30th April 2022

Analysis and implementation of Anti-spam techniques against email spamming

Mitra Abhi Sura

Project Abstract:

Nowadays Email communications is an essential service and has been greatly abused by spammers to spread malicious content, the most popular types of blatantly criminal spam are Nigerian letters and phishing that some users may fall prey to while causing others irritation and wasting their time. The distribution of spam has been improved in the past few years due to the improvement in internet technology. To counter the ever-growing issue of spam effective countermeasures need to be taken. Machine learning methods of recent are being used to successfully detect and filter spam emails.

In this project we will using a Content Based Filtering Technique this approach creates automatic filtering rules and classifies emails using machine learning approaches here we will be using the Naïve Bayesian classification which has been adopted to detect and control spam.

We were successful in our attempt at implementing and training a model which had a detection rate of 92.3%, a false positive rate of 0.7% and an overall accuracy of 95.7%.

Introduction:

Spam is usually said to be unsolicited usually commercial messages (such as emails, text messages, or Internet postings) sent to many recipients or posted in a large number of places. In simple words it can be termed as unwanted e-mail. Statistics show that nearly 85% of all emails are spam. Apart from wasting time, spam costs money to users with dial-up connections, wastes bandwidth, and may expose under-aged recipients to unsuitable content.

Anti-spam is software that aims to detect and block potentially dangerous email from user inboxes. It helps protect the confidentiality, integrity and availability of an individual user or an organization.

Anti-spam filters are one such way to implement anti-spamming they vary in functionality from Challenge/response Systems to content-based filters. The latter are generally more powerful, as spammers often spoof their IP’s and are coming up with various ways to circumvent pre-existing measures that have been in place for a long time. Content-based filters search for keyword patterns in the messages. These patterns need to be crafted by hand, and to achieve better results they need to be tuned to each user and to be constantly maintained which is a tedious tome consuming task, requiring expertise that an average user may not possess. Statistical, or Bayesian, filtering once set up requires no administrative maintenance. Instead, users mark messages as spam or non-spam and the filtering software learns from these judgements. Thus, it is matched to the end user's needs, and if users consistently mark/tag the emails, can respond quickly to changes in spam content.

Some Relevant Spam Statistics

In the following subsections, we will highlight some worldwide statistical observations. Besides, some country specific metrics will also be discussed.

- As of 2018 approximately 236 billion emails are exchanged daily
- 2018 saw an average of 14.5 billion spam emails daily
- Email spam costs businesses $20.5 billion every year.
- Scams and fraud comprise only 2.5% of all spam email however, phishing statistics indicate that identity theft makes up 73% of this figure.
- As many as 85% of all organizations have been targeted by phishing scams in 2021.
- Microsoft accounts are the most popular targets of phishing emails, accounting for 43% of all phishing attempts.
- United States has traditionally been the largest source of spam, however, in recent times it is not the case anymore. Though there were legislations such as CAN-SPAM (Controlling the Assault of Non-Solicited Pornography and Marketing Act) to protect the users, it did not have the expected deterrent effect on the spammers. USA houses world's top 70% spam gangs, responsible for coordinated worldwide spamming.
- As of April 2019, Brazil and Russia have conveniently overtaken USA and China (another substantial spam originating country), to produce approximately 16% and 14% of total volume of worldwide spam.

Motivation to explore the area

The field of anti-spamming is a fight between spammers trying to come up with new ways to circumvent current solutions and security specialists trying to thwart them at every step as the current anti-spam measures are mostly lagging behind the innovativeness of spammers.

Motivation behind this project was to analyze initiative is to address a gap that has risen over time in the field of spam email detection. We have discussed several existing approaches for anti-spamming techniques and highlighted the loopholes and the current situation. we have implemented a machine-learning based approach to anti-spamming as the current anti-spam measures have not been successful in their attempts to stop spam, we believe this justifies a shift towards machine learning based anti-spam approaches to be more widely in use as they can be tailored to specifically protect against spam emails and ha.

