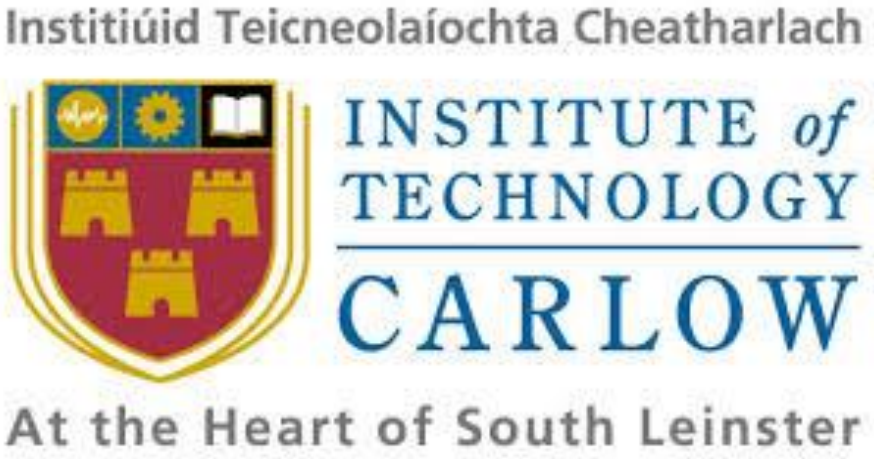

Email Spam Filter using Machine Learning

Functional Specification

Author: Hazel Murphy

Student ID: C00230058

Project Supervisor: James Egan

Date: Friday 27th November 2020

Abstract

The purpose of this document is to describe in detail how this application will operate. The document will include the functionality the application will provide, how the users will interact with the application and how the application will be displayed. The target users of this application are everyday email users and for specific organisations which will be determined on a case-by-case basis. The application will provide these users with an easy-to-use email spam filter classify tool.

Table of Contents

Introduction

To determine if an email is spam or not spam is an extremely important aspect for all email users. For any daily email user their primary purpose is to send and receive emails which are safe and secure to view. Opening emails which are spam can cause major impacts to the user for example it could download malware on their devices, the hacker could gain access to their sensitive information. The email spam filter tool will be a user-friendly system to use. It will display an easy-to-understand graphical user interface (GUI) to any user who logs in. Each user will have access to view and receive emails as often as they wish. Users will not need to worry if an email they have received in their inbox is spam and may contain for example malware. This is because the email spam filter tool will have classified this email as spam and the email will appear in the spam folder. The tool will classify the emails by using a supervised machine learning algorithm such as Naive Bayes.

Document scope

This Functional Specification document is a document which provides detailed information about how the email spam filter classification tool will function. This document clearly outlines the actors that will be interacting with the system and in what way they are expected to interact with the system, what they should expect from the system and what they can and cannot do while interacting with the system.

Project scope

This document focuses on all the scopes of the project, it will consider every aspect of the project. It will range from the web application that works as an email client, to the backend that classifies and stores emails as spam, to setting up the local email server to mimic an actual server setup.

Project overview

System functions

The system is a flask web application, when the user login to their account the application will display the user's emails. The email's will be scored using classification algorithms such as Naive Bayes, SVM, Random Forest and Logistic Regression. The outcome of the algorithm will determine whether the email is spam or not spam. If the email is spam it will be displayed in the spam folder for the user otherwise the email will be in their inbox folder. The administrator will be the only user who will has access to spam report and be able to view all the registered users for the web application.

Web application Functions

- System needs to be able to store mails from the mail server in the SQL database
- Spam filter needs to score emails stored in SQL database
- User needs to be able to register to the application
- Admin/User needs to be able to login to the application
- Admin/User needs to be able to view emails in their sent folder
- Admin/User needs to be able to view emails in their inbox
- Admin/User needs to be able to see spam in spam folder
- Admin/User needs to be able to logout
- Administrator needs to be able to view the spam report page
- Administrator needs to be able to view all registered users of the application

System actors

There are two actors in this system. The number of actors may change if additional features are added in the future. The actors communicating with the system are the administrator and a general user.

Administrator

The administrator will have access to the system where they can view their mails. They can view their sent, received, spam mails and their account information. They also the only user

who have access to view the spam reports and view all the registered users of the web application.

