

SPAM REVIEW DETECTION USING THE LINGUISTIC AND SPAMMER BEHAVIORAL METHODS

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KURUPATI SAITEJA (Reg. No: 37110397)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
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 – 600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KURUPATI SAITEJA (37110397)** who carried out the project entitled "**SPAM REVIEW DETECTION USING THE LINGUISTIC AND SPAMMER BEHAVIORAL METHODS**" under my supervision from **August 2020** to **March 2021**.

Internal Guide

Dr.Prem Jacob, M.E.,Ph.D.,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **MADDINENI PAVAN KUMAR (37110397)** hereby declare that the Project Report entitled “**SPAM REVIEW DETECTION USING THE LINGUISTIC AND SPAMMER**

BEHAVIORAL METHODS” is done by me under the guidance of **Dr.PREM JACOB, M.E.,Ph.D.**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** 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. S. Vigneswari, M.E., Ph.D., and Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering 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. PREM JACOB, M.E., Ph.D.**, for her 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 Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Online review systems play an important role in affecting consumers' behaviors and decision making, attracting many spammers to insert fake reviews to manipulate review content and ratings. To increase utility and improve user experience, some online review systems allow users to form social relationships between each other and encourage their interactions. In this paper, we aim at providing an efficient and effective method to identify review spammers by incorporating social relations based on two assumptions that people are more likely to consider reviews from those connected with them as trustworthy, and review spammers are less likely to maintain a large relationship network with normal users. The contributions of this paper are two-fold: (1) We elaborate how social relationships can be incorporated into review rating prediction and propose a trust-based rating prediction model using proximity as trust weight, and (2) We design a trust-aware detection model based on rating variance which iteratively calculates user-specific overall trustworthiness scores as the indicator for spamicity. Experiments on the dataset collected from Yelp.com show that the proposed trust-based prediction achieves a higher accuracy than standard CF method, and there exists a strong correlation between social relationships and the overall trustworthiness score.

INDEX

CHAPTER	TITLE	PageNo.
1.	INTRODUCTION	1
	1.1 DOMAIN INTRODUCTION	2
	1.1.1 SOCIAL NETWORK	2
	1.1.2 EXAMPLES	3
	1.1.3 OPPURTUNITIES AND CHALLENGES	3
	1.2 SECURE COMPUTING	4
	1.2.1 BASIC NEEDS FOR SECURE COMPUTING	4
	1.2.2 BENEFITS OF SECURE COMPUTING	4
2.	LITERATURE SURVEY	7
3.	AIM AND SCOPE	11
	3.1 PROPOSED SYSTEM	11
	3.2 ADVANTAGES OF PROPOSED SYSTEM	11
4.	EXPERIMENTAL OR MATERIALS AND METHODS	13
	ALGORITHMS USED	
	4.1 SYSTEM REQUIREMENTS	13
	4.1.1 HARDWARE REQUIREMENTS	13
	4.1.2 SOFTWARE REQUIREMENTS	13
	4.2 SOFTWARE ENVIRONMENT	13
	4.2.1 PYTHON	13
	4.2.2 PYTHON FEATURES	14
	4.3 GAMING PYTHON	14
	4.3.1 FIRST PYTHON PROGRAMMING	15
	4.4 MODULES	19
	4.4.1 MODULE DESCRIPTIONS	20
	4.4.2 REQUIREMENT ANALYSIS	21
	4.4.3 FUNCTIONAL REQUIREMENTS	21
	4.4.4 NON-FUNCTIONAL REQUIREMENTS	22

	4.5 ALGORITHMS USED	23
	4.5.1 RANDOM FOREST ALGORITHM	23
	4.5.2 DECISION TREE ALGORITHM	25
5.	RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS	27
	5.1 OUTPUT DESIGN	27
	5.2 SYSTEM STUDY	27
	5.2.1 FEASIBILITY STUDY	27
	5.3 DATA FLOW DIAGRAM	28
	5.4 UML DIAGRAMS	30
	5.4.1 GOALS	31
	5.4.2 USE CASE DIAGRAMS	31
	5.5 CLASS DIAGRAMS	33
	5.6 SEQUENCE DIAGRAM	33
	5.7 ACTIVITY DIAGRAM	34
	5.8 SYSTEM TESTING	35
	5.9 TYPES OF TESTS	36
6.	SUMMARY AND CONCLUSIONS	41
	6.1 CONCLUSION	41
	6.2 SUMMARY	41

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
1.1	SECURE COMPUTING	4
3.1	ARCHITECTURE DIAGRAM	12
4.1	ENTER NAME	18
4.2	WELCOME	19
5.1	DATA FLOW DIAGRAM	30
5.2	USE CASE DIAGRAM	32
5.3	CLASS DIAGRAM	33
5.4	SEQUENCE DIAGRAM	34
5.7	ACTIVITY DIAGRAM	35

CHAPTER-1

INTRODUCTION

It has become quite unpretentious to obtain any kind of information from any source across the world by using the Internet. The increased demand of social sites permits users to collect abundant amount of information and data about users. Huge volumes of data available on the sites also draw the attention of fake users. Twitter has rapidly become an online source for acquiring real-time information about users. Twitter is an Online Social Network (OSN) where users can share anything and everything, such as news, opinions, and even their moods. Several arguments can be held over different topics, such as politics, current affairs, and important events. When a user tweets something, it is instantly conveyed to his/her followers, allowing them to outspread the received information at a much broader level [2]. With the evolution of OSNs, the need to study and analyze users' behaviors in online social platforms has intensives. Many people who do not have much information regarding the OSNs can easily be tricked by the fraudsters. There is also a demand to combat and place a control on the people who use OSNs only for advertisements and thus spam other people's accounts. Recently, the detection of spam in social networking sites attracted the attention of researchers. Spam detection is a difficult task in maintaining the security of social networks. It is essential to recognize spams in the OSN sites to save users from various kinds of malicious attacks and to preserve their security and privacy. These hazardous maneuvers adopted by spammers cause massive destruction of the community in the real world. Twitter spammers have various objectives, such as spreading invalid information, fake news, rumors, and spontaneous messages. Spammers achieve their malicious objectives through advertisements and several other means where they support different mailing lists and subsequently dispatch spam messages randomly to broadcast their interests. These activities cause disturbance to the original users who are known as non-spammers. In addition, it also decreases the reputation of the OSN platforms. Therefore, it is essential to design a scheme to spot spammers so that corrective efforts can be taken to counter their malicious activities. Several research works have been carried out in the domain of Twitter spam detection. To encompass the existing state-of the-art, a few surveys have also been carried out on fake user

identification from Twitter. Tingmin et al. provide a survey of new methods and techniques to identify Twitter spam detection. The above survey presents a comparative study of the current approaches. On the other hand, the authors have conducted a survey on different behaviors exhibited by spammers on Twitter social network. The study also provides a literature review that recognizes the existence of spammers on Twitter social network. Despite all the existing studies, there is still a gap in the existing literature. Therefore, to bridge the gap, we review state-of-the-art in the spammer detection and fake user identification on Twitter. Moreover, this survey presents a taxonomy of the Twitter spam detection approaches and attempts to offer a detailed description of recent developments in the domain. The aim of this paper is to identify different approaches of spam detection on Twitter and to present a taxonomy by classifying these approaches into several categories. For classification we have identified four means of reporting spammers that can be helpful in identifying fake identities of users. Spammers can be identified based on: (i) fake content, (ii) URL based spam detection, (iii) detecting spam in trending topics, and (iv) fake user identification. Table 1 provides a comparison of existing techniques and helps users to recognize the significance and effectiveness of the proposed methodologies in addition to providing a comparison of their goals and results. Table 2 compares different features that are used for identifying spam on Twitter. We anticipate that these surveys will help reader's find diverse information on spammer detection techniques at a single point. This article is structured such that Section II presents the taxonomy for the spammer detection techniques on Twitter. The comparison of proposed methods for detecting spammers on Twitter is discussed in Section III. Section IV presents an overall analysis and discussion, whereas Section V concludes the paper and highlights some directions for future work.

1.1 DOMAIN INTRODUCTION

1.1.1 Social Network

Wikipedia defines a social network service as a service which “focuses on the building and verifying of online social networks for communities of people who share interests and activities, or who are interested in exploring the interests and activities of others, and which necessitates the use of software.”

A report published by OCLC provides the following definition of social networking sites: “Web sites primarily designed to facilitate interaction between users who share interests, attitudes and activities, such as Facebook, Mixi and MySpace.”

1.1.2 Examples of Social Networking Services

Examples of popular social networking services include:

Facebook: Facebook is a social networking Web site that allows people to communicate with their friends and exchange information. In May 2007 Facebook launched the Facebook Platform which provides a framework for developers to create applications that interact with core Facebook features

MySpace: MySpace is a social networking Web site offering an interactive, user-submitted network of friends, personal profiles, blogs and groups, commonly used for sharing photos, music and videos.

Ning: An online platform for creating social Web sites and social networks aimed at users who want to create networks around specific interests or have limited technical skills.

Twitter: Twitter is an example of a micro-blogging service. Twitter can be used in a variety of ways including sharing brief information with users and providing support for one’s peers.

Note that this brief list of popular social networking services omits popular social sharing services such as Flickr and YouTube.

1.1.3 Opportunities and Challenges

The popularity and ease of use of social networking services have excited institutions with their potential in a variety of areas. However effective use of social networking services poses a number of challenges for institutions including long-term sustainability of the services; user concerns over use of social tools in a work or study context; a variety of technical issues and legal issues such as copyright, privacy, accessibility; etc. Institutions would be advised to consider carefully the implications before promoting significant use of such services.

1.2 Secure Computing

Computer security Also known as cyber security or IT Security is information security as applied to computers and networks. The field covers all the processes and mechanisms by which computer-based equipment, information and services are protected from unintended or unauthorized access, change or destruction. Computer security also includes protection from unplanned events and natural disasters. Otherwise, in the computer industry, the term security -- or the phrase computer security -- refers to techniques for ensuring that data stored in a computer cannot be read or compromised by any individuals without authorization. Most computer security measures involve data encryption and passwords. Data encryption is the translation of data into a form that is unintelligible without a deciphering mechanism. A password is a secret word or phrase that gives a user access to a particular program or system.



Fig 1.1 Secure Computing

1.2.1 Basic needs for secure computing:

If you don't take basic steps to protect your work computer, you put it and all the information on it at risk. You can potentially compromise the operation of other computers on your organization's network, or even the functioning of the network as a whole.

1. Physical security:
2. Access passwords:
3. Prying eye protection
4. Anti-virus software
5. Firewalls
6. Software updates

7. Keep secure backups
8. Report problems

1.2.3 Benefits of secure computing:

- **Protect yourself - Civil liability:**

You may be held legally liable to compensate a third party should they experience financial damage or distress as a result of their personal data being stolen from you or leaked by you.

- **Protect your credibility - Compliance:**

You may require compliancy with the Data Protection Act, the FSA, SOX or other regulatory standards. Each of these bodies stipulates that certain measures be taken to protect the data on your network.

- **Protect your reputation – Spam:**

A common use for infected systems is to join them to a botnet (a collection of infected machines which takes orders from a command server) and use them to send out spam. This spam can be traced back to you, your server could be blacklisted and you could be unable to send email.

- **Protect your income - Competitive advantage:**

There are a number of “hackers-for-hire” advertising their services on the internet selling their skills in breaking into company’s servers to steal client databases, proprietary software, merger and acquisition information, personnel detailset *al*.

- **Protect your business – Blackmail:**

A seldom-reported source of income for “hackers” is to break into your server, change all your passwords and lock you out of it. The password is then sold back to you. Note: the “hackers” may implant a backdoor program on your server so that they can repeat the exercise at will.

- **Protect your investment - Free storage:**

Your server’s harddrive space is used (or sold on) to house the hacker's video clips, music collections, pirated software or worse. Your server or computer then becomes continuously slow and your internet connection speeds deteriorate due to

the number of people connecting to your server in order to download the offered wares.

CHAPTER-2

LITERATURESURVEY

2.1 Statistical features-based real-time detection of drifted Twitter spam

Authors: C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min

Twitter spam has become a critical problem nowadays. Recent works focus on applying machine learning techniques for Twitter spam detection, which make use of the statistical features of tweets. In our labeled tweets data set, however, we observe that the statistical properties of spam tweets vary over time, and thus, the performance of existing machine learning-based classifiers decreases. This issue is referred to as “Twitter Spam Drift”. In order to tackle this problem, we first carry out a deep analysis on the statistical features of one million spam tweets and one million non-spam tweets, and then propose a novel Lfun scheme. The proposed scheme can discover “changed” spam tweets from unlabeled tweets and incorporate them into classifier's training process. A number of experiments are performed to evaluate the proposed scheme. The results show that our proposed Lfun scheme can significantly improve the spam detection accuracy in real-world scenarios.

