A Study on Neural Network Models for Fake Profile Detection

Mr.Rachamalla Shyambabu

Assistant Professor, Department of CSE,
Malla Reddy College of Engineering for Women.,
Maisammaguda., Medchal., TS, India

Abstract—

We employ machine learning in the form of a synthetic neural network to assess the legitimacy of a friend request sent via Facebook. Furthermore, we detail the classes and libraries used in question. In addition, we explain the sigmoid function and its application in terms of the weights. Finally, the most crucial aspects of the social network page are taken into account in the proposed method.

Artificial neural networks, spotting fake profiles, social networks, malicious individuals are all discussed.

INTRODUCTION

With 2.46 billion active users as of 2017, Facebook is clearly the most popular social media platform. The information that users provide enables

1 social media platforms to generate income. The typical user has no idea that they are giving up rights by using a social networking site. The individual has nothing to benefit while social media giants like Facebook and Twitter earn a lot. Facebook generates income from adverts and user data every time a person posts information about their location, uploads a picture, expresses an opinion about a post, or tags other users in a post. The average quarterly revenue per user in the United States is $26.76

[2]. When multiplied by millions of users, the sum suddenly becomes significant.

As our daily lives become more and more reliant on computers, the typical person is more at risk of becoming a victim of crimes like hacking and identity theft. The victims of a data breach may not even be aware that they were attacked. There is currently minimal pressure on social networks to strengthen their data security. Hackers often aim at social media platforms like Facebook and Twitter. The banking sector is another potential target.

The hacking of social networking sites appears to be a daily news event. About 50 million Facebook users had their personal information compromised recently.

[3].

Facebook offers its users with a set of clearly specified provisions that describe the company's data handling practices

[4]. However, the policy does nothing to stop the persistent breaching of security and privacy. Facebook's safeguards seem to be circumvented by fake accounts.

Furthermore, bots and phony accounts provide a risk of personal information being taken fraudulently.

Software known as "bots" may be used to secretly observe a user's online behavior and collect data about them. Web scraping is the term for what you're doing. Furthermore, this behavior is not only acceptable, but also sanctioned by law. Bots may lurk in the background or pose as a friend request on social media in order to steal sensitive data.

This paper's proposed approach is meant to draw attention to the threats posed by a bot in the guise of a phony profile on your social media platforms. The answer would be coded into an algorithm.

Python is the programming language of choice. The system could tell the difference between a genuine friend request from a real person and a false one sent by a bot or someone just fishing for information. Since we need a training dataset from the social media firms in order to train our model and then check whether the accounts are false or genuine, our algorithm would need their assistance in order to function. The algorithm may also function as a browser add-on, adding a conventional layer to the user's online experience.

PART II: LOCATING THE CORE PROBLEM

To steal login credentials from unwary users, cybercriminals often construct phony accounts. A fictitious account will contact every person whose profile is publicly viewable by asking to be their buddy. These fake accounts employ photos of beautiful individuals to scam naive viewers. Upon the friend's acceptance, the fake profile's creator will begin sending out several requests to become friends with the friend's friends.

The content of the phony profile sometimes include links that take the victim to malicious third-party sites. A user's PC might be severely damaged if they are just inquisitive and click on the malicious link.

The price might range from being infected with a virus to having your machine transformed into a zombie by use of a rootkit. Even while Facebook employs stringent measures to prevent phony accounts from joining, all it takes is one to compromise the systems of many users.

Third, Related Papers

Late in 2012, a ranking graph-based method called Sybil rank was developed to effectively spot fraudulent profiles.

Spreading trust via early-terminated random walks [5]. It has an O-level computational complexity (nlogn). There is a ranking system in place for profiles based on a variety of factors.

messages and the people you've made connections with through time. Profiles with a high ranking are more likely to be authentic, while those with a low rank are more likely to be fraudulent. Unfortunately, it was discovered that this method was mostly inaccurate since it did not account for the potential that genuine accounts may be placed low and fraudulent profiles could be ranked high.

A new method, consisting of a series of stages, was presented by Sarode and Mishra [6] to identify bogus profiles. They accessed a large number of profiles using Facebook's graph API and then scripted out the process of extracting the data they found interesting. The characteristics used by the classification algorithm are derived from this collected data. The information is first stored in JSON and then converted to a structured format (CSV) more suitable for use with machine learning algorithms. The classifier will benefit in the future from these comma-separated variables.

Both unsupervised and supervised machine learning approaches were used by the writers. In this scenario, the accuracy of supervised machine learning approaches was over 98%. They created a training set and a test set for supervised machine learning. They trained the classifier using 80% of the data and validated it using the remaining 20%. After the algorithm has been executed, the profile receives feedback, at which point it must give identity to verify that it is not a fraudulent profile [6].

