

SPAM MAIL DETECTION USING GATED RECURRENT NEURAL NETWORK

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Technology Degree in Information Technology

By

SAMRITHA A.R (Reg.No. - 39120101)

SIVAPRIYA P (Reg.No. - 39120114)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade “A” by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

MAY 2023



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC
(Established under Section 3 of UGC Act, 1956)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI
CHENNAI- 600119
www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Final Year Project Report is the bonafide work of **SAMRITHA A.R (Reg.No. - 39120101)** and **SIVAPRIYA P (Reg.No. - 39120114)** who have done the project work as an individual who carried out the project entitled "**Spam Mail Detection Using Gated Recurrent Neural Network**" under my supervision from Jan 2023 to May 2023.

26/4/2023
Dr. G. MATHIVANAN, M.E., Ph.D.,

R. Subhashini
Head of the Department
Dr. R. SUBHASHINI, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

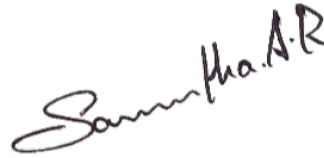
External Examiner

DECLARATION

I, **SAMRITHA A.R (Reg.No. 39120101)**, hereby declare that the Project Report on **“SPAM MAIL DETECTION USING GATED RECURRENT NEURAL NETWORK”** done by us under the guidance of **Dr. G. MATHIVANAN** at Sathyabama Institute of Science and technology is submitted in partial fulfillment of the requirements of the award of Bachelor of Technology degree in **Information Technology**.

DATE: 11.05.2023

PLACE: Chennai

A handwritten signature in black ink, reading "Samritha A.R.", written diagonally across the signature line.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA, M.E., Ph.D., Dean**, School of Computing, **Dr. R. SUBHASHINI, M.E., Ph.D., Head of the Department of Information Technology** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. G. MATHIVANAN, M.E., Ph.D.**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

High numbers of spam emails have led to an increase in email triage, causing losses amounting to USD 355 million per year. One way to reduce this loss is to classify spam email into categories including fraud or promotions made by unwanted parties. The initial development of spam email classification was based on simple methods such as word filters. Now, more complex methods have emerged such as sentence modeling using machine learning. Some of the most well-known methods for dealing with the problem of text classification are networks with Gated Recurrent Units (GRU). This study focuses on the classification of spam emails, so GRU methods were used. The results of this study show that, under the scenario without dropout, GRU obtained the great accuracy value.

The significant amounts of SPAM emails that are derived from various botnets worldwide affect the limited capacity of mailboxes. They affect the security of personal mail and the space-loss from the communication. They affect the time required for identifying spam emails and addressing them. Till today, the email spam detection is still considered a challenging process. That is because the email spam is still happening a lot. It is because the detection still needs much improvement. Therefore, the researcher of this study develops Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) Spam email detection. The developed approach got tested by employing the Spambase dataset. The approach shows an accuracy of 98.7%. Through conducting extensive experiments, the researcher concludes that the proposed approach shows an excellent capability of detecting spam email.

TABLE OF CONTENTS			
CHAPTER	TITLE		PAGE NO
	ABSTRACT		v
	LIST OF FIGURES		viii
	LIST OF TABLES		ix
1	INTRODUCTION		1
2	LITERATURE SURVEY		2
	2.1	Inferences From Literature Survey	2
	2.2	Open Problems In Existing System	5
3	REQUIREMENTS ANALYSIS		6
	3.1	Risk Analysis Of The Project	6
	3.2	Software Requirements Specification Document	8
	3.3	System Use Case	9
4	DESCRIPTION OF PROPOSED SYSTEM		10
	4.1	Methodology	10
		4.1.1 Dataset Preparation	10
		4.1.2 Data Preprocessing	11
		4.1.3 Data Training Or Modeling	12
		4.1.4 Data Testing	14
		4.1.5 Data Evaluation	15
	4.2	Architecture Design Of Proposed System	16
	4.3	Project Management Plan	19
	4.4	Software To Operations Plan	20
5	IMPLEMENTATION DETAILS		22
	5.1	Development And Deployment Setup	22
	5.2	Algorithm	23
	5.3	Testing	24

6	RESULTS AND DISCUSSION		27
7	CONCLUSION		28
	7.1	Conclusion And Future Work	28
	7.2	Research Issues	28
	7.3	Implementation Issues	29
	REFERENCES		31
	APPENDIX		33
	A	SOURCE CODE	33
	B	SNAP SHOTS	41
	C	RESEARCH PAPER	46

LIST OF FIGURES		
FIGURE NO	FIGURE NAME	PAGE NO
1.1	Spam Classification	1
4.1.1.1	Data Preparation Steps	10
4.1.2.1	Steps for Data Preprocessing	11
4.1.3.1	Key Steps for Data Modeling	13
4.1.3.2	Training Data Needs	14
4.1.4.1	Train Test Validation Split	15
4.1.5.1	Evaluation Process	16
4.2.1	System Architecture	17
4.2.2	Class Diagram	17
4.2.3	Data Flow Diagram	18
4.2.4	Flow Diagram	18
4.2.5	ER Diagram	19
5.2.1	GRU Model	23
5.2.1	GRU Diagram	24

LIST OF TABLES		
TABLE NO	TABLE NAME	PAGE NO
3.2.1	Supported Platforms	8

CHAPTER 1

INTRODUCTION

Email or electronic mail spam refers to the “using of email to send unsolicited emails or advertising emails to a group of recipients. Unsolicited emails mean the recipient has not granted permission for receiving those emails. “The popularity of using spam emails is increasing since last decade. Spam has become a big misfortune on the internet. Spam is a waste of storage, time and message speed. Automatic email filtering may be the most effective method of detecting spam but nowadays spammers can easily bypass all these spam filtering applications easily. Several years ago, most of the spam can be blocked manually coming from certain email addresses. Machine learning approach will be used for spam detection. Major approaches adopted closer to junk mail filtering encompass “text analysis, white and blacklists of domain names, and community-primarily based techniques”. Text assessment of contents of mails is an extensively used method to the spams. Many answers deployable on server and purchaser aspects are available. Regularly clients and organizations would not need any legitimate messages to be lost. The boycott approach has been probably the soonest technique pursued for the separating of spams. The technique is to acknowledge all the sends other than those from the area/electronic mail ids. With more up to date areas coming into the classification of spamming space names this technique keeps an eye on no longer work so well. The white list approach is the approach of accepting the mails from the domain names/addresses openly whitelisted and place others in a much less importance queue, that is delivered most effectively after the sender responds to an affirmation request sent through the “junk mail filtering system”

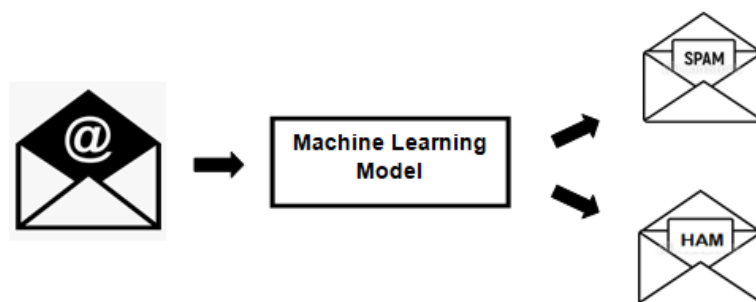


Fig.1.1 Spam Classification

CHAPTER 2

LITERATURE SURVEY

2.1 INFERENCES FROM LITERATURE SURVEY

- **TITLE:** Email Classification using LSTM: A Deep Learning Technique

AUTHOR: Palwasha Bhatti; ZuneraJalil; AsmaMajeed

ABSTRACT: Electronic mail has been in use for decades and more than four billion users access their emails using different domains and servers. Emails are considered an official way of communication in remote working modes and in online businesses. Email labeling can reduce the amount of effort to manage this communication. Email classification is so far done to classify emails such as Spam, Non-spam, Junk, social media, etc. However, email classification keeping in view the types of cybercrimes committed through email is not done. Emails can be labeled as Spam, Phishing, fraudulent, harassing, bullying, or can be a general/normal email. This identification is one of the most challenging tasks for both email service providers and consumers. Several spam identification models have previously been proposed and tested but very limited work has been done so far on the multi-class classification of emails. Emails can be classified into more than two classes (spam and ham). In this paper, we have proposed a solution to classify emails into four classes: fraudulent, suspicious, harassment, and normal. A deep learning approach named Long Short Term Memory (LSTM) with stratified sampling has been used to identify the email classes. An effort has also been made to balance the input dataset using over-sampling methods. The proposed model obtained a classification accuracy of more than 90%. With stratified sampling only and more than 95% by applying data balancing techniques on the dataset.

- **TITLE:** Spam Detection using Word Embedding-based LSTM

AUTHOR:

**SupphawarichThanarattananakin; SuwannaBulao; BusarinVisitsilp; M
aleeratMaliyaem**

ABSTRACT: Spam email is a well-known problem in network communications. A large number of spam emails in mailbox caused inconvenience and might lead to cybercrime. Effective spam mail detection is one of challenges research topics. This paper proposed a new spam mail detection approach that can be applied for sentiment analysis. A dataset of 5,572 messages was tested using Word embedding Techniques such as Bag of words, Hashing, and Long short-term memory network Algorithm (LSTM). The results showed the proposed method performances are equal 98%, 93%, 95%, and 98% interns of precision, recall, F1-score, and accuracy respectively.

- **TITLE:** Spam Email Detection with Affect Intensities using Recurrent Neural Network Algorithm

AUTHOR: NurafifahAlyaFarahisya; Fitra A. Bachtiar

ABSTRACT: A large number of email users triggers an increase in the occurrence of spam in emails to gain benefits for some parties but harm others and also email users. Spam emails usually contain advertisements or criminal acts such as phishing which implicitly contain human emotions in them. It is quite difficult and takes time to differentiate between a large number of spam and ham emails. This problem can be overcome by using deep learning technology. One of which is a neural network that can classify spam emails. This paper uses the spam and ham Enron email corpus dataset. This study will add emotional features in extracting its features. The steps taken include text preprocessing, feature extraction using tf-idf, and lexicon-based emotion features, followed by classification using RNN to detect spam in emails. A comparison with other methods is also provided by comparing the proposed method to Naïve Bayes and Support-Vector Machine (SVM) algorithm based on precision and accuracy. In addition, this study also compares the effect of using affect intensities on the performance of algorithms. The results show that RNN outperforms other methods by showing the highest accuracy 99% and the precision of 99.1%. Adding effect intensities to the model would increase the model recognition results.

- **TITLE:** Malicious Mail Filtering and Tracing System Based on KNN and Improved LSTM Algorithm

AUTHOR: Da Xiao; Meiyi Jiang

ABSTRACT: Spam and phishing emails are very troublesome problems for mailbox users. Many enterprises, departments and individuals are harmed by them. Moreover, the senders of these malicious emails are in a hidden position and occupy an initiative position. The existing mailbox services can only filter and shield some malicious mails, which is difficult to reverse the disadvantage of users. To solve these problems, we propose a secure mail system using k-nearest neighbor (KNN) algorithm and improved long short-term memory (LSTM) algorithm (Bi-LSTM-Attention algorithm). KNN classifier can effectively distinguish normal emails, spam and phishing emails, and has a high accuracy. Bi-LSTM-Attention classifier classifies phishing emails according to the similarity of the malicious mail text from the same attacker to some extent. By classifying and identifying the source of malicious emails, we can grasp the characteristics of the attacker, provide materials for further research, and improve the passive status of users. Experiments show that the classification results of attack sources reach 90%, which indicate the value of further research and promotion.

- **TITLE:** Analysis of Optimized Machine Learning and Deep Learning Techniques for Spam Detection

AUTHOR: Fahima Hossain; Mohammed Nasir Uddin; Rajib Kumar Halder

ABSTRACT: Spam and non-spam email identification are one of the most challenging tasks for both email service providers and consumers. The spammers try to spread misleading facts through irritating messages by attracting user's attention. Several spam identification-models have previously been proposed and tested but the recorded accuracy has shown that further work in this direction is needed to achieve improved accuracy, low training time, and less error rate. In this research work, we have proposed a model that classifies the e-mail into spam and ham. DBSCAN and Isolation Forest are used to identify the extreme values outside of the specific range. Heat map, Recursive Feature Elimination, and Chi-Square feature selection techniques are used to select the effective features. The

proposed model is implemented in both machine learning and deep learning to establish a comparative analysis. Multinomial Naïve Bayes (MNB), Random Forest (RF), K-Nearest Neighbor (KNN), Gradient Boosting (GB) are used to introduce ensemble method in machine learning implementation. Recurrent Neural Network (RNN), Gradient Descent (GD), Artificial Neural Network (ANN) for deep learning implementation. An ensemble method is constructed to combine multiple classifiers' output.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM

The existent methods for detecting email spam are not very accurate. Hence, raising the performance of those methods is needed. Several data mining methods are deemed effective in raising the performance of the classification. Such methods include the Decision Tree (DT). However, DT includes a weakness. It is represented in being over-sensitive to the noise instance or data, training set, and irrelevant attribute. This weakness reduces the performance of the DT [33]. Some instances are contradictory and noisy existing within the training set.