2.2 Automatically identifying fake news in popular Twitter threads

Authors: C. Buntain and J. Golbeck

Information quality in social media is an increasingly important issue, but web-scale data hinders experts' ability to assess and correct much of the inaccurate content, or “fake news,” present in these platforms. This paper develops a method for automating fake news detection on Twitter by learning to predict accuracy assessments in two credibility-focused Twitter datasets: CRED BANK, a crowdsourced dataset of accuracy assessments for events in Twitter, and PHEME, a dataset of potential rumors in Twitter and journalistic assessments of their accuracies. We apply this method to Twitter content sourced from BuzzFeed's fake news dataset and show models trained against crowdsourced workers outperform models based on journalists' assessment and models trained on a pooled dataset of both crowdsourced workers and journalists. All three datasets, aligned into a uniform format, are also publicly available. A feature

analysis then identifies features that are most predictive for crowdsourced and journalistic accuracy assessments, results of which are consistent with prior work. We close with a discussion contrasting accuracy and credibility and why models of non-experts outperform models of journalists for fake news detection in Twitter.

2.3 A performance evaluation of machine learning-based streaming spam tweets detection

Authors: C. Chen, J. Zhang, Y. Xie, Y. Xiang, W. Zhou, M. M. Hassan, A. AlElaiwi, and M. Alrubaian

The popularity of Twitter attracts more and more spammers. Spammers send unwanted tweets to Twitter users to promote websites or services, which are harmful to normal users. In order to stop spammers, researchers have proposed a number of mechanisms. The focus of recent works is on the application of machine learning techniques into Twitter spam detection. However, tweets are retrieved in a streaming way, and Twitter provides the Streaming API for developers and researchers to access public tweets in real time. There lacks a performance evaluation of existing machine learning-based streaming spam detection methods. In this paper, we bridged the gap by carrying out a performance evaluation, which was from three different aspects of data, feature, and model. A big ground-truth of over 600 million public tweets was created by using a commercial URL-based security tool. For real-time spam detection, we further extracted 12 lightweight features for tweet representation. Spam detection was then transformed to a binary classification problem in the feature space and can be solved by conventional machine learning algorithms. We evaluated the impact of different factors to the spam detection performance, which included spam to nonspam ratio, feature discretization, training data size, data sampling, time-related data, and machine learning algorithms. The results show the streaming spam tweet detection is still a big challenge and a robust detection technique should take into account the three aspects of data, feature, and model.

2.4 A model-based approach for identifying spammers in social networks

Authors: F. Fathaliani and M. Bouguessa

In this paper, we view the task of identifying spammers in social networks from a

mixture modeling perspective, based on which we devise a principled unsupervised approach to detect spammers. In our approach, we first represent each user of the social network with a feature vector that reflects its behavior and interactions with other participants. Next, based on the estimated users feature vectors, we propose a statistical framework that uses the Dirichlet distribution in order to identify spammers. The proposed approach is able to automatically discriminate between spammers and legitimate users, while existing unsupervised approaches require human intervention in order to set informal threshold parameters to detect spammers. Furthermore, our approach is general in the sense that it can be applied to different online social sites. To demonstrate the suitability of the proposed method, we conducted experiments on real data extracted from Instagram and Twitter.

2.5 Spam detection of Twitter traffic: A framework based on random forests and non-uniform feature sampling

Authors: C. Meda, E. Ragusa, C. Gianoglio, R. Zunino, A. Ottaviano, E. Scillia, and R. Surlinelli

Law Enforcement Agencies cover a crucial role in the analysis of open data and need effective techniques to filter troublesome information. In a real scenario, Law Enforcement Agencies analyze Social Networks, i.e. Twitter, monitoring events and profiling accounts. Unfortunately, between the huge amount of internet users, there are people that use microblogs for harassing other people or spreading malicious contents. Users' classification and spammers' identification is a useful technique for relieve Twitter traffic from uninformative content. This work proposes a framework that exploits a non-uniform feature sampling inside a gray box Machine Learning System, using a variant of the Random Forests Algorithm to identify spammers inside Twitter traffic. Experiments are made on a popular Twitter dataset and on a new dataset of Twitter users. The new provided Twitter dataset is made up of users labeled as spammers or legitimate users, described by 54 features. Experimental results demonstrate the effectiveness of enriched feature sampling method.

2.6 Existing system

- Tingminet *al.* provide a survey of new methods and techniques to identify Twitter spam detection. The above survey presents a comparative study of the current approaches.
- On the other hand, S. J. Somanet. *al.* conducted a survey on different behaviors exhibited by spammers on Twitter social network. The study also provides a literature review that recognizes the existence of spammers on Twitter social network.
- Despite all the existing studies, there is still a gap in the existing literature. Therefore, to bridge the gap, we review state-of-the-art in the spammer detection and fake user identification on Twitter

CHAPTER-3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 PROPOSED SYSTEM

- ❖ The aim of this paper is to identify fake user detection on Twitter and to present a framework by classifying these approaches into several categories. For classification, we have identified four means of reporting spammers that can be helpful in identifying fake identities of users. Spammers can be identified based on: (i) fake content, (ii) URL based spam detection, (iii) detecting spam in trending topics, and (iv) fake user identification.
- ❖ Moreover, the analysis also shows that machine learning-based techniques can be effective for identifying fake user on Twitter. However, the selection of the most feasible techniques and methods is highly dependent on the available data.



3.2 ADVANTAGES OF PROPOSED SYSTEM

- ❖ This study includes machine learning methodology proposed using real time datasets and with different characteristics and accomplishments.
- ❖ The proposed system is more effective and accurate than other existing systems. Tested with real time data's. Online review systems play an important role in affecting consumers' behaviors and decision making, attracting many spammers to insert fake reviews to manipulate review content and ratings.
- ❖ To increase utility and improve user experience, some online review systems allow users to form social relationships between each other and encourage their interactions.
- ❖ In this paper, we aim at providing an efficient and effective method to identify review spammers by incorporating social relations based on two assumptions that people are more likely to consider reviews from those connected with them as trustworthy, and review spammers are less likely to maintain a large relationship network with normal users.
- ❖ The contributions of this paper are two-fold: (1) We elaborate how social relationships can be incorporated into review rating prediction and propose a

trust-based rating prediction model using proximity as trust weight, and (2) We design a trust-aware detection model based on rating variance which iteratively calculates user-specific overall trustworthiness scores as the indicator for spamicity.

- ❖ Experiments on the dataset collected from Yelp.com show that the proposed trust-based prediction achieves a higher accuracy than standard CF method, and there exists a strong correlation between social relationships and the overall trustworthiness scores.

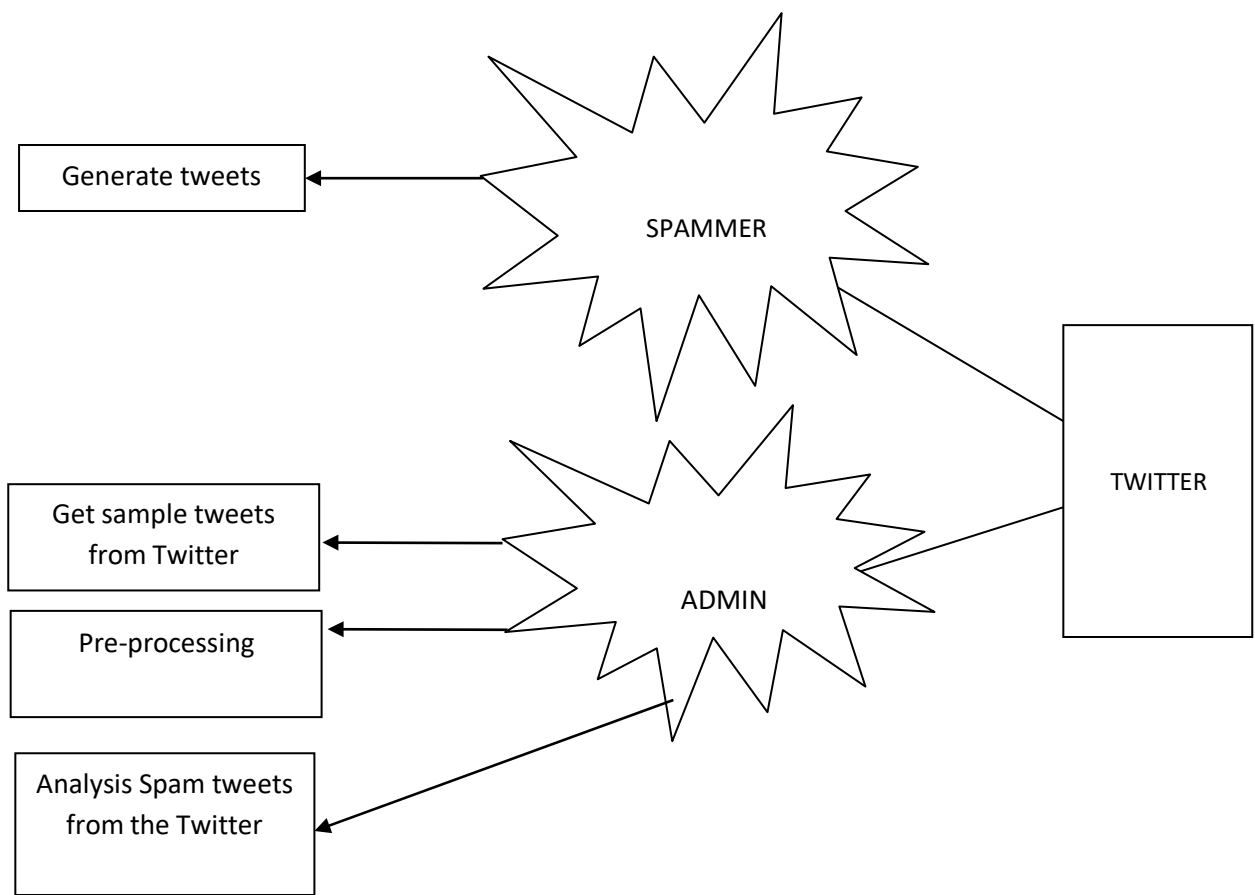


Fig 3.1 ARCHITECTURE DIAGRAM

CHAPTER-4

EXPERIMENTAL MATERIALS AND METHODS; ALGORITHMS USED

4.1 SYSTEM REQUIREMENTS:

4.1.1 Hardware requirements:

- System : Pentium Dual Core.
- Hard Disk : 120 GB.
- Monitor : 15" LED
- Input Devices : Keyboard, Mouse
- Ram : 4 GB.

4.1.2 Software requirements:

- Operating system: Windows 7/10.
- Coding Language :Python
- Tool : Pi-champ
- Database : MYSQL

4.2 SOFTWARE ENVIRONMENT

4.2.1 Python:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

4.2.2 Python features

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

4.3 Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python <https://www.python.org>.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <https://www.python.org/downloads/>.

- Follow the link for the Windows installer python-XYZ.msifile where XYZ is the version you need to install.
- To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

4.3.1 FirstPythonProgram

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt –

```
$ python
Python2.4.3(#1,Nov112010,13:34:43)
[GCC 4.1.220080704(RedHat4.1.2-48)] on linux2
Type"help","copyright","credits"or"license"for more information.
>>>
```

Type the following text at the Python prompt and press the Enter –

```
>>>print"Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and

continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension **.py**. Type the following source code in a test.py file –

```
print"Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows –

```
$ python test.py
```

This produces the following result –

```
Hello, Python!
```

Flask Framework:

Flask is a web application framework written in Python. Armin Ronacher, who leads an international group of Python enthusiasts named Pocco, develops it. Flask is based on Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects. Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.

The following table summarizes different http methods -

Table4.1 http methods.

Sr.No	Methods & Description
1	GET Sends data in unencrypted form to the server. Most common method.
2	HEAD Same as GET, but without response body

3	POST Used to send HTML form data to server. Data received by POST method is not cached by server.
4	PUT Replaces all current representations of the target resource with the uploaded content.
5	DELETE Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

In order to demonstrate the use of **POST** method in URL routing, first let us create an HTML form and use the **POST** method to send form data to a URL.

Save the following script as login.html

```
<html>
<body>
<form action="http://localhost:5000/login" method="post">
<p>Enter Name:</p>
<p><input type="text" name="nm"/></p>
<p><input type="submit" value="submit"/></p>
</form>
</body>
</html>
```

Now enter the following script in Python shell.

