

Assignment 4

Problem Statement:

Implement e-mail spam filtering using text classification algorithm with appropriate dataset.

Objective:

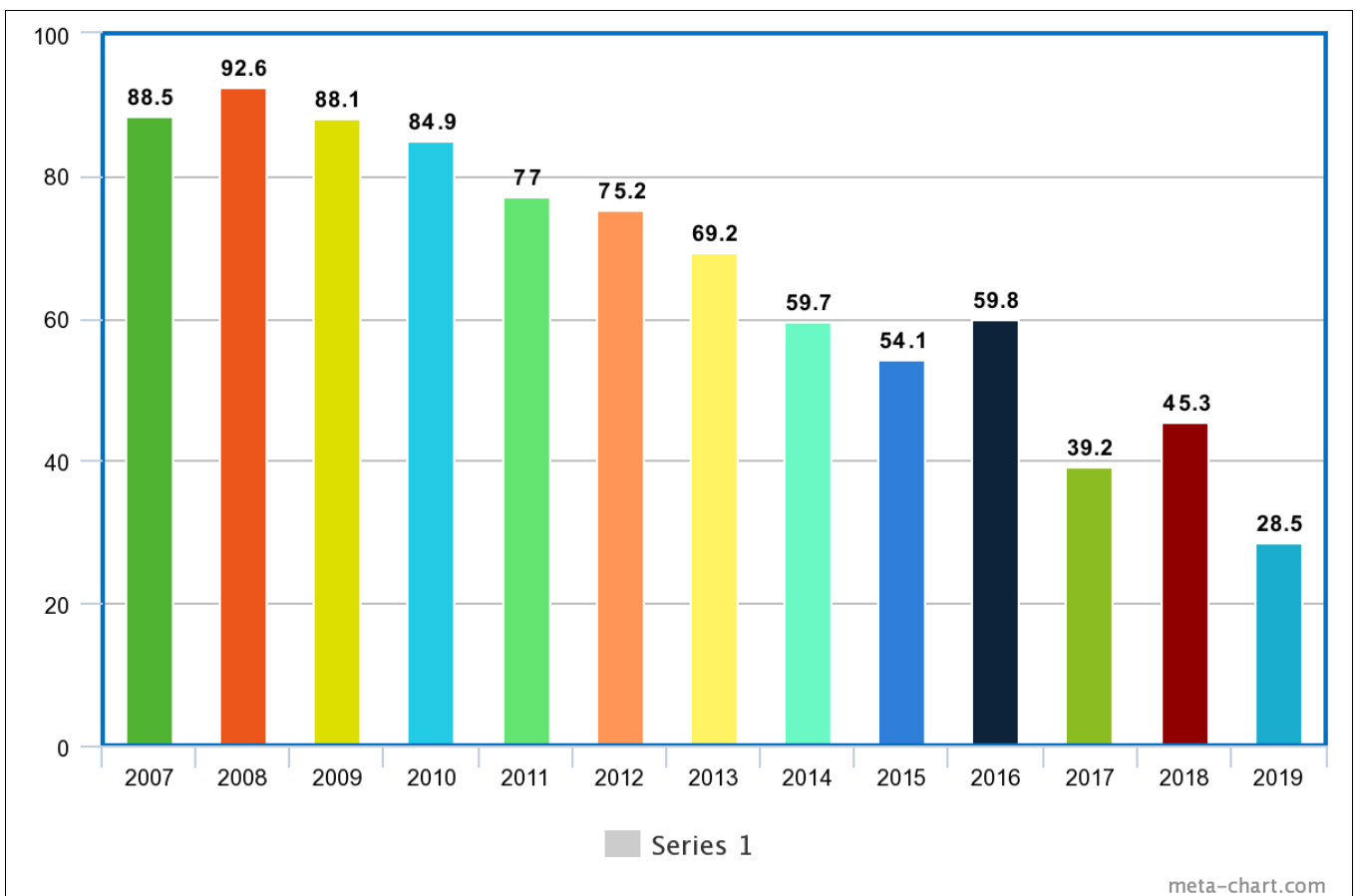
1. Basic concepts of Spam Filtering.
2. To study KNN algorithm.

