

Detection of Fake Profiles on Multiple Platforms (DFPMP)



By

Syed Muhammad Ali Naqi

Muhammad Sabeeh Waqas

Mirza Haidar

Syed Muhammad Hasan

Supervised by:

Dr. Haider Abbas

Submitted to the faculty of the Department of Electrical Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment of the requirements of a Bachelor's Degree in Electrical Engineering.

June 2023

In the name of ALLAH, the Most Benevolent, the Most Courteous.

CERTIFICATE OF CORRECTNESS AND APPROVAL

This is to officially state that the thesis work contained in this report

“Detection of Fake Profiles on Multiple Platforms”

is carried out by

Syed Muhammad Ali Naqi

Muhammad Sabeeh Waqas

Mirza Haidar

Syed Muhammad Hasan

under my supervision and that in my judgment, it is fully ample, in scope and excellence, for the degree of Bachelor of Electrical Engineering in Military College of Signals, National University of Sciences and Technology (NUST), Islamabad.

Approved by

Supervisor

Dr. Haider Abbas

Department of EE, MCS

Date: 27/4/2023

DECLARATION OF ORIGINALITY

We hereby declare that no portion of the work presented in this thesis has been submitted in support of another award or qualification in either this institute or anywhere else.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, and especially our supervisor, **Dr. Haider Abbas**
without your guidance.

The group members, through all adversities, worked steadfastly.

Plagiarism Certificate (Turnitin Report)

This thesis has an 16% similarity index. The Turnitin report endorsed by Supervisor is attached.

Syed Muhammad Ali Naqi
NUST Serial no. 00000295659

Muhammad Sabeeh Waqas
NUST Serial no. 00000295769

Mirza Haidar
NUST Serial no. 00000297369

Syed Muhammad Hasan
NUST Serial no. 00000288861

Signature of Supervisor

Contents

Chapter 1.....	9
Introduction	9
<u>s</u> 1.1 Overview	10
Figure 1: Number of Twitter users worldwide from 2019 to 2024 (<i>in millions</i>).....	10
1.2 Problem Statement	11
1.3 Proposed Solution	12
1.4 Working Principle	13
1.4.1 Profile Name Entry.....	13
1.4.2 Data Scraping.....	13
1.4.3 Account Grouping	13
1.4.4 Keyword-Based Data Search	14
1.4.5 Feedback System with Image Analysis	14
1.4.6 Sentimental Analysis	14
1.4.7 Botometer Data Collection.....	14
1.4.8 Percentage Assignment.....	14
1.4.10 Results Display	15
1.5 Objectives	15
1.5.1 General Objectives:	15
1.5.2 Academic Objectives:	15
1.6 Scope	16
1.7 Deliverables	16
1.7.1 Fake Profile Detection System	16
1.7.2 Hateful Speech Detection System	17
1.8 Relevant Sustainable Development Goals.....	17
1.9 Structure of Thesis.....	17
Chapter 2.....	19
Literature Review	19
2.1 Existing Solutions and their drawbacks	19
2.1.1 “simpin.com”.....	19
2.1.2 “zerofox.com”	20
2.1.2 “Botometer.com”	21
Figure 2: Features and Drawbacks of existing solutions	21
2.2 Research Papers.....	22
2.2.1 Research with Account-Based Features.....	22
2.2.2 Research with Text-Based Features	23
2.2.3 Work with Account- and Text-Based Features.....	24
Chapter 3.....	26

Design and Development	26
3.1 Profile Name Entry	26
3.2 Data Scraping	26
3.3 Account Grouping	28
3.4 OSINT	30
3.5 Feedback System with Image Analysis	31
Figure 4: VGG-16 model	35
3.6 Sentimental Analysis	35
Figure 5: Natural Language toolkit	36
3.7 Botometer Data Collection	37
3.8 Percentage Assignment	38
3.9 Hate Speech Analysis Model	39
3.10 Results Display	40
Figure 6: Working Model	40
Chapter 4	41
Detailed evaluation and analysis of the code	41
4.1 START_HERE.py	41
4.2 description1.py	41
4.3 USER_OSINT.py	43
4.4 affiliated_images_for_feature_matching(user_twitter_search, user_osint)	45
4.5 Tweet_Analysis.py	45
4.6 Botometer.py	46
• Tweets by day of week : gives the frequency of he tweet done per day of week	47
• Tweets by hour of day : frequency of the tweet done as per the time of the day	47
4.7 Percentage_giver()	52
4.8 Machine Learning Algorithm	53
4.9 Hate_Speech()	55
4.10 Website	56
Chapter 5	57
Conclusion	57
Chapter 6	58
Future Work	58
References	60

Chapter 1

Introduction

Social media sites like Twitter have permeated contemporary communication to such an extent that false accounts are becoming more and more common. Bot accounts, usually referred to as automated accounts, are established for a variety of malevolent objectives. These profiles have the power to distribute false information, sway public opinion, and even find potential terrorist recruits. Additionally, the use of false personas for online bullying and harassment may have very negative emotional and psychological effects on the targeted.

False Twitter accounts may compromise someone's identity by committing identity theft. Cybercriminals may establish a phony profile that looks to belong to a victim by using the victim's name, photo, and other personal information. They may then use this account to publish offensive material, participate in online bullying, or carry out other nefarious deeds that affect the victim's credibility and reputation.

By disseminating incorrect information about an individual, fake Twitter accounts may potentially harm that person's reputation. Cybercriminals may create fictitious accounts to disseminate untrue or inaccurate information about the victim, harming their reputation and causing them great grief.

The present techniques for finding phony Twitter identities often depend on rudimentary rule-based heuristics or manual human analysis, both of which may be costly, time-consuming, and unreliable. More sophisticated techniques are urgently required to identify these accounts correctly and efficiently since the number of phony profiles on Twitter keeps increasing.

As a result, the issue of locating bogus Twitter accounts utilizing machine learning and artificial intelligence approaches has surfaced. These techniques examine a variety of characteristics of a Twitter account, including the number of followers, the frequency of tweets, and the substance of the tweets, in order to accurately detect phony accounts. By detecting and reporting bogus personas that are used for this purpose, such an approach may be used to reduce online abuse. It can also help intelligence services identify possible threats.

In general, social media platforms, users, and authorities are quite concerned about the issue of spotting phony Twitter identities. We can contribute to maintaining the integrity of social media platforms and making the internet a safer place for all users by creating effective and dependable methods to identify these accounts.

1.1 Overview

Twitter uses a variety of strategies to draw in and keep users. The platform, first of all, has a user-friendly layout that makes it simple for individuals to register, establish an account, and begin using it. In addition, Twitter offers a number of tools for interaction between users, including direct messaging, following other users, enjoying and retweeting content, and retweeting other users' postings.

In addition, Twitter has developed into a center for news, information, and entertainment, with a large number of public figures, corporations, and celebrities utilizing the service to interact with their constituents. As a result, Twitter now has a stronger feeling of community and is a more desirable platform for users to join and remain active on.

To enhance user experience, Twitter also often upgrades its platform and adds new features. For instance, it now offers capabilities like audio tweets, live streaming, and Twitter Spaces, which let users take part in audio chats.

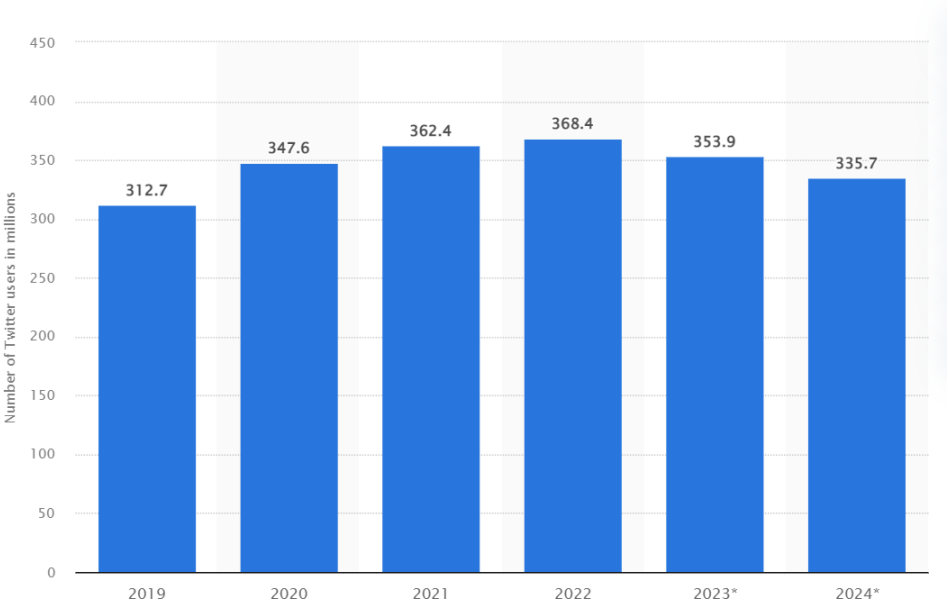


Figure 1: Number of Twitter users worldwide from 2019 to 2024(in millions)

As more people use Twitter, there is also a greater chance that some people or businesses may seek to set up accounts impersonating well-known people, celebrities, or brands. These impersonation accounts may be used for a variety of things, such as disseminating false information, defaming someone, or phishing personal data. The temptation for malevolent actors to establish phony accounts that closely resemble genuine ones grows as the number of possible targets rises.

1.2 Problem Statement

Due to the growth of social media and the possibility of abuse, the issue of identifying false Twitter identities has grown in importance in recent years. The safety and integrity of social media platforms are in danger due to the usage of bogus Twitter identities for a variety of nefarious activities, such as distributing misinformation, swaying public opinion, and indulging in online bullying and harassment.

Dramatic Increase Detected in Impersonation Attacks on Social Media

By **Jessica Ellis** | June 2, 2022

PhishLabs
Digital Risk Protection

Impersonations of brands and executives on social media have grown more than 300% and 250% year-over-year, respectively, according to the [Agari and PhishLabs Quarterly Threat Trends & Intelligence Report](#). This highlights the overall increase in social media activity and ease of accessibility for bad actors to repurpose stolen trademarks and intellectual property (IP) for a variety of malicious activities, including spreading misinformation and promoting [counterfeit products](#). Actors abuse this material to launch attacks on unsuspecting consumers by creating pages, posts, and other content that are nearly identical to legitimate corporate imagery and messaging.

Figure 2: Article that shows how much the fake accounts are increasing.

Simple rules-based heuristics or manual human analysis are two ways that are currently used to identify bogus Twitter accounts, but they are time-consuming, costly, and not totally accurate. As a result, there is a need for more sophisticated techniques that can quickly and precisely identify false accounts.

Therefore, the issue statement is how to create trustworthy and effective techniques for spotting fraudulent Twitter profiles that can successfully find and delete these accounts from social networking networks. In order to correctly detect and delete bogus accounts, the solution should make use of machine learning and artificial intelligence algorithms to examine numerous Twitter account characteristics, such as the number of followers, the frequency of tweets, and the substance of the tweets.

1.3 Proposed Solution

The suggested answer is to create a system for detecting phony Twitter accounts that reliably and quickly locates and deletes them from social media sites using machine learning and artificial intelligence approaches.

The algorithm will examine a range of Twitter account characteristics, including follower count, following accounts, description, joining date, tweet volume, content, etc. To find trends that point to a phony profile. The technology will also make use of sentiment analysis and natural language processing to spot hate speech and other damaging information that could be connected to phony accounts.

A large dataset of actual and fraudulent Twitter accounts will be gathered and categorized for training and testing in order to build the system. To discriminate between authentic and false profiles, the system will be trained on this dataset using a variety of machine learning methods, including deep learning.

For the purpose of identifying offensive material that may be connected to phony accounts, the system will also include hate speech identification technology. The hate speech detection technology will use sentiment analysis and natural language processing to identify offensive and damaging language, and it will take the necessary steps to remove such material and profiles from the site.

The development of a fake profile detection system that makes use of machine learning and artificial intelligence methods in order to precisely and quickly identify and delete phony Twitter accounts and related damaging information from social media platforms is the suggested answer, in conclusion. To provide all users with a safer and more secure online experience, the system will also include a hate speech recognition mechanism.

