Optimizing Spam Filtering With Machine Learning

Abstract

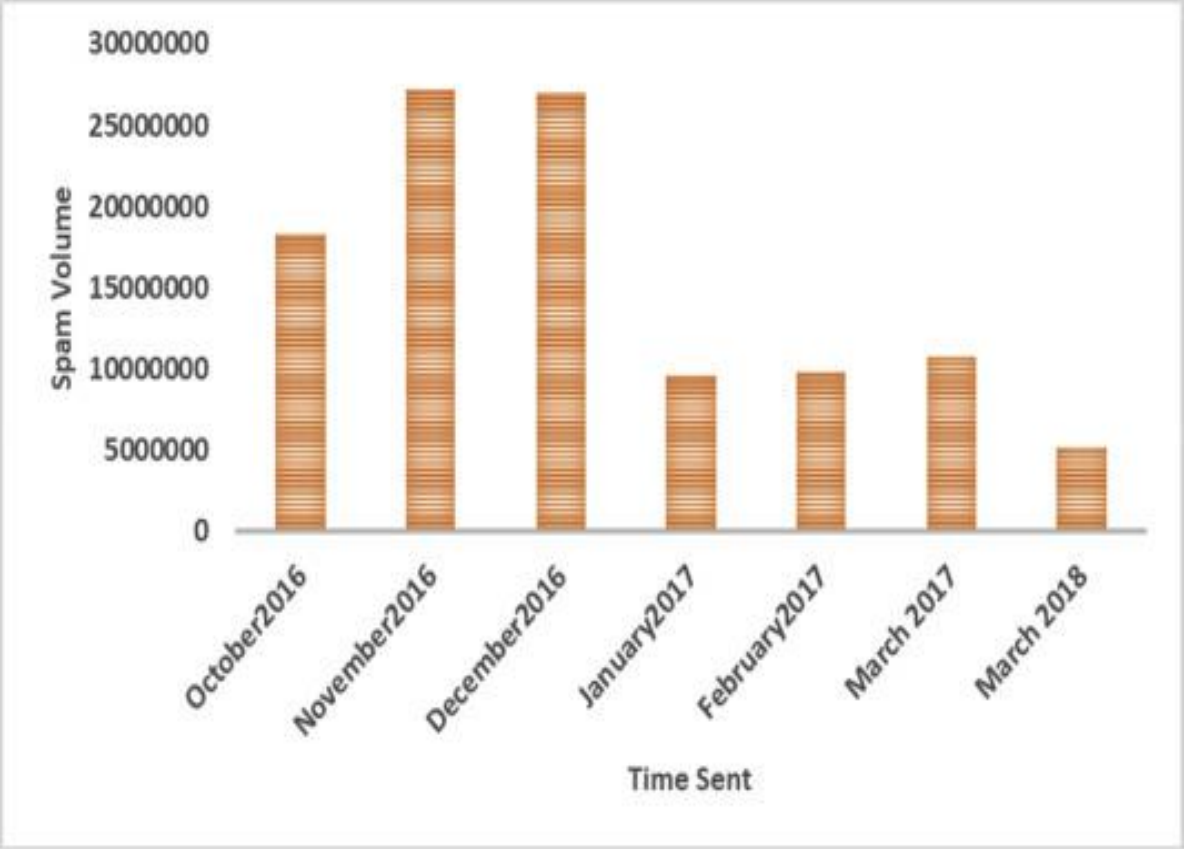
The upsurge in the volume of unwanted emails called spam has created an intense need for the development of more dependable and robust antispam filters. Machine learning methods of recent are being used to successfully detect and filter spam emails. We present a systematic review of some of the popular machine learning based email spam filtering approaches. Our review covers survey of the important concepts, attempts, efficiency, and the research trend in spam filtering. The preliminary discussion in the study background examines the applications of machine learning techniques to the email spam filtering process of the leading internet service providers (ISPs) like Gmail, Yahoo and Outlook emails spam filters. Discussion on general email spam filtering process, and the various efforts by different researchers in combating spam through the use machine learning techniques was done. Our review compares the strengths and drawbacks of existing machine learning approaches and the open research problems in spam filtering. We recommended deep leaning and deep adversarial learning as the future techniques that can effectively handle the menace of spam emails.

Introduction

In recent times, unwanted commercial bulk emails called spam has become a huge problem on the internet. The person sending the spam messages is referred to as the spammer. Such a person gathers email addresses from different websites, chatrooms, and viruses . Spam prevents the user from making full and good use of time, storage capacity and network bandwidth. The huge volume of spam mails flowing through the computer networks have destructive effects on the memory space of email servers, communication bandwidth, CPU power and user time . The menace of spam email is on the increase on yearly basis and is responsible for over 77% of the whole global email traffic . Users who receive spam emails that they did not request find it very irritating. It is also resulted to untold financial loss to many users who have fallen victim of internet scams and other fraudulent practices of spammers who send emails pretending to be from reputable companies with the intention to persuade individuals to disclose sensitive personal information like passwords, Bank Verification Number (BVN) and credit card numbers.

According to report from Kaspersky lab, in 2015, the volume of spam emails being sent reduced to a 12-year low. Spam email volume fell below 50% for the first time since 2003. In June 2015, the volume of spam emails went down to 49.7% and in

July 2015 the figures was further reduced to 46.4% according to anti-virus software developer Symantec. This decline was attributed to reduction in the number of major botnets responsible for sending spam emails in billions. Malicious spam email volume was reported to be constant in 2015. The figure of spam mails detected by Kaspersky Lab in 2015 was between 3 million and 6 million. Conversely, as the year was about to end, spam email volume escalated. Further report from Kaspersky Lab indicated that spam email messages having pernicious attachments such as malware, ransomware, malicious macros, and JavaScript started to increase in December 2015. That drift was sustained in 2016 and by March of that year spam email volume had quadrupled with respect to that witnessed in 2015. In March 2016, the volume of spam emails discovered by Kaspersky Lab is 22,890,956. By that time the volume of spam emails had skyrocketed to an average of 56.92% for the first quarter of 2016. Latest statistics shows that spam messages accounted for 56.87% of e-mail traffic worldwide and the most familiar types of spam emails were healthcare and dating spam. Spam results into unproductive use of resources on Simple Mail Transfer Protocol (SMTP) servers since they have to process a substantial volume of unsolicited emails. The volume of spam emails containing malware and other malicious codes between the fourth quarter of 2016 and first quarter of 2018 is depicted in the figure below.



To effectively handle the threat posed by email spams, leading email providers such as Gmail, Yahoo mail and Outlook have employed the combination of different machine learning (ML) techniques such as Neural Networks in its spam filters. These ML techniques have the capacity to learn and identify spam mails and phishing messages by analyzing loads of such messages throughout a vast collection of computers. Since machine learning have the capacity to adapt to varying conditions, Gmail and Yahoo mail spam filters do more than just checking junk emails using pre-existing rules. They generate new rules themselves based on what they have learnt as they continue in their spam filtering operation. The machine

learning model used by Google have now advanced to the point that it can detect and filter out spam and phishing emails with about 99.9 percent accuracy. The implication of this is that one out of a thousand messages succeed in evading their email spam filter. Statistics from Google revealed that between 50-70 percent of emails that Gmail receives are unsolicited mail. Google's detection models have also incorporated tools called Google Safe Browsing for identifying websites that have malicious URLs. The phishing-detection performance of Google have been enhanced by introduction of a system that delay the delivery of some Gmail messages for a while to carry out additional comprehensive scrutiny of the phishing messages since they are easier to detect when they are analyzed collectively. The purpose of delaying the delivery of some of these suspicious emails is to conduct a deeper examination while more messages arrives in due course of time and the algorithms are updated in real time. Only about 0.05 percent of emails are affected by this deliberate delay.

Though there are several email spam filtering methods in existence, the state-of-the-art approaches are discussed in this paper. We explained below the different categories of spam filtering techniques that have been widely applied to overcome the problem of email spam.