User

The user is the person who will be interacting with the system to access their emails. They will interact with the system by registering. Once they registered to the application, they will enter their email address and password to login. Once the login is successful the user will be able to view their emails (inbox, sent, spam), receive emails, view their account information and logout of the application.

System components

Hardware

- Windows machine hardware
    - Acting as mail server running local software
    - Acting as web server running local software

Software

- Mail server (hMailServer)
- Flask application

User groups

The primary user group of this system would ideally be for individual businesses and everyday email users. It would be implemented on a case-by-case basis. This system aims to provide a tool for an organisation to catch any incoming spam emails. This will increase the organization's security and less attacks will occur.

System requirements

Use case diagram

A Use Case diagram shows the actors and the relationships between the actors and use cases of the system. The use-case diagram might be modified slightly later while developing the system. The users will be able to register, login, receive emails, view sent, inbox and spam emails, and logout. There will be one administration account. The administration account will be used when required to manage, edit and maintain the application. The administration account will be the only account who has access to view the spam reports and all the registered users of the application.

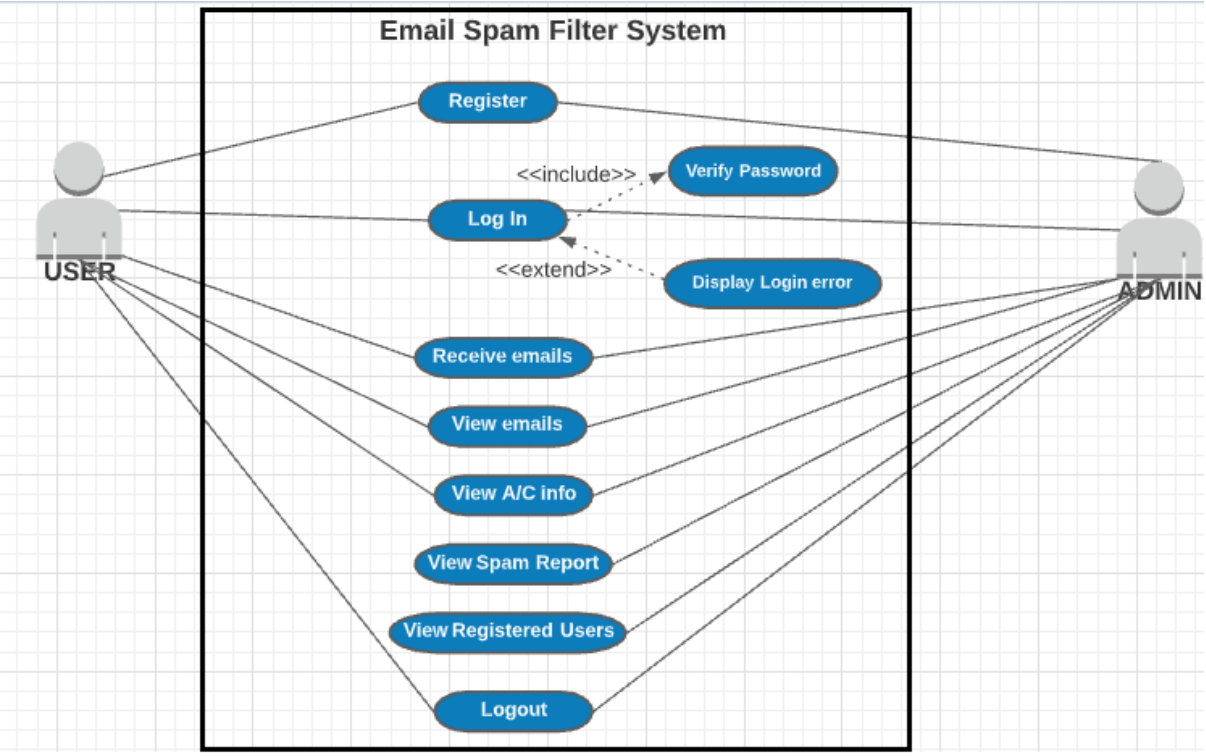

Figure 1: Use-case diagram of the system

Brief use cases

1. **Register**

Primary Actors: User, Admin

Precondition: The user has successfully registered to the web application.

Steps:

- User clicks on register button on the desktop application
- User enters their username, email address and password twice and clicks register

Expected Result: The user has registered to the system successfully

2. **Login**

Primary Actors: User, Admin

Precondition: The user has successfully registered with the system.

