Volume 15 • Issue 3 • June 2021

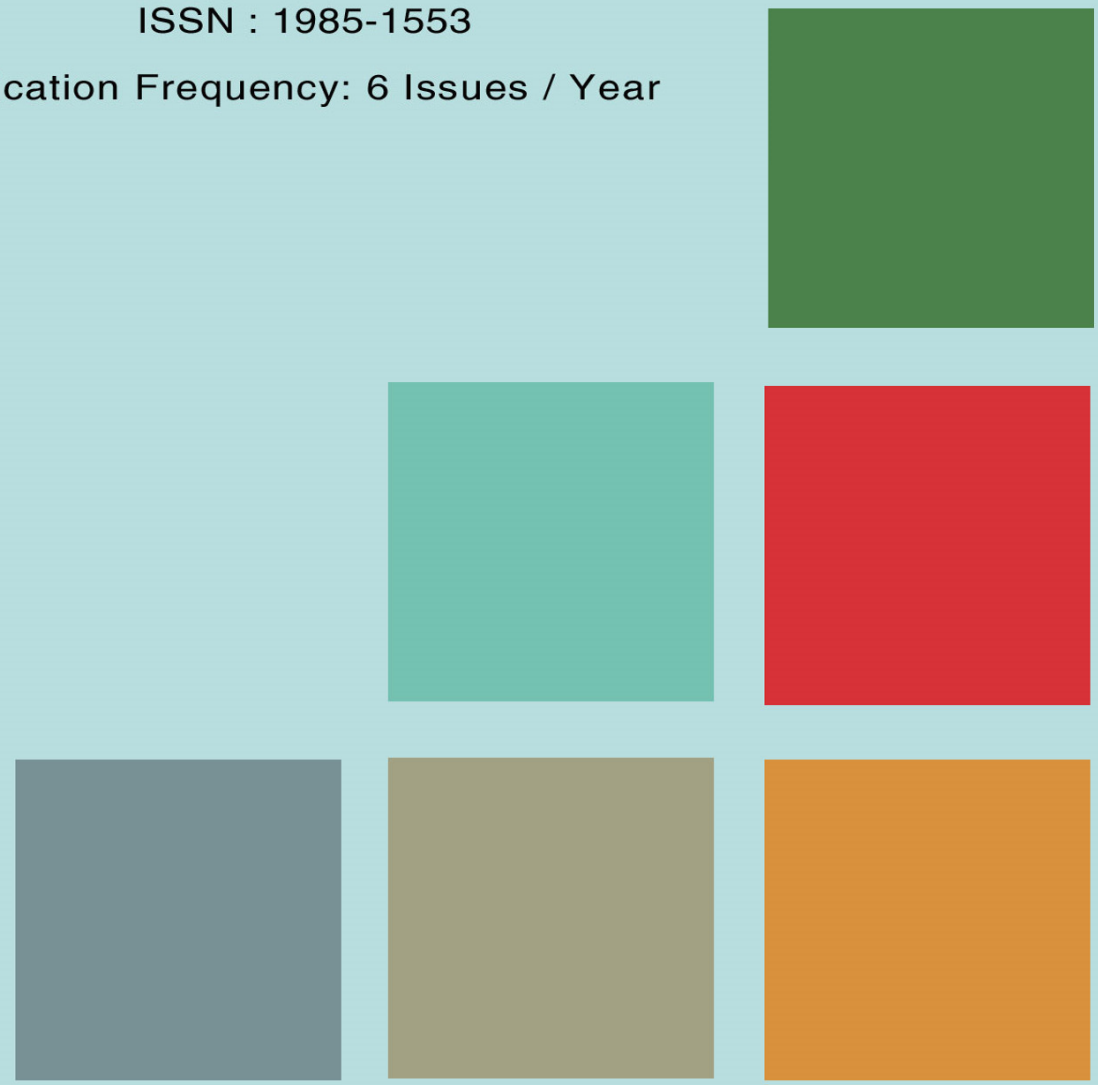
Editor-in-Chief
Dr. Chen-Chi Shing

INTERNATIONAL JOURNAL OF

COMPUTER SCIENCE AND SECURITY (IJCSS)

ISSN : 1985-1553

Publication Frequency: 6 Issues / Year



CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND SECURITY (IJCSS)

VOLUME 15, ISSUE 3, 2021

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 1985-1553

International Journal of Computer Science and Security is published both in traditional paper form and in Internet. This journal is published at the website https://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJCSS Journal is a part of CSC Publishers
Computer Science Journals
https://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND SECURITY (IJCSS)

Book: Volume 15, Issue 3, June 2021
Publishing Date: 30-06-2021
ISSN (Online): 1985 -1553

IJCSS Journal is a part of CSC Publishers
https://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2021

EDITORIAL BOARD

Dr. Sanaa Kaddoura
Department of Computing and Applied Technology, Zayed University
United Arab Emirates

Dr. Francesco Taglino
National Research Council
Italy

Dr. Rowanda Ahmed
Uskudar University
Turkey

TABLE OF CONTENTS

Volume 15, Issue 3, June 2021

Pages

EDITORIAL PREFACE

This is *Third* Issue of Volume *Fifteen* of the International Journal of Computer Science and Security (IJCSS). IJCSS is an International refereed journal for publication of current research in computer science and computer security technologies. IJCSS publishes research papers dealing primarily with the technological aspects of computer science in general and computer security in particular. Publications of IJCSS are beneficial for researchers, academics, scholars, advanced students, practitioners, and those seeking an update on current experience, state of the art research theories and future prospects in relation to computer science in general but specific to computer security studies. Some important topics cover by IJCSS are databases, electronic commerce, multimedia, bioinformatics, signal processing, image processing, access control, computer security, cryptography, communications and data security, etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 15, 2021, IJCSS appears with more focused issues. Besides normal publications, IJCSS intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

This journal publishes new dissertations and state of the art research to target its readership that not only includes researchers, industrialists and scientist but also advanced students and practitioners. The aim of IJCSS is to publish research which is not only technically proficient, but contains innovation or information for our international readers. In order to position IJCSS as one of the top International journal in computer science and security, a group of highly valuable and senior International scholars are serving its Editorial Board who ensures that each issue must publish qualitative research articles from International research communities relevant to Computer science and security fields.

IJCSS editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the Journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCSS. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJCSS provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Computer Science and Security (IJCSS)

Online Transaction Fraud Detection using Hidden Markov Model & Behavior Analysis

Niki Patel *patel104@cougars.csusm.edu*
Student / Computer Science and Information System
California State University San Marcos
San Marcos, 92096, USA

Yanyan Li *yali@csusm.edu*
Faculty / Computer Science and Information System
California State University San Marcos
San Marcos, 92096, USA

Ahmad Hadaegh *hadaegh@csusm.edu*
Faculty / Computer Science and Information System
California State University San Marcos
San Marcos, 92096, USA

Abstract

Card payment are mostly preferred by many for transactions instead of cash. Due to its convenience, it is the most accepted payment method for offline as well as online purchases, irrespective of region or country the purchase is made. Currently, cards are used for everyday activities, such as online shopping, bill pays, subscriptions, etc. Consequently, there are more chances of fraudulent transactions. Online transactions are the prime target as it does not require real card, only card details are enough and can be stored digitally. The current system detects the fraud transaction after the transaction is completed. Proposed system in this paper, uses Hidden Markov Model (HMM), which is one of the statistical stochastic models used to model randomly changing systems. Using Hidden Markov Model, a fraud transaction can be detected during the time of transaction itself and after 3 attempts of verification card can blocked at the same time. Behavior Analysis (BA) helps to understand the spending habits of cardholder. Hidden Markov Model helps to acquire high-level fraud analysis with a low false alarm ratio.

Keywords: Fraud Detection, Online Transactions, Hidden Markov Model, Behavior Analysis.

1. INTRODUCTION

Based on the Nielsen study for internet usage,100% of world's population uses internet and has seen a growth of 13,941% from year 2000 to 2020. 95% of global consumers have used online shopping method to make a purchase, according to Nielsen's Connected Commerce report. Globally the eCommerce rate has increased with a booming 600% since 2010 [1].

With the digital economy evolving rapidly, businesses of all sizes need to re-evaluate their position and tools when it comes to fraud management. Transaction fraud obliges genuine threats on e-commerce shopping. Due to increasing popularity of online transaction the types of online transaction frauds related with it are also escalating which disturbs the financial industry. Online payments are a prime target for fraudsters as they do not even need to have the real card, they only need the card details which can be stored digitally. It is also easier to get away with it because it's so much harder for the seller to verify who is really making the purchase.[2]

To overcome these problems numerous fraud detection techniques and algorithms have been proposed, data mining is used by many firms associated with fraud detection. But the data mining alone is not sufficient for detecting the fraud as it depends upon the data set containing history of

customer's transaction. The specifics of items bought by any individual transaction are generally not recognized to any Fraud Detection System (FDS) operating at the bank that issues credit cards. BA (Behavior Analysis) is executed for tackling this problem. An FDS operates at a credit card issuing bank. Each inbound transaction is presented to the FDS for authentication. FDS obtains the card details and transaction cost to confirm if the transaction is legitimate or not. The types of commodities bought in that transaction are unknown to the FDS. Bank rejects the transaction if it is found to be a fraud by FDS.

1.1. Problem with Existing System

As reported by Federal Trade Commission (FTC), 1,697,934 fraud reports were made in 2019, out of which 250,678, which is 15% of total frauds, were the reports with Payment Methods [3].

Figure 1 is the representation of frauds reported in various methods of payments with the corresponding total loss. Figure 1 illustrates that Wire Transfer and Credit cards accounts most of the frauds.

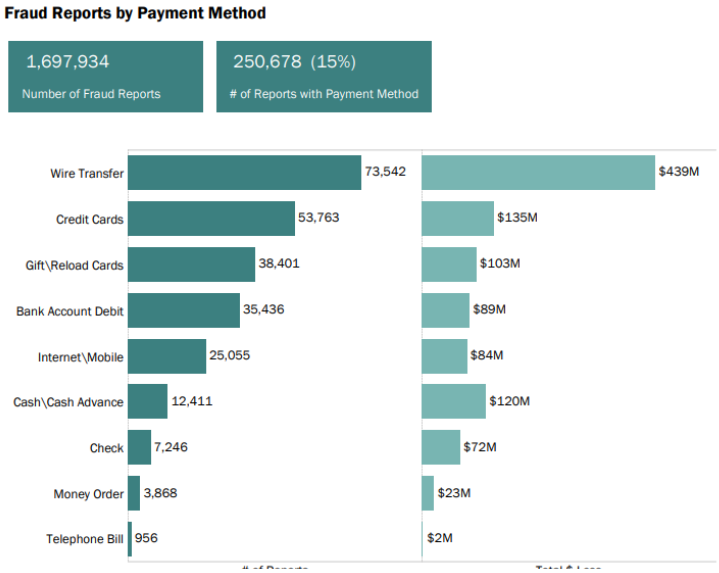


FIGURE 1: Wire Transfer and Credit cards accounts.

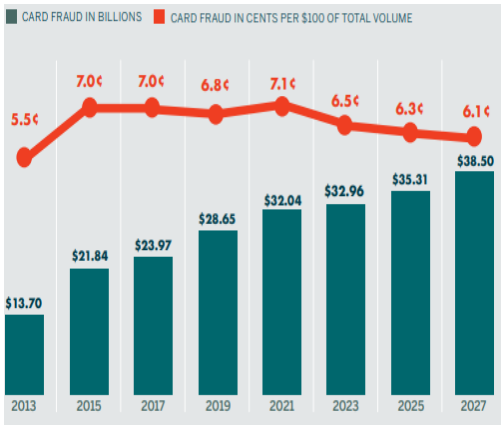


FIGURE 2

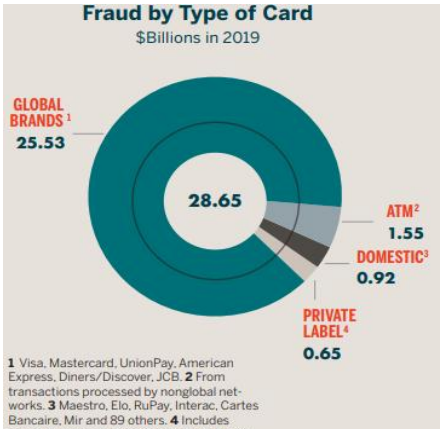


FIGURE 3

According to the Nilson Report [4], a total of $28.65 billion losses was generated worldwide in 2019 through card-based payment systems. Based on the previous years they have also predicted the below graph till 2027. (Figure 2).

Global Brand cards (American, Diners club/Discover, JCB, Mastercard, Visa and UnionPay) together lost $25.53 billion in 2019 due to frauds [4]. This is 2.7% more than in 2018. This totals to 89.11% gross fraud losses worldwide in 2019 as shown in Figure 3.

Most payment related frauds take place in eCommerce. Identity theft is the most typical types of frauds in eCommerce.[5]

Identity Theft is when an attacker or an imposter offense a user account, makes alteration to the personal data, and then tries to spend the money in buying products using the fake information. This type of frauds can be detected using Behavior Analysis (BA). BA can analyze any suspicious activity and can find any inconsistencies in the past personal data.

To give the customer the best experience in online payments and an increase in the mobile payments, banks have reduced the amount of verification procedures. Current Conventional (Rule-based) fraud Detection methods are failing to differentiate among fault or an odd transaction from real fraud. This method does not process the real time data, which in turn takes a huge amount of time to detect any fraud. The performance also degrades when system has to process a huge amount of data. Conventional (Rule-based) method may be challenging to cope with as the algorithms and rules are written manually and to configure them in a proper manner involves accurate, arduous, and labor-intensive programming for every thinkable fraud prospect. Multiple fraud methods are invented and to cope with those new methods, these rules have to be constantly adapted to the existing, evolving, and future fraud methods.

In current system a Fraud Detection is carried out after the fraud is already performed. Existing system observes a series of transactions made and then tries to classify them as a legitimate or fraudulent transaction.

2. RERLATED WORKS

Increasing frauds in eCommerce have led many researchers to perform a research in this area. Many different techniques and methods are invented and talked about which uses Machine Learning, Neural networks, and data mining.

Andrea Pozzolo and Olivier Caelen [6] in their recent study have mentioned about two methods of detecting fraud. One in Static method, which trains the detection model during every fixed interval of time (once in a month or year). The other method is Online method, which updates the model as soon as a new data of transaction enters. As the conduct of fraud changes every now and then, they have indicated that the Online method is way better than the Static method. Andrea Pozzolo and Olivier Caelen [7] in their work suggested that Random Forest is by far the best measure for detecting frauds.

Kuldeep Randhawa, et al. [8] in their comparative study of machine learning for fraud detection have used standard models in their initial phase and later on made use of hybrid models which created with the help of AdaBoost and voting methods. Based on the experiments they concluded that the voting method had most stable performance even with the presence of noise in them.

John Awoyemi et al. [9] operated an analysis using K-nearest neighbor, Naïve Bayes and logistic regression. Their work was done in Python. These were applied to transaction data and then was additionally resampled with the help of Synthetic Minority Over-Sampling Technique. Their results showed that K-nearest performed the best among other techniques which were calculated based on sensitivity, precision, accuracy, correlation, and specificity.

