EL-GY-9133 Machine Learning for Cyber-Security

Lab 1: E-mail Spam Filtering

Release Date: 10/01/2020; *Due Date*: Midnight, 10/16/2018

Overview

In this lab, you will design an e-mail spam filter using a Naïve Bayes and SVM based classification on the ling-spam dataset. You will explore the impact of feature selection and compare the performance of different variants of an NB classifier and also implement your own SVM based classifier.

Dataset

The ling-spam corpus contains e-mails from the Linguist mailing list categorized as either legitimate or spam emails. The corpus is divided into four sub-folders that contain the same emails that are pre-processed with/without lemmatization and with/without stop-word removal. The e-mails in each sub-folder partitioned into 10 "folds."

In this lab, we will use the first 9 folds from the ling-spam corpus as training data, and the 10^{th} fold as test data.

What You Have to Do

You will implement your e-mail spam filters in Python. You are free to use any Python libraries that are relevant to the problem.

- Download the ling-spam dataset from http://www.csmining.org/index.php/ling-spam-datasets.html Please use the "lingspam_public01" corpus with both lemmatization and stop-word enabled (under the lemm_stop folder).

- Your first goal is to perform feature selection using the information gain (IG) metric. From the training data, select the top-N features for N = {10, 100, 1000}. Note that feature selection based on the IG metric only accounts for the occurrence of (and not frequency with which terms appear) in the dataset.

- Next, implement the following classifiers:
    - Bernoulli NB classifier with binary features;
    - Multinomial NB with binary features; and
    - Multinomial NB with term frequency (TF) features.

- For each of the three classifiers above and for N = {10, 100, 1000} report the spam precision and spam accuracy. Spam precision is defined as the fraction of true spam e-mails among all e-mails predicted as spam, and spam recall is defined as fraction of true spam e-mails predicted as spam.

- Design a Support Vector Machine (SVM) based spam filter. This problem is open ended: for instance, you can choose to use either BF or TF and any feature selection method. Note that you should *NOT* use the test dataset in picking the hyper-parameters of your spam filter; instead use cross-validation on the training dataset.

- Finally, In Lecture 4/5 we covered "Adversarial Classification" (https://dl.acm.org/citation.cfm?id=1014066), an approach to update NB based e-mail spam filters in response to attacks that try to evade a basic NB filter. You are expected to implement the techniques presented in this paper. More specifically, you can make the following assumptions:
  - The baseline NB classifier uses the **top-10** terms identified using the IG metric and using Boolean features.
  - The adversary uses the **ADD-WORDS** strategy. Adding a term to an email incurs unit cost. The attacker seeks to find the *minimum cost* solution such that each spam email in the test set gets classified as legitimate by the baseline NB classifier.
  - Update the baseline NB classifier in response to the attacker's strategy above. You can assume that the defender pays a unit price for both false positives and false negatives.

What to Submit

- Your Python code in the form of a Google Colab notebook. You will be submitting your assignment on NYU CLASSES (since we need to use that for grading). More details to follow.
- Your Colab notebook should print:
  - a list of the top-10 words identified from Part (1) above, and
  - a list of spam precision and spam recall values for each of the three classifiers for N = {10, 100, 1000}. That is, your list should have 9 rows, one for each classifier and N combination.
  - For the SVM based spam filter, please describe your methodology, i.e., what kind of features you used, how many features you used and how you selected them, the parameters of the SVM and finally the precision and recall on the test dataset.
  - For the adversarial attack, report the False Negative rate of the baseline NB classifier before and after the attacker's modifications to test emails. Also, report the average "cost" of the attacker's modifications, averaged over all spam emails in the test set. Finally, report the False Negative and False Positive rate of the updated NB classifier.