- •

Content Based Filtering Technique: Content based filtering is usually used to create automatic filtering rules and to classify emails using machine learning approaches, such as Naïve Bayesian classification, Support Vector Machine, K Nearest Neighbor, Neural Networks. This method normally analyses words, the occurrence, and distributions of words and phrases in the content of emails and used then use generated rules to filter the incoming email spams [28].

- •

Case Base Spam Filtering Method: Case base or sample base filtering is one of the popular spam filtering methods. Firstly, all emails both non-spam and spam emails are extracted from each user's email using collection model. Subsequently, pre-processing steps are carried out to transform the email using client interface, feature extraction, and selection, grouping of email data, and evaluating the process. The data is then classified into two vector sets. Lastly, the machine learning algorithm is used to train datasets and test them to decide whether the incoming mails are spam or non-spam [28].

- •

Heuristic or Rule Based Spam Filtering Technique: This approach uses already created rules or heuristics to assess a huge number of patterns which are usually regular expressions against a chosen message. Several similar patterns increase the score of a message. In contrast, it deducts from the score if any of the patterns did not correspond. Any message's score that surpasses a specific threshold is

filtered as spam; else it is counted as valid. While some ranking rules do not change over time, other rules require constant updating to be able to cope effectively with the menace of spammers who continuously introduce new spam messages that can easily escape without been noticed from email filters [28]. A good example of a rule based spam filter is Spam Assassin [35].

- •

  Previous Likeness Based Spam Filtering Technique: This approach uses memory-based, or instance-based, machine learning methods to classify incoming emails based to their resemblance to stored examples (e.g. training emails). The attributes of the email are used to create a multi-dimensional space vector, which is used to plot new instances as points. The new instances are afterward allocated to the most popular class of its K-closest training instances [33]. This approach uses the k-nearest neighbor (kNN) for filtering spam emails.

- •

  Adaptive Spam Filtering Technique: The method detects and filters spam by grouping them into different classes. It divides an email corpus into various groups, each group has an emblematic text. A comparison is made between each incoming email and each group, and a percentage of similarity is produced to decide the probable group the email belongs to [137].

  Many researchers and academicians have proposed different email spam classification techniques which have been successfully used to classify data into groups. These methods include probabilistic, decision tree, artificial immune

system [4], support vector machine (SVM) [5], artificial neural networks (ANN) [6], and case-based technique [7]. It have been shown in literature that it is possible to use these classification methods for spam mail filtering by using content-based filtering technique that will identify certain features (normally keywords frequently utilised in spam emails). The rate at which these features appear in emails ascertain the probabilities for each characteristic in the email, after which it is measured against the threshold value. Email messages that exceed the threshold value are classified as spam [8]. ANN is a non-linear model that seeks to imitate the functions of biological neural networks. It is made up of simple processing components named neurons and carries out its computational operations by processing information [9,10]. Several research work have employed neural network to classify unwanted emails as spam by applying content-based filtering. These techniques decide the properties by either computing the rate of occurrence of keywords or patterns in the email messages. Literatures show that Neural Network algorithms that are utilised in email filtering attain moderate classification performance. Some of the most popular spam email classification algorithms are Multilayer Perceptron Neural Networks (MLPNNs) and Radial Base Function Neural Networks (RBFNN). Researchers used MLPNN as a classifier for spam filtering but not many of them used RBFNN for classification.

Support Vector Machines (SVM) has proved over the years to be one of the most powerful and efficient state-of-the-art classification techniques for solving the email spam problem [78]. They are supervised learning models that analyze data and identify patterns used for categorisation and exploring the relationship between variables of interest. SVM algorithms

are very potent for the identification of patterns and classifying them into a specific class or group. They can be easily trained and according to some researchers, they outperform many of the popular email spam classification methods [130,131]. This is because during training, SVM use data from email corpus. However, for high dimension data, the strength and efficacy of SVM diminish over time due to computational complexities of the processed data [132,133]. According to [134], SVM is a good classifier due to its sparse data format and satisfactory recall and precision value. SVM has high classification accuracy. Moreover, SVM is considered a notable example of "kernel methods", which is one of the central areas of machine learning. Decision tree is another machine learning algorithm that has been successfully applied to email spam filtering. Decision trees (DT) need comparatively minute effort from users during training of datasets. DT completely perform variable analysis or feature selection of the email corpus data training. The performance of a tree does not depend on the relationships among parameters. A great benefit of decision tree is its capacity to assign unambiguous values to problems, decisions, and results of every decision [135]. This decreases vagueness in decision-making. Another huge advantage of the decision tree compared to other machine learning techniques is the fact that it makes open all the likely options and follows each option to its end in one view, giving room for straightforward evaluation

among the different nodes of the tree. Despite the numerous advantages of Decision tree, it still has some drawbacks which are: unless there is appropriate pruning, controlling tree growth can be very difficult. Decision trees are a nonparametric machine learning algorithm that is incredibly adaptable and vulnerable to overfitting of training data [135]. This makes them to some extent poor classifiers and limit their classification accuracy. The different types of Decision trees that have been applied to email spam filtering are NBTree Classifier [80], C4.5/J48 Decision Tree Algorithm [81] and Logistic Model Tree Induction (LMT) [80]. Naïve Bayes is another wonderful machine learning algorithm that has been applied in email spam filtering. A Naive Bayes (NB) classifier simply apply Bayes' theorem on the context classification of each email, with a strong assumption that the words included in the email are independent of each other [38]. NB is desirable for email spam filtering because of its simplicity, ease of implementation and quick convergence compared to conditional models such as logistic regression [136]. It needs fewer training data. It is very scalable. No bottleneck is created by increase in the number of predictors and discrete unit of information [136]. NB can be used to solve both classification problems involving two or more classes. It can be used to make forecasting that is subject to or involving probability variation. They can

effectively manage continuous and discrete data. NB algorithms are not susceptible to irrelevant features. Naıve Bayes algorithm is predominantly famous in business-related and open-source spam filters [51]. This is because apart from the advantages listed above, NB needs little training time or speedy assessment to detect and filter email spam. NB filters need training that can be offered by the earlier set of non-spam and spam messages [136]. It keeps the record of the changes that take place in each word that occurs in legitimate, illegitimate messages, and in both. NB can be applied to spam messages in diverse datasets having different features and attribute [136]. Stochastic optimization techniques such as evolutionary algorithms (EAs) have also been applied to spam filtering. This is because they do not have any sophisticated mathematical computation. Also, they can handle the solutions generated, they seek to recognise individuals that have the optimal solutions for the problem [11]. Several earlier works exist that integrated Genetic Algorithms with Neural Networks [12] to enhance the performance of neural network algorithms. A related approach of evolutionary computation methods such as Genetic Algorithms (GAs) is Particle Swarm Optimization (PSO), which is a technique that can be used for optimizing many continuous nonlinear functions and classification techniques. PSO is inspired by the social behaviour of animals such as flocks of bird and shoal of fishes. It has been applied in

many areas of human endeavour such as neural network, swarm robotics, telecommunications, signal processing, data mining, and several other applications [129]. PSO algorithm operates on a population (swarm) of particles, with the characteristic of no crossover and mutation calculation as found in genetic algorithm. Every particle have a position and velocity. Each of the particle is a potential solution in the swarm. This makes it easy to implement [13]. What appears to be the most efficient spam filtering approach now is the automatic email filtering which have successfully been used for frustrating the malicious intentions of spammers. Some years back, the largest part of the spam email can be efficiently addressed by stopping emails originating from specified addresses or remove messages with specific subject lines. More deceitful and sophisticated techniques such as utilising arbitrary sender addresses and/or inserting haphazard characters to the beginning or the end of the message subject line are now been used by spammers to surmount the hurdle posed by the filtering methods [9]. Owing to the fact that a good number of real-world filters make use of the amalgamation of ML and application-specific knowledge in the form of hand-coded rules, comprehending the revolutionising attributes of spam is also germane, and many studies have been done on this subject [14,15]. However, in spite of the increasing research efforts on spam filtering, the growth of spam emails is still on

alarming rate. This is evident with spammers devising more sophisticated methods for dodging detection, a very good example are emails with stego images (i.e. images with information hidden inside).

