Analyzing the Behavior and Bot Detection among Twitter Users

Shaikh Sahil Ahmed
Information Technology
National Institute of Technology Karnataka
Mangalore, India
sahilahmed786001@gmail.com

Sampat Kr Ghosh
Information Technology
National Institute of Technology Karnataka
Mangalore, India
sampatghosh1995@gmail.com

Kushal Mondal
Information Technology
National Institute of Technology Karnataka
Mangalore, India
kushalmondal.192it008@nitk.edu.in

***Abstract*—In the field of social media, such as on Twitter, there are various accounts that are not genuine. So, it is difficult for a normal user to know about a genuine person. We are investigating the fundamental laws that drive the occurrence of behavioral similarities among Twitter users. We will use Machine Learning techniques to predict whether an account on Twitter is a Bot or a real user. We have performed a significant amount of feature engineering, along with feature extraction - selected features out of 20 helped us to identify whether an account is a bot or not. We implemented various algorithms that give better results also considering Area Under the Curve(AUC).**

***Index Terms*—behavior analysis, bot detection, machine learning.**

I. Introduction

Social media are powerful tools connecting millions of people across the globe. These connections form the substrate that supports information circulation, which ultimately affects the ideas, news, and opinions to which we are exposed. Social bots are accounts controlled by software, algorithmically generating content and establishing interactions. Many social bots carry out useful functions, such as spreading of news and publications and coordination of client activities. However, there is a expanding record of malicious applications of social bots. Some imitate human behavior to produce fake grassroots political support, promote terrorist information and recruitment, utilize the stock market, and spread rumors and plot theories.

A growing body of research is addressing social bot activity, its implications on the social network, and the detection of these accounts. In computer science, the various bot detection approaches typically apply machine learning based on account properties and/or tweet metadata. A typical method is to focus on classifying whether a user account is a bot or not. These attempts tend to make use of historical (timeline) data to compute properties like tweets per day, or statistical measures to identify periodic patterns in the tweeting behavior on an account level.

II. Literature Review

U. Ramakrishnan [1] described about new solution for the existing problem. They used Lexicon-Based approach and Statistical based approach as their methodology. They come to a conclusion whether the tweet is likely to be a positive tweet or negative tweet. The algorithm used by them gave accurate results but takes too long to execute them.

Bild [2] studied over aggregate user behavior and the retweet graph. They go through all the points such as Power Law Participation Momentum, Heavy-Tailed Rate Distribution, Heavy-tailed Interevent Distribution, Small-World, Assortative and Clustered Retweet Graph. They also described the structural differences from the followers graph that are more consistent with real world social networks.

Hana Anber [3] analysed over different information analysis techniques such as different hashtags, identification of influence, twitter's network-topology, event spread over the network, and analysis of sentiment. They described the techniques to identify the frequency distribution for each event.

Chithra R G [4] described to avoid the viral or unwanted or scary tweets from the media and this will helps to give the best recommendation towards the positive users through their experience in the social site. They clarified whether the person is passive, aggressive or passive-aggressive.

III. Methodology

The work used a collection of tweets made by groups of bots and genuine users to conduct experiments. The twitter data was taken and processed to make it eligible for quantitative analysis. The twitter data was categorized in several ways to support experiments of different types. Categorically the

twitter data was represented with sets of strings of chosen characters from a finite alphabet set. String analysis processes were applied to the sets of strings to get an insight to the tweets made by users and bots. The next phase of our work involves getting a quantitative result from the analysis. The whole implementation of the paper can be divided into 4 parts as Data collection, Data processing for analysis, Data visualization and Analysis. Further we visualized various techniques for bot detection. We used various machine learning models such as Decision Tree, Random Forest, etc which s described in the further subsections.

A. Data Collection

The dataset contains tweets and users data for one group of genuine users and three groups of twitter bots. There are more than 8.5 millions tweet data from more than 3000 genuine users. Three groups of bot users have more than more than 5 million tweet data from more than 3000 bots.

The tweet dataset is categorized into two parts. First part contained the columns 'user id', 'retweet status', 'in reply to'. This part was used to find the type of tweets each user made, if it was a tweet or retweet or reply to another tweet. Second part contained the columns associated with 'user id', 'hashtag numbers', 'url numbers' and 'mention numbers'. This part was used to identify whether the tweet made by a user had any urls or hashtags or mentions or it was just a normal text tweet. Thus, for each experiment we did not need most of the columns in the dataset at a time. That's why we created our own dataset in .csv format picking only the necessary columns from the main dataset. This made it easier for us to remove the inconsistencies and garbage values. We did this task for datasets of all 4 groups.

B. Data Processing for analysis

After streamlining and sanitizing the datasets we processed new datasets from it for our analysis. As mentioned earlier the first category of the dataset was used to find the type of tweets. Three letters from English alphabet were chosen to indicate the type of tweets: A = normal tweet, C = reply to a tweet, T = retweet. In the dataset, if the tweet was a reply it had a value of the user id to whom it was replied to in the column associated with 'reply to user id'; if the tweet was a retweet the column associated with 'retweet status id' had id of the status to which it was retweeted and if the tweet was a normal one then these two columns were blank. From this observation, we wrote a python script to check the type of a tweet and created a string containing only the characters A,C,T for each user. Then The string along with the user id was stored in a different csv file for further operations.