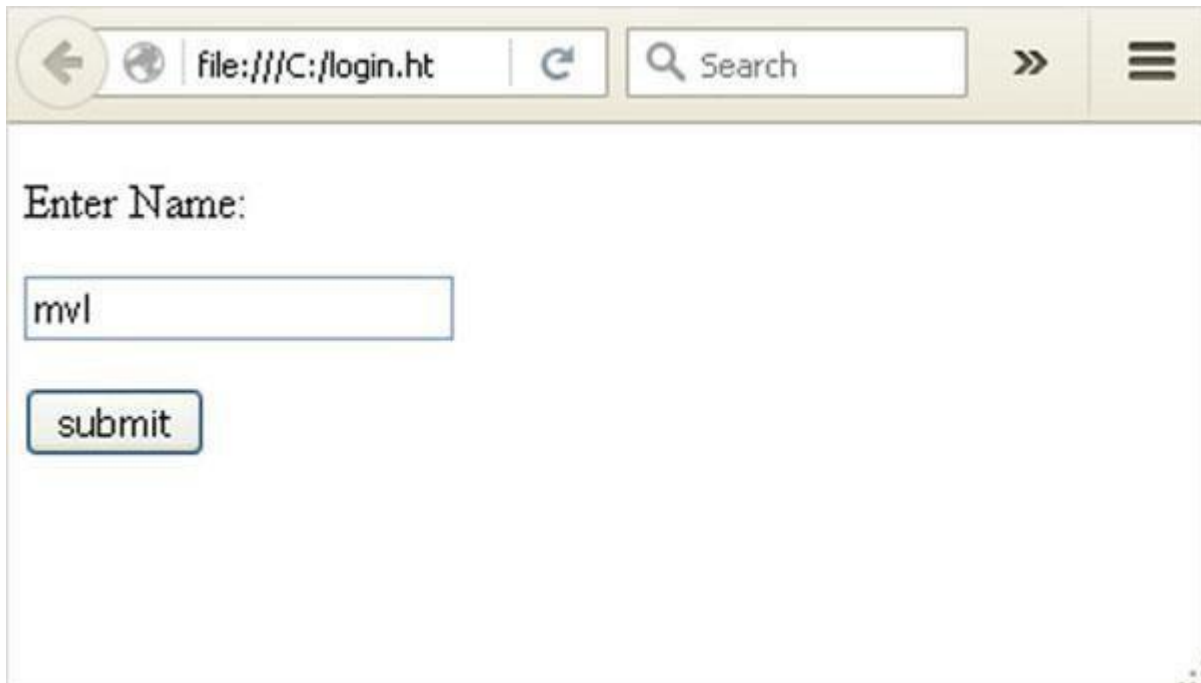
```
from flask import Flask, redirect, url_for, request
app = Flask(__name__)
```

```

@app.route('/success/<name>')
def success(name):
    return'welcome %s'% name
@app.route('/login',methods=['POST','GET'])
def login():
    ifrequest.method=='POST':
        user=request.form['nm']
        return redirect(url_for('success',name= user))
    else:
        user=request.args.get('nm')
        return redirect(url_for('success',name= user))
if __name__ == '__main__':
    app.run(debug =True)

```

After the development server starts running, open **login.html** in the browser, enter name in the text field and click **Submit**.



The screenshot shows a web browser window with the address bar displaying 'file:///C:/login.ht'. The main content area contains a form with the label 'Enter Name:'. Below the label is a text input field with the value 'mvl'. Below the input field is a button labeled 'submit'.

Fig 4.1 enter name

Form data is POSTed to the URL in action clause of form tag.

http://localhost/login is mapped to the **login()** function. Since the server has received data by **POST** method, value of 'nm' parameter obtained from the form data is obtained

by –

```
user = request.form['nm']
```

It is passed to **‘/success’** URL as variable part. The browser displays
a **welcome** message in the wi



Fig 4.2 welcome.

Change the method parameter to **‘GET’** in **login.html** and open it again in the browser. The data received on server is by the **GET** method. The value of **‘nm’** parameter is now obtained by –

```
User = request.args.get('nm')
```

Here, **args** is dictionary object containing a list of pairs of form parameter and its corresponding value. The value corresponding to **‘nm’** parameter is passed on to **‘/success’** URL as before.

4.4 MODULES:

- ❖ Admin Module
- ❖ Data Collection
- ❖ Train and Test
- ❖ Machine Learning Technique
- ❖ Detection of Fake User

4.4.1 MODULE DESCRIPTIONS:

Admin Module:

In the first module, we develop the Online Social Networking (OSN) system module. We build up the system with the feature of Online Social Networking System, Twitter. Where, this module is used for admin login with their authentication.

Data Collection:

We will be using a Python Library called Tweepy to connect to the Twitter API and collect the data. We download tweets containing certain key words, to incorporate the words or hash tags that contain relevant keyword related to fake users.

Some of the most important fields are:

- Text, which contains the text included in the tweet.
- Created at, this is a timestamp of when the tweet was created.
- user, which contains information about the user that created the tweet, like the username and user id.

Train and Test:

- ❖ We present the proposed framework for metadata features are extracted from available additional information regarding the tweets of a user, whereas content-based features aim to observe the message posting behavior of a user and the quality of the text that the user uses in posts.

Machine Learning Technique:

- ❖ The number of features, which are associated with tweet content, and the characteristics of users are recognized for the detection of spammers. These features are considered as the characteristics of machine learning process for categorizing users, i.e., to know whether they are spammers or not.
- ❖ In order to recognize the approach for detecting spammers on Twitter, the labeled collection in pre-classification of fake user and legitimate user has been done. Next, those steps are taken which are needed for the construction of labeled collection and acquired various desired properties.

- ❖ In other words, steps which are essential to be examined to develop the collection of users that can be labeled as fake user or legitimate user. At the end, user attributes are identified based on their behavior, e.g., who they interact with and what is the frequency of their interaction.
- ❖ In order to confirm this instinct, feature of users of the labeled collection has been checked. Two attribute sets are considered, i.e., content attributes and user behavior attributes, to differentiate one user from the other.

Detection of Fake User:

- ❖ In this module, we implement the collection of tweets with respect to trending topics on Twitter. After storing the tweets in a particular file format, the tweets are subsequently analyzed.
- ❖ Labeling of fake user is performed to check through all datasets that are available to detect the malignant.
- ❖ Feature extraction separates the characteristics construct based on the language model that uses language as a tool and helps in determining whether the user is fake or not.
- ❖ The classification of data set is performed by shortlisting the set of tweets that is described by the set of features provided to the classifier to instruct the model and to acquire the knowledge for spam detection.
- ❖ The fake user detection uses the classification technique to accept tweets as the input and classify the fake user and legitimate user.

4.4.2 REQUIREMENT ANALYSIS

Requirement analysis, also called requirement engineering, is the process of determining user expectations for a new modified product. It encompasses the tasks that determine the need for analysing, documenting, validating and managing software or system requirements. The requirements should be documentable, actionable, measurable, testable and traceable related to identified business needs or opportunities and define to a level of detail, sufficient for system design.

4.4.3 FUNCTIONAL REQUIREMENTS

It is a technical specification requirement for the software products. It is the first step

in the requirement analysis process which lists the requirements of particular software systems including functional, performance and security requirements. The function of the system depends mainly on the quality hardware used to run the software with given functionality.

Usability

It specifies how easy the system must be use. It is easy to ask queries in any format which is short or long, porter stemming algorithm stimulates the desired response for user.

Robustness

It refers to a program that performs well not only under ordinary conditions but also under unusual conditions. It is the ability of the user to cope with errors for irrelevant queries during execution.

Security

The state of providing protected access to resource is security. The system provides good security and unauthorized users cannot access the system there by providing high security.

Reliability

It is the probability of how often the software fails. The measurement is often expressed in MTBF (Mean Time Between Failures). The requirement is needed in order to ensure that the processes work correctly and completely without being aborted. It can handle any load and survive and survive and even capable of working around any failure.

Compatibility

It is supported by version above all web browsers. Using any web servers like localhost makes the system real-time experience.

Flexibility

The flexibility of the project is provided in such a way that is has the ability to run on different environments being executed by different users.

Safety

Safety is a measure taken to prevent trouble. Every query is processed in a secured manner without letting others to know one's personal information.

4.4.4 NON- FUNCTIONAL REQUIREMENTS

Portability

It is the usability of the same software in different environments. The project can be run in any operating system.

Performance

These requirements determine the resources required, time interval, throughput and everything that deals with the performance of the system.

Accuracy

The result of the requesting query is very accurate and high speed of retrieving information. The degree of security provided by the system is high and effective.

Maintainability

Project is simple as further updates can be easily done without affecting its stability. Maintainability basically defines that how easy it is to maintain the system. It means that how easy it is to maintain the system, analyse, change and test the application. Maintainability of this project is simple as further updates can be easily done without affecting its stability.

4.5 ALGORITHMSUSED:

4.5.1 The random forest algorithm

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result.

One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning.

Important hyperparameters

The hyperparameters in random forest are either used to increase the predictive power of the model or to make the model faster. Let's look at the hyperparameters of sklearn's built-in random forest function.

1. Increasing the predictive power

Firstly, there is the `n_estimators` hyperparameter, which is just the number of trees the algorithm builds before taking the maximum voting or taking the averages of predictions. In general, a higher number of trees increases the performance and makes the predictions more stable, but it also slows down the computation.

Another important hyperparameter is `max_features`, which is the maximum number of features random forest considers to split a node. Sklearn provides several options, all described in the documentation.

The last important hyperparameter is `min_sample_leaf`. This determines the minimum number of leafs required to split an internal node.

2. Increasing the model's speed

The `n_jobs` hyperparameter tells the engine how many processors it is allowed to use. If it has a value of one, it can only use one processor. A value of “-1” means that there is no limit.

The `random_state` hyperparameter makes the model's output replicable. The model will always produce the same results when it has a definite value of random state and if it has been given the same hyperparameters and the same training data.

Lastly, there is the oob score (also called oob sampling), which is a random forest cross-validation method. In this sampling, about one-third of the data is not used to train the model and can be used to evaluate its performance. These samples are called the out-of-bag samples. It's very similar to the leave-one-out-cross-validation method, but almost no additional computational burden goes along with it.

Advantages and disadvantages of the random forest algorithm

One of the biggest advantages of random forest is its versatility. It can be used for both regression and classification tasks, and it's also easy to view the relative importance it assigns to the input features.

Random forest is also a very handy algorithm because the default hyperparameters it uses often produce a good prediction result. Understanding the hyperparameters is pretty straightforward, and there's also not that many of them.

One of the biggest problems in machine learning is overfitting, but most of the time this won't happen thanks to the random forest classifier. If there are enough trees in the forest, the classifier won't overfit the model.

The main limitation of random forest is that a large number of trees can make the algorithm too slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model. In most real-world applications, the random forest algorithm is fast enough but there can certainly be situations where run-time performance is important and other approaches would be preferred.

And, of course, random forest is a predictive modeling tool and not a descriptive tool, meaning if you're looking for a description of the relationships in your data, other approaches would be better.

4.5.2 Decision Tree Algorithm

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

Classification is a two-step process, learning step and prediction step, in machine learning. In the learning step, the model is developed based on given training data. In the prediction step, the model is used to predict the response for given data. Decision Tree is one of the easiest and popular classification algorithms to understand and interpret.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Types of Decision Trees

Types of decision trees are based on the type of target variable we have. It can be of two types:

1. **Categorical Variable Decision Tree:** Decision Tree which has a categorical target variable then it called a Categorical variable decision tree.
2. **Continuous Variable Decision Tree:** Decision Tree has a continuous target variable then it is called Continuous Variable Decision Tree.

Example:- Let's say we have a problem to predict whether a customer will pay his renewal premium with an insurance company (yes/ no). Here we know that the income of customers is a significant variable but the insurance company does not have income details for all customers. Now, as we know this is an important variable, then we can build a decision tree to predict customer income based on occupation, product, and various other variables. In this case, we are predicting values for the continuous variables.

Important Terminology related to Decision Trees

1. **Root Node:** It represents the entire population or sample and this further gets divided into two or more homogeneous sets.
2. **Splitting:** It is a process of dividing a node into two or more sub-nodes.
3. **Decision Node:** When a sub-node splits into further sub-nodes, then it is called the decision node.
4. **Leaf / Terminal Node:** Nodes do not split is called Leaf or Terminal node.
5. **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say the opposite process of splitting.
6. **Branch / Sub-Tree:** A subsection of the entire tree is called branch or sub-tree.
7. **Parent and Child Node:** A node, which is divided into sub-nodes is called a parent node of sub-nodes whereas sub-nodes are the child of a parent node.