The two common approaches used for filtering spam mails are knowledge engineering and machine learning. Emails are classified as either spam or ham using a set of rules in knowledge engineering. The person using the filter, or the software company that stipulates a specific rule-based spam-filtering tool must create a set of rules. Using this method does not guarantee efficient result since there is need to continually update the rules. This can lead to time wastage and it is not suitable especially for naive users. Machine learning approach have proved to be more efficient than knowledge engineering approach. No rule is required to be specified, rather a set of training samples which are pre-classified email messages are provided. A particular machine learning algorithm is then used to learn the classification rules from these email messages [16]. Several studies have been carried out on machine learning techniques and many of these algorithms are being applied in the field of email spam filtering. Examples of such algorithms include Deep Learning, Naïve Bayes, Support Vector Machines, Neural Networks, K-Nearest Neighbour, Rough sets, and Random Forests. The contributions of this work are given as follows:

- a.

  We did a comprehensive evolutionary survey of the most important features of email spam, the evolution and developments. Through this, we highlighted some interesting research gaps and research directions.

- b.

  We discussed the architectures of spam filters and the application of ML techniques to spam filtering process of Gmail, Yahoo mail and Outlook mail. The different components of the email spam filter were vividly discussed.

- c.

  We presented an elaborate study of several techniques applied to email spam filtering and presented a phenomenal review of literatures on spam email filtering over the period (2004–2018).

- d.

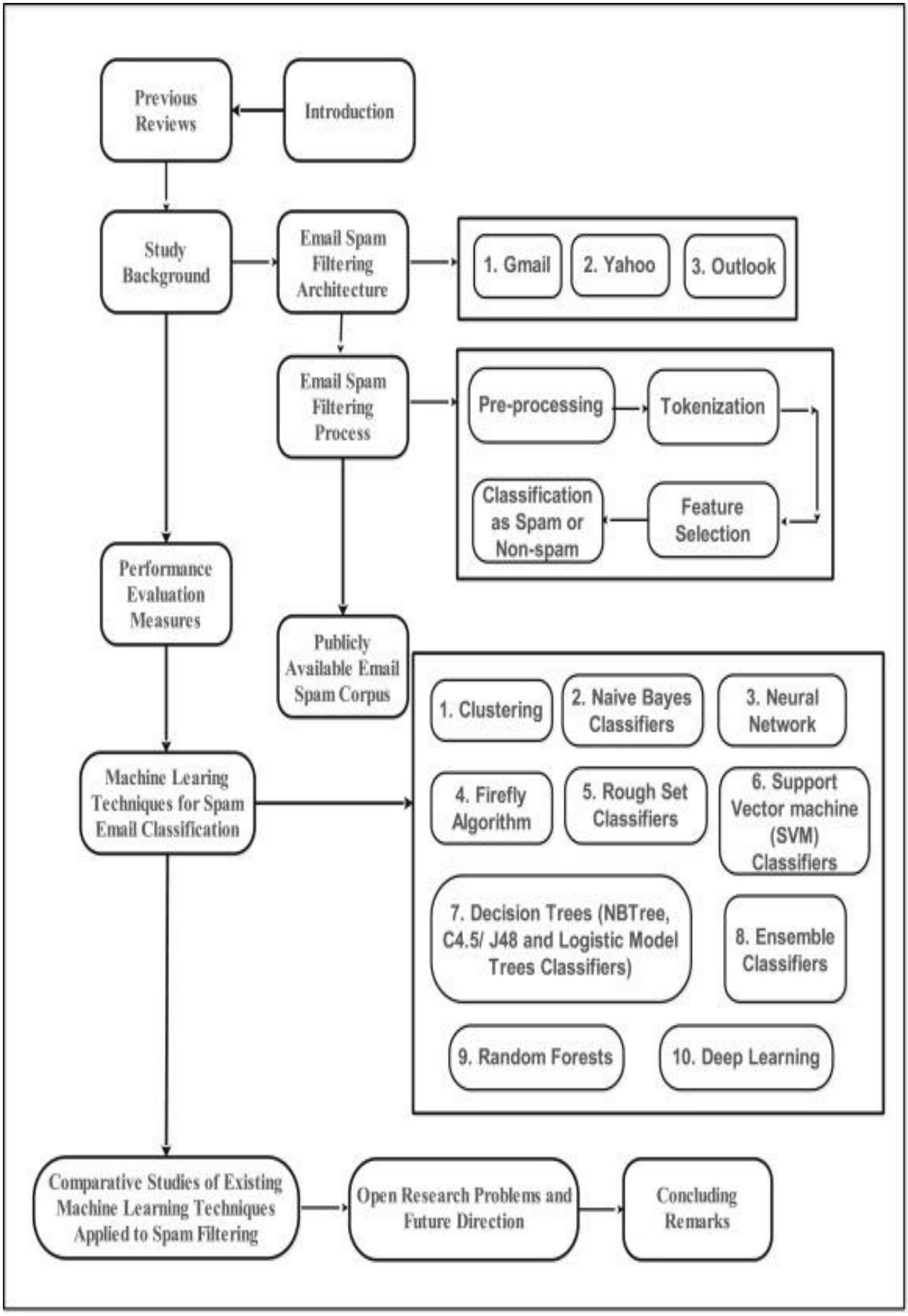
  We exposed researchers to some powerful machine learning algorithms that are not yet explored in spam filtering.

- e.

  We stated in clear terms our findings on some open research problems in relation to spam filtering and recommended proactive steps for the development of machine learning techniques to curb future evolving of new variants of spam that might find it easy to evade filters.

The rest of this paper is organized as follows: Section 2 gives a succinct account of previous reviews, Section 3 is the background discussion, Section 4 describes the performance measures for evaluating the effectiveness of spam filters, Section 5 explains the machine learning algorithms that have found application in spam filtering, Section 6 is the comparative studies of existing machine learning techniques used in spam filtering, Section 7 unveils open research problems in machine learning for spam filtering and future direction before concluding in Section 8.

To increase the readability of the manuscript and also enhance the understanding of the readers, the structure of this paper is depicted in figure below:


Previous Reviews
Introduction
Study Background
Email Spam Filtering Architecture
1. Gmail
2. Yahoo
3. Outlook
Email Spam Filtering Process
Pre-processing
Tokenization
Feature Selection
Classification as Spam or Non-spam
Performance Evaluation Measures
Publicly Available Email Spam Corpus
1. Clustering
2. Naive Bayes Classifiers
3. Neural Network
Machine Learing Techniques for Spam Email Classification
4. Firefly Algorithm
5. Rough Set Classifiers
6. Support Vector machine (SVM) Classifiers
7. Decision Trees (NBTree, C4.5/ J48 and Logistic Model Trees Classifiers)
8. Ensemble Classifiers
9. Random Forests
10. Deep Learning
Comparative Studies of Existing Machine Learning Techniques Applied to Spam Filtering
Open Research Problems and Future Direction
Concluding Remarks

Related work

There is a rapid increase in the interest being shown by the global research community on email spam filtering. In this section, we present similar reviews that have been presented in the literature in this domain. This method is followed so as to articulate the issues that are yet to be addressed and to highlight the differences with our current review. Lug presented a brief survey to explore the gaps in whether information filtering and information retrieval technology can be applied to postulate Email spam detection in a logical, theoretically grounded manner, in order to facilitate the introduction of spam filtering technique that could be operational in an efficient way. However, the survey did not present the details of the Machine learning algorithms, the simulation tools, the publically available datasets and the architecture of the email spam environment. It also fails short of presenting the parameters used by previous researches in evaluating other proposed techniques. Wang reviewed the different techniques used to filter out unsolicited spam emails. The paper also to categorized email spams into different hierarchical folders, and automatically regulate the tasks needed to response to an email message. However, some of the limitations of the review article are that; machine learning techniques, email spam architecture, comparative analysis of

previous algorithms and the simulation environment were all not covered.