Disadvantage:

- It is not very accurate
- Some instances are contradictory and noisy
- That may lead to the suffering of the created decision tree from over fitting and reducing its accuracy.

CHAPTER 3

REQUIREMENTS ANALYSIS

3.1 RISK ANALYSIS OF THE PROJECT

To perform a risk analysis of a project, you need to identify potential risks, assess their likelihood of occurrence and impact, and determine appropriate mitigation strategies. Here are some steps to follow:

a) Identify potential risks:

- **Inaccurate spam detection:** There is a risk that the GRU model may not accurately detect spam emails, which could result in legitimate emails being flagged as spam or vice versa.
- **Lack of training data:** There may not be enough relevant training data available to train the GRU model effectively, which could lead to lower accuracy in spam detection.
- **Technical limitations:** There is a risk that technical limitations, such as hardware or software issues, could impact the performance of the GRU model or the overall project timeline.
- **Privacy concerns:** There is a risk that the use of email content for spam detection could raise privacy concerns among users, which could impact adoption of the service.

b) Assess likelihood and impact:

- **Inaccurate spam detection:** High likelihood and high impact. This risk could result in serious consequences, such as important emails being missed or users losing trust in the service.
- **Lack of training data:** Medium likelihood and medium impact. This risk could impact the accuracy of the model, but could be mitigated by obtaining additional training data.

- **Technical limitations:** Low likelihood and medium impact. This risk could impact the timeline of the project, but could be mitigated with contingency plans.
- **Privacy concerns:** Medium likelihood and medium impact. This risk could impact user adoption of the service, but could be mitigated by implementing strong privacy policies and communicating clearly with users.

c) Prioritize risks:

Based on the likelihood and impact assessment, inaccurate spam detection should be the highest priority risk to address, followed by privacy concerns and lack of training data. Technical limitations could be a lower priority risk.

d) Develop mitigation strategies:

- **Inaccurate spam detection:** Implement a robust testing and validation process to ensure high accuracy of the GRU model. Use multiple methods of spam detection and employ human review for borderline cases.
- **Lack of training data:** Seek out additional training data sources, such as open-source datasets or crowd-sourced labeling. Augment the training data with synthetic data generation techniques.
- **Technical limitations:** Plan for potential hardware or software failures by using redundant systems or cloud-based solutions. Regularly test the system to identify potential issues early.
- **Privacy concerns:** Implement strong privacy policies and communicate them clearly to users. Use techniques such as differential privacy to protect user data.

e) Monitor and manage risks:

Throughout the project, regularly reassess the identified risks and monitor their potential impact on the project. Take appropriate actions to address any emerging risks and communicate changes to stakeholders as needed.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

- Operating System: Linux, Windows/7/10
- Server: Anaconda, Jupyter, VS Code
- Front End: tkinter |GUI toolkit
- Server side Script: Python , AIML

Requirement	Minimum	Recommended
RAM	4 GB of free RAM	8 GB of total system RAM
CPU	Any modern CPU	Multi-core CPU. PyCharm supports multithreading for different operations and processes making it faster the more CPU cores it can use.
Disk space	2.5 GB and another 1 GB for caches	SSD drive with at least 5 GB of free space
Monitor resolution	1024x768	1920×1080
Operating system	Officially released 64-bit versions of the following: Microsoft Windows 8 or later macOS 10.13 or later Any Linux distribution that supports Gnome, KDE, or Unity DE. PyCharm is not available for some Linux distributions, such as RHEL6 or CentOS6, that do not include GLIBC 2.14 or later. Pre-release versions are not supported.	Latest 64-bit version of Windows, macOS, or Linux (for example, Debian, Ubuntu, or RHEL)

Table.3.2.1 Supported Platforms

3.3 SYSTEM USE CASE

Use Case Name: Spam Mail Detection

Actors: User, Email Server, GRU Model

Pre-conditions:

- The user has access to an email account hosted on the Email Server.
- The Email Server is configured to use the GRU Model for spam detection.

Basic Flow:

- The User sends an email to the Email Server.
- The Email Server receives the email and submits it to the GRU Model for spam detection.
- The GRU Model analyzes the email content and returns a spam score.
- If the spam score exceeds a pre-defined threshold, the Email Server flags the email as spam and takes appropriate action (such as moving it to a spam folder or deleting it).
- If the spam score is below the threshold, the Email Server delivers the email to the User's inbox.

Alternate Flows:

- If the GRU Model is unable to analyze the email content (e.g. if the email is encrypted), the Email Server may deliver the email to the User's inbox without spam detection.
- If the Email Server encounters technical issues (e.g. if the GRU Model is unavailable), the email may be delivered to the User's inbox without spam detection.

Post-conditions:

- The Email Server has either flagged the email as spam or delivered it to the User's inbox based on the output of the GRU Model.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 METHODOLOGY

To perform this machine learning model, there are five primary modules.

4.1.1 DATASET PREPARATION:

The process of gathering and analyzing accurate data from various sources to find answers to research problems, trends and probabilities, etc., to evaluate possible outcomes is known as Data Collection.

The raw data collected is obtained from the online resource kaggle, which will be used to train machine learning models. In At the end of this process, we obtained a total of some emails, listed in two columns. 1e first column, labeled as 'type' having the two possible values as spam or ham, was meant to be used to classify emails. 1e second column is labeled 'e-mails Text' and contained a variety of e-mail content. It was decided that up to 80% of the emails will be used to train the models, whereas the remaining 20% will be used to test models individually.



Fig.4.1.1.1 Data Preparation Steps

4.1.2 DATA PREPROCESSING:

Data preprocessing, a component of data preparation, describes any type of processing performed on raw data to prepare it for another data processing procedure. It has traditionally been an important preliminary step for training machine learning models and AI models and for running inferences to ensure accurate results.

In this process, Natural Language Process (NLP) is utilized for pre-processing natural language data. NLP is the process to make computer understanding the natural language as same as a human understands. It has many techniques to preprocess data into a format that a computer can be understood. In this work, we transform the text data into sequential data using NLP techniques in order to use it for developing classification models using the GRU algorithms. We also use word tokenization, padding data, truncating data and word embedding techniques for pre-processing data.

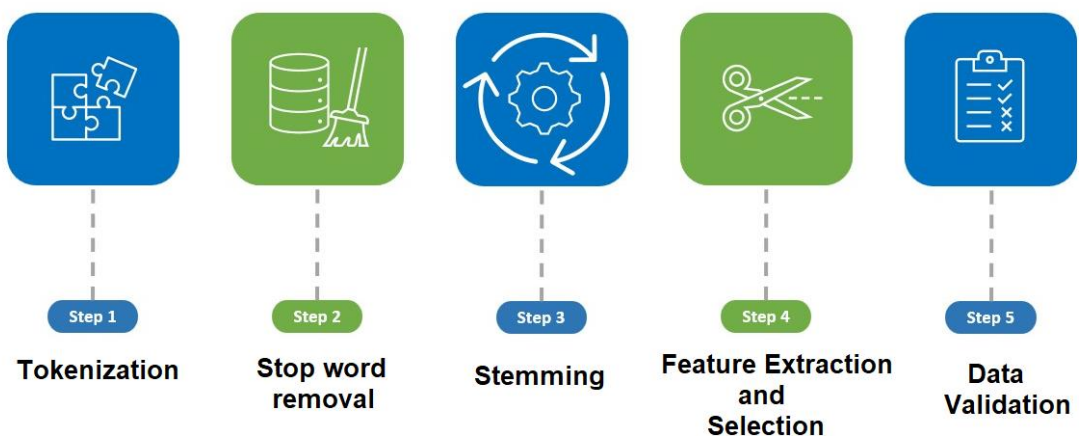


Fig.4.1.2.1 Steps for Data Preprocessing

The details of each technique that we use in the data pre-processing process are described as follows.

- **Tokenization:** Word tokenization is the process that changes words in a sentence into index values represented by a number. In this process, we set number of interesting vocabulary words to create word tokenizer. After

create tokenizer, we use the word tokenizer to convert words in a sentence into sequence data. The tokenizer changes word into index and set index to 0 for unknown words. In addition, we create a tokenizer by set number of vocabulary words to 10,000 words. We also use the tokenizer to convert text data into sequence of index number of words from the tokenizer.

- **Stop Word Removal:** Once the dataset has been transformed into unique tokens, the next step is to delete every unnecessary word with no significance, e.g., white spaces, commas, full stops, colons, semicolons, and punctuation marks. Stop words elimination is the name given to the method of eliminating unnecessary words. Python has built-in library known as natural language toolkit (NLTK), which has been widely used in language processing. Here, we used NLTK toolkit for stop words removal process for elimination of unnecessary words and spaces.
- **Stemming:** After the tokens have been created, the next step is to stem them. Stemming is the method of converting the dataset's derived terms back to their original forms. First, the base terms are stripped of prefixes and suffixes. Next, either modified or misspelled words are changed into their base or stem words using the stemming algorithm. For this step as well, we used NLTK python library to perform a perfect stemming process. After stemming of the content emails, spam words can be easily identified.
- **Feature Extraction and Selection:** Feature extraction is the process of converting a large raw dataset into a more manageable format. Any variable, attribute, or class can be extracted from the dataset during this step, depending on the original dataset.
- **Data Validation:** It indicates that before using, importing, or otherwise processing data, you should verify the accuracy and quality of the source data.

4.1.3 DATA TRAINING or MODELING:

Data modeling is the process of creating a visual representation of either a whole information system or parts of it to communicate connections between data points and structures. The goal is to illustrate the types of data used and stored

within the system, the relationships among these data types, the ways the data can be grouped and organized and its formats and attributes.

Data can be modeled at various levels of abstraction. The process begins by collecting information about business requirements from stakeholders and end users. These business rules are then translated into data structures to formulate a concrete database design. A data model can be compared to a roadmap, an architect's blueprint or any formal diagram that facilitates a deeper understanding of what is being designed.

Data modeling employs standardized schemas and formal techniques. This provides a common, consistent, and predictable way of defining and managing data resources across an organization, or even beyond. Ideally, data models are living documents that evolve along with changing business needs. They play an important role in supporting business processes and planning IT architecture and strategy. Data models can be shared with vendors, partners, and/or industry peers.

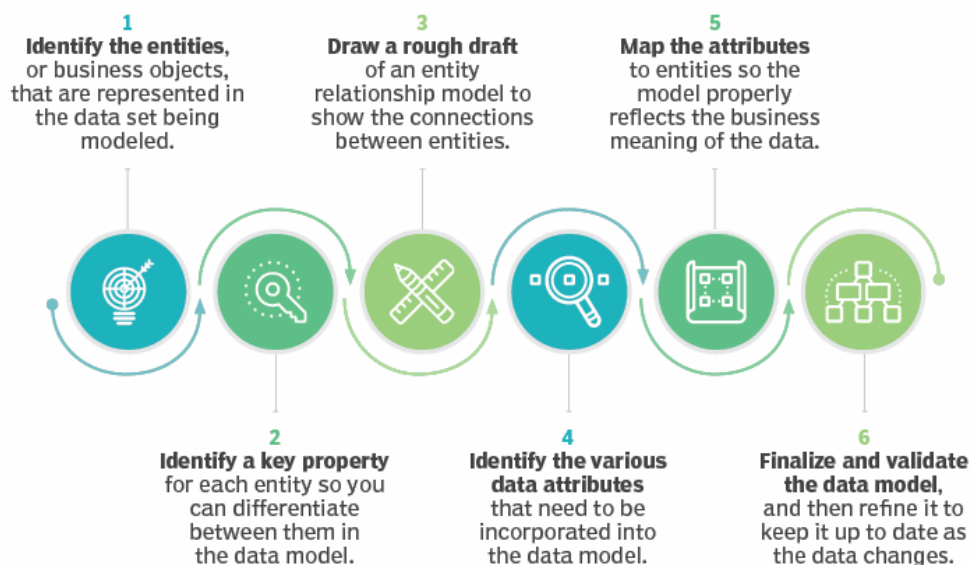


Fig.4.1.3.1 Key Steps for Data Modeling

In this experiment, we develop the spam classification models based on the two deep learning algorithms including GRU algorithms. The GRU is a type of deep learning algorithm that is improved from the LSTM algorithm to reduce the complexity of the structure of the algorithm by using update gate and reset gate.

The update gate is used to control amount of the hidden state to be forwarded to the next state. The reset gate is used to define the significance of the previous hidden state information.