Features are extracted from a large number of profiles using a batch processing algorithm.

The categorization of false profiles uses the Resilient Back Propagation technique in neural networks in conjunction with support vector machines.

Sybil Frame performs hierarchical categorization.

Both content-based and structure-based strategies are available.

To derive the prior knowledge about nodes and edges, a content-based technique investigates the dataset and extracts relevant information. The structure-based method combines a Markov random field and a kind of belief propagation called loopy belief propagation to establish links between nodes. In the first phase of the Sybil Frame method, a content-based approach is employed, whereas in the second phase a structure-based approach is used.

In the first phase, we look at clickstreams and how people you know have been recommending things to you. Vote Trust employs a voting mechanism that aggregates user actions in order to identify phony profiles via the use of trust-based vote assignment and a global votes total. In spite of its limitations—including the sale of actual accounts that have been compromised—it is still regarded a first line of protection

. [7]

ITEM 4: A SUGGESTION FOR RESOLVING THE PROBLEM

To solve this problem, we use machine learning, specifically an artificial neural network, to calculate the likelihood that a friend request is genuine. The Sigmoid function is used to each equation at each neuron (node) to constrain the values to lie between 0.0 and 1.0. This might be multiplied by 100 at the output end to give us the probability that the request is malicious. For our purpose, we need simply deploy a single deep neural network, which would have a single hidden layer.

Each input neuron would be a different feature of each profile that has been chosen ahead of time and converted into a numerical value (for example, gender as a binary number, female 0 and male 1) and, if necessary, divided by an arbitrary number (for example, age is always divided by 100) to reduce the weight of any one feature. In this case, the neurons stand in for nodes. With this setup, each node would be in charge of making only one choice.

The decision-making process would be aided by the objects' relative importance and inherent biases. The result would be a percentage representing the likelihood that the friend request is fake. The used neural network is shown in Fig. 1.

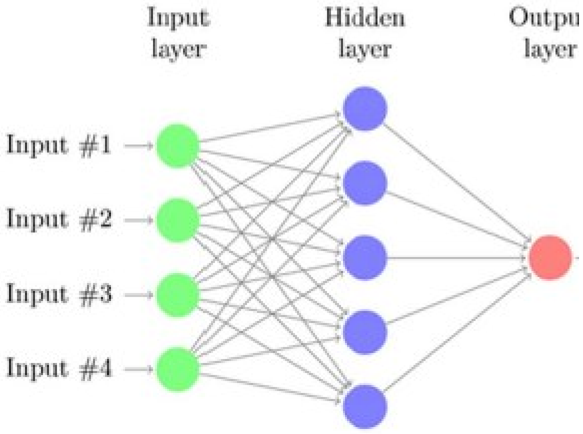

Figure 1: Neural Network.

Given that we identify enough false accounts, we may scrape data from Facebook or other social networking sites to use as a training data set. In this way,

In order for our deep learning algorithm to understand the patterns of bot behavior, we must minimize the final cost function by tweaking the weights and biases of each neuron and recalculating the equations. In this article, we provide an overview of the relevant libraries and classes.

V. RESULTS OF THE SIMULATIONS

I. Procedures A. Actualization

To facilitate our deep learning method, we've implemented it in Python. Pandas, NumPy, MATLAB, theano, scikit-learn, Keras, and TensorFlow are just some of the libraries that have been put to use.

Old and new fictitious profiles are kept in an Excel spreadsheet. Algorithm then saves the information in a data frame.

There will be a training set and a test set created from this data. In order to train our model, we'll need access to some data from the various social media platforms.

In order to identify false profiles in the training set, we employ the following features: account age, gender, user age, link in the description, number of messages exchanged, number of friend requests received, entered location, location by IP, fake or not. All of these factors are evaluated, and their results are recorded.

When training a model, it is common practice to give a value of (1) to the Gender parameter if the profile can be reliably identified as female or male. The same method is used for a wide variety of settings. The place of birth is also considered.

However, the United States comes in at a close second

[8]. China has the most bot activity of any nation.

We have already established that the algorithm is multi-layered.

The buried layer, for instance, consists of 128 nodes. An input layer and an output layer are also included. A one-deep machine learning method only employs a single hidden layer. This structure is meant to resemble a neural network, thus the several levels. Here, we refer to it as an ANN

[9] since it is an artificial neural network. Artificial intelligence (AI) algorithms may utilize this technique to find answers to challenging situations. It is widely used in pattern recognition, facial identification, and the education of AI assistants like Siri, Alexa, and others. To mimic the behavior of a human brain, this model is used. Each parameter would be represented by its own node; for instance, the Age parameter would have its own node, while the Gender parameter might have another. An outcome (decision) is generated on the basis of the inputs given. As a result, the buried layer receives the inputs.