CHAPTER-5

RESULTS AND DISCUSSION, PERFORMANCE ANALYSIS

5.1 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

5.2 SYSTEM STUDY

5.2.1 Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

1. Economical feasibility
2. Technical feasibility
3. Social feasibility

Economical feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

5.3 Data Flow Diagram:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

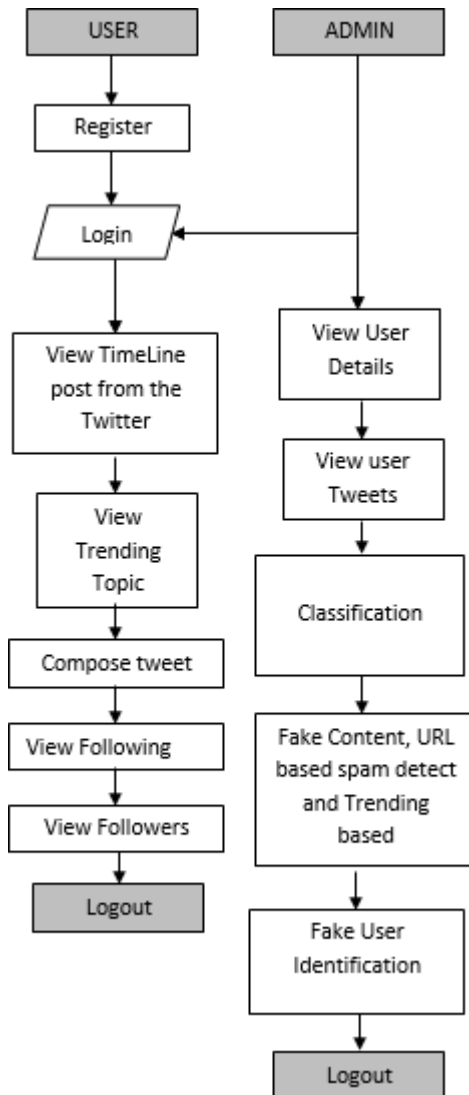


Fig 5.1 data flow diagram

5.4 UML diagrams

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling

Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

5.4.1 Goals:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

5.5 Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

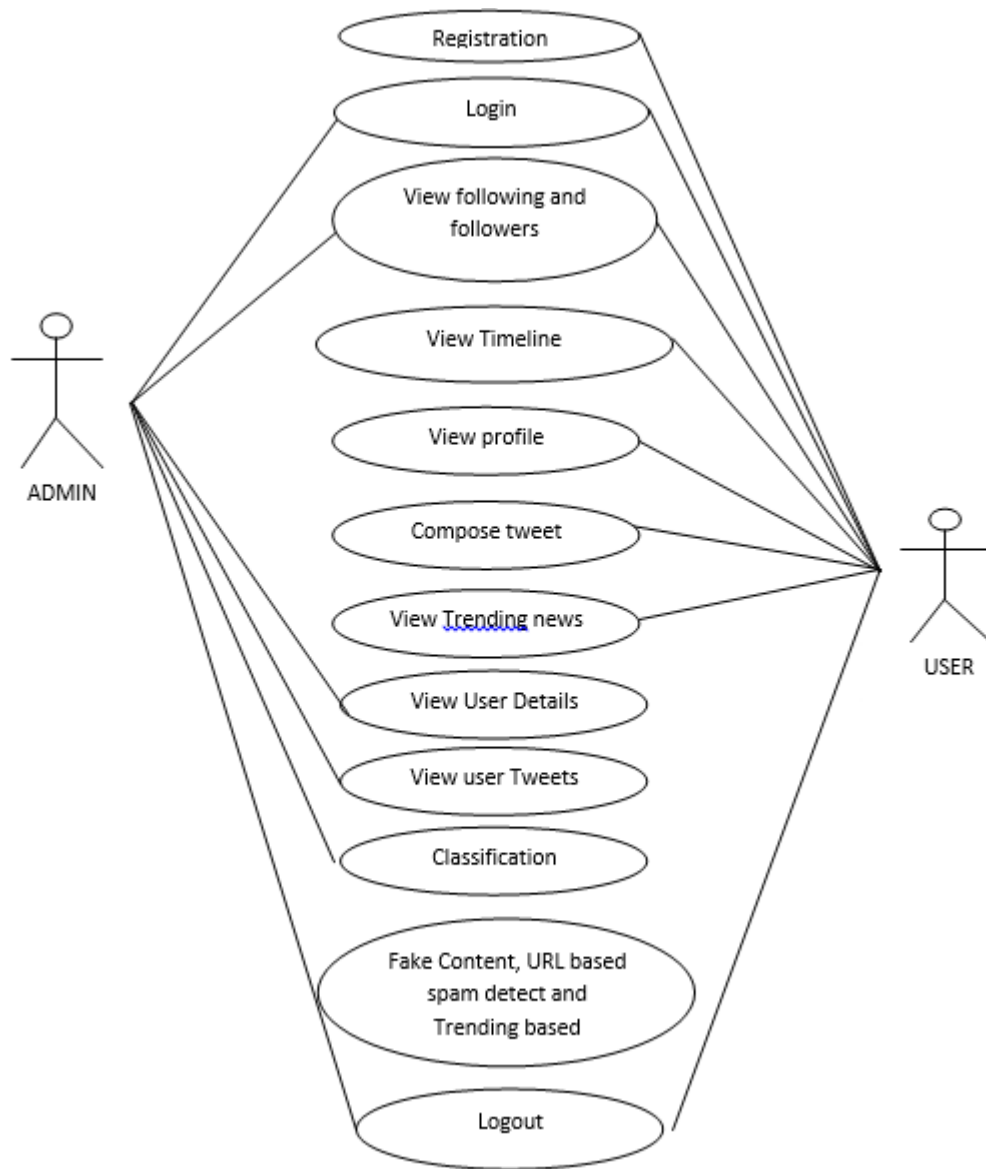


Fig 5.2 Use Case Diagram

5.5 Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

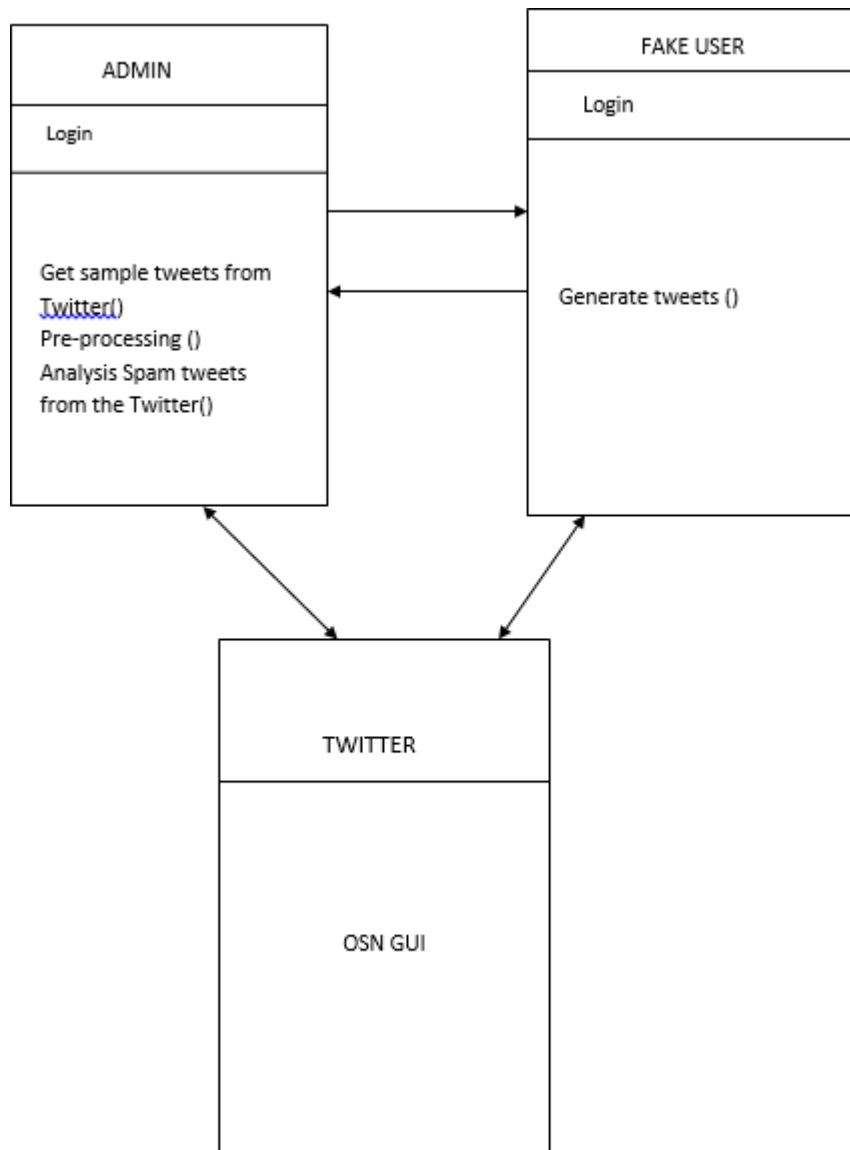


Fig 5.3 Class Diagram

5.6 Sequence Diagram:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

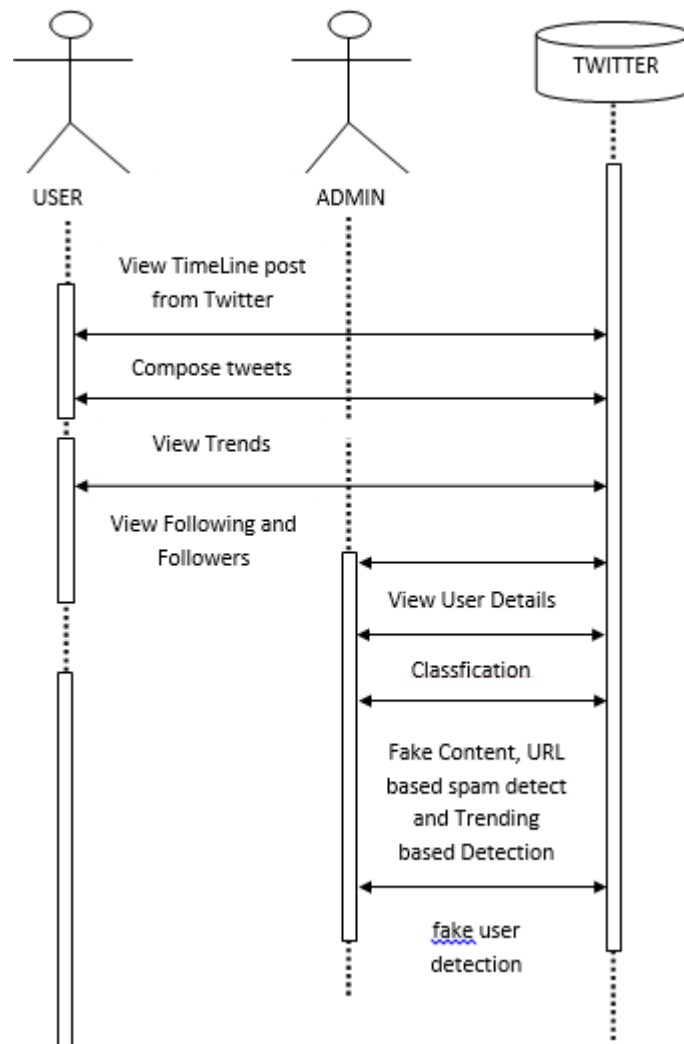


Fig 5.4 Sequence Diagram

5.7 Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

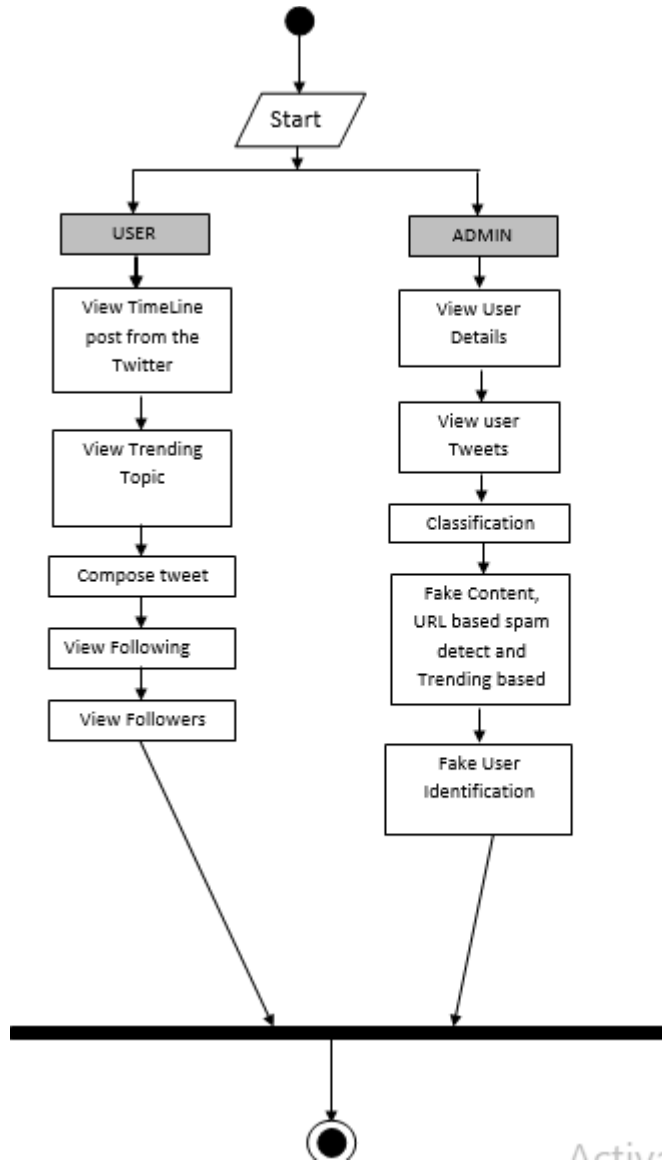


Fig 5.7 Activity Diagram

5.8 System Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

5.9 TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test Strategy And Approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