You Dao et al. [10] in their work "Online Credit Card Fraud Detection: A hybrid Framework with Big Data Technologies" designed a fraud detection framework with the help of big data. They were able to accomplish three most important objectives: capacity to merge several detection models for an enhanced accuracy, capability to process huge amount of data, and detecting fraud in real time.

Abhimanyu Roy et al. [11] made use of deep learning for the fraud detection in online transactions. They used artificial neural network approach where they were able to pre-label almost 80 million transactions as fraudulent and legitimate. High performance cloud computing was also used by them for this approach.

Shiyang Xuan et al. [12] took help of random forest for detecting frauds. For their experiments they used two types of random forests namely Random-tree-based random forest and CART-based, to train the normal and fraud behavior features. Data of an e-commerce company from China was taken into consideration for efficacy of two methods. As a result, random forest was able to get good results for small dataset, it faced problems with the imbalanced data.

W.N. Robinson and A.Aria [13] in their store-centric approach detected sequential anomalies with the help of hidden markov model. They introduced store-model divergence method (SMDM) that was able to work great with various parameters. SMDM pertains sequence analysis to specified records of sequential typed transactions which will then uncover anomalies as probable fraud. SMDM proved to be beneficial where all the transactions are profoundly aggregated and had several, typed and continuous transactions.

2.1. Difference between ML Fraud Detection and Conventional (Rule-Based) Fraud Detection

Recently Machine Learning (ML) Fraud Detection has come into limelight and due to its more accurate results, the fraud detection industry is moving from Conventional (Rule-based) Fraud Detection to ML Fraud detection.

Here are some differences between ML Fraud Detection and Conventional Fraud Detection [14]:

Conventional (Rule-based) Fraud Detection:

- Conventional (Rule-based) Fraud Detection_systems comprise of algorithms that are written and set manually by analysts that detect fraud. Due to these manual algorithms, Conventional method is straightforward. This also requires manually adjusting/adding scenarios which can scarcely identify implied correlations.
- Conventional Fraud Detection make use of legacy software which can barely handle real-time data. Even though this scheme is still in use [15], it is an old method, technology, computer system, or application program, "of, relating to, or being a previous or outdated computer system".
- Further, since they are not capable of managing real-time data, they take immense amount of time to detect any fraud. This can lead to delays in detecting frauds.
- Manual algorithms are set in Conventional method which makes them to detect only the most obvious fraud pursuits.
- Conventional Fraud detection uses several verification procedures, and this may be troublesome for the user.

Machine Learning (ML) based Fraud Detection:

- Machine Learning Fraud Detection can create algorithms which are able to process a huge number of datasets.
- This method does not require manual work for detecting possible frauds, and so they work automatically and detects potential fraud situations. Since they are automatic and not manual, they can find hidden and implied correlations.
- ML based Fraud Detection are efficient for processing real-time data contrasting Conventional based method.

- Machine Learning algorithms are smart enough to get along with the Behavior Analysis (BA) , which in turn aids in cutting the all over number of verification procedures.
- The time required for verification procedures is also less compared to the Conventional Fraud Detection.

2.2. Proposed System

2.2.1. Intro to HMM

The Hidden Markov Model is based on enhancing the Markov Chain. Markov Chain is basically a model which depicts about the probabilities of sequences of *states*, random variables. Each state takes up some values from some sets. Sets can comprise of tags, words, or symbols such as weather. In a Markov chain, it makes a powerful hypothesis that to forecast the future, all that is required is current state. This means the state that is before the current state has no influence on the future state, and the only state that matters is the current state. Figure 4 explains the Markov chain to predict tomorrow's weather. For that we can check on today's weather, but we are not able to check yesterday's weather [16].

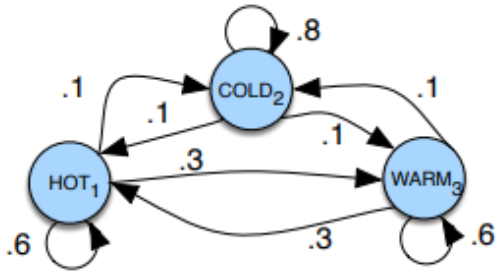


FIGURE 4

In Figure 4, the states are represented as nodes, and transactions with corresponding probabilities as edges. The represented values are probabilities whose sum must be equal to 1. As shown in figure 4, if we start from COLD (2) state, there is a probability of 0.8 that it will be COLD again tomorrow and 0.1 probability to become either HOT or WARM. Same thing can be set for the HOT and WARM states.

Formulation of Markov assumption for a sequence of state variables $q_1, q_2, \ldots\ldots q_i$, where the current state matters but not the past is as follows:

$P(q_i = a|q_1 \ldots.. q_{i-1}) = P(q_i = a|q_{i-1})$

A Markov chain can formally consist of following components.

- S = $s_1, s_2, \ldots\ldots s_N$, are a set of N states
- A= $a_{11}, a_{12}, \ldots\ldots a_{n1}\ldots\ldots a_{nm}$, are Transaction probability matrix where each a_{ij} states the probability of transacting from state i to j.
- π = $\pi_1, \pi_2, \ldots\ldots, \pi_N$, are initial probability distribution over states.

When we need to calculate probability for a sequence that is observable events, Markov chain is useful. But in some cases, for which we want to calculate probabilities, they are hidden and are not observable. For example, we are not able to observe a transaction taking place. Therefore, for those events, *Hidden Markov Model (HMM)* is used, which allows us to find probabilities for both observed and hidden events.

A Hidden Markov Model can formally consist of following components [17]:

- S = $s_1, s_2, \ldots\ldots s_N$ refer to a set of N states
- A = $a_{11}, a_{12}, \ldots\ldots a_{n1}\ldots\ldots a_{nm}$, are Transaction probability matrix A, where each a_{ij} states the probability of transacting from state i to j.

- O = O_1, O_2, ……O_N, are sequence of T observations where V = V_1, V_2,, ……,V_N are individual symbols
- B = $b_i(o_t)$, are sequence of observations likelihoods, each one having the probability of observation generated from a state *i.*
- π = π_1, π_2, ……, π_n , are initial probability distribution over states.

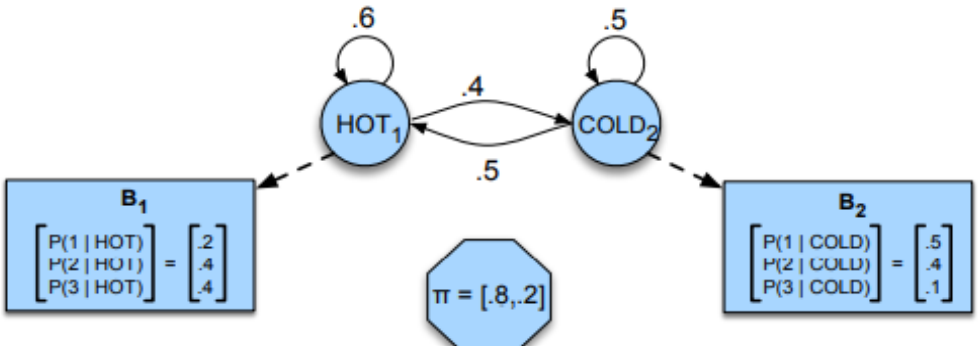


FIGURE 5

Figure 5 has two hidden states HOT(H) and COLD(C). Sequence of observations O, each an integer (1, 2, 3) shown in the matrices B1 and B2, refer to the number of ice creams eaten by a person on a given day. Hidden sequence Q of states of weather (HOT or COLD), needs to found out that made a person to have an ice cream. Table 1 is the simplified table of figure 5.

	p(...|H)	p(...|C)
p(1|...)	0.2	0.5
p(2|...)	0.4	0.4
p(3|...)	0.4	0.1

TABLE 1

The numbers in the table indicates the random probabilities. The probabilities of 1, 2 and 3 ice creams on a hot day are 20%, 40% and 40%, which totals up to 100%. Therefore, it can be guessed that on hot days, someone ate 3 ice creams, with 40% probability and on cold days ate 1 ice cream with a probability of 50%. Hence it can be guessed that, if someone ate 3 ice creams, the odds are 4 to 1 indicating that it was a hot day. But if someone ate 2 ice creams, the odds are 4 to 4, indicating that it was not clear of a hot or a cold day. Similarly eating 1 ice cream has odds of 2 to 5, which clearly indicates of a cold day.

2.2.2. Application of HMM in Fraud Detection

HMM is the perfect model to be implemented for the Fraud Detection system as there are many events which are not observable. Here in this system, can detect frauds just based on the cardholder's spending habits. HMM has the most reduced number of False Positive transaction which are identified as malicious by FDS even though they are legitimate, which makes it perfect to use for Fraud Detection.

The details of items purchased in Individual transactions are usually not known to any Fraud Detection System (FDS) running at the bank that issues credit cards to the cardholders. For each transaction taking place, it is sent to Fraud Detection System (FDS) for verification. The fraud

detection system (FDS) accepts the card details such as Card number, CVV number, card type, expiry date, and amount to authenticate, if the transaction is legitimate or not.

When HMM is implemented, clusters of training sets are created to detect the spending habits of cardholder. Details regarding the purchases, like number of items bought, or description of the items are not known to FDS. FDS only works with the amount of transaction. HMM makes a prediction for three price ranges for any transaction: Low, Medium, and High. Based on the amount of various transaction, it forms a cluster in either low, medium, or high price range. According to the spending habits of the cardholder, it tries to figure out any variations occurring. Initial set of probabilities is selected based on the previous data and makes a sequence for future transactions. If FDS figures out the transaction is fraud, it creates an alarm, and a module of security information comes up. This module has various security questions such as date of birth, and some personal information, such as account number. If the security questions are answered correctly, the transaction goes through. We assume that only the cardholder is the one who knows the personal information. If the security questions are not answered correctly in 3 attempts, the transaction is rejected, and the card is blocked.

3. TECHNIQUES AND ALGORITHM USED

As discussed in the earlier section, for a full specification of HMM, we need two estimation model parameters, N and M, with a total of three probability distributions A, B and π. We can write them as a complete set as ƛ = {A, B, π}. The observation sequence O discussed in the above section can be produced through various possible state sequences.

Let's take a sequence Q = q_1, q_2, ……q_N, , whereq_1 is the initial state. The probability of getting O from this given state sequence can be stated as P(O | Q, ƛ) = $\prod_{t=1}^{N} P(O_t \mid q_t, ƛ)$. This equation can also be stated as below:
P(O | Q, ƛ) = bq_1 (O_1)* bq_2 (O_2)…….. bq_N (O_N)

Also probability of state sequence Q can be stated as P(Q | ƛ) = $\pi_{q_1} a_{q_1 q_2} * a_{q_2 q_3} \ldots\ldots\ldots a_{q_{N-1} q_N}$
Therefore, probability of making observation sequence O stipulated by ƛ can be stated as below:
P(Q | ƛ) = $\sum_{all\ Q} P(O \mid Q, ƛ) P(Q \mid ƛ)$ [18][19]

We get the HMM parameters for each user or cardholder. The forward-backward algorithm begins with all the initial parameters and congregates to the most nearby values.

In current system, after getting the HMM parameters, we acquire symbols from the training data of a cardholder and make an initial state series of all the symbols. Let's consider O_1, O_2, ……, O_N as one sequence. Based on the cardholder's transaction history in time t, this observation sequence is created. This input sequence is added to HMM to calculate the probability of approval. Suppose we get probability $\alpha_1$as α_1 = P (O_1, O_2, ……O_N | ƛ). If we consider O_{N+1} as a new sequence at a particular time t + 1, when a transaction is in process. Now since we have N+1 sequence, to consider only N sequence, O_1 has to be removed and hence sequences from O_2 to O_{N+1} are considered.

We get new probability as:
α_2 = P (O_2, O_3, ……O_{N+1}| ƛ).

From this we get:
Δα = α_1 -α_2

If we get results as Δα > 0, HMM contemplates O_{N+1} new sequence with lowest probability and hence the transaction is believed as a fraud transaction only if the change in the percentage in probability is more than the threshold value that is set beforehand:
Δα / α_1 ≥ threshold [20] .

Considering this system, there are three price ranges set for the spending profile of any card holder such as High (H) which ranges from ($501 to card limit), Medium(M) which ranges from ($101 to $500), and Low(L) which ranges from ($0 to $100). We can consider symbols set as V = [L, M, H] . We also define HMM parameters like State Transition Probability Matrix A, Observation Symbol Probability Matrix B, and Initial State Probability Vector π. All these three parameters are considered in the training phase of the HMM.

For the first 10 transactions, the system does not have enough information to detect a fraud based on the transactions. Therefore, for each transaction, user is asked some security questions. This is done until the user has done at least 10 transactions. Next, the HMM model gets the data for future verification based on the spending habits of the user. The algorithm and the description of the model are explained below.

3.1. Algorithm

The Algorithm section includes the training and detection phases. Training is done based on clustering scheme and it includes the following steps.

1. Based on Cardholder spending habits, identify the profile of cardholder
2. Calculation of probability relies upon the quantity of time that is past since entering current state
3. Create training sequence to train the model

Figure 6 shows the architecture of implementation of Training Algorithm.

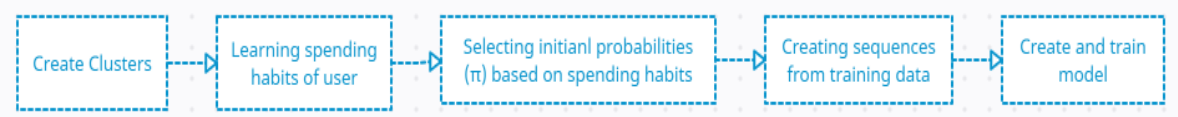


FIGURE 6

Detecting Fraud includes the following steps:

1. Creating observation symbol B_{N+1}
2. Creating the new sequence with addition of B_{N+1} in current sequence
3. Analyzing the difference in the probability and testing the outcome with training phase
4. If results are similar, the transaction is legitimate; otherwise, fraud alarm is created, and the system asks for verification.

Figure 7 shows the architecture of implementation of Detection Algorithm.

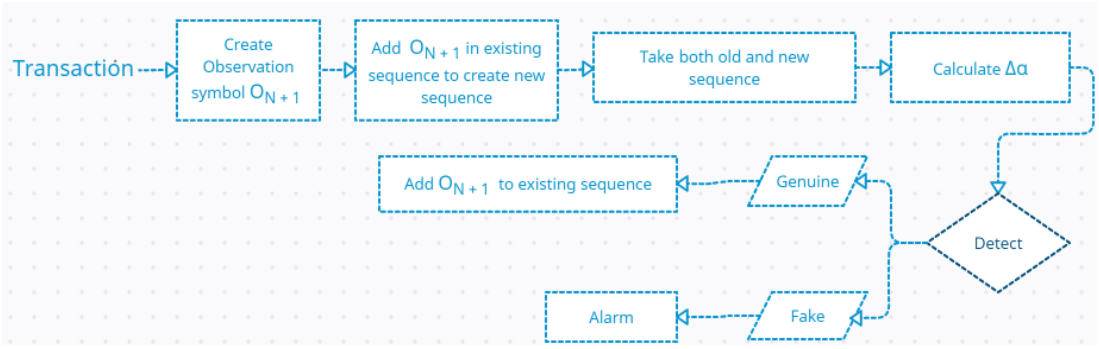


FIGURE 7

3.2. Model Description: Online Shopping, Fraud Detection System

In the current existing systems, banks or other payment gateways asks for card information such as CVV, expiry month and year. But in cases where card is stolen or lost, all these details are readily available on the card itself, which makes it easier for any fraudulent to commit fraud. In

addition, currently bank asks to provide a secure password for online purchases which is also not secure. This proposed model uses HMM to detect if the transaction is fraud or not at the time of transaction. This model has two modules namely: Online shopping & Fraud Detection.

