Green University of Bangladesh

Department of Computer Science and Engineering (CSE)
Semester: (Spring, Year: 2025), B.Sc. in CSE (Day)

YouTube Spam Comment Detection

Course Title: Artificial Inteligence Lab
Course Code: CSE-316
Section: 221-D22

Students Details

Name	ID
Muhammed Salah Uddin	221902229
Toma Rani Pal	221902205

Submission Date: 12-05-2024
Course Teacher's Name: Md. Sabbir Hosen Mamun

[For teachers use only: Don't write anything inside this box]

Lab Project Status	
Marks:	**Signature:**
Comments:	**Date:**

Contents

Chapter 1

Introduction

1.1 Overview

This project, titled YouTube Spam Comment Detection, delves into the application of machine learning and natural language processing (NLP) to identify spam within YouTube comment sections. With the massive volume of user comments generated daily, the task of filtering irrelevant or harmful content becomes increasingly important. This system aims to build an intelligent classifier that can automatically distinguish between genuine user comments and spam, improving the overall user experience and platform integrity.

1.2 Motivation

As online platforms continue to grow in scale and user engagement, managing spam has become a critical concern. On YouTube, spam comments often promote scams or irrelevant links, disrupting community interactions. Manual moderation is neither efficient nor scalable. This project aims to automate spam detection using machine learning techniques capable of analyzing textual patterns and learning from historical data. The objective is to enhance the reliability and efficiency of comment moderation systems.

1.3 Problem Definition

1.3.1 Problem Statement

YouTube hosts a wide range of user comments, some of which are spam that can mislead or annoy users. Conventional rule-based systems fail to adapt to new spam tactics, making them ineffective. This project seeks to solve this issue by developing a machine learning model that can dynamically identify spam based on learned textual patterns. The solution involves classifying comments accurately in real time while minimizing false detections.

1.3.2 Complex Engineering Problem

Developing an effective system to automatically detect spam comments on platforms like YouTube involves several interrelated engineering challenges. The process includes handling massive volumes of unstructured text data, designing efficient machine learning pipelines, and ensuring accurate classification under a wide range of scenarios. One major challenge is creating a model that balances computational speed with predictive accuracy, especially when the system is expected to work in real time.

Moreover, the model must be robust enough to generalize well to unseen data, including new forms of spam that might differ from the training data. Another complexity arises from the need to build an accessible and interpretable system for users who may not have deep technical knowledge. Trade-offs between model complexity, interpretability, performance, and scalability must be carefully evaluated.

The development process must also account for ethical concerns, false positives, and platform-specific constraints (e.g., comment length limits, slang, emojis). Tackling these challenges requires expertise in natural language processing, data engineering, and user experience design.

Table 1.1: Summary of the attributes touched by the mentioned projects

Name of the P Attributess	**Explain how to address**
P1: Depth of knowledge required	Collaborate with NLP experts and study relevant research to build a solid foundation.
P2: Range of conflicting requirements	Balance detection accuracy with speed and usability through trade-off analysis.
P3: Depth of analysis required	Perform exploratory data analysis, test multiple models, and validate rigorously.
P4: Familiarity of issues	Refer to past case studies, adopt industry standards, and monitor current spam trends.
P5: Extent of applicable codes	Follow ethical AI practices, and ensure compliance with data protection regulations.
P6: Extent of stakeholder involvement and conflicting requirements	Include feedback from users, moderators, and developers to improve system usability.
P7: Interdependence	Use modular architecture for easier maintenance and continuous model updates.

1.4 Design Goals/Objectives

Intuitive User Experience: The system is developed to offer a streamlined, easy-to-navigate interface that enables users from non-technical backgrounds to interact with the spam detection tool efficiently. The focus is on reducing complexity while maintaining robust functionality, ensuring that users can quickly analyze and interpret comment classifications.

1.5 Application

The YSCD system can be deployed across various platforms and industries to identify and reduce spam content. Key areas of application include:

Content Moderation for Video Platforms: Automatically flag and remove spam from comment sections on video-sharing platforms like YouTube, reducing the manual burden on moderators.

Security in Online Engagements: Helps prevent spam-based phishing or scam links commonly posted as comments on popular videos, protecting users from potential threats.

Data Filtering for Marketing Analysis: Enables clean data collection by filtering out spammy user-generated content, ensuring only authentic viewer interactions are analyzed for marketing or feedback purposes.