1.4 Working Principle

The working principle of the proposed fake profile detection system involves the use of machine learning and artificial intelligence techniques to analyze various attributes of a Twitter account and identify patterns that are indicative of a fake profile.

1. Profile Name Entry
2. Data Scraping
3. Account Grouping
4. Keyword-Based Data Search
5. Feedback System with Image Analysis
6. Sentimental Analysis
7. Botometer Data Collection
8. Percentage Assignment
9. Hate Speech Analysis Model
10. Results Display

1.4.1 Profile Name Entry

Entering the profile name against which to check for fake profiles on the website is the first step. This may be a public personality, a brand, or any other person or organization having a Twitter account.

1.4.2 Data Scraping

Once the profile name has been supplied, the system will scrape all Twitter accounts' data associated with that name. This includes the user's profile description, location, number of followers, and other pertinent information.

1.4.2.1 Scraping Users' Names

Search for all the profiles related to that name and scrape the usernames of all Twitter profiles that come up on the search list.

1.4.2.2 Scraping Users' Data

Scrape the data of every account against all usernames that include profile description, location, date of joining, number of followings, number of followers, and other pertinent information.

1.4.3 Account Grouping

The algorithm will then group accounts with similar characteristics based on the user descriptions. Multiple accounts with identical profile descriptions will be placed together, for instance.

1.4.4 Keyword-Based Data Search

Next, the algorithm will utilize the group's keywords to search Google for all relevant information. This will assist in identifying any further information about the group's members that may not be accessible on Twitter.

1.4.5 Feedback System with Image Analysis

A feedback mechanism will be utilized to sort accounts according to their profile pictures. This may assist in further narrowing the account grouping.

1.4.6 Sentimental Analysis

The sentimental analysis will be used to examine user activity, including the sentiment of tweets and the posting frequency. This may aid in the identification of accounts that may be engaged in spamming or other undesirable activity.

1.4.7 Botometer Data Collection

The system will gather data from every user's Botometer across all groups. Botometer is an algorithm that uses machine learning to identify Twitter bots. This may assist detect any suspected fraudulent accounts that may be manipulating their followers or interaction using bots.

1.4.8 Percentage Assignment

The data of users in each category will be compared, and each account will be assigned a percentage depending on the likelihood that it is a false profile. To estimate the possibility of a fake profile, the algorithm will analyze many parameters, including the number of followers, engagement rate, sentiment analysis, and Botometer score.

1.4.8.1 For Google verified account

If the OSNT tool finds the Google-verified account, then the system gives 100% authenticity to that account, and the remaining accounts are going to get a percentage on the basis of their resemblance to that verified account.

1.4.8.1 For non-verified account

If the OSNT tool doesn't find any Google-verified account, then there are two cases:

1.4.8.1.1 Verified followers:

If the user's list of followers includes verified accounts, the system will provide that account with the highest level of credibility.

1.4.8.1.1 Machine Learning Algorithm:

If the user's list of followers doesn't include verified accounts, the system will pass that account to a machine-learning algorithm that decides the percentage of the authenticity of that account.

1.4.10 Results Display

On the website, the percentage of authenticity and user information will be published. This will help people make educated judgments about whom to follow and interact with on Twitter.

1.5 Objectives

1.5.1 General Objectives:

1. Using methods from machine learning and artificial intelligence, design a dependable and efficient system for identifying bogus Twitter accounts.
2. Using machine learning and artificial intelligence approaches, construct a dependable and effective system for identifying bogus Twitter accounts.
3. To establish a safer online environment for all users by recognizing and eliminating fraudulent personas that might be used for malevolent objectives, such as propaganda dissemination or online bullying and harassment.
4. To provide intelligence agencies with an efficient method for recognizing possible threats on social media networks.
5. To educate social media users about the risks associated with fake accounts and the significance of online safety and security.

1.5.2 Academic Objectives:

1. Exploring and analyzing the current literature on false profile detection systems, machine learning, and artificial intelligence approaches.
2. Analyzing the strengths and limits of several ways to detect fake profiles, and to determine the most successful and efficient strategies.
3. To analyze the strengths and limits of several methods for detecting false profiles and to determine the most successful and efficient strategies.
4. To perform tests and assessments to determine the precision and efficacy of the proposed method for identifying fake Twitter accounts.

5. To contribute to scholarly knowledge in the field of social media analytics, particularly in the areas of detecting fraudulent profiles and online disinformation.
6. To disseminate the findings of the research through publications in academic journals and conferences, and to contribute to the academic community's understanding of the challenges and opportunities in social media analytics.

1.6 Scope

This project aims to build a system for detecting bogus Twitter profiles using machine learning and artificial intelligence approaches. The technology will be intended to automatically detect false Twitter accounts and deliver a percentage of authenticity based on the study of many parameters, including the number of followers, tweet frequency, and tweet content. The technology will also feature a recognition module for hate speech to identify tweets containing damaging or abusive words.

The project will involve data collection, data preprocessing, feature extraction, model selection, and model evaluation. The system will be implemented using Python programming language and various machine learning libraries such as scikit-learn, CNN, and Keras. The system will be deployed on a website, where users can enter the profile name to check for fake profiles, view the percentage of realness, and access the hate speech analysis report.

The project's scope includes the design, development, and assessment of the system, as well as the dissemination of its results through academic papers and conferences. Various parties, including social media platforms, intelligence agencies, and individual users, will find the system valuable for identifying and eliminating bogus accounts and fostering a safer online environment.

1.7 Deliverables

1.7.1 Fake Profile Detection System

This is the main deliverable of the project, and it involves the development of a system that can detect fake Twitter profiles with a high degree of accuracy. The system will use machine learning and artificial intelligence techniques to analyze various attributes of a Twitter account, such as the number of followers, tweet frequency, and the content of the tweets. The system will also incorporate a feedback mechanism to improve its accuracy over time.

1.7.2 Hateful Speech Detection System

In addition to detecting fake profiles, the project will also involve the development of a system that can detect hateful speech on Twitter. The system will use natural language processing techniques to analyze the content of tweets and identify those that contain hateful or abusive language.

1.7.3 User Interface

The project will also involve the development of a user interface that allows users to enter the name of a Twitter profile and receive a report on the percentage of fake accounts associated with that profile, as well as any instances of hateful speech. The user interface will be intuitive and easy to use, with clear and concise reports.

1.7.4 Documentation

Finally, the project will deliver documentation that includes a detailed description of the system's functionality, the algorithms used to detect fake profiles and hateful speech, and the user interface. The documentation will also include instructions on how to use the system and any necessary installation and configuration instructions. Additionally, the documentation will include any relevant research and analysis conducted during the development of the system.

1.8 Relevant Sustainable Development Goals

The Socio-Economic problem that we are trying to solve through our project is cyberbullying.

Our final year project emphasizes the **sixteenth** Sustainable Development Goal.

Promote peaceful and inclusive societies for sustainable development, provide access to justice for all, and build effective, accountable, and inclusive institutions at all levels.

1.9 Structure of Thesis

Chapter 2 contains the literature review and the background and analysis study this thesis is based upon.

Chapter 3 contains the design and development of the project.

Chapter 4 introduces a detailed evaluation and analysis of the code.

Chapter 5 contains the conclusion of the project.

Chapter 6 highlights the future work needed to be done for the commercialization of this project.

Chapter 2

Literature Review

Understanding the current items on the market and identifying their strengths and flaws is essential for the development of a new product. Herein lies the value of the literature review. By doing a comprehensive literature analysis, the developers may collect pertinent information about comparable current goods, such as their features, functions, and customer feedback.

In the context of the Detection of Fake Profiles on social media, a literature study may give vital insights into the numerous market-available goods, such as which data-gathering platform is the most convenient, which parameters are most beneficial, and which methods are the most effective. By studying previous work, developers may find market gaps and generate fresh ideas for creating a superior solution.

In addition, a literature study may assist improve the characteristics of current goods by suggesting areas for enhancement. For example, if a product lacks certain features or has limits, its creators might enhance these qualities and deliver a superior product to the market.

Overall, completing a literature study is a crucial stage in building a new product since it helps to detect market gaps, comprehend current goods, and improve the product's characteristics.

Our research is divided into the following points.

1. Existing solutions and their drawbacks
2. Research Papers

2.1 Existing Solutions and their drawbacks

2.1.1 “simpin.com”

Certainly! "smipin.com"[1] is a website that offers online reputation management services, which include monitoring and enhancing an individual's or company's online visibility. One of the primary services provided by "smipin.com" is the detection and elimination of impersonation. When someone establishes a fake account or profile using another person's name, picture, or other personal information, this constitutes impersonation. This may be detrimental to the individual or business being impersonated since it can result in the propagation of unpleasant comments or

misleading information online. Additionally, impersonation may be a type of identity theft, which can have severe legal and financial repercussions.

"smipin.com" uses numerous methods and approaches to monitor the Internet for instances of impersonation in order to fight impersonation. This may include monitoring social media sites, search engines, and other online channels where impersonation is possible. If impersonation is found, "smipin.com" may collaborate with the appropriate platform or website to get the bogus account or profile deleted.

Overall, "smipin.com" aims to assist people and businesses in managing their online reputations and defending themselves against online dangers, such as impersonation. It is essential to recognize, however, that online reputation management is a complicated, continuing process that demands careful attention and planning. Although "smipin.com" may provide beneficial services in this area, it is always wise to perform extensive research and speak with reputable consultants before making any choices about the management of your internet reputation.

2.1.2 “zerofox.com”

The website zerofox.com [2] provides a variety of online security and threat prevention services, including detection and elimination of impersonation. The impersonation-specific service is intended to assist people and organizations in identifying instances of impersonation on social media and other online platforms and in removing the impersonating accounts or profiles.

The impersonation detection and eradication service provided by "zerofox.com" identifies probable cases of impersonation using a mix of human and machine-based methodologies. This may include monitoring social media accounts and other online channels for signs such as fraudulent profiles, unusual account activity, and trademark infringement. After identifying an incident of impersonation, "zerofox.com" collaborates with the appropriate platform or website to get the impersonating account or profile terminated.

In addition to detection and removal services, "zerofox.com" also provides training and education tools to assist people and businesses understand the hazards associated with impersonation and other online security threats and learn how to defend themselves proactively.

Overall, "zerofox.com" aims to assist people and companies in protecting themselves from online security risks and preserving their online reputation. It is essential to emphasize, however, that internet security is a continuous process that demands constant awareness and attention to detail. Although "zerofox.com" may provide important services in this area, it is always prudent to

perform extensive study and speak with reputable consultants prior to making choices about internet security and reputation management.

2.1.2 “Botometer.com”

"Botometer.com" [3] is a website that gives a tool for determining if a Twitter account is likely to be a bot. Bots are automated accounts that may be used for several objectives, including spamming, disseminating false information, and influencing online debates.

Simply input the Twitter handle or username you desire to study in order to utilize "Botometer.com." The program will then provide a report with a "bot score" reflecting the account's chance of being a bot. The bot score is determined by a variety of criteria, such as account activity, demographics of followers, and content analysis.

In addition to the bot score, "Botometer.com" gives a comprehensive report that details the account's activity patterns, content topics, and other indicators that may suggest bot activity. This information may aid in the identification of possible bots and the protection against their impact.

Overall, "Botometer.com" aims to assist people and organizations in understanding the existence of bots on Twitter and protecting themselves from their impact. It is essential to note, however, that "Botometer.com" is not perfect and may not be able to identify every instance of bot activity.

In addition, it is always advisable to use a variety of tools and strategies to assess online activity and make educated judgments about online engagement and strategy.























	 smip	Botometer®	 ZEROFOX
cover twitter			
detect human impersonation			
user friendly			
level of automation			
dependency on twitter API			
hated speech analysis			
price			

Figure 2: Features and Drawbacks of existing solutions

2.2 Research Papers

We categorized the fake profile detection research into three categories:

- 1) research with account-based features
- 2) research with text-based features
- 3) research with both account- and text-based features.