Online shopping module is the same as other online shopping websites. If the use is already registered, the user can login to the site; otherwise, the registration should be made first. The system asks a few security questions which will be used for authentications purposes. Only the users know the security questions and answers entered. After logging into the website, products can be selected and added into the cart for further transactions.

In the fraud detection section, once the user clicks for the payment, the system asked the user for card details. This includes card number, cvv, and expiration month and year. Once entered, the system checks the information with database. If the system finds that there is some variance in current transaction and the past transactions, the fraud detection module is activated.

Since this system is designed like other shopping website, the data of registered users, the order details, and the transaction amount are stored in the backend SQL database tables. If there are less than 10 transactions in the database for that user, then it asks directly for all the personal authentication questions that were set up during the registration. After the database is loaded with 10 transactions, it starts to compare the current transaction with the previous transactions before making a decision. Currently in this system, 10 transactions are taken into account to understand the behavior of the user, which can be increased as per requirements. In future, more than 10 transactions can be used from which HMM will learn the user's behavior for more accurate analysis.

The HMM model fetches the data from the SQL database tables and trains the data based on the users spending habits and how data are classified based on the threshold value set beforehand. The transaction amount and transaction category of the current purchase are compared with the past transactions. After performing the calculation for transaction probability, the system declares if the transaction is legitimate or fraud. If the system declares the transaction as fraud, it adds one more verification step that is the security questions. If the user enters correct answers transaction goes through. In case the transaction is fraud and wrong answers are submitted, after 3 attempts the card and account are blocked and no further transactions will be allowed from that account.

In cases where a user forgets the answers to the questions, the user can request to unblock his account by filling up the unblock request form. Figure 8 shows an activity diagram of the proposed system and Figure 9 shows the sequence diagram of the proposed system.

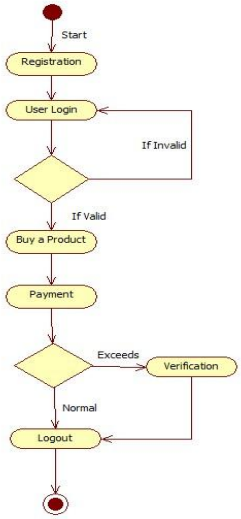


FIGURE 8

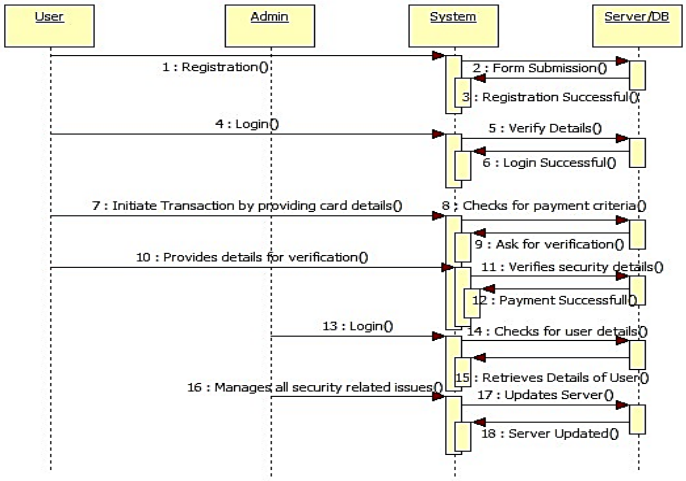


FIGURE 9

Figure 10 depicts a complete diagram of working of the proposed system. The diagram starts with the user login, which if validated by the system, will allow user to go further. If the credentials entered by user are invalid, system will prompt for correct credentials. After user is logged in, products can be viewed and purchased. During that same time system will make a connection with the server. When customer buys a product system will fetch all the details related to the customer from the backend SQL tables. The incoming transaction will be mapped by the HMM algorithm based on previous transactions of the user and will make a decision whether the transaction is fraud or genuine. If detected as genuine, transaction will go through and order will be placed. If detected as a fraud transaction, additional information will be asked to user. If additional details entered correct, transaction will go through or else after 3 attempts, card will be blocked.

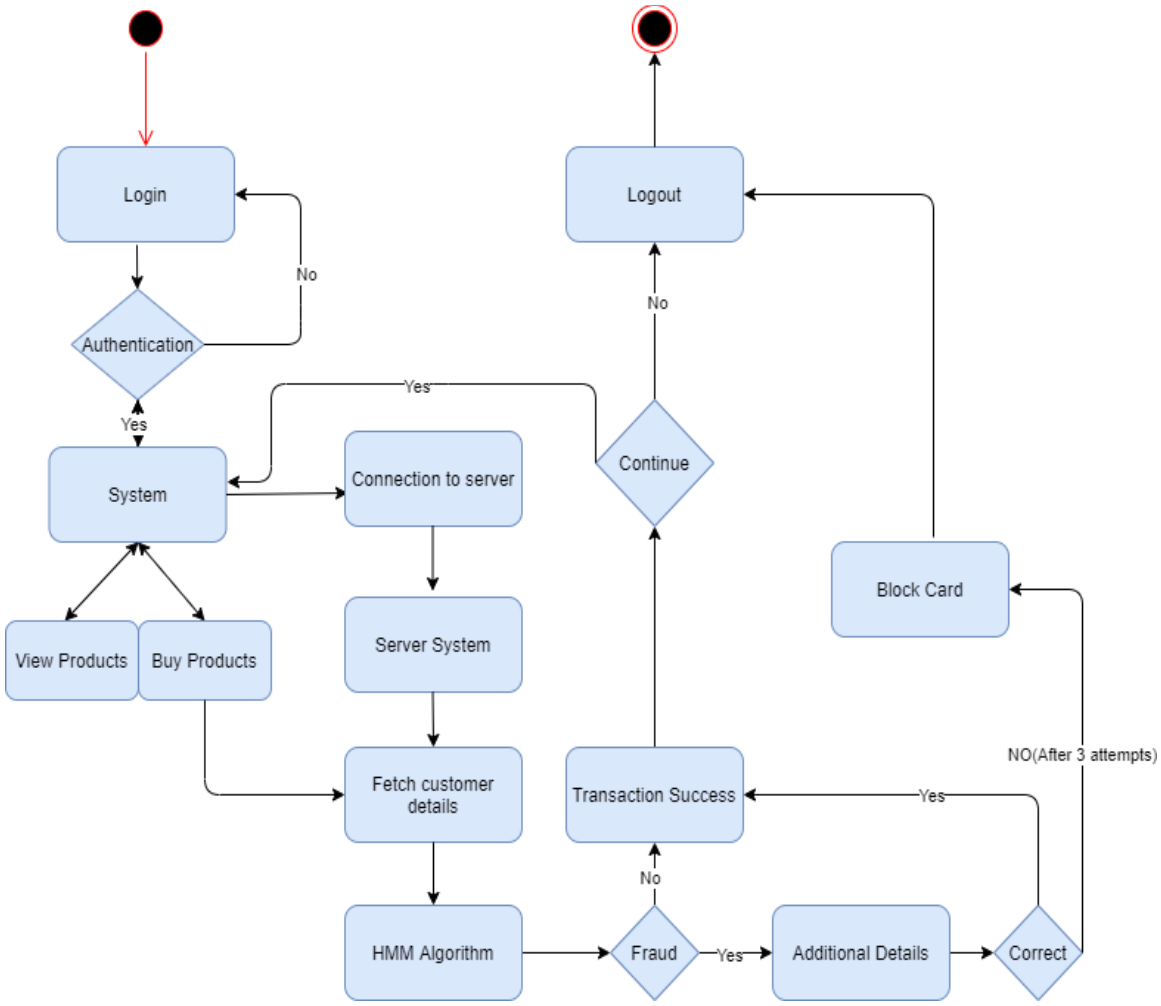


FIGURE 10

4. RESULTS AND DISCUSSION

Due to the security and privacy reasons, we are not able to perform the simulation the real time data as they are not provided by any Card issuing banks. Thus, for our proposed system a random set of total 20 transactions of a user are taken. Each transaction is classified into the purchased category and price range set before that is, High, Medium and Low. The detection system works after learning from first 10 transactions. After 10^{th} transaction, the Fraud Detection starts detecting fraud on every incoming transaction. Following is the random data entered while making purchase on the online shopping module of the system with category of purchase and

transaction amount. This data is fetched from the SQL database tables where details of all the users and transactions are stored. Table 2 shows the dataset taken for this system, showing all the transactions and categories.

No. of Trans	Amount	Category	No. of Trans	Amount	Category
1	10	Art & crafts	**11**	500	Electronics
2	50	Shoes	**12**	45	Purse
3	80	Shoes	**13**	20	Art & crafts
4	150	Purse	**14**	5	Art & crafts
5	1000	Rent	**15**	1000	Rent
6	180	Furniture	**16**	50	Purse
7	250	Furniture	**17**	68	Shoes
8	70	Purse	**18**	94	Shoes
9	55	Purse	**19**	330	Electronics
10	95	Shoes	**20**	320	Electronics

TABLE 2

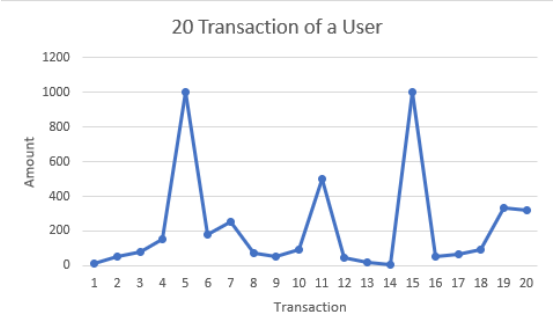


FIGURE 11

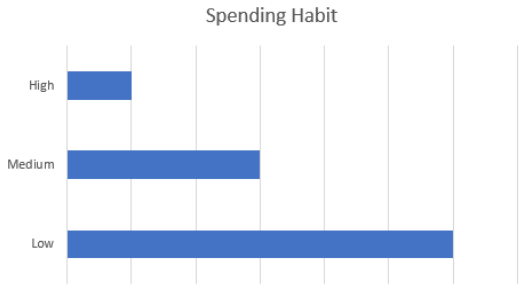


FIGURE 12

Figure 11 displays a graph of Transaction of the user or card holder according to the amount spend. Figure 12 depicts the graph of the spending habit of the user based on the price range set before. The graph shows that users spend almost 60% in the Low range which is from $0 to $100 followed by Medium range with 30% that is between $101 to $500 and the least is spent in High Range that comprises of just 10% ranging between $501 to $1000.

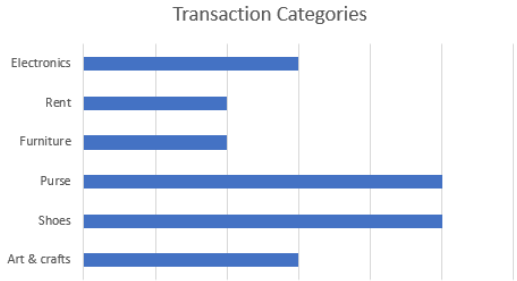


FIGURE 13

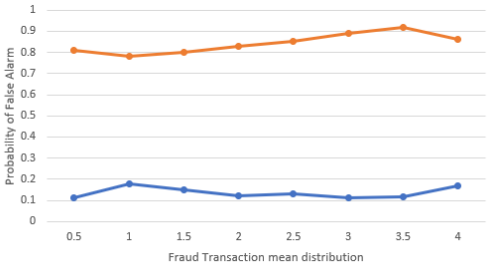


FIGURE 14

Figure 13 is a graphical representation of the Categories in which a user spends. In this system, we have taken Categories such as Art & Crafts, Shoes, Purse, Furniture, Rent and Electronics. It is shown that a user mostly spends on two categories that is Shoes and Purse with around 25%,

followed by Arts & Crafts and Electronics with 15% and least is 10% spend on Furniture and Rent.

Figure 14 shows the Mean Distribution of Fraud Detection. In this graph Probability of Fraud Transaction is compared with Genuine Transaction. As illustrated, when probability of fraud transaction goes down, the probability of genuine transaction goes up vice-versa. This vice versa has helped in finding out the false alarm for the fraud transaction detection. Therefore, the instances where the probability for false alarm is more compared to threshold probability, a fraudulent alarm takes place, and a transaction is declined.

Unlike other fraud detection systems, proposed system is capable of detecting the fraud transaction the same time the transaction is made. The proposed system can detect the fraud on real time data without taking enormous amount of time contrasting the other systems. When compared to other approaches, the proposed system is approximately in the middle for the computation time. However, using this system in a practical environment, the computation time can be low which can be further enhanced using upgraded HMM algorithms. Both less and more number of transactions can be computed in satisfactory amount of time, in milliseconds and seconds.

Theoretically when considering probability matrix B, the processing time rises due to encoding of probability matrix B. Also, more processing time and memory can be used up when dealing with larger matrices. This can be fixed by allocating memory to specified matrix size which can give a better linear execution.

In addition to the processing time, verification procedure has minimal steps which in turn does not negotiate with the customer experience.

Proposed system can be most beneficial to the users who can save their hard-earned money from being used up by fraudulent. All e-commerce businesses can massively stop the losses that incurs to them due to the fraud transactions taking place. And banks can reduce their time expended in the wake of uncovering the frauds and finance that time in some other matter.

5. CONCLUSION

In the proposed system, we have discussed the usefulness of Hidden Markov Model for Detecting Fraud in Online Transaction. The steps that are used in the transaction process are considered as stochastic process of Hidden Markov Model, while the price ranges of transactions are considered as observation symbols and the items purchased are taken as states of Hidden Markov Model (HMM). The proposed system is also ascendable for handling huge amount of transactions. This system gives fast results unlike the existing system. In this system, any incoming transaction will be checked for legitimate, or fake based on the user's spending habits. This system will also make decisions of fake or legitimate transactions based on the set threshold values. Accuracy of this Fraud Detection system is nearly 80%, and has high processing speed, according to the comparative studies made. Proposed system can be used in almost all the online shopping websites for various merchandises. It can also be implemented as an extra layer for payments gatewaying Banks and other e-commerce platforms. It is most suitable for Online Transaction Fraud Detection since there is no need to check the original user as it maintains the log of users.

6. REFERENCES

[1] Nilson Company. Connected Commerce. Connectivity is Enabling Lifestyle Evolution. November 2018.

[2] Stojanovic A., Aouada D., Ottersten B Bahnsen A.C., "Cost-sensitive credit card fraud detection using Bayes minimum risk," in *12th International Conference on Machine Learning and Applications (ICMLA)*, pp. 333-338, 2013.

[3] Nilson Company. Connected Commerce. Issue 1187. Dec 2020 Card Fraud Worldwide.

[4] About US English. (2020, November 09). Retrieved March, 2021, from https://nilsonreport.com/publication_chart_and_graphs_archive.php?1=1&year=2020.