Significant use cases

- On the level of an Individual user:
    - Anti-spam software also helps prevent malicious emails that trick users to divulge personal and confidential information.
    - It is one of the best ways to avoid malware and viruses, as there's no opportunity to click on the harmful links included in some spam mail.
    - Anti-spam software "declutters" email inboxes, reducing emails that are distracting and time-wasting.

    - It blocks potential threats so that the end user doesn't have to deal with them and come to a decision.
    - it's essential for users who may not be very knowledgeable about IT or cybersecurity hazards

- Importance of Anti-spamming for a Business/Organization
    - Provides an additional layer of protection
        - Will help prevent different types of attacks and threats, such as spam, phishing, and malware, including ransomware, virus, and trojan.
    - Many businesses are subject to strict privacy and data storage regulations
        - Is true especially if they deal with sensitive or confidential information.
        - To continue operations, they may have to meet certain conditions including always using spam filtering to reduce the risk of data breaches.
    - Protect your business's reputation
    - A security breach can be extremely damaging for a business's reputation, especially if client data is compromised.
    - Anti-spam software helps ensure this doesn't occur, by increasing security and filtering out potentially harmful emails.

Background/related work:

Challenge/response systems

- Originally designed in 1997 by Stan Weatherby
- Called Email Verification
- A challenge–response system is a type of spam filter that automatically sends a reply with a challenge to the sender of an incoming e-mail.
    - In this reply, the purported sender is asked to perform some action to assure delivery of the original message, which would otherwise not be delivered.
    - The action to perform typically takes relatively little effort to do once, but great effort to perform in large numbers.
- Senders that have previously performed the challenging action, or who have previously been sent e-mail(s) to, would be automatically whitelisted.
- Various Challenges used in challenge response system
    - Simply sending an (unmodified) reply to the challenging message.
    - A challenge that includes a web URL
        - which can be loaded in an appropriate web browsing tool to respond to the challenge, so simply clicking on the link is sufficient to respond to the challenge.

- A challenge requiring reading natural language instructions on how to reply, with a special string or passcode in the reply.
- For example, converting a date string (such as 'Thu Jan 12 08:45:44 2012') into its corresponding timestamp (1326379544).
- Systems can attempt to produce challenges for which auto response is very difficult, or even an unsolved Artificial Intelligence problem.
- example is a "CAPTCHA" test in which the sender is required to view an image containing a word or phrase and respond with that word or phrase in text.

- Drawbacks
  - Many senders will find the additional burden annoying.
  - The requirements of the challenge may also be too confusing or simply go unread, causing messages to be delayed or go completely unconfirmed.
  - They don’t reject unsolicited bulk email at the SMTP layer.
  - Although C/R systems are simple to implement and don't require any software updates, they have numerous weaknesses when applied as anti-spam systems:
    - Misplaced Burden
      - Burden of anti-spam system placed on senders
      - One should not place the burden of an anti-spam system on the sender.
      - Any legitimate e-mail, where the C/R system mistakenly places the burden on an innocent sender to confirm incoming e-mail so that the recipient doesn't have to review any unconfirmed e-mail messages.
      - The recipient is receiving a benefit, which the sender is paying for
    - Automated Messages are hard to confirm
      - many automated systems don't offer enough information to discern what e-mail address the automated e-mail messages will come from.
    - System Stalemates
      - If the sender and recipient both use challenge/response systems, the two entities may never be able to communicate.
    - Confirmations Are Easily Defeated
    - Challenges Cause Abuse
      - A spammer could easily use the challenge/response system to bombard a victim with challenges by sending a large amount of e-mail messages to the C/R system that appear to be from the victim's e-mail address

        - Spam Is Still Accepted
            - Even though spam may not show up at the challenge/response system's inbox, the e-mail is accepted from the spammer for delivery.
            - E-mail from spammers should be rejected at the SMTP layer

