Homework 1: Text Classification
Due date: Anytime on Thursday February 6, 2024

In this assignment, you will be implementing the text classification methods we have studied in class using common NLP/ML packages. This assignment uses social media data from Twitter, consisting of tweets ('text' column of Tweets_5K.csv) that are rated for three categories of sentiment: positive, negative, and neutral. In brief, you will be comparing the performance of Naive Bayes and Logistic Regression classifiers run on text that has or has not been preprocessed.

Notes:

- You may discuss this assignment with classmates. However, all submitted code and answers must be your own work (and you must understand what you submit). Please list the names of anybody you talked to and list any online resources (e.g., Wikipedia) you used while working on this assignment.
- You may **not** use ChatGPT or similar to complete this assignment.
- Please hand in the code and your answers to the conceptual questions. The written answers can either be in the same .py file or in a separate pdf file, but they must be well-annotated and easy to find.
- Note that you will be working with packages in this assignment, but please make sure that you understand what the packages are doing. One important skill when working in this area is being able to Google around and look through documentation to find helpful resources/functions/packages, so that is part of this assignment. That being said, I'm also a resource, so please come to office hours or email me if you have questions.
- Some things that you may or may not find helpful as you work on this assignment:
    - DictVectorizer
    - Accuracy_score
    - Train_test_split
    - Naive Bayes
    - SpaCy
    - NLTK
- Please do not use CountVectorizer, unless you understand what it's doing under the hood! It may lead you astray.
- I highly recommend using Colab, and am happy to answer any questions about it.

Part 0: Paper Exercises

1. But before we turn to coding, please complete Exercise 4.2 from SLP by hand to make sure you understand Naive Bayes theoretically.

Part 1: Preparing Data

1. **Load the data:** Create (i) a list of all Tweets in the dataset, `raw_tweets` and (ii) a list of the sentiments, `labels`, corresponding to each raw tweet encoded as integers (-1 meaning negative, 0 meaning neutral, 1 meaning positive).

2. **Basic preprocessing:** You will start by implementing very basic preprocessing of the tweets, by only splitting the tweets on whitespace ("tokenization"). Create a list `basic_preproc_tweets`, which is a list of preprocessed tweets (which are now lists of words).

3. **Featurize (bag of words):** From your preprocessed tweets, create a bag of words matrix, `basic_preproc_bow`. The rows should be documents and the columns should be words (i.e., features). The cell values should be the number of times a given word occurs in a given document (without smoothing). Note: You will be doing this again later in the homework.

4. **Create a training set and a test set:** You will need to define a training set (used to learn model parameters) and a test set (used for testing your model on unseen documents) - please use a 80%/20% split, meaning that 80% of your available data will be in the training set and the remaining 20% will be in the test set. The data are pre-shuffled, so please make the first 80% of the data the training set, and the last 20% of the data the test set.

Questions to answer and hand-in (you may need to write additional code and/or print statements to answer these questions):

1. What are the dimensions of your feature matrix (X)?
2. What is the value of X[1460][1460]?
3. What does this value mean? What feature does the 1460th column represent?

Part 2: Implementing Naive Bayes

1. **Implement and run Naive Bayes (with add-1 smoothing):** You may do so by hand, if you choose, or using scikit-learn, a commonly used package for machine learning. You can read about implementing Naive Bayes here to find out specifically what calls you should use to create your classifier, train your classifier, and use your trained classifier to make predictions on your test set of unseen data. Report the model's accuracy on the unseen test set. How does this compare to a classifier that always outputs the most frequent category in the training set? Note: There are different forms of Naive Bayes listed (e.g., Gaussian, Multinomial, Complement, etc.) – be sure to use the correct one that we discussed in class.

Questions to answer and hand in:

1. Report the model's accuracy on the unseen test set. How does this compare to a classifier that always outputs the most frequent category in the training set (what is that classifier's accuracy)?

Part 3: Implementing Logistic Regression

1. **Implement and run Logistic Regression:** Now, use scikit-learn to implement Logistic Regression for sentiment analysis on the Twitter dataset. Please set max_iter = 150 or higher. Notice that the calls are similar to those you used for Naive Bayes, one of the benefits of using such packages. Report the model's accuracy on the unseen test set. How does this compare to a classifier that always outputs the most frequent category in the training set, as well as the Naive Bayes classifier.

Questions to answer and hand in:

1. In class, we learned how to do binary logistic regression between 2 options. Here, there are three possible classifications, so sklearn uses a "one vs. rest" scheme where it learns a binary logistic regression model for each possible label (i.e., one logistic regression which learns to separate positive from non-positive tweets, a second that learns to separate negative from non-negative tweets, and a final logistic regression that learns to separate neutral from non-neutral tweets. How many parameters did your multiclass logistic regression model learn?
2. Report the model's accuracy on the unseen test set. How does this compare to (i) a classifier that always outputs the most frequent category in the training set and (ii) the Naive Bayes classifier from Part 2?

Part 4: Your turn

Questions to answer:

1. Propose and implement at least one addition to the workflow that you think will improve performance (but please still use Naive Bayes or Logistic Regression). Specifically, you can add a pre-processing step (e.g., lowercasing, lemmatization, removing stop words, using only the X most frequent words), add a feature (or class of features), etc. Explain why you chose this change and why you think it might help performance. Report the classifier's accuracy on the test set. Did this help as expected? **Note:** You will not be graded on whether your change *actually* improved performance. You may implement this change however you would like, but I will point you to the spaCy or nltk packages, which are popular in NLP.
2. Finally, take the best performing model and look at the tweets that were incorrectly categorized. Do you observe any patterns in what the model is making mistakes on? What do you think could be done to further improve results? Provide a sample of tweets

as well as their true label and their predicted label to justify your answers. **Note:** You do not need to know how to implement the proposed change.