Similarly the second category of the dataset was processed. Here tweets were categorized by its content in two different levels. The first level had 3 types of contents and were indicated using three characters: N = tweet contains no entities, E = tweet contains one more entity, either hashtag, or mentions or urls, X = tweet contains entities of mixed types. In the dataset if the tweet had hash tags it had the count in the column associated with number of hashtags, if the tweet had a url it had the count in the column associated with number of urls, if the tweet had mention it had the count in the column associated with number of mention and if it was a normal text these columns had a value 0. From this observation we appended the python script to check the type of tweet and created strings with only the characters N,E,X for each user and stored in a different file.

The second category had an extended part where the tweets were divided into 6 types. N = normal text tweet, U = tweet that contains one or more urls, H = tweet that contains one or more hashtags, M = tweet that contains one or more mentions, X = tweet contains entities of mixed type. Therefore we created a string with characters N,U,H,M,X only for each user and stored in a different file. These preprocessing created new datasets that were used for the next step of implementation where we performed some visualization on the dataset and string matching operations.

C. Data visualization

We plotted histograms of string length against the number of strings for each group of users (1 group of genuine users, 2 groups of bot users) where length is represented on x-axis and number of strings on y-axis. The plotting provide us with some insights.

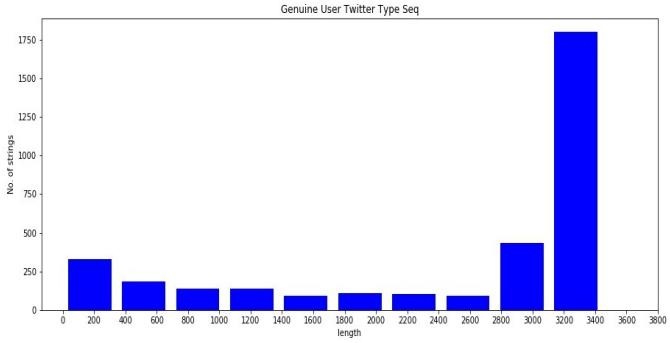

Fig. 1. Genuine user type seq.

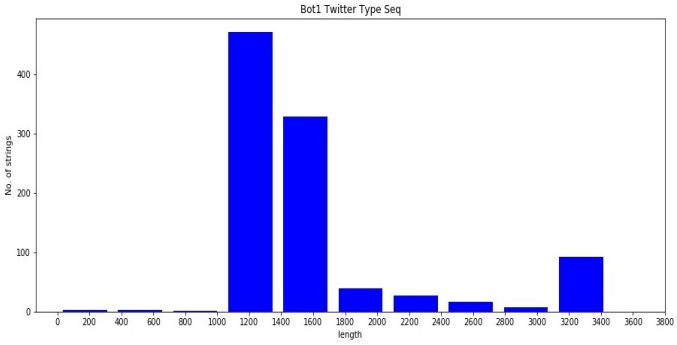

Fig. 2. Bot 1 user type seq.

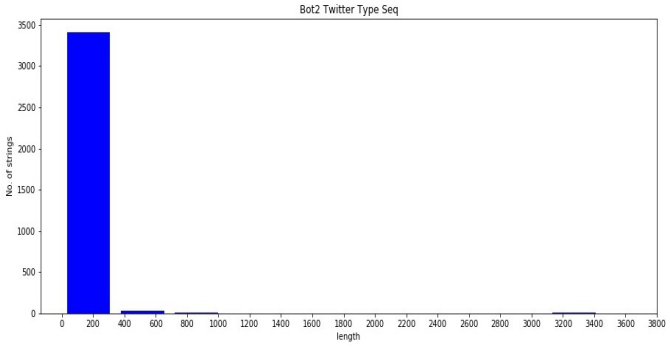

Fig. 3. Bot 2 user type seq.

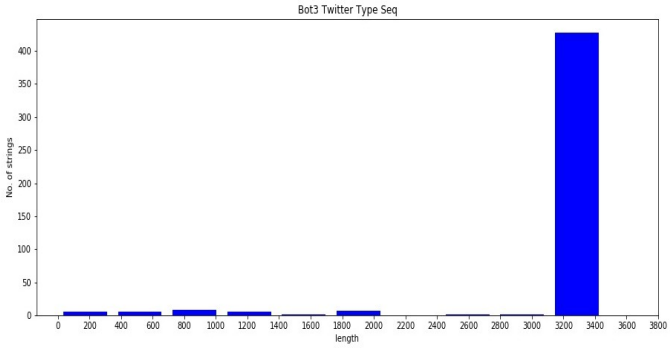

Fig. 4. Bot 3 user type seq.

The plot for genuine users group shows that nearly half of genuine users have a very high number of tweets. And the other half have a nearly uniform distribution of different numbers of tweets ranging from 100 to 3000. This is because of the presence of celebrity or highly interactive twitter users. The plot for genuine user type seq is shown in "Fig. 1", the plot for bot 1 user type seq is shown in "Fig. 2",the plot for bot 2 user type seq is shown in "Fig. 3" and the plot for bot 3 user type seq is shown in "Fig. 4". In case of bot users a very high number of bots are very active in twitter where other bots are not so active at all. This means only a few bots are dormant in twitter space where others are creating high number of contents or spams. Among the three bot groups bot 1 group is more similar to the genuine users. This kind of bot activity is harder to detect.