The paper titled “Spam filtering and email-mediated applications” chronicles the details of email spam filtering system. It then presented a framework for a new technique for linking multiple filters with an innovative filtering model using ensemble learning algorithm. The article also explained the notion of operable email (OE) in an email-mediated application. Furthermore, a demonstration was made of OE in executing an email assistant and other intelligent applications on the world social email network . However, the survey paper did not cover recent articles as it was published more than a decade ago. Cormack reviewed previously proposed spam filtering algorithms up to 2008 with specific emphasis on efficiency of the proposed systems. The main focus of the review is to explore the relationships between email spam filtering with other spam filtering systems in communication and storage media. The paper also scrutinized the characterization of email spams, including the user's information requirements and the function of the spam sieve as a constituent of a huge and complex information system. However, certain important components of spam filters were not considered in the survey. These includes; the architecture of the system, the simulation environment and the comparative analysis of the performance of the reviewed filters.

Sanz, Hidalgo, and Pérez detailed the research issues related to email spams, in what way it affects users, and by what means users and providers can reduce it effects. The paper also enumerates the legal, economic, and technical measures used to mediate the email spams. They pointed out that based on technical measures, content analysis filters have been extensively used and proved to have reasonable percentage of accuracy and precision as a result, the review focused more on them, detailing how they work. The research work explained the organization and the procedure of many machine learning approaches utilized for the purpose of filtering email spams. However, the review did not cover recent research articles in this area as it was published in 2008 and comparative analysis of the different content filters was also missing. A brief study on E-mail image spam filtering methods was presented by . The study concentrated on email antispam filtering approaches used to transfer from text-based techniques to image-based methods. Spam and the spam filters premeditated to reducing it have spawned an upsurge in creativeness and inventions. However, the study did not cover machine learning techniques, simulation tools, dataset corpus and the architecture of email spam filtering techniques.

Bhowmick and Hazarika presented a broad review of some of the popular content-based e-mail spam filtering methods. The paper focused mostly on machine learning algorithms for spam

filtering. They surveyed the important concepts, efforts, effectiveness, and the trend in spam filtering. They discussed the fundamentals of e-mail spam filtering, the changing nature of spam, the tricks of spammers to evade spam filters of e-mail service providers (ESPs), and also examined the popular machine learning techniques used in combating the menace of spam. Laorden *et al.* presented a detailed revision of the usefulness of anomaly discovery used for Email spam filtering that decreases the requirement of classifying email spam messages and only works with the representation of single class of emails. The review contains a demonstration of the first anomaly based spam sieving method, an improvement of the method, which used a data minimization technique to the characterized dataset corpus to decrease processing phase while retaining recognition rates and an investigation of the appropriateness of selecting non-spam emails or spam as a demonstration of normality.

This current review differed from the previous reviews presented in the preceding paragraph by focusing more on revisiting machine learning techniques used for email spam filtering. The review intends to cover the architecture of the email spam filtering systems, parameters used for comparative analysis, simulation tools and the dataset corpus.

3.Background

Here we discussed the architecture of email server and the stages in processing email. We explained the different stages involved in pre-processing and feature selection.

3.1. Email spam filtering architecture

Spam filtering is aimed at reducing to the barest minimum the volume of unsolicited emails. Email filtering is the processing of emails to rearrange it in accordance to some definite standards. Mail filters are generally used to manage incoming mails, filter spam emails, detect and eliminate mails that contain any malicious codes such as virus, trojan or malware. The workings of email is influence by some basic protocols which include the SMTP. Some of the widely used Mail User Agents (MUAs) are Mutt, Elm, Eudora, Microsoft Outlook, Pine, Mozilla Thunderbird, IBM notes, Kmail, and Balsa. They are email clients that assists the user to read and compose emails. Spam filters can be deployed at strategic places in both clients and servers.

Spam filters are deployed by many Internet Service Providers (ISPs) at every layer of the network, in front of email server or at mail relay where there is the presence of firewall [25]. The firewall is a network security system that monitors and manages the incoming and outgoing network traffic based on predetermined security rules. The email server serves as an incorporated anti-spam and anti-virus solution

providing a comprehensive safety measure for email at the network perimeter [26]. Filters can be implemented in clients, where they can be mounted as add-ons in computers to serve as intermediary between some endpoint devices [27]. Filters block unsolicited or suspicious emails that are a threat to the security of network from getting to the computer system. Also, at the email level, the user can have a customized spam filter that will block spam emails in accordance with some set conditions [28].

3.1.1. How Gmail, Yahoo and Outlook emails spam filters work

Different spam filtering formulas have been employed by Gmail, Outlook.com and Yahoo Mail to deliver only the valid emails to their users and filter out the illegitimate messages. Conversely, these filters also sometimes erroneously block authentic messages. It has been reported that about 20 percent of authorization based emails usually fail to get to the inbox of the expected recipient. The email providers have designed various mechanisms for use in email anti-spam filter to curtail the dangers posed by phishing, email-borne malware and ransomware to email users. The mechanisms are used to decide the risk level of each incoming email. Examples of such mechanisms include satisfactory spam limits, sender policy frameworks, whitelists and blacklists, and recipient verification tools. These mechanisms can be used by single or multiple users. When the satisfactory spam thresholds is too low it can lead to more spam evading the spam filter and entering the users' inboxes. Meanwhile having a very high threshold can lead to some important emails being

isolated unless the administrator redirects them. This section discusses the operations of Gmail, Yahoo and Outlook emails anti-spam filters.

3.1.1.1. Gmail filter spam

Google's data centers makes use of hundreds of rules to determine whether an email is valid or spam. Every one of these rules depicts specific features of a spam and certain statistical value is connected with it, depending on the likelihood that the feature is a spam. The weighted importance of each feature is then used to construct an equation. A test is conducted using the score against a sensitivity threshold decided by each user's spam filter. And consequently, it is classified as a lawful or spam email. Google is said to be using state of the art spam detection machine learning algorithms such as logistic regression and neural networks in its classification of emails. Gmail also use optical character recognition (OCR) to shield Gmail users from image spam. Also, machine-learning algorithms developed to combine and rank large sets of Google search results allow Gmail to link hundreds of factors to improve their spam classification. The evolving nature of spam over time revolves around factors such as domain reputation, links in message headers and others. These can make messages to unexpectedly end up in the spam folder. Spam filtering principally works on the foundation of “filters” settings that are continuously updated with the emergence of state of the art tools, algorithms, discovery of new spam and the feedback from Gmail users about likely spammers. Many spam filters employ text filters to

eradicate hazards posed by spammers depending on the senders and their history.

3.1.1.2. Yahoo mail filter spam

Yahoo mail is the first free webmail providers in the world with over 320 million users. The email provider has its own spam algorithms that it uses to detect spam messages. The basic methods used by Yahoo to detect spam messages include: URL filtering, email content and spam complaints from users. Unlike Gmail, Yahoo filter emails messages by domains and not IP address. Yahoo mail uses combination of techniques to filter out spam messages. It also provide mechanisms that prevent a valid user from being mistaken for a spammer. Examples are ability of the users to troubleshoot SMTP Errors by referring to their SMTP logs. Another one is the complaint feedback loop service that helps a user to maintain a positive reputation with Yahoo. Yahoo whitelisting (internal whitelisting and Return Path Certification) is also provided. Unlike blacklisting, a whitelist blocks by letting the user specify the list of senders to receive mail from. The addresses of such senders are placed on a trusted-users list. Yahoo mail spam filters allows the user to use a combination of whitelist and other spam-fighting feature as a way to reduce the number of valid messages that are erroneously classified as spam. On the other hand, using whitelist alone will make the filter to be very strict and the implication is that any unapproved user would be blocked automatically. Many anti-spam systems use automatic whitelist. In this case, an anonymous sender's email address is

checked against a database; if there is no history of spamming, their message is sent to the recipient's inbox and they are added to the whitelist.

