Address Fraud: Monkey Typed Address Classification for e-Commerce Applications

T. Ravindra Babu
Flipkart Internet Private Limited
Bangalore, India
ravindra.bt@flipkart.com

Vishal Kakkar
Flipkart Internet Private Limited
Bangalore, India
vishal.kakkar@flipkart.com

ABSTRACT

E-Commerce companies face a wide variety of fraudulent activities. Machine Learning models are continuously built to detect them for their effective mitigation. Randomly typed alphanumeric characters as customer addresses is one peculiar fraud. The orders are placed with such addresses. We refer to them as monkey-typed addresses. Accurate methods of identifying such addresses is important in reducing operational cost. The current work presents machine learning based approaches that classify a given address as normal or monkey-typed with a high accuracy. The approach integrates address preprocessing, novel feature generation and classification.

KEYWORDS

Geographical addresses, monkey-typed, machine learning, classification

ACM Reference format:
T. Ravindra Babu and Vishal Kakkar. 2017. Address Fraud: Monkey Typed Address Classification for e-Commerce Applications. In *Proceedings of ACM SIGIR eCom Workshop, Tokyo, Japan, Aug 2017 (SIGIR eCom),* 7 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

Online e-commerce companies facilitate its users to order items online and deliver them to the suggested addresses. Users register themselves by entering one or more of their addresses that would ensure that an ordered shipment is positively delivered to them. Thus the customer addresses form important part of e-commerce companies. A number of challenges in relation to the addresses, especially, in developing countries include[2] lack of pre-defined structure, absence of geolocation information, spell variations in address components due to different literacy levels in the population of customers, etc.

In order to avail price discounts or with possible fraudulent intent, there are occasions where orders are placed with addresses containing randomly typed alphanumeric characters. We refer to them as *monkey-typed addresses* in this work. Early identification of such addresses in the operations chain is necessary to reduce operations cost. For example, when such addresses remain undetected, it would lead to activation of a number of stages in the operations chain from order management system to supply chain leading to ship corresponding packed item to delivery hub for delivery. This results in avoidable mis-utlilization of resources.

Address systems are well studied. Davis and Fonseca [7] discuss theory of addresses while discussing on certainty of locations in address geocoding systems. Address structures differ across countries. In Japan, the houses are numbered according to their date of construction and most streets do not carry names. In Korea, the houses are numbered within a neighbourhood with reference to a hierarchy of area names instead of numbers. The cited work also notes that in countries like India and Brazil, a complete and organised address database that could be correlated with their geolocations is lacking. In view of absence of proper recommended structure for the addresses, it is empirically noticed that a given address is written with different hierarchy levels by different residents of the same dwelling. We work on such a system which has multiple variants in writing an address to the same given location which we refer to *normal addresses* in contrast to *monkey-typed addresses*. We discuss more elaborately on various aspects of such addresses and present sample addresses in Section 3.1.

In order to classify these two classes of addresses, we propose two novel solutions that classify monkey-typed addresses with very high classification accuracy. The methods integrate the following aspects.

- Address preprocessing
- Novel way of feature generation
- Pattern Classification

We organise the paper in the following manner. We present motivation for this work and discuss related literature in Section 2. Section 3 contains proposed methods, discussion on challenges in data, and preprocessing steps. Section 4 contains a discussion on experimentation and results. We present approaches to solve class imbalance problem in Section 5. Section 6 contains summary and directions for further work.

2 MOTIVATION

E-Commerce companies face a wide variety of fraudulent activities. They include payment frauds, return frauds, delivery frauds, lost-and-stolen articles, fake reviews and fraudulent ratings. Such frauds are prevalent in both developing and developed countries. They are well discussed and many solutions are proposed [1, 3, 10, 15, 16]. The nature of fraud, which is the focus of the proposed work, was not reported earlier. It is interesting explore possible reasons for such a fraud. Generally, most e-Commerce platforms offer a number of attractive discounts. In order to reach a larger customer base, the discounts offered are limited to purchase of one

SIGIR eCom, Aug 2017, Tokyo, Japan

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

or a small number of units. Fraudulent users, presumably resellers, would order multiple items beyond the prescribed limit through false addresses. The resellers are profited by selling those products in the market at a price higher than the discounted price. Since the objective of the company is to serve larger customer base, such fraud defeats this purpose. In this context, they possibly resort to monkey typed addresses in order not get detected through multiple orders. With a nexus at later stages, the orders could get delivered too. The orders that contain monkey typed addresses could also possibly be created by fraudsters to cause a strain on the system. When unchecked at the time of order, such addresses, would lead to mis-utilization of resources. This necessitates a solution to the problem. The solution should be deployable and trackable. In view of this, we approach this as a 2-class classification problem.