Steps:

- User visits the website
- User enters their email address and password on the login screen
- User clicks Login
- If credentials match the database for this user a successfully login will occur
- The user will have access to their account

Expected Result: The user successfully logs into the application and is presented with the welcome screen.

3. **Receive Emails**

Primary Actors: User, Admin

Precondition: The recipient is successfully logged into the system. An email is successfully sent to the recipient.

Steps:

1. The user will login to the system
2. The user will navigate to their inbox folder to view their email
3. The users most recent email received will appear at the top of the list on the inbox screen

Expected Result: The email is successfully received and is visible to the user in the correct folder.

4. **View Emails**

Primary Actors: User, Admin

Precondition: The user is successfully logged into the system.

Steps:

1. The user will login to the system
2. The user will navigate to the specific folder (sent, inbox or spam folder).

Expected Result: An email which seems suspicious is displayed in the spam folder and has a spam score of 1 in the database. An email received with spam score of 0 will be displayed in the inbox folder. All the users sent mails will be displayed in the sent folder.

5. **View Account Information**

Primary Actors: User, Admin

Precondition: The user is successfully logged into the system.

Steps:

1. The user is successfully logged into the system
2. The user visits the account page.
3. The users username and email address they used to register for the application will display.

Expected Result: Their account information displays.

6. **View Spam report**

Primary Actors: Admin

Precondition: The administrator has successfully logged in to the system.

Steps:

1.The administrator is successfully logged into the system

2.The administrator visits the report page

Expected Result: The spam report is displayed to the Administrator

7. **View Register report**

Primary Actors: Admin

Precondition: The administrator has successfully logged in to the system.

Steps:

1. The administrator is successfully logged into the system
2. The administrator visits the register page

Expected Result: All registered users are displayed to the Administrator

8. **Logout**

Primary Actors: User, Admin

Precondition: The user wants to logout of the system.

Steps:

1. The user clicks the “Logout” button
2. The user will be redirected to the homepage of the application.

Expected Result: The user successfully logs out of the system and the session is destroyed. They should not be able to access emails until they log back into the application.

External tools/libraries and systems

Software

- Mail server (hMailServer)

The hMailServer is being used as the local mail server, mails sent and received by the mail server are stored locally. The data being stored in the mail server is scraped into the SQL database which is processed using a machine learning model.

- Flask application

Flask is a web application framework which will be used to create a web application to house the functions of my system. The flask application will be an easy-to-use application. It will contain one administrator account. This account will have access to more pages than a general user. The flask app will be querying the SQL database to display the emails, generate spam reports and display all the registered users of the application.

Supplementary Specification

FURPS+

FURPS is a technique to prioritise the requirements needed by the users. FRUPS acronym stands for Functionality, Usability, Reliability, Performance, Supportability. The FURPS+ model will be used to define the supplementary specification for the system. The plus symbol is used to specify constraints around the design, implementation, interface and physical constraints.

Functionality

This element represents the main product features.
In this system the following functionality will be required:

- The system must allow users to receive and view emails.
- The system is responsible for not allowing any unsafe email through to the users. It must classify these emails as spam or not spam and display the email in the appropriate folder within the application.
- The system should allow admins to receive, view emails, view the spam reports and view all the registered users of the application.

Usability

Usability includes the requirements needed for the user to interact with the system. It should describe how easy a user can interact with the system

- The flask web application should be user-friendly and clear.
- Users must be able to register with the web application and login.
- The flask web application should be user-friendly for the admins to allow them to view the spam reports clearly and all the registered users.
- The system should work with 90% of web services.

Reliability

This metric indicates the possibility of the system failure.
Reliability includes different elements like availability, recoverability, and accuracy.
This system will be created to be as robust as possible.

- The design and layout of the system will allow for a quick and efficient recovery time if an availability issue occurs.
- The admin user should have access to the web application at any time to and view the reports.
- The system must be secure.

Performance

Performance relates to how the system is performing based on the data being put through the system, the system response time, start-up time, and recovery time.

- The web application should load for the user in under twenty seconds
- The web application should be efficient and operate at good quality speed.

Supportability

This section allows for other requirements needed such as testability, maintainability, adaptability, compatibility, configurability, installability, localizability, etc.