3.1.1.3. Outlook email spam filter

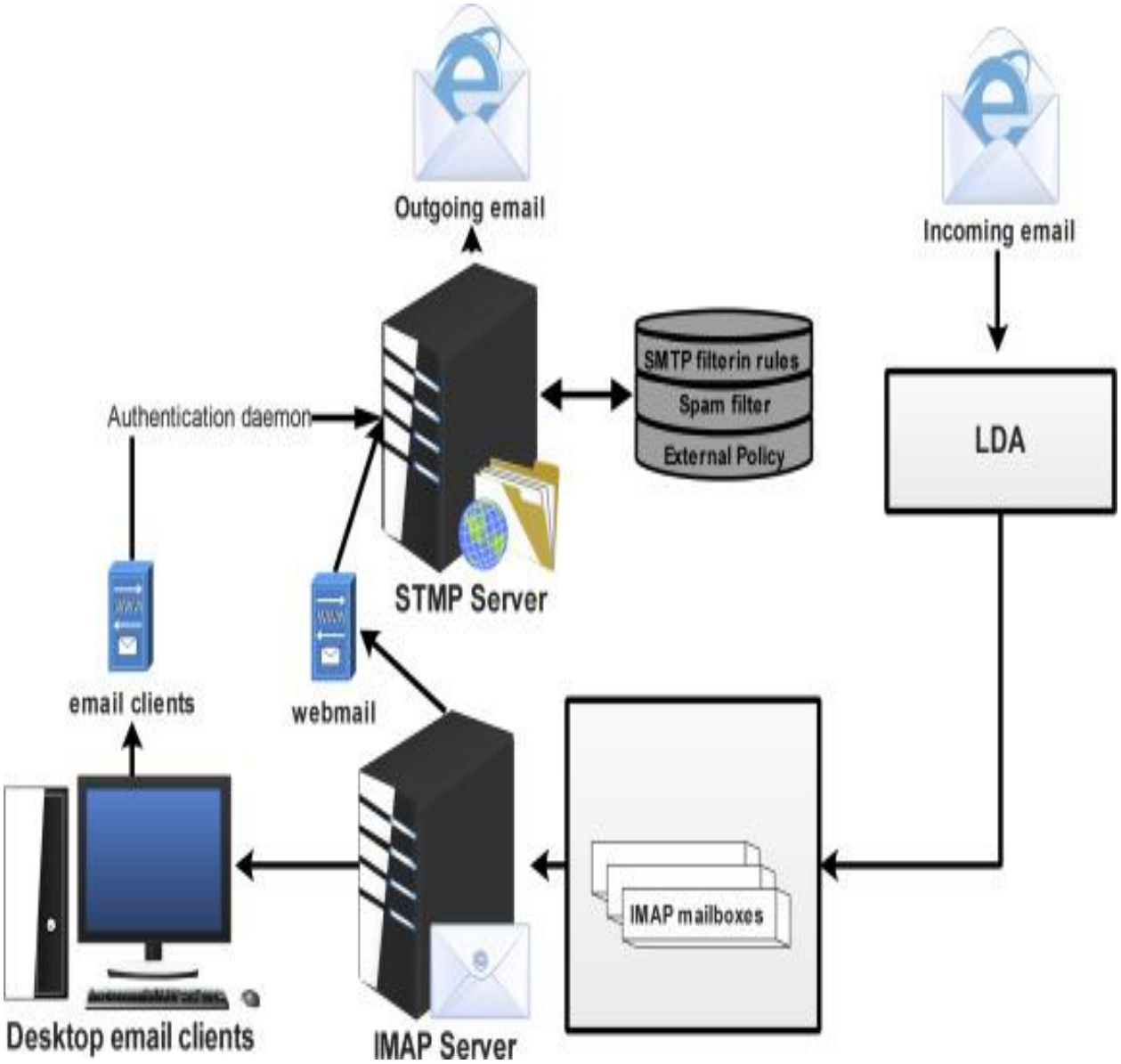
After Gmail and Yahoo mail, we discussed Outlook from Microsoft in this section and how it handles spam filtering. In 2013, Microsoft changed the name of Hotmail and Windows Live Mail to Outlook.com. Outlook.com was patterned after Microsoft's Metro design language and directly imitates the interface of Microsoft Outlook. Outlook.com is a collection of applications from Microsoft, one of which is Outlook webmail service. Outlook webmail service allows the users to send and receive emails in their web browser. It allows the users to connect cloud storage services to their account so that when they want to send an email with file attachments, they can select files from not only their computer and OneDrive account but also from Google Drive, Box, and Dropbox account. Moreover, Outlook webmail service also allows users to encrypt their email messages and disallow the recipient from forwarding the email. Whenever a message is encrypted in Outlook.com, it is only the person with the password that will be able to decrypt the message and read it. This is a security measure that guarantees that only the intended recipient is permitted to read the message. The main difference between Outlook.com webmail service and the MS Outlook desktop application is that Outlook desktop application allows you to send and receive emails, via an email server,

while Outlook.com is an email server. Outlook.com webmail service on-the-other-hand is for business and professionals who rely on email. Moreover, MS Outlook desktop application is a commercial software that comes along with the Microsoft Office package. It is a computer software program that provides services like email management, address book, notebook, a web browser and a calendar which allows users to plan their programmes and arrange upcoming meetings. About 400 million users are using Outlook.com. Statistics shows that their site receives about eight billion emails a day and out of which 30%–35% of those emails are delivered to the users' inboxes. Outlook.com have its own distinctive methods of filtering email spams.

3.2. Email spam filtering process

An email message is made up of two major components which are the header and the body. The header is the area that have broad information about the content of the email. It includes the subject, sender and receiver. The body is the heart of the email. It can include information that does not have a pre-defined data. Examples include web page, audio, video, analog data, images, files, and HTML markup. The email header is comprised of fields such as sender's address, the recipient's address, or timestamp which indicate when the message was sent by intermediary servers to the Message Transport Agents (MTAs) that function as an office for organising mails. The header line usually starts with a “From” and it goes

through some modification whenever it moves from one server to another through an in-between server. Headers allow the user to view the route the email passes through, and the time taken by each server to treat the mail. The available information have to pass through some processing before the classifier can make use of it for filtering .
Figure below depicts a mail server architecture and how spam filtering is done.



The necessary stages that must be observed in the mining of data from an email message can be categorised into the following:

- **Pre-processing:** This is the first stage that is executed whenever an incoming mail is received. This step consists of tokenization.
- **Tokenization:** This is a process that removes the words in the body of an email. It also transforms a message to its meaningful parts. It takes the email and divides it into a sequence of representative symbols called tokens. Subramaniam, Jalab and Taqa emphasised that these representative symbols are extracted from the body of the email, the header and subject. Guzella and Caminhas asserted that the process of replacing information with distinctive identification symbols will extricate all the characteristics and words from the email exclusive of taking into account the meaning
- **Feature selection:** Sequel to the pre-processing stage is the feature selection phase. Feature selection a kind of reduction in the measure of spatial coverage that effectively exemplifies fascinating fragments of email message as a compressed feature vector. The technique is beneficial when the size of the message is large and a condensed feature representation is needed to make the task of text or image matching snappy . Advance fee fraud, including inheritance, lottery, visa and customs-clearance scams, Romance scams, including marketing sex enhancement drugs to cure erection dysfunctional,

online dating, military scams, Ads for porn sites, Ads for miscellaneous external sites, earning big money through “work-from-home” jobs, online shopping, pleading and gift requests, business proposals and others. Some of the most important features for spam filtering include: Message body and subject, Volume of the message, Occurrence count of words, Circadian patterns of the message (spam messages usually have many semantic discrepancies), Recipient age, Sex and country, Recipient replied (indicates whether the recipient replied to the message), Adult content and Bag of words from the message content. Sender Account Features used for spam filtering include: Sender Country (The distribution of countries as stated by users on their profile and as revealed by their IP address), Sender IP address, Sender Email, Sender & Recipient Age, Sender Reputation. The less important features are: Geographical distance between sender and receiver, Sender's date of birth, Username and password of the sender, Account lifespan, Sex of sender and Age of recipient. The recognition of spam e-mails with minimum number of features is important in view of computational complexity and time. Feature selection involves processes like stemming, noise removal and stop word removal steps.

3.3. Publicly available email spam corpus

The dataset contained in a corpus plays a crucial role in assessing the performance of any spam filter. Though there are many conventional datasets that are usually used for classifying text, it is just of recent

that some researchers in the field of spam filtering are making the corpus used for evaluating the effectiveness of their proposed filter available to the public. A comprehensive list of the corpora made available to the public in the different techniques reviewed in this paper are in . Individual corpus possesses incredibly distinctive qualities which are indicated by the related information applied in the experiments conducted to evaluate the performance of the spam filter.