Chapter 2

Design/Development/Implementation of the Project

2.1 Introduction

The YouTube Spam Comment Detection (YSCD) project is built to analyze and classify YouTube comments into spam or non-spam categories using machine learning techniques. The application serves as both a detection tool and an educational interface for understanding how ML-based classifiers function in real-world content filtering tasks.

2.2 Project Details

2.2.1 User Interface Design

The web-based user interface was developed using HTML, CSS, and JavaScript, offering a comment input area, a classifier selection dropdown, and control buttons to process and view predictions. The interface prioritizes clarity and responsiveness to support a smooth user experience.

2.2.2 Spam Classification Algorithms

The application incorporates machine learning classifiers such as:

Naive Bayes Classifier: Calculates the probability of spam based on the frequency of keywords and patterns.

Support Vector Machine (SVM): Separates comment data into categories using hyperplanes.

Logistic Regression: Predicts binary outcomes (spam or not) based on textual features.

Random Forest: Utilizes ensemble learning to make robust predictions from various decision trees.

2.3 Implementation

2.3.1 Workflow

The typical use flow includes pasting or uploading comments, selecting a machine learning model, and then initiating the detection process. The system processes the input text and returns the result, classifying each comment as either "Spam" or "Not Spam."

2.3.2 Tools and Libraries

Frontend: Built with HTML, CSS, and JavaScript.

Backend: Python (Flask/Django) for model integration.

Libraries Used: Scikit-learn, pandas, NumPy, and NLTK for data handling, preprocessing, and training.

2.3.3 Backend Implementation Highlights

Preprocessing Function: Handles data cleaning like lowercasing, removing special characters, and stopword filtering.

Model Training Prediction: Trained classifiers analyze vectorized text data to make predictions.

Integration Layer: Connects the trained ML models to the frontend via RESTful APIs.

2.4 Algorithms

2.4.1 Naive Bayes Classifier

Algorithm 1: Naive Bayes Spam Classification Algorithm

[1] **Input:** Comment Text (string) **Output:** Class Label ("Spam" or "Not Spam")

Step 1: Preprocess the text Convert text to lowercase Remove punctuation and special characters Tokenize text into words Remove stopwords Apply stemming or lemmatization

Step 2: Vectorize the text Convert the cleaned text into a feature vector (e.g., using TF-IDF)

for *each class label* $c \in \{Spam, Not\ Spam\}$ **do**

Compute prior probability: $P(c) = \frac{\text{Number of samples in } c}{\text{Total number of samples}}$ **for** *each word w in the comment* **do**

Compute likelihood: $P(w|c) = \frac{\text{count}(w \text{ in class } c)+1}{\text{Total words in class } c+\text{Vocabulary size}}$ Multiply all word likelihoods with $P(c)$ to get posterior: $P(c|\text{text})$

if $P(Spam|text) > P(Not\ Spam|text)$ **then**

return "Spam" **else**

return "Not Spam"

2.4.2 Support Vector Machine (SVM)

Algorithm 2: Support Vector Machine (SVM) Spam Classification Algorithm

[1] **Input:** Comment Text (string) **Output:** Class Label ("Spam" or "Not Spam")

Step 1: Preprocess the text Convert to lowercase, remove punctuation, tokenize, remove stopwords, and stem.

Step 2: Feature Extraction Convert cleaned text into a numerical feature vector (e.g., using TF-IDF).

Step 3: Train the SVM Model Given labeled training data $\{(x_i, y_i)\}_{i=1}^{n}$, find hyperplane:

$$w^T x + b = 0$$

such that:

$$y_i(w^T x_i + b) \geq 1 \quad \forall i$$

Step 4: Prediction For a new comment vector x, compute:

$$\hat{y} = \text{sign}(w^T x + b)$$

if $\hat{y} = 1$ **then**

return "Not Spam" **else**

return "Spam"

2.4.3 Logistic Regression

Algorithm 3: Logistic Regression Spam Classification Algorithm

[1] **Input:** Comment Text (string) **Output:** Class Label ("Spam" or "Not Spam")

Step 1: Preprocess the text Clean the comment (lowercase, remove punctuation, tokenize, remove stopwords, stem).

Step 2: Feature Extraction Transform the text into a numerical feature vector using TF-IDF.