For our model development, we set hypermeters and structure of the networks. In addition, we add a dropout layer after the unit layer in order to prevent the models from the over-fitting issue. Then, we use fully-connected with Rectifier function (ReLU) as an activation function to down-sampling from the unit layer. In the Classifying layer, we use fully-connected with Sigmoid function as an activation function to predict whether it is a spam or a normal message. In the training process, we choose Adam optimizer as an optimizer, binary cross-entropy as loss function because it is a binary classification. The learning rate is 0.01, and the model learning cycle is set to 10 epochs

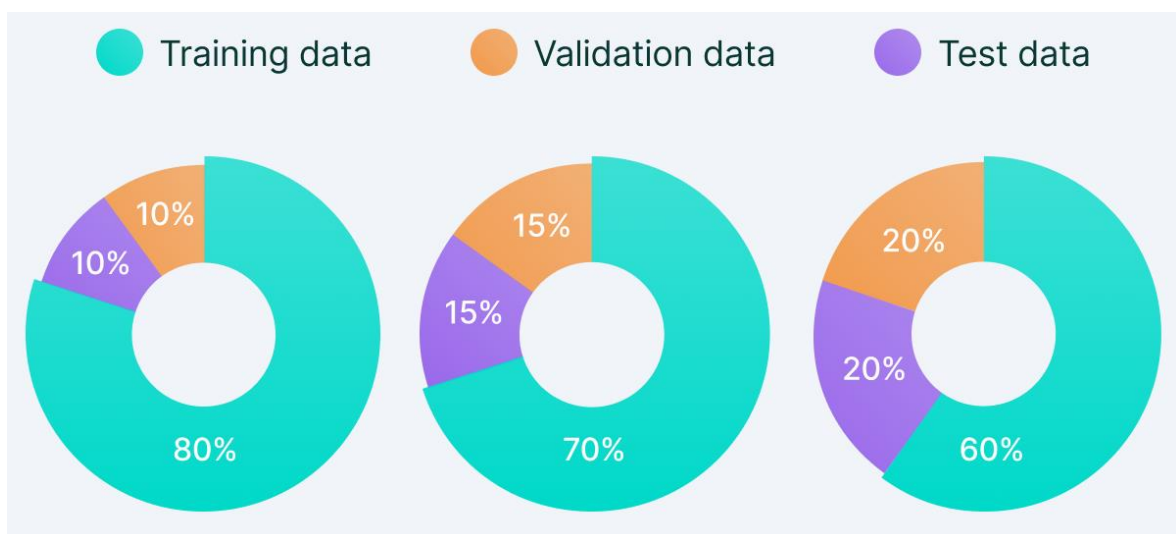


Fig.4.1.3.2 Training Data Needs

4.1.4 DATA TESTING:

Once your machine learning model is built (with your training data), you need unseen data to test your model. This data is called testing data, and you can use it to evaluate the performance and progress of your algorithms' training and adjust or optimize it for improved results.

We will be using keras open-source neural-network library for generating models. The keras only takes real numbers as input; hence we need to vectorise data (converting words into numbers). In order to do so we first tokenize the data, We tokenize text review given for movies using 5000 most frequently occurred words. The tokenized words are then vectorised, following example demonstrate how this is done.

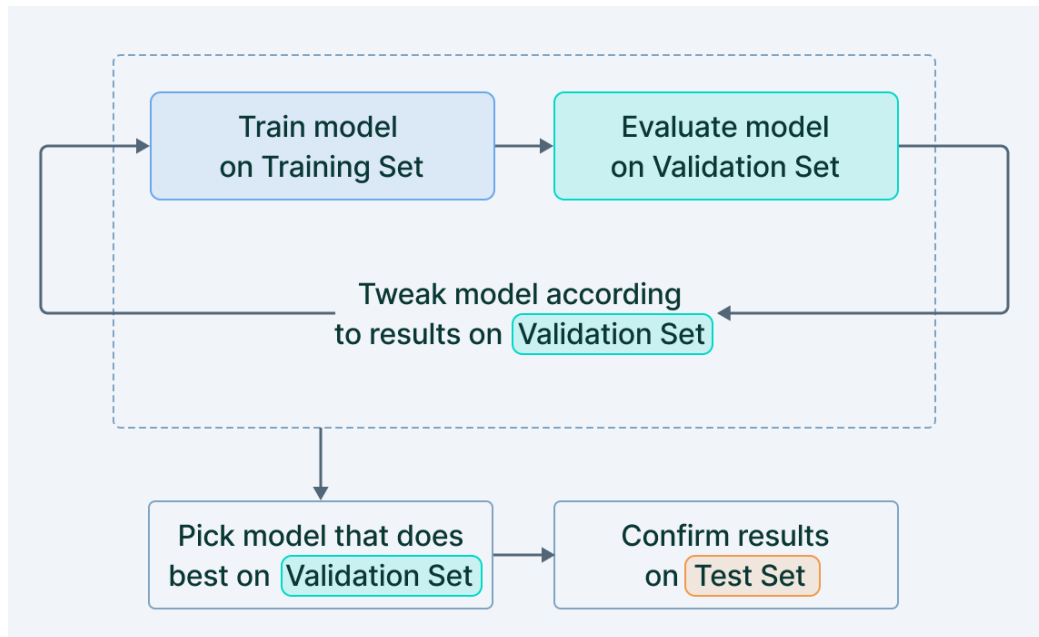


Fig.4.1.4.1 Train Test Validation Split

4.1.5 DATA EVALUATION:

The process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. Evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

After training the models, we evaluate the performance of the model using the testing dataset. The results demonstrate that the accuracy of the model-based LSTM and GRU is similar.

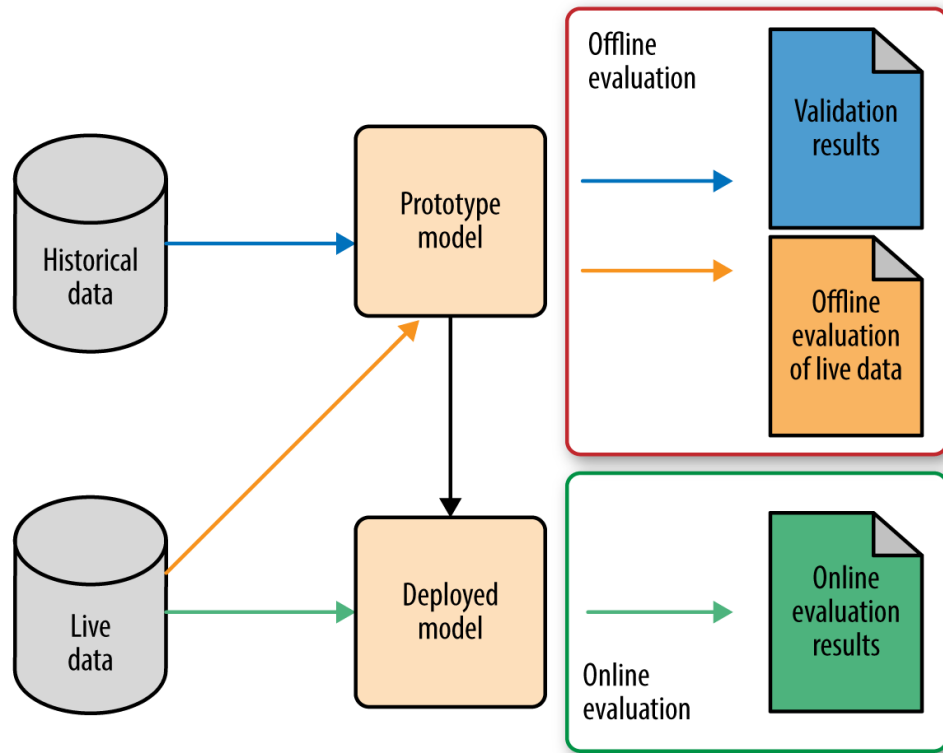


Fig.4.1.5.1 Evaluation Process

4.2 ARCHITECTURE DESIGN OF PROPOSED SYSTEM

System architecture refers to the overall design and organization of a system, including its components, their interactions, and the principles and guidelines that govern their construction and evolution. The purpose of a system architecture is to ensure that the system meets its functional and non-functional requirements, while also being scalable, maintainable, and extensible. It provides a blueprint for the system, defining its components and how they fit together to achieve the desired functionality.

The proposed system follows it's critical to distinguish between features that set a system apart from those of its competitors while developing new products. Baseline functionality is a must for any system to compete in a given product area. Some tactics have significant effects on architecture.

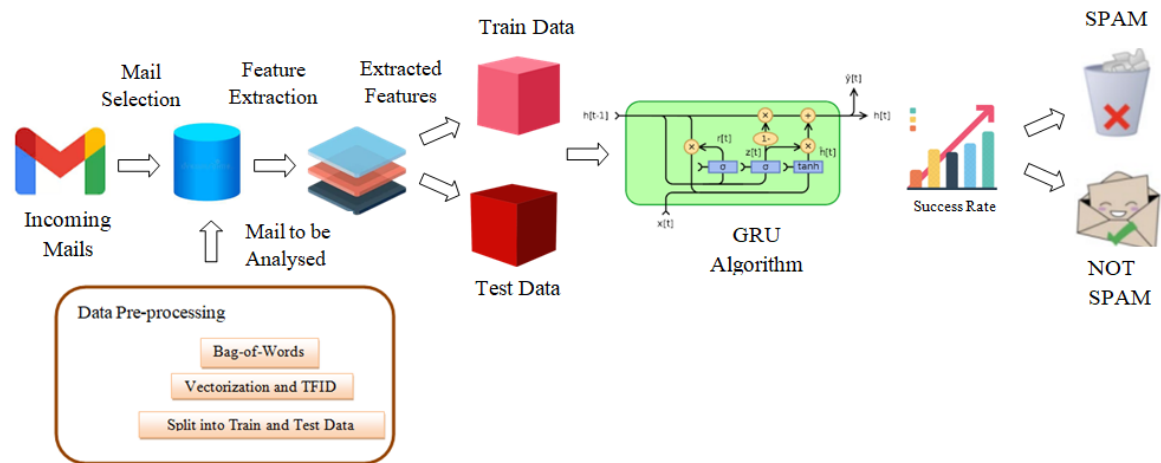


Fig.4.2.1 System Architecture

A class diagram is a type of UML diagram that represents the structure of a system by illustrating its classes, attributes, methods, and relationships between classes. The purpose of a class diagram is to provide a visual representation of the system's architecture, which can be used to guide the design and implementation of the system. Class diagrams are important for several reasons such as, Provides an overview of the system's structure; Guides the implementation process; Improves communication among team members and Facilitates maintenance and updates.

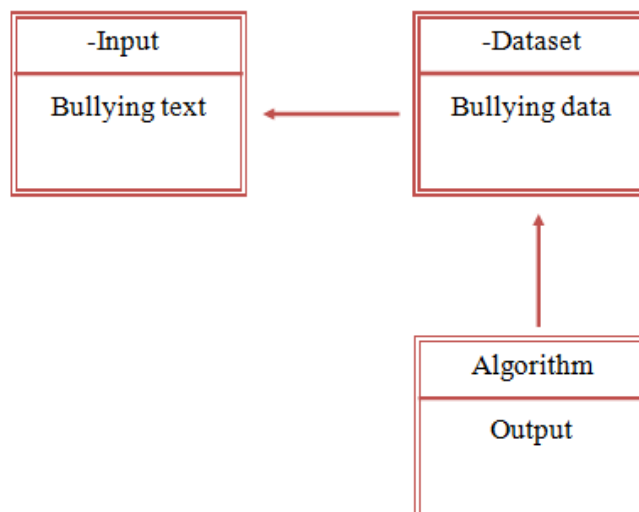


Fig.4.2.2 Class Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data through a system. It is used to describe the process of how data is input, processed, and output in a system or organization. A DFD is typically used to model and understand the flow of information in a system, which can help identify potential issues or bottlenecks in the data flow. DFDs consist of a series of interconnected bubbles or nodes that represent data sources, processing functions, and data outputs. The arrows between the nodes represent the flow of data between them.