- The web application should be accessible from all browsers and device such as chrome, Firefox, internet explorer using an iPhone, laptop, mobile phone.
- In the future to add more features the system should be designed to be able to add these easily. e.g. add an additional spam filter classification method.
- A usage guide must be available if requested.

+ (FURPS+)

The plus symbol includes any extra elements which are not covered in the previous headings of FURPS.

Security

- The content of the emails will be stored in a SQL database.
- The user's passwords credentials which are stored in a database must be encrypted. The passwords must be hashed and salted using a suitable and secure hashing algorithm.

- The web application will be activated once the user has entered their credentials successfully. If the credentials are incorrect, a generic message is displayed - “Login unsuccessful. Please check your email and password”.
- The spam report can only be viewed by logging in as the administrator no other user should have access to the reports.

Metrics

Criteria	**Description**
Spam detection	In the process of selecting and evaluating the machine learning mode the use of confusion matrix to compare the results of the models against a dataset where we already predict the answer.
Spam detection	In the process of selecting and evaluating the machine learning model we are seeking a low level of Type I error (false positive), to stop important emails being classified as spam.
Spam detection	In the process of selecting and evaluating the machine learning model we are seeking a low level of Type II error (false negatives), to stop spam being classified as a normal mail.
Spam detection	Ensure the Recall (True positive rate) of the machine learning model is the correct proportion of actual positives was identified correctly.
Spam detection	Ensure the Precision of the machine learning model is sufficient.
Security	The web application is not vulnerable to SQL injection attacks and uses an object relational mapper.
Security	The web application is resistant to brute force attacks
Security	All user information is stored using a hashing key to encrypt user information.
Errors	Low level of application errors.
Usability	Users are prompted appropriately at all stages of user input
Usability	The system is easy to use and intuitive in design
Usability	The system makes use of labels for accessibility
Reliability	The system works and does not throw exceptions during standard usage

Figure 2: Metrics

(Top 10 model evaluation metrics for classification ML models, 2020)

Precedent for this application?

Similar Products

Existing solutions currently available on the market include the following:

Gmail

Gmail was developed by Google as is a free to use email client. Through POP and IMAP protocols users can sync email content to their Gmail accounts. Users can access their Gmail accounts via the web. Gmail scans emails automatically to filter spam and malware. The spam filtering elements within Gmail features a community-driven system. If a user marks an email as spam, the system retrieves information, and this allows the system to identify similar future messages like the mail marked for all other users. (Gmail, 2020)

Microsoft Outlook

Microsoft Outlook was developed by Microsoft. It is a free to use email client. The Junk Email Filter feature within outlook doesn't stop delivery of junk email messages, however it does move any emails which are suspected to be spam into the Junk e-mail folder. A user can change the Junk Email Filter settings in the Junk E-mail Options dialog box. (Overview of the Junk Email Filter, 2020)

Mozilla Thunderbird

Mozilla Thunderbird was developed by Mozilla Foundation. It is a free to use and open-source cross-platform email client. The spam filter feature is enabled by default. You can determine what happens to the mail which is marked as junk/spam by setting the system-wide preferences element. (Thunderbird and Junk / Spam Messages | Thunderbird Help, 2020)

How they differ?

The spam classification tool is trained on a business-by-business basis to suit the needs of the organisation. This differs from publicly available email clients. It also differs from the likes of mimecast which only provide a spam classification tool and not an email client interface. The three solutions mentioned above are much larger than the system being developed. They

are more developed and focused on large organisations. This is because they offer more advanced features.

Project plan

The delivery date for the final product is the 30th of April 2021. The below table is my project plan I have created and tend to follow to successfully complete this project on time.

Sprint #	Plan	Due Date	Deliverable
0	Complete Necessary Research	13/11/2020	Research Manual
1	Spam classification machine learning model	20/11/2020	Discovery for spam classification machine learning model
2	Complete Necessary Research on how the project will function	27/11/2020	Functional Specification
3	Implement Naïve Bayes and SVM model	4/12/2020	Build python function for Naïve Bayes and SVM model
4	Complete Necessary Research on the design of the project	11/12/2020	Design Manual
5	Implement Random Forest and Logistic Regression model	16/12/2020	Build python function for Random Forest and Logistic model
6	Presentation 1	18/12/2020	Present my project to my supervisor and receive feedback
7	Implement Word embedding	22/12/2020	Build python function for Word embedding
BREAK	**Christmas Holiday**		
8	Plan tests	08/01/2021	Plan tests to create confusion matrix, type I error, type II error, precision and recall to determine which is best to use