We have not come across any peer-reviewed, published work on address fraud. Most of the discussions in blogs that relate address fraud [6] to false declaration of one's address linked to credit card fraud. The problem we propose to solve is unique, but falls in the broad class of fraudulent activities that e-commerce companies face. In view of this, we discuss some closely related works.

E-commerce companies face a number of fraudulent activities. They include frauds caused by buyers, sellers and logistic partners. Some relate to exploiting discounts offered by the companies to resell the products for profit. The activities also include number of fraudulent claims and counter claims about non-receipt of a genuine product, claims of receipt of part of shipments as against a promised list, etc. Some are more subtle and difficult to detect. We discuss some such frauds and their solutions.

Credit card fraud detection is one important activity. They are broadly classified into two [3], viz., application fraud, where fraudsters apply for new cards with false information and behavioural fraud, which include mail theft, stolen cards, counterfeit card, usage in the absence of the card holder, etc. In general, models incorporate binary classification approaches. The work provides a detailed account of credit card fraud, data mining approaches to solve the problem, feature identification, and comparative experimental results. The experiments are conducted on real-life dataset using logistic regression, support vector machines and random forests. Trust fraud which relates seller fraud in hiking their own ratings and reducing competitor's ratings is rampant in e-commerce industry [16]. The work provides interesting insights on its prevalence in Chinese C2C (customer to customer) e-commerce industry. The sources of fraud include professional scammers, and nexus between sellers, scammers, buyers, trust fraud companies and logistic companies. The work also traces historical evolution of trust models. It proposes a dynamic trust model that integrates weightage for transaction amount, time decay to account for recency of transactions, and trust factors such as quality, service and shipping speed. The solution is shown to provide reliable trust scores. Anomalous behaviour indicates potential fraud activity. Anomaly detection with edge-attributed graphs on complex heterogeneous networks is explored in [15]. With product ratings by users as data, the proposed work identifies networks of users that are anomalous. The proposed approach extends to heterogeneous networks, support independent edge attributes and has ability to identify and rank anomalies. It is shown to have high precision and is scalable. Opinion fraud [1] relates to spamming the reviews either to defame or artificially hype a target product. User generated reviews play significant role in e-commerce retail. People with fraud intent spam the reviews with an intent to distort the perception about the product. Fake online review detection is studied in [1]. The work presents a unsupervised, scalable framework that exploits the correlation of labels between users and products, in order to detect fraudulent users and online fake reviews. User profiling is used to detect fraud in mobile communication networks [10]. The fraud in this case relates to illegal use of services which could cause loss to service provider. The work provides multiple approaches to solve the problem, such as feed-forward neural network for normal and fraud classes, Gaussian mixture model for user behaviour modeling to detect sudden changes in behaviour, and cost-sensitive classification.

We propose two approaches to classify a given user address as anomalous or genuine. We discuss them in detail in Section 3.

3 PROPOSED METHODS

The problem can be defined as, *classification of a given address into either normal or monkey-typed accurately and efficiently.* Efficiency is required for real time detection of such fraud in customer on-boarding process. The solution is also useful to clean the historical database of customer data. A study of monkey-typed and normal addresses initially indicated that a possible solution can be obtained through language modeling. We experimented with standard language models and found as not effective as shown later in Section 3.3. In view of this, we propose to solve this problem in the following two ways of pattern representation.

(1) **Approach-1:** In this approach, we derive text features as groups of characters. Through machine learning models, we learn from data of those combinations of characters that are normal or abnormal and culminate in classification of an address.
(2) **Approach-2:** In this approach, we exploit importance of vowels in English [5]. We mark vowels and non-vowels as binary features. Regularity in the presence of vowel helps to classify a normal address vis-a-vis an abnormal address.

    The motivation for Approach-2 is its possible generality across a large country like India, where different languages are spoken and the address words would be distinctly different. However, the words even though expressed in English, do have similar vowel combination. It should be mentioned that the method would not work if (a) the syllables contain only consonants or (b) addresses contain only numbers. We, however, did not come across monkey typed addresses as numbers alone, possibly because the addresses are expressed as characters and fraudsters mimic them with jumbled characters.

We achieve this objective through the following steps of the proposed methods.