Checksum based filtering

- Checksum-based filter exploits the fact that the emails are sent in bulk, that is that they will be identical with small variations.
- Checksum-based filters strip out everything that might vary between emails, reduce what remains to a checksum (checksum is calculated using a hash function), and look that checksum up in a database.
- such as the Distributed Checksum Clearinghouse which collects the checksums of emails that email recipients consider to be spam (some people have a button on their email client which they can click to nominate a message as being spam)
- If the checksum is in the database, the message is likely to be spam.
- **Advantages**
    - It lets ordinary users help identify spam, and not just administrators, thus vastly increasing the pool of spam fighters.
- **Drawbacks**
    - To avoid being detected in this way, spammers will sometimes insert unique invisible gibberish known as hashbusters into the middle of each of their emails.
    - This makes each message have a unique checksum.

DNS-based blacklists

- A Domain Name System blocklist is a service for operation of mail servers to perform a check via a Domain Name System (DNS) query
- whether the sending host's IP address is blacklisted for email spam.
- Most mail server software can be configured to check such lists, typically rejecting or flagging emails from such sites.
- Working
    - When a mail server receives a connection from a client, and wishes to check that client against a DNSBL it does the following:
        - Take the client's IP address and reverse the order of octets
        - Append the DNSBL's domain name

        - Look up this name in the DNS as a domain name
        - This will return either an address, indicating that the client is listed; or an "NXDOMAIN" ("No such domain") code, indicating that the client is not.

- Drawbacks
    - Legitimate emails blocked along with spam from shared mail servers
    - Lists of dynamic IP addresses
    - Lists that include "spam-support operations “
    - Some lists have unclear listing criteria and delisting may not happen automatically nor quickly.
    - Because lists have varying methods for adding IP addresses and/or URIs, it can be difficult for senders to configure their systems appropriately to avoid becoming listed on a DNSBL.

Statistical content filtering

- These systems primarily rely on the examination of the body or content of the email
- A common class of spam detectors for quite some time now has been the ‘Content based Filtering’ method and several of its variations.
- In these systems, a thorough analysis is done on the host message to find out patterns in message texts, these are then matched with predefined and confirmed spam patterns and a score is recorded.
- A decision of spam or ham is taken after comparing the cumulative score against a threshold value.
- One such method is Bayesian filtering is one of the most effective and intelligent solutions to filter spam emails nowadays. Bayesian filters must be “trained to work effectively”. After training, to classify an email message as spam, one can check if it exceeds a specific threshold
- **Advantages**:
    - Statistical, or Bayesian, filtering once set up requires no administrative maintenance
    - Instead, users mark messages as spam or non-spam and the filtering software learns from these judgements.
    - It is matched to the end user's needs, and if users consistently mark/tag the emails, can respond quickly to changes in spam content.
    - Can be trained on a per-user basis.
    - Has low rate of false positives (where legitimate email is incorrectly classified as spam).
    - The word probabilities are unique to each user

        - Therefore, it can evolve over time with corrective training whenever the filter incorrectly classifies an email.
        - As a result, Bayesian spam filtering accuracy after training is often superior to pre-defined rules.

- **Disadvantages**
    - May be susceptible to Bayesian poisoning.
        - A spammer practicing Bayesian poisoning will send out emails with large amounts of legitimate text
        - Spammer tactics include insertion of random innocuous words that are not normally associated with spam.
        - Thereby decreasing the email's spam score, making it more likely to slip past a Bayesian spam filter.
    - Technique used to try to defeat Bayesian spam filters is to replace text with pictures.
        - The whole text of the message, or some part of it, is replaced with a picture where the same text is "drawn".
        - The spam filter is usually unable to analyze this picture.