In the subsequent sections, we highlight the key findings, limitations, and future scope of the proposed models.

2.2.1 Research with Account-Based Features

Gurajala *et al.*[4] For the identification of false profiles, a pattern-matching algorithm-based approach was presented. Using the method of social web crawling, 62 million user profiles were gathered. The filtering method yielded 724 494 groups comprising a total of 6 958 523 accounts. For the purpose of further narrowing 724 494 groupings, their screen names were examined, and almost similar ones were discovered. Using map-reduction methods and pattern recognition, a highly reliable subset of fictitious user accounts was detected.

In 2016, **Gurajala *et al.***[4] Examined the features of Twitter accounts that were fraudulent. Based on the profile name and other characteristics, they grouped user profiles into fake and legitimate accounts. Their technology accurately identified a portion of users as Fake, and via human verification, all projected bogus accounts have been confirmed. Following a time gap of fewer than 40 seconds, their investigation indicated that bogus accounts are established in groups on weekdays at certain dates and times. On a database of 2016 Twitter users, the program detected fake accounts with a 93 percent accuracy rate.

BalaAnand *et al.*[4] The user's nonverbal behavior was used to identify Fake users on the social networking site. On social networking platforms, the nonverbal behavior data of the user helps identify numerous accounts and identity fraud. As a dataset, they used Wikipedia's publicly accessible logs of prohibited user information and applied classic machine learning-based classifiers such as SVM, RF, and adaptive boosting algorithms using the cross-validation method. Using the adaptive boosting classifier, they attained the highest accuracy in performance.

Cresci et al.[4] The author proposed a methodology for identifying fraudulent Twitter followers. They used numerous datasets, including Fast Followers, Inter Twitter, and Twitter technology, each of which included a sample size of 469, 1481, 1169, 1337, and 845, respectively. Furthermore, the dataset was divided into two baseline datasets: the baseline human dataset (1950 accounts) and the baseline artificial dataset (1950 accounts). For the identification of false and spam accounts, a set of ML algorithms are deployed. The SVM classifier correctly categorized more than 95% of accounts into their respective groups.

2.2.2 Research with Text-Based Features

This section highlights the research that was done using the textual features of the OSN websites to detect fake, malicious, or spam accounts. Swe and Myo proposed a model to detect the fake account of OSN websites using a blacklist instead of a traditional spam words list. The blacklist is created by using the topic modeling approach and a keyword extraction approach. The evaluation is done on the 1KS-10KN dataset and also on the Social HoneyPot dataset. The traditional spam-word-list-based approach on the 1KS-10KN dataset achieved a precision value of 0.854, recall of 0.904, and F-measure of 0.879, whereas using the proposed methods, the precision, recall, and F1-measure were 0.958, 0.950, and 0.954, respectively. The fake account detection rate using the proposed model was 95.4%. The false positive rate of the spam-word-list-based approach is 0.154, and the false positive rate of the blacklist-based approach using the social honeypot dataset is 94.9%. The detection rate of the spam-word-list-based approach using the social-honeypot-based approach is 91.1%. The blacklist-based approach achieves acceptable accuracy and reduces the false positive rate. Their model does not require profile- and network-based features, and hence, it reduces the time and cost overhead for extracting these features.

Clark et al. Used natural language processing (NLP) to detect automation on Twitter. Their model uses natural language text from humans to provide a criterion for identifying accounts with automated messages. Two datasets were collected: first, geo-tweets from the most active 1000 users, referred to as the Geo-Tweet dataset to classify humans and robots. The second collection of data was obtained from the social honeypot experiment. They found that the accuracy of the model on the Geo-Tweet dataset increased by increasing the tweets bin's size. The model achieved a true positive rate of 86%, which means that most robots have been identified successfully. The

model uses textual data on its own for prediction purposes, so it is flexible and can be applied to any text data beyond the Twitter-sphere.

Khan *et al.* Segregated spammers and bloggers from real experts on Twitter. Approximately 0.4 million tweets were collected from approximately 3200 user profiles active in disseminating health-related information on Twitter. They used the Hyperlink-Induced Topic Search (HITS) approach to classify spammers and bloggers and isolated them from experts in a specific area. Their model does not require a large amount of pre-classified data to differentiate bloggers from true experts. The top 30% of the 3200 profiles were marked as bloggers with a precision score of 0.70. Galán-García *et al.* Have developed a model for identifying fake profiles on Twitter. They collected 1900 tweets referring to 19 separate tweeter accounts. The tweets were modeled using the vector space model, and then, supervised classification models were applied. In the best case, the accuracy of the model was 68.47% using the sequential minimal optimization technique.

2.2.3 Work with Account- and Text-Based Features

Chakraborty *et al.* have proposed a framework called social profile abuse monitoring. They gathered information from the Twitter profile of 5000 users along with their 200 latest tweets. The SVM classifier was used to analyze the dataset. They introduced a four-class classification model for calculating profile similarity indexing based on fine-grained interface similarity characteristics. The F1-score of the proposed approach was 70%, with a precision value of 0.60. Hua and Zhang proposed a spam profile identification interface on Twitter. The dataset included 173 spam accounts and 285 nonspam account screen names for a total of 458 screen names from emails sent by Twitter to followers. The threshold and associative classification techniques were used to achieve an accuracy value of 79.26%. This model is slower, but an iterative version of the model can be developed to improve performance in the future. Singh *et al.* suggested a model for identifying malicious, nonmalicious, and celebrity users on Twitter. The dataset comprised 7500 users of the website, and it was divided into a 70:30 ratio for training and testing. A total of five classifiers, namely, BayesNet, NB, social media optimization (SMO), J48, and RF, were used for model development. In the best case, their model achieved an accuracy of 99.80% using the RF classifier.

Cresci *et al.* Developed a framework for the identification of fake profiles on Twitter. They divided the work into three phases: first, they studied the existing features and rules for the identification of anomalies in various contexts, such as the Academy and the Media. Second, it developed a dataset for human and fake profile detection, and finally, it designed machine-based classifiers designed over the collection of rules and revised features. Their experiment showed that the rules used in the Media domain were not useful for the detection of a fake follower, whereas the rules and features associated with Academia produced a good result. The built-in model successfully detected 95% of fake profiles on training data of Twitter. Alsaleh *et al.* [4] have developed a model to detect Sybil accounts on Twitter. 1.8 million Twitter accounts were collected, of which 48.05% were human, 44.42% Sybil, and the remaining 7.53% were hybrid accounts. Seventeen features were extracted from the dataset, and five ML-based classifiers were used to classify the accounts into three classes. The Weka tool was used for the experiment and found the best results with the combination of MLP and gradient descent, where the human, hybrid, and Sybil F1-scores were 0.94, 0.22, and 0.93, respectively.

El Azab *et al.* Used the minimum weighted function collection to detect fake profiles on Twitter. They defined a minimized set of key factors that affect the identification of fake Twitter accounts and then used various classification methods to assess the factors. They used a dataset called: the fake project.¹³ The dataset consists of 1481 human accounts and 3000 fake profiles. David *et al.* Suggested a model for selecting the appropriate features to identify fake accounts. The dataset of 853 bot profiles and the most recent 1000 tweets in each timeline was collected over a week. This was complemented by 791 manually labeled human accounts between April and June 2016, most of which were Mexican users. User accounts and part of their timelines were extracted via the Twitter API. SVM, DT, NB, RF, and single-layer feedforward artificial neural networks (ANNs) have been used to classify the accounts. The highest average accuracy was 94% achieved with an RF classifier operating on 19 features. The relatively low variation of results across classifiers supports the belief that consistency features and subset selection of features play a significant role in incorrect predictions. Despite the convergence toward 91–92%, the remaining methods have not been similarly effective in terms of growth.

Chapter 3

Design and Development

3.1 Profile Name Entry

When it comes to checking for fake profiles on Twitter, the first step is to log in to our website where the fake profile checker is available. Once logged in, the user needs to enter the profile name they want to check for authenticity. This profile could belong to a public personality, a brand, or any other person or organization having a Twitter account.

It is important to note that the user should enter the exact username or handle of the profile they want to check. This is because there may be multiple profiles with similar names or handles, and searching for the wrong profile could yield inaccurate results.

Our website is very interactive and easy to use for any person. I person that can make a social media account can use our website. It's that easy to use for a normal user.

3.2 Data Scraping

Selenium is a popular tool for software testing that is also used for web scraping. It enables the automation of online browsers, which is beneficial for activities like completing forms, pressing buttons, and traversing websites.

Selenium is used to automate the process of looking for and navigating to Twitter user profiles in the context of web scraping Twitter accounts. This may be a time-consuming task when performed manually, but Selenium makes it fast and efficient.

Selenium creates an instance of a web browser (such as Chrome or Firefox) and controls it through an application programming interface. This makes it possible to automate online activities such as clicking links, filling out forms, and scraping data from websites.

In the instance of Twitter profile scraping, Selenium is used to automate the process of searching for user profiles based on their complete names and then extracting the needed data from the accounts. This enables the quick collection of data from a large number of profiles for further analysis.

Data scraping is the extraction of data from a variety of internet sources. In the case of detecting Fake Twitter accounts, data scraping is crucial for obtaining information about the profile in the issue. Selenium, which is a robust framework for automating web browsers, is one of the most used technologies for online scraping.

In addition, Twitter data may be scraped via bot accounts. Accounts for these bots are established particularly for data scraping and may be programmed to conduct certain activities, such as looking for profiles and harvesting information from several accounts.

After the user provides the profile name to the false profile checker, the system scrapes data from all Twitter accounts linked with that name. This includes the user's profile description, location, number of followers, and any other pertinent information that may be used to determine whether or not the profile is Fake.

The first phase of data scraping is to search for all profiles associated with the provided name. Using Selenium, the fake profile checker can automate the search process and scrape the usernames of all Twitter accounts on the search results list. This is accomplished by emulating user behavior in order to search for profiles, click on relevant links, and collect the required data. After scraping all usernames, the next step is to scrape the data of every account against all usernames. This includes:

- Username,
- Full name,
- Location,
- Description,
- Followers,
- Following,
- Join Date,
- Verification status,
- Lists,
- Likes,
- Birthday,
- Profession,
- Tweets,
- Website

Using Selenium, the fake profile checker can automate the data scraping process and extract the relevant information from each account associated with the username.

3.3 Account Grouping

Scikit-learn (Also known as sklearn) is a popular open-source machine-learning library for Python. It provides a wide range of tools and algorithms for various machine learning tasks such as classification, regression, clustering, and dimensionality red.

Some of the features provided by scikit-learn include:

Data preprocessing: handling missing data, feature scaling, feature selection, and encoding categorical variables.

Model selection and evaluation: splitting data into training and testing sets, cross-validation, and hyperparameter tuning.

Supervised learning: algorithms for classification (e.g. logistic regression, decision trees, and support vector machines) and regression (e.g. linear regression, random forests, and neural networks).

Unsupervised learning: algorithms for clustering (e.g. k-means, hierarchical clustering, and DBSCAN) and dimensionality reduction (e.g. principal component analysis and t-SNE).

Assembling methods: combining multiple models to improve performance (e.g. bagging, boosting, and stacking).

The use of unsupervised machine learning (ML) algorithms can be helpful in identifying and grouping accounts with similar characteristics based on their profile descriptions. This can aid in identifying potential fake profiles, as well as providing insight into the types of accounts that are associated with the supplied profile name. Unsupervised ML algorithms do not require labeled data to learn from, unlike supervised learning algorithms. Instead, they analyze patterns and relationships within the data to identify clusters or groups of similar data points.

Agglomerative clustering is a type of hierarchical clustering algorithm that is used to group data points together based on their similarity. It is a machine learning (ML) algorithm used for unsupervised learning.

The algorithm starts by treating each data point as a separate cluster. It then iteratively merges the two closest clusters into a single cluster, until all the data points are contained in a single cluster.

The distance between clusters is typically measured using one of several distance metrics, such as Euclidean distance, Manhattan distance, or cosine distance. The choice of distance metric can have

a significant impact on the clustering results, and it is often chosen based on the characteristics of the data being clustered.

