 **SAVEETHA SCHOOL OF ENGINEERING **

**SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES**

**CAPSTONE PROJECT REPORT**

SPAM CLASSIFICATION

**COURSE CODE & NAME:**

CSA1357 - THEORY OF COMPUTATION WITH PUSH DOWN AUTOMATA

**SUBMITTED BY**

* Krishna Subramanian -192211042
* Aaditya parthiban-192211037
* .

**SUBMITTED TO** Dr. K. V. Kanimozhi

**ABSTRACT**

The Spam Classification System (SCS) is a C-based application designed to identify and filter spam emails efficiently. By implementing a spam classification algorithm in C, the system classifies incoming emails as spam or non-spam with a focus on simplicity and performance. The application provides a command-line interface for managing email classifications, training data, and evaluating the classification accuracy. This system leverages basic machine learning techniques to ensure effective spam detection and improve email security.

**INTRODUCTION**

The Spam Classification System (SCS) is developed to address the challenge of spam emails using the C programming language. The system utilizes simple machine learning algorithms to categorize emails into spam and non-spam classes. The application is designed for performance and simplicity, making it suitable for environments where resources are constrained.

The system includes a command-line interface for interacting with the application, allowing users to input email content, train the classification model with sample data, and evaluate the performance of the classifier. By focusing on the core principles of spam detection, the SCS provides a straightforward solution for managing spam emails.

**PROBLEM STATEMENT**

The primary goal of the Spam Classification System is to enhance email security by accurately classifying incoming emails as spam or non-spam. The system should handle different types of spam effectively while minimizing misclassification. Key functionalities include:

- Email Classification: Classify emails based on content using a simple classification algorithm.

- Model Training: Train the classification model with sample email data to improve accuracy.

-Performance Evaluation: Assess the accuracy of the classification model using performance metrics.

The system is implemented in C to leverage its efficiency and low-level access to resources.

**OBJECTIVES**

- Accurate Classification: Implement a classification algorithm to distinguish between spam and non-spam emails with high accuracy.

- Simple Interface: Provide a command-line interface for users to input emails, train the model, and view results.

- Model Training: Develop a method for training the classification model using sample data.

- Performance Monitoring: Evaluate the model's performance using metrics such as accuracy and error rate.

**EXPECTED OUTCOMES**

The Spam Classification System is expected to:

- Achieve Accurate Classification: Implement an effective classification algorithm with high accuracy.

- Provide a Functional Interface: Offer a command-line interface for easy interaction with the system.

- Enable Model Training: Allow users to train the classification model with sample data.

- Evaluate Performance: Provide tools for assessing the model's accuracy and effectiveness.

**MATERIALS AND METHODS**

Materials:

- Development Environment: GCC compiler or any standard C development environment.

- Programming Language: C (Standard C90 or higher).

- Libraries: Standard C libraries for file handling and string manipulation.

- Testing Tools: Basic testing utilities and performance metrics for evaluating classification accuracy.

Methods:

1. Requirement Analysis:

- Define functional and non-functional requirements for spam classification.

- Gather requirements for classification accuracy, model training, and user interaction.

2. System Design:

- Design the architecture using flowcharts to outline the email classification process and model training.

- Define data structures for storing email content and classification results.

3. Development:

- Command-Line Interface: Implement a CLI for inputting email content, training the model, and viewing classification results.

- Classification Algorithm: Develop a simple classification algorithm using keyword matching or basic statistical techniques.

- Model Training: Implement methods for training the model with sample email data.

4. Testing:

- Conduct unit testing to ensure each component functions correctly.

- Perform integration testing to verify the interaction between components.

- Evaluate classification accuracy using sample email datasets and performance metrics.

5.Deployment:

- Deploy the application in a suitable environment for testing and user interaction.

- Ensure proper handling of email content and classification results.

6. Documentation and Maintenance:

- Provide user and developer documentation for using and maintaining the application.

- Implement a feedback system for continuous improvement and address any post-deployment issues.

**PROGRAM**

#include <stdio.h>

#include <string.h>

// Function prototypes

void classifyEmail(const char \*emailContent);

void trainModel();

void evaluateModel();

int main() {

int choice;

char emailContent[1024];

while (1) {

printf("\nSpam Classification System\n");

printf("1. Classify Email\n");

printf("2. Train Model\n");

printf("3. Evaluate Model\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

getchar(); // Consume newline character

switch (choice) {

case 1:

printf("Enter email content: ");

fgets(emailContent, sizeof(emailContent), stdin);

classifyEmail(emailContent);

break;

case 2:

trainModel();

break;

case 3:

evaluateModel();

break;

case 4:

return 0;

default:

printf("Invalid choice. Please try again.\n");

}

}

}

// Function to classify email content

void classifyEmail(const char \*emailContent) {

// Placeholder for classification logic

// Implement simple classification algorithm here

printf("Classifying email...\n");

printf("Result: Spam\n"); // or "Not Spam"

}

// Function to train the classification model

void trainModel() {

// Placeholder for model training logic

// Implement model training with sample data

printf("Training model...\n");

}

// Function to evaluate the classification model

void evaluateModel() {

// Placeholder for model evaluation logic

// Implement evaluation using performance metrics

printf("Evaluating model...\n");

}

­­­­

**OUTPUT**

The system allows users to classify emails, train the model, and evaluate its performance through a command-line interface. The email classification results are displayed based on the implemented algorithm.

**CONCLUSION**

The Spam Classification System developed in C offers a practical solution for managing spam emails. By implementing a basic classification algorithm and providing a command-line interface, the application ensures effective spam detection and email security. The system’s simplicity and performance make it suitable for various environments, while its accuracy and usability provide valuable tools for managing spam effectively.

**REFERENCES**

- Mitchell, Tom M. "Machine Learning." McGraw-Hill Education, 1997. This book provides foundational knowledge on machine learning algorithms and their applications.

- Russell, Stuart J., and Peter Norvig. "Artificial Intelligence: A Modern Approach." Prentice Hall, 2020. This text covers various AI techniques, including those used in spam classification.

- Witten, Ian H., et al. "Data Mining: Practical Machine Learning Tools and Techniques." Morgan Kaufmann, 2016. This book provides insights into practical machine learning techniques used in classification tasks.