URL filtering

- Most spam/phishing messages contain an URL that they entice victims into clicking on. Thus, a popular technique since the early 2000s consists of extracting URLs from messages and looking them up in databases such as Spamhaus' Domain Block List (DBL), SURBL, and URIBL

Country-based filtering

- Some email servers expect to never communicate with particular countries from which they receive a great deal of spam. Therefore, they use country-based filtering – a technique that blocks email from certain countries. This technique is based on country of origin determined by the sender's IP address rather than any trait of the sender.

Honeypots

- Another approach is simply creating an imitation MTA that gives the appearance of being an open mail relay, or an imitation TCP/IP proxy server that gives the appearance of being an open proxy. Spammers who probe systems for open relays and proxies will find such a host and attempt to send mail through it, wasting their time and resources, and potentially, revealing information about themselves and the origin of the spam they are sending to the entity that operates the honeypot. Such a system may simply discard the spam attempts, submit them to **DNSBLs**, or store them for analysis by the entity operating the honeypot that may enable identification of the spammer for blocking.

Hybrid filtering

- **SpamAssassin,** Policyd-weight and others use some or all of the various tests for spam and assign a numerical score to each test. Each message is scanned for these patterns, and the applicable scores tallied up. If the total is above a fixed value, the message is rejected or flagged as spam. By ensuring that no single spam test by itself can flag a message as spam, the false positive rate can be greatly reduced.

Outbound spam protection

- Outbound spam protection involves scanning email traffic as it exits a network, identifying spam messages and then taking an action such as blocking the message or shutting off the source of the traffic. While the primary impact of spam is on spam recipients, sending networks also experience financial costs, such as wasted bandwidth, and the risk of having their IP addresses blocked by receiving networks.
  Outbound spam protection not only stops spam, but also lets system administrators track down spam sources on their network and remediate them – for example, clearing malware from machines which have become infected with a virus or are participating in a botnet.

After researching on the topic, we decided to implement Content filtering using a Naive Bayesian approach

Technical details of the project:

How the naive Bayes classifier is used to filter spam

- The Naïve Bayes theorem is used twice:
  - The First time is to compute the “Spamicity” of a word
  - The Second time to consider the “Spamicity” of several words and combine their “Spamicity” to determine a message's overall probability of being spam.
  - Sometimes Used to deal with rare word (but we will not be using this here)

Computing the probability that a message containing a given word is spam

- Let's suppose the suspected message contains the word "geass".
- the probability that a message containing a given word is spam is given by:

$$\Pr(S|W) = \frac{\Pr(W|S) \cdot \Pr(S)}{\Pr(W|S) \cdot \Pr(S) + \Pr(W|H) \cdot \Pr(H)}$$

- Where:
  - P(S|W) is the probability that a message is a spam, knowing that the word "geass" is in it
  - Pr(S) the overall probability that any given message is spam

- Pr(W|S) is the probability that the word "geass" appears in spam emails
- Pr(H) is the overall probability that any given message is not spam
- Pr(W|H) is the probability that the word "geass" appears in ham emails.

- Statistics show that the current probability of any message being spam is 80%, at the very least

$$\Pr(S) = 0.8; \Pr(H) = 0.2$$

- Most Bayesian spam detection software assumes that there is no a priori
  - That is reason for any incoming message to be spam rather than ham, and considers both cases to have equal probabilities of 50%

$$\Pr(S) = 0.5; \Pr(H) = 0.5$$

  - Filters using the above Hypothesis are known as “not biased”

- This Simplifies the previous formula to:

$$\Pr(S|W) = \frac{\Pr(W|S)}{\Pr(W|S) + \Pr(W|H)}$$

- Functionally equivalent to asking, "what percentage of occurrences of the word "geass" appear in spam emails
- This quantity is called "spamicity" of the word "geass"
- The number Pr(W|S) used in this formula is approximated to the frequency of emails containing "geass" in the emails identified as spam during the learning phase.
- Similarly, Pr(W|H) is approximated to the frequency of emails containing "geass" in the emails identified as ham during the learning phase.
- For these approximations to make sense, the set of emails needs to be big and representative enough.
- It is also advisable that the data set of emails conforms to the 50% hypothesis about repartition between spam and ham.
  - This means that the number of spam and ham emails in the data set need to be equal in order to line up with the “nonbiased” hypothesis we have taken.
  - Our training set consists of a total of 702 mails
    - With 260 mails divided equally between spam and ham