[5] Anderson, Keith & Durbin, Erik & Salinger, Michael. (2008). Identity Theft. Journal of Economic Perspectives. 22. 171-192. 10.1257/jep.22.2.171.

[6] A. Dal Pozzolo, O. Caelen, Yann-A¨el Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. Expert Syst. Appl., 41:4915–4928, 2014.

[7] Andrea Dal Pozzolo, Giacomo Boracchi, Olivier Caelen, Cesare Alippi, and Gianluca Bontempi. Credit card fraud detection and concept-drift adaptation with delayed supervised information. 2015 International Joint Conference on Neural Networks (IJCNN).

[8] Kuldeep Randhawa, Chu Kiong Loo, Manjeevan Seera, Chee Peng Lim and Asoke K. Nandi, "Credit card fraud detection using AdaBoost and majority voting," *IEEE Access*, vol. 6, pp. 14277-14284, 2018.

[9] John O. Awoyemi, Adebayo Olusola Adetunmbi, and Samuel Adebayo Oluwadare. Credit card fraud detection using machine learning techniques: A comparative analysis. 2017 International Conference on Computing Networking and Informatics (ICCNI), pages 1–9, 2017.

[10] You Dai, Jin Yan, Xiaoxin Tang, Han Zhao and Minyi Guo, "Online Credit Card Fraud Detection: A Hybrid Framework with Big Data Technologies", IEEE TrustCom/BigDataSE/ISPA, pp 1644 -1651, 2016.

[11] A. Roy and J. Sun and R. Mahoney and L. Alonzi and S. Adams and P. Beling, "Deep learning detecting fraud in credit card transactions," in *Systems and Information Engineering Design Symposium (SIEDS)*, pp. 129-134, 2018.

[12] S. Xuan, G. Liu, Z. Li, L. Zheng, S. Wang and C. Jiang, "Random forest for credit card fraud detection," 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, pp. 1-6, doi: 10.1109/ICNSC.2018.8361343.

[13] William N. Robinson, Andrea Aria, Sequential fraud detection for prepaid cards using hidden Markov model divergence, Expert Systems with Applications, Volume 91,2018, Pages 235-251, ISSN 0957-4174.

[14] Fraud detection: How machine learning systems help Reveal scams in Fintech, healthcare, and ecommerce. (2020, February 27).

[15] K. Bennett. Legacy systems: coping with success. Published in: IEEE Software (Volume: 12, Issue: 1, Jan. 1995). Pages 19-23. DOI: 10.1109/52.363157.

[16] Daniel Jurafsky & James H. Martin. Speech and Language Processing. December 30, 2020.

[17] Stamp, M. A Revealing Introduction to Hidden Markov Models.

[18] V.Bhusari & S.Patil. “Application of hidden Markov Model in Credit Card Fraud Detection” November 2011. International Journal of Distributed and Parallel Systems (IJDPS) Vil.2, No.6.

[19] Rabiner, L. R. (1989, February). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. February 1989. Proceedings of the IEEE, Vol.77, No.2.

[20] A.Prakash (December 2012)A Novel Hidden Markov Model for Credit Card Fraud Detection. International Journal of Computer Applications, Vol.59, No.3.

[21] T.Chetcuti & A. Dingli,(2008) Using Hidden Markov Models in Credit Card Transaction Fraud Detection.

[22] Bunke, H., & Caelli, T. (2001). *Hidden Markov models: Applications in computer vision*. Singapore: World Scientific.

[23] H.Zhou, G.Sun . (2019). *A Scalable Approach for Fraud Detection in Online E-Commerce Transactions with Big Data Analytics*. Tech Science Press, Vol.60, No.1.

Efficient Tree-based Aggregation and Processing Time for Wireless Sensor Networks

David Fotue *david.fotue@univ-yaounde1.cm*
Département d'Informatique
Ecole Normale Supérieure de Yaoundé
BP 47, Yaoundé, Cameroun

Houda Labiod *houda.labiod@telecom-paristech.fr*
Informatique Réseaux et Sécurité (INFRES)
Télécom ParisTech
46, Rue Barrault, France

Abstract

Tree-based data aggregation suffers from increased data delivery time because the parents must wait for the data from their leaves. In this paper, we propose an Efficient Tree-based Aggregation and Processing Time (ETAPT) algorithm using Appropriate Data Aggregation and Processing Time (ADAPT) metric. A tree structure is built out from the sink, electing sensors having the highest degree of connectivity as parents; others are considered as leaves. Given the maximum acceptable latency, ETAPT's algorithm takes into account the position of parents, their number of leaves and the depth of the tree, in order to compute an optimal ADAPT time to parents with more leaves, so increasing data aggregation gain and ensuring enough time to process data from leaves. Simulations were performed in order to validate our ETAPT. The results obtained show that our ETAPT provides a higher data aggregation gain, with lower energy consumed and end-to-end delay compared to Aggregation Time Control (ATC) and Data Aggregation Supported by Dynamic Routing (DASDR).

Keywords: Wireless Sensor Networks, Aggregation Time, Aggregation Gain, Parent, Leaf.

1. INTRODUCTION

Recent innovations in micro-electro-mechanical technologies bring significant advantages to the development of low-power, low-cost multifunctional sensors equipped with storage, computing and communication capabilities. Wireless Sensor Networks (WSNs) are ad hoc wireless networks that consist of a large number of small devices, known as sensors, scattered over a particular geographical area [1]. In recent years, WSNs are seen as a reality, due to the potential applications in various domains such as industrial, biological, medical, military, nuclear science, forest fire detection and so on. The lack of a predefined communication infrastructure increases the challenges of designing of communication techniques for these networks, especially in hostile environments, where it is often difficult to replace sensor batteries after deployment and where communication infrastructures are not accessible or available.

In WSNs, all sensors send their data towards the central sink, which is the final recipient of the sensed information. Sensors are equipped with a limited amount of storage capacity, and powered by batteries with a finite life, making power saving an important issue in achieving long-lived wireless multi-hop networks [2]. In WSNs, each sensor covers a defined area, collecting local data and sending it towards the main sink. It may happen that some sensors deployed in the monitored area sense common data. Therefore, much energy will be wasted if all these data are forwarded towards the sink. Data aggregation schemes exploiting in-network processing have been proposed as efficient techniques to conserve energy by locally processing the data as much as possible in order to reduce the amount of data transmitted by each sensor towards the sink [1].

As the sink has to receive the data from sensors in a timely manner, this data aggregation has a relationship with the data aggregation time [3]. We need to determine the data aggregation time that each parent in the tree should spend in aggregating the data sent from its leaves. As the network topology can vary, some parents might have many leaves, making it very expensive for a parent to store all incoming data in its buffer. Failing to account for data aggregation time may lead to a longer waiting time for each parent and increase overall data delivery latency.

We propose an Efficient Tree-based Aggregation and Processing Time (ETAPT) algorithm using Appropriate Data Aggregation and Processing Time (ADAPT) metric. Given the maximum acceptable latency, ETAPT's algorithm takes into account the position of parents, their number of leaves and the depth of the tree, in order to compute an optimal ADAPT time to parents with more leaves, so increasing data aggregation gain and ensuring enough time to process data from leaves. Simulations were performed in order to validate ETAPT. The results obtained show that our ETAPT provides a higher data aggregation gain, with lower energy consumed and end-to-end delay compared to Aggregation Time Control (ATC) [4] and Data Aggregation Supported by Dynamic Routing (DASDR) [5].

The remainder of this paper is organized as follows: Section 2 outlines related work. Section 3 formulates the problem and presents our proposition. Section 4 presents our model and describes notation. Section 5 presents our approach. Section 6 presents performance metrics and comparative results and Section 7 concludes the paper.

2. RELATED WORK

As our WSN focuses on gathered the data from the environment, it is important to forward the data in a timely manner towards the sink. Several approaches have been proposed concerning data aggregation time. Actually, we use tree and cluster structure for data aggregation because they are useful in environment monitoring where the maximum data values received by the sink provide the most useful information [2].

[4] proposes dynamic Aggregation Time Control (ATC) based on the number of leaves of the root node in a tree-structure. ATC adjusts the aggregation time of the sensor.to assign more aggregation time to sensors having more children in order to give more possibilities to aggregate the data. Simulations show that ATC has a high aggregation gain. However, ATC cannot be adopted to the multi hop sensor networks since it requires the global knowledge of the network. In addition, the broadcast scheme used during the construction of the tree needs a high communication overhead and decreases network performance. [5] proposes Data Aggregation Supported by Dynamic Routing (DASDR), which can adapt to different scenarios without incurring much overhead. Sensor nodes that monitor events are concentrated in space as far as possible and data packets flow to the sink along different paths. Dynamic routing constructs a depth potential field, which aims to guarantee that packets will reach the sink eventually and a queue potential field, which makes packets more spatially convergent, making data aggregation more efficient. Simulations show that DASDR improves the data aggregation ratio, saves energy, and scales well with network size. [6] focuses on a real-time data delivery, but do not take the data aggregation and processing into account. [7] proposes a cascading time-out in which sensors schedule their time-outs based on their position in the aggregation tree. A sensor's time-out happens after its leaves' time-outs, so enabling a sensor to aggregate the data from all its children. [8] computes the data aggregation time-out for clustered WSNs. The time-out is calculated for each sub-tree in the cluster taking into account packet transmission and cascading delay, leading to a reduction in aggregation time and energy use. [9] develops an approach which delivers the data to the sink within the deadline. They estimate the time-out of each sensor in the tree, so that the data generated by each sensor is delivered to the sink before the deadline. [10] proposes to construct a centralized and decentralized structure in the network in order to reduce the transmission delay during the collection of data. [11] proposes a Delay-minimized Energy-efficient Data Aggregation (DEDA) algorithm to minimize data aggregation latency. The physical

distance between sensors is takes into account in DEDA to save the transmission energy in order to improve network lifetime.

Our proposal is similar to the one used by [4] and [5]. However we take into account the position of parents, their number of leaves and the depth of the tree, in such a way those parents with more leaves will be dynamically allocated an appropriate aggregation time, so maximizing the data aggregation gain and improving network performance.

3. PROBLEM STATEMENT AND PROPOSITION

In this section, we formulate the problem addressed in this paper and present our proposal.

3.1 Problem Statement

In our context (spatial aggregation), the data gathered by sensors that are close to each other do not vary much over time. Tree-based data aggregation results in increased data delivery time because the parents must wait for the data from their leaves. Since the network topology can be random, as shown in Figure 1, some parents may have many leaves, making it very expensive for a parent to store all incoming data in its buffer. [4] shows that if a parent waits for the data from its leaves for a long time, it collects more data and hence Data Aggregation Gain (DAG) increases. DAG is the ratio of traffic reduction due to aggregation to the total traffic without aggregation [12]. However, this long waiting time means that the data delivery time to the sink

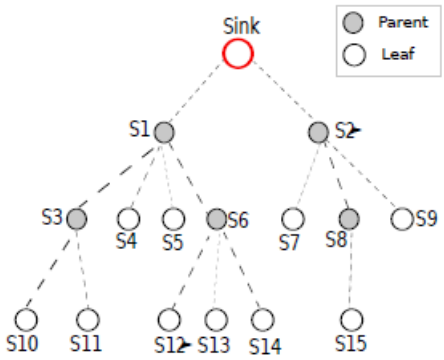


FIGURE 1: WSN: Tree Construction.

may increase. Thus, it is important to consider the time taken by parents to aggregate and process the data, because it takes more time to aggregate and process the data than to transmit the data towards the sink. Lacking of attention to the data aggregation and processing time may increase the overall data delivery latency or reduce the DAG. [13] shows that neglected the data aggregation and processing time may increase the overall data delivery latency or reduce the DAG.

3.2 Proposition : ETAPT Algorithm

We propose an Efficient Tree-based Aggregation and Processing Time (ETAPT) algorithm using the Appropriate Data Aggregation and Processing Time (ADAPT) metric to calculate the data aggregation and processing time for parent nodes as shown in Figure 2. After have been built the tree out from the sink, in order to elect sensors with the highest degree of connectivity as parents and sensors with the lowest degree of connectivity as leaves as shown in Figure 2. Given the maximum acceptable latency, ETAPT's calculation takes into account the position of parents, their number of leaves and the depth of the tree, in order to compute for each parent an optimal ADAPT before aggregating and processing the data from its leaves. So, allocating an appropriate aggregation time (A_{ggTime}) to parents with more leaves in order to increase the DAG, thus ensuring enough time to process the data from leaves.

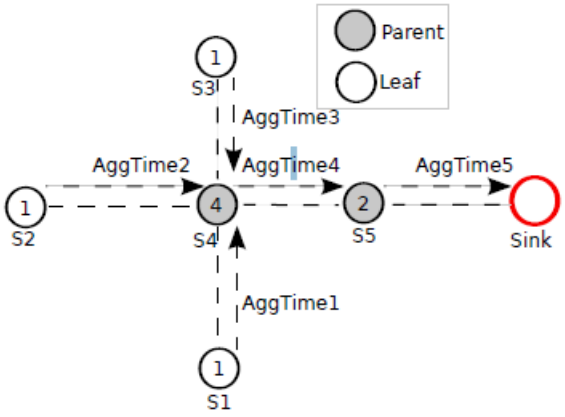


FIGURE 2: Distribution of Aggregation Time.

4. NETWORK MODEL AND NOTATION

4.1 Network Model

The proposed WSN can be modeled as a connected graph G = (S, E), where S is the set of N fixed sensors, and E is the set of wireless links. We use the locality model suggested in [14] to determine network connectivity. The probability of a link between two sensor nodes S_i and S_j is given by:

$$P = \begin{cases} 1 \; if \; D(S_i, S_j) \leqslant R \\ 0 \; if \; D(S_i, S_j) > R \end{cases} \quad (1)$$

Where $D(S_i, S_j)$ is the Euclidean distance between sensors S_i and S_j, and R is the locality radius.

4.2 Notation

Let s € S. Let Path (s_1, s_k) be the sequence: s_1, s_2...s_k. We define Hop (s_1, s_k) = k-1 as the number of hops from sensor s_1 to s_k. Let d(s) be the degree of sensor s. δ is the minimum transmission time between two sensors of the same Hop in the tree, and ensures that there is a difference in the waiting times at consecutive Hop of the tree. We define:

$$L_{EAF} = s \mid s \in S, d(s) = 1 \quad (2)$$

as the set on leaves in the tree,

$$M = S - L_{EAF} - sink \quad (3)$$

as the set on parents in the tree and,

$$Hop_{Distance}(s) = d(s, sink) \quad (4)$$

as the number of hops of the Path (s, sink). We recall that Hop (sink) = 0. Let the depth of the tree be:

$$Depth = Max_{s \in L_{EAF}}(d(s, sink)) \quad (5)$$

the number of hops from the sink to the deepest leaf in the tree (the maximum number of hops towards the sink in the tree). We define the weighted length of the Path (s_1, s_k) as:

$$WPath(s_1, s_k) = \sum_{i=1}^{k-1} d(s_i) \quad (6)$$

the sum of the degrees of the descendant sensors. Let L'_{EAF} be all the leaves in a subtree rooted at sensor s, s € M. We define the maximum weighted depth of the subtree as:

$$MaxWPath(s) = Max_{s_i \in L'_{EAF}}(WPath(s_i, s)) \quad (7)$$

the maximum degree of all the descendant sensors in L'_{EAF} to root to sensor s in the subtree. For all (s € L_{EAF}), $M_{ax}WP(s) = 0$. Finally, T_{max} be the maximum acceptable latency.

