# Naive Bayes Spam Filter

CMSC 197, Machine Problem 3

---

Deadline: October 8, 2024 @ 11:59 AM

**MP3: Naïve Bayes Spam Filter**

In this problem set, you will be creating a spam classification model using Naïve Bayes Classifier. You will apply the classifier for a subset of the TREC06 Public Spam Corpus – dataset for spam algorithms. The zip file containing the dataset with its corresponding label is uploaded to LMS.

Emails within repositories 0-70 belong to the train set and the rest belong to the test set.

**Instructions for Submission:**

Submit a pdf report that addresses the questions (at the end of this page). A github link of the jupyter notebook of your code solution must be embedded on the pdf report. Make sure that each code cell is executed before saving and/or committing it to github.

If you have additional analysis and experiments, you could include that in the report. Filename should be hw4-surname.pdf. Examples: hw4-ambita.pdf. hw4-delaCruz.pdf.

You can use the sklearn package for

- preprocessing the data (standardization and dividing into train and test sets)
- evaluate the model
- comparison with naïve bayes of sklearn

But you can't use it for

- Fitting the model (You must implement it from scratch)
- Evaluating the model using accuracy, precision, recall, etc. These metrics must be implemented from scratch.

Packages you're allowed to use: numpy, pandas, matplotlib, email

# Naïve Bayes Spam Classifier

Naive bayes assign some patterns to certain features. For the case of spam classification, the classifier assigns whether a certain email is spam or ham. The model typically uses a bag of words to identify spam email. Every word has a certain probability of occurring in spam or ham database. If the occurrence of the word is greater in spam compared to ham, then an email is probably a spam.

A naïve bayes filter estimates the joint probability over a set of features and the classes by making the assumption that each feature is conditionally independent. Firstly, we break the email $D$ we want to classify into a set of individual tokens $w_1, w_2, w_3, \ldots, w_d$ extracted from the training set after parsing and removing the stop words, each having a

spam/ham emails. The probability of email D is equal to the probability of receiving the list of tokens $w_1, w_2, w_3, \ldots, w_d$.

$$P(D) = P(w1, w2, w3, \ldots, wd) .$$

Bayesian combines the prior probabilities of individual word/tokens to obtain the overall score show in the following formula. Therefore by this naïve Bayes assumption becomes:

$$P(D) = \prod_{i=1}^{d} P(w_i)$$

Now, we have two sets of labels $w$ containing spam and ham. The assumption is that a particular token is independent of the other tokens given that we have either spam or ham. We express the probability of a given email that it is a ham or spam by the given formula:

For the probability of an email given ham:

$$P(w1, w2, w3, \ldots, wd \mid c = H) = \prod_{i=1}^{d} P(w_i \mid c = H)$$

For the probability of an email given spam:

$$P(w1, w2, w3, \ldots, wd \mid c = S) = \prod_{i=1}^{d} P(w_i \mid c = S)$$

# Method of Implementation

**Preprocessing (10 points)**

Split the dataset into three(3) groups: training set for ham, training set for spam, and the testing set.

Folders 0-70: Train Set

Folders 71-127: Test set

Train sets included 21, 300 emails and 16, 522 emails for the test set. The labels contained in the label file was attached to its corresponding email.

Remove words from the document which may not contribute to the information we want to extract. These includes dropping the alphanumeric characters and punctuation marks.

Also, remove stop words, more popularly known as meaningless words, from the email body since those words are not useful in classification as well as reduce the dimensionality of the dictionary. A text file with filename, **stop_words.txt**, is also uploaded in LMS.

After doing these methods, extract a list of unique words from the training set along with its summed number of occurrences from the spam and ham set. To limit the cardinality of the dictionary, we can extract only the 10000 most common words (common means that these words have the highest frequencies/occurences in the dataset).

***Note**: You can use the "email" package in python to extract the body of the email. Watch out for problems in encodings and use the email charset aliases accordingly.*

### Creating the feature matrices (5 points)

Create feature matrices for both the spam training set and ham training set with a dimensionality of 10000 (this was the specified number of different words in the dictionary). For each word in the dictionary, traverse through each file and check its occurrence. 1 denotes the existence of a word in the email and 0 otherwise. Thus, a matrix whose rows denote the number of files of training/testing set and columns denoting 10000 words will be generated. Every email in the training as a feature vector.

Consider that the higher the cardinality of the vocabulary, the higher the dimensionality of the matrix. The matrix grows with respect to the number of emails considered. Hence, slowing the program.

### Computing the Priors (5 points)

The prior probabilities for spam and ham are computed by the following formula:

$$P(c = ham) = \frac{N_{ham}}{N_{doc}}$$

$$P(c = spam) = \frac{N_{spam}}{N_{doc}}$$

Where $N_{ham}$ is the number of spam emails in the training set, $N_{spam}$ is the number of spam emails in the training set, and $N_{doc}$ is the total number of emails

### Computing the Likelihood of each word (15 points + 5 points for Laplace smoothing)

A vector containing the number of occurrences of each word in the dataset will be created. One vector will contain the number of occurrences of each word in the ham vocabulary

and another vector for the spam vocabulary. Thus for word $w_i$, its probability of occurring as spam or ham is

$$P(w_i|spam) = \frac{count(w_i,\ spam)}{(\ \sum_{w \in V} count(w, spam)}$$

$$P(w_i|ham) = \frac{count(w_i,\ ham)}{(\ \sum_{w \in V} count(w, ham)\ )}$$

Where $count(w_i,\ spam)$ and $count(w_i,\ ham)$ refer to the number of times the word appears in spam/ham dictionary and $(\ \sum_{w \in V} count(w, spam)$ and $(\ \sum_{w \in V} count(w, ham)\ )$ refer to the total number of occurrences of each word in spam/ham emails. *Note*: $(\ \sum_{w \in V} count(w, c)$ not equal to the number of ham/spam messages, and it's not equal to the total number of unique words in spam/ham messages.