(1) **Preprocess the addresses:** The addresses at the time of entry would contain a number of special characters such as $, ^,%, *,/, etc. Earlier studies [2] show that they are not helpful in classification of the addresses. Such terms are eliminated. The addresses also contain mixture of upper and lower case. They are all converted to lower case.

(2) **For Approach-1, generate fixed length features that are devoid of repititve substrings:** The normal as well as monkey-typed addresses are of non-uniform length. In order to bring tractability such addresses are reduced to fixed length substrings that form features. We consider two different lengths of substrings. Repeated substrings of characters are identified and removed. This is done so as to avoid over-representation of repeating substrings.
(3) **For Approach-2, we make use of presence of vowels in an address:** In this approach, we transform the address data into an binary sequence of vowel and non-vowels. Thus normal and monkey-typed addresses form a sequence of binary features. The present work is applicable for terms containing vowels like in Indian languages, although expressed in English. Some examples are k**o**r**a**m**a**ng**a**l**a**, an area name in Banaglore, and dh**a**ul**a** k**ua**n, an area name in Delhi. It may be noted here that lengths of different patterns containing binary values features are not same. In the given dataset, the maximum of pattern lengths is noted. It is extended by another 25% of its length in order to take care of unseen test patterns, say, l. Each pattern is appended with 0's in order till length of pattern reaches common length, l.
(4) **Label the patterns as normal or monkey-typed addresses:** In order to classify the addresses using supervised methods, the addresses are labeled as normal and monkey-typed. This is a manual activity.
(5) **Training, Test and Cross-validation:** The dataset is divided into training and test datasets. We consider multiple such sets by randomly selecting from those two sets in a mutually exclusive manner.
(6) **Classify the data and record the average performance:** We carry out experiments, with different mutually exclusive train-test combinations. The performance is presented.

For classifying a given address as normal or monkey-typed, it is not possible to obtain all combinations of monkey-typing possibilities. We initially place our efforts in understanding the conventional addresses as well as monkey-typed addresses. Subsequently we devise methods for effective classification of such addresses. We discuss about the nuances of the data and its transformation as it passes through important stages of preprocessing.

3.1 Data Description

The number of addresses is in the order of millions. Monkey-typed addresses form a very small percentage of the total address set. Table 2 contains a sample of normal addresses. The table contains addresses as entered by the registered users. In the table, in order to maintain privacy of the users, we replaced specific dwelling identity with x's. It can be noticed from the data that a number of challenges exist in processing these addresses. The details that could have been provided in subsequent stages of address entry, such as city name, and ZIP code would form part of the main address. It should further be noted that some city names are inadvertently spelled in different ways. Some examples are *BANAGALORE* in place of *BANGALORE* and *Bhubaneswar* in place of *Bhubaneshwar*. The preprocessing models also need to account for some city names that are renamed officially such as {*Benguluru, Bangalore*} or {*Gurgaon, Gurgram*}. We observe presence of many variants of these words when we study a large set of customer entered addresses.

Table 1: Sample Spell Variants identified through clustering

Spell variants of a word *apartment* as entered by users
apartment apartmenet apartmanet apartmenst
apartmenrt apartmennt apartment appratment
aparatment appratmant aparment apartent
apprment apartement apartament apartnment
apprtement apartmment aparttment apartrment
apartememt apaprtment aparrtment apaertment
apartmt apartme apartmernt apartmeant
apartmemt apartmebt apatrtments apaternments
apasrtments apartmensts apartmenmts aparntments

For example, in a dataset of about 10000 addresses of Bangalore, we notice about 18 different spell variations of an area name "koramangala", 61 variants of "Bannerghatta Road", 263 variants of the word "apartments". The variants are obtained through clustering the addresses [2]. Table 1 contains a sample of spell variants of the word, "apartments". They occur possibly either due to an oversight while entering and due to varying literacy levels. In the preprocessing step, these variants are replaced by a representative term in order to reduce the vocabulary.

Table 3 contains monkey-typed, and potentially fraudulent addresses. It can be noted that they contain many combinations of characters. Interestingly, some intelligent fraud customers, leave spaces while entering monkey typed addresses, to make them appear as normal addresses which typically contain multiple words. The study of such addresses provide interesting insights. It is noticed that such customers also write random characters with a mix of upper and lower case characters. Thus the proposed solution should overcome the issues of inter-word spaces, and mixed case letters. Further, in such randomly typed letters, either vowels are extremely rare or completely dominate and remain repetitive.

Figure 1 contains a word-cloud of a sample of such addresses after splitting them into 4-character long strings. We can observe from the figure that frequently used substrings predominantly match with successive letters on *qwerty* keyboard although there exist other combinations too.

