CSCI 3907/6907: Introduction to Statistical NLP

Assignment #3

Instructions:

- ✓ This assignment is due on **November 15, 2020, by 11:59 pm**. You have 7 days grace period for the entire semester, afterwards 10% of your grade will be deduced for every late day.
- ✓ Submit your solution in a zipped folder through blackboard.
- ✓ All code must compile and run to receive full credit for coding parts.
- ✓ Include citations for any online resources used or group discussions.

In this assignment we will be working on a Twitter dataset designed for detecting hate speech. The dataset has 32K tweets labeled as hatred, 1, or non-hatred, 0, and includes 2 files, one for training and one for testing datasets, as provided by Analytics Vidhya. The dataset can be downloaded from Blackboard.

Part 0: [5 points]
To start working on this assignment you need to download the dataset and preprocess the text to prepare it for this assignment. Make sure all words are in lower case and no special characters like # or @ are available.
Remove URLs, Hashtags, Mentions, Reserved words (RT, FAV), Emojis, and Smileys.
Note that the provided data is already split into train and test dataset.

Part 1:
Vector Space [15 points]
Refer to the demo code in lecture 8 and use it to:

1- Find the most common words to the words 'hate' and 'like'.
2- Use the FreqDist from NLTK library to find the most 10 common words used for hatred sentiments and non-hatred sentiments and bar plot their distribution.
3- Word2Vec vectors have been shown to sometimes exhibit the ability to solve analogies. As an example, for the analogy "king" - "man" + "woman" gives a queen as most common. This can be implemented using genism as follows
model.wv.most_similar(positive=['king', 'woman'], negative=['man'])
use this concept to solve the analogy for 'love'+'life'-'hate'

Part 2:
Sentiment Analysis
In this part, you will use **scikit-learn**, a machine learning toolkit in Python, to implement text classifiers to detect hate speech in tweets. Please read all instructions below carefully.

Please use the following:

a) The demo code provided to you in lecture 6 as a starter code

b) Chapter 4 in the book as a reference,
https://web.stanford.edu/~jurafsky/slp3/4.pdf

c) Scikit-learn modules in your implementation with reference to the following scikit-learn documentation:

Bag-of-words (or ngrams) feature extraction using CountVectorizer:
http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
Use binary features (1/0 rather than counts).

Naïve Bayes classifier:
http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

Logistic Regression classifier:
http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

Cross validation:
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.cross_validate.html

Classification report:
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

1- Using the scikit-learn modules described above, Implement the following models:
   a) [10 points] A Naïve Bayes classifier with add-1 smoothing using binary bag-of-ngrams features (with unigrams and bigrams).
   b) [10 points] Logistic Regression classifier with L2 regularization (and default parameters) using binary bag-of-words features.
   c) [10 points] Logistic Regression classifier with L2 regularization using binary bag-of-ngrams features (with unigrams and bigrams).

2- Report the performance (accuracy and F1) for each classifier. [10 points].

3-[optional bonus question – 10 points]

In this part, you are asked to implement a model that combines the advantages of generative and discriminative models: a logistic regression classifier with Naïve Bayes features.

In Naïve Bayes, we used the MLE probabilities of words/bigrams given a class label. We can use the ratio of these probabilities as features in logistic regression. Define the count vector for the hatred class **p** as the smoothed sum of features for all instances that belong to that class:

$$p = 1 + \sum_{i:y(i)=1}^{.} f(x_i)$$

where $f(x_i)$ is the binary feature vector for example x_i. Each element in p is the count for a specific word/bigram with add-1 smoothing. Similarly, the count vector for the non-hatred class **q** is defined as:

$$q = 1 + \sum_{i:y(i)=0}^{.} f(x_i)$$

The log-count ratio is defined as:

$$\mathbf{r} = \log\left(\frac{\mathbf{p}/||\mathbf{p}||_1}{\mathbf{q}/||\mathbf{q}||_1}\right)$$

which is the ratio of the hatred to non-hatred to non-hatred likelihoods for each word/bigram (the $||x||_1$ notation is the L_1 norm, which is the vector sum since all features are nonnegative). Now instead of using the binary feature vectors $f(x_i)$ as input to the logistic regression classifier, use the element-wise product of the log-count ratio vector and each feature vector: $r \odot f(x_i)$
Report the performance of this model on the CR dataset using the two sets of features: (a) bag-of-words only, and (b) bag-of-ngrams (unigrams and bigrams).