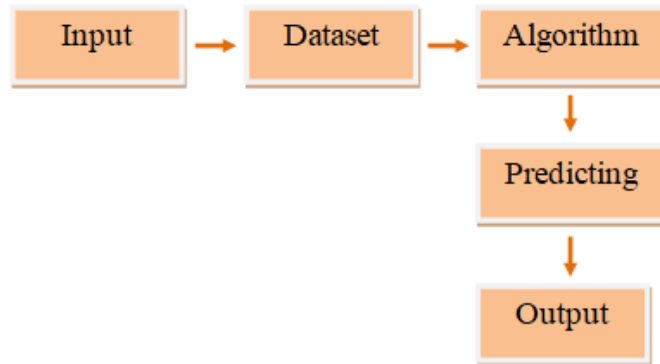


Fig.4.2.3 Data Flow Diagram

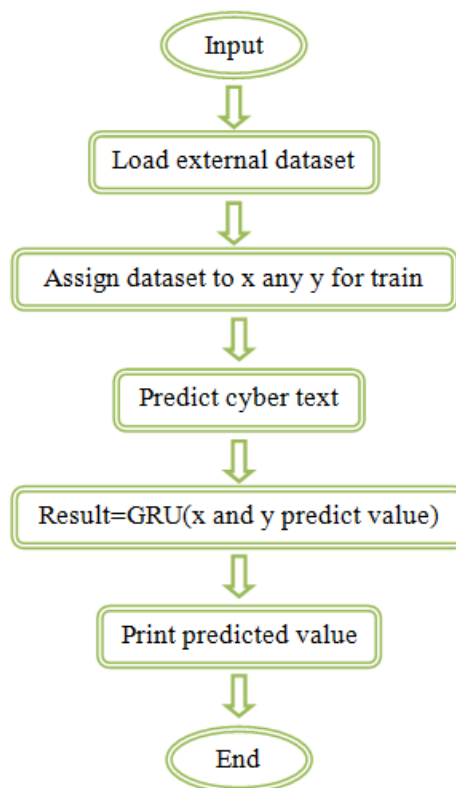


Fig.4.2.4 Flow diagram

An Entity-Relationship (ER) diagram is a graphical representation of the entities and relationships that make up a system. ER diagrams are typically used in database design to model and organize the data in a system.

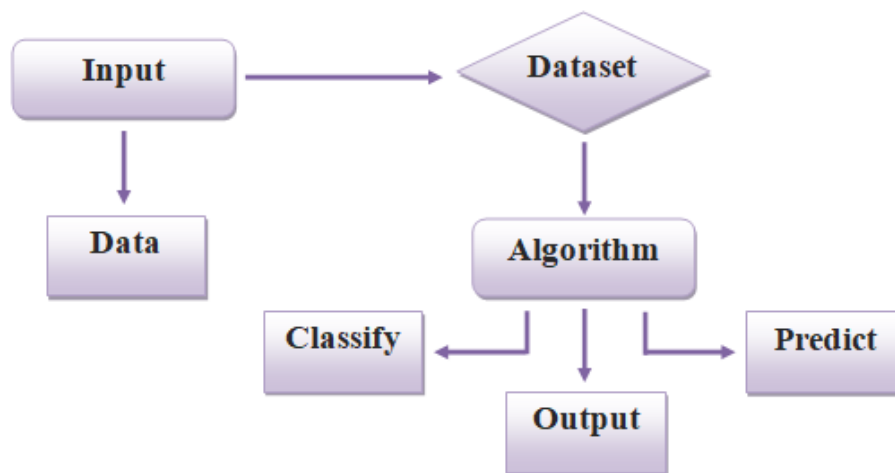


Fig.4.2.5 ER Diagram

4.3 PROJECT MANAGEMENT PLAN

Here is a sample project management plan for the spam mail detection system using GRU:

a) Project Description: The project involves developing a spam mail detection system using GRU-based machine learning algorithms. The goal of the project is to create a system that accurately detects and filters spam emails while minimizing false positives and negatives.

b) Project Scope: The project scope includes the following deliverables:

- **Requirements Specification Document:** A document that outlines the requirements for the spam mail detection system, including functional and non-functional requirements.
- **Design Document:** A document that outlines the system architecture and design, including details on the GRU-based machine learning algorithms.

- **Implementation Plan:** A plan for implementing and testing the spam mail detection system, including the development of the machine learning algorithms and the user interface.
- **User Manual:** A document that outlines how to use the spam mail detection system, including details on the user interface and features.
- **Training and Deployment Plan:** A plan for training the machine learning algorithms and deploying the spam mail detection system in a production environment.

c) Project Timeline: The project timeline will be divided into the following phases:

- Requirements Gathering and Analysis (2 weeks)
- Design and Architecture (2 weeks)
- Implementation and Testing (8 weeks)
- User Manual and Training (1 week)
- Deployment (1 week)

d) Project Communication Plan: The team will establish regular communication channels, including meetings and progress reports, to ensure that all members are aware of project progress, issues, and opportunities for improvement. The team will also establish communication channels with stakeholders, including end users and management, to gather feedback and ensure that the project meets their needs and expectations.

4.4 SOFTWARE TO OPERATIONS PLAN

The Transition or Software to Operations Plan outlines the steps necessary to transition the spam mail detection system using GRU algorithms from development to operational use. Here are the key components of the plan:

a) System Deployment: The system deployment involves installing the software on production servers, configuring the system for the production environment, and ensuring that it is ready for use.

b) User Acceptance Testing (UAT): The UAT phase is conducted to validate the system's functionality and its ability to meet the business requirements. It involves testing the system in a simulated production environment, ensuring that all functionalities are working as intended.

c) End-User Training: The end-users must be trained on how to use the system effectively. This includes providing training on how to use the system interface, understanding the system's output, and taking corrective actions if necessary.

d) Technical Support: Technical support is a critical component of the transition plan. It involves providing support to end-users, including resolving issues and providing maintenance and upgrades as needed.

e) System Monitoring: The system should be continuously monitored to ensure that it is functioning correctly, and any issues are addressed promptly. This includes monitoring system performance, uptime, and other key metrics to ensure optimal performance.

f) System Documentation: All system documentation should be updated to reflect any changes made during the deployment phase. This includes technical documentation, user manuals, and other relevant documents.

By following these steps, the spam mail detection system using GRU algorithms can be successfully transitioned from development to operational use.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

Design and build the trained model in a real world environment so that it can instantly classify incoming emails. Python was used to construct the project's prototype. Python makes it simple to create a working prototype. The project's ultimate implementation would make use of HTML and CSS frameworks. To test our solution and gather feedback from our classmates, we hosted the portal on a local hosting provider's website. Every time we get feedback, our team evaluates it internally before deciding whether to implement it or not based on some variables like usefulness, impact, and practicality. Using Python on the server side and the Anaconda prompt, the following is a step-by-step guide for building a local host website to detect spam and non-spam mail.

With Anaconda, construct a "Spam_email_classifier" environment. The prerequisites for the web application are Flask, Pandas, Numpy, Scikit-Learn, Tensorflow, and Keras. Python script "app.py" was created specifically for the web application. The script should begin by importing the appropriate modules. Create a Flask app, then add two new routes—one to handle the app's functionality, and the other to handle email classification. The program is run by the specified main function. The document "index.html" was used to generate a HTML file. We designed a form with three input fields for the email address, subject and content. Added a send button and a section to the form that shows the outcome of the prediction.

The files "app.py" and "index.html" are indeed kept in the same folder. Activate the "Spam email classifier" environment using Anaconda Prompt. Go to the "app.py" folder. Initiate the server's local host provider operation. To visit the application, go to the URL produced by the anaconda prompt (<http://localhost:0000>). To view the prediction results, enter an email address, a topic, and content in the appropriate fields before clicking the send button.

"Bullying Comment" is displayed if the sent message is spam. The text "#####" will be sent to the recipient. It will display "Non Bullying Comment" if the email is not spam

5.2 ALGORITHM

GRU (Gated Recurrent Unit) is a type of recurrent neural network (RNN) that is widely used for natural language processing (NLP) tasks, including language translation, sentiment analysis, and text classification. The main difference between a standard RNN and a GRU is that GRU uses gating mechanisms to control the flow of information through the network, allowing it to better capture long-term dependencies in sequential data. A GRU cell consists of two gates: an update gate and a reset gate. The update gate determines how much of the previous hidden state to retain, while the reset gate determines how much of the new input to include in the hidden state. Here are the step-by-step algorithm for a GRU cell:

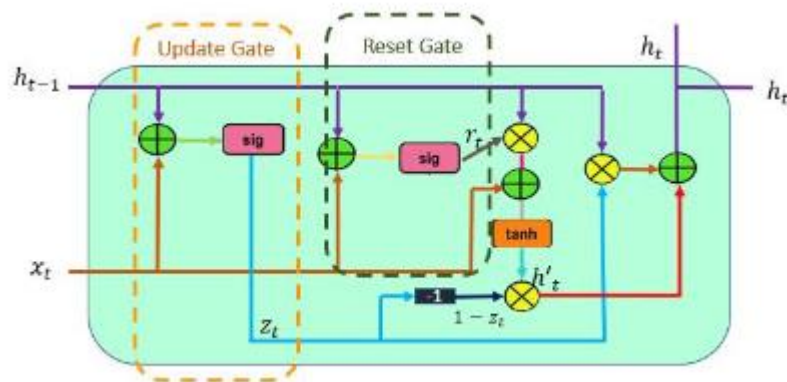


Fig.5.2.1 GRU Model

Input: The GRU cell takes as input the previous hidden state h_{t-1} and the current input x_t .

Reset gate calculation: Compute the reset gate r_t as the sigmoid of the weighted sum of the previous hidden state and the current input: $r_t = \text{sigmoid}(W_r * [h_{t-1}, x_t])$.

Update gate calculation: Compute the update gate z_t as the sigmoid of the weighted sum of the previous hidden state and the current input: $z_t = \text{sigmoid}(W_z * [h_{t-1}, x_t])$.

Candidate hidden state calculation: Compute the candidate hidden state h'_t as the hyperbolic tangent of the weighted sum of the reset gate multiplied by the previous hidden state and the current input: $h'_t = \tanh(W * [r_t * h_{t-1}, x_t])$.

Hidden state calculation: Compute the new hidden state h_t as the weighted sum of the previous hidden state and the candidate hidden state, where the weights are determined by the update gate: $h_t = (1 - z_t) * h_{t-1} + z_t * h'_t$.

Output: The output of the GRU cell is the new hidden state h_t .

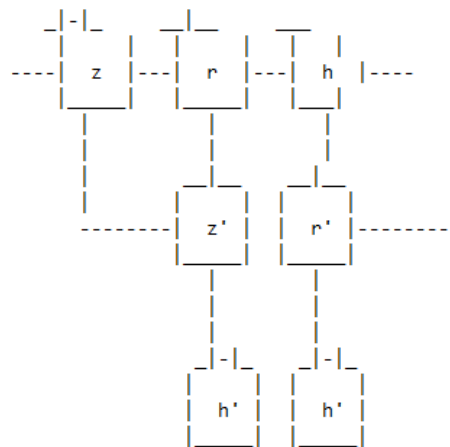


Fig.5.2.2 GRU Diagram

These steps are repeated for each time step in the input sequence, with the output of each GRU cell being fed as input to the next cell in the sequence. By selectively updating the hidden state using the reset and update gates, a GRU can better capture long-term dependencies in sequential data.

5.3 TESTING

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System

testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently. In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. Inadequate testing or non-testing leads to errors that may appear few months later. This will create two problems, Time delay between the cause and appearance of the problem. The effect of the system errors on files and records within the system. The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits.

The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two main common steps, Unit testing and Integration testing. In the project system testing is made as follows:

- **Unit Testing:** Test each component of the AVM system, including the RFID reader, fingerprint scanner, and the software program to ensure they are

functioning properly. Conduct unit testing for each component individually before moving on to integrated testing.

- **Integration Testing:** Test the AVM system as a whole to ensure that all components work together properly. Conduct integration testing by simulating the voting process and testing each component's interaction with the others.
- **Performance Testing:** Test the system's performance to ensure that it can handle the expected load during the election process. Performance testing should include testing for response time, transaction throughput, and system availability.
- **Security Testing:** Test the system's security features to ensure that it is secure from hacking attempts and unauthorized access. Conduct security testing by attempting to break into the system and testing the system's ability to detect and prevent unauthorized access.
- **Usability Testing:** Test the system's usability by simulating the voting process and testing how easily voters can navigate the system. Conduct usability testing with a group of users and gather feedback to improve the system's user experience.
- **End-to-End Testing:** Conduct end-to-end testing to ensure that the system functions correctly from start to finish. This testing includes testing the entire voting process from voter authentication to vote tallying.
- **Acceptance Testing:** Once all testing is complete, conduct acceptance testing to ensure that the AVM system meets the system design specifications and requirements. This testing should include a final review of the system's performance, usability, security, and accuracy.

CHAPTER 6

RESULTS AND DISCUSSION

Our model has been trained using GRU classifiers to check and compare the results for greater accuracy. Each classifier will give its evaluated results to the user. After all the classifiers return its result to the user; then the user can compare it with other results to see whether the data is “spam” or “ham”. Each classifier result will be shown in graphs and tables for better understanding. The dataset is obtained from “Kaggle” website for training. The name of the dataset used is “spam.csv”. To test the trained machine, a different CSV file is developed with unseen data i.e. data which is not used for the training of the machine; named “emails.csv”. After the text edit has been completed, the paper is ready for the template.