D. Analysis

We have tried to find the longest common substring (LCS) length among these strings of each group. The efficient approach to find the LCS has a time complexity of O(n+m), where n is the length of one string and m is the length of the string. This approach uses dynamic programming paradigm. When this approach is extended to k number of strings then the time complexity becomes O((n+m)*k). We have tried to find the length of longest of common substrings from the k value of 2 to n number of strings. Here we have faced a limitation of computing ability of our machines. For example, we had roughly 3400 strings of genuine users and trying to find the maximum of longest common substring of any 2 strings, we are required to check $\binom{3400}{2}$ combination of string pairs which equals 5778300 combinations. Each of these combinations roughly takes more than 6800 operations to get longest common substring length.

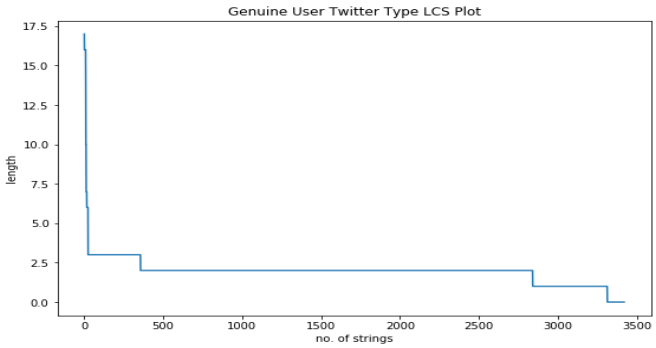

Fig. 5. Genuine user type Twitter LCS plot

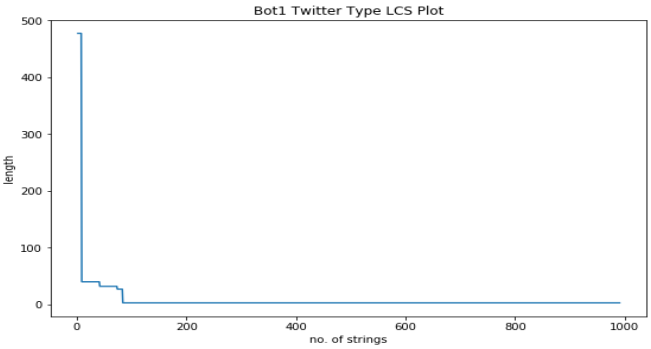

Fig. 6. Bot user type Twitter LCS plot.

The probability of finding the longest common substring (LCS) is very high in longer strings than the shorter ones since each string had very small number unique characters. We also observed that if L is the LCS of k strings then L is a superstring of the LCS of k+1 strings. Therefore we can feed the LCS, L of k strings to find the LCS of (k+1) strings by performing the LCS find operation on 2 strings only ie. (k+1)th string and L. This made the process faster to find the LCS among k number of strings. Although this approach assumes that the LCS will most probably among the longest strings and thus eliminate many combination of strings it produces a result. The genuine user type twitter LCS plot is shown in "Fig. 5" and bot user type twitter LCS plot is shown in "Fig. 6.".

E. Data Fitting and Error calculation

In this section we discuss about the analysis made on the LCS data points after modelling a curve on them. We study the nature of the curve and try to observe the nature of the model. We modelled data points from genuine and bot users and noted the parameters and calculated the RMSE (Root Mean Squared Error) for determining the best fitting. "Fig 7" shows data

fitting of Twitter Genuine User B3, ”Fig. 8” shows data fitting of Twitter Genuine User Content B3 type and ”Fig 9” shows data fitting of Twitter Genuine User Content B6 type LCS for genuine users shows exponential distribution.