9	Presentation 2	15/01/2021	Present my project to my supervisor and receive feedback
10	Research flask application	20/01/2021	Discovery for flask application
11	Design flask application	05/02/2021	A front end for the website that is easy to use and looks good
12	Build initial flask application	19/02/2021	Implement the web app with all the required features such as secure login page
13	Research hMailServer	23/02/2021	Discovery for hMailServer setup
14	Implement hMailServer	05/03/2021	Build the hMailServer
15	Train models	06/03/2021	Get datasets and train the 4 models, getting all scores for analysis
16	Implement cron-job	12/03/2021	Embed the python function as periodic cron-job into flask application
17	Implement models into cron-job	13/03/2021	Embed the machine learning models into the cron-job
18	Test the development of all functionality in application	26/03/2021	Iterate on Flask application until all functionality in application
19	Test spam classification	31/03/2021	Test all spam classification features
20	Test spam report	02/04/2021	Test spam report features
21	Exploratory testing phase	10/04/2021	Test full application, report any flaws found
22	Review	15/04/2021	Complete any final tasks, review final project

23	Technical Manual	19/04/2021	Complete the technical manual and submit to the supervisor
24	Final Report	19/04/2021	Complete the final report for the project and submit to my supervisor
25	Final Product	23/04/2021	Final product completed successfully
26	Research Poster or Usability	23/04/2021	Complete the research poster for the project and submit to my supervisor
27	Project Demo	30/04/2021	Demo the final product to the lectures in college

Figure 3: Project plan

(Spam Filtering System With Deep Learning, 2020)

Functional Requirements

#	Function	Description	Importance	Tech Issues	Dependencies
1	Registration	Allows new users to setup an account on hMailServer	High	N/A	hMailServer
2	Login	Allows user to login via thunderbird	High	Connect their account to thunderbird specific settings need to be applied	Function 1
3	Send Mails	Allow user to send mails via thunderbird **For demo purposes*	High	N/A	Function 2
4	Scrape and classify emails from the hMailServer	Stores the emails in the correct format in the SQL database	High	N/A	hMailServer

		to be classified as spam			
5	Display emails	Display emails to the user on the correct page on the flask app	High	Querying the database	SQL database
6	Logout	Allows the user to logout	High	N/A	The user must be logged into the flask app

Figure 4: Functional Requirements

**Thunderbird is used for demoing purposes. It allows a user to send an email in real-time. I did not design any features within thunderbird.*

Testing

There are multiple functionalities to test:

- Spam classification
- Spam report
- Individual system functionalities e.g. register and login

To test the spam report, a batch of 1000 emails will be sent to each registered users inbox. The emails sent will have documented predicted spam classification values. The spam report will be then compared to the predicted values and any discrepancies mitigated.

Using exploratory testing students will be asked to use the system, to test functionalities such as login, logout, view spam, view inbox etc. Any issues that arise will be solved for and mitigated.

Bibliography

Medium. 2020. *Top 10 Model Evaluation Metrics For Classification ML Models*. [online] Available at: <https://towardsdatascience.com/top-10-model-evaluation-metrics-for-classification-ml-models-a0a0f1d51b9> [Accessed 25 November 2020].

En.wikipedia.org. 2020. *Gmail*. [online] Available at: <https://en.wikipedia.org/wiki/Gmail> [Accessed 25 November 2020].

Support.microsoft.com. 2020. *Overview Of The Junk Email Filter*. [online] Available at: <https://support.microsoft.com/en-us/office/overview-of-the-junk-email-filter-5ae3ea8e-cf41-4fa0-b02a-3b96e21de089> [Accessed 25 November 2020].

Support.mozilla.org. 2020. *Thunderbird And Junk / Spam Messages | Thunderbird Help*. [online] Available at: <https://support.mozilla.org/en-US/kb/thunderbird-and-junk-spam-messages> [Accessed 25 November 2020].

Medium. 2020. *Spam Filtering System With Deep Learning*. [online] Available at: <https://towardsdatascience.com/spam-filtering-system-with-deep-learning-b8070b28f9e0> [Accessed 25 November 2020].