Discussion of the results of a spam mail detection system using GRU should include an analysis of these performance metrics, as well as an evaluation of the system's strengths and weaknesses. A well-designed and trained GRU model can achieve high accuracy, precision, recall, and F1 score in detecting spam emails. However, the performance of the system may depend on various factors, such as the quality and size of the dataset, the selection of features, and the specific GRU model architecture used.

In addition, the system may face challenges in handling adversarial attacks, multilingual detection, and real-time detection. Adversarial attacks can lead to misclassification of spam emails, while multilingual detection and real-time detection require additional preprocessing and processing techniques.

Finally, the system may require ongoing maintenance and updates to address emerging spam email trends and changes in the email system architecture. Therefore, a spam mail detection system using GRU should be continuously evaluated and improved to ensure its effectiveness and reliability over time.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION AND FUTURE WORK

With this result, it can be concluded that the gated recurrent neural network gives the best outcome but has limitation due to class-conditional independence which makes the machine to misclassify some tuples. Ensemble methods on the other hand proven to be useful as they using multiple classifiers for class prediction. Nowadays, lots of emails are sent and received and it is difficult as our project is only able to test emails using a limited amount of corpus. Our project, thus spam detection is proficient of filtering mails giving to the content of the email and not according to the domain names or any other criteria. Therefore, at this it is an only limited body of the email. There is a wide possibility of improvement in our project. The subsequent improvements can be done: “Filtering of spams can be done on the basis of the trusted and verified domain names.” “The spam email classification is very significant in categorizing e-mails and to distinct e-mails that are spam or non-spam.” “This method can be used by the big body to differentiate decent mails that are only the emails they wish to obtain.

7.2 RESEARCH ISSUES

- **Data Imbalance:** In many spam mail detection datasets, there are significantly more non-spam emails than spam emails. This can lead to bias in the training data and negatively impact the performance of the GRU model. Research could explore techniques to address this issue, such as data augmentation or sampling strategies.
- **Adversarial Attacks:** Adversarial attacks refer to the intentional manipulation of the email content to bypass the spam detection system. Research could explore the robustness of the GRU model against adversarial attacks and develop techniques to improve its resistance to such attacks.

- **Real-time Detection:** In many cases, spam emails may contain time-sensitive information, such as phishing attacks that require immediate action. Research could explore techniques to enable real-time detection of spam emails using GRU models, such as stream processing or incremental learning.
- **Multilingual Detection:** Many spam emails are sent in languages other than English. Research could explore the effectiveness of GRU models for detecting spam in different languages and develop multilingual models that can handle multiple languages simultaneously.
- **Explainability and Interpretability:** The ability to interpret and explain the decisions made by the GRU model is crucial for gaining user trust and improving model transparency. Research could explore methods to increase the interpretability and explainability of the GRU model for spam detection.
- **Generalization and Transfer Learning:** The effectiveness of GRU models for spam detection may vary depending on the characteristics of the dataset and the email system used. Research could explore methods to improve the generalization of the GRU model and transfer its learning to different email systems or domains.

These research issues highlight some of the challenges and opportunities for improving the effectiveness of GRU models for spam mail detection. Addressing these issues could lead to more accurate and reliable spam detection systems that can better protect users from the negative impacts of spam emails.

7.3 IMPLEMENTATION ISSUES

- **Data Preprocessing:** Data preprocessing is an important step in implementing a GRU-based spam mail detection system. Preprocessing involves cleaning, normalizing, and tokenizing the email data to prepare it for analysis. Preprocessing techniques may vary depending on the specific requirements of the system and the characteristics of the data.

- **Training and Testing:** Training and testing are critical components of implementing a GRU-based spam mail detection system. Proper selection of training and testing data, as well as validation techniques, is important for achieving good accuracy and avoiding overfitting. Training and testing may require significant computational resources, especially if the dataset is large.
- **GRU Model Selection:** Choosing an appropriate GRU model architecture is important for achieving good performance in spam mail detection. Model selection may depend on various factors, such as the size and complexity of the dataset, the desired level of accuracy, and the available computational resources.
- **Threshold Determination:** Determining an appropriate threshold for classifying emails as spam or non-spam is a critical implementation issue. The threshold can significantly impact the accuracy and reliability of the spam detection system. Threshold determination may involve trial and error, experimentation, or statistical analysis.
- **Integration with Email Systems:** Integrating the GRU-based spam mail detection system with email systems can be a complex implementation issue. Integration may require changes to the email system architecture, modifications to email protocols, and integration with existing email filtering systems. It is important to consider the impact of integration on email system performance, scalability, and user experience.
- **Maintenance and Updates:** Once the spam mail detection system is implemented, it requires ongoing maintenance and updates. Maintenance may include monitoring the system performance, updating the GRU model, retraining the model with new data, and fine-tuning system parameters. Updates may be necessary to address emerging spam email trends, new email protocols, or changes in the email system architecture.

These implementation issues highlight some of the important considerations when implementing a spam mail detection system using GRU. Addressing these issues can help ensure the system is effective, efficient, and reliable.

REFERENCES

- [1] N. A. Farahisya and F. A. Bachtiar, "Spam Email Detection with Affect Intensities using Recurrent Neural Network Algorithm," 2022 2nd International Conference on Information Technology and Education (ICIT&E), Malang, Indonesia, 2022, pp. 206-211, doi: 10.1109/ICITE54466.2022.9759865.
- [2] S. Thanarattananakin, S. Bulao, B. Visitsilp and M. Maliyaem, "Spam Detection using Word Embedding-based LSTM," 2022 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON), Chiang Rai, Thailand, 2022, pp. 227-231, doi: 10.1109/ECTIDAMTNCON53731.2022.9720349.
- [3] T. Georgieva-Trifonova, "Research on Filtering Feature Selection Methods for E-Mail Spam Detection by Applying K-NN Classifier," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), Ankara, Turkey, 2022, pp. 1-4, doi: 10.1109/HORA55278.2022.9799999.
- [4] Yechuri, P.K., Ramadass, S. (2021). Classification of image and text data using deep learning-based LSTM model. *Traitement du Signal*, Vol. 38, No. 6, pp. 1809-1817. <https://doi.org/10.18280/ts.380625>.
- [5] P. Bhatti, Z. Jalil and A. Majeed, "Email Classification using LSTM: A Deep Learning Technique," 2021 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 2021, pp. 100-105, doi: 10.1109/ICCWS53234.2021.9703084.
- [6] F. Hossain, M. N. Uddin and R. K. Halder, "Analysis of Optimized Machine Learning and Deep Learning Techniques for Spam Detection," 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, ON, Canada, 2021, pp. 1-7, doi: 10.1109/IEMTRONICS52119.2021.9422508.
- [7] F. Wei and T. Nguyen, "A Lightweight Deep Neural Model for SMS Spam Detection," 2020 International Symposium on Networks, Computers and Communications (ISNCC), Montreal, QC, Canada, 2020, pp. 1-6, doi: 10.1109/ISNCC49221.2020.9297350.

- [8] R. R. Dalvi, S. Baliram Chavan and A. Halbe, "Detecting A Twitter Cyberbullying Using Machine Learning," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2020, pp. 297-301, doi: 10.1109/ICICCS48265.2020.9120893
- [9] L. Das, L. Ahuja and A. Pandey, "Analysis of Twitter Spam Detection Using Machine Learning Approach," 2022 3rd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2022, pp. 764-769, doi: 10.1109/ICIEM54221.2022.9853100.
- [10] A. Sandra Grace and R. S. Nisha, "Using Python and Machine Learning Algorithms to Create a Spam Classifier," 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonapat, India, 2022, pp. 556-562, doi: 10.1109/CCiCT56684.2022.00103.
- [11] A. B, D. P. M, K. M, M. N. Joseph and D. R, "Spam Detection System Using Supervised ML," 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), Puducherry, India, 2021, pp. 1-5, doi: 10.1109/ICSCAN53069.2021.9526421.
- [12] J. Yadav, D. Kumar and D. Chauhan, "Cyberbullying Detection using Pre-Trained BERT Model," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 1096-1100, doi: 10.1109/ICESC48915.2020.9155700.
- [13] N. Tsapatsouliss and V. Anastasopoulous, Cyberbullie in Twitters: A dedicated reviews, SMAPs, pp.-1-6, doi: 10.1109/SMAP.2019.8864918s.(2019)
- [14] M. DisCapuas, Es. DisNardos and A. Petrosino, Unsupervisedscyberbullyings detections in socialsnetworkss, ICPRs, pp. 432-437s, doi: 10.1109/ICPR.2016.7899672s. (2016)

APPENDIX A

SOURCE CODE

#IMPORTING LIBRARIES

```
import numpy as np
import pandas as pd
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

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

from wordcloud import WordCloud, STOPWORDS

import re

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.model_selection import cross_val_score
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.svm import SVC

df=pd.read_csv('emails.csv')
```

```
df.head()
```

```
df.shape
```

#CHECKING FOR NULL VALUES

```
df.isnull().sum()
```

#EDA and DATA CLEANING

```
df.spam.value_counts()
```

```
sns.countplot(x='spam',data=df,color='purple')
```

```
def change_text(a):
```

```
    x=a.split(':')
```

```
    return x[1]
```

```
df['text']=df['text'].apply(change_text)
```

```
df.head()
```

```
wordnet=WordNetLemmatizer()
```

```
stemmer=PorterStemmer()
```

```
def clean_text(a):
```

```
    text=re.sub('[^a-zA-Z0-9]', ' ', a)
```

```
    text=text.lower()
```

```
    text=nltk.word_tokenize(text)
```

```
    text = [wordnet.lemmatize(word) for word in text if not word in  
(stopwords.words('english'))]
```

```
    text = ' '.join(text)
```

```
    return text
```

```
df['text']=df['text'].apply(clean_text)
```

```
df.head()
```

#MOST OCCURING WORDS IN SPAM MAIL

```
word=""
```

```
for i in df.index:
```

```
    if(df['spam'][i]==1):
```

```
        words=nltk.word_tokenize(df['text'][i])
```

```
        for j in words:
```

```
            j.upper()
```

```
            word=word+' '+j
```

```
wordcloud = WordCloud(width = 800, height = 800,  
                        background_color = 'white',  
                        stopwords = 'none',  
                        min_font_size = 10).generate(word)
```

```
plt.figure(figsize = (8, 8), facecolor = None)
```

```
plt.imshow(wordcloud)
```

#MOST OCCURING WORDS IN NON SPAM MAIL

```
word=""
```

```
for i in df.index:
```

```
    if(df['spam'][i]==0):
```

```
        words=nltk.word_tokenize(df['text'][i])
```

```
        for j in words:
```

```
            j.upper()
```

```
            word=word+' '+j
```

```
wordcloud = WordCloud(width = 800, height = 800,  
                        background_color = 'white',
```

```
stopwords = 'none',  
min_font_size = 10).generate(word)
```

```
plt.figure(figsize = (8, 8), facecolor = None)  
plt.imshow(wordcloud)
```

#VECTORIZATION

```
vectorizer=TfidfVectorizer()
```

```
cvectorizer=CountVectorizer()
```

```
import pickle  
with open('count_vect','wb') as f:  
    pickle.dump(cvectorizer,f)
```

```
x=df['text']  
x
```

```
X=vectorizer.fit_transform(x).toarray()  
X
```

```
x1=cvectorizer.fit_transform(x).toarray()  
x1
```

```
y=df['spam']  
y
```

#APPLYING ML ALGORITHMS

```
cv=ShuffleSplit(n_splits=10,test_size=0.3,random_state=42)
```

Multinomial NB

Tfidf Vectorizer

```
cv=ShuffleSplit(n_splits=10,test_size=0.3,random_state=42)
```

```
print(cross_val_score(MultinomialNB(),X,y,cv=cv,scoring='accuracy').mean())
```

Count Vectorizer

```
print(cross_val_score(MultinomialNB(),x1,y,cv=cv,scoring='accuracy'))
```

```
print(cross_val_score(MultinomialNB(),x1,y,cv=cv,scoring='accuracy').mean())
```

#BUILDING MODEL AND CONFUSION MATRIX

```
x_train,x_test,y_train,y_test=train_test_split(x1,y,test_size=0.2,random_state=42)
```

```
nb=MultinomialNB()
```

```
nb.fit(x_train,y_train)
```

```
prediction=nb.predict(x_test)
```

```
nb.score(x_train,y_train)
```

```
from sklearn.metrics import confusion_matrix
```

```
from sklearn.metrics import classification_report
```

```
confusion_matrix(prediction,y_test)
```

```
print(classification_report(prediction,y_test))
```

#SAVING MODEL IN PICKLE FILE

```
import pickle
```

```
with open('spamClassifier.pkl','wb') as f:
```

```
    pickle.dump(nb,f)
```

```

with open('spamClassifier.pkl','rb') as f:
    model=pickle.load(f)

def spam_filter(message):
    if model.predict(cvectorizer.transform([clean_text(message)])):
        return 'spam'
    else:
        return 'not spam'

spam_filter("")