The input parameters for agglomerative clustering are:

- **The data matrix:** a matrix of $n \times m$ dimensions, where n is the number of observations and m is the number of features.
- **The distance metric:** a measure of dissimilarity between observations. Common distance metrics include Euclidean distance, Manhattan distance, and cosine distance.
- **The linkage criterion:** a rule for combining clusters based on the distances between observations. Common linkage criteria include single linkage, complete linkage, and average linkage.
- **Threshold distance Or Number of Clusters:** a cutoff value for the distance between clusters. Clusters that are further apart than the threshold distance are not merged.

Agglomerative clustering[5] processes the data by iteratively merging the two closest clusters based on the chosen distance metric and linkage criterion. At each step, the distance matrix is updated to reflect the distances between the newly merged cluster and the remaining clusters. This process continues until all observations are contained within a single cluster.

Agglomerative clustering can present data in several ways. One common approach is to use a dendrogram, which is a tree-like diagram that shows the hierarchical structure of the clusters. The dendrogram starts with each observation as a separate cluster and shows how the clusters are merged at each step of the algorithm. Another approach is to assign each observation to a cluster label, which can be used for further analysis or visualization.

If a threshold distance is provided, agglomerative clustering will stop merging clusters once the distance between clusters exceeds the threshold. This can be useful for controlling the granularity of the resulting clusters or for identifying clusters at a specific level of similarity.

In the case of the fake profile checker, unsupervised ML algorithms can be used to analyze the profile descriptions of the scraped Twitter accounts and group them into clusters based on similarities in the descriptions. For example, if multiple accounts have identical profile descriptions or use similar phrases and keywords, they may be grouped together.

Once the accounts have been grouped into clusters, further analysis can be performed to determine whether the accounts within each cluster are legitimate or potentially fake. This can involve

examining the frequency and type of tweets, engagement rates, and other factors to determine whether the accounts exhibit suspicious behavior.

Overall, the use of unsupervised ML algorithms can enhance the fake profile-checking process by providing a more efficient and accurate way to identify and group accounts with similar characteristics. By leveraging the power of machine learning, the fake profile checker can quickly and effectively identify potential fake profiles on Twitter.

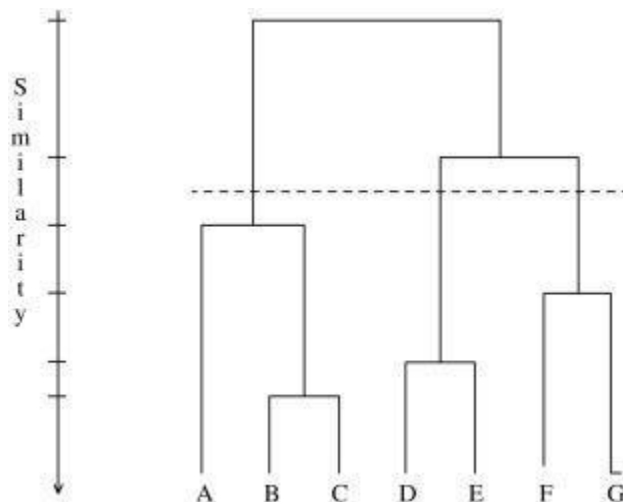


Figure 3: Agglomerative clustering algorithm

3.4 OSINT

OSINT (Open-Source Intelligence) is a process of collecting information from publicly available sources, including social media, online databases, and other open sources, to gather intelligence or conduct investigations. The fake profile checker algorithm can leverage OSINT techniques to further enhance the identification process of potential fake profiles.

Once the algorithm has grouped Twitter accounts with similar characteristics based on their profile descriptions using unsupervised ML algorithms, it can utilize the group's keywords to conduct a broader OSINT search on various open-source platforms, including Google.

The OSINT search can help to gather additional information about the group's members that may not be accessible on Twitter. This information can be used to verify the legitimacy of the accounts and identify any inconsistencies or red flags that may indicate potential fake profiles.

For example, if the group's keywords include "political activism" and "protests," the OSINT search may reveal news articles, social media posts, and other online content related to political protests or activism that the group's members have participated in. This information can be used to verify

the legitimacy of the accounts and determine whether they exhibit behavior consistent with real individuals or fake profiles.

Overall, the use of OSINT techniques can be a valuable tool in the fake profile-checking process. By leveraging open-source intelligence, the algorithm can gather additional information that may not be available on Twitter alone, which can help to improve the accuracy of the identification process.

3.5 Feedback System with Image Analysis

In the face recognition library by Ageitgey, the ResNet-34 architecture is used as a feature extractor for face recognition. The pre-trained ResNet-34 model is fine-tuned on a large dataset of faces using transfer learning and a triplet loss function to learn highly discriminative features for face recognition.

More specifically, the pre-trained ResNet-34 model is used as a convolutional neural network (CNN) feature extractor, which takes as input an image of a face and produces a feature vector, or embedding, that represents the facial features of the person in the image. The ResNet-34 architecture is well-suited for this task because of its deep and complex structure, which enables it to learn highly discriminative features at multiple scales.

The pre-trained ResNet-34 model is fine-tuned on a large and diverse dataset of faces, which The ResNet-34 architecture is used as a feature extractor for face recognition in the face recognition library created by ageitgey. The pre-trained ResNet-34 model is fine-tuned using transfer learning and a triplet loss function on a huge dataset of faces to discover highly discriminative features for facial recognition.

The pre-trained ResNet-34 model is used as a convolutional neural network (CNN) feature extractor, which takes an image of a face as input and generates a feature vector, or embedding, that represents the facial characteristics of the individual in the picture. The ResNet-34 architecture is well-suited to this job because to its deep and complex structure, which allows it to learn highly discriminative features across various scales.

Using transfer learning, the pre-trained ResNet-34 model is fine-tuned on a wide and diversified face dataset, including the VGGFace and FaceScrub datasets. Transfer learning is a deep learning approach in which a model previously trained on a big dataset is refined on a smaller dataset for a particular purpose. This helps the model to acquire task-specific characteristics more quickly and with less data than if it were trained from the beginning.

Training the model using a triplet loss function pushes the model to learn embeddings that minimize the distance between embeddings of the same person's face and increase the distance between embeddings of different people's faces. This assists the model in learning highly discriminative features for face recognition that are resilient to position, lighting, and other conditions.

Overall, employing the ResNet-34 architecture as a feature extractor for face recognition in conjunction with transfer learning and a triplet loss function allows the face recognition model in ageitgey's library to achieve excellent accuracy and resilience in identifying faces.

The pre-trained DNN model for face recognition was trained using both the VGGFace and FaceScrub datasets. The VGGFace dataset has over 2.6 million photos of over 2,600 individuals, while the FaceScrub dataset includes over 100,000 photographs of 530 individuals.

The VGG16 model was first trained using the ImageNet dataset, a large-scale image recognition dataset including over 1.2 million pictures over 1,000 categories. The photos in the collection depict a range of items, settings, and backdrops and are tagged with their respective class names.

The VGG16 architecture was created by the Visual Geometry Group (VGG) at the University of Oxford, and it was one of the top-performing models in the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC), an annual competition in which participants are tasked with classifying images into 1,000 categories.

The VGG16 architecture is comprised of sixteen convolutional layers, including five max pooling layers and three fully linked layers. The convolutional layers include modest 3x3 filters, allowing the model to acquire highly discriminative characteristics on a local scale. The max pooling layers reduce the spatial dimension of the feature maps and allow the model to acquire spatial invariance to tiny translations.

Overall, the VGG16 model has been shown to be an effective feature extractor for a variety of computer vision tasks, such as object identification, scene recognition, and picture captioning, and it has become a popular option for transfer learning in deep learning applications.

Using a feedback system that ranks accounts based on their profile images may assist refine the grouping of Twitter accounts and increase the accuracy of the false profile checker algorithm.

Utilizing deep learning models such as ResNet-34 and VGG16 to accomplish this feedback mechanism is one method. These models have been pre-trained on big-picture datasets and are capable of classifying photos based on their attributes and properties.

The ResNet-34 [5] is a deep residual network that can reliably and precisely categorize pictures. It employs skip connections to enhance the network's performance and has 34 layers.

The VGG16 model is a convolutional neural network designed for large-scale image recognition applications. It has sixteen layers and can accurately categorize photos into a thousand categories. In order to apply these models in the bogus profile checker method, the algorithm may take the profile images from Twitter accounts and run them through the ResNet-34 and VGG16 models. The models will then categorize the photos based on their attributes and allocate them to groups according to their similarities.

The computer may then utilize this information to improve account groupings based on profile images. For instance, accounts with similar profile photographs might be grouped together, whilst accounts with distinct profile pictures can be divided into separate groups.

Using deep learning models such as ResNet-34 and VGG16 may increase the accuracy of the false profile checker algorithm by refining the categorization of accounts based on their profile images. Includes the VGGFace and FaceScrub datasets, using transfer learning. Transfer learning is a technique in deep learning where a pre-trained model on a large dataset is fine-tuned on a smaller dataset for a specific task. This allows the model to learn task-specific features faster and with less data than training a model from scratch.

The fine-tuning process involves training the model with a triplet loss function, which encourages the model to learn embeddings that minimize the distance between embeddings of the same person's face and maximize the distance between embeddings of different people's faces. This helps the model to learn highly discriminative features for face recognition that are robust to variations in pose, illumination, and other factors.

Overall, using the ResNet-34 architecture as a feature extractor for face recognition, combined with transfer learning and a triplet loss function, enables the face recognition model in the ageitgey's library to achieve high accuracy and robustness in recognizing faces.

The pre-trained DNN model for face recognition is trained on a combination of the VGGFace and FaceScrub datasets. The VGGFace dataset contains over 2.6 million images of more than 2,600 people, while the FaceScrub dataset contains over 100,000 images of 530 people.

The VGG16 [6] model was originally trained on the ImageNet dataset, which is a large-scale image recognition dataset consisting of over 1.2 million images in 1,000 categories. The images in the

dataset have a wide variety of objects, scenes, and backgrounds, and are labeled with their corresponding class labels.

The VGG16 architecture was developed by the Visual Geometry Group (VGG) at the University of Oxford, and it was one of the top-performing models in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014, which is an annual competition where participants are challenged to classify images into 1,000 categories.

The VGG16 architecture consists of 16 convolutional layers, including five max pooling layers, and three fully connected layers. The convolutional layers have small 3x3 filters, which enables the model to learn highly discriminative features at a local scale. The max pooling layers reduce the spatial dimensionality of the feature maps and enable the model to learn spatial invariance to small translations.

Overall, the VGG16 model has been shown to be a powerful feature extractor for various computer vision tasks, including object recognition, scene recognition, and image captioning, and it has become a popular choice for transfer learning in deep learning applications.

The use of a feedback mechanism that sorts accounts according to their profile pictures can help further refine the grouping of Twitter accounts and improve the accuracy of the fake profile checker algorithm.

One approach to implementing this feedback mechanism is by utilizing deep learning models like ResNet-34 and VGG16. These models are pre-trained on large image datasets and can classify images based on their features and characteristics.

The ResNet-34 model is a deep residual network that can accurately classify images with high accuracy. It has 34 layers and utilizes skip connections to improve the performance of the network.

The VGG16 model is a convolutional neural network that has been trained on large-scale image recognition tasks. It has 16 layers and can classify images into 1,000 categories with high accuracy.

To utilize these models in the fake profile checker algorithm, the algorithm can extract the profile pictures of the Twitter accounts and pass them through the ResNet-34 and VGG16 models. The models will then classify the images based on their features and assign them to categories based on their similarities.

The algorithm can then use this information to refine the grouping of accounts by the profile picture. For example, accounts with similar profile pictures can be grouped together, while accounts with dissimilar profile pictures can be separated into different groups.

Overall, utilizing deep learning models like ResNet-34 and VGG16 can help to improve the accuracy of the fake profile checker algorithm by refining the grouping of accounts based on their profile pictures.