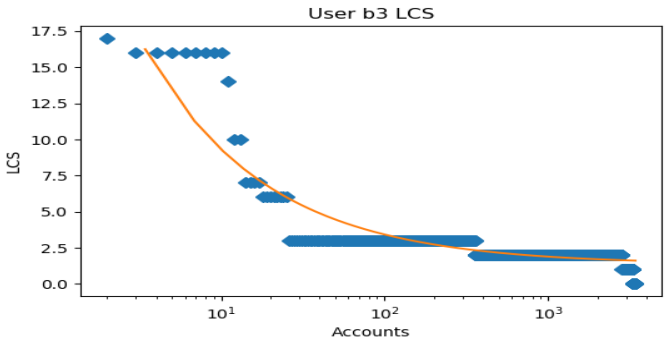

Fig. 7. Twitter Genuine User B3 type LCS fit

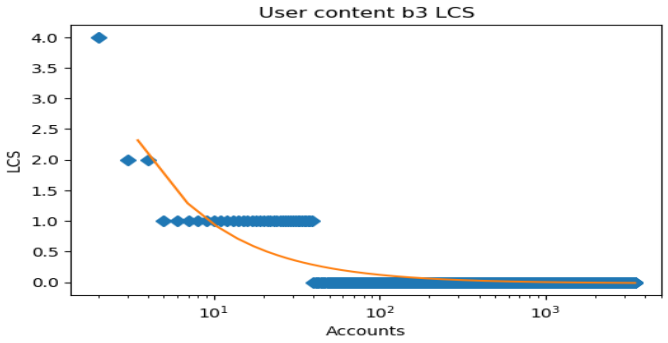

Fig. 8. Twitter Genuine User Content B3 LCS fit

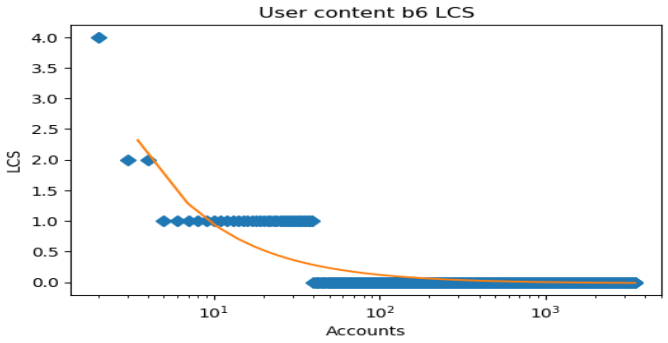

Fig. 9. Twitter Genuine User Content B6 LCS fit

”Fig 10” shows data fitting of Twitter Bot1 User B3, ”Fig. 11” shows data fitting of Twitter Bot1 User Content B3 type and ”Fig 12” shows data fitting of Twitter Bot1 User Content B6 type LCS for Bot users shows exponential distribution. The performance metrics for distribution of each type of users is shown in Table 1.

We observed that in case of genuine users LCS lengths over number of accounts fit an exponential model. We also

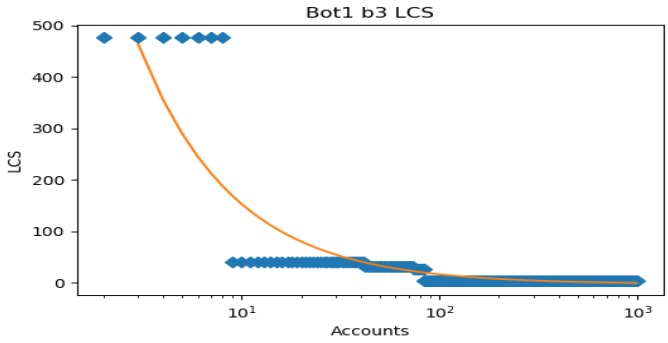

Fig. 10. Twitter Bot1 User B3 LCS fit

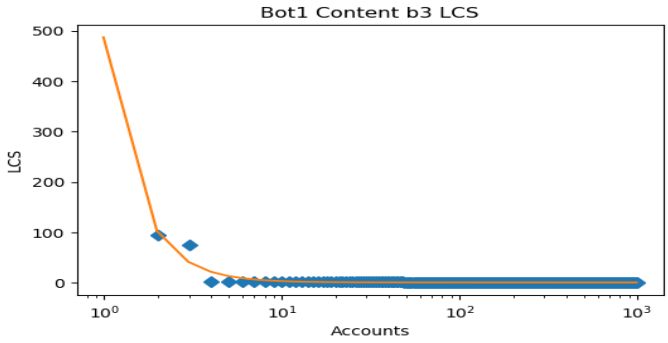

Fig. 11. Twitter Bot1 User Content B3 LCS fit

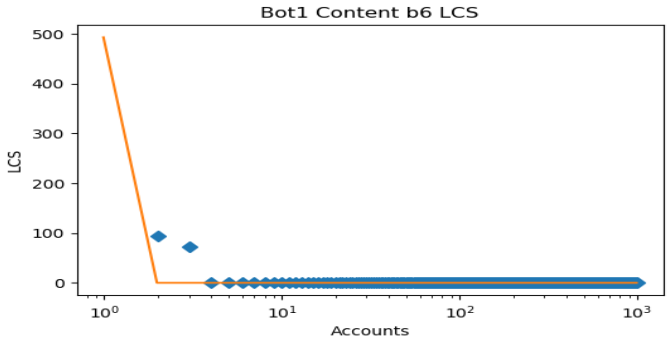

Fig. 12. Twitter Bot1 User Content B6 LCS fit

TABLE I
PERFORMANCE METRICS

Type of User	Mean	RMSE	Standard Dev.
Genuine User B3	0.58492	0.55002	1.35090
Genuine User Content B3	0.83897	0.06210	-0.01912
Genuine User Content B6	0.83897	0.06210	-0.01912
Bot1 User B3	0.91066	19.92882	-2.89101
Bot1 User Content B3	2.25318	1.39522	0.98916
Bot1 User Content B3	91.14581	3.78255	0.21414

observed that for genuine users the behaviour is heterogeneous where many users have several levels of similarities. On the other hand the Pareto or Power-Law model best represents the LCS length over number of accounts plot for bots. This shows that a large number of bots are not very active and shows homogeneous behaviour have less similarity where as comparatively smaller number of bots have large similarity and highly active. This observation gives us a reason to state that the similarity in the DNA string representation of the users' tweet content and nature of tweet(whether a tweet, reply or retweet) shows different behaviours for genuine and bot users. Although there can be bot users group that behaves fairly similar to genuine users group. Such bot users group will be harder to detect from a pool of users groups. For example Figure 13, 14 and 15 shows behaviour of LCS length over number of accounts of one such bot group.