However, there are cases when we encounter a word in a dictionary which is not included in the trainset. In that case the probability of a word occurring given the classification will be 0 which will make the probability of the class given the word also equal to 0. To address the problem, lambda smoothing is introduced where a lambda is added to the numerator and add lambda times the number of words in the vocabulary. A slight modification to the formula above is added. For instance,

$$P(w_i|c) = \frac{count(w_i,\ c) + \lambda}{(\ \sum_{w \in V} count(w, c)\ ) + \lambda|V|}$$

Since we are adding a value $\lambda$ in the numerator, $\lambda|v|$ is added to the count of the documents belonging to the class in order to make the resultant sum of all the probabilities of words in the spam emails as one.

If $\lambda$=1, it's called as laplace smoothing.

We then have the following formula for computing the likelihoods of each word, with laplace smoothing.

$$P(w_i|spam) = \frac{count(w_i,\ spam) + \lambda}{(\ \sum_{w \in V} count(w, spam)\ ) + \lambda|V|}$$

$$P(w_i|ham) = \frac{count(w_i,\ ham) + \lambda}{(\ \sum_{w \in V} count(w, ham)\ ) + \lambda|V|}$$

## Classifying the emails (10 points + 10 points for computing the log probabilities)

Based on the probabilities of words given the label computed, determine the probability of ham and the probability of spam given the document. Whichever has a

higher value will be the classification of the email. Baye's rule was applied to the computed likelihoods.

$$P(c = H \mid w) = \frac{\coprod_{i=1}^{d} P(w_i|c = H)P(c = H)}{\coprod_{i=1}^{d} P(w_i|c = H) \cdot P(c = H) + \coprod_{i=1}^{d} P(w_i|c = S) \cdot P(c = S)}$$

$$P(c = S \mid w) = \frac{\coprod_{i=1}^{d} P(w_i|c = S)P(c = S)}{\coprod_{i=1}^{d} P(w_i|c = H) \cdot P(c = H) + \coprod_{i=1}^{d} P(w_i|c = S) \cdot P(c = S)}$$

If $P(S \mid w_d) > P(H \mid w_d)$, it is classified as spam and ham otherwise. Since the denominators of the formula to solve $P(S \mid x_d)\ and\ P(H \mid x_d)$ are similar, we can omit it from our computation. This further simplifies the equation.

However, multiplying probabilities, which are between 0 and 1 by definition, can result in floating-point underflow. Since log(xy) = log(x) + log(y), it is better to perform all computations by summing up logs of probabilities rather than multiplying probabilities. Class with highest un-normalized log probability score is still the most probable. We take the logarithm on both sides as shown below:

$$\log\left(P(c \mid w_d)\right) = \log\left(\coprod_{i=1}^{d} P(w_i|c)P(c)\right)$$

This derives the sum of the likelihood probabilities of each word and the class prior probability:

$$\log\Big(P(c \mid w_d)\Big) = \sum_{i=1}^{d} \log(w_i|\mathrm{c}) + \log\left(P(c)\right)$$

**Testing the Classifier (10 points)**

After you're done with the Naïve Bayes Classifier, implement code for classifying an unknown message and try it on the test set.

## Performance Evaluation (10 points)

In order to test the performance of above mentioned six methods, filter the accuracy namely percentage of messages classified correctly. Table below shows the evaluation measures for spam filters.

| Evaluation Measure | Evaluation Function |
|---|---|
| Accuracy | $Acc = \frac{TN + TP}{TN + TP + FP + FN}$ |
| Recall | $r = \frac{TP}{TP + FN}$ |
| Precision | $P = \frac{TP}{TP + FP}$ |

Where accuracy, recall, precision, *FP*, *FN*, *TP*, and *TN* are defined as follows:

Accuracy – percentage of correctly identified spam and ham emails
Recall – percentage of spam emails that are correctly classified as spam
Precision – number of relevant documents identified as a percentage of all documents identified. This shows the noise that the classifier presents to the user (i.e. how many of spam emails will actually be classified as spam).
False Positive Rate(FP) – number of misclassified ham emails
False Negative Rate(FN) – number of misclassified spam emails
True Positive Rate(TP) – correctly classified spam emails
True Negative Rate(TN) = correctly classified ham email

# Results and Discussion

**Guide questions:**

1. What is the effect of removing stop words in terms of precision, recall, and accuracy? Show a plot or a table of these results.
2. Experiment on the number of words used for training. Filter the dictionary to include only words occurring more than k times (1000 words, then k > 100, and k = 50 times). For example, the word "offer" appears 150 times, that means that it will be included in the dictionary.
3. Discuss the results of the different parameters used for Lambda smoothing. Test it on 5 varying values of the $\lambda$ (e.g. $\lambda$ = 2.0, 1.0, 0.5, 0.1, 0.005), Evaluate performance metrics for each.
4. What are your recommendations to further improve the model?

# References

[1] Hovold, John. Naive Bayes spam filtering using word-position-based attributes and length-sensitive classification thresholds, 2005.

[2] Doshi, H., Zalte, M. Performance of Naïve Bayes Classifier – Multinomial Model on Different Categories of Documents. International Journal of Computer Application, 2011.