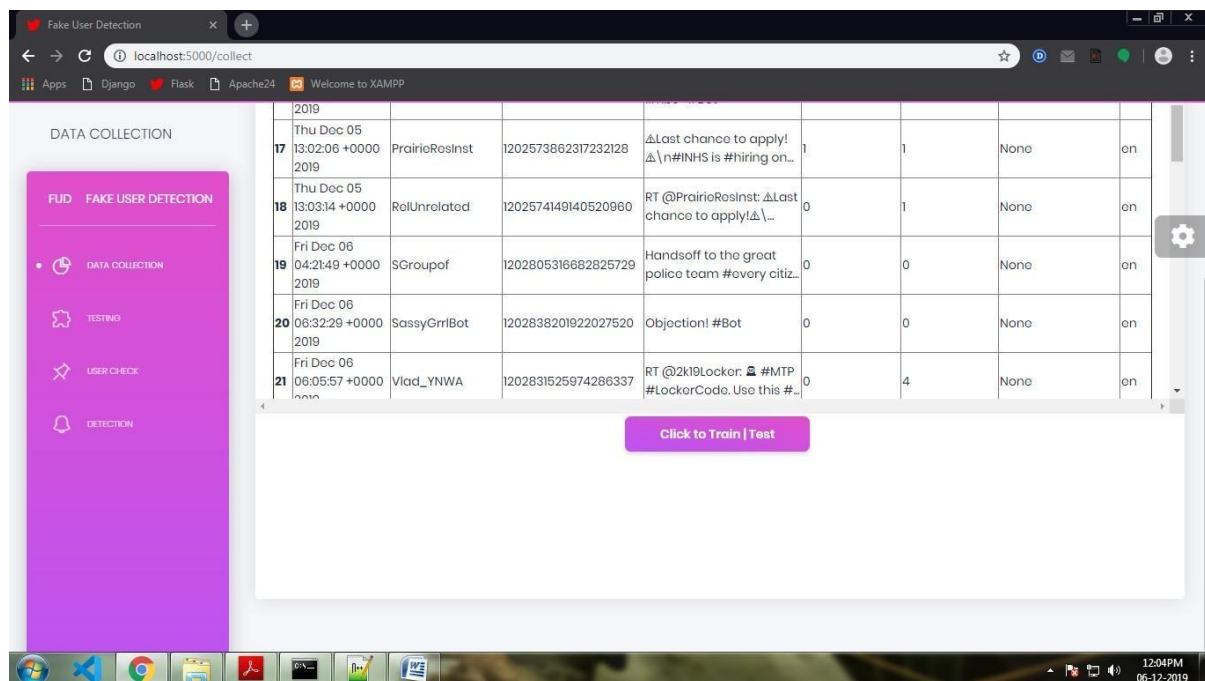
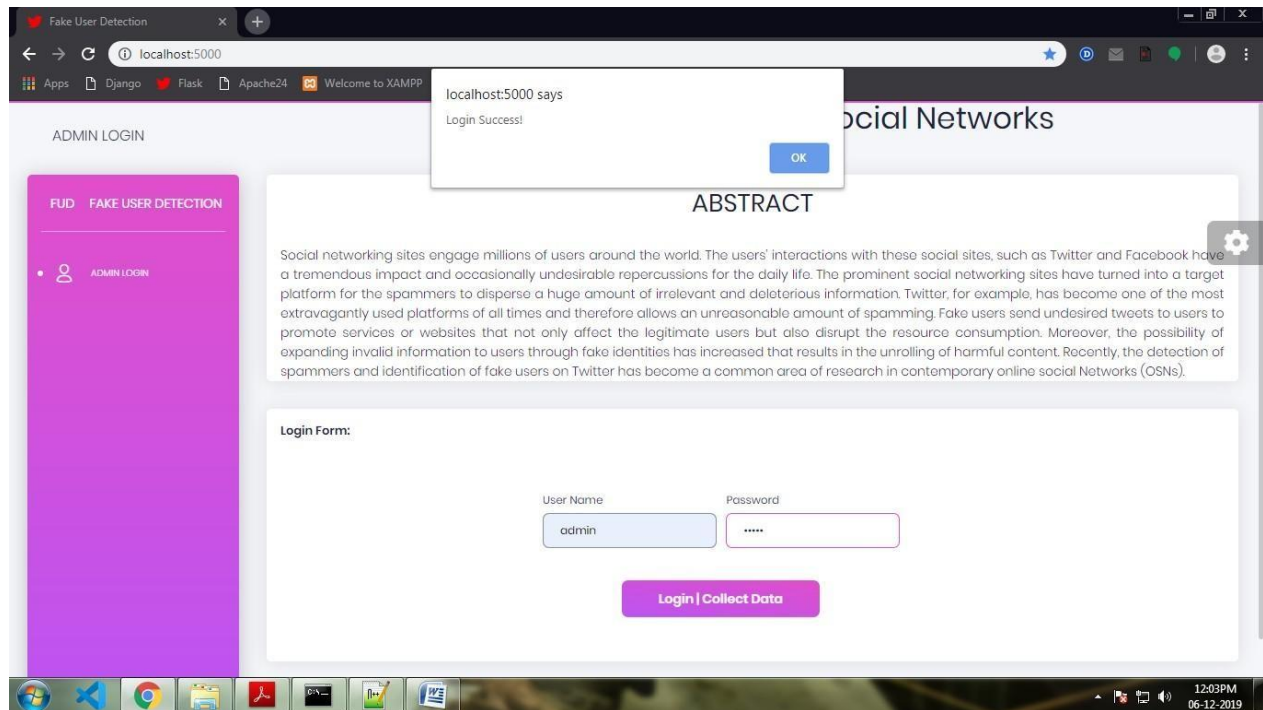
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

5.10 Output Screen Shots



127.0.0.1:5000/check

USER CHECK

FUD FAKE USER DETECTION

DATA COLLECTION

TESTING

USER CHECK

DETECTION

User Check



Fake User Detection

Enter a username...

id	id_str	Name	Username	Followers_count	Listed_count	Friends_count	Favorites_count	Created_at	Verified	Default_profile
0	752247917457641473	752247917457641473	KoohejiCoohejiCoo	86	0	5	1	2016-07-10 21:07:34	False	False

Next | Detect

6:00 PM 2/25/2021

127.0.0.1:5000/detect

USER CHECK

FUD FAKE USER DETECTION

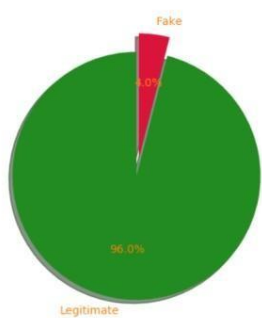
DATA COLLECTION

TESTING

USER CHECK

DETECTION

Followers_count	Listed_count	Friends_count	Favorites_count	Verified	Default_profile	Default_profile_image	Location	Statuses_count	URL	age_in_days
50	1	469	9257	False	False	False	True	2494	False	4283



96.0% Legitimate

4.0% Fake

✓ Legitimate User

6:01 PM 2/25/2021

CHAPTER-6

SUMMARYANDCONCLUSIONS

6.1 CONCLUSION

In this paper, we performed a review of techniques used for detecting spammers on Twitter. In addition, we also presented taxonomy of Twitter spam detection approaches and categorized them as fake content detection, URL based spam detection, spam detection in trending topics, and fake user detection techniques. We also compared the presented techniques based on several features, such as user features, content features, graph features, structure features, and time features. Moreover, the techniques were also compared in terms of their specified goals and datasets used. It is anticipated that the presented review will help researchers find the information on state-of-the-art Twitter spam detection techniques in a consolidated form. Despite the development of efficient and effective approaches for the spam detection and fake user identification on Twitter [34], there are still certain open areas that require considerable attention by the researchers. The issues are briefly highlighted as under: False news identification on social media networks is an issue that needs to be explored because of the serious repercussions of such news at individual as well as collective level [25]. Another associated topic that is worth investigating is the identification of rumor sources on social media. Although a few studies based on statistical method shave already been conducted to detect the sources of rumors, more sophisticated approaches, e.g., social network based approaches, can be applied because of their proven effectiveness.

6.2 SUMMARY

Online review systems play an important role in affecting consumers' behaviors and decision making, attracting many spammers to insert fake reviews to manipulate review content and ratings. To increase utility and improve user experience, some online review systems allow users to form social relationships between each other and encourage their interactions. In this paper, we aim at providing an efficient and effective method to identify review spammers by incorporating social relations based on two assumptions that people are more likely to consider reviews from those connected with them as trustworthy, and review spammers are less likely to maintain a large relationship network with normal users. The contributions of this paper are two-fold: (1)

We elaborate how social relationships can be incorporated into review rating prediction and propose a trust-based rating prediction model using proximity as trust weight, and (2) We design a trust-aware detection model based on rating variance which iteratively calculates user-specific overall trustworthiness scores as the indicator for spamicity. Experiments on the dataset collected from Yelp.com show that the proposed trust-based prediction achieves a higher accuracy than standard CF method, and there exists a strong correlation between social relationships and the overall trustworthiness scores

REFERENCES

- [1] C. Chen, S. Wen, J. Zhang, Y. Xiang, J. Oliver, A. Alelaiwi, and M. M. Hassan, "Investigating the deceptive information in Twitter spam," *Future Gener. Comput. Syst.*, vol. 72, pp. 319-326, Jul. 2017.
- [2] I. David, O. S. Siordia, and D. Moctezuma, "Features combination for the detection of malicious Twitter accounts," in *Proc. IEEE Int. Autumn Meeting Power, Electron. Comput. (ROPEC)*, Nov. 2016, pp. 1-6.
- [3] M. Babcock, R. A. V. Cox, and S. Kumar, "Diffusion of pro- and anti-false information tweets: The black panther movie case," *Comput. Math. Org. Theory*, vol. 25, no. 1, pp. 72-84, Mar. 2019.
- [4] S. Keretna, A. Hossny, and D. Creighton, "Recognising user identity in Twitter social networks via text mining," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2013, pp. 3079-3082.
- [5] C. Meda, F. Bisio, P. Gastaldo, and R. Zunino, "A machine learning approach for Twitter spammers detection," in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2014, pp. 1-6.
- [6] W. Chen, C. K. Yeo, C. T. Lau, and B. S. Lee, "Real-time Twitter content polluter detection based on direct features," in *Proc. 2nd Int. Conf. Inf. Sci. Secur. (ICISS)*, Dec. 2015, pp. 1-4.
- [7] H. Shen and X. Liu, "Detecting spammers on Twitter based on content and social interaction," in *Proc. Int. Conf. Netw. Inf. Syst. Comput.*, pp. 413-417, Jan. 2015.
- [8] G. Jain, M. Sharma, and B. Agarwal, "Spam detection in social media using convolutional and long short term memory neural network," *Ann. Math. Artif. Intell.*, vol. 85, no. 1, pp. 21-44, Jan. 2019.
- [9] M. Washha, A. Qaroush, M. Mezghani, and F. Sedes, "A topic-based hidden Markov model for real-time spam tweets filtering," *Procedia Comput. Sci.*, vol. 112, pp. 833-843, Jan. 2017.
- [10] F. Pierri and S. Ceri, "False news on social media: A data-driven survey," 2019, arXiv:1902.07539. [Online]. Available: <https://arxiv.org/abs/1902.07539>
- [11] S. Sadiq, Y. Yan, A. Taylor, M.-L. Shyu, S.-C. Chen, and D. Feaster, "AAFA: Associative affinity factor analysis for bot detection and stance classification in Twitter," in *Proc. IEEE Int. Conf. Inf. Reuse Integr. (IRI)*, Aug. 2017, pp. 356-365.