We will further discuss challenges in the data and preprocessing methods that are necessary for working with the proposed approach in the following section.

3.2 Address Data Preprocessing

Both the classes of addresses of normal and monkey-typed have many specific observations. Normal address data have the following challenges.

(1) Control and special characters, that are inadvertently introduced during data entry, storage and retrieval.
(2) Upper and lower case letters
(3) Many spell variants of a given word
(4) Missing spaces, such as, *IndustrialLayout* in place of Industrial Layout

Table 2: A Sample of Normal Addresses

Sl. No.	Address
1	x-2xx, Beta 2 Greater Noida, Delhi
2	x-x, POLO ROAD ,OLD GUPTA COLONY , OPPOSITE SHANI MANDIR, DELHI
3	Plot no.-xxx/2xxxx, Jagamohan Nagar, PO-Khandagiri, Near ITER College, Bhubaneswar
4	No xxx,1st stage,4th cross, behind LVK Garments, Opp SRS Water tank, PEENYA, BANAGALORE
5	XXX/X, KENDRIYA VIHAR, SEC-11, KHARGHAR,NAVI MUMBAI

Table 3: A Sample of Monkey-typed Addresses

Sl. No.	Address
1	OEGVOQCS
2	ddfkddd
3	afadfsf
4	gdfgtdf
5	fjrjnvhejvnjdjdfogjfn vmfjgfnl

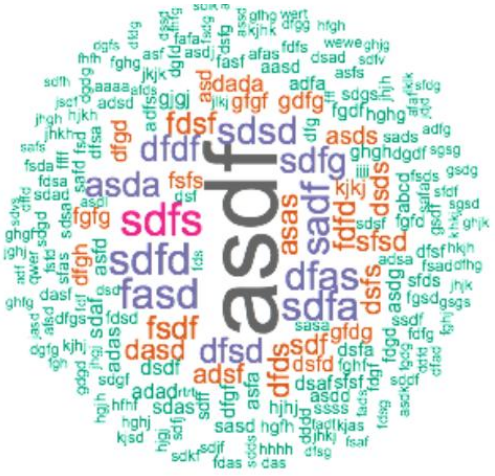

Figure 1: Word Cloud of Monkey-typed Addresses. The addresses are split into 4-character long strings. The word cloud corresponds to them.

(5) Undesirable spaces such as, *Industrial Lay out* in place of *Industrial Layout*
(6) Terms that do not help classification[2] with the proposed solution framework, such as, ZIP codes, city names, time stamp, direction to reach address, landmarks, etc.
(7) Repeated terms such as those which are accidentally typed twice while entering the addresses
(8) Lack of structure where address-words appear at different locations by different users while writing an address of a location. For example, *3rd Stage, BTM Layout* is also written as *BTM Layout, 3rd Stage*.

The monkey-typed addresses data have the following challenges.

(1) Variable length strings with maximum number reaching nearly 100 characters.
(2) Many of them have repeated substrings such as *asdf**asdf**gij* etc.
(3) Some addresses are provided as multiple monkey-typed strings so as to mimic a normal address.
(4) Combination of upper and lower case characters

The above discussion presents a list of all the challenges present in addresses. For the problem under consideration, some challenges like accidental merging or separation of words is not important since we ultimately remove the spaces in the address while generating features as we present in the following discussion.

We take following **preprocessing** or **data preparation** steps while generating features.

(1) Removing control and special characters, such as, $, ^,%, *,/,
(2) Reducing all data to lower case
(3) Generating equivalent set of words with the help of clustering of empirical data [2]
(4) After incorporating these changes, identifying unique addresses among the datasets
(5) Further steps for **Approach-1**
    - Remove spaces between address words to convert it into a **single string without spaces**
    - Split the strings into k-character substrings to form features, for k=4 and 8.
    - Blank spaces remain in the last string when the address string length is not divisible by substring length 'k'. Replace those blanks that remain after "n modulo k" split, with *'s, where n and k are lengths of full string and substring respectively.
(6) Further steps for **Approach-2**
    - Identify vowels.
    - Place '1' for vowel and '0' for the rest. For example, for the word *anjaneyacomplex*, highlighting vowels, it forms the word, ***a**nj**a**ne**ya**c**o**mpl**e**x*. The corresponding features are 100101010100010.
    - Append 0's to arrive at common pattern length, as discussed earlier.

