Due 03/09 (11:59pm)

Instructions

Submit your solutions on Canvas. Your submission must include the Python code for all questions. Please submit your code all in one file. Adding comments may help the instructor understand your code and is therefore encouraged. **The solution for each question must be generated as a Python variable or output files with specific names as instructed**. All solutions should be generated by running your code without any customization or modification by the instructor. If your code requires input files, you can assume all input files are in the working directory -- **the full path to the file (which changes with computer) should not appear in your code**. Your files should end in .ipynb (Python Notebook).

Problems

1. Download the "sully.csv" file. You will create a WordCloud of the *Sully* reviews using a processed DTM (the processing steps are described below). The maximum word frequency should be 50. Feel free to set the other parameters as you desire (e.g., rotation). Create a DTM from the "review" column, using the following pre-processing steps:
   - Turn every letter to lower case
   - Remove punctuation and numbers
   - Remove stopwords using the default sklearn stopword list ***excluding*** *not* and words ending in *n't* (i.e., the stopword list used should **not** contain *not* and words ending in *n't*)
   - Remove extra whitespace
   - Apply stemming
   - Generate unigrams

   Output variables:
   - `docs1:` (5 points) A Python list containing the *Sully* reviews.
   - `mystopwords:`(5 points) A vector of the default stopwords excluding *not* and words ending in *n't.*
   - `dtm1:`(5 points) A processed DTM (according to the above criteria).
   - `tf_dict1:`(10 points) A Python dictionary with terms as keys and the frequency of each term as the values.
   - `wordcld:`(5 points) A *wordcloud* object that you will call *.generate_from_frequencies(tf_dict1)* on. You will then need to call *plt.imshow(wordcld,interpolation="bilinear")*, then *plt.axis("off")*, and finally *plt.show()*. Please see the lecture example for more details.

2. Apply the same pre-processing steps as in question #1, above, to *docs1* except instead of unigrams you will generate bigrams. You will then use the resulting DTM to generate a histogram.

   Output variables:
   - `dtm2:`(5 points) A DTM consisting of bigrams.

- `Ctf2:`(5 points) A two column pandas DataFrame with terms as one column and corpus term frequencies in the other, sorted from highest CTF to lowest CTF.
- `terms2` and `freq2:`(5 points) A list of the top 20 bigram terms and a list of the top 20 bigram term frequencies from *ctf2*.
- (5 points): A histogram barplot displaying the top 20 most frequent bigrams plotted in descending order (you may make the bars whatever color you'd like, please make sure to label the y-axis "Frequency" and the title as "Bigram Term Frequency").

3. "*Complaints.csv*" contains consumer complaints received by Wells Fargo & Company. Wells Fargo responded to each complaint with explanations; the consumers may or may not agree with the explanations provided. In this file, the column "*Consumer.complaint.narrative*" is the text of each complaint. Use the nltk VADER lexicon and nltk SentimentIntensityAnalyzer to estimate the sentiment score of each complaint. What are the five most negative complaints (use "compound")?

   Output variables:
   - `sentiment_score:`(5 points) A list of compound sentiment scores of all complaints.
   - `doc_score_df:`(10 points) A two column DataFrame with one column as the complaints and the second column as the compound sentiment scores, sorted from lowest compound score to highest.
   - (5 points): Are there any common elements to point out about the five most negative complaints? (note that you can use the line *pd.set_option('display.max_colwidth', None)* to force pandas to display all of the text in a pandas DataFrame column)

4. Apply latent semantic analysis (LSA) to create a topic model for the "Message" column in "*news_label.csv*". Use the entire data without any partitioning. Set *k* (number of topics) equal to 10. Implement the following preprocessing steps when creating the DTM:

Turn every letter to lower case

- Remove punctuations and numbers
- Remove the words in the default sklearn stopwords list
- Strip whitespace
- Apply stemming
- Terms that appear in all of the documents must be discarded (use the max_df parameter)
- Use a customized dictionary that consists of only the unigrams whose total frequency is at least 100.
- Use normalized TFIDF as the weights in the final DTM (i.e., after you obtain the custom dictionary and are creating the DTM that topic modeling will be applied to).

Output variables:

- `term4:` (5 points) A DataFrame of the 10 terms and scores most associated with topic 3.
- `term5:` (5 points) A DataFrame of the 10 terms and scores most associated with topic 5.
- (10 points) Between topics 4 and 5, which is more related to "space" and which is more related to "med".
- (10 points) Which document is most associated to topic 4?