In the following Section 5, we describe our ETAPT algorithm.

5. EFFICIENT TREE-BASED AGGREGATION AND PROCESSING TIME (ETAPT) ALGORITHM

5.1 Assumptions

We assume in our approach that:

- Sensors are deployed in an area of size L.
- Sensors are homogeneous (same computing, memory...) and fixed.
- Each sensor maintains a list of identifies (Id) of its neighbors.
- Each sensor keeps track of its own degree of connectivity value d(s).
- Each leaf has one parent that is responsible for forwarding the received data towards the sink.
- Leaves can only sense and transmit their data to their parents.
- Aggregation of multiple packets results in one packet.
- A single sink is the final recipient of all the sensed data.
- T_{max} is the maximum acceptable latency.

As shown in Figure 2, the tree is built out from the sink, taking into account the degree of connectivity of sensors d(s). The sensors with the highest degree of connectivity are selected as parents and those with lowest degree of connectivity as leaves. Given T_{max}, ETAPT will determine the ADAPT for each parent based on its position, its number of leaves and the depth of the tree. We assume that every sensor generates a data packet of the same length periodically, and multiple packets can be combined into one packet after the data aggregation process. Any packets arriving after the ADAPT time calculation are discarded. The algorithm consists of two major procedures: $M_{ax}WPath$, $Hop_{Distance}$, degree of sensor and average waiting and aggregation time's determination.

5.2 $M_{ax}WPath$, $Hop_{Distance}$ and Degree of Sensor Determination

The first step consists in determining by each sensor in the tree: its degree d(s), $M_{ax}WPath(s)$ and $Hop_{Distance}(s)$.The Sink broadcasts a beacon message as a Request$M_{ax}WPath$ with a $Hop_{Distance}$ field, which is incremented as the beacon travels through the tree as shown in Figure 3(a). Every sensor, on receiving the Request$M_{ax}WPath$, adds its $Hop_{Distance}$ value to the beacon, and forwards it to its neighbours. In order to reply to the Request$M_{ax}WPath$ message, every sensor, starting from the deepest leaf, calculates its own degree and the $M_{ax}WPath$ to its parent, generates a Reply$M_{ax}WPath$ message and forwards it to its parent as shown in Figure 3(b).

FIGURE 3: Beacon Structure.

Suppose that s € M is a parent. It calculates and saves its own d(s) and M_{ax}WPath (s) based on the ReplyM_{ax}WPath it receives, generates a new ReplyM_{ax}WPath including its own M_{ax}WPath and forwards it to its parent. The ReplyM_{ax}WPath messages are propagated in a cascading manner along the tree towards the sink. When the sink has received all the ReplyM_{ax}WPath messages, it chooses the largest M_{ax}WPath value from among them and sets:

$$MaxWPath(Sink) = Largest(MaxWPath). \quad (8)$$

5.3 Determination of average waiting and aggregation times

The second phase of ETAPT consists in determining the average waiting time Avg per sensor in order to determine the aggregation Time A_{ggTime} in the tree. The Avg for each sensor (s) is based on T_{max}, M_{ax}WP(s) and Hop (s). When the sink receives a request from an external user specifying Tmax, the sink, based on the information it received in the first step, calculates the Avg per sensor and A_{ggTime} in the tree as follows:

$$Avg_{wait} = \frac{(T_{max} - \delta * Depth)}{MaxWPath(sink)} \quad (9)$$

We assume that T_{max} > (Depth x δ). After the sink has calculated the Avg, it broadcasts a new beacon message through the network including T_{max} and Avg. Every sensor, on receiving the new beacon message, calculates its A_{ggTime} as follows:

$$Agg_{Time} = Avg_{wait} * MaxWPath(s) + (Depth - Hop_{Distance}(s)) * \delta \quad (10)$$

δ is the minimum transmission time between two sensors of the same $Hop_{Distance}$ in the tree.

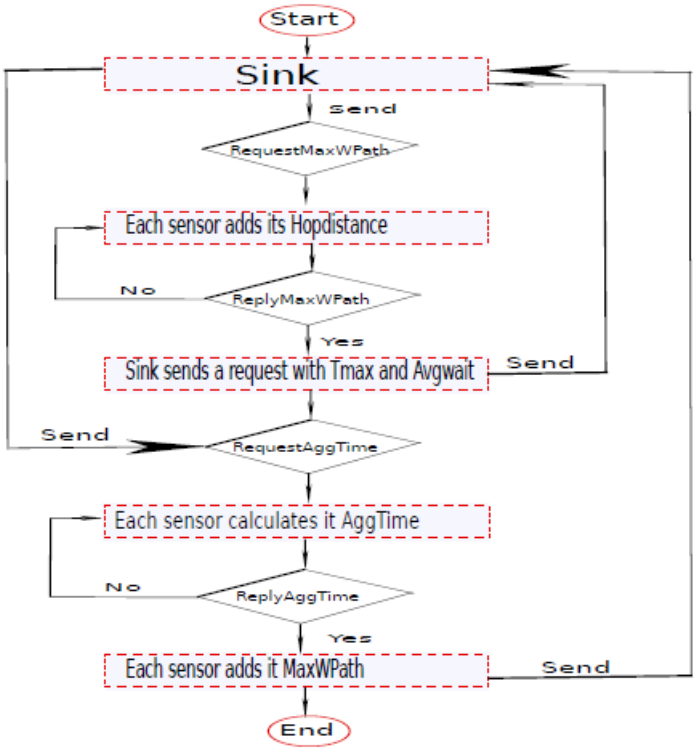


FIGURE 4: ETAPT Algorithm.

5.4 Illustration

Consider a simple topology consisting of 15 sensors as shown in Figure 1. We want to calculate d(s), Hop (s), $M_{ax}WP$ (s) and A_{ggTime} (s) for each sensor in the tree. We suppose that T_{max} = 5s and δ = 0.2s. Taking into account equation (9), the Avg = 0.64s and the Depth = 3. The ADAPT time calculation is summarized in Table 1.

S	d (S)	$Hop_{Distance}$ (S)	$MaxWPath$ (S)	Agg_{Time} (S)
S_1	5	1	7	4.88
S_2	4	1	4	2.96
S_3	3	2	2	1.48
S_4	1	2	0	0.20
S_5	1	2	0	0.20
S_6	4	2	3	2.12
S_7	1	2	0	0.20
S_8	2	2	1	0.84
S_9	1	2	0	0.20
S_{10}	1	3	0	0
S_{11}	1	3	0	0
S_{12}	1	3	0	0
S_{13}	1	3	0	0
S_{14}	1	3	0	0
S_{15}	1	3	0	0

TABLE 1: ADAPT Calculation.

In the following Section 6, we define the performance metrics and present comparative results.

6. PERFORMANCE METRICS AND COMPARATIVE RESULTS

6.1 Performance Metrics

The following metrics are used to evaluate our approach:

- Data Aggregation Gain (DAG)

DAG is defined as the ratio of the benefit of traffic reduction due to aggregation to the total traffic generated without aggregation.

$$DAG = 1 - \frac{P_{Aggregated}}{\sum_{i=1}^{N} P_{Generated_i}} \quad (11)$$

$P_{Aggregated}$ is the total number of data packets aggregated by parents.

- Aggregation Time (AggTime)

A_{ggTime} is defined as the appropriate time need by a parent to aggregate the data from its leaves.

- End-to-End Delay (E2EDelay)

$Delay_{E2E}$ is the average of the time difference between sensed data leaving a sensor and it being received by the sink.

$P_{Received}$ is the total number of data packets received by the sink. $T_{Received}$ is the reception time at the sink, $T_{Transmission}$ is the transmission time from each sensor. The lower the value, the more promptly is data delivered to the sink.

$$Delay_{E2E} = \frac{\sum_{i=1}^{P_{Received}} (T_{Received_i} - T_{Ransmission_i})}{P_{Received}} \quad (12)$$

- Energy Consumed (EC)

Often, sensors are deployed in a hostile environment where replacing the batteries is not always possible. A good choice of energy model is essential to optimize sensor network lifetime. Our approach assumes that sensors are usually in the active mode. The energy model used is the same as in [15]. For each pair of sensors (s_i, s_j), the energy consumed when sending a packet of m bits over a distance D can be calculated as:

Sending sensor energy consumption:

$$E_{Ti}(m, D) = E_{elec} * m + E_{amp} * m * D^2 \quad (13)$$

Receiving sensor energy consumption:

$$E_{Rj}(m) = E_{elec} * m \quad (14)$$

The total energy consumed by each pair (s_i, s_j) is:

$$E_T(m, D) = E_{Ti}(m, D) + E_{Rj}(m) \quad (15)$$

E_{Ti} is the energy consumed for the transmission of a packet by the source S_i, E_{Rj} is the energy consumed to receive a packet s_j, E_{elec} is the energy consumed to run the transmitter and receiver, E_{amp} is the energy used by the amplifier and D is the Euclidean distance between s_i and s_j.

6.2 Simulation Set-up

We implemented a simulation of our network topology using QualNet 5.0. A topology is totally described by the number of stationary sensors N belonging to the network and their locations.

Throughout our analysis, we deploy 100 fixed sensor nodes inside a square area L. The sink is placed at the top left corner of L. During the execution of our simulations, a given source and destination pair remains in the evaluated set until communication between them fails due to energy depletion. We repeated the experiments 20 times for the same topology, with the 95% confidence interval of each data. We took the average value of these 20 runs. Initially, each sensor was charged with an energy of 10^4 Joules. In the analysis, we set T_{max} = 3s, 4s, 5s, 6s.

The parameters are described in Table 2.

Parameters	Description	Value
E	Full energy of sensor	10000 Joules (J)
E_{elec}	Energy of trans/receiver	50 (nJ/bit)
E_{amp}	Energy of amplifier	100 (pJ/bit)
L	Simulation area	1000m x 1000m
P_{Length}	Packet length	2 Kbits
Traffic rate	UDP traffic	4 packets/sec
MAC	MAC layer	IEEE 802.11b
T_{max}	Maximum acceptable latency	Between [3, 4, 5, 6]s
B	Bandwidth	128 (kbps)
R	Locality radius (m)	20m
N	Number of sensors	Between [20...100]

TABLE 2: Simulation Parameters

6.3 Comparative Results

We ran simulation to compare our ETAPT strategy vith ATC [4] and DASDR [5] described in Section 2. Figure 5 depicts the evolution of DAG as a function of T_{max}. We can see that as T_{max} increases, the DAG increases for all the three methods. This shows that as T_{max} increases, each parent has enough time to aggregate its data efficiently. ETAPT, with an average DAG of 90%, outperforms DASDR and ATC, which give 84% and 73.5% respectively. This is because, in ETAPT, the Agg_{Time} of a leaf is proportional to MaxWP (leaf). A leaf with a small MaxWP should transmit the data quickly to its parent; only leaves having the same MaxWP value have the same Agg_{Time}. However, DASDR and ATC use a cascading time-out. This means that sensors at the same Hop in the tree have the same A_{ggTime}, consequently increasing the amount of data loss due to congestion at intermediate parents.

We now evaluate the evolution of DAG as a function of the number of sensors, as shown in Figure 6. In the analysis, we set T_{max} = 3s and we observe the evolution. We see a decreasing of DAG from [60-80] sensors, that is due to the fact that some leaves are disconnected to their parents resulting in a tree with disconnected sub-trees. We can see that for all algorithms, as the number of sensors increases, DAG also increases in each algorithm. That means that the three algorithms continue to deliver data accurately towards the sink as the number of sensors increases. ETAPT achieves the best DAG with an average of 86.4%, compared to 78.4% for DASDR and 71.4% for ATC.

After a packet has been sent along a path P_i ($_i$=1....,k), we must perform an energy reduction operation on each sensor along the path except for the sink. Thus, after a packet is sent by a sensor, the energy level of that sensor is decremented by the amount of energy required to send the data packet. A sensor is considered non-functional if its energy level reaches zero. Figure 7 shows the evolution of the total EC for different techniques with a varying value of T_{max}.

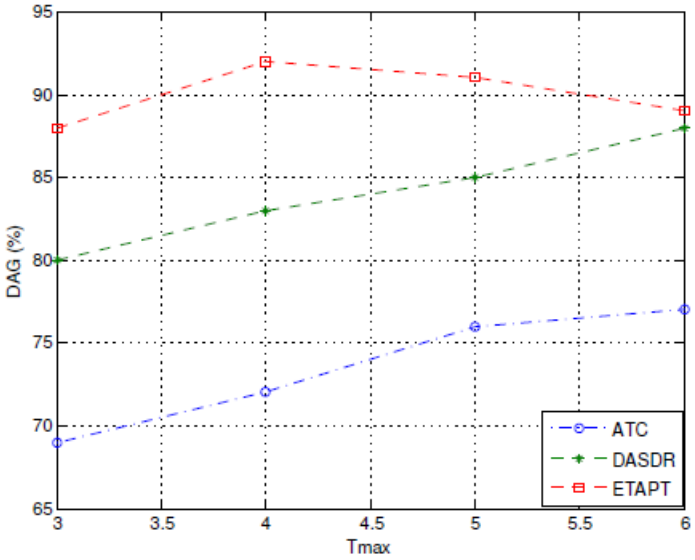


FIGURE 5: Evolution of DAG vs $T_{max.}$

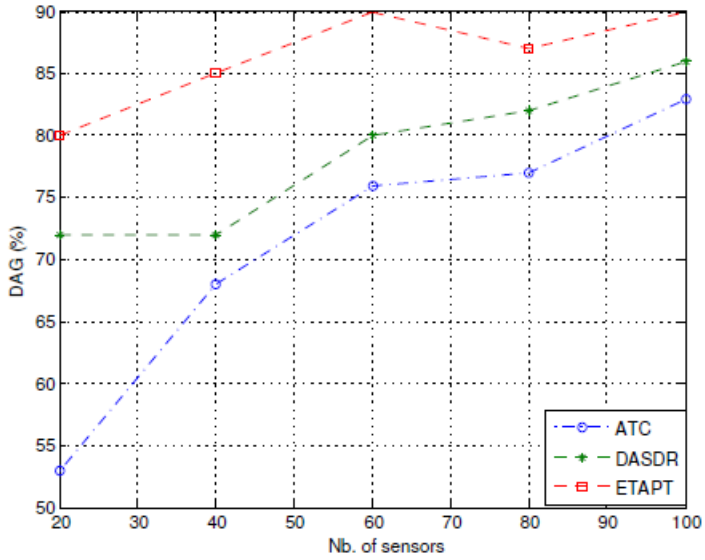


FIGURE 6: Evolution of DAG vs Number of sensors.

We observe that ATC and DASDR have a higher energy consumption than ETAPT. That is due to the fact that in the construction of the tree, we elect sensors having the highest degree of connectivity as parents instead of these with the highest identifier, as in ATC and DASDR. Thus, each sensor has exactly one parent that forwards its data, considerably reducing concurrent transmissions in the network. Our proposal reduces the total EC compared to DASDR and ATC by around 35% and 67% respectively. We evaluated the evolution of the total EC with increasing number of sensors, as shown in Figure 8. We observe a decreasing of EC with increasing number of sensors. That is due to the fact that in dense network, parent nodes might have many leaves which helps by reducing the number of parents necessary to transmit the data in the tree, and hence reduces the EC. The average maximum energy is obtained by ATC with around 45J. An improvement is obtained by DASDR, which uses only around 25J. ETAPT outperforms both, with an average EC of just 16J.