Theory:

In the new era of technical advancement, electronic mail (emails) has gathered significant users for professional, commercial, and personal communications. In 2019, on average, every person **received 130 emails each day**, and overall, **296 Billion** emails were sent that year.

Because of the high demand and huge user base, there is an upsurge in unwanted emails, also known as spam emails. At times, more than **50% of the total emails were spam**. Even today, people lose millions of dollars to fraud every day.

But, in the figure shown below, it can be observed that the quantity of such emails has decreased significantly after 2016 because of the evolution of the software that can detect these spam emails and can filter them out.



Percentage of emails marked as Spam (Source: **Statista**)

Steps to implement a Spam-classifier using the k-NN algorithm:

Many several techniques are present in the market to detect spam emails. If we want to classify broadly, there are five different techniques based on which algorithms decide whether any mail is spam.

Content-Based Filtering Technique

Algorithms analyze words, the occurrence of words, and the distribution of words and phrases inside the content of emails and segregate them into spam and non-spam categories.

Case Base Spam Filtering Method

Algorithms trained on well-annotated spam/non-spam marked emails try to classify incoming emails into two categories.

Heuristic or Rule-Based Spam Filtering Technique

Algorithms use pre-defined rules as regular expressions to give a score to the messages in the emails. They segregate emails into spam and non-spam categories based on the scores generated.

The Previous Likeness Based Spam Filtering Technique

Algorithms extract the incoming mails' features and create a multi-dimensional space vector and draw points for every new instance. Based on the KNN algorithm, these new points get assigned to the closest class of spam and non-spam.

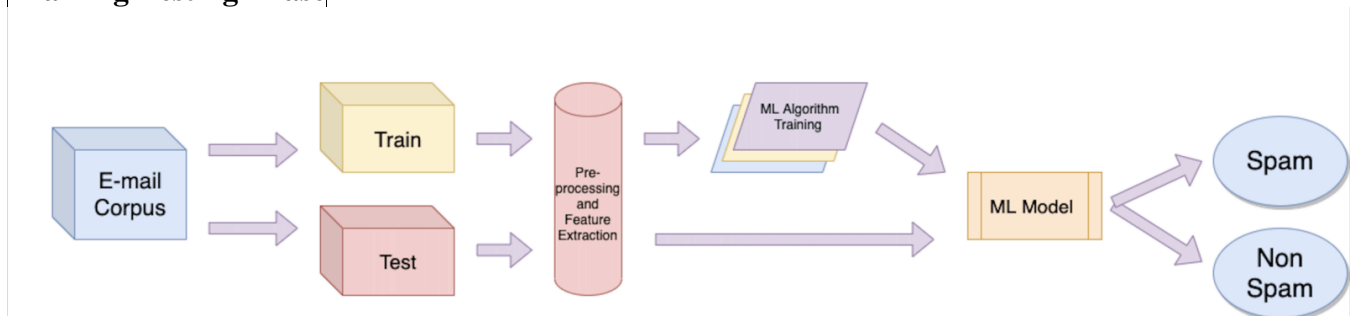
Adaptive Spam Filtering Technique

Algorithms classify the incoming emails into various groups and, based on the comparison scores of every group with the defined set of groups, spam and non-spam emails got segregated.

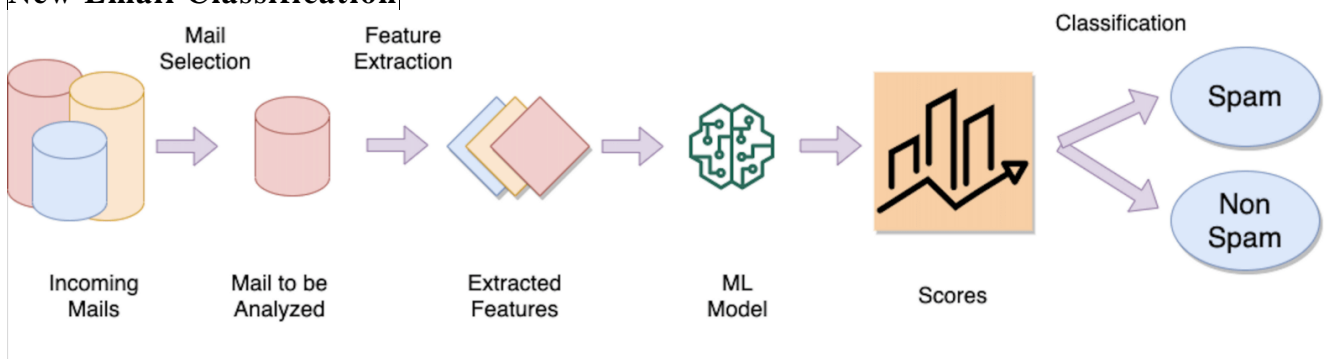
K-NN based algorithms:

K-NN based algorithms are widely used for clustering tasks. Let's quickly know the entire architecture of this implementation first and then explore every step. Executing these **five steps**, one after the other, will help us implement our spam classifier smoothly.

Training Testing Phase



New Email Classification



Step 1: E-mail Data Collection

The dataset contained in a corpus plays a crucial role in assessing the performance of any spam filter. Many open-source datasets are freely available in the public domain. The two datasets below are widely popular as they contain many emails.

Enron corpus datasets (Created in 2006 and having 55% spam emails)










Trec 2007 dataset (Created in 2007 and having 67% spam emails)

Train/Test Split: Split the dataset into train and test datasets but make sure that both sets balance the numbers of ham and spam emails (ham is a fancy name for non-spam emails).

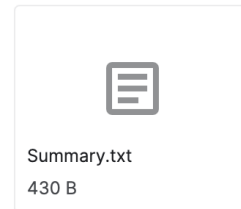
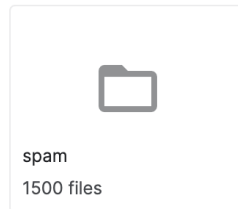
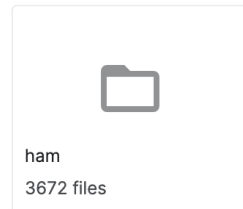
Enron Corpus Dataset on Kaggle

Data Explorer

49.49 MB

- ▼  enron1
 - ▶  ham
 - ▶  spam
 - ▶  Summary.txt
- ▶  enron2
- ▶  enron3
- ▶  enron4
- ▶  enron5
- ▶  enron6

< **enron1** (2 directories, 1 files)



Step 2: Pre-processing of E-mail content

As the dataset is in text format, we will need the pre-processing of text data. At this step, we mainly perform tokenization of mail. Tokenization is a process where we break the content of an email into words and transform big messages into a sequence of representative symbols termed tokens. These tokens are extracted from the email body, header, subject, and image.

These days, senders have options to attach inline images to the mail. These emails can be categorized as spam, not based on their mail content but on the images' content. This was not an easy task until google came up with the open-source library Tesseract. This library extracts the words from images automatically with certain accuracy. But still, Times New Roman and Captcha words are challenging to read automatically.

Step 3: Feature Extraction and Selection

After pre-processing, we can have a large number of words. Here we can maintain a database that contains the frequency of the different words represented in each column. These attributes can be categorized on another basis, like:

Essential attributes: Frequency of repeated words, Number of semantic discrepancies, an Adult content bag of words, etc.

Additional Attributes: Sender account features like Sender country, IP address, email, age of Sender, Number of replies, Number of recipients, and website address.

Note: These web addresses are converted in the word format only. For example, https://www.google.com/ can be converted to "HTTP google." Sometimes these processes are called Normalization.

Less important attributes: Geographical distance between sender and receiver, Sender's date of birth, Account lifespan, Sex of Sender, and Age of the recipient.

You must be clear that the more the number of attributes, → more the time complexity of the model.

These attributes can be tremendous, so techniques like Stemming, noise removal, and stop-word removal can be used. One of the famous stemming algorithms is the Porter-Stemmer Algorithm.

Some general things that we do in stemming are:

Removing suffixes (-ed, -ing, -full, -ness, etc.)

Removing prefixes (Un-, Re-, Pre-, etc.)