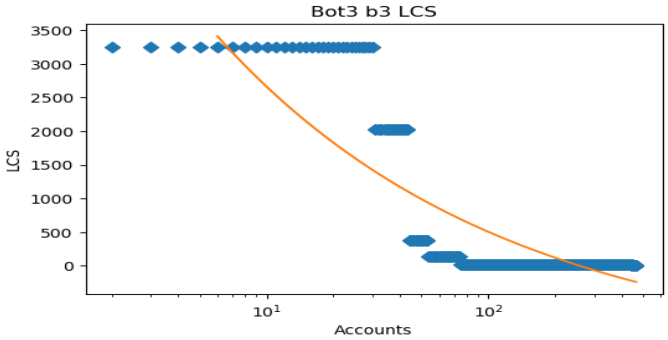

Fig. 13. Bot users(3) Twitter Type LCS fit

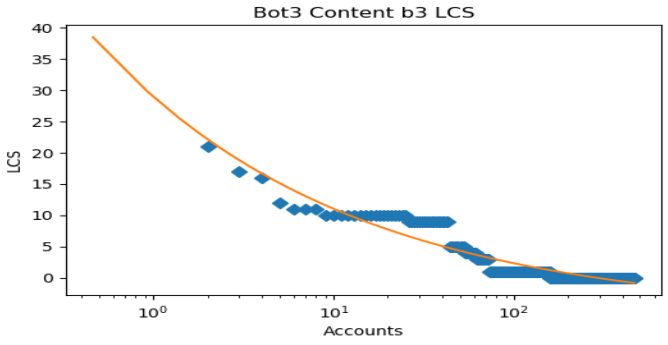

Fig. 14. Bot users(3) Twitter Content LCS fit

F. Bot Detection

We used the dataset as referenced in [27]. Various attributes present in dataset are id, id_str, screen_name, location, description, URL, followers_count, friends_count, listed_count, created_at, favorites_count, verified, statuses_count, lang, status, default_profile, default_profile_image, has_extended_profile, name, bot. The training dataset is chosen as "training_data_2_csv_UTF.csv" and the testing dataset is named "test_data_4_students.csv". Further we plotted the number of users as bot friends vs followers and non bot friends vs

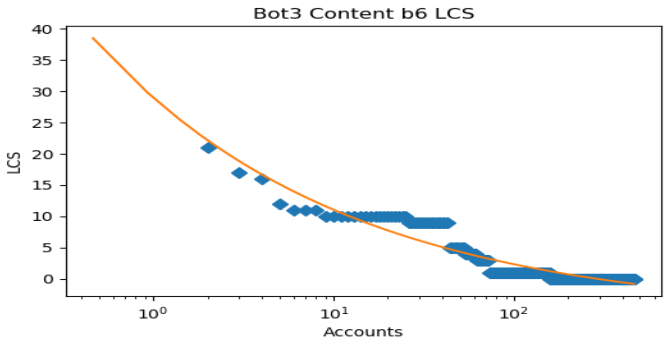

Fig. 15. Bot users(3) Twitter Content LCS fit

followers taking friends count on x-axis and followers count on y-axis. The plot for bot friends vs followers and for non bot friends vs followers is shown in "Fig. 16.". We used three

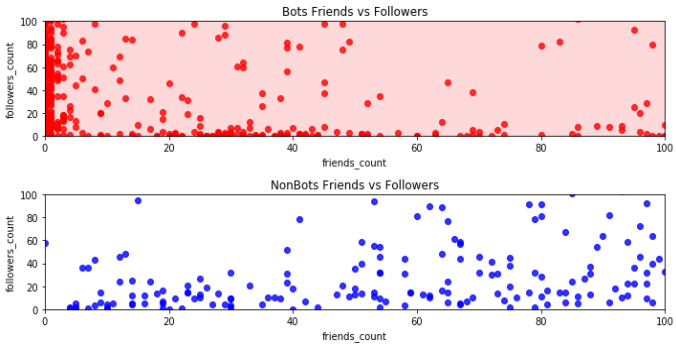

Fig. 16. Followers count

algorithms to go through the experiment. For bot detection among the twitter users, various machine learning techniques as Decision Tree, Multinomial Naive Bayes, Gaussian Naive Bayes, Bernoulli Naive Bayes, Random Forest, AdaBoost, Logistic Regression, and K Neighbors Classifier.

1) Decision Tree: Decision Tree is one the supervised learning method which is used for regression as well as for classification problems. A decision tree can be used to visually and explicitly represent decisions.

2) Naive Bayes: It is a family of probabilistic algorithms based on Bayes theorem with the "naive" assumption of conditional independence between every pair of a feature. We have considered three cases of Naive Bayes:

a) Multinomial Naive Bayes: It is used for discrete count values.

b) Bernoulli Naive Bayes: It is useful if the feature vectors are binary.

c) Gussian Naive Bayes: It is used in classification and it assumes that features follow a normal distribution.

3) Random Forest: Random Forests are used for regression and classifying problems that executes by constructing a swarm of decision trees during training phase.

4) AdaBoost: Multiple classifiers are combined to increase the accuracy of the classifiers. AdaBoost is also known as an iterative ensemble method.

5) Logistic Regression: It is used widely to inspect and elaborate the relationship between a set of predictor variables and binary response variable.

6) K Nearest Neighbors: It is a supervised machine learning algorithm which is used to solve regression as well as classification problems.

IV. Experiment and Results

A. LCS analysis and detection of groups

The heterogeneous and homogeneous behavior can be a good indicator for the type of the account groups. If we observe homogeneity it is a group of bots since they have similar behavior. On the other hand heterogeneity indicates human group since their natures are not centered. To capture this trend we calculate area under the curve(AUC) of the LCS curves we have found for each group and each type. To calculate the AUC we have taken the equation of the curve and performed definite integral on it with bounds set on range of numbers of accounts. A bigger value for AUC denotes homogeneity and a lower value shows heterogeneity. The calculated AUC are shown in table 2.