Combining individual probabilities

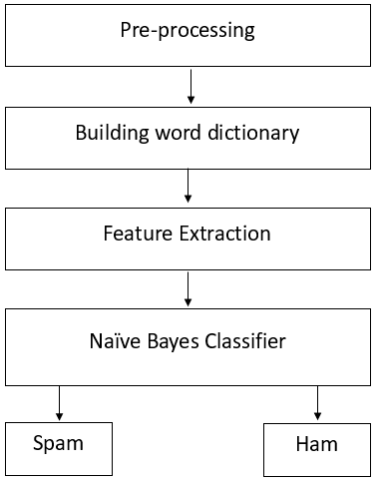
- Determining whether a message is spam or ham based only on the presence of the word "geass" is inherently prone to error.
- Thus, we consider several words and combine their spamicity to determine a message's overall probability of being spam.

$$p = \frac{p_1 p_2 \cdots p_N}{p_1 p_2 \cdots p_N + (1 - p_1)(1 - p_2) \cdots (1 - p_N)}$$

- Where:
    - **p** is the probability that the suspect email is spam
    - **p1** is the probability p(W1|S) that the first word (for example "geass") appears, given that the email is spam
    - **p2** is the probability p(W2|S) that the second word (for example “gate") appears, given that the email is spam
    - This occurs for n such words.
- Spam filtering software based on this formula is sometimes referred to as a naive Bayes classifier, as "naive" refers to the strong independence assumptions between the features

Email Spam Filtering Technique Implementation

- Pre-processing
- Building word dictionary
- Feature Extraction
- Training and testing Classifier



Pre-processing Stage

- Pre-processing of the Ling-spam corpus dataset:
- Each file has a subject and a body we extracted the body from the txt file and performed the following on it:
    - Lemmatization
        - The process of grouping together the different inflected forms of a word so they can be analyzed as a single item
        - Each word is run through the code which returns then stores the word in its common base form
    - Removal of stop words non-words and special characters
        - A set called “stop_words” which contained English stop words was used.
        - The contents of the original file where then erased and then words which were not present in “stop_words” and non-words where not rewritten into the txt file after lemmatization

Building word dictionary

- A word dictionary basically contains all the words used in the data set provided to it and contains the frequency of each word across the entire data set.
- We have chosen to exclude all words with length less than 3 and have taken 3000 of the most common words to make the word dictionary.
- Sample Output with 30 of the most common words from our word dictionary after performing word dictionary building on 702 preprocessed emails

```
[('order', 1495), ('address', 1336), ('report', 1290), ('mail', 1133), ('language', 1107), ('send', 1080), ('email', 1066), ('p
rogram', 1041), ('list', 969), ('one', 925), ('name', 890), ('receive', 826), ('free', 801), ('work', 801), ('money', 797), ('i
nformation', 684), ('business', 669), ('please', 657), ('university', 601), ('day', 577), ('follow', 545), ('internet', 533),
('call', 492), ('check', 486), ('http', 479), ('linguistic', 460), ('include', 452), ('com', 450), ('number', 446), ('need', 42
8)]
```

