**Group Assignment 1: Job Description Salaries**

**Team Members:** Samruddhi Somani, Julia Wu, Leon Chen, Daniel Lai, Andrew Rueben, Bejan Sadeghian

**Part A**

A1) What are the top 5 parts of speech in this corpus of job descriptions? How frequently do they appear?

You can see the top five parts of speech below in Figure 1 and Table 1.

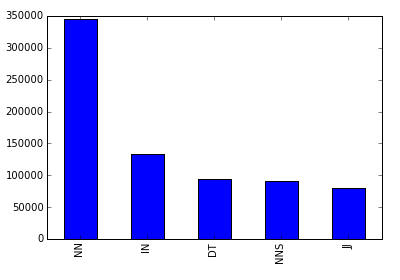
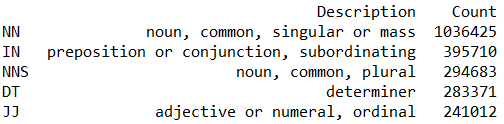


Figure 1: Top 5 Parts of Speech Frequencies

Table 1: POS Descriptions & Counts



A2) Does this corpus support Zipf’s law? Plot the most common 100 words in the corpus against the theoretical prediction of the law. For this question, do not remove stop words. Also do not perform stemming or lemmatization.

Yes, this corpus does support Zipf’s Law in the fact that as rank increases, frequency decreases exponentially. Figure 2 below shows the top 100 words support this.

In Figure 3 below we included the top 10,000 words to see when Zipf’s law would break down. You can see at the end between 103 and 104 the harmonic series diverges, which is mentioned in Zipf’s Law.

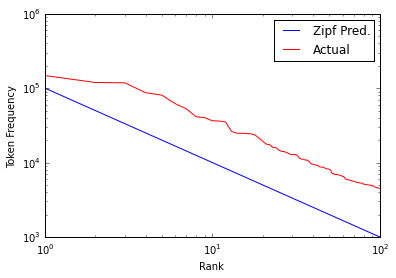


Figure 2 : Top 100 Words

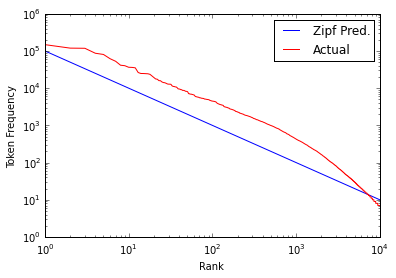


Figure 3: Top 10,000 Words

A3) If we remove stopwords and lemmatize the corpus, what are the 10 most common words? What is their frequency?

Figure 4 shows the ten most common words from our bag of words with stop words removed and lemmatization performed. Figure 5 gives these word’s frequencies.

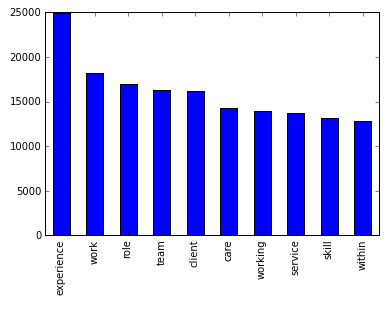


Figure 4: Top 10 Words After Stop Words Were Removed and Lemmatization

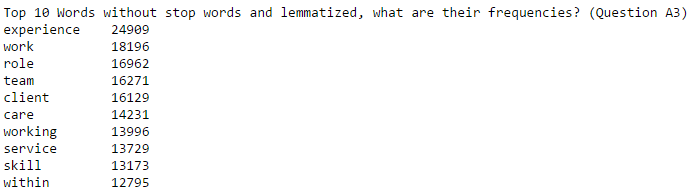


Figure 5: Frequencies of Top Ten Words After Stop Words Were Removed and Lemmatization

Part B

Preface) Here we will describe our data munging methodology.

We used 15,000 descriptions from the total data set. We chose the 15,000 randomly but ensured that our sample set was 50% of the descriptions with normalized salaries above the 75th percentile and 50% below the 75th percentile, as was recommended in class. From there we split our dataset so that 60% of the data would be in our training set and 40% would be in our test set.

For the data itself we used the TfidfVectorizer to create a sparse matrix for our x variables. In this function we specified to use TF-IDF smoothing (smooth\_idf = True). For our model we used the Multinomial Naïve Bayes approach.

B1) Create a classification model with *all* words and the bag-of-words approach. How accurate is the model (show the confusion matrix)?