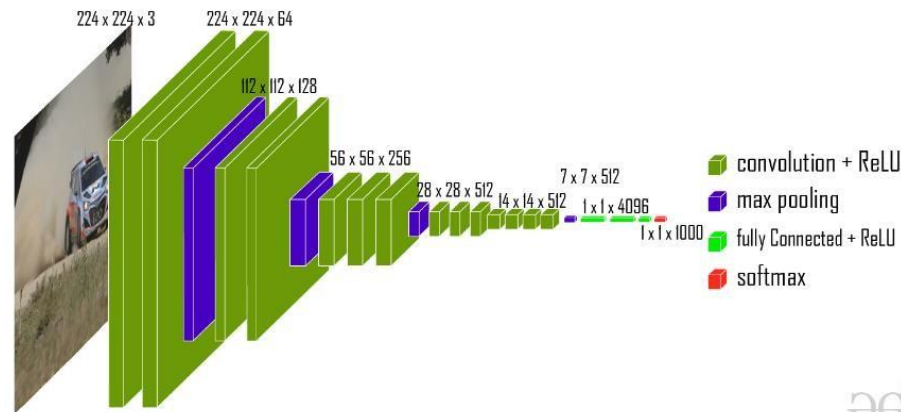


Figure 4: VGG-16 model.

3.6 Sentimental Analysis

VADER is a lexicon-based sentiment analysis tool designed to analyze and categorize sentiment in a social media text. It is a rule-based approach that uses a pre-built lexicon of words and phrases with associated sentiment scores to determine the sentiment of a given text. VADER uses a sentiment lexicon that is built specifically for social media text, which includes emoticons, slang, and other informal languages commonly used in social media. The lexicon contains over 7,500 lexical features, including positive and negative polarity, intensity modifiers, and degree modifiers. Each word or phrase in the lexicon is assigned a sentiment score between -1 (most negative) and 1 (most positive), with 0 being neutral.

To perform sentiment analysis using VADER, the text is first tokenized into individual words and phrases. Each word or phrase is then compared to the lexicon to obtain its sentiment score. The overall sentiment of the text is determined by aggregating the individual scores, taking into account the intensity and degree modifiers present in the text. The output of VADER is a sentiment score between -1 and 1, where negative values indicate negative sentiment, positive values indicate positive sentiment, and 0 indicates neutral sentiment.

VADER can also provide additional information about the sentiment of the text, such as the positive, negative, and neutral scores, as well as the compound score, which is a normalized score between -1 and 1 that represents the overall sentiment of the text. In addition, VADER can identify specific words and phrases in the text that contributed to the sentiment score.

Sentiment analysis is a process of analyzing text to determine the emotional tone of the author, such as whether they are positive, negative, or neutral. VADER is a lexicon and rule-based sentiment analysis tool that has been shown to be particularly effective in analyzing social media content.

By using VADER to analyze user activity, we can gain insights into the sentiment of tweets and other social media posts, as well as the frequency with which they are posted. This information can be used to identify accounts that may be engaged in spamming or other undesirable activity, such as posting excessively negative or inflammatory content.

For example, if we notice that a particular user account is posting a high volume of tweets with a consistently negative sentiment, this could be an indication that the account is engaging in spamming or other undesirable behavior. Similarly, if we notice that a user account is posting an unusually high volume of tweets in a short period of time, this could also be a sign of spamming or other automated activity.

Overall, using sentiment analysis with VADER can be a powerful tool for identifying potentially problematic user accounts and helping to maintain a healthy and positive social media environment.

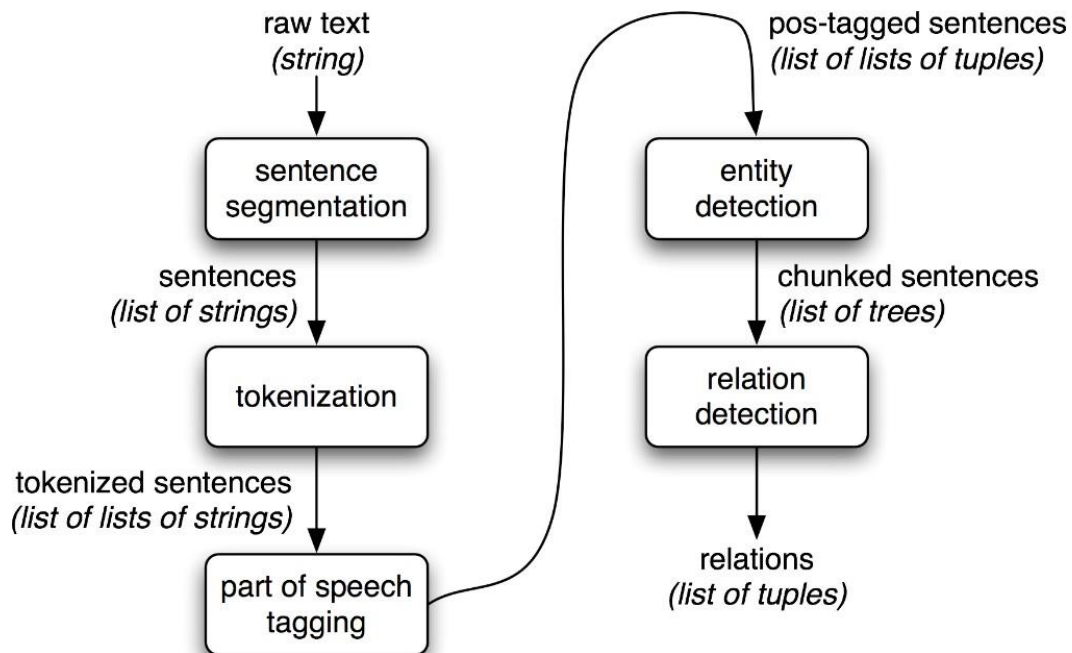


Figure 5: Natural Language toolkit

3.7 Botometer Data Collection

Botometer is an algorithm that is designed to identify Twitter bots, which are automated accounts that can be used to manipulate social media metrics such as followers, likes, and retweets. By using machine learning techniques to analyze patterns in account activity, Botometer can identify accounts that exhibit characteristics commonly associated with bots.

By gathering data from every user's Botometer across all groups, we can gain insights into the prevalence of bots within our social media ecosystem. This information can be used to detect suspected fraudulent accounts that may be manipulating their followers or interaction using bots. For example, if we notice that a particular account has a high Botometer score, indicating that it is likely to be a bot, this could be a sign that the account is engaging in spamming or other manipulative behavior. Similarly, if we notice that a group of accounts have unusually high levels of interaction with each other, this could also be a sign of bot activity.

Overall, by using Botometer to analyze user activity, we can gain insights into the prevalence of bots within our social media ecosystem and take steps to detect and address any suspected fraudulent accounts. This can help to ensure that our social media environment remains fair, transparent, and free from manipulation.

The selenium-based bot enters a list or file containing the username of the Twitter accounts into the Botometer online tool and gets important information about the account. the information we are scrapping for each account from Bototmeter is:

- **Bot Score:** the value between 0 – 5, 0 being not bot and 5 being for sure a bot. this value tells whether the entered account is a bot or not. This score is generated after passing through a machine learning model trained on 19 different datasets. (source bototmeter FAQ)
- **Echo-chamber:** accounts that engage in follow-back groups and share and delete political content in high volume.
- **Fake followers:** bots purchased to increase follower counts.
- **Financial:** bots that post using hashtags
- **Self-declared:** bots from botwiki.org
- **Spammer:** accounts labeled as spambots from several datasets
- **Other:** miscellaneous other bots obtained from manual annotation, user feedback, etc.
- **Recent tweets per week:** number of tweets done by the accounts.

- **Retweet ratio:** the percentage of the tweets by the account that are retweeted.
- **Most recent post:** the date of the latest tweet

Tweets by day of week give the frequency of the tweet done per day of the week.

Tweets by an hour of the day: frequency of the tweet done as per the time of the day.

3.8 Percentage Assignment

The data of users in each category will be compared, and each account will be assigned a percentage depending on the likelihood that it is a false profile. To estimate the possibility of a fake profile, the algorithm will analyze many parameters, including:

- Username score: using NLP model nltk, to give the similarity of the username of both of the accounts (the real and fake).
- Name Score: using NLP model nltk to give the similarity of the name of the both accounts.
- Location Score: calculating if the location of the account is the same or not.
- Description score: using NLP to detect how much the description of the real and the fake account match each other.
- Follower score: comparing the followers count of the fake and the real account.
- Following score: comparing the following count of the fake and the real count.
- Following/Followers ratio: calculation the ratio of the following and followers of the fake and the real account.
- Join date score: comparing the join date of the real and the fake account here the more the fake account's joining date is far from the real account the less the score. Also is the fake account joining date is before the real account then it gives a bonus score.
- Verification: checking if the fake or the real accounts are verified. If the fake account is verified, then it's given a bonus score. If both have similar status, then the average score whereas if real is verified and fake is into then the fake is given less score.
- Birthday score: comparing the birthdate of both account.
- Profession: check and compare the profession of both accounts using NLP, if it's the same or not.
- List score: compare the list count of both accounts in ratio.
- Likes score: compare the likes count of both accounts in a ratio.
- Tweets score: compare the no. of tweets of each account in a ratio.

- Website score: simple check if both have entered any valid website link or not. Its weight is very low.
- Bot score: as the fake account has already been scored with the Botometer so we pass the fake account Botometer parameters directly as score (Bot Score, Echo Chamber, Fake Followers, Financial, Self-Declared, spammer)
- Ratio of Bot score: these are the Botometer parameter (Bot score, Echo Chamber, Fake Followers, Financial, Self-Declared, spammer). Compares each Botometer parameter of both accounts.
- Most recent post: compare the most recent post of both accounts, this gives the idea of the activity of each account.
- Recent tweets per week: compare the recent tweets per week value of the real and the fake account.
- Retweet ratio score: compare the retweet ratio of the real and the fake account. Her ethe retweet ratio of the real account should be more than the fake.
- Tweets by the day of the week score: here we use cos similarity to find the similarity between the tweeting frequency of both accounts as per day of the week.
- Tweets by the hour of the day: here we use cos similarity to find the similarity between the tweeting frequency of both accounts as per hour of the day.
- Sentiment score: it compares the overall tweets sentiments of both accounts and gives a score.
- Positive score: compare the positive tweets ratio of both accounts.
- Neutral score: compare the neutral tweets ratio of both accounts.
- Negative score: compare the negative tweets ratio on both accounts.
- Picture matching score: check if the fake account and the real accounts have been marked as the same picture (data is pulled from the pic_grouping table, and the list of the respective group is checked if it contains this fake account username, if yes then the fake account is using the same picture as the real user).

3.9 Hate Speech Analysis Model

The purpose of a Hate Speech Analysis Model is to automatically identify and classify text as either hate speech or not. The model is trained using machine learning algorithms on a labeled

dataset of hate speech and non-hate speech text. Once trained, the model can be used to analyze new texts and determine whether they contain hate speech or not.

Hate speech can have serious negative consequences, such as spreading bigotry and hatred, inciting violence, and causing emotional harm to targeted individuals and communities. By automatically detecting hate speech, the model can help social media platforms and other online communities to remove or flag offensive content, protect users from harm, and promote a safe and inclusive online environment. Additionally, the model can assist law enforcement agencies in identifying potential hate crimes and in enforcing hate speech laws.

3.10 Results Display

On the website, the percentage of authenticity and user information will be published. This will help people make educated judgments about whom to follow and interact with on Twitter.

