CMPS142-Spring 2018
Homework 3 (Part 1)

Handed out: May 19, 2018
Due: May 26, 2018 at 11:59 PM

- There are no late days allowed for this homework. Your submission will be considered late even if you are late by a second.
- You have to to solve this homework in groups of 2 or 3. Collaborating with any one not enrolled in the class, (except the course staff), or taking help from any online resources for the homework problems is strictly forbidden.
- One (and only one) member of the group has to submit the homework using his/her account on canvas. All group members will get points for that submission.
- How to submit your solutions: Your group's solution to each problem must be typed up separately (in at least an 11-point font) and submitted in the appropriate 'Problem' box on the Canvas website as a PDF file. **This means that if the homework has N problems, you will submit N separate pdf files on Canvas, one for each problem per group!** For example, submit the pdf that contains your group's solution to the first problem in the box titled 'Problem 1'.
- **Each pdf file** should clearly mention the names, email addresses and student ids of all group members. If you forget a group member's name, they will not get points for that problem.
- It is your responsibility to ensure that the files that you submit are not corrupted in any way. After you submit a file, please download it and verify that there aren't any issues. Any related requests will not be entertained after the due date.
- You are very strongly encouraged, but not required, to format your solutions in LATEX. You can use other softwares but handwritten solutions are not acceptable.
- Please try to keep the solution brief and clear.
- The Computer Science Department of UCSC has a zero tolerance policy for any incident of academic dishonesty. If cheating occurs, consequences within the context of the course may range from getting zero on a particular homework, to failing the course. In addition, every case of academic dishonesty will be referred to the student's college Provost, who sets in motion an official disciplinary process. Cheating in any part of the course may lead to failing the course and suspension or dismissal from the university.

Problem 1: Feature Engineering [50 Points]

The first step in Machine Learning is instance *representation* or *feature extraction*. In this question, you will extract useful features for an SMS (text message) spam classification problem. This is a binary classification problem. In this problem, given an SMS you want to build a classifier for identifying whether it is *spam* or *ham* (not-spam). You will be using the dataset provided to you in form of the train (train_file_cmps142_hw3) and the test (test_file_cmps142_hw3) files. Open the file containing the training set. You will see that it contains one line per instance (SMS). The first

word contains the label (*ham*/*spam*), and the rest of the line contains the text of the SMS. As you see, the text is a simple string and before we apply our favorite classifier, we need to extract features for the text. **Remember that features have to be extracted only from the text, and not the labels.**

First, we will pre-process the data to extract features from the training set (without looking at the test set). Then we will extract the same set of features for the test set. We require that you use Python version 2.7 and NLTK version 3.2.2 for this problem. NLTK is a Python library that provides functionality for basic Natural Language Processing. Learn more about NLTK here: `https://www.nltk.org/` Please note that you might not get the answer we expect from you if you use different version and in that case, you will not get full credit. Also, please retain the order of instances in the test and training sets in your pre-processing.

Finally, you may run into some encoding issues when preprocessing dataset files. To avoid this, you need to set the default encoding to 'uft-8' as following:

```
reload(sys)
sys.setdefaultencoding('utf-8')
```

1.1 Preprocessing the Training set [20 points]

Do the following steps (in order) for the **text** of each SMS in the **training set**:

1. **STEP 1: Convert to lowercase** the text of each SMS. Remember that your ML classifier doesn't know English. Lowercasing the text helps it identify that *Hello* is the same as *hello*.

2. **STEP 2: Tokenize** the text of each SMS. Tokenization is the process of splitting a piece of text (like a sentence) into tokens (words). You can learn more about it here: `https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html`. We require that you use NLTK's tokenizer for this task. You can access it using `from nltk import word_tokenize`. Notice that after tokenization, you should have converted the string of text for each SMS into a *list* of tokens. Answer the following questions about the collection of these lists of tokens in your report (after applying Steps 1 and 2).

   (a) [1 point] What is the total number of *distinct* token-types in your training set after this step? Notice that the question is NOT asking for the total number of tokens, but the number of different types they belong to (after you remove repetitions). E.g., in the following sample text corpus: $\langle$*very*, *very*, *small*, *corpus* $\rangle$, there are 4 tokens and 3 token-types.

3. **STEP 3: Remove Stopwords**. Stopwords are words that occur very frequently in text data, but are usually not very informative like *is, a, an, the,* etc. In text processing, it is common to remove these words from the input in order to help the classifier. NLTK has a list of English stopwords which can be accessed as:
   ```
   from nltk.corpus import stopwords
   print stopwords.words('english')
   ```
   Remove stopwords from every SMS's text.

4. **STEP 4: Remove Punctuations**. Punctuations may or maynot be informative (depending on the task), but in this question you will remove all punctuations from the text. For consistency, you will be using Python's list of punctuations. It can be accessed as:
   ```
   import string
   print string.punctuation
   ```
   Remove punctuations from every SMS's text.