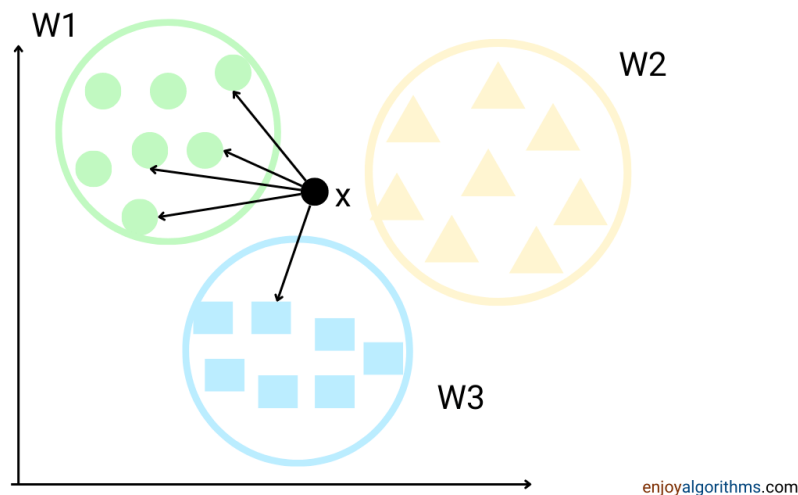
List of stop words ["i", "me", "my", "him", "his", "himself", "she", "her", "hers", "herself",

Example dataset format

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

Step 4: KNN (K-Nearest Neighbour) Implementation

Similar to the Nearest Neighbour algorithm, the K-Nearest Neighbour algorithm serves the purpose of clustering. Still, instead of giving just one nearest instance, it looks at the closest K instances to the new incoming instance. K-NN classifies the new instances based on the frequency of those K instances. The value of K is considered to be a hyper parameter that needs tuning. To tune this, one can take one of the famous Hit and Trial approaches, where we try some K values and then check the model's performance.



To find the nearest instance, one can use the Euclidean distance. One can use the Scikit-learn library to implement the K-NN algorithm for this task.

$$Euclidean = \sqrt{\sum_i^k (X_i - Y_i)^2}$$

Step 5: Performance Analysis

Our algorithm is ready, so we must check the model's performance. Even a single missed important message may cause a user to reconsider the value of spam filtering. So we must be sure that our algorithm will be as close to 100% accurate. But some researchers feel that more than accuracy as the evaluation parameter for spam classification is needed.

According to the below table (also known as the confusion matrix), we must evaluate our spam classification model based on four different parameters.

Accuracy : $(TP + TN)/(TP + FP + FN + TN)$

Precision: $TP / (TP + FP)$

Sensitivity: $TP / (TP + FN)$

Specificity: $TN / (TN + FP)$

More advanced algorithms are available for this classification, but you can easily achieve more than 90% accuracy using k-NN-based implementation.

Gmail, Yahoo, and Outlook Case Study

Gmail

Google data centers use thousands of rules to filter spam emails. They provide the weightage to different parameters; based on that, they filter the emails. Google's spam classifier is said to be a state of an art technique that uses various techniques like Optical character recognition, linear regression, and a combination of multiple neural networks.

Yahoo

Yahoo mail is the world's first free webmail service provider, with more than 320 million active users. They have their filtering techniques to categorize emails. Yahoo's basic methods are URL filtering, email content, and user spam complaints. Unlike Gmail, Yahoo filter emails messages by domain and not the IP address. Yahoo also provides users with custom-filtering options to send mail directly to junk folders.

Outlook

Microsoft-owned mailing platform widely used among professionals. In 2013, Microsoft renamed Hotmail and Windows Live Mail to Outlook. At present, outlook has more than 400 Million active users. Outlook has its distinctive feature based on which it filters every incoming mail. Based on their official website, they have provided the list of spam filters they use to send any mail in the junk folder, which includes:

Safe Senders list

Safe Recipients list

Blocked Senders list

Blocked Top-Level Domains list

Blocked Encodings list

Oral Questions

What is Porter Stemmer's Algorithm?

Why did you use the k-NN algorithm for this problem?

Is this supervised learning or unsupervised learning?

What are the different algorithms that can replace k-NN here?

What steps can be taken to improve accuracy further?

Conclusion

In terms of the Number of spam emails sent daily and the Number of money people lose every day because of these spam scams, Spam-filtering becomes the primary need for all email-providing companies. So, in this way we have discussed the complete process of spam email filtering using advanced machine learning technologies. We also have closed one possible way of implementing our spam classifier using one of the most famous algorithms, k-NN. We also discussed the case studies of well-known companies like Gmail, Outlook, and Yahoo to review how they use ML and AI techniques to filter such spammers.