Specifically, the readCSV() function will be used to take in the data. We include this into our training data. To tidy up the data, we use the dropna() function.

While developing an AI, proper data entry ranks high on the list of priorities. To make use of these parameters, we turn them into a numeric form, analogous to an enumerated data type. By dividing the Account Age by 52, we get the number of weeks since the account was created. The profile's real IP address is compared to the one given in the IP address field. One (1) is stored in locationMatch if a match is found. If there is no match, locationMatch will have the value zero (0). For Gender, we follow the same steps. We compared the link in the description to the link in the training set using the match() function. If the URL was discovered, a boolean value of true or false is stored in the url found variable. If true, the description parameter of the Link will have the value one (1) set to it. A zero (0) is issued if the condition is not met.

Following this, we divide the total number of messages by the account's age to get the value for the Number of messages sent out parameter. The parameter Number of friend requests issued is then calculated by dividing this total by the account's age. To modify the data set's columns, we utilize the Iloc() function. Both the input and output sets are processed using this manner.

Then, we randomly choose to divide both the input and output sets in half. This will give us four unique collections. The first group of data is sent down the input line. The second half of the input set may be found in this variable. In this case, the second input set will be used for the input test. You may think of this variable as holding the first half of the input sequence.

The first batch of outputs has been routed to the output train. It stores the remaining half of the output set in a separate variable. The second group of outputs will be used for the output test. The first half of the output set may be found in this variable.

As a result, we use the standardscaler() class

[10]. By doing so, we may transform the information into a normal distribution with a mean of zero (0) and a standard deviation of one (1). For the input train parameter, we make use of the fitTransform() method, while for the input set parameter, we make use of the transform() method. It normalizes the information to fit the desired pattern.

The next stage in the process involves the usage of the Keras library's Sequential function to build the model. After initializing the Sigmoid function, the output layer is created using the add() method of the Sequential class. Subsequently, we use Stochastic Gradient Descent as an optimizer to assemble the model. After that, we use the fit() function to customize the neural network to our data collection.

The input test variable, which stores half of the input data, must be verified for correctness. To do this, we use the predict() function. The final tally is expressed as a percentage.

Our database now has a new column for this outcome, which is stored in the output pred variable. After that, the evaluate() function receives the input test and output test variables as input and output respectively. score is set to the final result of the evaluate() function. Whether a profile is genuine or not is determined by the score.

B. Libraries

Using various libraries, we can facilitate machine learning and data mining with relative ease. Python's ubiquity, compactness, and multiple ready-to-use modules make it ideal for mathematical models, making it the language of choice for AI development today.

Pandas

[11] is a great package for data analysis

[12]. Finance, economics, statistics, analytics, etc. are just a few of the many academic and professional sectors in which it finds application. Various datasets may be accessed, modified, and optimized with ease.

It is essential to properly prepare the datasets and identify the most important aspects for further usage. NumPy, another library, deserves a nod as wel

l [13]. Since we are working with a big quantity of numbers that are highly interdependent, NumPy may be utilized for scientific computing, and is most often employed for multi-dimensional matrix multiplication.

The readCSV() function is to be used to read the data in. This is included into the data used for training. To eliminate unnecessary information, we use the dropna() procedure. Building an ANN relies heavily on getting the data right. Afterwards, we make a numerical representation of the parameters, such as Account Age, Gender, etc., in the style of an enumerated data type. Using the Account Age, we get the number of weeks since the account was created. Our next step is to examine whether or not the profile's stated IP address matches the user's real IP address. If there is a matching place, locationMatch will have the value 1. LocationMatch will be set to zero (0) if there is no match. For the purpose of Gender, the same procedure is used. The match() function was used to compare the link in the description to the link in the description of the training set. The url found variable is then set to true or false, depending on the outcome of the query. If so, the description parameter of the Link will be set to the value (1). If it is not true, then a value of zero (0) will be issued.

After that, we divide the total number of messages by the account's age to get the value for the Number of messages sent out parameter. Next, we divide the total number of friend requests made by the account's age to get the value for the Number of friend requests made field. To alter the order of the columns in the dataset, we use the Iloc() function. Both the input and the output sets are analyzed using this technique.

Next, we do a halving operation on both the input and output sets. As a result, we will have four unique collections. The initial batch of input data is loaded into the input train. In this variable, you'll find the remaining data from the input sequence. The input test will use the second data set. The first half of the input set may be found in this variable.

The first set of outputs is loaded into the output train. Half of the output set is stored in this variable. The second group of outputs is the ones chosen for the output test. In this variable, you'll get the first half of the final result.

