lot of repetitive, messy, uncleansed data. Our expectation for word frequencies leaned towards actionable skill sets and technical program languages. Our observation was a bit different than what we expected. After pulling out the top 10 most frequent terms, there were very little meaningful terms that appeared at the head of the frequency table. Top 10 frequencies can be seen in Fig X.

As you can see in the figure above, there was minimal skill set terms and nothing for technical languages from the frequency table. Going back to an earlier point, our focus was not solely on TF, but the isolation and analysis of TF and IDF together. Therefore, while creating the TDM, weight is defaulted on TF, but by weighting on TF-IDF, it allows for a more well-rounded model. In Fig. X, the top 10 terms and frequencies using TF-IDF are a bit different then the terms using TF as weight.

Two terms were particularly interesting, siri and deloitte. This would require further analysis to help explain why the TF-IDF was so high in these two terms, but intriguing at very least. The full TF-IDF frequency table can be seen above. Taking the analysis one step further, text association was analyzed. In Fig. X below, an example was given using the word “python”, to see what terms were correlated above 0.25. Ruby was the highest correlated word associated with python. Interesting enough, Ruby was and is still considered the predecessor to Python.



### Topic Modeling (LSA)

One method that is common for text analysis is Latent Dirichlet Allocation (LDA). This method groups documents into topics. The results of an LDA is a probabilistic grouping of documents based the terms in the entire set of documents.

The jobs data was cleaned to remove extraneous characters and to remove terms which we did not want to be used in the category. Specifically, our business purpose is to rank jobs based on job similarity, so the name of the company was removed from the text of the document. Otherwise it was likely that the name of the company repeated multiple times in a document would become a strong influence in the document breakdown in this analysis. Additionally, the standard text cleaning methods of converting to lowercase, removing punctuation, and cleaning whitespace were performed. The corpus was then converted to a document term matrix (DTM).

Next, an LDA was run on the DTM. The first model run was with the number of topics set at k=5. The next model was at k=50. Due to the time consuming nature of running the LDA, these were the only allocations that were run. The top 10 terms for each lda was produced to get a representation of the major topics in each allocation. Next the topic probabilities were calculated to identify the increased likelihood of a topic based on each term. Additionally, term probabilities were calculated to identify the probability of each term occurring in each of the topics.

The resumes were loaded into a document term process using the same clean-up and process that was used to prepare the jobs data for the LDA. Next, the topic likelihoods and term likelihoods were calculated for the resumes using the posterior() function to calculate based on the previous LDA. The most likely topic for each resume was also calculated.

The next analysis that was planned was to use the LDA to identify the most relevant words (based on a not-yet identified cut-off) for that document. Then we anticipated that we would be able to search resumes for those particular terms only and perform a distance measure from each topic. Additionally, it was anticipated that an analysis would be performed to identify how much of the data was covered by the LDA. Currently it is unknown whether the model is underfit or overfit. Further, more analysis can be performed to identify the characteristics of each topic more than just getting the top 10 characteristics.

One of the goals of this analysis is to identify jobs that a candidate is a good fit for. LDA allows for the reduction of dimensionality in the final analysis. It also produces the probabilities that words terms will exists in documents which is useful in calculating whether another document (the resume) has high likelihood of matching the job posting based on comparison of terms. One of the concerns when selecting this method was that LDA is best at categorizing documents. It may have problems with identifying when a document doesn’t fit into any of the topics and will simply select the best topic (by default). This requires additional work beyond LDA to identify how well the document/resume matches the topic. In other words, it identifies the closest topic easily, but doesn’t indicate when the document doesn’t belong in any topic.

The LDA was expected to help in the following ways:

1. Categorize the jobs data into topics
2. Identify important terms from each topic

Running the LDA with k=50 resulted in a very wide topic model. A visual review (below) shows that many of the terms that are identified are not exceptionally representative of specific requirements. It is not expected that employers will look for “will” or “business” or “time” in a resume. Additionally, this model does not clearly represent the most important terms to employers. One of the interesting trends in the data is that Siri is a top term in more than one of the topics models identified. Also, machine learning makes a good showing in Topic 5.



In addition to the top terms in the document, the LDA produced the probabilities that a term would occur in a document within each topic, and the probability that document would match a topic. The output for this information is not clear enough to post here but is available in the code on GitHub.

Running the LDA with k=5 resulted in a much simpler topic model (top 10 terms below). Unfortunately, this also resulted in even less critical terms being in the top of each document. This is likely because the jobs data is full of narratives and many of the words are used multiple times. The more critical tools and processes are only listed few (or one) times and therefore are not strongly represented here.