TABLE II
AUC MATRIX

Type of User	LCS Category	AUC
Bot1	Tweet type B3	8316.76
Bot1	Tweet content type B3	1138.01
Bot1	Tweet content type B6	211.78
Bot2	Tweet type B3	9405.27
Bot2	Tweet content type B3	3938.20
Bot2	Tweet content type B6	17.82
Bot3	Tweet type B3	129978.03
Bot3	Tweet content type B3	620.97
Bot3	Tweet content type B6	620.97
Genuine User	Tweet type B3	6674.83
Genuine User	Tweet content type B3	41.02
Genuine User	Tweet content type B6	41.02

After finding the AUC for each group of accounts and each type of categorization we find the average AUC for each specific account groups which gives us fruitful result. At table 3 we observe that human operated accounts have least average AUC whereas bot operated accounts have more AUC. This confirms the behavioral trend of both parties. Humans show more heterogeneous nature while bots show homogeneous nature. Thus we can establish that getting the LCS and then analysing them on the basis of homogeneity can detect a group of bots from a group of humans.

B. Bot Detection

There is no correlation between id, statuses_count, default_profile, default_profile_image and target variable. There is strong correlation between verified, listed_count, friends_count, followers_count and target variable. We cannot perform correlation for categorical attributes. So we will take

TABLE III
AVERAGE AUC FOR GROUPS

Type of User	Average AUC
Bot1	3222.1866
Bot2	4453.7688
Bot3	43739.9954
Genuine user	2252.2957

screen_name, name, description, status into feature engineering. While used verified, listed_count for feature extraction. We split the dataset "training_data_2_csv_UTF.csv" as 70% for training data and the remaining 30% as testing data and performed the experiments based on the mentioned algorithms.

1) Decision Tree Classifier (DT): After performing experiments we got Training Accuracy as 0.88707 and Test Accuracy as 0.87857. The plot for the Decision Tree ROC curve is shown in "Fig. 17".

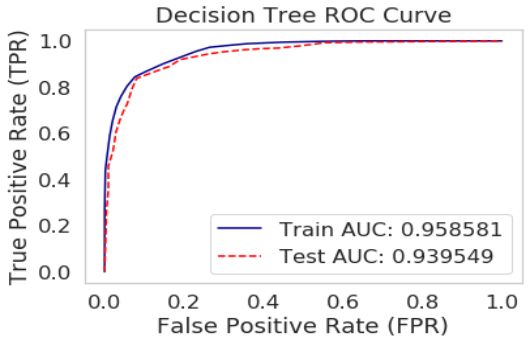

Fig. 17. Decision Tree ROC curve.

2) Multinomial Naive Bayes Classifier (MNB): After performing experiments we got Training Accuracy as 0.67961 and Test Accuracy as 0.69762. The plot for the Multinomial Naive Bayes ROC curve is shown in "Fig. 18".

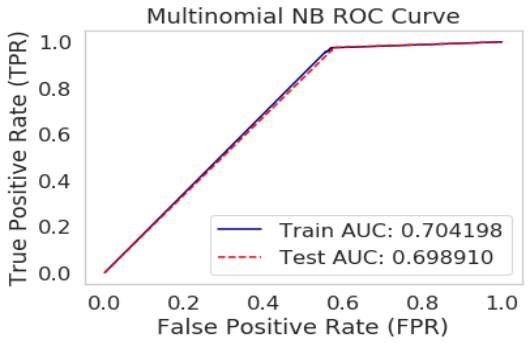

Fig. 18. Multinomial Naive Bayes ROC curve.

3) Bernoulli Naive Bayes Classifier (BNB): After performing experiments we got Training Accuracy as 0.80685 and Test Accuracy as 0.79643. The plot for the Bernoulli Naive Bayes ROC curve is shown in "Fig. 19".

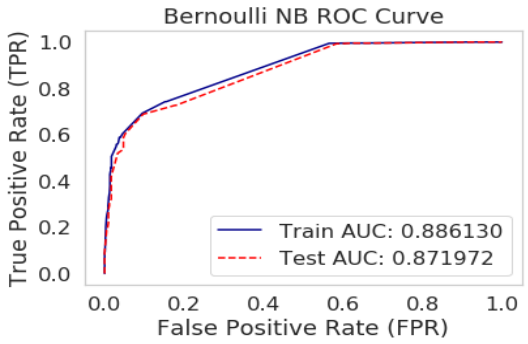

Fig. 19. Bernoulli Naive Bayes ROC curve.

4) Gaussian Naive Bayes Classifier (GNB): After performing experiments we got Training Accuracy as 0.61778 and Test Accuracy as 0.64643. The plot for the Gaussian Naive Bayes ROC curve is shown in "Fig. 20".

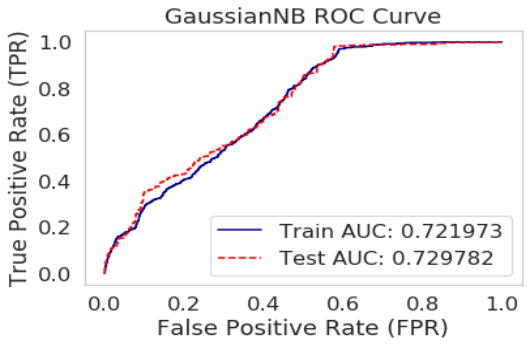

Fig. 20. Gaussian Naive Bayes ROC curve.

5) Random Forest Classifier (RF): After performing experiments we got Training Accuracy as 0.86765 and Test Accuracy as 0.85595. The plot for the Random Forest ROC curve is shown in "Fig. 21".