Table 2. Publicly available email spam corpus.

Dataset name	Number of messages		Rate of spam	Year of creation	References
	Spam	Non-spam			
Spam archive	15090	0	100%	1998	Almeida and yamakami
Spambase	1813	2788	39%	1999	Sakkis et al
Lingspam	481	2412	17%	2000	Sakkis et al
PU1	481	618	44%	2000	Attar et al
Spamassassin	1897	4150	31%	2002	Apache spamassassin
PU2	142	579	20%	2003	Zhang et al
PU3	1826	2313	44%	2003	Zhang et al
PUA	571	571	50%	2003	Zhang et al
Zh1	1205	428	74%	2004	Zhang et al
Gen spam	31,196	9212	78%	2005	Cormack and lynam
Trec 2005	52,790	39,399	57%	2005	Androutsopoulos et al
Biggio	8549	0	100	2005	Biggio et al
Phishing corpus	415	0	100	2005	Abu-nimeh et al
Enron-spam	20170	16545	55%	2006	Koprinska et al

Dataset name	Number of messages		Rate of spam	Year of creation	References
	Spam	Non-spam			
Trec 2006	24,912	12,910	66%	2006	Androutsopoulos et al
Trec 2007	50,199	25,220	67%	2007	Debarr and wechsler
Princeton spam image Benchmark	1071	0	100%	2007	Wang et al
Dredze image spam Dataset	3297	2021	62%	2007	Dredze, gevaryahu and elias-bachrach
Hunter	928	810	53%	2008	Gao et al
Spamemail	1378	2949	32%	2010	Csmininggroup

4. Analysis

4.1. Performance evaluation measures

Spam filters are usually evaluated on large databases containing ham and spam messages that are publicly available to users. An example of the performance measures that are used is classification accuracy (Acc). It is the comparative number of messages rightly classified, the percentage of messages rightly classified is used as an added measure for evaluating performance of the filter. It has however been highlighted that using Accuracy as the only performance indices is not sufficient. Other performance metrics such as recall, precision and derived measures used in the field of information retrieval must be considered, so also is false

positives and false negatives used in decision theory. This is very important because of the costs attached to misclassification. When a spam message is wrongly classified as ham, it gives rise to a somewhat insignificant problem, because the only thing the user need to do is to delete such message. In contrast, when a non-spam message is wrongly labeled as Spam, this is obnoxious, because it indicates the possibility of losing valuable information as a result of the filter's classification error. This is very imperative especially in settings where Spam messages are automatically deleted. Therefore, it is inadequate to evaluate the performance of any Machine Learning algorithm used in spam filter using classification accuracy exclusively. Furthermore, in a setting that is lopsided or biased where the number of spam messages utilized for testing the performance of the filter is very much higher than that of ham messages, the classifier can record a very high accuracy by concentrating on the detection of spam messages solely. In a real world environment where there is nothing like zero probability of wrongly categorizing a ham message, it is required that a compromise be reached between the two kinds of errors, depending on the predisposition of user and the performance indicators used. The formulae for calculating the classification accuracy and classification error are depicted in Eqs. (1) and (2) below:

Assuming

- NH = Number of non-spam messages to be classified

- NS = Number of spam messages to be classified

5. Materials & methods

Of recent, spam mail classification is normally handled by machine learning (ML) algorithms intended to differentiate between spam and non-spam messages. Machine learning algorithms achieve this by using an automatic and adaptive technique. Rather than depending on hand-coded rules that are susceptible to the perpetually varying characteristics of spam messages, ML methods have the capacity to obtain information from a set of messages provided, and then use the acquired information to classify new messages that it just received. According to [53], ML algorithms have the capacity to perform better based on their experience. In this section we will review some of the most popular machine learning methods that have been applied to spam detection.

5.1. Clustering technique

Clustering deals with classifying a group of patterns into related classes. Clustering is a type of approach used in dividing objects or case examinations into comparatively similar collections known as clusters. Clustering techniques have drawn the attention of many researchers and academicians of recent and it has been applied in different fields of application. Clustering algorithms which are unsupervised learning tools are used on e-mail spam datasets

which usually have true labels. Provided there are appropriate representations, a good number of clustering algorithms have the ability to classify e-mail spam datasets into either ham or spam clusters. Whissell and Clarke [54] proved this in their research work on e-mail spam clustering. The outputs were remarkably noteworthy as their technique performed better than those of existing modern semi-supervised techniques, thereby demonstrating that clustering can be a formidable technique for filtering spam e-mails. It classifies objects or opinions in such a manner that objects in the same group are more similar to each other than to those in other group. Two types of clustering methods that have been used for spam classification are density-based clustering and K-nearest neighbours (kNN). In [55], density based clustering is another method of document clustering which has been exploited to solve spam classification problem. As asserted by the authors, the method have the capacity to process encrypted messages, thereby upholding its confidentiality. The success of the technique is subject to its ability to locate sensitive comparators. The comparators are normally characterised with either speed or sensitivity. Locating the comparators that have speed and are sufficiently sensitive is the major barrier to the success of this technique.

kNN is a distribution free method, which does not rely on assumptions that the data is drawn from a given probability distribution [56]. This is rather important because in the real world, nearly all of the applied data disobey the standard hypothetical

postulations made (such as Gaussian mixture, linearly separable, and others). Non-parametric algorithms like kNN can be used to salvage such situation. In kNN classifier, the classification model is not built from data, rather classification is carried out by matching the test instance with K training examples and decision is made as to which group it belong to depending on the resemblance to K closest neigbours [57]. The kNN is termed a lazy learner since the training data points is not used by it to perform generalisation. Simply put, there is no obvious training stage and if it exists it is extremely small. The implication is that the algorithm has a moderately speedy training phase. The absence of universality necessitates kNN to store all the training data. To be precise, the entire training data is required throughout the testing phase as decisions are made based on the complete training data set. The contradiction is quite clear here that there is no significant training stage, rather there is an extensive testing stage. There is an overhead cost of both time and memory. Additional time may possibly be required in the most awful case. Added memory is required to store all training data neigbours [57]. The authors in [58] opined that some of the strengths of kNN algorithm includes: there is no explicit training phase or it is very minimal. Once data is loaded into memory, it begins its classification process.

In [58], the steps involved in a simple kNN algorithm for filtering spam mails is described in the algorithm below. Here Neighbours(*d*) return the k nearest neighbours of d, Closest (*d*, *t*) return the closest

elements of t in d, and test Class(S) return the class label of S. A simple kNN algorithm for spam email classification is in the algorithm below:

Algorithm 1 kNN Algorithm for Spam Email Classification

1: Find Email Message class labels.
2: **Input** *k*, the number of nearest neighbors
3: **Input** *D*, the set of test Email Message;
4: **Input** *T*, the set of training Email Message.
5: *L*, the label set of test Email Message.
6: Read Data File (Training Data)
7: Read Data File (Testing Data)
8: **for** each *d* in *D* and each *t* in *T* **do**
9: Neighbors(*d*) = {}
10: **if** |Neighbors (*d*) | < *k* **then**
11: Neighbors(*d*) = Closest (*d*, *t*)∪ Neighbors(*d*)
12: **end if**
13: **if** |Neighbors(*d*)| ≥ *k* **then**
14: *restrain(M, xj, yj)*
15: **end if**
16: **end for** 17: **return** Final Email Message Classification (Spam/Valid email)
18: **end**

Source code

```
import numpy as np # scientific computation
import pandas as pd # loading dataset file
import matplotlib.pyplot as plt # Visulization
import nltk  # Preprocessing our text
from nltk.corpus import stopwords # removing all the stop words
from nltk.stem.porter import PorterStemmer # stemming of words
```

```
#Load our dataset
df = pd.read_csv("spam.csv",encoding="latin")
df.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

```
#Give concise summary of a DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

```
]: #Returns the sum fo all na values
   df.isna().sum()
```

```
]: v1              0
   v2              0
   Unnamed: 2    5522
   Unnamed: 3    5560
   Unnamed: 4    5566
   dtype: int64
```

```
df.rename({"v1":"label","v2":"text"},inplace=True,axis=1)
```

```
# bottom 5 rows of the dataframe
df.tail()
```