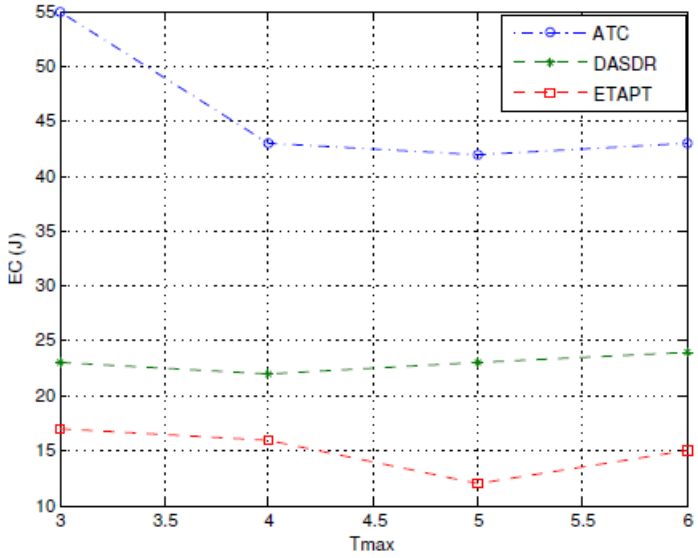


FIGURE 7: Evolution of EC vs $T_{max.}$

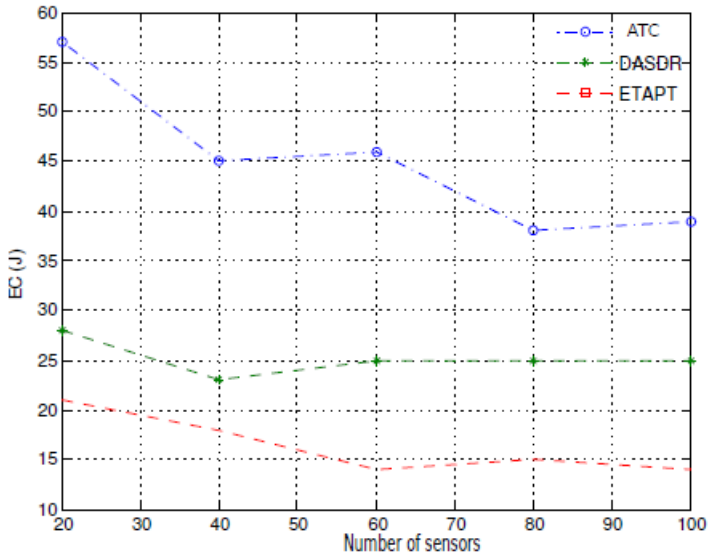


FIGURE 8: Evolution of EC vs Number of sensors.

Figure 9 shows Agg_{Time} vs. the locality radius. In this analysis, we set T_{max} = 6s, and vary the locality radius of sensors among [20, 30, 40, 50, 60] m. We can see that as locality radius increases, the Agg_{Time} decreases in all methods. That is because increasing the locality radius creates a disjoint network in which some sensors are not connected. This decreases the degree of connectivity of parents, and considerably reduces the Agg_{Time} of each parent. ETAPT reduces the Agg_{Time} compared to DASDR and ATC by around 31% and 60% respectively.

Figure 10 depicts the evolution of Agg_{Time} vs. the depth of the network. We set T_{max} = 6s, and vary the depth of the network among [3, 4, 5, 6]. As we have seen in Section 5, Agg_{Time} is a function of the depth of the network. We observe that as the depth of the network increases, Agg_{Time} also

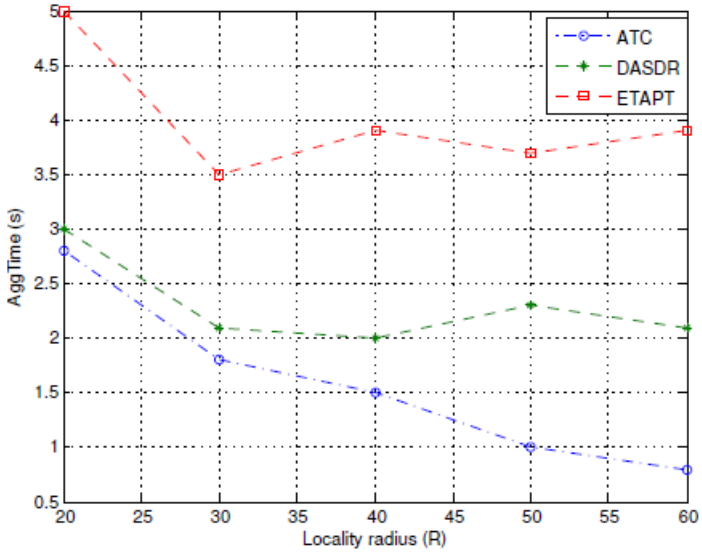


FIGURE 9: Evolution of A_{ggtime} vs Locality radius.

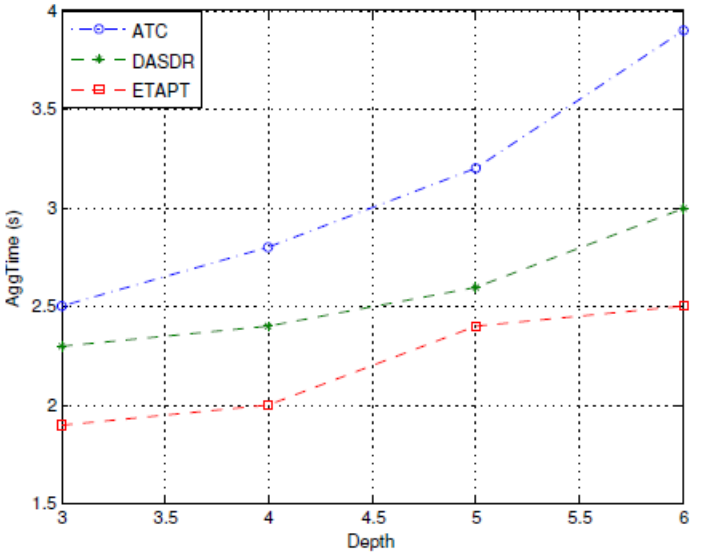


FIGURE 10: Evolution of A_{ggtime} vs Depth.

increases because, the deeper the tree, the more time parents in the tree will need to aggregate the data from leaves. In all three methods, while increasing the Depth, ETAPT reduces the Agg_{Time} compared to DASDR and ATC by around 17% and 40% respectively.

Figure 11 depicts the evolution of $Delay_{E2E}$ vs. the degree of connectivity. We set T_{max} = 6s, and vary the degree of connectivity of the network among [5, 10, 15, 20] with a network consisting of 200 sensors. ETAPT has a smaller $Delay_{E2E}$ compared to DASDR and ATC. This is because there is no need for each parent to synchronize with other parents in the tree before sending data.

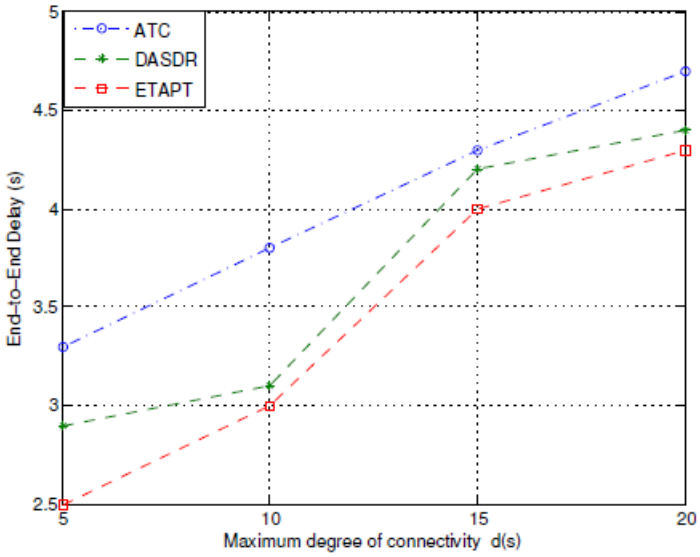


FIGURE 11: Evolution of of $Delay_{E2E}$ vs Degree of connectivity.

7. CONCLUSION

In this paper, we have proposed an efficient ETAPT algorithm using the ADAPT metric. Given the maximum acceptable latency, ETAPT's calculation takes into account the position of each parent, its number of leaves and the depth of the tree, allocating an ADAPT time to parents with more leaves, so increasing the data aggregation gain and ensuring enough time to process data from leaves. The results obtained show that our ETAPT provides a higher data aggregation gain with lower energy consumed, Agg_{Time} and $Delay_{E2E}$ compared to the alternative DASDR and ATC methods. Our suggested ETAPT algorithm is particularly useful in resource-constrained networks, since it does not need synchronization among sensors in the network.

In the future, we will take into account the cost of maintaining the tree in dynamic networks, evaluate the overhead as proposed in [16]. Later, we will study the relationship between waiting time and data aggregation gain in order to make it scalable in more complex WSNs.

8. REFERENCES

[1] D.Fotue, F.Melakessou, H.Labiod and T.~Engel. “Effect of Sink Location on Aggregation Based on Degree of Connectivity for Wireless Sensor Networks.” In proceedings of the 5th IEEE International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Jun 30-Jul 02, 2011, pp. 271-276.

[2] E.Fasolo, M.Rossi, J.Widmer and M.Zorzi. “In-Network Aggregation Techniques for Wireless Sensor Networks: A survey.” IEEE Wireless Communications, vol 14, no. 2, pp. 70-87, 2007.

[3] J.Stankovic, T.Abdelzaher, C.Lu, L.Sha and J.~Hou. “Real time Communication and Coordination in Embedded Sensor Networks.” IEEE Journal, vol 91, pp. 1002-1022, 2003.

[4] J. Y.Choi, J. W.Lee, K.Lee, S.Choi, W. H.Kwon and H. S.Park. “Aggregation Time Control Algorithm for Time Constrained Data Delivery in Wireless Sensor Networks.” In Proceedings of the 63rd IEEE Vehicular Technology Conference (VTC), May 2006, pp. 563-567.

[5] J.Zhang, Q.Wu, F.Ren, T.He and C.Lin. “Effective Data Aggregation Supported by Dynamic Routing in Wireless Sensor Networks.” In Proceedings of the IEEE International Conference on Communications (ICC), May 2010, pp.1-6.

[6] T. He, J.Stankovic, C.Lu and T.Abdelzaher. “SPEED: a Stateless Protocol for Real-time Communication in Sensor Networks.” In Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS), 2003, pp. 46-55.

[7] I.Solis and K.Obraczka. “The impact of timing in data aggregation for sensor networks.” In Proceedings of the IEEE International Conference on Communications (ICC), 2004.

[8] S.G.Quan and Y.Y.Kim. “Fast Data Aggregation Algorithm for Minimum Delay in Clustered Ubiquitous Sensor Networks.” In Proceedings of the IEEE International Conference on Convergence and Hybrid Information Technology (ICHIT), 2008, pp. 327-333.

[9] A.Sivagami, K.Pavai and D.Sridharan. “Latency Optimized Data Aggregation Timing Model for Wireless Sensor Networks.” International Journal of Computer Science Issues (IJCSI), vol. 7, no 3, 2010.

[10] C.Cheng, C.K.Tse and M.Lau. “A Delay-Aware Data Collection Network Structure for Wireless Sensor Networks.” IEEE Sensors Journal, vol. 11, no 3, pp. 699-710, 2011.

[11] H.N.Le, V.Zalyubovskiy, C.Hyunseung and J.Zhao. “Delay-minimized Energy-efficient Data Aggregation in Wireless Sensor Networks.” In Proceedings of the International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), Oct 10-12, 2012, pp. 401-407.

[12] U.Roedig, A.M.Barroso and C.J.Sreenan. “Determination of Aggregation Points in Wireless Sensor Networks.” In Proceedings of the 30th International Conference EUROMICRO, Aug 31- Sep 3, 2004, pp. 503-510.

[13] C.Lu, B.Blum, T.Abdelzaher, J.Stankovic and T.He. “RAP: a Real-time Communication Architecture for Large-Scale Wireless Sensor Networks.” In Proceedings of the IEEE Eighth Real-Time and Embedded Technology and Applications Symposium, 2002, pp. 55-66.

[14] E.Zegura, K.Calvert and M.Donahoo. “A Quantitative Comparison of Graph-based Models for Internet Topology.” IEEE Transactions Journal on Networking, 1997.

[15] Z.Cheng, M.Perillo and W.Heinzelman. “General network lifetime and cost models for evaluating sensor network deployment strategies.” IEEE Transactions Journal on Mobile Computing, vol. 7, pp. 484-497, 2008.

[16] D. Fotue “Efficient Data Aggregation and Routing in Wireless Sensor Networks”PhD Thesis, Télécom ParisTech, pp. 1-190, January 31st, Paris, France, 2014.

Twitter Based Sentiment Analysis of Each Presidential Candidate Using Long Short-Term Memory

Dhruval Shah *shah042@cougars.csusm.edu*
Computer Science and Information Systems,
California State University San Marcos
San Marcos, CA, 92096, USA

Yanyan Li *yali@csusm.edu*
Computer Science and Information Systems,
California State University San Marcos
San Marcos, CA, 92096, USA

Ahmad Hadaegh *ahadaegh@csusm.edu*
Computer Science and Information Systems,
California State University San Marcos
San Marcos, CA, 92096, USA

Abstract

In the era of technology and internet, people use online social media services like Twitter, Instagram, Facebook, Reddit, etc. to express their emotions. The idea behind this paper is to understand people's emotion on Twitter and their opinion towards Presidential Election 2020. We collected 1.2 million tweets in total with keyword like *"RealDonaldTrump"*, *"JoeBiden", "Election2020" and* other election related keywords using Twitter API and then processed them with natural language processing toolkit. A Bidirectional Long Short-Term Memory (BiLSTM) model has been trained and we have achieved 93.45% accuracy on our test dataset. We then used our trained model to perform sentiment analysis on the rest of our dataset. With the sentiment analysis results and comparison with 2016 Presidential Election, we have made predictions on who could win the US Presidential Election in 2020 with pre-election twitter data. We have also analyzed the impact of COVID-19 on people's sentiment about the election.

Keywords: Sentiment Analysis, LSTM, Deep Learning, Natural Language Processing, Data Mining.

1. INTRODUCTION

The US presidential election is one of the most important events occurring in 2020 in the United States. Political candidates' tweet about their campaign and their views on various topics and so do the daily twitter users. Due to the advancements in technology, many people have started using platforms like Twitter and Reddit to discuss and share their views on the news and hot topics around the globe. Everyday users post millions of tweets and express their emotions or views on the platform [1]. Performing data analysis on the tweets can be used by the ruling party and opposition party to decide their election strategy. As we are collecting real-time tweets, we can find the live response of public towards the election. Thus, collecting tweets and performing sentimental analysis on them would give us the predictions of who could win the US General Election 2020. In this paper we discuss our approach of using tweets related to US Presidential Elections in 2016 and 2020 and perform sentimental analysis on both. We then try to find the comparisons between the two and make predictions for 2020 Election based on sentimental analysis.