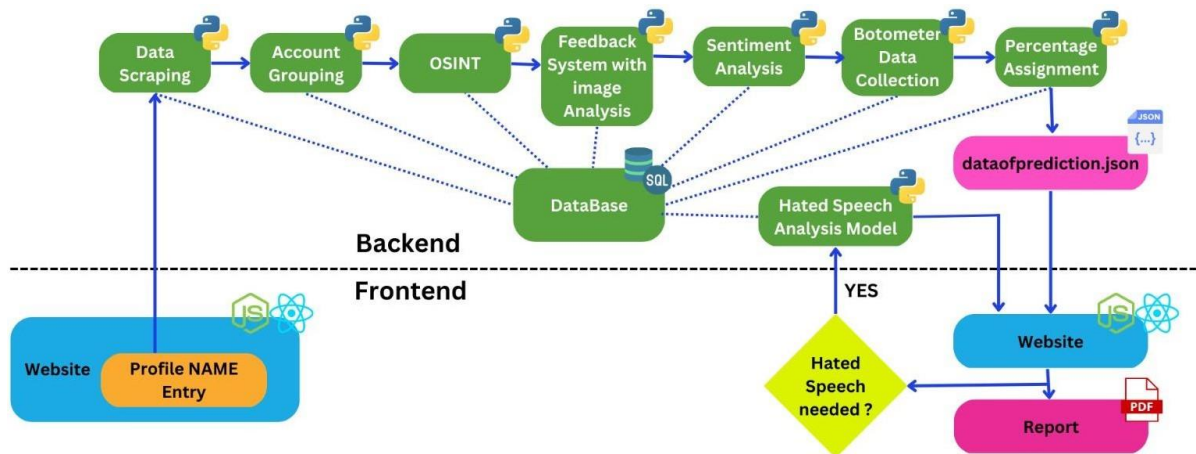


Figure 6: Working Model

Chapter 4

Detailed evaluation and analysis of the code

4.1 START_HERE.py

The program will be initiated by START_HERE.py. All the functions will be called from here.

To start, there is a function called "enteruser()" that is located in a Python module named "save1.py". This function takes two parameters: the user's full name and the depth in which to look for profiles. The function uses Selenium, which is a package that allows automated web browsing, to search for a user on Twitter based on their full name.

Twitter's algorithm then returns a list of profiles that match the provided name. The function extracts the usernames of the profiles from this list. These usernames are then passed to another function called "getusernames()" that is located in a different module named "User.py".

The "getusernames()" function also uses Selenium to scrape information from the profiles of the users whose usernames were provided. The information that is scraped includes the user's full name, location, description, birthday, number of tweets, number of likes, number of followers and following, verification status, profile picture, website, profession, and number of lists.

Once the data is scraped, it is stored in a database that is based on MYSQL. This database can be accessed later for further analysis of the scraped data.

Overall, this process allows for automated data collection and analysis of Twitter profiles, which can provide insights into user behavior and characteristics.

4.2 description1.py

The next step is to group profiles based on their descriptions.

The 'Des Clustering()' method applies agglomerative clustering on a collection of Twitter user profiles. Each dictionary in 'data1' provides information on a Twitter user's profile, such as their username, complete name, and profile description.

The program collects each profile's description, complete name, and username, then merges them into a single string named 'Desc1'. Additionally, it generates a set of dictionaries named 'Info', each of which includes the username and combined description text for a single profile.

The function then utilizes the scikit-learn library's 'TfidfVectorizer' to turn the list of description strings into a matrix of weighted feature vectors. The 'AgglomerativeClustering' method is then

performed to the feature vectors using the input parameter 'Num' to specify the number of clusters. If 'distance threshold1' is used, the algorithm will cease merging clusters when the distance between clusters reaches the specified threshold.

The method iterates over each cluster and determines the usernames of the profiles belonging to that cluster based on their respective description string. The resultant list of usernames for each cluster is saved together with a cluster label and an empty 'Full Name' field in a list of dictionaries named 'final list'.

The function then produces the 'final list', which has a list of dictionaries representing clusters and a list of usernames corresponding to each cluster.

In summary, the function handles the data by transforming the profile descriptions into a matrix of feature vectors, applying agglomerative clustering to the feature vectors, and then extracting the list of usernames for each cluster that is generated. The method provides a collection of dictionaries representing clusters and a list of related usernames.

The clusters are then returned to the main programme Description1.py, where the cluster leaders are initially or temporarily allocated based on the number of maximum followers or the verification status.

The description will also be evaluated, and only relevant data will be collected in order to execute OSINT for a specific cluster.

Then, all clusters will be stored in the database's "group info" table. The next step is to group profiles based on their descriptions.

The 'Des Clustering()' method applies agglomerative clustering on a collection of Twitter user profiles. Each dictionary in 'data1' provides information on a Twitter user's profile, such as their username, complete name, and profile description.

The program collects each profile's description, complete name, and username, then merges them into a single string named 'Desc1'. Additionally, it generates a set of dictionaries named 'Info', each of which includes the username and combined description text for a single profile.

The function then utilizes the scikit-learn library's 'TfidfVectorizer' to turn the list of description strings into a matrix of weighted feature vectors. The 'AgglomerativeClustering' method is then performed on the feature vectors using the input parameter 'Num' to specify the number of clusters. If 'distance threshold1' is used, the algorithm will cease merging clusters when the distance between clusters reaches the specified threshold.

The method iterates over each cluster and determines the usernames of the profiles belonging to that cluster based on their respective description string. The resultant list of usernames for each cluster is saved together with a cluster label and an empty 'Full Name' field in a list of dictionaries named 'final list'.

The function then produces the 'final list', which has a list of dictionaries representing clusters and a list of usernames corresponding to each cluster.

In summary, the function handles the data by transforming the profile descriptions into a matrix of feature vectors, applying agglomerative clustering to the feature vectors, and then extracting the list of usernames for each cluster that is generated. The method provides a collection of dictionaries representing clusters and a list of related usernames.

The clusters are then returned to the main programme Description1.py, where the cluster leaders are initially or temporarily allocated based on the number of maximum followers or the verification status.

The description will also be evaluated, and only relevant data will be collected in order to execute OSINT for a specific cluster.

Then, all clusters will be stored in the database's "group info" table.

4.3 USER_OSINT.py

Start User() This function initiates the feedback process for OSINT. For the preliminary grouping, input is necessary. This feedback is responsible for flagging the famous person's Twitter account and providing description clustering for those famous people. To do this, OSINT is carried out.

To verify the original Twitter account of a person, we must recognize that he or she is a celebrity. Google will only then include their data in the Knowledge panel. The issue is that while the Google knowledge panel includes the correct description of the person, it does not always include the correct Twitter id. For this, an algorithm is developed that verifies whether or not the celebrity has a Twitter account. If yes, what is his or her Twitter handle?

Start User() is a function that extracts and saves the data of Twitter ids (just the group header) in a list. This list of usernames is supplied to a function known as link ().

This function is responsible for displaying the complete names of the persons whose usernames are connected. In order to do this, we must first recognize that if there is a Google knowledge panel for a certain person, then there must be a corresponding Wikipedia page. OSINT is all about discovering vulnerabilities and exploiting them to get the desired data. These vulnerabilities target

SEO in particular. To get to the genuine famous person based on merely the Twitter user id handle, we must first input the Twitter id handle, the entire name of the account, and the Wikipedia text. "user-id handle plus Fullname plus Wikipedia." The link() method then just scrapes the first link on the page and scrapes the name from that name. This item is added to the list to search1[]. This is sent back to Start User ().

After the list to serach1 is acquired, the OSINT() method is called. Due to their notoriety, these individuals will always have a Google knowledge panel. From the Google knowledge panel, we can now get a description of these celebrities. For Twitter id, there are specific instances
In the knowledge panel, a Twitter account is listed.

There is no mention of a Twitter account in the knowledge panel.

This an extremely uncommon instance in which the referenced Twitter account is a hoax. Out of fifty attempts using OSINT, we have only discovered two such accounts after rigorous testing.

In order for OSINT to function with high precision, it is necessary to develop an algorithm that can operate in each of the aforementioned circumstances. Therefore, the OSINT followers() method is employed.

The OSINT followers() method is used to validate the Twitter followers of all investigated accounts. Through the account's followers, it is possible to determine if the account is legitimate or not. The issue is how to authenticate the followers. We use Google's knowledge panel to authenticate our followers. We search Google for the names of the first 20 persons (followers of the account under investigation) (first 20 followers that appear on the followers' page are to be confirmed since the first 20 followers shown will be the account's most prominent followers). If a Google knowledge panel appears, we know that the sought individual is renowned. Now we will examine the Twitter accounts listed in the Google Knowledge panel. If it is the same as the Twitter id of the follower. The account under investigation is genuine. If the followers' Twitter id handle and the Google Knowledge Panel's Twitter id handle do not match, the understudy account is not valid, and a list of actual Twitter ids is compiled (of people under study). If the Twitter ID is not validated, the list is added with the text "null." This indicates that the user does not have an official Twitter account.

4.4 affiliated_images_for_feature_matching(user_twitter_search, user_osint)

A function called `affiliated_images_for_feature_matching(user_twitter_search, user_osint)`

The parameters of the function are `user_twitter_search` and `user_osint`.

`User_twitter_search`(the string that contains the name that was entered by the user.): There exists a folder that is named after this string. It contains all the image files(profile pictures) of the Twitter accounts that appear after the search query is done on Twitter

`User_osint`(the string that contains the real name of the person under study): There exists a folder that is named after the osint search query that is stored as a string in the `user_osint` string.

The use of this function is to detect images that contain similar objects as the images affiliated images of the person under study. A list of the usernames is returned after the function call

Thus the list of matched faces and duplicates along with the list of affiliated images are then obtained through a function called `filter()` This function saves the 3 lists list of matched faces, the list of duplicate images, and the list of affiliated images in a way that the list of matched faces and list of duplicates is save din one list ajd the affiliated image list is separated. These lists are then saved in MySQL.

4.5 Tweet_Analysis.py

The program is named `Tweet_Analysis.py`. this function will be called after `pics_Grouping`. Its input parameters will be a list of usernames. Then we will use each username and using the `snsrape` module to scrape the tweets of the profile. The number of tweets to extract can be adjusted accordingly. `Snsrape` uses Twitter's API to extract tweets and return the tweet's data in the form of a dictionary containing: `tweet-id`, `tweet-text`, `datetime`, and `username`.

Then for each profile, the tweets scrapped will be pushed to the database table "`users_Tweets_Temp`".

For each profile we will make a list of tweets and pass it to the module which is VADER sentiment analysis.

To calculate the sentiment score for each word, the Vader model assigns a polarity score to each word in the tweet. The polarity score ranges from -1 to 1, where -1 indicates extremely negative sentiment, 0 indicates neutral sentiment, and 1 indicates extremely positive sentiment.

The model then combines the polarity scores of all the words in the tweet to provide an overall sentiment score for the tweet. The overall sentiment score ranges from -1 to 1, where -1 indicates extremely negative sentiment, 0 indicates neutral sentiment, and 1 indicates extremely positive

sentiment. In addition to the overall sentiment score, Vader sentiment analysis also provides three additional scores: positivity, negativity, and neutrality. These scores indicate the relative proportions of positive, negative, and neutral sentiments in the tweet.

For example, if a tweet has an overall sentiment score of 0.5, it indicates that the tweet has a positive sentiment. The positivity score may be 0.8, indicating that 80% of the sentiment in the tweet is positive, while the negativity score may be 0.1, indicating that only 10% of the sentiment in the tweet is negative. The neutrality score may be 0.1, indicating that the remaining 10% of the sentiment in the tweet is neutral. Then on each tweet, we will perform Sentiment analysis using VADER Sentiment analysis. Which will return the overall sentiment, positivity, negativity, neutrality of each tweet. Then for each person we will take average of each parameters then push the data to table "Sentiments".

4.6 Botometer.py

In this code we prepared a function named Botometer_analysis that takes six arguments (i.e. Input_user, List0_file1, Logs_Print, Print_detail, Output_Options, Do_Headless) the default input of the arguments are Logs_Print=1, Print_detail=1, Output_Options=0, Do_Headless=0, here the arguments means :

Input_user: enter the list or file address.

List0_file1: 0 for if you are sending List, 1 for if you are sending file address

Logs_Print,: 0 for not print logs, 1 for print logs.

Print_detail: 0 for NOT print the user scrapped detail, 1 for print the user scrapped detail.

Output_Options: 0 for getting a csv form data output file, 1 for pushing in SQL DB

Do_Headless: 0 for watching the bot working, 1 for headless.

