Text Analytics Group Assignment 1 (Fall 2015)

Date due: Wednesday Sep 9 by start of class on Canvas. You are responsible for forming your own group (5-6 students).

This assignment has two parts, A and B. In A, you will perform some basic text mining tasks just to familiarize yourself with the nuances of putting text mining theories to practice with Python scripts. If you are more comfortable with R, please feel free to use it; however, in that case I or the TA will not be able to help you with coding issues. Part B involves building and testing classification models to predict salaries from the text contained in the job descriptions. The data for this assignment can be found at http://www.kaggle.com/c/job-salary-prediction

Part A (basic text mining)

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

Hint: nltk.org is a great resource for exploring text mining with Python. There are many examples that are similar to the questions in this assignment.

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 stopwords. Also do not perform stemming or lemmatization.

Hint: Check http://www.garysieling.com/blog/exploring-zipfs-law-with-python-nltk-scipy-and-matplotlib

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

Part B (predict salary from job description)

In this section, you will create classification models to predict *high* (75th percentile and above) or *low* (below 75th percentile) salary from the text contained in the job descriptions. Ignore all other data provided on kaggle.com that is not text (except the actual salary data, which you need to create the binary output of high/low salary).

Get the training data on kaggle.com (train_rev1), divide it randomly into training data (60%) and validation data (40%) to build and validate the model respectively. Do not use the validation data provided on kaggle.com. Use the Naïve Bayes classifier in Python (we will discuss the theory of Naïve Bayes later in this course. There are two versions of Naïve Bayes – binomial and multinomial; for the moment, let's not bother about the theoretical underpinnings! You can use either one for this assignment). For all models below, show the *confusion matrix*.

Hint: For part B, check out http://www.nltk.org/book/ch06.html (esp 1.3) for illustrations.

Also look at http://nbviewer.ipython.org/gist/rjweiss/7158866 (you may have to download additional Python libraries than the ones I have mentioned before)

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

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.

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.

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?

Deliverables

The deliverable for this assignment is a file with the python scripts and the outputs (including plots & tables where applicable) and your responses to the various questions. Do not forget to write the names of all group members.