### Jaccard Similarity

When discussing our plan of attack for this project and what models to use, Jaccard Similarity immediately stuck out as a valuable tool that would be of value in matching specific resume skills and attributes to highly sought after attributes of an ideal job candidate or employee.

Jaccard Similarity uses a relatively simple concept to compare similarities and differences between sets of data. In this case, we used a vector consisting of words we determined (both by other models deployed as well as tools including custom Dictionaries, etc.) were frequently listed in data job postings, either as a minimum requirement, a preferred attribute, or even something that would give a candidate a leg up in the job search process. This vector provided us with our “key words”, or the words that would ideally be present in a resume of someone seeking a job in the data field. These “key words” are commonly present in data job postings and if also present in a resume, would in theory indicate the candidate was qualified for the job requirements. However, as mentioned in class, a data model is only as good as its attributes, making this part of the process one of the first determinants of how truly valuable the model would be in real world settings.

We felt the Jaccard method was appropriate and sufficient for our specific business question and question of interest. We were not as concerned about “meaning” in the words within a resume, but simply seeing if the words we felt were “key words” in data job postings (and therefore contained in our vector list ) were present at all in an individual’s resume. To do this, the Jaccard method uses binary scores of 1 and 0 to denote if a word has been matched between data sets. If a word from the tested dataset has a 0, it has not been matched and if the word from the tested set has a 1 it has been matched to a word within the predetermined key words. If Jaccard is not only a relatively simple method to determine but also gave us a straightforward “metric”, defined as a number between a 0 and a 1 by taking a ratio of the similarities and the total cases. A Jaccard similarity “score” that fell closer to 1 meant there were a relatively high amount of matched words or similarity between the two datasets.

For data sets to compare to our vector of key words, we used actual resumes provided as text files. We imported the text from each resume and removed any words in the resume that were not in our vector list. Finally, we used the resulting data (after removing the words that did not appear in our vector) to run the Jaccard method. We were interested in looking at resumes with a variety of education backgrounds, rather than testing only resumes of data science students. We felt that this would help us assess whether our model was truly a good model or if it was only good in very specific cases. If the model did not perform well in real world cases then there would really be no use for it. For this project, the resume collection is an area of concern simply due to lack of data quantity in resume form. In the future, we would advise, as always in data modeling, to ensure not only having a large sample size N of resumes, but also a varied random sample set within the group collected. This is an area we felt could be improved in the future.

We expected the relationship between the amount of words a resume had that were also in our vector list to suggest that the individual, at the very least, had the “bare minimum” qualifications required by most data job postings to pass a “first screen” test. Usually the first screening in the interview process is done within HR and essentially is simply a straightforward process of checking off a list of qualifications determined by their company in comparison with the candidates resume information. Therefore, we concluded that while there are too many outside variables ( i.e. poor interview performance, not fitting with the company culture, etc.) in whether the individual would ultimately be determined as “good for the job” and as a result be more likely to receive a job offer. However, we felt that we could at the very least determine if they would likely pass the first “screen” of the interview process by having the essential attributes desired by the interviewing company. Getting your foot in the door is sometimes half the battle In the job interview process, and without at the very least the key qualifications, a candidate is likely to be overlooked or dismissed within a few minutes. Our hope is to maximize a candidate’s likelihood of at least passing the screen round of an interview, which in this case would mean having similarity between their resume and the qualifications frequently listed for job they want.

As mentioned earlier, a benefit of using the Jaccard similarity method is its simplicity. It looks for specific words within a data set that match a predetermined list of “target words” in another dataset. The end result the Jaccard method gives is an overall percentage (with a score between 1 and 0) that tells you how similar the two data sets are based on the amount of matches or similarities found between the two. While for our case this was sufficient, it did not address issues such as the relationship between words, the meaning of words, etc. It simply scores matched cases. It also depends highly on the formatting, language, misspellings, etc. of the two datasets and may miss matches due to one of these or other factors.

Some of the resumes run through our model contained a high amount (usually equal to a 0.6 or above Jaccard score), while others did not. The more similarities or matches cases found, the more likely we felt an individual would be to passing the screen test of the interview process. Interestingly, we found that some of our classmates and group members resumes would have little to no matches found between their resume and common data job posting qualifications previous to being a part of Bellarmine’s MSA program. However, after participating in the MSA program, these highly preferred attributes were more likely to be found in the same person’s resume which is an encouraging sign that this program is preparing students well for a future in the data field.

## Evaluation

Did we solve the business problem?

Further analysis could be done on the frequent terms to determine what other key terms could be used to scrape jobs outside of just the word “data”. This would allow us to capture jobs that are similar, using some of the same methods listed in our report, which may not contain “data” as a key term.

## Deployment