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['label'] = le.fit_transform(df['label'])
```

```
#Splitting data into train and validation sets using train_test_split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state = 0)

##train size 80% and test size 20%
```

Given data is imbalanced one, we are balancing the data

```
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

# import SMOTE module from imblearn library
# pip install imblearn (if you don't have imblearn in your system)
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_resample(X_train, y_train.ravel())

print('After OverSampling, the shape of train_X: {}'.format(X_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))

print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
Before OverSampling, counts of label '1': 581
Before OverSampling, counts of label '0': 3876

After OverSampling, the shape of train_X: (7752, 7163)
After OverSampling, the shape of train_y: (7752,)

After OverSampling, counts of label '1': 3876
After OverSampling, counts of label '0': 3876
```

```
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\smart\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
True
```

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
```

```
import re
corpus = []
length = len(df)
```

```
for i in range(0,length):
    text = re.sub("[^a-zA-Z0-9]"," ",df["text"][i])
    text = text.lower()
    text = text.split()
    pe = PorterStemmer()
    stopword = stopwords.words("english")
    text = [pe.stem(word) for word in text if not word in set(stopword)]
    text = " ".join(text)
    corpus.append(text)
```

[18]:
```
corpus
```

[18]:
```
['go jurong point crazi avail bugi n great world la e buffet cine got amor wat',
 'ok lar joke wif u oni',
 'free entri 2 wkli comp win fa cup final tkt 21st may 2005 text fa 87121 receiv entri question std txt rate c appli 08452810
075over18',
 'u dun say earli hor u c alreadi say',
 'nah think goe usf live around though',
 'freemsg hey darl 3 week word back like fun still tb ok xxx std chg send 1 50 rcv',
 'even brother like speak treat like aid patent',
 'per request mell mell oru minnaminungint nurungu vettam set callertun caller press 9 copi friend callertun',
 'winner valu network custom select receivea 900 prize reward claim call 09061701461 claim code kl341 valid 12 hour',
 'mobil 11 month u r entitl updat latest colour mobil camera free call mobil updat co free 08002986030',
 'gonna home soon want talk stuff anymor tonight k cri enough today',
 'six chanc win cash 100 20 000 pound txt csh11 send 87575 cost 150p day 6day 16 tsandc appli repli hl 4 info',
 'urgent 1 week free membership 100 000 prize jackpot txt word claim 81010 c www dbuk net lccltd pobox 4403ldnw1a7rw18',
 'search right word thank breather promis wont take help grant fulfil promis wonder bless time',
 'date sunday',
 'xxxmobilemovieclub use credit click wap link next txt messag click http wap xxxmobilemovieclub com n qjkgighjjgcbl',
 'oh k watch',
 'eh u rememb 2 spell name ye v naughti make v wet',
```

[19]:
```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=35000)
X = cv.fit_transform(corpus).toarray()
```

```
import pickle ## importing pickle used for dumping models
pickle.dump(cv, open('cv1.pkl', 'wb')) ## saving to into cv.pkl file
```

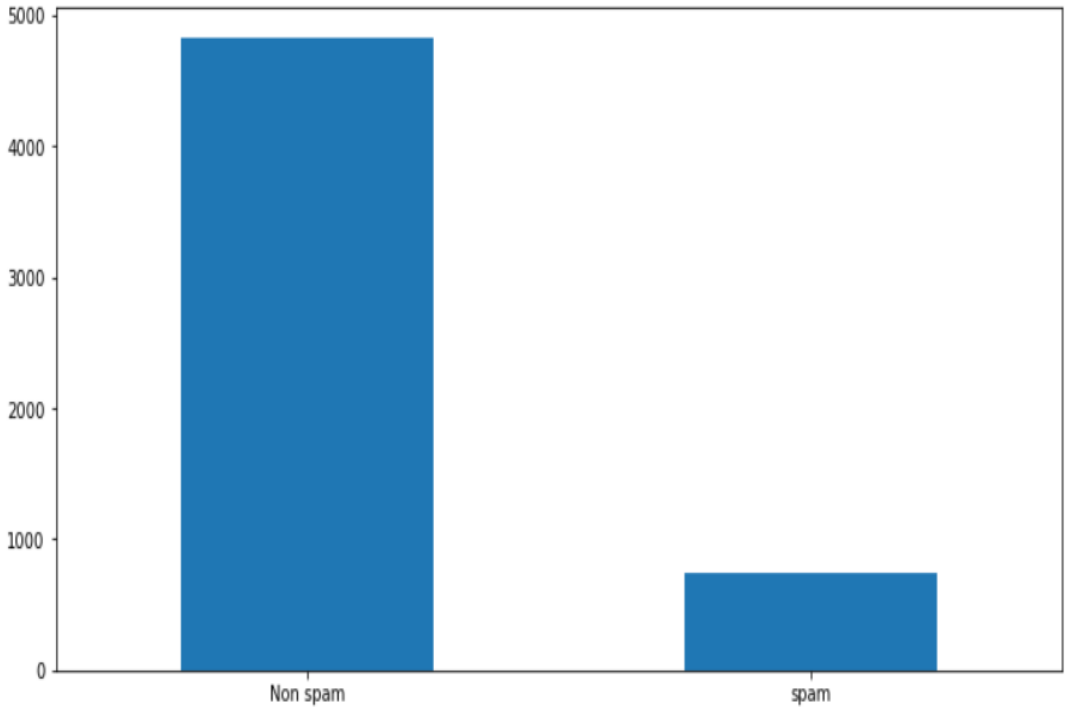
```
df.describe()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
count	5572	5572	50	12	6
unique	2	5169	43	10	5
top	ham	Sorry, I'll call later	bt not his girlfrnd... G o o d n i g h t . . .@"	MK17 92H. 450Ppw 16"	GNT:-)"
freq	4825	30	3	2	2

```
df.shape
```

```
(5572, 5)
```

```
df["label"].value_counts().plot(kind="bar",figsize=(12,6))
plt.xticks(np.arange(2), ('Non spam', 'spam'),rotation=0);
```



```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_train_res, y_train_res)
```

```
DecisionTreeClassifier()
```

```
from sklearn.ensemble import RandomForestClassifier
model1 =RandomForestClassifier()
model1.fit(X_train_res, y_train_res)
```

```
RandomForestClassifier()
```

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
```

```
#Fitting the model to the training sets
model.fit(X_train_res, y_train_res)
```

```
MultinomialNB()
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
#Fitting the model to the training sets
model = Sequential()
```

```
X_train.shape
```

```
(4457, 7163)
```

```
model.add(Dense(units  = X_train_res.shape[1],activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(Dense(units=100,activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(Dense(units=100,activation="relu",kernel_initializer="random_uniform"))
```

```
model.add(Dense(units=1,activation="sigmoid"))
```

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=['accuracy'])
```

```
generator = model.fit(X_train_res,y_train_res,epochs=10,steps_per_epoch=len(X_train_res)//64)
```

```
generator = model.fit(X_train_res,y_train_res,epochs=10,steps_per_epoch=len(X_train_res)//64)
```

```
Epoch 1/10
121/121 [==============================] - 26s 203ms/step - loss: 0.1404 - accuracy: 0.9565
Epoch 2/10
121/121 [==============================] - 24s 199ms/step - loss: 0.0220 - accuracy: 0.9950
Epoch 3/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0155 - accuracy: 0.9965
Epoch 4/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0163 - accuracy: 0.9962
Epoch 5/10
121/121 [==============================] - 24s 195ms/step - loss: 0.0158 - accuracy: 0.9965
Epoch 6/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0132 - accuracy: 0.9974
Epoch 7/10
121/121 [==============================] - 23s 194ms/step - loss: 0.0130 - accuracy: 0.9972
Epoch 8/10
121/121 [==============================] - 24s 196ms/step - loss: 0.0120 - accuracy: 0.9974
Epoch 9/10
121/121 [==============================] - 23s 193ms/step - loss: 0.0103 - accuracy: 0.9977
Epoch 10/10
111/121 [==========================>...] - ETA: 1s - loss: 0.0128 - accuracy: 0.9972WARNING:tensorflow:Your input ran out of da
ta; interrupting training. Make sure that your dataset or generator can generate at least `steps_per_epoch * epochs` batches (i
n this case, 1210 batches). You may need to use the repeat() function when building your dataset.
121/121 [==============================] - 23s 193ms/step - loss: 0.0128 - accuracy: 0.9972
```