Feature Extraction

- Feature extraction refers to the process of transforming raw data into numerical features that can be processed while preserving the information in the original data set. It yields better results than applying machine learning directly to the raw data.
- Feature Extraction aims to reduce the number of features in a dataset by creating new features from the existing ones (and then discarding the original features).
- These new reduced set of features should then be able to summarize most of the information contained in the original set of features. In this way, a summarized version of the original features can be created from a combination of the original set.
- Once the word dictionary is complete, we can extract an n-dimensional word count vector (the specific feature extracted in this project) for each email in the training set. The frequency of n words in the training file is represented by each word count vector.
- Most of the values tend to be zero in the frequency list.
- **EXAMPLE: -**
    - Suppose we have x words in our dictionary. Each word count vector contains the frequency of x dictionary words in the training file.
- A matrix is created in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that text sample.
- Let us consider a few sample texts from a document
- document = [“One Student helps Two Students”, “Two Students help Four Students”, “Each Student helps many other Students at College.”]

	at	each	four	student	students	college	help	helps	many	one	Other	two
Document 1	0	0	0	1	1	0	0	1	0	0	0	1
Document 2	0	0	1	0	2	0	1	0	0	0	0	1
Document 3	1	1	0	1	1	1	0	1	1	0	1	0

- All of the data above would be put in the form of a nested array with each separate array within the larger array representing the features of a particular document.
- The words are not kept as strings when the feature extraction procedure is completed. Rather, they are assigned a certain index value. The number of occurrences of the jth word in the dictionary in the ith file will be the value at index 'ij.' In this scenario, index 0 would be assigned to 'at,' index 1 to 'each,' index 2 to 'four,' and so on.

Training and testing Classifier

- The training and testing of the classifier was done on 260 emails out of which 130 are ham and 130 were spam emails.
- The classifier Identified 1 HAM email as SPAM and 10 SPAM emails as HAM the rest of the identification was correct.

Experimental Results:

```
In the HAM mails our model detected

Ham detected: 129
Spam detected: 1

In the Spam mails our model detected

Ham detected: 10
Spam detected: 120
```

Observations:

- **True Positives (TP):** The number of spam emails classified as spam.
    - **TP** = 120
- **True Negatives (TN):** The number of ham emails classified as ham.
    - **TN** = 129
- **False Positives (FP):** The number of ham falsely classified as spam.
    - **FP** = 1
- **False Negatives (FN):** The number of spam falsely classified as ham.
    - **FN =** 10
- **Detection Rate (DR):** TP / (TP + FN).
    - **DR** = 0.923
- **False positive Rate (FPR):** FP/ (TN + FP).
    - **FPR** = 0.007
- **Overall Accuracy (OA):** (TP + TN) / (TP + FP + FN +TN).

- **OA**= 0.957

FINAL CODE

```
import os.path
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from collections import Counter
import numpy as np
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import confusion_matrix

def prepro(train_dir):
    files = [os.path.join(train_dir,f) for f in os.listdir(train_dir)]
    lemm = WordNetLemmatizer()
    stop_words = set(stopwords.words('english'))
    for text in files:
        file = open(text,'r+')
        line = file.read()
        words = line.split()
        new = ""
        file.truncate()
        file.close()
        f_new = open(text,'w')
        for r in words:
            if r.isalpha() == True:
                if not r in stop_words:
                    r = lemm.lemmatize(r)
                    f_new.write(" "+r)
        f_new.close()


def make_Dictionary(train_dir):
    emails = [os.path.join(train_dir,f) for f in os.listdir(train_dir)]
    all_words = []
    for mail in emails:
        file = open(mail,'r+')
        line = file.read()
        words = line.split()
        all_words += words
        file.close()
    word_dict = Counter(all_words)
    for item in list(word_dict):
            if len(item) < 3:
                del word_dict[item]
    return word_dict.most_common(3000)
```

```
def extract_features(mail_dir):
   emails = [os.path.join(mail_dir,f) for f in os.listdir(mail_dir)]
   features_matrix = np.zeros((len(emails),3000))
   docID = 0;
   for mail in emails:
      file = open(mail,'r+')
      line = file.read()
      words = line.split()
      file.close()
      for word in words:
         wordID = 0
         for i,d in enumerate(dictionary):
            if d[0] == word:
               wordID = i
               features_matrix[docID,wordID] = words.count(word)
      docID = docID + 1
   return features_matrix

train_dir = 'C:\\Users\\Aravo\\OneDrive\\Desktop\\FIS projecct\\ling-spam\\train-mails'
test_dir = 'C:\\Users\\Aravo\\OneDrive\\Desktop\\FIS projecct\\ling-spam\\test-mails'
dictionary = make_Dictionary(train_dir)
train_labels = np.zeros(702)
train_labels[351:701] = 1
train_matrix = extract_features(train_dir)
model = MultinomialNB()
model.fit(train_matrix,train_labels)
test_matrix = extract_features(test_dir)
test_labels = np.zeros(260)
test_labels[130:260] = 1
result = model.predict(test_matrix)
c1 = confusion_matrix(test_labels,result)
print("In the HAM mails our model detected\n")
print("Ham detected:",c1[0][0])
print("Spam detected:",c1[0][1])
print("\nIn the Spam mails our model detected\n")
print("Ham detected:",c1[1][0])
print("Spam detected:",c1[1][1])
```