There are many techniques available to do sentiment analysis. For this work, we are going to use Recurrent Neural Network based Long-short Term Memory (LSTM) algorithm. To better

understand how LSTM works and how we can use it for our twitter data, we referenced Tang Y. and Liu J.'s work on "Gated Recurrent Units for Airline Sentiment Analysis of Twitter Data" [2], After that we trained LSTM model using labelled tweets for the prediction and sentiment analysis. In section 4, we will discuss about this approach in more detail under Methodology. Section 2 introduces the background of the US election and Twitter where we will discuss the first use of twitter for election campaigning and its impacts on results. In section 3, we will discuss prior work that is emoji-based sentimental analysis on US Elections and why it is not accurate. In section 4, we will discuss about the implementation and methodology of our work in detail. We labelled the data by using VADER API and got 93.45% accuracy on our test data. In this section, we will discuss about this in more detail. In section 5, we will do analysis of our resulted data and subsequently, we will analyze the results of the 2016's and 2020's election. We will take reference of Donna Ladkin's work [3] to understand why Clinton lost the election even though she had more positive sentiment than Trump in 2016. In section 6, we will conclude why our work is more accurate than the previous works and predict the chances of who wins the presidential candidate in 2020.

2. BACKGROUND

Social media is a platform where people express their views and opinions on various issues. Twitter currently receives around 500 million tweets every day (As per the data from August 2013) in which people mostly share their opinions about trending news or headlines [4]. Political parties hire experts who do sentiment analysis of these tweets and help them to decide their strategies for the election campaign. One way to predict the possibility of a candidate winning the election is by doing the sentiment analysis on the tweets.

In 2008, Barack Obama used Twitter to promote his candidacy for the presidential elections [5]. Obama's campaign employed 100 individuals to run his digital presence. This made a huge impact on the election results. Since then, before each election, parties hire research associates to do public sentiment analysis to conduct election manifesto. Through this way, they can understand what their voters want from the next president. People from all over the world tweet about the presidential candidates and the US elections in order to reflect their respective views. Democratic party's presidential candidate Joe Biden and Republican party's presidential candidate Donald Trump were the top ten political trending twitter topics on Twitter on August 12, 2020.

3. RELATED WORK

English is the highest spoken language in the world. Most research efforts are done in English language for sentiment analysis. In 2009, A. Tumasjan, T. O Sprenger, P. G Sandner and I. M Welpe did prediction with Twitter in 2009 and found that 50% users tweeted only one time in their corpus and rest 50% made 90% of the tweets [6]. Their data set was of approx. 100K tweets which is too ambiguous to predict for the large population of a country.

In the paper "A novel classification approach based on Naïve Bayes for Twitter sentiment analysis", they used positive and negative words file to predict the sentiment of a sentence with Naïve Bayes classification. But since the number of positive and negative words are not always equal, there are chances of getting Biased result. [7] To address this issue, they proposed two ways. First one is by counting the number of positive and negative words in calculating weights while seconds approach identifies significant words to predict the class of test document. By this way they achieved 85.33 % accuracy. While it's almost impossible to have all the positive words in positive words file and negative words in negative words file, applying this model to our 10 M tweets dataset may give less accuracy and there are chances of getting wrong sentiment for many tweets.

Gautam, G., & Yadav, D proposed set of techniques of Machine Learning with Semantic analysis. [8] They did performance comparison of Naïve Bayes, Maximum Entropy, Support Vector Machine and Semantic Analysis (WordNet). And they got accuracy ranging between 88.2% to

89.9% accuracy. The accuracy is quite good if we use their approach but this approach will fail to predict correct sentiment sometimes as the word order is not preserved.

Sharma, P. & Moh, [9] did prediction of Indian Election Using Sentiment Analysis on Hindi Twitter. For this work, they used three different algorithms which are Naïve Bayes, SVM and Dictionary Based. They got 62.1%, 78.4% and 34% accuracy respectively.

B. Joyce and J. Deng gathered tweets from 2016's presidential election based on specific words like Hillary Clinton and Donald Trump [10]. They have used Twitter's streaming API for tweets collections. The keywords they included were ***democratic*** and ***republican*** and the full name of the top political candidates like **Donald Trump**. Since they used Twitter's official tweets streaming API, they collected around 79 million tweets by two month's 10,000 unique users. Then they filtered 1.9 million tweets that contain emojis. After deleting retweets, the data set further depleted to 783K tweets. With the *Emojis Sentiment Ranking*, they determine each tweet's emotional orientation. For that, they extracted the emoji from the sentence and then converted it into Unicode. Then applied regex on Unicode and found score from the customize list of 522 unique emoji characters. This score will determine emotional orientation or sentiment of the sentence. To build a sentiment classifier, they used a multinomial Naïve Bayes classifier.

Since not every sentence contains an emoji, only judging the result based on emojis will not be accurate. There may be some tweets where the user has added an emoji as a sarcastic tone. Also, word ordering matters while doing sentiment analysis. For example, one sentence is *I have to read this book* and another sentence is *I have this book to read.* Both examples have different meaning. This simple example is one of the examples where we can say word ordering matters while doing sentiment analysis.

To overcome the above problem, in our work we developed a model that will try to analyze the sentiment based on word number. Deep learning approaches like Long Short-Term Memory (LSTM) is one of the Recurrent Neural Network approaches where model is trained with the word's ordering whereas in the already existing work, they only considered emojis while doing the analysis. Due to the above-mentioned reasons, LSTM seems to be the perfect fit for doing sentiment analysis. So, we will be using LSTM model for our analysis.

4. METHODOLOGY

This section discusses about our methodology. To start with, we downloaded Twitter's 55,000 prelabelled tweets. For training, we will used random 40K tweets and the rest of 15K tweets for testing purposes. But these tweets were not sufficient to train the ML model. As the part of next process, we will collect tweets data of Joe Biden and Donald Trump. For this, we used following keywords: "JoeBiden", "DonaldTrump", "BidenHarris", "US Election 2020", "TrumpPence". We collected around 1.25M tweets for the period of October 2019 to July 2020. Figure 1 shows our proposed system workflow.

4.1. System Overview

With the tweets collected, we did data cleaning and data preprocessing. In Data Preprocessing, we check if sentence language is English or not. If the language is different then we remove those sentences from our dataset. In data cleaning, we remove unwanted texts, URLs, language slangs, emojis, hashtags, mentions, and retweets. After data preprocessing and data cleaning, we were left with almost 1.2M tweets. We randomly took 55K tweets from this dataset. We labelled these tweets using VADER sentiment API [11]. Once these 55K tweets were labelled, we divided 40K tweets as training and rest 15K tweets for testing to make training dataset big enough for model training. We passed these 80K tweets data to train our LSTM model. Before passing tweets to the model, we implemented embedding layer. We used Word2Vc model which initializes random weights and learns to embed all words in the dataset. Once embedding is done, we use LSTM model for RNN layers. To prevent our model from overfitting, we used dropout techniques. It drops irrelevant information from the network as they don't contribute to the

process of enhancing our model accuracy. We have used dense layer in this model which connects every input with every output using weights. We have used Relu activation function as it helps complex relationship in the data to be captured by the model. Relu is more efficient than Tanh or Sigmoid activation function because Relu activates certain number of neurons.

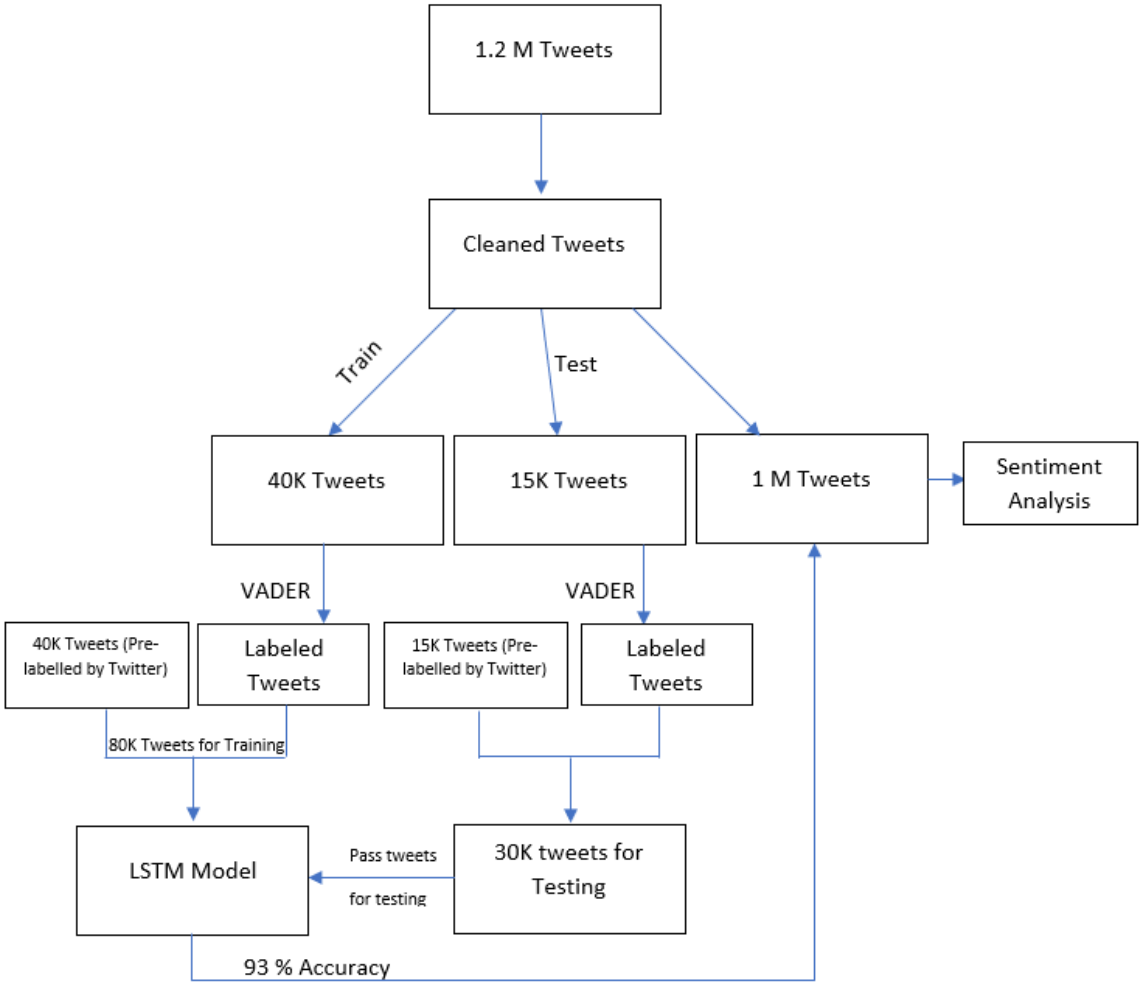


FIGURE 1: Project process workflow.

To test our model, we passed the rest 15K pre labelled tweets from twitter and 15K labelled tweets from VADER API. We got 93.45 % test accuracy which is better than previous works [10].

We passed rest of tweets to this model for labelling. Once these tweets were labelled, we counted the number of positive and negative sentiment tweets. We also passed 2016's tweets data and labeled them. Once the data from both the elections was labelled, we counted number of positive and negative tweets in these to do the analysis.

We have divided the process into 4 major sections below. Data Gathering, Data Processing, Model Training and Testing on test data. All of these steps are very crucial for training the model. The model must be fed with correctly labeled data in order to generate the most accurate results. To give a broader overview, here is the overall process. Using Get Old Tweets (GOT3) API [12] we will collect the tweets. The raw data might be very informal and unstructured. It may contain hyperlinks, emojis, mentions, and retweets. To convert it into trainable data, data cleaning is required. Data cleaning will be performed on tweets that were collected from October 1, 2019 to July 31, 2020 and also October 1, 2015 to November 7, 2016. During the data processing,

hashtags, mentions, emojis, links (photo, video, or gifs) and retweets get removed. After that, the data is divided into 80-20 % for training and testing respectively. Once we have the data ready, we use it to feed into the model to train and then test its accuracy with the test data set.

4.2. Data Gathering

Initially, we start collecting tweets using Tweepy streaming API. Since February 1, 2018, standard users cannot access past tweets for more than 7 days. So, we used GOT3 API (Get Old Tweets). GOT3 is a GitHub repository which uses URLlib for fetching tweets from Twitter's advance search. Like this, we collected tweets from October 1, 2019 to July 31, 2020, and October 1, 2015 to November 7, 2016. We randomly selected 1.25M tweets from this time period and their geographical location as United States. By doing data preprocessing, we checked each tweet and filtered out non-English tweets. For this, we used Python's Langdetect library. Then we did data cleaning. In this, we filtered out tweets which only contained emojis, images, gifs, or video. Of 1,250,000 tweets, we obtained 1,200,718 tweets that were containing tweets and may contain images, video, or gifs.

For 2020's data, we used keywords such as "US Elections" with the twitter usernames "realDonaldTrump" and "JoeBiden". Then we collected 2016's tweets for presidential candidates Donald Trump and Hillary Clinton using the same GOT3 API. We collected tweets containing keywords like US Elections, realDonaldTrump, HillaryClinton. Some of the keywords and hashtags we used for data gathering are US Elections, US Elections 2020, TrumpPence, republican election, presidential elections, election 2020, donald trump for 2020, BidenHarris. And we also collected election related tweets from the twitter handles JoeBiden, POTUS, DonaldTrump, KamalaHarris, Mike_Pence. In the data gathering, we only collected tweets from United States. We did this by filtering tweets by its location (in our case it's United States).

Example 1: @JoeBiden VP Biden I wish you wouldn't debate Trump unless he: Debate #1 Trump shows taxes Debate #2 Tells Putin stop killing our soldiers Debate #3 Put sanctions on Putin for 2016 election hacks & I wish you'd publicly challenge him with this. #Biden/Rice 2020

Example 2: Best @realDonaldTrump political ad I've seen yet. @TheDemocrats. will attempt just about anything to impact the 2020 election. Pure evil. cc @Scavino45 @parscale

Example1 and Example 2 are the sample tweets from the training dataset and the words in bold are those words which will would cause problem while training model. So, we need to remove these words, special characters, URLs and images from the tweets. We remove them during Data Processing.

4.3. Data Processing

As the language used on Twitter is informal and unstructured, directly using this data will result in failure of our model to understand some words and it will decrease model accuracy. Thus, we performed the following steps to each tweet we collected.

1. Converted all tweets to lowercase as some text mining algorithms are case sensitive.
2. Browsed through all the tweets and replaced words like - no, not, never, cannot, don't, doesn't with not.
3. Removed repeated words and slang from the sentence and cleaned it.

For example, the sentence is Hiiii! Good to see u then we will remove Hiii and replace it with a word Hi and we will replace word u with you. So, after processing the sentence we will get Hi! Good to see you. We did this using python's nltk tokenizing. After data processing our two example tweets become the following:

Processed Example 1: i wish you would not debate trump unless he debates 1 trump shows taxes debate 2 tells putin stop killing our soldiers debate 3 put sanctions on putin for 2016 election hacks i wish you would publicly challenge him with this biden/rice 2020.

Processed Example 2: best realdonaldtrump political ad I have seen yet. thedemocrats will attempt just about anything to impact the 2020 election. pure evil. cc scavino45 parscale

As we have converted tweets to the trainable tweets, now we can proceed with the model training.