[12] M. U. S. Khan, M. Ali, A. Abbas, S. U. Khan, and A. Y. Zomaya, "Segregating spammers and unsolicited bloggers from genuine experts on Twitter," IEEE Trans. Dependable Secure Comput., vol. 15, no. 4, pp. 551-560, Jul./Aug. 2018.

A. PLAGIARISM REPORT

Plag- SPAMMER DETECTION AND FAKE USER
IDENTIFICATION ON SOCIAL NETWORKS.docx

ORIGINALITY REPORT

16%

SIMILARITY INDEX

9%

INTERNET SOURCES

12%

PUBLICATIONS

8%

STUDENT PAPERS

PRIMARY SOURCES

1

Cody Buntain, Jennifer Golbeck. "Automatically Identifying Fake News in Popular Twitter Threads", 2017 IEEE International Conference on Smart Cloud (SmartCloud), 2017

Publication

5%

2

web-tools.uts.edu.au

Internet Source

4%

3

Submitted to University of Wales Institute, Cardiff

Student Paper

www.jpinfotech.org

2%

4

Internet Source

2%

5

nrl.northumbria.ac.uk

Internet Source

1%

6

Zhijie Zhang, Rui Hou, Jin Yang. "Detection of Social Network Spam Based on Improved Extreme learning Machine", IEEE Access, 2020

Publication

1%

7	Submitted to Bridgepoint Education Student Paper	1 %
8	worldscientific.com Internet Source	1 %
9	www.ijrte.org Internet Source	<1 %
10	Thayakorn Dangkesee, Sutheera Puntheeranurak. "Adaptive Classification for Spam Detection on Twitter with Specific Data", 2017 21st International Computer Science and Engineering Conference (ICSEC), 2017 Publication	<1 %
11	Xu Chen, Yuyu Yuan, Lilei Lu, Jincui Yang. "A Multidimensional Trust Evaluation Framework for Online Social Networks Based on Machine Learning", IEEE Access, 2019 Publication	<1 %

B. JOURNAL REPORT

SPAM REVIEW DETECTION USING THE LINGUISTIC AND SPAMMER BEHAVIORAL METHODS

KURUPATI SAITEJA,¹,MADDINENI PAVAN KUMAR,²,PREM JACOB,³

1.B.E. COMPUTER SCIENCE AND ENGINEERING,SATHYABAMA INSTITUTE OF
SCIENCE AND TECHNOLOGY,CHENNAI,INDIA

2.B.E. COMPUTER SCIENCE AND ENGINEERING,SATHYABAMA INSTITUTE OF
SCIENCE AND TECHNOLOGY,CHENNAI,INDIA

3.M.E.,Ph.D.,SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY,CHENNAI,
INDIA

ABSTRACT:

Online review structures expect a critical activity in impacting purchasers' practices and dynamic, attracting various spammers to install fake studies to control study substance and assessments. To grow utility and improve customer experience, some online review structures license customers to shape social associations between each other and bolster their interchanges. At this moment, target giving a profitable and amazing system to recognize review spammers by combining social relations subject to two suppositions that people will undoubtedly consider overviews from those related with them as dependable, and review spammers are less disposed to keep up a tremendous relationship arrange with conventional customers. The duties of this paper are two-wrinkle: (1) We clarify how social associations can be combined into review rating desire and propose a trust-based rating conjecture model using closeness as trust weight, and (2) We structure a trust-careful area model reliant on rating vacillation which iteratively

determines customer unequivocal all in all constancy scores as the pointer for spamicity. Tests on the dataset assembled from Yelp.com show that the proposed trust-based estimate achieves a higher exactness than standard CF procedure, and there exists a strong association between social associations and the general constancy scores.

Introduction:

Wikipedia portrays a relational association organization as a help which "revolves around the structure and checking of online casual networks for systems of people who offer premiums and activities, or who are excited about researching the premiums and activities of others, and which requires the usage of programming."

A report appropriated by OCLC gives the going with significance of long range casual correspondence goals: "Locales in a general sense expected to support association between customers who offer interests, mindsets and activities, for instance, Facebook, Mixi and MySpace."

Relational associations can give an extent of points of interest to people from an affiliation:

Support for learning: Social frameworks can improve easygoing learning and reinforce social relationship inside get-togethers of understudies and with those drew in with the assistance of learning.

Support for people from an affiliation: Social frameworks can be used by all people from an affiliation, and not just those connected with working with understudies. Casual associations can help the improvement of systems of preparing.

Interfacing with others: Passive usage of casual networks can give significant business understanding and analysis on institutional organizations (regardless of the way this may offer climb to moral concerns).

LITERATURE SURVEY

1) Statistical features based consistent area of glided Twitter spam

Makers: C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min

Twitter spam has become an essential issue

nowadays. Late works base on applying AI frameworks for Twitter spam acknowledgment, which use the truthful features of tweets. In our named tweets instructive assortment, regardless, we see that the authentic properties of spam tweets change after some time, and hence, the presentation of existing AI based classifiers reduces. This issue is suggested as "Twitter Spam Drift". In order to deal with this issue, we initially complete a significant assessment on the genuine features of one million spam tweets and one million non-spam tweets, and a short time later propose a novel Lfun plan. The proposed arrangement can discover "changed" spam tweets from unlabeled tweets and merge them into classifier's planning method. Different investigations are performed to survey the proposed plot. The results show that our proposed Lfun plan would altogether be able to improve the spam area precision in genuine circumstances.

2) Automatically perceiving fake news in notable Twitter strings

Makers: C. Buntain and J. Golbeck

Information quality in online life is a verifiably critical issue, yet web-scale data ruins masters' ability to assess and address a huge piece of the misguided substance, or "fake news," present in these stages. This paper develops a system for robotizing fake news distinguishing proof on Twitter by making sense of how to predict precision examinations in two acceptability focused Twitter datasets: CREDBANK, an openly bolstered dataset of exactness assessments for events in Twitter, and PHEME, a dataset of potential gossipy goodies in Twitter and journalistic evaluations of their correct nesses. We apply this procedure to twitter content sourced from Buzz Feed's fake news dataset and show models arranged against openly bolstered

workers outmaneuver models subject to journalists' assessment and models arranged on a pooled dataset of both freely upheld pros and feature writers. All of the three datasets, balanced into a uniform setup, are moreover openly available. A component examination by then recognizes features that are commonly judicious for freely upheld and journalistic accuracy evaluations, outcomes of which are unsurprising with prior work. We close with a discussion separating precision and legitimacy and why models of non-masters outmaneuver models of journalists for fake news revelation in Twitter.

3) A presentation appraisal of AI based spouting spam tweets acknowledgment

Makers: C. Chen, J. Zhang, Y. Xie, Y. Xiang, W. Zhou, M. M. Hassan, A. AlElaiwi, and M. Alrubaian

The universality of Twitter pulls in a consistently expanding number of spammers. Spammers send unwanted tweets to twitter customers to propel destinations or organizations, which are destructive to customary customers. In order to stop spammers, researchers have proposed different frameworks. The point of convergence generally works is on the utilization of AI methods into Twitter spam distinguishing proof. Nevertheless, tweets are recuperated in a spilling way, and Twitter gives the Streaming API to architects and masters to find a workable pace dynamically. There doesn't have an introduction evaluation of existing AI based spouting spam recognizable proof procedures. At this moment, vanquished any prevention by means of doing an introduction appraisal, which was from three remarkable pieces of data, feature, and model. A significant ground-truth of in excess of 600 million open tweets was made by using a business URL-based security contraption. For consistent spam area, we further evacuated 12 lightweight features for tweet depiction. Spam revelation was then changed to a matched portrayal issue in the component space and can be comprehended by standard AI counts. We evaluated the impact of different components to the spam area execution, which included spam to nonspam extent, feature discretization, planning data size, data examining, time-related data, and AI computations. The results show the spouting spam tweet acknowledgment is

so far a significant test and a fiery recognizable proof technique ought to think about the three pieces of data, feature, and model.

4) A model-based system for recognizing spammers in relational associations

Makers: F. Fathaliani and M. Bouguessa

At the present time, see the endeavor of recognizing spammers in casual associations from a mix showing perspective, considering which we devise a principled independent approach to manage distinguish spammers. In our strategy, we at first address each customer of the casual network with a component vector that reflects its lead and joint efforts with various individuals. Next, considering the evaluated customers incorporate vectors, we propose a true framework that uses the Dirichlet scattering to perceive spammers. The proposed approach can normally isolate among spammers and genuine customers, while existing independent systems require human mediation in order to set easygoing limit parameters to perceive spammers. Furthermore, our system is general as in it might be applied to different online social regions. To display the propriety of the proposed system, we drove researches certifiable data removed from Instagram and Twitter.

5) Spam area of Twitter traffic: A structure subject to unpredictable woods and non-uniform component analyzing

Makers: C. Meda, E. Ragusa, C. Gianoglio, R. Zunino, A. Ottaviano, E. Scillia, and R. Surlinelli

Law Enforcement Agencies spread a basic activity in the assessment of open data and need convincing strategies to channel hazardous information. In a real circumstance, Law Enforcement Agencies research Social Networks, for instance Twitter, watching events and profiling accounts. Sadly, between the colossal proportion of web customers, there are people that usage microblogs for bothering others or spreading malevolent substance. Customers' portrayal and spammers' ID is a useful framework for ease Twitter traffic from uninformative substance. This work proposes a structure that abuses a non-uniform component testing inside a diminish box Machine Learning System, using a variety of the Random Forests Algorithm to recognize spammers inside Twitter

traffic. Tests are made on a notable Twitter dataset and on another dataset of Twitter customers. The new given Twitter dataset is included customers named as spammers or certified customers, depicted by 54 features. Preliminary outcomes show the amplex of improved part testing system

EXISTING SYSTEM

□ Tingminet al. give an investigation of new methodologies and strategies to perceive Twitter spam disclosure. The above review shows a comparable examination of the present systems.

□ On the other hand, S. J. Somanet. Al. driven an examination on different practices showed by spammers on Twitter relational association. The assessment furthermore gives a composing review that sees the nearness of spammers on Twitter casual association.

□ Despite all the present examinations, there is so far an opening in the present composition. In like manner, to vanquish any obstruction, we review top tier in the spammer area and fake customer recognizing evidence on Twitter

DISADVANTAGES OF EXISTING SYSTEM

- No capable strategies used.
- No consistent data's used.
- more complex

PROPOSED SYSTEM:

□ The purpose of this paper is to recognize fake customer revelation on Twitter and to show a framework by masterminding these techniques into a couple of arrangements. For portrayal, we have perceived four techniques for declaring spammers that can be helpful in recognizing fake characters of customers. Spammers can be perceived reliant on: (I) fake substance, (ii) URL based spam acknowledgment, (iii) distinguishing spam in floating focuses, and (iv) fake customer ID.

ADVANTAGES OF PROPOSED SYSTEM

- This study joins AI strategy proposed using progressing datasets and with different characteristics and accomplishments.
- The proposed structure is more fruitful and correct than other existing systems.
- Tested with persistent data's.

SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

- System: Pentium Dual Core.
- Hard Disk: 120 GB.
- Monitor : 15'' LED
- Input Devices : Keyboard, Mouse
- Ram: 4 GB.

SOFTWARE REQUIREMENTS:

- Operating system: Windows 7/10.
- Coding Language :Python
- Tool : Pi-champ
- Database : MYSQL

SOFTWARE ENVIRONMENT

Python

MODULES:

- ❖ Admin Module
- ❖ Data Collection
- ❖ Train and Test
- ❖ Machine Learning Technique
- ❖ Detection of Fake User

MODULE DESCRIPTIONS:

Admin Module:

In the first part, and to develop an online social networking system (OSN) module. We are building the system with functionality of the online social networking system, in one. Where, as a module with its own administrator account is used in authentication.