In order to standardize our measurements, we use the standardscaler() class [10]. Then, we may transform the information into a normal distribution with a mean of zero (0) and a standard deviation of one (1). Both the fitTransform() and transform() methods are applied to the input train and input set parameters, respectively. In doing so, the data is transformed into the desired form for dissemination.

The next phase of the process involves building the model with the help of the Sequential module included in the Keras library. In order to initialize the Sigmoid function and to create the output layer, we use the add() method of the Sequential class. Then, we use Stochastic Gradient Descent to optimize the model and build it. Then, we use the fit() function to customize the neural network to our data collection.

We need to verify the data in input test, which represents half of the whole input set. The predict() function allows us to do this. A percentage is then derived from the calculated value.

We create a new column in our data store to keep track of this information once it has been compiled and placed in the output pred variable. The evaluate() function receives its arguments from the input test and output test variables. score has been set to reflect the outcome of the evaluate() function. A profile's score is used to help establish its veracity.

REFERENCES

[1] https://www.statista.com/topics/1164/social-networks/

[2] https://www.cnbc.com/2018/01/31/facebook-earnings-q4-2017-

arpu.html

[3] https://www.cnet.com/news/facebook-breach-affected-50-millionpeople/

[4] https://www.facebook.com/policy.php

[5] Qiang Cao, Michael Sirivianos, Xiaowei Yang, and Tiago Pregueiro.

2012. Aiding the detection of fake accounts in large scale social online

services. In Proceedings of the 9th USENIX conference on Networked

Systems Design and Implementation (NSDI'12). USENIX Association,

Berkeley, CA, USA, 15-15.

[6] Akshay J. Sarode and Arun Mishra. 2015. Audit and Analysis of

Impostors: An experimental approach to detect fake profile in an online

social network. In Proceedings of the Sixth International Conference on

Computer and Communication Technology 2015 (ICCCT '15). ACM,

New York, NY, USA, 1-8. DOI:

https://doi.org/10.1145/2818567.2818568

[7] Devakunchari Ramalingam, Valliyammai Chinnaiah. Fake profile

detection techniques in large-scale online social networks: A

comprehensive review. Computers & Electrical Engineering, Volume

65, 2018, Pages 165-177, ISSN 0045-7906,

https://doi.org/10.1016/j.compeleceng.2017.05.020.

[8] https://www.enigmasoftware.com/top-20-countries-the-most-cybercrime

[9] pages.cs.wisc.edu/~bolo/shipyard/neural/local.html

[10] https://stackoverflow.com/questions/40758562/can-anyone-explain-mestandardscaler

[11] https://pandas.pydata.org

[12] https://www.tutorialspoint.com/python_pandas/index.htm

[13] http://www.numpy.org

[14] https://www.mathworks.com/products/matlab.html

[15] http://www.deeplearning.net/software/theano/

[16] https://scikit-learn.org/stable/

[17] https://keras.io

[18] https://www.tensorflow.org

[14] is a popular tool for doing matrix multiplications and for plotting and visualizing mathematical models for analysis. For similar purposes, the Theano library

[15] is also great for doing calculations on expressions containing arrays of data in several dimensions. The Scikit-learn (sklearn) machine learning package

[16] may be used for a variety of purposes, including data plotting and visualization, analysis, and mining. It is based on the Python libraries NumPy, SciPy, and matplotlib. Support vector machines, random forests, gradient boosting,

kmeans, and DBSCAN are just a few of the many classification, regression, and clustering methods available in Scikit-learn. Its origins may be traced back to 2007 when it was created as a Google summer coding project.

[17] and TensorFlow [18] are the two most important libraries. Keras may be used independently of any of these libraries (TensforFlow, CNTK, Theanoc)[16]. Facilitates rapid prototyping and experimentation. Keras's fundamental structure serves as a template for structuring layered data. Open-ended Neuro-Electronic Intelligent Robot Operating System (project ONEIROS) funded its first development. The Greek origin of the word Keras means "horn."

TensorFlow

[18] is another popular library that was originally built by Google's artificial intelligence (AI) team for internal usage. The library is currently available as open source software and is particularly well-suited to machine learning, especially when neural networks are involved.

VI. CONCLUSIONS

In this study, we use machine learning to analyze the likelihood that a friend request is genuine using an artificial neural network. At each and everyadjustment of neuronal weights and the final cost function bias. This document provides an overview of the various libraries and classes that involved. The sigmoid function and its application are also covered. utilized weights and how they were calculated. In addition, we take into account the most important aspects of a social network page essential to finding a solution A Sigmoid function is applied to every neuron (node). Here, we make advantage of a Facebook- or other social network-provided training data set. Thus, the described deep learning method might use backpropagation to understand the patterns of bot activity, thus reducing the