4.4. Model Training

Model training in machine language is the process of feeding an ML algorithm with data to help identify and learn good values for all attributes involved [13]. There are several types of machine learning models, of which the most common ones are supervised and unsupervised learning [14]. We will train the model using Long Short-Term Memory (LSTM) algorithm. LSTM is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed forward neural networks, LSTM has feedback connections. It can not only process single data points but also entire sequences of data. As we have used Bidirectional LSTM, $\vec{a}^{\langle t \rangle}$ will represent forward and $a^{\leftarrow <t>}$ will represent backward activation.

$$\hat{y}^{<t>} = g(W_y[\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y)$$

Unlike standard feedforward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data [15].

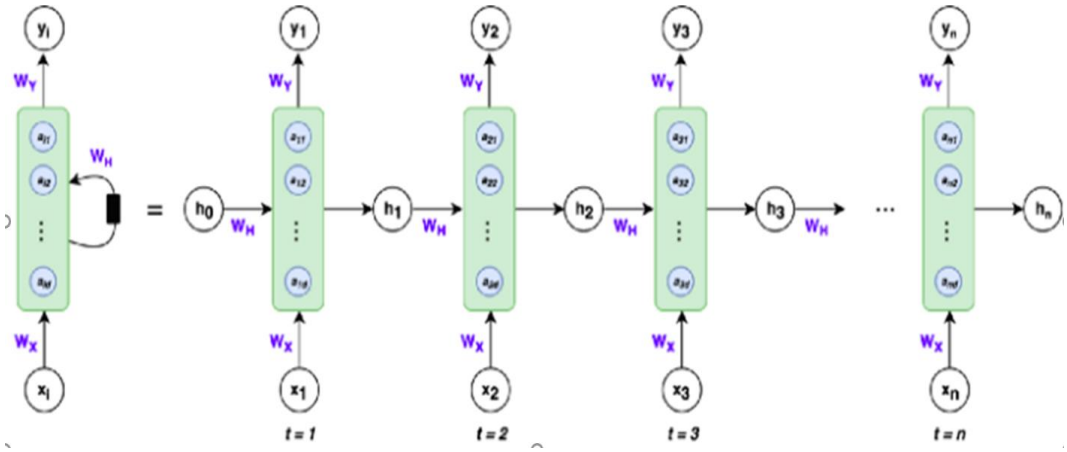


FIGURE 2: Recurrent Neural Network [16].

In a traditional neural network, all inputs and outputs are independent of each other but while doing sentiment analysis input word sequence also matters. In Figure 2 we show the architecture of RNN. Hence, we need to remember the previous work while guessing the next word. So, RNN remembers every information upon time [17]. RNN converts individual activation into dependent activation by providing the same weights to each layer which decreases the increasing parameters and memorizing each previous output by giving each output to the next hidden layer. So now, these hidden layers can be joined together so that the weights and bias of all the hidden layers is the same as a single recurrent layer.

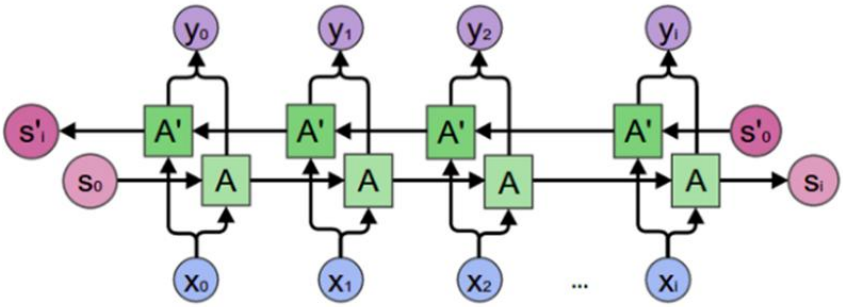


FIGURE 3: Long Short-Term Memory [18].

As shown in Figure 3. Bidirectional recurrent networks are just the same as putting two independent RNNs together. This Bi-LSTM will have two networks, forward and backward information sequences at every step. This will run input in two ways, one is from future to past and another is from past to future. The only difference from bidirectional to unidirectional is that it also runs backward, and you can preserve information from the future using two hidden states. There are some activation functions available like relu, tanh, softmax. We have used relu activation function.

Tuning Parameters	Bi-LSTM Model
Learning Rate	0.01
Dropout	0.5
Embed Size	64
Vocab Size	1000
No Filters	256
Max Length	280

TABLE 1: Model parameters.

We passed English tweets labeled positive or negative to the model. The parameters used in the model are presented in Table 1. Once the training is completed, we saved the model using model.save() which will help us to label the data more rapidly and use the already saved data and there will be no need of saving the data again.

4.5. Testing

Model testing is the process of checking the correctness of the model on the pre-labeled data. To test the model, we passed the pre-labeled data to the saved model. Once the test data is passed to the BiLSTM, the number of parameters is calculated. The number of parameters is the connection between layers and the biases in every layer. The vocab size and max_length is fixed which are 1000 and 280 respectively.

After 25 epochs, the model accuracy is 93.45 % which means that out of 100 sentences, the model will correctly label 93 sentences as positive or negative sentiments. BiLSTM keeps records of previous words and also the next words. For example,

Sentence 1: The person carrying black suitcase is Mr. Mosbey.

Sentence 2: They were shocked when they saw the person carrying water in the desert!

In the above sentences, we cannot say what will be the next word after carrying is black suite case or water. It depends on the context of a sentence. As word orderings are preserved, model accuracy will increase.

Name	Positive %	Negative %
Trump	52.04	47.96
Biden	59.54	40.46

TABLE 2: Sentiment analysis result for 2020 presidential election related tweets.

5. ANALYSIS

For 2020 US Presidential Election predictions, we passed tweets of Trump and Biden that we have collected from October 2019. As shown in Table 2, we got 52% positive sentiment for Trump and 59.5% positive sentiment for Biden. For simplicity, we removed neutral tweets from the consideration and kept only positive or negative sentiment tweets. To have a better understanding of the sentiment, we collected tweets before and after the Covid-19 pandemic for 2020.

We used the same keywords and twitter handles to collect tweets for before and after Covid-19. We took tweets from October 2019 to March 2020 as before Covid-19 tweets and March 2020 to July 2020 as after Covid-19 tweets. The reason behind doing before and after Covid-19 is to analyze how much sentiment has changed in people towards Trump and Biden. Many strategists believe that Trump Administration has failed to stop the Covid-19 in the United States. This means that people are not happy with the Trump administration which can be analyzed by the data we collected.

Name	Positive %	Negative %
Trump (Before)	48.81	51.18
Biden (Before)	54.54	45.36
Trump (After)	49.10	50.90
Biden (After)	66.67	33.33

TABLE 3: Sentiment analysis result before and after Covid-19.

As shown in Table 3, there is a slight increase in people's positive sentiment for Trump. Whereas for Biden, people's positive sentiment for him has increased by 12%. After collecting this data, we collected 2016's data and tried finding some correlation between both the elections.

Name	Positive %	Negative %
Trump	55.73	44.27
Clinton	78.26	21.74

TABLE 4: Sentiment analysis result for 2016 presidential election related tweets.

We have collected 2016's tweet data. The results we got is very appalling. After a significant amount of training and testing, as shown in Table 4 we got 55.73 % positivity for Trump and 78.26 % positivity for Clinton. Usually, positivity denotes that people are in favor of making Clinton the President. But she could not make it and instead Trump won the election to become the President.

To understand this in more detail, we took reference of Donna Ladkin's work [3] where it states the reason why Clinton lost the election even after winning the popular vote by 2.8 million votes in

2016. Trump won the election because Electoral college votes in the States in which he won the popular votes exceeded that of those held by the States in which Clinton won the popular votes. This is the fifth time when a candidate has won by the popular votes but later lost because the Electoral college didn't vote for them.

After comparing both the election data, people have more positive sentiment towards Biden compared to Trump in 2020. So, the possibility of winning the election is higher for Biden as opposed to Trump provided that the Electoral college votes to go for Biden.

6. CONCLUSION AND FUTURE WORK

We have compared the previous work on Machine Learning based Lexicon sentiment analysis with our work on positive/negative words-based Machine Learning algorithm. Though positive/negative words-based Machine Learning algorithm should work well, it failed to give correct sentiment in many cases. So, instead of labelling tweets with positive and negative words, we used libraries to label the tweets and trained model using LSTM. This approach gave better accuracy and reduced the labelling time of each tweet. After that we wanted to find the reason as to why Clinton lost the election in 2016 to Trump despite having much positive sentiment compared to Trump. We discussed and concluded that though public sentiment was much positive for Clinton, the electoral votes didn't go for Clinton in 2016 and thus, she lost the election. In 2020 elections, public sentiment is more positive for Biden compared to Trump. So, if the electoral votes also go for Biden, then he could win the 2020 US General Election.

According to the article [19], there are more republican twitter users to democrats. So, there are chances of getting biased result. So, in order to get more accurate result, we should consider other social media sources like Reddit, Facebook, Blogposts, chat rooms and email campaigns. For these social medias too, our model can be used to predict sentiment for each candidate. To use our model, the data should be in text form (Images, URLs should be removed). And before passing data to the model, the data should be cleaned, structured and labelled.

For future work, we can use official Twitter API to get the more data. GetOldTweets API will fetch very less tweets compare to Twitter Premium APIs. Also, one can use retweets, likes, and share count to weight on other parameters. Also, we can consider keeping track of trending tweets and the social media account getting more tweets in order to consider the current popularity of a candidate. These parameters can help us understand and predict sentiment in better way. For future elections result predictions, our model can be used for sentiment analysis as we have labelled data using VADER API and our model accuracy is 93.45%.

7. REFERENCES

[1] Hao, M., Rohrdantz, C., Janetzko, H., Dayal, U., Keim, D. A., Haug, L.-E., & Hsu, M.-C. (2011). Visual sentiment analysis on twitter data streams. 2011 IEEE Conference on Visual Analytics Science and Technology (VAST). doi:10.1109/vast.2011.6102472.

[2] Tang Y. and Liu J., Gated Recurrent Units for Airline Sentiment Analysis of Twitter Data, Technical Report, Stanford University, 2011.

[3] Ladkin, D. (2017). How did that happen? Making sense of the 2016 US presidential election result through the lens of the ‘leadership moment.’ Leadership, 13(4), 393–412.

[4] Twitter Usage Statistics, https://www.internetlivestats.com/twitter-statistics/, accessed: 2020-08-08.

[5] Social Media on the Campaign Trail: Barack Obama and Donald Trump, accessed on August 8 2020, https://contentgroup.com.au/2017/09/social-media-campaign-trail-obama-trump/.

[6] A. Tumasjan, T. Sprenger, P. Sandner, and I. Welpe. Predicting elections with Twitter: What 140 characters reveal about political sentiment, Proceedings of the International AAAI Conference on Web and Social Media (ICWSM), pp. 178–185, 2010.

[7] A novel classification approach based on Naïve Bayes for Twitter sentiment analysis. (2017). KSII Transactions on Internet and Information Systems, 11 (6). https://doi.org/10.3837/tiis.2017.06.011.

[8] Gautam, G., & Yadav, D. (2014). Sentiment analysis of twitter data using machine learning approaches and semantic analysis. 2014 Seventh International Conference on Contemporary Computing (IC3). doi:10.1109/ic3.2014.6897213.

[9] P. Sharma and T. Moh, "Prediction of Indian election using sentiment analysis on Hindi Twitter," 2016 IEEE International Conference on Big Data (IEEE Big Data 2016), 2016, pp. 1966-1971, doi: 10.1109/BigData.2016.7840818.

[10] B. Joyce and J. Deng, "Sentiment analysis of tweets for the 2016 US presidential election," 2017 IEEE MIT Undergraduate Research Technology Conference (URTC), Cambridge, MA, 2017, pp. 1-4, doi: 10.1109/URTC.2017.8284176.

[11] Vader Sentiment API, https://pypi.org/project/vaderSentiment/.

[12] Jefferson Henrique, Get Old Tweet in Python, https://github.com/Jefferson-Henrique/GetOldTweets-python, accessed: 2020-08-08.

[13] What Is Model Training, https://oden.io/glossary/model-training/, accessed: 2020-08-08.

[14] Alloghani M., Al-Jumeily D., Mustafina J., Hussain A., Aljaaf A.J. (2020) A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science. In: Berry M., Mohamed A., Yap B. (eds) Supervised and Unsupervised Learning for Data Science. Unsupervised and Semi-Supervised Learning. Springer, Cham. https://doi.org/10.1007/978-3-030-22475-2_1.

[15] Long Short-Term Memory, https://en.wikipedia.org/wiki/Long_short-term_memory/.

[16] Khuong, Ben. RNN Figure. The Basics of Recurrent Neural Networks (RNNs). 11 June 2020, medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f.

[17] Khuong, Ben. RNN Definition. The Basics of Recurrent Neural Networks (RNNs). 11 June 2020, medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f.

[18] Aggarwal, Raghav, Bi-LSTM, published in 4 July 2019, medium.com/@raghavaggarwal0089/bi-lstm-bc3d68da8bd0.

[19] How Democrats and Republicans Use Twitter, Pew Research Center, published in Oct 15, 2020, https://www.pewresearch.org/politics/2020/10/15/differences-in-how-democrats-and-republicans-behave-on-twitter/ accessed: 2021-04-15.

INSTRUCTIONS TO CONTRIBUTORS

The *International Journal of Computer Science and Security (IJCSS)* is a refereed online journal which is a forum for publication of current research in computer science and computer security technologies. It considers any material dealing primarily with the technological aspects of computer science and computer security. The journal is targeted to be read by academics, scholars, advanced students, practitioners, and those seeking an update on current experience and future prospects in relation to all aspects computer science in general but specific to computer security themes. Subjects covered include: access control, computer security, cryptography, communications and data security, databases, electronic commerce, multimedia, bioinformatics, signal processing and image processing etc.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCSS.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Started with Volume 15, 2021, IJCSS is appearing with more focused issues. Besides normal publications, IJCSS intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJCSS LIST OF TOPICS

The realm of International Journal of Computer Science and Security (IJCSS) extends, but not limited, to the following:

- Authentication and authorization models
- Computer Engineering
- Computer Networks
- Cryptography
- Databases
- Image processing
- Operating systems
- Programming languages
- Signal processing
- Theory
- Communications and data security
- Bioinformatics
- Computer graphics
- Computer security
- Data mining
- Electronic commerce
- Object Orientation
- Parallel and distributed processing
- Robotics
- Software engineering

CALL FOR PAPERS

Volume: 15 - **Issue: 4**

i. Submission Deadline : June 30, 2021 **ii. Author Notification:** July 31, 2021

iii. Issue Publication: August 2021

CONTACT INFORMATION

Computer Science Journals Sdn BhD

B-5-8 Plaza Mont Kiara, Mont Kiara

50480, Kuala Lumpur, MALAYSIA

Phone: 006 03 6204 5627

Fax: 006 03 6204 5628

Email: cscpress@cscjournals.org

COMPUTER SCIENCE JOURNALS SDN BHD
B-5-8 PLAZA MONT KIARA
MONT KIARA
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6204 5627
FAX: 006 03 6204 5628
EMAIL: cscpress@cscjournals.org