```
: def new_review(new_review):
    new_review = new_review
    new_review = re.sub('[^a-zA-Z]', ' ', new_review)
    new_review = new_review.lower()
    new_review = new_review.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    new_review = [ps.stem(word) for word in new_review if not word in   set(all_stopwords)]
    new_review = ' '.join(new_review)
    new_corpus = [new_review]
    new_X_test = cv.transform(new_corpus).toarray()
    print(new_X_test)
    new_y_pred = loaded_model.predict(new_X_test)
    print(new_y_pred)
    new_X_pred = np.where(new_y_pred>0.5,1,0)
    return new_y_pred
new_review = new_review(str(input("Enter new review...")))
```

```
Enter new review...hello fg hou hkl
[[0 0 0 ... 0 0 0]]
1/1 [==============================] - 0s 45ms/step
[[0.9784973]]
```

```
from sklearn.metrics import confusion_matrix,accuracy_score, classification_report
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test,y_pred)
print(cm)
print('Accuracy Score Is Naive Bayes:- ' ,score*100)
```

```
[[935  14]
 [ 13 153]]
Accuracy Score Is:-  97.57847533632287
```

```
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test,y_pred)
print(cm)
print('Accuracy Score Is:- ' ,score*100)

cm1 = confusion_matrix(y_test, y_pred1)
score1 = accuracy_score(y_test,y_pred1)
print(cm1)
print('Accuracy Score Is:- ' ,score1*100)
```

```
[[796 153]
 [ 17 149]]
Accuracy Score Is:-  84.75336322869956
[[855  94]
 [ 14 152]]
Accuracy Score Is:-  90.31390134529148
```

```
121/121 [==============================] - 24s 196ms/step - loss: 0.0120 - accuracy: 0.9974
Epoch 9/10
121/121 [==============================] - 23s 193ms/step - loss: 0.0103 - accuracy: 0.9977
Epoch 10/10
111/121 [==========================>...] - ETA: 1s - loss: 0.0128 - accuracy: 0.9972WARNING:tens
```

```
from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy Score Is:- ' ,score*100)
```

```
[[937  12]
 [ 16 150]]
Accuracy Score Is:-  97.48878923766816
```

```
]: from sklearn.metrics import confusion_matrix,accuracy_score
   cm = confusion_matrix(y_test, y_pr)
   score = accuracy_score(y_test,y_pr)
   print(cm)
   print('Accuracy Score Is:- ' ,score*100)
```

```
[[937  12]
 [ 16 150]]
Accuracy Score Is:-  97.48878923766816
```

Saving our model

By comparing the all the model , we can come to a conclusion that ANN is the best model

```
]: model.save('spam.h5')
```

```
# Importing essential libraries
from flask import Flask, render_template, request
import pickle
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from tensorflow.keras.models import load_model
# Load the Multinomial Naive Bayes model and CountVector
```

```
# Load the Multinomial Naive Bayes model
loaded_model = load_model('spam.h5')
cv = pickle.load(open('cv1.pkl','rb'))
app = Flask(__name__)
```

```
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

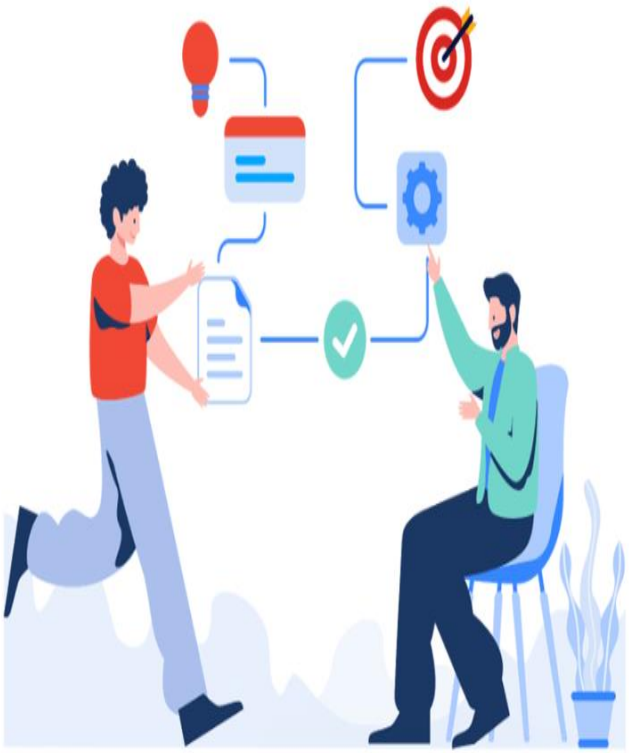
```
@app.route('/Spam',methods=['POST','GET'])
def prediction(): # route which will take you to the prediction page
    return render_template('spam.html')

@app.route('/predict',methods=['POST'])
def predict():
    if request.method == 'POST':
        message = request.form['message']
        data = message

    new_review = str(data)
    print(new_review)
    new_review = re.sub('[^a-zA-Z]', ' ', new_review)
    new_review = new_review.lower()
    new_review = new_review.split()
    ps = PorterStemmer()
    all_stopwords = stopwords.words('english')
    all_stopwords.remove('not')
    new_review = [ps.stem(word) for word in new_review if not word in   set(all_stopwords)]
    new_review = ' '.join(new_review)
    new_corpus = [new_review]
    new_X_test = cv.transform(new_corpus).toarray()
    print(new_X_test)
    new_y_pred = loaded_model.predict(new_X_test)
    new_X_pred = np.where(new_y_pred>0.5,1,0)
    print(new_X_pred)
    if new_review[0][0]==1:
       return render_template('result.html', prediction="Spam")
    else :
       return render_template('result.html', prediction="Not a Spam")
```

SCREEN LAYOUTS

127.0.0.1:5000/#about

SPAM DETECTION

Spam Detector for Short Message Service (SMS).



127.0.0.1:5000/#about

SmartBridge

Home About

Spam

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion dollars industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration of user. So Spam SMS is one of the major issues in wireless communication world and it grows day by day.

Today most of the SMS's are Spam SMS which consists of Credit Card offer, discount offers, traffic plans, promotions etc. Spam messages include advertisements, free services, promotions, awards, etc. People are using the ubiquity of mobile phone devices is expanding day by day as they give a vast variety of services by reducing the cost of services.To avoid such Spam SMS people use white and black list of numbers. But this technique is not adequate to completely avoid Spam SMS. To tackle this problem it is needful to use a smarter technique which correctly identifies Spam SMS. Natural langugae processing technique is useful for Spam SMS identification. It analyses text content and find patterns which are used to identify Spam and Non-Spam SMS.

Conclusion

In this study, we reviewed machine learning approaches and their application to the field of spam filtering. A review of the state of the art algorithms been applied for classification of messages as either spam or ham is provided. The attempts made by different researchers to solving the problem of spam through the use of machine learning classifiers was discussed. The evolution of spam messages over the years to evade filters was examined. The basic architecture of email spam filter and the processes involved in filtering spam emails were looked into. The paper surveyed some of the publicly available datasets and performance metrics that can be used to measure the effectiveness of any spam filter. The challenges of the machine learning algorithms in efficiently handling the menace of spam was pointed out and comparative studies of the machine learning technics available in literature was done. We also revealed some open research problems associated with spam filters. In general, the figure and volume of literature we reviewed shows that significant progress have been made and will still be made in this field. Having discussed the open problems in spam filtering, further research to enhance the effectiveness of spam filters need to be done. This will make the development of spam filters to continue to be an active research field for academician and industry practitioners researching machine learning techniques for effective spam filtering. Our

hope is that research students will use this paper as a spring board for doing qualitative research in spam filtering using machine learning, deep leaning and deep adversarial learning algorithms.