Summary:

In this project we attempted to analyze various anti-spamming techniques, after the analysis and weighing the pros and cons of the many methods available, we observed that a spammer has been able to come up with many ways or ideas to thwart a traditional spam filter software. We decided upon performing anti-spamming using statistical content filtering which implements the Naïve Bayes theorem. The major reason for implementing this theorem is if users consistently mark/tag emails as spam the system can quickly respond to changes in spam content making it highly flexible and able to adapt and identify new spam messages. Over a period of time, it evolves itself based on users input. This leads Bayesian spam filtering accuracy after training to be often superior to anti-spam measures implemented using pre-defined rules.

For lemmatization and modelling our classifier we used an open-source Python library called Scikit-learn which has powerful tools for data analysis and data mining. The data set's features were extracted and passed through the function available for our modelling. After successfully training our model we achieved a detection rate of 92.3%, a false positive rate of 0.7% and an overall accuracy of 95.7%.

Conclusion:

We have explored various problems associated with spam and different methods and techniques in the issue. We have concluded that Traditional spam filter software's are unable to cope with vast volumes of spam that slip past anti-spam defenses. Machine learning approaches have provided researchers with better ways to combat spam. Since email now can be classified with less human intervention thus making the control easier and more accurate.

There is no single measure that can claim to perform spam filtering perfectly that is with 0% false positive and 0% false negative.

If we had the opportunity to do the project differently, we would have included different algorithms and not just the Naïve bayes and perform a comparison between them. We would have started working towards integrating our filter in an actual email inbox to gauge its efficiency and effectiveness. If we get the opportunity to do so, we would like to implement a hybrid anti-spam measure that takes more than one of the approaches to perform spam filtering.

References:

dataprot.net

Proceedings of the workshop on Machine Learning in the New Information Age, G. Potamias, V. Moustakis and M. van Someren (eds.), 11th European Conference on Machine Learning, Barcelona, Spain, pp. 9-17, 2000.

ASIF KARIM , SAMI AZAM , BHARANIDHARAN SHANMUGAM , KRISHNAN KANNOORPATTI , AND MAMOUN ALAZAB A *Comprehensive Survey for Intelligent Spam Email Detection*

Ling Spam dataset taken from https://github.com/yolanda93/spam_filter the ling-spam folder

https://www.templetons.com/brad/spam/challengeresponse.html

https://www.greenviewdata.com/wiki/Challenge_Response_Systems

https://www.dnsbl.info/

https://datatracker.ietf.org/doc/html/rfc6471

A Bayesian Approach to Filtering Junk E-Mail Mehran - Sahami Susan Dumaisy David Heckermany Eric Horvitzy

https://www.process.com/products/pmas/whitepapers/intro_bayesian_filtering.html

https://scikit-learn.org/stable/