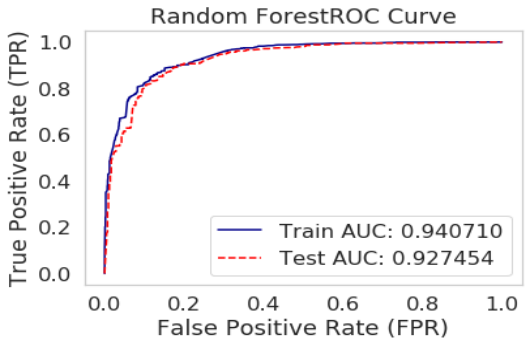

Fig. 21. Random Forest ROC curve.

6) AdaBoost Classifier (AB): After performing experiments we got Training Accuracy as 0.90393 and Test Accuracy as 0.86429. The plot for the AdaBoost ROC curve is shown in "Fig. 22".

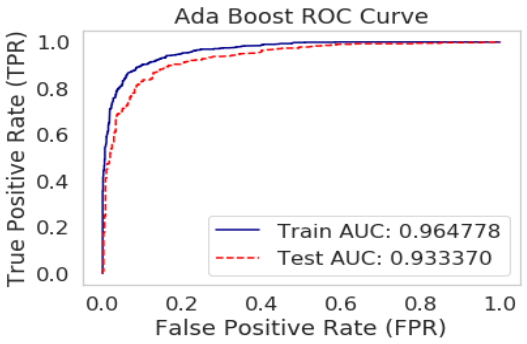

Fig. 22. AdaBoost ROC curve.

7) Logistic Regression Classifier (LR): After performing experiments we got Training Accuracy as 0.69494 and Test Accuracy as 0.70238. The plot for the Logistic Regression ROC curve is shown in "Fig. 23".

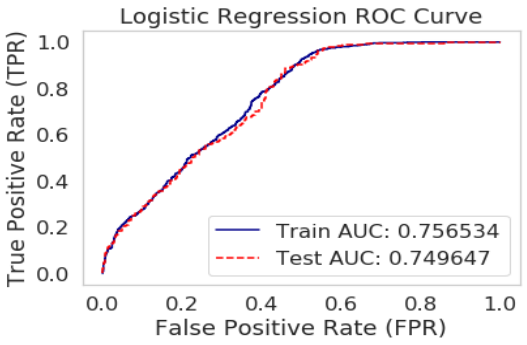

Fig. 23. Logistic Regression ROC curve.

8) K Nearest Neighbors Classifier (KNN): After performing experiments we got Training Accuracy as 0.86510 and Test Accuracy as 0.83214. The plot for the K Nearest Neighbors ROC curve is shown in "Fig. 24".

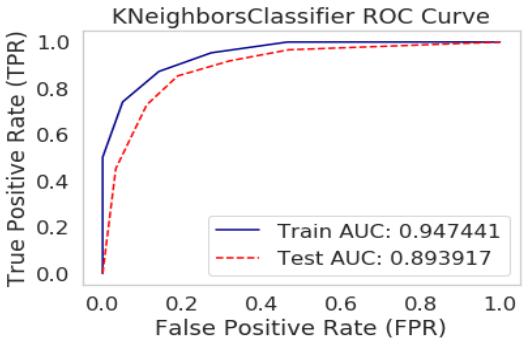

Fig. 24. K Nearest Neighbors ROC curve.

We found out the train and test accuracy as well as the train Area Under the Curve (AUC) and test Area Under the Curve for all the classifiers. The train and test accuracy as well as train and test Area under the curve is shown in Table IV.

TABLE IV
TRAIN TEST ACCURACY AND AUC

Classifier	Train Acc.	Test Acc.	Train AUC	Test AUC
DT	0.88707	**0.87857**	0.95858	**0.93963**
MNB	0.67961	0.69762	0.70419	0.69891
BNB	0.80685	0.79643	0.88613	0.87197
GNB	0.61778	0.64643	0.72197	0.72978
RF	0.86152	0.85714	0.93542	0.92685
AB	**0.90393**	0.86429	**0.96477**	0.93337
LR	0.69494	0.70238	0.75653	0.74964
KNN	0.86510	0.85714	0.94744	0.89391

After plotting all the classifiers, we found out various performance metrics such as Recall, False Positive Rate, Precision and F-measure. Various performance metrics for all the classifiers is shown in Table V.

TABLE V
PERFORMANCE METRICS FOR BOT DETECTION

Classifier	Recall	False Positive rate	Precision	F-measure
DT	**0.91122**	0.14879	0.83693	**0.8725**
MNB	0.62596	0.06217	0.97122	0.76127
BNB	0.875	0.2539	0.68825	0.77047
GNB	0.58522	**0.03676**	**0.988**	0.73505
RF	0.89182	0.17136	0.81055	0.84924
AB	0.88946	0.15742	0.82973	0.85856
LR	0.63402	0.10138	0.94724	0.75961
KNN	0.81651	0.15099	0.85371	0.8347

V. CONCLUSION AND FUTURE WORK

We addressed the problem of detecting Twitter bots. We got good accuracy using Decision Tree Classifier as compared to other classifiers as recall is 0.91122 and F-measure is 0.8725. Further Gaussian Naive is better as compared with others with respect to precision as 0.988. False positive rate for Gaussian Naive is least among all classifiers. We expect that the proposed work could perform well on other datasets and the relevant tasks is explained by the fact that it does not depend on any features related to Twitter, but builds on the very nature of bots. Such type of programs will be more predictable than humans every time, as long as the content is generated through automatic processes. In future, the models with optimization can be used to improve the predictability measurement and accuracy of bot detection.