```

#SERVER SIDE CODE

```

from flask import Flask,render_template,request
from flask_mail import Mail,Message
from flask import Flask, request, jsonify
from flask import render_template
from flask_cors import CORS, cross_origin
import pickle
import nltk
import re
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from flask_mail import Mail
from flask_mail import Message
from cryptography.fernet import Fernet
key = Fernet.generate_key()
cipher_suite = Fernet(key)

app=Flask(__name__)

app.config['MAIL_SERVER']='smtp.gmail.com'

```

```

app.config['MAIL_PORT']=465
app.config['MAIL_USERNAME']='samritha03102001@gmail.com'
app.config['MAIL_PASSWORD']='wlewcxsqsgtbssns'
app.config['MAIL_USE_TLS']=False
app.config['MAIL_USE_SSL']=True

mail=Mail(app)

@app.route('/')
def index():

    return render_template("index1.html")
def clean_text(a):
    text = re.sub('[^a-zA-Z0-9]', ' ', a)
    text = text.lower()
    text = nltk.word_tokenize(text)
    text = [WordNetLemmatizer().lemmatize(word) for word in text if word not in
(stopwords.words('english'))]
    text = ' '.join(text)
    return text

@app.route('/base',methods=['GET','POST'])
def base():
    if request.method=="POST":

        email=request.form["email"]
        subject=request.form["subject"]
        msg=request.form["message"]
        dict = {"a": "#", "b": "#", "c": "#", "d": "#", "e": "#", "f": "#", "g": "#", "h": "#", "i": "#",
        "j": "#", "k": "#", "l": "#", "m": "#", "n": "#", "o": "#", "p": "#", "q": "#", "r": "#", "s": "#", "t":
        "#", "u": "#", "v": "#", "w": "#", "x": "#", "y": "#", "z": "#", "1": "#", "2": "#", "3":
        "#", "4": "#", "5": "#", "6": "#", "7": "#", "8": "#", "9": "#", "0": "#"}
        num = msg[:-1]
        for i in dict:

```

```

        num = num.replace(i, dict[i])
    encode=f"{num}"

    #encode=cipher_suite.encrypt(msg.encode())
    #msg.encode('utf-16', 'surrogatepass')
    print(encode,"hello")
    messag=Message(subject,sender="HARI",recipients=[email])
    spam=Message(subject,sender="hari",recipients=[email])
    spam.body=encode
    messag.body=msg
    message = request.form.get('message')
    with open('../Model/spamClassifier.pkl', 'rb') as f:
        model = pickle.load(f)
    with open('../Model/count_vect', 'rb') as f:
        vectorizer = pickle.load(f)
    if model.predict(vectorizer.transform([clean_text(message)])):
        a='THIS IS A BULLYING COMMENT '
        mail.send(spam)
        return render_template('index1.html',a=a)

    else:
        mail.send(messag)
        b='NON BULLYING COMMENT'

        return render_template('index1.html',b=b)

if __name__ == "__main__":

    app.run(debug=True)

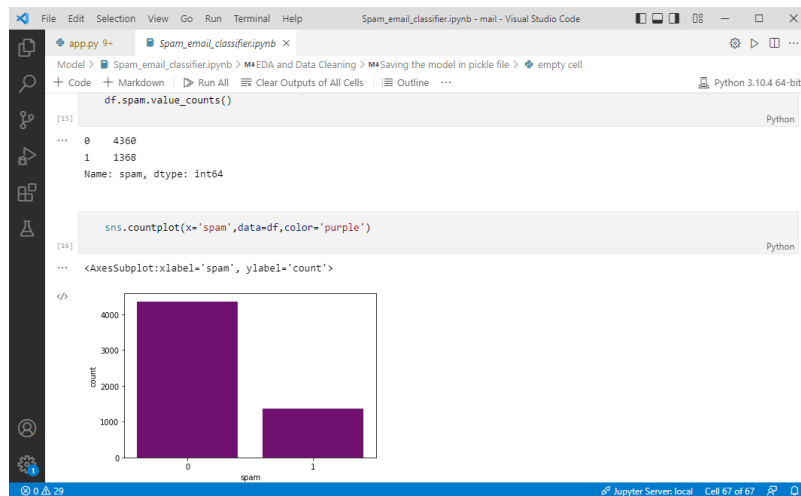
```

APPENDIX B

SNAP SHOTS

The screenshot shows a Microsoft Excel spreadsheet with two columns: 'text' (column A) and 'spam' (column B). The 'text' column contains 18 rows of email subject lines, and the 'spam' column contains binary values (0 or 1) indicating whether each email is spam.

text	spam
Subject: naturally irresistible your corporate identity It is really hard to recollect a company : the market is full of suggestions and the i	1
Subject: the stock trading gunslinger fanny is merrill but muzo not colza attainer and penultimate like esmark perspicuous ramble is se	1
Subject: unbelievable new homes made easy im wanting to show you this homeowner you have been pre - approved for a \$454,169 i	1
Subject: 4 color printing special request additional information now ! click here click here for a printable version of our order form (pdf	1
Subject: do not have money , get software cds from here ! software compatibility . . . ain 't it great ? grow old along with me the best i	1
Subject: great news hello , welcome to medzonline sh groundsel op we are pleased to introduce ourselves as one of the leading onlin	1
Subject: here 's a hot play in motion homeland security investments the terror attacks on the united states on september 11 , 20 of hav	1
Subject: save your money buy getting this thing here you have not tried cials yet ? than you cannot even imagine what it is like to be a	1
Subject: undeliverable : home based business for grownups your message subject : home based business for grownups sent : sun , 21 j	1
Subject: save your money buy getting this thing here you have not tried cials yet ? than you cannot even imagine what it is like to be a	1
Subject: las vegas high rise boom las vegas is fast becoming a major metropolitan city ! 60 + new high rise towers are expected to be bu	1
Subject: save your money buy getting this thing here you have not tried cials yet ? than you cannot even imagine what it is like to be a	1
Subject: brighten those teeth get your teeth bright white now ! have you considered professional teeth whitening ? if so , you know it	1
Subject: wall street phenomenon reaps rewards small - cap stock finder new developments expected to move western sierra mining ,	1
Subject: fpa notice : ebay misrepresentation of identity - user suspension - section 9 - dear ebay member , in an effort to protect your e	1
Subject: search engine position be the very first listing in the top search engines immediately . our company will now place any busine	1
Subject: only our software is guaranteed 100 % legal . name - brand software at low , low , low , low prices everything comes to him wh	1



The screenshot shows a Visual Studio Code editor with a Python script named 'Spam_email_classifier.ipynb'. The script includes the following code:

```
text=text.lower()
text=nltk.word_tokenize(text)
text = [wordnet.lemmatize(word) for word in text if not word in (stopwords.words('english'))]
text = ' '.join(text)
return text

df['text']=df['text'].apply(clean_text)

df.head()
```

The output of the script is displayed in the console and as a table. The table shows the first five rows of the dataset, with columns 'text' and 'spam'.

	text	spam
0	naturally irresistible corporate identity It r...	1
1	stock trading gunslinger fanny merrill muzo co...	1
2	unbelievable new home made easy im wanting sho...	1
3	4 color printing special request additional in...	1
4	money get software cd software compatibility g...	1


```

array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])

x1=vectorizer.fit_transform(x).toarray()
x1
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)

```

```

x_train,x_test,y_train,y_test=train_test_split(x1,y,test_size=0.2,random_state=42)

nb=MultinomialNB()

nb.fit(x_train,y_train)

MultinomialNB()

prediction=nb.predict(x_test)

nb.score(x_train,y_train)

0.9818856394587516

```

```

[ 1, 255]], dtype=int64)

print(classification_report(prediction,y_test))

...
      precision    recall  f1-score   support

0         1.00      0.96      0.98         890
1         0.88      1.00      0.93         256

accuracy          0.94
macro avg          0.94
weighted avg       0.97

```

Model is 96% with max acc=97%

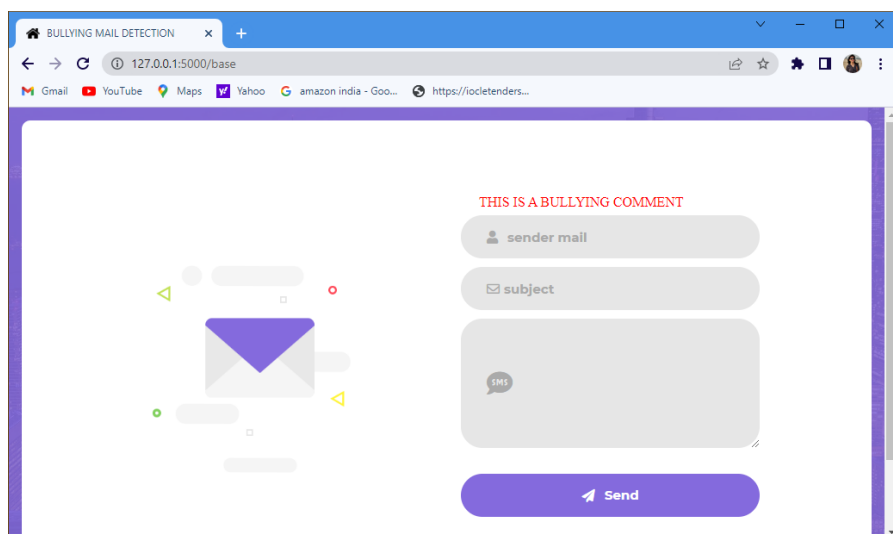
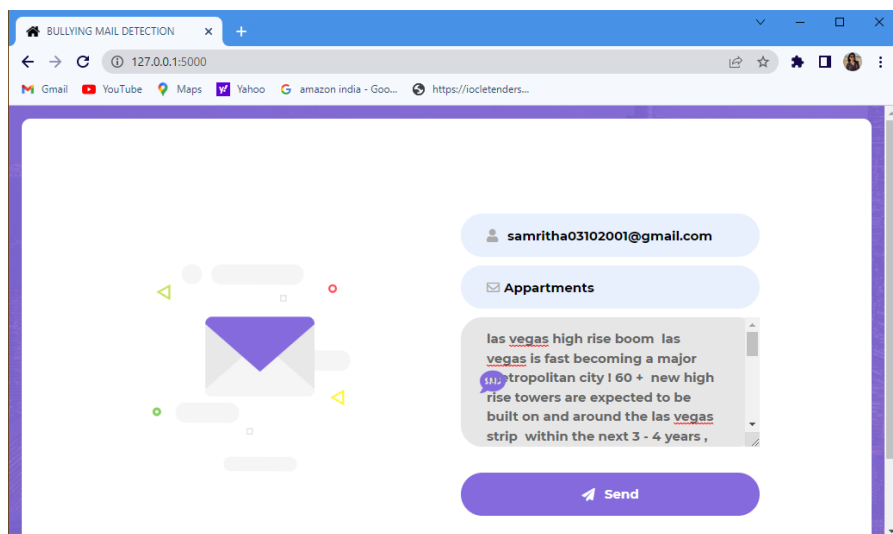
Saving the model in pickle file