The purpose of the function is to use the selenium-based bot to enter the list or file containing username of the twitter accounts into the Botometer online tool and get the important information about the account. the information we are scrapping for each account from bototmeter is:

- **Bot Score:** the value between 0 – 5 , 0 being not bot and 5 being for sure a bot. this value tells that that whether the enter account is a bot or not . This score is generated after passing through a machine learning model trained of 19 different datasets. (source bototmeter FAQ)

- **Echo-chamber:** accounts that engage in follow back groups and share and delete political content in high volume
- **Fake follower:** bots purchased to increase follower counts
- **Financial:** bots that post using cashtags
- **Self declared:** bots from botwiki.org
- **Spammer:** accounts labeled as spambots from several datasets
- **Other:** miscellaneous other bots obtained from manual annotation, user feedback, etc.
- **Recent tweets per week:** no of tweets done by the accounts
- **Retweet ratio:** the percentage of the tweets by the account that are retweeted
- **Most recent post:** the date of the latest tweet
- **Tweets by day of week :** gives the frequency of he tweet done per day of week
- **Tweets by hour of day :** frequency of the tweet done as per the time of the day

When the function is called the necessary libraries are imported (selenium) then the chrome drive is called. Then the Botometer is called using the website <https://botometer.osome.iu.edu/>. The bot then enters the login credentials which are the twitter account credentials. After that the username for which we want the result of the Botometer is inserted and then the necessary data is gathered as per requirement.

Botometer_SQL.py:

It uses the Botometer.py function Botometer_analysis to input the usernames of the currently group of the current session into the Botometer tool and get the result (mentioned in the botmeter.py) of the each usernames and stores it in the table userprofile_temp of the mysql database. Before this the function creates the necessary columns in the table userprofile_temp so that the data can be entered into it. This whole function takes place after the feedback module is executed and before the Percentage_giver so that the data is prepared for the Percentage_giver.

The columns created are :
 Bot_Score,Echo_Chamber,Fake_Followers,Financial,Self_Declared,Spammer,Most_Recent_Post,Likes_Count,Recent_tweets_per_week,Retweet_Ratio,Tweets_by_day_of_week,Sun,Mon,Tues,Wed,Thurs,Fri,Sat,Tweets_by_hour_of_day,12AM,1AM,2AM,3AM,4AM,5AM,6AM,7AM,8

AM,9AM,10AM,11AM,12PM,1PM,2PM,3PM,4PM,5PM,6PM,7PM,8PM,9PM,10PM,11PM.

(Note: these are all the values are get from the Botometer tool for the each entered user)

Percentage_giver:

It is the code that tags the real and fake account of each of the group in the session and then gives the percentage to each of the fake account using machine learning and nlp based on their resemblance to the real account of the group. Move over this function creates the final json file (named: dataofprediction.json) of the whole session each detail so that is can be used to display data on the website.

Functions:

The code consists of 11 functions i.e

tomongo() : converts the final json file containing all the information, including the prediction score into the mongodb db so that it can be used for displaying data on website.

extract_nouns_keywords(text): A function to extract nouns and important keywords from a string

similarity_score(str1, str2): A function to give a similarity score between two strings

weighted_percentage(data, weight): A function that gives the weightage to the data

date_diff(date_str): A function that gives the difference is two enter dates. the difference is time is seconds.

date_diff1(date_str): A function that give the difference between the current time and the enter date.

fan_detect_words(list): A function that detects whether the entered word is resembling with fan account and contains keywords that make the account fan account. The keywords can be : fan', 'fc', 'parody', 'fp'

fan_detect_letters(list): A function that detects whether the entered word contains any letter that resembling with fan account used letters.

fan_detect_overall(list): A function uses the “fan_detect_words” and “ fan_detect_letter_ function to return the list of string that are fan account associated.

Rater(real,real_df,fake_df): A function that gives the entered account scoring of being similar to the real account. It compares the real and the each fake account of the group on the basis of 38 different parameter . The value of each of the parameter of both , the real and the current account,

is calculated and is given values between 0 and 1. Following are the 38 different parameter with there description:

1. Usernames score: using nlp model nltk, to give the similarity of the username of both of the account (the real and fake)
2. Name Score: using nlp model nltk to give the similarity of the name of the both accounts.
3. Location Score: calculating if the location of the both account is same or not
4. Description score: using nlp to detect how much the description of the real and the fake account matches to each other.
5. Follower score: comparing the followers count of the fake and the real account
6. Following score: comparing the following count of the fake and the real count
7. Following/Followers ratio: calculation the ratio of the following and followers of the fake and the real account
8. Join date score: comparing the join date of the real and the fake account here the more the fake account joining date is far from the real account the less the score. Also is the fake account joining date is before the real account then its given bonus score.
9. Verification: checking if the fake or the real account are verified. If the fake account is verified then its given bonus score . If both have similar status then average score whereas if real is verified and fake is into then the fake is given less score.
10. Birthday score: comparing the birthdate of the both account
11. Profession: check and compare the profession of the both accounts using nlp, if its same or not.
12. List score: compare the list count of the both accounts in ratio
13. Likes score: compare the likes count of the both accounts in a ratio
14. Tweets score: compare the no. of tweets of each account ina ratio
15. Website score: simple check if both have entered any valid website link of not. Its weight is very low.
16. Bot score: as the fake account have already been scored with the botometer so we pass the fake account Botometer parameters directly as score (Bot Score,Echo Chamber,Fake Followers, Financial,Self Declared, spammer)

17. Ratio of Bot score: these are the Botometer parameter (Bot Score, Echo Chamber, Fake Followers, Financial, Self Declared, spammer) . Compares the each Botometer parameter of the both accounts.
18. Most recent post: compare the most recent post of the both account, this gives idea of the activity of each account.
19. Recent tweets per week: compare the recent tweets per week value of the real and the fake account
20. Retweet ratio score: compare the retweet ratio of the real and the fake account. Her ethe retweet ratio of the real account should be more than the fake.
21. Tweets by the day of the week score: here we use cos similarity to find the similarity between the tweeting frequency of the both account as per fay of the week.
22. Tweets by the hour of the day: here we use cos similarity to find the similarity between the tweeting frequency of the both account as per hour of the day.
23. Sentiment score: it compares the overall tweets sentiments of the both accounts and gives a score
24. Positive score: compare the positive tweets ratio of the both accounts
25. Neutral score: compare the neutral tweets ratio od the both account
26. Negative score: compare the negative tweets ratio o the both accounts
27. Picture matching score: check if the fake account and the real accounts have been marked as the same picture (data is pulled from the pic_grouping table, the list of the respective group is check if it contains this fake account username , if yes then the fake account is using the same picture as the real user)

All these parameter give a score between 0 and 1 and then is then weighted as per its importance . this weightage is prepared after testing and studying multiple research papers. Here are the weightages we have set (between 0 and 1, if you see more than 1 hence it is given bonus weightage):

- Username = 2.0
- Fullname = 2.0
- Location = 0.2
- Description = 1.0

- Followers = 1.0
- Following = 1.0
- F_F_ratio = 1.0
- joindate = 1.0
- verified = 10.0
- Birthday = 0.5
- Profession = 0.2
- Lists = 1.0
- Likes = 1.0
- Tweets = 0.7
- Website = 0.2
- Bot_Score = 2.0
- ratio1 = 0.5
- Echo_Chamber = 0.5
- ratio2 = 0.2
- Fake_Followers = 0.5
- ratio3 = 0.2
- Financial = 0.5
- ratio4 = 0.2
- Self_Declared = 0.5
- ratio5 = 0.2
- Spammer = 0.2
- ratio6 = 0.2
- MRT = 1.0
- Recent_tweets_per_week = 0.5
- Retweet_Ratio = 0.5
- Tweets_by_day_of_week = 1.0
- Tweets_by_hour_of_day = 1.0
- fan = 10

Now the percentage is generated based on these scores and is entered in the MySQL database. The percentage is entered in the table `userprofil_temp` across the fake account row under the column name "prediction". (note here the more the score, the more the fake account is mimicking the real account)

hence we can see that we not only analyzed the general information count (i.e followers, following) but also the pictures with similar, tweeting habits and frequency moreover fake followers (Botometer parameter), and bot activity to give the percentage of the fake account.

Note: all the above handle the null case such that if null is present for any parameter of the real account then automatically the weight of that parameter is dropped to zero hence this parameter does not contribute to the percentage of prediction and if the null is represented in any parameter of the fake account then that parameter score for the fake account is given 0 scores.

4.7 Percentage_giver()

This is the main function that is called when the percentage giver module is run. This function takes no arguments and uses the rater function in it. The purpose of this function is to make a proper data frame of the groups of the current session such that each group's each account is passed through the Rater function to be given a prediction score.

First of all, it connects to the database which has been updated after the feedback and osint results, also the Botometer values have been added to it. The function looks for the group in the `after_feedback_groups1_info` and pulls each row containing the list of group member usernames (under the usernames column) then three possible cases:

Case 1: If the real account of the group has been found using the osint tool.

Case 2: if the real account of the real is not found using the osint but there is a verified account in the group so consider it real.

Case 3: the real account is not found from osint moreover there is no verified account in the group hence we call a pre-trained ML model to find the real account from the group on the basis of genetic real and fake account behaviors.

To check if we have case 1, the code searches each username (from the groups' username list) in the `feed_back_groups1_info` table column `ID_twitter`, if there is any username from the list, in the column `ID_twitter`, this username account is considered real (this username is inserted in this column via osint previously)

To check if we have case 2, it is for sure that case 1 failed, hence now each username from the group list is searched in the userprofile_temp table and its verification column is checked, if we find any username with verification column 1 then it is considered real for the group.

If case 1 and case 2 fails then automatically case 3 is applied i.e machine learning model that is trained on multiple parameters is given all of the groups' user accounts to find the real and fake accounts and give a score of prediction

Note: in case 1 the real account is given the max real score hence we are sure that we have found the real account, for case 2 we are somewhat sure of the real account hence the score is less than the case 1 real account. in case 3 we are not sure if the found account is for sure real hence the real account is given the least real account score (here the real accounts prediction score will be 100,95,90 for case 1, case 2, and case 3 respectively, and is always greater than the fake accounts of the group)

after entering any of the three cases two data frame is prepared of all the details of each account of the group's usernames. One data frame is of real account names real_df and the second one is all the fake accounts of the group names fake_df.

Each data frame is passed to the rater to give the score, the returned score is then added back into the userprofile_temp table column prediction.

then all the results i.e () are added into the json file dataofpredction.sjon (it is a json file of the list of dictionaries of dictionary).

in the end, the data of the prediction file is sent to the function to Mongo dB to convert into a Mongo dB to be used to display data on the website.

4.8 Machine Learning Algorithm

Machine learning program that performs various tasks to train and evaluate a Random Forest Classifier algorithm on a dataset. The first step is to load a dataset from a CSV file using the pandas library. Then, the program preprocesses the data by converting the "Joining date" column to a datetime object and extracting year, month, and day features. The feature columns and target column are defined next.

The dataset is then split into training and testing sets using the train_test_split function from the sklearn.model_selection library. A Random Forest Classifier model is defined with 100 estimators, and it is trained on the training data using the fit method.

The model is then evaluated on the test data using the `accuracy_score` function from the `sklearn.metrics` library. Finally, the program makes predictions on new data using the trained model. The program preprocesses the new data by converting the "Joining date" column to a datetime object and extracting year, month, and day features. The program removes the Joining date from the feature columns as it is not necessary for prediction. The program then makes a prediction on the new data using the `predict` method of the trained model.

This section of the code starts by loading a CSV file called 'data.csv' into a pandas dataframe using the `read_csv()` method.

Next, it converts the 'join-date' column, which contains the date that a Twitter account was created, to a pandas datetime object using the `to_datetime()` method. This makes it easier to extract useful features such as year, month, and day from the date.

Then, it creates three new columns in the dataframe representing the year, month, and day that each Twitter account was created. This is done using the `apply()` method with a lambda function that extracts the corresponding attribute from the datetime object for each row in the 'Joining date of Twitter' column.