5. **STEP 5: Stemming** removes morphological affixes from words, leaving only the word stem. Apply NLTK's Porter Stemmer to each token in the text of each SMS. You can access PorterStemmer as: `from nltk.stem.porter import PorterStemmer`. Answer the following questions in your report after applying Steps 1 to 5:

   (a) [5 points] One of the SMSs in the training set reads: 'I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.'. What is the list of tokens that you get for this SMS?

6. **STEP 6: Ignoring infrequent tokens.** Throw away all token-types that appear less than 5 times in the entire training corpus. In other words, if the number of occurrences of a token-type in the training corpus is < 5, remove all occurrences of the token-type.

(a) [2 points] What is the total number of *distinct* token-types in your training set after this step? This set of all distinct token-types is usually called the *Vocabulary*, V. In this question you are reporting the size of your vocabulary, $|V|$.

7. **STEP 7: Feature-vector representation.** In this last step, you will convert the processed text (from Steps 1 through 6) of each instance (SMS) into a feature vector representation to be consumed by a Machine Learning classification algorithm. For this question, we will design the feature vector such that there is one feature for each word in the Vocabulary, i.e. the dimensionality of this feature vector is $|V|$ and the i^{th} dimension corresponds to the i^{th} word in your vocabulary. Also, the value of the i^{th} element of the feature vector for an SMS represents the **count** of the i^{th} word. This representation is commonly referred to as a *Bag-of-words* representation because each text is represented as simply a collection of words (and the representation does not model the order of words in that text).

   (a) [8 points] Submit your processed training set as a comma separated file. It should have $|V| + 1$ columns: one each for the words in your vocabulary, and the last column for the label. Also, it should have exactly 4459+1 rows. The first row should be the header, and after that there should be one row per instance (SMS). Please retain the order of instances from train_file_cmps142_hw3. Naturally, the header has $|V| + 1$ columns. The last column should be named *label*, and names of each of the first $|V|$ columns should be the token-type from the vocabulary that it represents. Name your file as HW3_lastName_train.csv, where you would replace 'lastName' with the last name of the person who is submitting the report. In your report, as a response to this part of the question, write the name of the file you submitted.

   (b) [1 point] Does you header have a column named 'yellow'?

   (c) [1 point] Does you header have a column named 'music'?

   (d) [1 point] What is the sum of the second row (excluding the 'label' column)? Note that the second row corresponds to the first SMS whose text was "Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat..."

   (e) [1 point] How many columns in your file sum to 0?

1.2 Processing the test set [20 points]

You will now pre-process the test set (file test_file_cmps142_hw3).

1. **STEP 1: Preprocessing SMS texts.** For each SMS text in the test set, do the following: (1) convert the text to lowercase, (2) tokenize, (3) remove stopwords, (4) remove punctuations, and (5) apply Porter Stemmer as before. Use the same functions, tools etc. that you used for processing the training set.

   (a) [5 points] One of the SMSs in the test set reads: 'Hi. Wk been ok - on hols now! Yes on for a bit of a run. Forgot that i have hairdressers appointment at four so need to get home n shower beforehand. Does that cause prob for u?"'. What is the list of tokens that you get for this SMS?

2. **STEP 2: Feature-vector representation.** Now, represent each test instance in the *same* feature space and representation as the train instances. For this problem, the feature space is the vocabulary space. Answer the following questions about test set processing in your report (all of them are about the csv file you generate in this part of the question):

   (a) [8 points] Submit your processed test set as a comma separated file. Name your file as HW3_lastName_test.csv, where you would replace 'lastName' with the last name of the person who is submitting the report. In your report, as a response to this part of the question, write the name of the file you submitted.

   (b) [1 point] What is the number of rows in your test file (including the header row)?

   (c) [1 point] What is the number of columns in your test file (including the last column for the label)?

   (d) [1 point] What are the names of the first 5 columns (in order) in your train file, HW3_lastName_train.csv, and the names of the first 5 columns (in order) in your test file, HW3_lastName_test.csv?

   (e) [1 point] Does your header contains the token 'head'?

   (f) [1 point] How many columns in your file sum to 0?

   (g) [2 points] Why is it important to represent the test instances in the same feature space as the training instances?

1.3 Submitting your code [10 points]

Submit your code for pre-processing and feature extraction as a .zip file named HW3_lastName_code.zip, where you would replace 'lastName' with the last name of the person who is submitting the report. The code should be well-documented (should have descriptive variable names and comments clearly indicating which step the code is performing and what file, train or test, is it processing). It should have a ReadMe file, which should explain how to run your code. Do not put your report or the processed train and test files inside this zip file. Submit them separately. In your report, as a response to this part of the question, write the name of the zip file you submitted.