REFERENCES

[1] U. Ramakrishnan, R. Shankar, Ganesha Krishna, "Sentiment analysis of twitter data: Based on user-behavior" in International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 7 (2015) pp. 16291-16301.

[2] Bild, D. R., Liu, Y., Dick, R. P., Mao, Z. M., and Wallach, "Aggregate Characterization of User Behavior in Twitter and Analysis of the Retweet Graph" in ACM Transactions on Internet Technology, Vol. 15, No. 1, Article 4, Publication date: February 2015.

[3] Hana Anber, Akram Salah, A. A. Abd El-Aziz, "A Literature Review on Twitter Data Analysis", International Journal of Computer and Electrical Engineering, Volume 8, Number 3, June 2016.

[4] Chithra R G, Harshitha G M, Anuprakash M P, Rakshitha H B, "Behavioural Analysis of Tweeter data : A Classification Approach" in International Journal of Engineering Research Technology (IJERT), ISSN: 2278-0181, 2019.

[5] B. Suh, L. Hong, P. Pirolli, "E. H. Chi. Want to be retweeted? Large scale analytics on factors impacting retweet in twitter network" In proc. of SocialCom, 2010.

[6] L. Uysal and W. B. Croft. "User oriented tweet ranking: A filtering approach to microblogs", In Proc. of CIKM, 2011.

[7] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. H. Chi., "Short and tweet: Experiments on recommending content from information streams" In Proc. of CHI, 2010.

[8] F. Abel, Q. Gao, G.-J. Houben, and K. Tao., "Analyzing temporal dynamics in twitter profiles for personalized recommendations in the social web" In Proc. of WebSci, 2011.

[9] C. Honeycutt and S. C. Herring. "Beyond microblogging: Conversation and collaboration via twitter", In Proc. of HICSS, 2009.

[10] Y. Liu, R. Cen, M. Zhang, S. Ma, and L. Ru, "Identifying web spam with user behavior analysis," in Proceedings of the 4th international workshop on Adversarial information retrieval on the web, ser. AIRWeb '08. New York, NY, USA: ACM, 2008, pp. 9–16.

[11] Y. Tang, S. Krasser, Y. He, W. Yang, and D. Alperovitch, "Support vector machines and random forests modeling for spam senders behavior analysis," in Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE, 2008, pp. 1–5.

[12] Y. Zhu, X. Wang, E. Zhong, N. Liu, H. Li, and Q. Yang, "Discovering spammers in social networks," in AAAI Conference on Artificial Intelligence, 2012.

[13] N. Dai, B. D. Davison, and X. Qi, "Looking into the past to better classify web spam," in Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web, ser. AIRWeb '09.

[14] C. P. De Wang, Danesh Irani, "A perspective of evolution after five years: A large-scale study of web spam evolution," International Journal of Cooperative Information Systems, vol. 23, no. 02, June 2014.

[15] P. Kolari, A. Java, T. Finin, T. Oates, and A. Joshi, "Detecting spam blogs: A machine learning approach," in AAAI, 2006, pp. 1351–1356.

[16] A. Mukherjee, B. Liu, J. Wang, N. S. Glance, and N. Jindal, "Detecting group review spam," in WWW (Companion Volume), 2011, pp. 93–94.

[17] C. Castillo, D. Donato, L. Becchetti, P. Boldi, S. Leonardi, M. Santini, and S. Vigna, "A reference collection for web spam," SIGIR Forum, vol. 40, no. 2, pp. 11–24, Dec. 2006.

[18] B. Wu and B. D. Davison, "Identifying link farm spam pages," in Proceedings of the 14th International World Wide Web Conference. ACM Press, 2005, pp. 820–829.

[19] "Spamrank - fully automatic link spam detection," in In Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb), 2005.

[20] B. Markines, C. Cattuto, and F. Menczer, "Social spam detection," in AIRWeb, 2009, pp. 41–48. New York, NY, USA: ACM, 2009, pp. 1–8.

[21] J. Preece, J. Lazar, E. Churchill, H. de Graaff, B. Friedman, and J. Konstan, "Spam, spam, spam, spam: How can we stop it," in CHI '03 Extended Abstracts on Human Factors in Computing Systems, ser. CHI EA '03, New York, NY, USA, 2003, pp. 706–707.

[22] D. Wang, D. Irani, and C. Pu, "A study on evolution of email spam over fifteen years," in Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on, Oct 2013, pp. 1–10.

[23] P. T. Metaxas and J. DeStefano, "Web spam, propaganda and trust," in AIRWeb, 2005, pp. 70–78.

[24] L. Becchetti, C. Castillo, D. Donato, R. A. Baeza-Yates, and S. Leonardi, "Link analysis for web spam detection," TWEB, vol. 2, no. 1, 2008.

[25] J. Abernethy, O. Chapelle, and C. Castillo, "Web spam identification through content and hyperlinks," in AIRWeb, 2008, pp. 41–44.

[26] D. Wang, D. Irani, and C. Pu, "Evolutionary study of web spam: Webb spam corpus 2011 versus webb spam corpus 2006." in Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2012, pp. 40–49.

[27] https://www.kaggle.com/charvijain27/detecting-twitter-bot-data/data.