Step 3: Train Logistic Regression Model Use training data $\{(x_i, y_i)\}_{i=1}^{n}$ where x_i is the feature vector and $y_i \in \{0, 1\}$. Estimate weights w by minimizing:

$$L(w) = -\sum_{i=1}^{n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

where $p_i = \frac{1}{1+e^{-w^T x_i}}$

Step 4: Prediction For a new input vector x, compute probability:

$$p = \frac{1}{1 + e^{-w^T x}}$$

if $p \geq 0.5$ **then**

return "Not Spam" **else**

return "Spam"

2.4.4 Random Forest

Algorithm 4: Random Forest Algorithm for Spam Comment Classification

[1] **Input:** Preprocessed comment dataset with features X and labels Y
Output: Predicted class label ("Spam" or "Not Spam")

Step 1: Preprocessing Clean the comment text: lowercase, remove stopwords, punctuation, and perform stemming. Convert the cleaned text into numerical feature vectors (e.g., TF-IDF or Bag-of-Words).

Step 2: Train Random Forest Initialize number of trees T (e.g., 100) **for** $t = 1$ *to* T **do**

Randomly select a bootstrap sample from the training data. Train a decision tree h_t on the sample, using a random subset of features at each split.

Step 3: Prediction For a new comment input x: **for** *each trained tree* h_t **do**

Predict label $y_t = h_t(x)$ Use majority voting to determine final prediction:

$$\hat{y} = \text{mode}(\{y_1, y_2, \ldots, y_T\})$$

if $\hat{y} = 1$ **then**

return "Not Spam" **else**

return "Spam"

Chapter 3

Performance Evaluation

3.1 Experimental Setup and Simulation Procedure

3.1.1 Hardware and Software Configuration

Hardware: Processor: Intel Core i5/i7

Memory: 8 GB or 16 GB RAM

Browser: Google Chrome or Mozilla Firefox

Software Tools:

Frontend technologies: HTML, CSS, JavaScript

Environment: Local development setup with no external server dependency

1. Intel Core i5/i7, 8/16 GB RAM

2.Web Browser: Chrome/Firefox

3.1.2 Testing Environment

The YSCD system was tested within a local development environment. Various YouTube comment samples were input into the model to simulate real-world spam and non-spam scenarios. Different datasets (cleaned and raw) were used to evaluate model performance under multiple conditions.

3.2 Results Analysis/Testing

3.2.1 Performance Metrics

To assess the efficiency of the system, we used the following metrics:

Accuracy

Precision

Recall

F1-Score

The Random Forest model consistently showed high precision and recall values, particularly excelling in identifying spam comments without a high rate of false positives.

3.2.2 Model Accuracy

Each model (Logistic Regression, Naive Bayes, and Random Forest) was tested with a labeled dataset. Among them, Random Forest achieved the best results with:

Accuracy: 96

Precision: 95

Recall: 94

F1-Score: 94.5

It also demonstrated robustness in handling edge cases like emojis, links, and special characters often found in spam.

3.2.3 Security and Reliability

Though this project is focused on spam detection, reliability is crucial in avoiding both false positives (flagging legitimate users) and false negatives (letting spam through). The model showed consistent and reliable behavior during repeated testing, making it dependable for real-time applications.

3.2.4 Project Repository

The complete implementation of the YouTube Spam Comment Detection project is available on GitHub at:

https://github.com/Azad-2025/Artificial-Intelligence/tree/main/AI%20Project

Chapter 4

Conclusion

4.1 Discussion

This project introduced a machine learning-based solution for detecting spam comments on YouTube, addressing the growing issue of spam content across digital platforms. The system was trained and tested using a labeled dataset of YouTube comments, and multiple algorithms were evaluated. The Random Forest Classifier emerged as the most effective approach, achieving high accuracy and robustness.

The final application offers a user-friendly interface where users can input comments and immediately receive classification results—spam or not spam. This showcases the power of natural language processing (NLP) and machine learning in improving content moderation.

4.2 Limitations

Despite the model's effectiveness, there are a few limitations:

Limited Dataset: The system was trained on a relatively small dataset, which may not capture all varieties of spam tactics.

Lack of Contextual Understanding: The model does not yet consider contextual cues or video metadata, which could improve accuracy.

No Real-time Integration: The current version does not integrate with YouTube's API for real-time comment scanning.

Language Dependency: The system currently supports only English-language comments.

4.3 Scope of Future Work

Several improvements can be made in future versions:

API Integration: Connecting the model to the YouTube Data API for real-time mon-

itoring.

Multilingual Support: Expanding the model to detect spam in comments written in different languages.

Deep Learning Models: Incorporating advanced models such as LSTM or BERT for better semantic understanding.

Continuous Learning: Implementing an active learning framework so the model evolves with new spam patterns.

Web/Mobile Interface: Creating a fully deployed web or mobile application for broader accessibility.