2. Defining Features and Target Variable

This section of the code defines the list of features and the target variable that will be used to train the machine learning model. The features are a combination of numerical and categorical variables related to Twitter accounts, such as bot score, number of likes, and joining date. The target variable is binary, with a value of 1 indicating a real account and a value of 0 indicating a fake account.

- 'Bot_Score',
- 'Likes',
- 'Retweet_ratio',
- 'Tweets_by_day_of_week_list',
- 'Tweets_by_hour_of_day_list',
- 'Following', 'Followers',
- 'Year of joining',
- 'Month of joining',
- 'Day of joining',
- 'tweetno'

3. Splitting Data into Training and Testing Sets

This section of the code uses the `train_test_split()` method from scikit-learn to split the data into a training set and a testing set. The `test_size` parameter is set to 0.2, which means that 20% of the data will be used for testing and 80% for training. The `random_state` parameter is set to 42 to ensure that the same split is obtained every time the code is run.

4. Training the Random Forest Classifier

This section of the code defines a random forest classifier with 100 trees and a random state of 42. It then trains the model on the training set using the `fit()` method.

5. Evaluating Model Performance

This section of the code uses the trained model to predict the target variable for the testing set using the `predict()` method. It then calculates the accuracy score, which represents the proportion of correctly classified instances in the testing set, using the `accuracy_score()` function from scikit-learn. The accuracy score is printed to the console.

6. Making Predictions on New Data

In this the algorithm predict the given id is real or fake and if real then the percentage of the realness of the account.

4.9 Hate_Speech()

This will be triggered only when the user wishes to perform hated speech analysis on the real profile in the end of program. The function 'Hate_Speech()' in `Speech_Detection.py` will take two parameters username, depth, and number of users who were at top in spreading hate tweets.

Then again using scraper we will scrape the tweets that are either involved in trend against username, or name of real profile, then again perform vader sentiment analysis on each of the tweet but this time we will only keep those tweets which are negative and keep the record of usernames. then we will highlight profiles having large number of hated tweets against real profile. And return the data in form of .JSON format where It will be displayed.

Data to be displayed::

1: trends in form of hashtags #username, #name.

2: top users. (then on each profile we can perform OSINT to tell if they are famous and real or not)

3: hated tweets.

4: starting ids.

4.10 Website

The website for the fake profile checker will have a frontend made using React, a popular JavaScript library for building user interfaces. The frontend will be responsible for displaying the results of the authenticity check to the user, including the percentage of authenticity and user information.

Once the authenticity check is complete, the backend will send the results to the frontend. The React components will then render the results on the screen in a user-friendly manner, making it easy for users to understand the authenticity of the Twitter account in question.

The percentage of authenticity will be displayed prominently on the website, giving users a quick and easy way to gauge the authenticity of the account. This percentage will be calculated based on the algorithm's analysis of the Twitter account's data, including the account's profile information, follower count, activity, and other pertinent information.

In addition to the authenticity percentage, the website will also display the user information that was collected during the data scraping process. This information may include the user's profile description, location, date of joining, number of followings, number of followers, and other pertinent details.

By providing this information to users, the website can help people make educated judgments about which Twitter accounts to follow and interact with. This can help to prevent users from unwittingly following or interacting with fake or fraudulent accounts, ultimately making Twitter a safer and more trustworthy platform.

Chapter 5

Conclusion

In conclusion, the detection of fake profiles on Twitter is a critical issue that has the potential to cause significant harm, including the spread of propaganda, manipulation of public opinion, and online bullying. The existing methods for detecting fake profiles are often unreliable, time-consuming, and expensive.

This project proposes a solution that uses machine learning and artificial intelligence techniques to accurately and efficiently detect fake profiles on Twitter. The proposed system has several benefits, including the ability to improve the accuracy and effectiveness of detecting fake profiles, which will help to create a safer online environment for all users.

The system also includes a hate speech detection model that will be useful in identifying and removing accounts that engage in harmful and discriminatory behavior. The project has academic objectives such as conducting research on the effectiveness of machine learning and artificial intelligence techniques in detecting fake profiles and creating a framework for future studies in this field.

The deliverables of this project include a working prototype of the fake profile detection system, a research paper detailing the methodology and results, a user-friendly website to access the system, and a comprehensive report on the hate speech analysis conducted.

Overall, this project is an important step towards improving the safety and integrity of social media platforms, and the proposed system has the potential to make a significant impact in the fight against fake profiles and online bullying.

Chapter 6

Future Work

The future work of this project can involve several aspects, including:

1. Integration with other social media platforms: Currently, the fake profile detection system is designed specifically for Twitter. However, it can be extended to other social media platforms such as Facebook, Instagram, and LinkedIn. This can be achieved by developing separate models for each platform or by using a unified model that can work across different platforms.

2. Continuous improvement of the detection model: The performance of the current model can be further improved by incorporating new features or by using more advanced machine learning algorithms. For instance, the use of deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can potentially improve the accuracy of the model.

3. Collaboration with social media companies: The fake profile detection system can be integrated with the APIs of social media companies, allowing them to monitor and remove fake profiles more efficiently. This can also involve collaboration with social media companies to share data and improve the accuracy of the detection model.

4. Integration with other security systems: The fake profile detection system can be integrated with other security systems such as intrusion detection systems and antivirus software to provide a comprehensive security solution for users.

5. Development of a user-friendly interface: The current system requires users to enter the name of the profile they want to check for fake accounts. However, a more user-friendly interface can be developed that allows users to easily scan their entire Twitter follower list for fake accounts.

Overall, the future work of this project can focus on improving the accuracy, efficiency, and usability of the fake profile detection system, and extending its capabilities to other social media platforms and security systems.

References

- [1] "Social Media Impersonation Detection & Removal", <https://smipin.com/impersonation-detection-removal>
- [2] "Zerofox impersonations solution", <https://www.zerofox.com/solutions/impersonations/>
- [3] "Botometer," <https://botometer.com>
- [4] P. Kumari, S. Singh, and S. Verma, "Fake profile detection on social networking websites: A ..." in IEEE Xplore, <https://ieeexplore.ieee.org/document/9373932>
- [5] "Image classification," Devopedia, <https://devopedia.org/natural-language-toolkit/>
- [6] "Agglomerative clustering," ScienceDirect, <https://www.sciencedirect.com/topics/computer-science/agglomerative-clustering/>
- [7] "Chatgpt," <https://chatgpt.com>

Fake Profile Detection On Multiple Platforms

ORIGINALITY REPORT

16%

SIMILARITY INDEX

5%

INTERNET SOURCES

12%

PUBLICATIONS

6%

STUDENT PAPERS

PRIMARY SOURCES

1	Pradeep Kumar Roy, Shivam Chahar. "Fake Profile Detection on Social Networking Websites: A Comprehensive Review", IEEE Transactions on Artificial Intelligence, 2020 Publication	8%
2	Submitted to Higher Education Commission Pakistan Student Paper	1%
3	Submitted to Infile Student Paper	1%
4	Submitted to Segi University College Student Paper	<1%
5	Submitted to University of North Texas Student Paper	<1%
6	www.cranfield.ac.uk Internet Source	<1%
7	"Sentiment Analysis and Ontology Engineering", Springer Science and Business Media LLC, 2016 Publication	<1%

8	www.umich.edu Internet Source	<1 %
9	Submitted to Cerritos College Student Paper	<1 %
10	Submitted to European University Student Paper	<1 %
11	Submitted to King's College Student Paper	<1 %
12	ritvik19.medium.com Internet Source	<1 %
13	www.ameerpet.org Internet Source	<1 %
14	Submitted to Middle East College of Information Technology Student Paper	<1 %
15	"Computer Vision - ACCV 2016 Workshops", Springer Science and Business Media LLC, 2017 Publication	<1 %
16	Submitted to University of Huddersfield Student Paper	<1 %
17	Submitted to University of London External System Student Paper	<1 %

"MultiMedia Modeling", Springer Nature, 2017

18

<1 %

19

Submitted to International University of
Malaya-Wales

Student Paper

<1 %

20

Submitted to Lebanese American University

Student Paper

<1 %

21

Submitted to Luton Sixth Form College,
Bedfordshire

Student Paper

<1 %

22

escholarship.org

Internet Source

<1 %

23

Submitted to Indiana University

Student Paper

<1 %

24

researchonline.ljmu.ac.uk

Internet Source

<1 %

25

Submitted to Texas A&M University, College
Station

Student Paper

<1 %

26

Submitted to Nottingham Trent University

Student Paper

<1 %

27

Submitted to UC, Irvine

Student Paper

<1 %

28

www.nature.com

Internet Source

<1 %

29

zhehou.github.io

Internet Source

<1 %

30

"The 2020 International Conference on Machine Learning and Big Data Analytics for IoT Security and Privacy", Springer Science and Business Media LLC, 2021

Publication

<1 %

31

Submitted to Napier University

Student Paper

<1 %

32

Submitted to University of Reading

Student Paper

<1 %

33

dzone.com

Internet Source

<1 %

34

www.ijiemr.org

Internet Source

<1 %

35

Ivan Srba, Gabriele Lenzini, Matus Pikuliak, Samuel Pecar. "Chapter 14 Addressing Hate Speech with Data Science: An Overview from Computer Science Perspective", Springer Science and Business Media LLC, 2021

Publication

<1 %

36

Submitted to University of Central Lancashire

Student Paper

<1 %

37

Submitted to University of Greenwich

Student Paper

<1 %

38

Submitted to University of Leicester

Student Paper

<1 %

39

Submitted to University of Westminster

Student Paper

<1 %

40

hdl.handle.net

Internet Source

<1 %

41

mdpi-res.com

Internet Source

<1 %

42

Submitted to AlHussein Technical University

Student Paper

<1 %

43

Submitted to University of Ibadan

Student Paper

<1 %

44

ece.vt.edu

Internet Source

<1 %

45

Submitted to Purdue University

Student Paper

<1 %

46

Submitted to University of Warwick

Student Paper

<1 %

47

medium.datadriveninvestor.com

Internet Source

<1 %

48

oliveai.com

Internet Source

<1 %

49

proceedings.neurips.cc

Internet Source

<1 %

	www.politesi.polimi.it Internet Source	<1 %
51	"Advances in Computer Vision", Springer Science and Business Media LLC, 2020 Publication	<1 %
52	Submitted to University of Surrey Student Paper	<1 %
53	bigdatakb.com Internet Source	<1 %
54	deepai.org Internet Source	<1 %
55	digitalrepository.unm.edu Internet Source	<1 %
56	journals.ametsoc.org Internet Source	<1 %
57	methods-sagepub-com-spjimrlibrary.knimbus.com Internet Source	<1 %
58	restaurantguru.com Internet Source	<1 %
59	www.coursehero.com Internet Source	<1 %
60	www.lmm.jussieu.fr Internet Source	<1 %

61	"Deep Learning-Based Face Analytics", Springer Science and Business Media LLC, 2021 Publication	<1 %
62	201-shi.yolasite.com Internet Source	<1 %
63	Adam Rehn, Aidan Possemiers, Jason Holdsworth. "Efficient hierarchical clustering for single-dimensional data using CUDA", Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '18, 2018 Publication	<1 %
64	Hua, Willian, and Yanqing Zhang. "Threshold and Associative Based Classification for Social Spam Profile Detection on Twitter", 2013 Ninth International Conference on Semantics Knowledge and Grids, 2013. Publication	<1 %
65	Submitted to University of Strathclyde Student Paper	<1 %
66	www.arxiv-vanity.com Internet Source	<1 %
67	www.mdpi.com Internet Source	<1 %

68

Derek C. Stanford. "Clustering or Automatic Class Discovery: Hierarchical Methods", A Practical Approach to Microarray Data Analysis, 2003

Publication

<1 %

69

Ahsan Baidar Bakht, Sajid Javed, Syed Qasim Gilani, Hamad Karki, Muhammad Muneeb, Naoufel Werghi. "DeepBLS: Deep Feature-Based Broad Learning System for Tissue Phenotyping in Colorectal Cancer WSIs", Journal of Digital Imaging, 2023

Publication

<1 %

Exclude quotes Off

Exclude matches Off

Exclude bibliography On