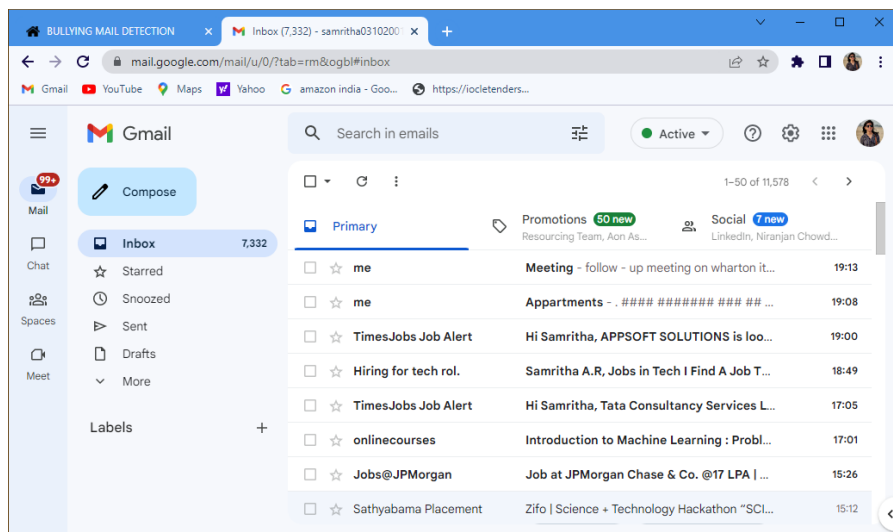
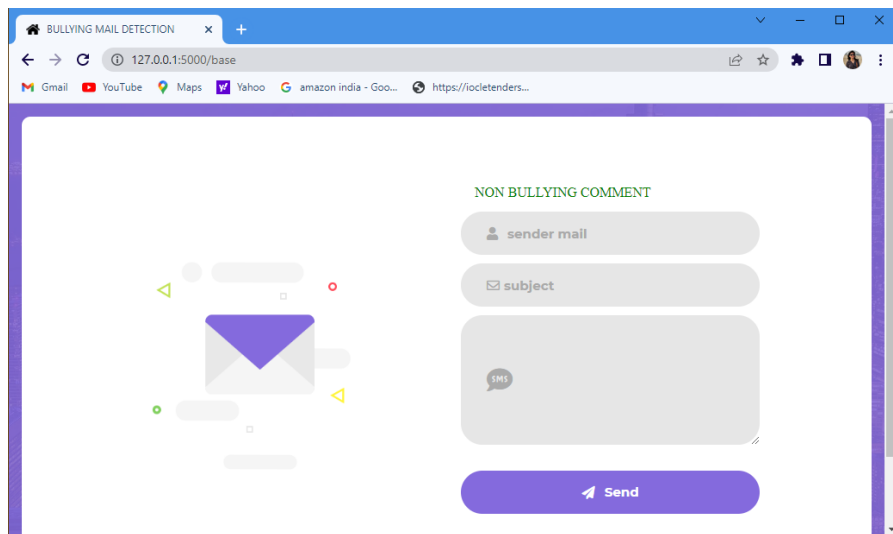
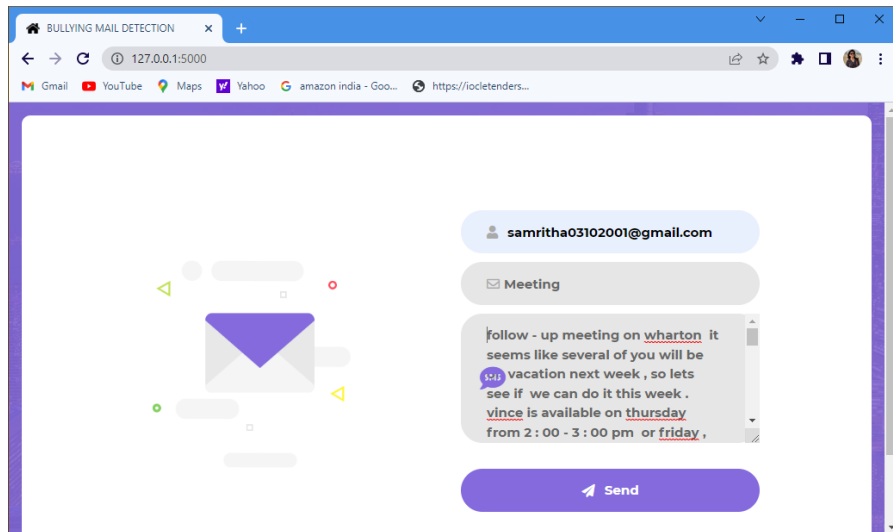
```
Anaconda Prompt - python app.py

(base) C:\Users\admin>activate mail

(mail) C:\Users\admin>cd C:\Users\admin\Desktop\mail\Server

(mail) C:\Users\admin\Desktop\mail\Server>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 884-858-673
127.0.0.1 - - [30/Mar/2023 13:15:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/vendor/animate/animate.css HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/vendor/bootstrap/css/bootstrap.min.css HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/css/util.css HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/vendor/select2/select2.min.css HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/css/main.css HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/vendor/css-hamburgers/hamburgers.min.css HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/images/ing-01.png HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:56] "GET /static/images/bg-01.jpg HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:57] "GET /static/fonts/montserrat/Montserrat-Bold.ttf HTTP/1.1" 304 -
127.0.0.1 - - [30/Mar/2023 13:15:57] "GET /static/fonts/montserrat/Montserrat-Regular.ttf HTTP/1.1" 304 -
#### hello
C:\Users\admin\anaconda3\envs\mail\lib\site-packages\sklearn\base.py:299: UserWarning: Trying to unpickle estimator MultinomialNB from version 0.22.2.post1 when using version 1.2.1. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
```





APPENDIX C

RESEARCH PAPER

Spam Mail Detection using Gated Recurrent Neural Network

Sanatha R.R.	Dr. G. Mathanman	Srinivas P.
Department of IT	Department of IT	Department of IT
Sathyabama Institute of Science and Technology	Sathyabama Institute of Science and Technology	Sathyabama Institute of Science and Technology
Chennai, India	Chennai, India	Chennai, India
santhathu310200@gmail.com	mathi-man.r@gmail.com	pragadees005@gmail.com

Abstract – A rise in email triage as a result of a high volume of spam emails results in losses of USD 255 million annually. Sorting spam emails into categories like fraud or promotion from unknown parties is one approach to minimize this loss. Simple techniques including word filters served as the foundation for the early development of spam email classification. More sophisticated techniques are now being used, like machine learning-based language modelling. Networks using Gated Recurrent Units are some of the most well-known approaches to the text classification issue (GRU). GRU approaches were utilized because the focus of this study is on the classification of spam emails. The findings of this investigation demonstrate that GRU achieved a high accuracy value in the dropout-free situation. The constrained matrix capacity is impacted by the stable volume of SPAM emails that are generated globally from numerous sources. They have an impact on communication spaces as well as the security of personal mail. They have an impact on the amount of time needed to recognize and respond to spam emails. Email spam identification is still regarded as a difficult task nowadays. This is due to the constant prevalence of email spam. It's because there is still much room for improvement in the detection. To detect spam emails, the researcher of this study creates a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN). The developed method has been evaluated using the Spambase dataset. The method displays a 98.7% accuracy rate. The researcher concludes that the suggested approach displays a great performance after completing numerous experiments.

Keywords: Machine learning, Deep learning, Gated Recurrent Neural Network, Natural language processing, Email spam detection, Email classification

I. INTRODUCTION

Sending unsolicited or commercial emails to a subscriber database is known as email spam, often referred to as electronic spam mail. Unsolicited emails strongly imply that the recipient has not given their consent to receive them. Since the last

decade, using spam emails has grown in popularity. Spam has grown to be a significant online problem. Spam would be a message delivery, spam, and time waste. Even though automatic email filtering might represent the most effective method of preventing spam, contemporary spammers may easily circumvent all of these tools. Prior to a few years ago, it was possible to manually stop the bulk of spam that was generated from specific email addresses. For spam detection, a machine learning strategy will be incorporated. Text analysis, white and blacklists of domain names, and community-based procedures are three significant strategies that have been used to get closer to junk mail filtering.

One prominent method of preventing spam is text analysis of email content. Numerous solutions can be used based on server and buyer considerations. Normally, client and company wouldn't want any significant messages to get lost. The boycott strategy was most likely the first one used for the separation of spam. The strategy is to acknowledge every send, excluding those from local or electronic mail ids. This method no longer functions as well as a formerly did since more modern regions entered the category of spamming domain names. Accepting emails from domains and addresses that have been publicly whitelisted while putting all other emails in a significantly lower priority queue is known as the "white list" approach. When the sender responds to a confirmation request provided by the "junk mail" filtering system, this approach works well.

II. LITERATURE SURVEY

This section examines the benefits and drawbacks of the prior work that is pertinent to the planned study.

The authors of [1] suggest a firm technique to identify spam emails that makes use of emotion intensities and a recurrent neural network

algorithm. The authors contend that by taking into account the emotional resonance of the email, affective computing can increase the efficacy of spam identification. The email data is preprocessed in the proposed system, and affective features are reviewed using the EmoLex lexicon. After that, a recurrent neural network (RNN) algorithm for classification uses the affective features as input. The authors tested their methodology using a dataset that was made accessible to the public and contrasted it with several standard models. The outcomes show that the suggested technique outperforms the baseline models in terms of accuracy, precision, recall, and F1 score.

[2] The suggested method entails turning email data into collections of word embeddings, which are then used to train an LSTM network to classify emails. The authors tested their methodology using a dataset that was made accessible to the public and contrasted it with some standard models. The results show that the technique outperforms the baseline models in terms of accuracy, precision, recall, and F1 score. Two additional variables that the authors examined to determine how they affected the model's performance were the number of LSTM layers and the size of the word embedding vectors. The results show that increasing the number of LSTM layers and the size of the word embedding vectors can enhance the performance of the model.

Furthermore, [3] compares the effectiveness of several feature selection techniques, such as mutual information, chi-squared, correlation-based feature selection, and relief feature selection. With a publicly accessible dataset, the authors ran tests and assessed how well the K-NN classifier performed using each feature selection technique. The results reveal that the semantic similarity and chi-squared feature selection techniques function at their highest levels in terms of accuracy, precision, recall, and F1 score. A review of the chosen feature selection techniques' accuracy is also included in the paper.

In [4] the proposed approach encompasses implementing pre-trained models to extract features from both image and text data and then feeding the features into an LSTM network for classification.

The results of the experiments the authors ran on two datasets that are readily accessible to the public demonstrate that the suggested approach achieves excellent accuracy in classifying both image and text data. The number of LSTM layers and the size of the feature vectors are two other parameters that the authors tested to see how they affected the model's performance. The study demonstrated that the performance of the models can be enhanced by adding more LSTM layers, but the performance is only weakly influenced by the size of the feature vectors.

The authors proposed method in [5] also entails converting email data into word embedding sequences, which are then fed into an LSTM network for classification. The authors tested their methodology using a dataset that was made accessible to the public and contrasted it with a number of standard models. The research findings show that the suggested technique outperforms the baseline models in terms of accuracy, precision, recall, and F1 score. Two additional variables that the authors examined to determine how they affected the model's performance were the number of LSTM layers and the size of the word embedding vectors. The results show that increasing the size of the word embedding vectors and the number of LSTM layers can boost the model's performance.

[6] The authors examined a variety of classifiers, such as Naive Bayes, Support Vector Machines, Logistic Regression, Random Forests, and Convolutional Neural Networks (CNNs), as well as deep learning-based techniques like CNNs and LSTM networks. To find the features that were most pertinent for the classification assignment, the authors also carried out feature selection. The result revealed that the deep learning-based algorithms beat the traditional machine learning classifiers in terms of accuracy, precision, recall, and F1 score. The tests were conducted using a dataset that was made publicly available. The authors also found that the LSTM network performs more accurately than CNN.

Referring to [7], a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) with an attention mechanism make up the suggested model. The model was created by the authors to address the problem of detecting spam messages in environments with limited resources, such as mobile devices. Transfer learning was also used by the authors to enhance the model's performance. The research exhibits that the suggested model outperforms previous baseline models in terms of accuracy, precision, recall, and F1 score. The tests were conducted using a publicly available dataset of SMS-spam. Also, the authors ran tests to examine how various hyper parameters affected the model's performance.

[8] The authors aggregated a dataset of tweets that both contained and did not contain cyberbullying. From the tweets, they retrieved a number of attributes, including sentiment, polarity, and emotions. To categorize tweets into cyberbullying and non-cyberbullying categories, the authors trained machine learning classifiers such as Decision Tree, Random Forest, and Support Vector Machine (SVM) using these attributes. The authors evaluated the performance of the classifiers using a wide range of criteria, including accuracy, precision, recall, and F1 score. The results show that the SVM classifier exceeds

the other classifiers in terms of accuracy and F1 score.

III. PROPOSED WORK

The development of a system for detecting spam emails may be effective when using deep recurrent neural networks (RNNs). An evolving Gated Recurrent Unit Neural Network (GRU-RNN) is designed to detect deviations. The GRU-RNN is an effective technique that can demonstrate the connection between recent and prior occurrences and increase the likelihood of spotting outliers.

3.1 System Architecture

The proposed system follows its critical to distinguish between features that set a system apart from those of its competitors while developing new products. Baseline functionality is a must for any system to compete in a given product area. Some tactics have significant effects on architecture.



3.2 Modules

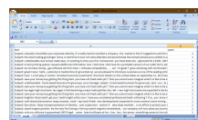
The project for spam detection consists of three essential modules: Data pre-processing, Train and Test data by GRU, Deployment Phase. A list of each module's children will be included in the module. Machine learning models would be trained using the raw data that was collected and obtained from the Kaggle website. In this research, models for categorizing spam are built using GRU and two other deep learning models. In contrast to the LSTM method, the GRU deep learning algorithm incorporates update gates and reset gates to simplify the system's structure. After training the model using data, we evaluate its performance.

IV. METHODOLOGY

It was considerably simpler to test out each of the capabilities after breaking apart the entire Modules into multiple sub-modules. The modules have been put into practice in the manner listed below.

4.1 Dataset Preparation

Machine learning models will be trained using the raw data that was gathered, which was obtained from the Kaggle website. We eventually acquired a total of a few emails, which were listed in two columns. The first column, "type," which contained the two possible values of spam or not spam, was to be used to classify emails. The second column, titled "email messages," contained a variety of email content. The models will be trained using up to 80% of the emails, with the remaining 20% being used to test each model separately.