Data Collection:

We'll use the Python Tweepy library pickup that he called her and gave her to twitter to join the API. We upload tweets that contain certain keywords or

terms that incorporate bogus relay tags which are relevant keywords to users.

Some of the most important areas are:

- Text, which contains the text included in the Tweet.
- Create at, that's what's in the timekeeping, is the Tweet in which it was created.
- Football temperature as the mourning valley that contains information about the user created the tweet, so the username.

Train and Test:

- ❖ We present the proposed framework for metadata features are extracted from the information available on a user's tweets with content-based features aimed at keeping the message displaying user behavior and text quality as experience user extension.

Machine Learning Technique:

- ❖ The number of features, which are associated with tweet content, and the characteristics of users are recognized for the detection of spammers. These features are considered as the characteristics of machine learning process for categorizing users, i.e., to know whether they are spammers or not.
- ❖ In order to recognize the approach for detecting spammers on Twitter, the labelled collection in pre-classification of fake user and legitimate user has been done. Next, those steps are taken which are needed for the construction of labeled collection and acquired various desired properties.
- ❖ In other words, steps which are essential to be examined to develop the collection of users that can be labelled as fake user or legitimate user. At the end, user attributes

are identified based on their behavior, e.g., who they interact with and what is the frequency of their interaction.

- ❖ In order to confirm this instinct, features of users of the labelled collection has been checked. Two attribute sets are considered, i.e., content attributes and user behavior attributes, to differentiate one user from the other.

DATAFLOW DIAGRAM

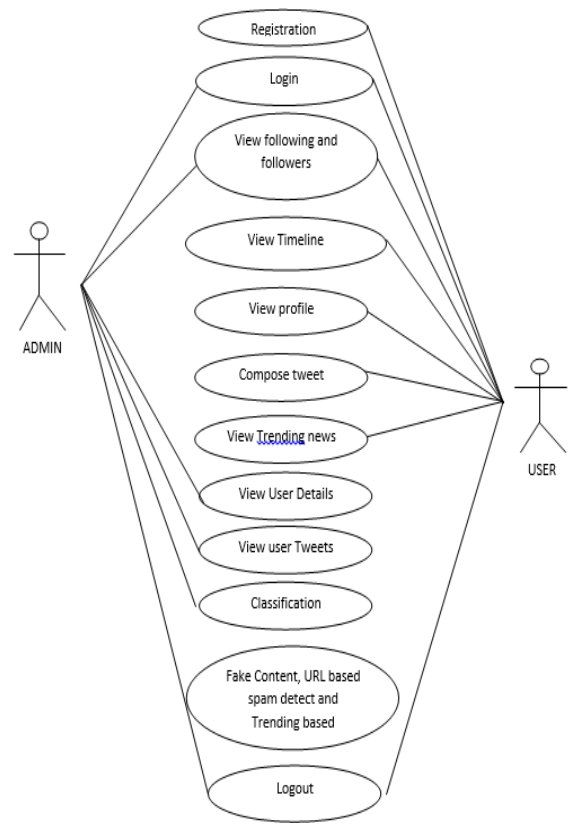
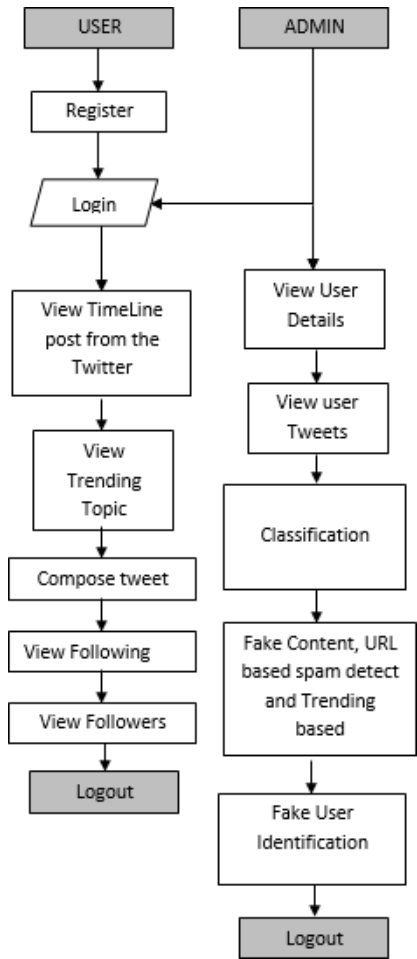
Detection of Fake User:

- ❖ In this module, we implement the collection of tweets with respect to trending topics on Twitter. After storing the tweets in a particular file format, the tweets are subsequently analyzed.
- ❖ Labelling of fake user is performed to check through all datasets that are available to detect the malignant.
- ❖ Feature extraction separates the characteristics construct based on the language model that uses language as a tool and helps in determining whether the user is fake or not.
- ❖ The classification of data set is performed by shortlisting the set of tweets that is described by the set of features provided to the classifier

USECASE DIAGRAM

to instruct the model and to acquire the knowledge for spam detection.

- ❖ The fake user detection uses the classification technique to accept tweets as the input and classify the fake user and legitimate user.



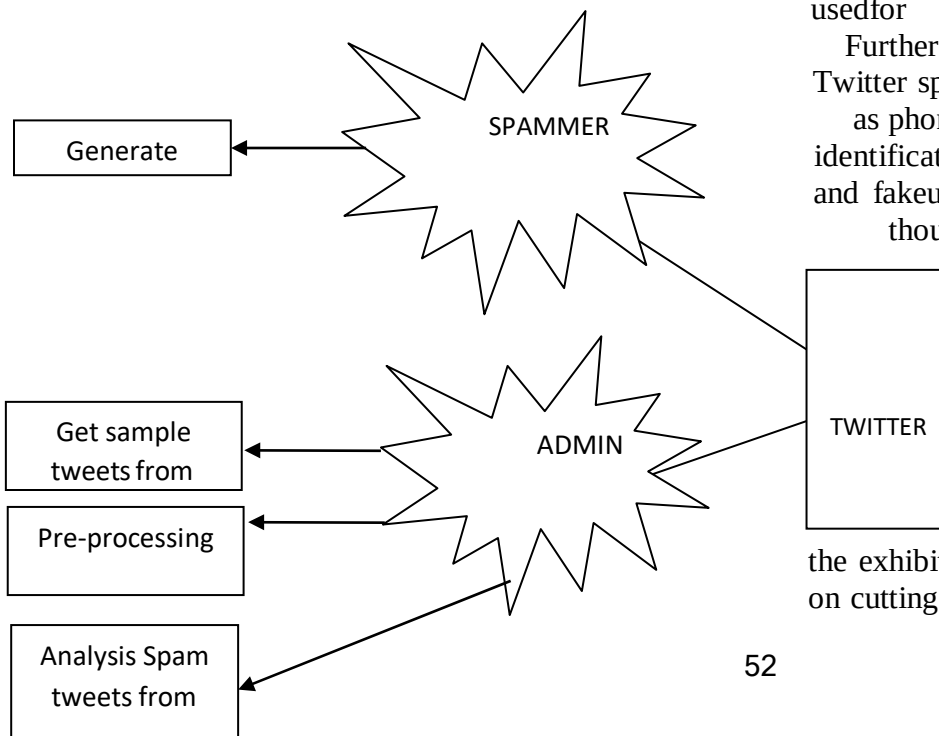
SYSTEM ARCHITECTURE

CONCLUSION:

Right now, played out a survey of procedures used for distinguishing spammers on Twitter.

Furthermore, we likewise presented taxonomy of Twitter spam location approaches and ordered them as phony substance recognition, URL based spam identification, spam discovery in slanting subjects, and fake user discovery strategies. We additionally thought about the exhibited procedures

dependent on a few highlights, for example, client features, content highlights, chart highlights, structure highlights, and time features. Additionally, the strategies were likewise thought about in terms of their predefined objectives and datasets utilized. It is anticipated that the exhibited audit will help analysts find the data on cutting edge Twitter spam location systems in a



merged form. Despite the improvement of proficient and powerful methodologies for the spam recognition and phony client identification on Twitter [34], there are as yet certain open territories that require considerable consideration by the scientists.

The issues are briefly featured as under: False news recognizable proof via web-based networking media systems is an issue that should be investigated in light of the serious repercussions of such news at individual just as collective level [25]. Another related subject that is worth investigating is the ID of gossip sources on social media. Albeit a couple of concentrates dependent on factual methods have as of now been directed to identify the sources of rumors, increasingly modern methodologies, e.g., social networkbased approaches, can be applied as a result of their proven effectiveness.

REFERENCES

1. B. Erşahin, A. Aktaş, D. Kilinç, and C. Akyol, "Twitter fake account detection," in Proc. Int. Conf. Comput. Sci. Eng. (UBMK), Oct. 2017, pp. 388_392.
2. F. Benevenuto, G. Magno, T. Rodrigues, and V. Almeida, "Detecting spammers on Twitter," in Proc. Collaboration, Electron. Messaging, Anti- Abuse Spam Conf. (CEAS), vol. 6, Jul. 2010, p. 12.
3. S. Gharge, and M. Chavan, "An integrated approach for malicious tweets detection using NLP," in Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT), Mar. 2017, pp. 435_438.
4. T. Wu, S. Wen, Y. Xiang, and W. Zhou, "Twitter spam detection: Survey of new approaches and comparative study," Comput. Secur., vol. 76, pp. 265_284, Jul. 2018.
5. S. J. Soman, "A survey on behaviors exhibited by spammers in popular social media networks," in Proc. Int. Conf. Circuit, Power Comput. Technol. (ICCPCT), Mar. 2016, pp. 1_6.
6. A. Gupta, H. Lamba, and P. Kumaraguru, "1.00 per RT #BostonMarathon #prayforboston: Analyzing fake content on Twitter," in Proc. eCrime Researchers Summit (eCRS), 2013, pp. 1_12.
7. F. Concone, A. De Paola, G. Lo Re, and M. Morana, "Twitter analysis for real-time malware discovery," in Proc. AEIT Int. Annu. Conf., Sep. 2017, pp. 1_6.
8. N. Eshraqi, M. Jalali, and M. H. Moattar, "Detecting spam tweets in Twitter using a data stream clustering algorithm," in Proc. Int. Congr. Technol., Commun. Knowl. (ICTCK), Nov. 2015, pp. 347_351.
9. C. Chen, Y. Wang, J. Zhang, Y. Xiang, W. Zhou, and G. Min, "Statistical features-based real-time detection of drifted Twitter spam," IEEE Trans. Inf. Forensics Security, vol. 12, no. 4, pp. 914_925, Apr. 2017.

C. SOURCE CODE

```
import tweepy
from tweepy import Cursor
import unicodedcsv
import numpy as np
import pandas as pd
from unidecode import unidecode
import json
import os

try:
    os.stat(os.getcwd()+'\\collectindividual')
    path = os.getcwd()+'\\collectindividual'
except:
    os.mkdir(os.getcwd()+'\\collectindividual')
    path = os.getcwd()+'\\collectindividual'
pd.set_option('display.max_columns', 1000)
pd.set_option('display.max_rows', 1000)
#pd.set_option('display.width', 1000)
#pd.set_option('max_colwidth', 30)
pd.set_option('colheader_justify', 'center')

# Authentication and connection to Twitter API.
consumer_key ='YBAUoHlkgYKQ9fDljywpM1DGd'
consumer_secret ='67yxNaSftHGGKfbqYe4G8nYeXZIXSe9F3il68HML0SX5ysU047'
access_key ='120092298-stEl5XDozGPotvSYLFonT1HlbK4TAuofi9NzDm7x'
access_secret ='UiAKoMghPOCoRZ9D3BHFakBHaVcnAJJLV3h0aBmMH8aoR'

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)
```

```

words =
["#bot", "#b0t", "#updates", "#hourly", "#automatically", "#generating", "#generated", "#every",
"#computer-generated", "#twitterbot", "#automated"]
date_since = "2019-11-01"
my_list_of_dicts = []
my_demo_list = []
# Collect tweets
def data_collection():
for search_words in words:
tweets =
tweepy.Cursor(api.search,q=search_words,lang="en",since=date_since).items(5)
print(search_words)
for json_tweet in tweets:
my_list_of_dicts.append(json_tweet._json)
with open('tweet_json.txt','w') as file:
file.write(json.dumps(my_list_of_dicts, indent=4))
with open('tweet_json.txt', encoding='utf-8') as json_file:
all_data = json.load(json_file)
for each_dictionary in all_data:
created_at = each_dictionary['created_at']
tweet_id = each_dictionary['id']
text = each_dictionary['text']
retweet_count = each_dictionary['retweet_count']
favorite_count = each_dictionary['favorite_count']
place = each_dictionary['place']
lang = each_dictionary['lang']
username = each_dictionary['user']['screen_name']
my_demo_list.append({'created_at': created_at,
'tweet_id': str(tweet_id),
'text': str(text),
'favorite_count': int(favorite_count),
'retweet_count': int(retweet_count),

