Homework 2: Triage [starter code]

INTRODUCTION

Victims of natural disasters have urgent needs for food, water, shelter, medicine, and other forms of aid. These needs are often communicated through text messages, social media posts, and local newspapers. Because of their ability to automatically process large amounts of text, NLP techniques can play an important role in ensuring that people receive potentially life-saving aid.

Your goal for this homework is to perform **Text Classification** on messages sent in the aftermath of natural disasters. Specifically, you will determine **whether or not a message is about aid.**

DATA

The data for this assignment contains about 26K documents from several major natural disasters:

- Earthquake in Haiti (2010) (https://en.wikipedia.org/wiki/2010_Haiti_earthquake) (Links to an external site.)
- (Links to an external site.)Floods in Pakistan (2010) (https://en.wikipedia.org/wiki/2010_Pakistan_floods (Links to an external site.))
- Earthquake in Chile (2010) (https://en.wikipedia.org/wiki/2010_Chile_earthquake (Links to an external site.))
- Hurricane Sandy in North America (2012) (https://en.wikipedia.org/wiki/Hurricane_Sandy (Links to an external site.))
-

These documents are either text messages, social media (Twitter) posts, or snippets from news articles. In addition to the specific events listed above, the dataset contains a number of news articles spanning dozens of different disasters. All messages have been translated and annotated by humans on the crowdsourcing platform CrowdFlower (now Figure-Eight). However, some of the translations are not perfect, and you may encounter some words in other languages. Unfortunately, NLP researchers often have to work with "messy" data. If you're curious about the crowdsourcing translation effort for messages from Haiti in particular, feel free to check out this paperLinks to an external site..

Your task is to classify each document as being aid-related (class "**aid**") or not aid-related (class "**not**"). Messages that are aid-related include individuals' requests for food/water/shelter/etc. The **aid** class also includes news reports about dire situations and disaster relief efforts.

Below are several examples of aid-related documents (belonging to class "**aid**"):

- Hello Good Morning We live on 31 Delmas we are without water without food and what we had have finished Please do something for us!
- I am sending this SMS from Layah district for my sister whose house has got destroyed in a flood. So, the problem she faces now is that she hasn't got any 'Watan Card'or any financial aid from the government. She has 5 children too.
- Redcross came to my house and gave my family food ... Guess were not getting power anytime soon . #sandy #RedCross
- Relief officials have stressed the vital importance of bringing in clean drinking water and sanitation equipment to avoid deadly epidemics that in a worst case scenario could claim as many or more lives than the tsunami itself.
-

Below are several examples of non-aid-related documents (belonging to class "**not"**):

- A cold front is found over Cuba this morning. It could cross Haiti tomorrow. Isolated rain showers are expected over our region tonight.
- Hurricane : A storm which New Yorkers use as an excuse to drink and eat junk food in their pajamas for 48 hours . #sandy
- By secret ballot, the Council elected Pakistan, Bahrain and the Republic of Korea from the Asian States, while Iran and Saudi Arabia did not receive enough votes to qualify.
-

The data is divided into a **training set**, **development (validation) set**, and **test set**. Recall that the training set is used to compute the statistics for your model. These statistics are then used to classify the documents in the development and test sets. For this assignment, you have access to the training set and the dev set. The test set is hidden, but your submission will be evaluated on it as well.

- The training set is located in: **data/train/**
- The dev set is located in: **data/dev/**
- Within each of these directories, documents belonging to class aid are located in **aid.txt**. Documents belonging to class not are located in **not.txt**. Every line in these text files is another document.
-

Finally, note that the data you are given is already **preprocessed**; all punctuation has been removed (except hashtags and apostrophes) and all text has been converted to lowercase. Depending on the specific NLP task, preprocessing can significantly improve performance. You do not need to do any additional preprocessing.

ALGORITHM

You will be using **Naïve Bayes**, following the pseudocode on page 7 of Chapter 4 of Jurafsky and Martin (3rd edition ms), using Laplace (add-1) smoothing. Your first classifier will use words as features, add the log-prob scores for each token, and make a binary decision between **aid** and **not**. In addition to this, you will explore the effects of **stop-word filtering**. This means removing common words like "the", "a", and "it" from train, dev, and test sets. We have provided a stop list with the starter code in the file:

```
data/english.stop
```

Finally, you will implement a version of Naïve Bayes that uses **bigram features** instead of bag-of-words (unigram) features. **Note**: Depending on the dataset and potentially due to factors like overfitting, it is not guaranteed that bigram features always perform better.

ASSIGNMENT

Train a **Naïve Bayes** classifier on the disaster relief data set provided with the starter code. During the training phase, you use the fact that a document is aid-related or not (the hand-labeled "true class") to help compute the correct statistics. During the testing phase, you only use this label to compute your accuracy.

Task 1- Your first task is to implement the classifier using unigram (bag-of-words) features and Laplace smoothing. Add your code in the **addExample()** function to train your model. You will predict whether a given document is aid-related or not in the **classify()** function.

Task 2- Next, evaluate your model again with the stop words removed. Does this approach affect accuracy (for the current given data set)?

Task 3- Next, implement a version of the Naive Bayes classifier that uses bigram features instead of unigram features.

- Hint 1: Remember to add a start token and an end token in order to capture bigrams containing the first and last words.
- Hint 2: When implementing add-1 smoothing with bigram features, V = # of unique bigrams.
-

WHERE TO MAKE YOUR CHANGES

To ensure that your code works properly not only when run from the command line but also when executed by our autograder, you should limit your changes to addExample() and classify() methods within NaiveBayes.py. You will also need to add some data structures for your work; where to add them will be mentioned in the start code. Apart from adding data structures, changes beyond addExample() and classify() are not necessary. Importantly, note that main() is **NOT** executed by the autograder so you cannot rely on anything added there.

The autograder also directly manipulates the booleans FILTER_STOP_WORDS and USE_BIGRAMS. If the FILTER_STOP_WORDS switch is True, then your code should handle stopword removal. Similarly, if the USE_BIGRAMS switch is true, your code should build a model that uses bigram features instead of unigram features.

EVALUATION

Your classifier will be evaluated on the training, development and held-out/unseen test sets from the Disaster Relief data. All three versions of your classifier (bag-of-words, stopword filtering, and bigrams) will be evaluated.

RUNNING THE CODE

In your terminal, execute

```
$ python NaiveBayes.py
```

This requires the NaiveBayes.py file to be located in the same directory as **data/**. This is the default in the starter code.

Adding a flag (-*f* or-*b*)...

```
$ python NaiveBayes.py -f
```

Flag -f invokes stop word filtering, and will set the FILTER_STOP_WORDS boolean to True.

```
$ python NaiveBayes.py -b
```

Flag -b involves your bigram implementation, and will set the USE_BIGRAMS boolean to True.

These commands will output the model's performance on the train and dev sets.

Attention: Please use these flags (-f and -b) one at at time. Using both together will lead to only one flag actually taking effect. This is done because stopword removal will be evaluated in isolation from bigrams.

SUBMITTING YOUR SOLUTION

Submit your assignment via **Gradescope** (www.gradescope.com). We expect the following files in your final submission:

- NaiveBayes.py

You will only be able to see your score on the dev set. The test set scores will be released after the deadline.