Assignment 1
E-Mail SPAM Detection

Preface: Spam email is unsolicited and unwanted junk email sent out in bulk to an indiscriminate recipient list. Typically, spam is sent for commercial purposes. It can be sent in massive volume by botnets, networks of infected computers – CISCO.

Spam detection is one of the oldest computer security challenges and the one which remained relevant throughout the history of E-mail communication and shall continue to do so in future as well. Machine Learning has shown great promise in addressing spam detection and is widely used for this purpose.

Problem Statement

In this assignment, you are going to build your own custom spam detection system. Assume that you are working as security engineer at some organization and you are asked to solve the problem of rampant email spam affecting employees in the organization. For whatever reason, you are instructed to develop a custom solution instead of using commercial options.

Provided with administrator access to the private email servers, you can extract a body of emails for analysis. All the emails are properly tagged by recipients as either "spam" or "ham" (non-spam), so you do not need to spend too much time cleaning the data.

An intuitive method to deal with this problem is to scan through all the email messages and check for words which are very often used by spam messages. There is a pattern associated with spam messages. If an email contains such words, then you can identify or classify it as a spam otherwise as ham. For instance, you notice that the word "lottery" appears in the spam data a lot, but seldom appears in regular emails.

Before using any known Machine Learning models, you are required to first develop a naïve algorithm which will classify a message between spam or ham using a "blacklist" of words.

Task 1: Develop a naïve classification algorithm which will classify spam and ham using a blacklist. Hint: You may think about creating the blacklist with words which are present only in spam messages.

1) Download 2007 TREC Public Spam Corpus. This is a lightly cleaned raw email message corpus containing 75,419 messages collected from an email server over a three-month period in 2007. One-third of the dataset is made up of spam examples, and the rest is ham.
2) Split dataset into nonoverlapping training and test sets, in which the training set consists of 70% of the data (an arbitrarily chosen proportion) and the test set consists of the remaining 30%.

3) Using the training data set create a blacklist of words and finally test it using test data. Calculate the prediction accuracy and confusion matrix.
    a. To create the blacklist of words, you will have to use various libraries from Natural Language Toolkit (NLTK), such as classification, tokenization, stemming etc.
    b. To support some of the pre-processing a python file (email_read_util.py) is attached which contains certain helper functions.

You will observe that the classification accuracy of your naïve classification method using blacklist is decent but not as good as commercial products. Therefore, you now have to use some known Machine Learning method to improve the accuracy.

Task 2: Employ Naïve Bayes Classifier to develop a spam detection system using same dataset. To do that, you will be required to pre-process the data and make it usable for the naïve bayes classifier to accept.

a. Scikit-learn provides a simple class, sklearn.naive_bayes.MultinomialNB, that you can use to generate quick results for this experiment.
b. You can reuse a lot of the earlier code for parsing the email files and pre-processing the labels.
c. However, if you decide to try passing in the entire email subject and plain text body (separated by a new line) without doing any stopword removal or stemming with NLTK. You can use the function shown below (do not forget to import email_read_util.py)

```
def read_email_files():
    X = []
    y = []
    for i in xrange(len(labels)):
        filename = 'inmail.' + str(i+1)
        email_str = extract_email_text(os.path.join(DATA_DIR, filename))
        X.append(email_str)
        y.append(labels[filename])
    return X, y
```

d. To convert a body of text into a feature vector is to use the **bag-of-words** representation, which goes through the entire corpus of documents and generates a vocabulary of tokens used throughout the corpus vocabulary comprises a feature, and each feature value is the count of how many times the word appears in the corpus. Check sklearn.feature_extraction.CountVectorizer class

    For example, consider a hypothetical scenario in which you have only three messages in the entire corpus:

```
tokenized_messages: {
                        'A': ['hello', 'mr', 'bear'],
                        'B': ['hello', 'hello', 'gunter'],
                        'C': ['goodbye', 'mr', 'gunter']
                    }
# Bag-of-words feature vector column labels:
# ['hello', 'mr', 'doggy', 'bear', 'gunter', 'goodbye']
vectorized_messages: {
                        'A': [1,1,0,1,0,0],
                        'B': [2,0,0,0,1,0],
                        'C': [0,1,0,0,1,1]
                    }
```

e. You can also try using the term frequency/inverse document frequency (TF/IDF) vectorizer instead of raw counts. TF/IDF normalizes raw word counts and is in general a better indicator of a word's statistical importance in the text. It is provided as sklearn.feature_extraction.text.TfidfVectorizer

f. Finally test your model and report classification accuracy, false positive and false negative.

Submission Note:

1) Deadline: 11:59 PM on 13-02-2022
2) Update the Leader Board whenever you have the results: https://docs.google.com/spreadsheets/d/155daoECZoM3ain0jLzLbmHDy4y0T3hxs-eQV2EuJ5Uw/edit?usp=sharing
3) Submit your complete codes.
4) Submit a Report explaining your approach and snapshots of the results.
5) Strictly all submissions should be done on Moodle.
6) Late submissions will incur 50% penalty.

Other Important Resources for Beginners:

1) Tutorial on Fundamental of Machine Learning
   a. https://www.youtube.com/watch?v=PPLop4L2eGk

2) Tutorial on Naïve Bayes:
   a. https://scikit-learn.org/stable/modules/naive_bayes.html
   b. Video: https://www.youtube.com/watch?v=j1uBHvL6Yr0

3) Email System - https://www.youtube.com/watch?v=tfPfwDrfSP8