We ran two different Multinomial Naïve Bayes models one using the random sample of all of the data [Table 2] and the other using 50/50 split of the classes [Table 3]. The reason we used the 50/50 split was to allow our function an equal sample for both classes to train on to achieve a better accuracy. The random sample predicted zeros for nearly everything because our training sample in that case was mostly zeros (lows). You can see that the random sample model gave us an accuracy of 75.017% whereas the 50/50 split model gave us an accuracy of 80.15%.

Table 2: Summary Bag of Words Model (Random Sample)

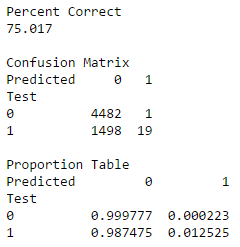
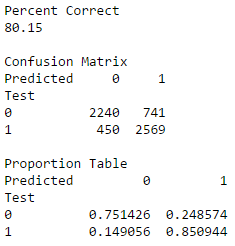


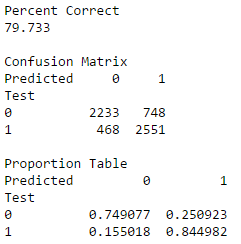
Table 3: Summary Bag of Words Model (Equal Sizes)



B2) Speculate before running the following analysis whether lemmatization would help improve the accuracy of classification. Now create a classification model after lemmatization. Did the classification accuracy increase relative to B1? Comment on your speculation versus the actual results you obtained.

We believed that lemmatization would in fact help improve the accuracy of our model. Table 4 shows the summary for our Multinomial Naïve Bayes approach on a bag of words with lemmatization. Our accuracy came to be 79.733% which is a bit lower than expected but nearly negligible compared to the approach from B1. This slight degradation could be due to lemmatization taking out structural cues.

Table 4: Summary of the Lemmatized Bag of Words Model



B3) If you got better results with lemmatization, retain the lemmatized data, else use the original one. Now speculate whether stopwords removal would help increase the accuracy of the model. Take out the stopwords, build a classification model and check the accuracy, and compare with that in B1 & B2. Also show the top 10 words (excluding stopwords) that are most indicative of (i) high salary, and (ii) low salary.

We used the 50/50 dataset from B1 since Lemmatization actually showed worse accuracy. We speculated that stop words should help us improve accuracy because we are essentially removing noise. Our model gave us an accuracy of 80.067% which is marginally better than our previous best (B1 at 80.15%). Table 5 shows us the summary for the stop words removed.

Table 5: Summary Stop Words Removed

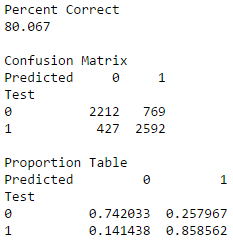


Figure 6 shows the top ten words for low and high salaries respectively. What we did was inverse the classes to be able to pull the largest coefficients the Naïve Bayes function in Python gave us to show the tokens most indicative for each respective class. We did this to remain in accordance with the guide provided.

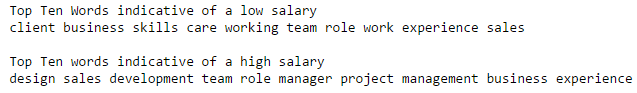


Figure 6: Top Ten Words Indicative of Low/High Salaries

Since we saw overlap in the words using the coefficients method described above, we decided to try using log probability ratios to determine the tokens that are mostly found in one class but not the other. Figure 7 shows us the ten words most indicative of higher and lower salaries respectively using this method. We used the log probability ratio between high salary class and low salary class for each token from the Naïve Bayes function in Python. So for example if we have a token that has a high probability of being in the high class and a low probability of being in the low class we can say that it is indicative of the high class and vice versa.

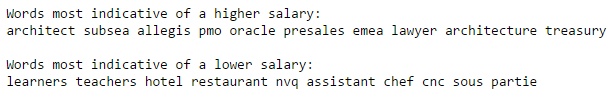


Figure 7: Ten Most Indicative Words of Higher/Lower Salaries

B4) Use the job descriptions without lemmatiztion and stopword removal. Add parts-of-speech bigrams to the bag-of-words, and run a new classification model. Does the accuracy increase over the results in B1?

Table 6 shows us the confusion matrix for the prediction results using POS Bi-Grams as our tokens. The accuracy actually dropped significantly with POS bi-grams, we ended up getting an accuracy of 63.635%. Despite the accuracy dropping, it was impressive that just using POS Bi-Grams we were able to achieve such a decent accuracy.

Note: We checked with Sam (TA) and he confirmed to use just POS Bi-Grams for our bag of words.

Table 6: Confusion Matrix POS Bi-Grams