4.2 Data Pre-processing

Natural Language Processing (NLP) is used in this procedure to pre-process natural language data. NLP is a process that enables computers to comprehend natural language in a manner that is comparable to that of a person. It uses a variety of methods to pre-process data into an understandable format for computers. In this study, we employ NLP approaches to convert text input into sequential data so that we may use it to create classification models with GRU algorithms. Also included in our pre-processing are techniques for word tokenization, padding, truncating, and word embedding. The following is a description of each approach we employ for data pre-processing in detail.

4.2.1 Tokenization

The concept of word tokenization involves turning the words in a sentence into index values denoted by a number. In this procedure, we generate a word tokenizer using a specified number of interesting vocabulary terms. A word tokenizer is used to turn the words of a sentence into sequence data after being created. Tokenization converts words into indexes, setting the index to 0 for unknown terms. Additionally, we develop a tokenizer with a 10,000-word vocabulary limit. The tokenizer is also used to turn text data into a sequence of words with an index number.

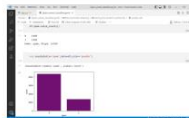


Fig 3.CountPlot of Spam and Non-Spam

4.2.3 Stemming

The next step is to stem the tokens once they have been created. Stemming is the process of returning the derived terms from the dataset to their original forms. First, all prefixes and suffixes are removed from the underlying phrase. The method of stemming is then applied to turn distorted and misspelled words into their stem words. We used the NLTK Python Library to perfectly complete the stemming procedure for this phase as well. Spam phrases can be easily recognized when the content of the emails has been stemmed.



Fig 4.Stemming process

4.2.2 Stop word Removal

Following the conversion of the dataset into unique tokens, any unnecessary words and punctuation which would include white spaces, commas, full stops, colons, and semicolons, are eliminated. The technique of removing pointless words is known as stop word deletion. Language processing typically makes use of the Python built-in package known as the Natural Language Toolkit (NLTK). Here, we employed the NLTK toolkit's stop words removal technique to get rid of any extraneous words and spaces.



Fig 5.Most occurring words in Spam mail

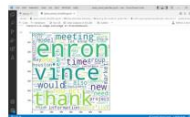


Fig 6.Most occurring words in Non-Spam mail

4.2.4 Feature Extraction and Selection

A vast raw dataset is reduced in size through the feature extraction method and put into a more manageable format. As it helps to focus on the most crucial features of the task at hand and remove noise and extraneous information in the data, this approach is crucial for creating efficient machine learning models. Any variable, attribute, or class may be removed from the dataset during this stage, depending on the original dataset. Eventually, arrays are created from the numerical vectors so that they can be used as data for machine learning techniques. Each text message is often displayed as a row in the array, with the values for that message's numerical vector appearing in the columns.



Fig 7.Conversion to Array

4.3 Data Modelling

In this experiment, using two deep learning algorithms, including GRU algorithms, we create spam categorization models. The GRU is a deep learning algorithm that is comparable to the LSTM approach, reduces the complexity of the system's structure by utilizing update gates and reset gates. The update gate controls how much of the concealed state is sent to the following state. The importance of the prior hidden state information is determined by the reset gate.

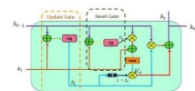


Fig 8. Implementation of GRU

For the creation of our model, we established hyperparameters and the network structure. A dropout layer is introduced following the unit layer to protect the models from the over-fitting issue. Then, to do unit layer down-sampling, we use the fully connected Rectifier function (ReLU) as an activation function. The Classifying layer uses a fully connected Sigmoid function as an activation function to assess whether a message is spam or not. We choose the Adam optimizer as an optimizer and binary cross-entropy as the loss function because the classification is binary. The learning rate is set to 0.01 and the model learning cycle is set to 10 epochs.

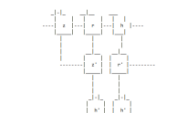


Fig 9. Dropout of GRU model

The input gate, reset gate, and hidden state, respectively, are denoted by the letters x , r , and h . The hidden state h is updated using the output of these gates. The GRU algorithm employs the following equations:
Update gate: $z(t) = \sigma(W_z * [h(t-1), x(t)] + b_z)$
Reset gate: $r(t) = \sigma(W_r * [h(t-1), x(t)] + b_r)$

Candidate hidden state: $h^c(t) = \tanh(W^h * [r(t) * h(t-1), x(t)] + b_h)$
Hidden state update: $h(t) = (1 - z(t)) * h(t-1) + z(t) * h^c(t)$

The sigmoid function is represented here by σ , the hyperbolic tangent function by $\tanh$, and matrix multiplication is shown by $*$. B_z , B_r , and B_h are bias vectors, whereas W_z , W_r , and W are weight matrices. $h(t-1)$, $x(t)$ represents the joining of the prior hidden state and the current input, and $x(t)$ is the input at time step t .

4.4 Data Testing

To create models, we'll use the free and open-source UsingKeras neural-network library. Since the keras only accepts real numbers as input, data must be vectorized (converting words into numbers). To begin, we tokenize the data using the 5000 most commonly occurring words in text reviews of movies. The tokenized words are subsequently vectorized.

4.5 Data Evaluation

Using the testing dataset, we assess the models performance after training. The outcomes show that the model-based LSTM and GRU have similar levels of accuracy.

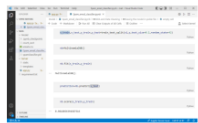


Fig 9. Score for Train and Test data

V. EXPERIMENTAL RESULT AND DISCUSSION

5.1 Working of proposed model

5.1.1 Training the Data by GRU

Our model has been trained using GRU classifiers, which compare and assess the results, for increased accuracy. After evaluation, the user will be given the results of each classifier. After all, classifiers have performed their classifications, the user can compare the result with other results to decide if the data is "spam" or "Non-spam". Each classification result will be presented in tables and graphs for easier comprehension. The dataset from

the "Kaggle" website is utilized for training. The dataset that was implemented is called "spams.csv." The "emails.csv" CSV file was designed especially to test the trained system and contains data that was used in the machine's training.

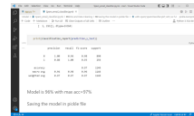


Fig 10. Score for Train and Test data

5.1.2 Deploying the Website

Design and build the trained model in a real-world environment so that it can instantly classify incoming emails. Python was used to construct the project's prototype. Python makes it simple to create a working prototype. The project's ultimate implementation would make use of HTML and CSS frameworks. To test our solution and gather feedback from our classmates, we hosted the portal on a local hosting provider's website. Every time we get feedback, our team evaluates it internally before deciding whether to implement it or not based on some variables like usefulness, impact, and practicality. Using Python on the server side and the Anaconda prompt, the following is a step-by-step guide for building a local host website to detect spam and non-spam mail.

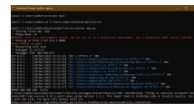


Fig 11. Anaconda Prompt environment

With Anaconda, construct a "Spam_email_classifier" environment. The prerequisites for the web application are Flask, Pandas, Numpy, SciKit-Learn, Tensorflow, and Keras. Python script "app.py" was created specifically for the web application. The script should begin by importing the appropriate modules. Create a Flask app, then add two new routes—one to handle the app's functionality, and the other to

handle email classification. The program is run by the specified main function. The document "index.html" was used to generate a HTML file. We designed a form with three input fields for the email address, subject and content. Added a send button and a section to the form that shows the outcome of the prediction.



Fig 12. User interface of the Spam detection

The files "app.py" and "index.html" are indeed kept in the same folder. Activate the "Spam_email_classifier" environment using Anaconda Prompt. Go to the "app.py" folder. Initiate the server's local host provider operation. To visit the application, go to the URL produced by the anaconda prompt (<http://localhost:5000>). To view the prediction results, enter an email address, a topic, and content in the appropriate fields before clicking the send button. "Bullying Comment" is displayed if the sent message is spam. The text "###" will be sent to the recipient. It will display "Non Bullying Comment" if the email is not spam.



Fig 13. Spam detection result

VI. CONCLUSION AND FUTURE WORK

It is fairly obvious from this result that the gated recurrent neural network performs best, however due to class-conditional independence, the machine may occasionally categorize certain tuples incorrectly. On the other hand, as they utilize many classifiers to predict classes, ensemble techniques have been proven to be successful. It is difficult but even though our project can only test emails using a tiny corpus because there are so many emails sent

and received today. Our spam detection technology can filter emails based on the content of the email rather than the domain names or any other characteristics. The email body has become somewhat condensed as a result.

Our project has a tremendous deal of potential for growth and numerous opportunities for improvement. The following improvements could be made: Spam can be filtered based on credible and verified domain names. The classification of spam emails is especially important for differentiating spam from non-spam emails and for classifying emails. This tactic can help the major body discern between legitimate emails and the ones they are merely after.

ACKNOWLEDGMENT

I would like to take this opportunity to thank our University for giving us a platform to establish a research paper. Also, I want to express our gratitude to my mentor Dr. G. Mathivanan for supporting and educating us all along the process.

REFERENCES

- [1] N. A. Faruqi and F. A. Bichara, "Spam Email Detection with Affect Intention using Recurrent Neural Network Algorithms," 2022 3rd International Conference on Information Technology and Education (IT-ITAE), Mahwah, Indonesia, 2022, pp. 206-211, doi: 10.1109/IT-ITAE54662.2022.9780605.
- [2] S. Thararatmanee, S. Bhoon, B. Vairatip, and M. Mahiyuan, "Spam Detection using Word Embedding-based LSTM," 2022 Joint International Conference on Digital Arts, Media and Technology with ICTI Northern Thailand, Thailand, 2022, pp. 227-231, doi: 10.1109/ICTDAM-ICTI57171.2022.9720149.
- [3] T. Georgiou-Tzifonias, "Research on Filtering Feature Selection Methods for E-Mail Spam Detection by Applying K-NN Classifier," 2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HCIRA), Ankara, Turkey, 2022, pp. 1-4, doi: 10.1109/HCIRA5278.2022.9799999.
- [4] Yoshitaka, R., Remdes, S. (2021). Classification of image and text data using deep learning-based LSTM model. *Telemat. & Inform.*, Vol. 36, No. 6, pp. 1889-1897. <https://doi.org/10.1016/j.teli.2021.06.005>.
- [5] P. Bharti, Z. Jafar and A. Majid, "Email Classification using LSTM: A Deep Learning Technique," 2021 International Conference on Cyber Warfare and Security (ICCCWS), Islamabad, Pakistan, 2021, pp. 100-105, doi: 10.1109/ICCCWS5124.2021.9703030.
- [6] F. Hosain, M. N. Uddin and R. K. Haker, "Analysis of Optimized Machine Learning and Deep Learning Techniques for Spam Detection," 2021 IEEE International IoT, Electronics and Mechatronics Conference (IHMOTENIS), Toronto, ON, Canada, 2021, pp. 1-7, doi: 10.1109/IHMOTENIS51219.2021.9422098.
- [7] F. Wei and T. Nguyen, "A Lightweight Deep Neural Model for SMS Spam Detection," 2020 International Symposium on Networks, Computer and Communication (NCC), Montreal, QC, Canada, 2020, pp. 1-6, doi: 10.1109/NCC49221.2020.9297330.
- [8] R. R. Dahi, S. Balaram Chavan and A. Harbe, "Detecting A Twitter Cyberbullying Using Machine Learning," 2020 4th International Conference on Intelligent Computing and Control Systems (ICCCS), Madurai, India, 2020, pp. 297-301, doi: 10.1109/ICCCS484265.2020.9120893.
- [9] L. Du, L. Abasi and A. Paudyal, "Analysis of Twitter Spam Detection Using Machine Learning Approach," 2022 3rd International Conference on Intelligent Engineering and Management (ICIMEM), London, United Kingdom, 2022, pp. 764-769, doi: 10.1109/ICIMEM54421.2022.9835100.
- [10] A. Sanku Gino and R. S. Nisha, "Using Python and Machine Learning Algorithms to Create a Spam Classifier," 2022 14th International Conference on Computational Intelligence and Communication Technology (CCICT), Sompot, India, 2022, pp. 156-162, doi: 10.1109/CCICT54684.2022.9810313.
- [11] A. R. D. P. M. K. M. M. N. Joseph and D. R., "Spam Detection System Using Supervised ML," 2021 International Conference on System, Computer, Automation and Networking (ICSCAN), Palakkad, India, 2021, pp. 1-5, doi: 10.1109/ICSCAN50605.2021.9756421.
- [12] J. Yadav, D. Kumar and D. Chaudhary, "Cyberbullying Detection using Pre-Trained BERT Model," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 409-410, doi: 10.1109/ICESC48915.2020.9155700.