```

```

'place': str(place),
'lang': str(lang),
'username':str(username)
})
#print(my_demo_list)
df = pd.DataFrame(my_demo_list)
df = df.drop_duplicates(subset ="tweet_id", keep ='first').reset_index(drop=True)
df =
df[['created_at','username','tweet_id','text','favorite_count','retweet_count','place','lang']]
#print(df)
return df
# for tweet in tweets:
#     print(tweet)
#     tweet = pd.DataFrame()
#     print(tweet)
def getExceptionMessage(msg):
words = msg.split(' ')
errorMsg = ""
for index, word in enumerate(words):
if index not in [0,1,2]:
errorMsg = errorMsg + ' ' + word
errorMsg = errorMsg.rstrip("\}")
errorMsg = errorMsg.lstrip("\ ")
return errorMsg
def download_user_bulk(dl_user):
try:
user_obj = api.get_user(dl_user)
with open(os.path.join(path,'coba'+dl_user+'.csv'), 'wb') as file:
writer = unicodedcsv.writer(file, delimiter = ',', quotechar = '"')
# Write header row.
writer.writerow(["id",
"id_str",

```

```

"Name",
"Username",
"Followers_count",
>Listed_count",
"Friends_count",
"Favorites_count",
"Created_at",
"Verified",
"Default_profile",
"Default_profile_image",
"Location",
"Statuses_count",
"Description",
"URL",
"Geo_enabled",
])
# Gather info specific to the current user.
user_info = [user_obj.id,
user_obj.id_str,
user_obj.name,
user_obj.screen_name,
user_obj.followers_count,
user_obj.listed_count,
user_obj.friends_count,
user_obj.favourites_count,
user_obj.created_at,
user_obj.verified,
user_obj.default_profile,
user_obj.default_profile_image,
user_obj.location,
user_obj.statuses_count,
user_obj.description,

```

```

user_obj.url,
user_obj.geo_enabled,
]
writer.writerow(user_info)
# Show status if success
print("Wrote tweets by %s to CSV." % dl_user)
except tweepy.TweepError as e:
print ("\n"+str(e.api_code) +":"+ getMessage(e.reason)+"\n")
def download_user(dl_user):
try:
user_obj = api.get_user(dl_user)
with open('coba.csv', 'wb') as file:
writer = unicodcsv.writer(file, delimiter = ',', quotechar = '"')
# Write header row.
writer.writerow(["id",
"id_str",
"Name",
"Username",
"Followers_count",
>Listed_count",
"Friends_count",
"Favorites_count",
"Created_at",
"Verified",
"Default_profile",
"Default_profile_image",
"Location",
"Statuses_count",
>Description",
"URL",
"Geo_enabled",
])

```

```

# Gather info specific to the current user.
user_info = [user_obj.id,
user_obj.id_str,
user_obj.name,
user_obj.screen_name,
user_obj.followers_count,
user_obj.listed_count,
user_obj.friends_count,
user_obj.favourites_count,
user_obj.created_at,
user_obj.verified,
user_obj.default_profile,
user_obj.default_profile_image,
user_obj.location,
user_obj.statutes_count,
user_obj.description,
user_obj.url,
user_obj.geo_enabled,
]
writer.writerow(user_info)
# Show status if success
print("Wrote tweets by %s to CSV." % dl_user)
except tweepy.TweepError as e:
print ("\n"+str(e.api_code) + ":" + getExceptionMessage(e.reason)+"\n")
from flask import Flask, render_template, request, jsonify, session
from retrieve_tweet import data_collection,download_user,download_user_bulk
import joblib
import pandas as pd
import os
import glob
import matplotlib.pyplot as plt
import matplotlib

```

```

matplotlib.rcParams['text.color'] = 'tab:orange'
from analisis_data_profil import preprocess, preprocess_bulk
app = Flask(_name_)
app.secret_key = 'twitter'
app.config['SEND_FILE_MAX_AGE_DEFAULT'] = 1
#SESSION_TYPE = 'filesystem'
model = joblib.load('finalized_model_without.sav')
path = os.getcwd()+"\\collectindividual'
def prediction_bulk(df):
    skrip = preprocess_bulk(df)
    pred_proba=model.predict_proba(skrip)
    y_pred=model.predict(skrip)
    percentage=pred_proba[:,1]
    joins=' '.join(map(str, percentage))
    perc=float(joins)*100
    percent=(str(perc)+"%")
    #print(y_pred)
    df = df.drop(df.columns[[17,18,19,20,21,22]], axis = 1)
    return df,percent,y_pred
def prediction(df):
    skrip,tab = preprocess(df)
    pred_proba=model.predict_proba(skrip)
    y_pred=model.predict(skrip)
    percentage=pred_proba[:,1]
    joins=' '.join(map(str, percentage))
    perc=float(joins)*100
    percent=perc
    print(y_pred)
    return percent,tab,y_pred
@app.route("/")
def login():
    return render_template('login.html')

```

```

@app.route("/collect")
def collect():
    df = data_collection()
    df=df.sort_values(by=['username']).reset_index(drop=True)
    df.to_csv('collect.csv')
    return render_template('collect.html',df=df.to_html())

```

```

@app.route("/test")
def test():
    df = pd.read_csv("collect.csv")
    res = pd.DataFrame()
    for ind in df.index:
        uname = df['username'][ind]
        print(uname)
        download_user_bulk(df['username'][ind])
    all_files = glob.glob(path + "/*.csv")
    li = []
    for filename in all_files:
        df = pd.read_csv(filename, index_col=None, header=0)
        li.append(df)
        fr,per,stat=prediction_bulk(df)
        pr = per
        st = stat
        fr['Percentage'] = pr
        if st == False :
            Result = "Legitimate User"
            fr['State'] = Result
        else:
            Result = "Fake/Bot User"
            fr['State'] = Result
        res = res.append(fr)
    res=res.reset_index(drop=True)

```

```

    return render_template('test.html',result=res.to_html())
@app.route("/check", methods=['POST','GET'])
def check():
    dl_user = request.form.get('chat_in')
    if dl_user == None:
        return render_template('check.html')
    else:
        session['username'] = dl_user
        download = download_user(dl_user)
        df=pd.read_csv('coba.csv')
        return render_template('check.html',df=df.to_html())
@app.route("/detect")
def detect():
    all_files = glob.glob(path + "/*.csv")
    for filename in all_files:
        if filename.endswith('.csv'):
            os.unlink(filename)
            #print(filename)
    my_var = session.get('username', None)
    print(my_var)
    df=pd.read_csv('coba.csv')
    prediksi,tab,y_pred=prediction(df)
    tab['Username']=my_var
    tab=tab.set_index('Username')
    tab=tab
    labels = 'Legitimate', 'Fake'
    size1 = 100-float(prediksi)
    size2 = prediksi
    sizes = [size1, size2]
    colors = ['forestgreen', 'crimson']
    explode = (0, 0.15)
    fig1, ax1 = plt.subplots()

```

```

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', colors =colors,
        shadow=True, startangle=90)
ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.
plt.savefig('static/images/graph/'+my_var+'.png',transparent=True)
img = 'static/images/graph/'+my_var+'.png'
return
render_template('detect.html',prediction=prediksi,tab=tab.to_html(),y_pred=y_pred,my_
var=my_var,img=img)

```

```

if __name__ == "__main__":
    app.run(debug=True)

```

```

import pandas as pd
import tweepy
from time import sleep
import csv
from datetime import datetime, date, time, timedelta
import datetime as dt
import io
import numpy as np
import joblib
import pickle
import re

def preprocess_bulk(df):
    #df=pd.read_csv('coba.csv')
    df=pd.DataFrame(df)
    df["Created_at"]= pd.to_datetime(df["Created_at"])

    #Count age in days
    now = pd.Timestamp('now')
    df['age_in_days'] = (now - df['Created_at']).dt.days
    #Number of tweets/age

```

```

df['Ratio Statuses_count/age']=(df['Statuses_count']/df['age_in_days']).round(3)
#Number of favourites/age
df['Ratio Favorites/age']=(df['Favorites_count']/df['age_in_days']).round(3)
#Ratio friends/follower
df['Ratio Friends/Followers']=(df['Friends_count']/df['Followers_count']).round(3)
#Count length of bio
df['Length_of_Bio']=df['Description'].astype(str).str.len()
#Measure reputation
df['Reputation']= df['Followers_count']/((df['Followers_count']+ df['Friends_count'])
df1=df.drop(['Geo_enabled','id_str','Created_at'], axis=1)
df1['URL'] = pd.notnull(df1['URL'])
df1['Location']=pd.notnull(df1['Location'])
bag_of_word
='bot|b0t|updates|hourly|automatically|generating|generated|every|computer-
generated|twitterbot|automated|'
df2=df1
df2['contains_bot_name']=df2['Description'].astype(str).str.contains(bag_of_word,
flags=re.IGNORECASE, regex=True)
df2=df2.drop(['Description'], axis=1)
df2['Length_of_Bio'].fillna(0, inplace = True)
df2['Reputation'].fillna(0, inplace = True)
df2['contains_bot_name'].fillna(False, inplace=True)
df2=pd.DataFrame(df2)
db=df2
db=pd.DataFrame(db)
db['URL'].fillna(False,inplace=True)
db.replace(np.inf, 0)
db.replace(-np.inf, 0)
db = db.replace(np.NaN, 0)
db = db.replace([np.inf, -np.inf], 0)
db=db.drop(['id'], axis=1)
db = db[['Followers_count',

```

```

        'Listed_count',
        'Friends_count',
        'Favorites_count',
        'Verified',
        'Default_profile',
        'Default_profile_image',
        'Location',
        'Statuses_count',
        'URL',
        'age_in_days',
        'Ratio Statuses_count/age',
        'Ratio Favorites/age',
        'Ratio Friends/Followers',
        'Length_of_Bio',
        'contains_bot_name',
        'Reputation',
    ]]
    data_prob=db.values.tolist()
    db=pd.DataFrame(db)
    return data_prob
def preprocess(df):
    df=pd.read_csv('coba.csv')
    df=pd.DataFrame(df)
    df["Created_at"]= pd.to_datetime(df["Created_at"])
    #Count age in days
    now = pd.Timestamp('now')
    df['age_in_days'] = (now - df['Created_at']).dt.days
    #Number of tweets/age
    df['Ratio Statuses_count/age']=(df['Statuses_count']/df['age_in_days']).round(3)
    #Number of favourites/age
    df['Ratio Favorites/age']=(df['Favorites_count']/df['age_in_days']).round(3)
    #Ratio friends/follower

```

```

df['Ratio Friends/Followers']=(df['Friends_count']/df['Followers_count']).round(3)
#Count length of bio
df['Length_of_Bio']=df['Description'].astype(str).str.len()
#Measure reputation
df['Reputation']= df['Followers_count']/(df['Followers_count']+ df['Friends_count'])
df1=df.drop(['Geo_enabled','id_str','Created_at'], axis=1)
df1['URL'] = pd.notnull(df1["URL"])
df1['Location']=pd.notnull(df1['Location'])
bag_of_word
='bot|b0t|updates|hourly|automatically|generating|generated|every|computer-
generated|twitterbot|automated|'
df2=df1
df2['contains_bot_name']=df2['Description'].astype(str).str.contains(bag_of_word,
flags=re.IGNORECASE, regex=True)
df2=df2.drop(['Description'], axis=1)
df2["Length_of_Bio"].fillna(0, inplace = True)
df2["Reputation"].fillna(0, inplace = True)
df2["contains_bot_name"].fillna(False, inplace=True)
df2=pd.DataFrame(df2)
db=df2
db=pd.DataFrame(db)
db['URL'].fillna(False,inplace=True)
db.replace(np.inf, 0)
db.replace(-np.inf, 0)
db = db.replace([np.inf, -np.inf], 0)
db=db.drop(['id'], axis=1)
db = db[['Followers_count',
        'Listed_count',
        'Friends_count',
        'Favorites_count',
        'Verified',
        'Default_profile',

```

```
'Default_profile_image',
'Location',
'Statuses_count',
'URL',
'age_in_days',
'Ratio Statuses_count/age',
'Ratio Favorites/age',
'Ratio Friends/Followers',
'Length_of_Bio',
'contains_bot_name',
'Reputation',
]]
data_prob=db.values.tolist()
db=pd.DataFrame(db)

return data_prob,db
```