Table 4 illustrates processing of the addresses for Approach-1. It contains addresses in three stages such as (a) raw form as entered by customers, (b) preprocessed and (c) feature form. We provide illustrations for both 8-character and 4-character long features. Different feature subsequence lengths are studied through experiments and reported in Section 4. In the table, we consider two normally written addresses and three monkey-typed addresses. It is interesting to note the variety of letter combinations and word spaces that fraudsters use in writing monkey-typed address. The '*'s at the end of a word indicate blanks that are necessary to complete 8-character bound. It should however be noted here that as discussed in Section 3.2, repeat substrings are removed.

Table 5 contains features for the preprocessed addresses for Approach-2. The preprocessed addresses are taken from Table 4. We consider each of these binary digits as features.

Table 4: Addresses and Features for Approach-1

Preprocess Stage	Sample Data
Raw Addresses	ADDRESS-ID Anjaneya Temple, om Shanthitemple road,Hegnahalli cross, SunkadaKatte
	ADDRESS-ID KR Layout, JP Nagar, 4 Phase, Kalayana Magnum Techpark
	ADDRESS-ID hshsshshshsdfhdhsh
	ADDRESS-ID scbmdbsdvgjfsgk
	ADDRESS-ID GTYSF, KJHJUH, KJ, KERALA, DJUHGJDHI,IDJIDJID, KGJFKHKDF, IJKJKL, FIJKJFGH, K IJGFKHJ, DFKL, FI, FKIJKLDFL, FI
Preprocessed Addresses	anjaneyatempleomshanthitempleroad hegnahallicrosssunkadakatte
	krlayoutjpnagar4phasekalayanamagnum techpark
	hshshshsdfhdhsh
	scbmdbsdvgjfsgk
	gtysfkjhjuhkjkeraladjuhgjdhiidjidjidkgj fkjhkdfijkjklfijkjfghkijgfkhjdfklfifkijkldflfi
Features (8-char long)	anjaneya templeom shanthit empleroa dhegnaha llicross sunkadak atte****
	krlayout jpnagar4 tphaseba ngalaore
	hshshshs jshshs**
	scbmdbsd vgjfsgk*
	gtysfkjh juhkjker aladjuhg jdhiidji dkgjfkjh kdfijkjk lfijkjfg hkijgfkh jdfklfif kijkldfl fi******
Features (4-char long)	anja neya temp leom shan thit empl eroa dheg naha llic ross sunk adak atte
	krla yout jpna gar4 tpha seba ngal aore
	hshs hshs jshs hs**
	scbm dbsd vgjf sgk*
	gtys fkjh juhk jker alad juhg jdhi idji dkgj fkjh kdfi jkjk lfij kjfg hkij gfkh jdfk lfif kijk ldfl fi**

3.3 Experiments with Language model

Initial approach to solve this problem is by using statistical language models, where a probability is assigned to a character sequence, $\{w_1, w_2, \ldots, w_n\}$. It is expected that the scores for monkey typed addresses such as **gtysfkjhjuhkjkeraladjuhgjdhiidjidjidkgj** will be distinctly different from that of a normal address. We conducted experiments with *kylm language modeling toolkit* [13] and generated scores for all the addresses. We notice that the scores overlap. We experimented with different values of n. Table 6 contains scores for a sample set of addresses for n=3 and 7. In the table, we present 2 examples each of normal and monkey typed addresses for n=3 and n=7. For n=3, it can observed that score for the monkey typed address **kjnlknmlmklnmklnm** is closer to normal address placed in sl no.1. The score for the second monkey typed address falls within the range of normal addresses, viz., {0.0091,0.042}. For n=7, the range for monkey typed and normal addresses are {0.0104, 0.020} and {0.002,0.28} which overlap. This motivated us to look at alternate approaches to solve the problem.

Table 5: Preprocessed Addresses and Features for Approach-2

Address sample and corresponding Features	Sample Data
Address-1	anjaneyatempleomshanthitempleroad hegnahallicrosssunkadakatte
Features	10010101010001000010000100000110 01001010010010000010101001
Address-2	krlayoutjpnagar4phasekalayanamagnum techpark
Features	00010110000101000010101010101010010 01000100
Address-3	hshshshsdfhdhsh
Features	000000000000000
Address-4	scbmdbsdvgjfsgk
Features	000000000000000
Address-5	gtysfkjhjuhkjkeraladjuhgjdhiidjid jidkgj
Features	00000000010000101010010000110010 010000
Address-6	fkjhkdfijkjklfijkjfghkijgfkhj dfklfifkijkldflfi
Features	00000001000000100000001000000 00000100100000001

Table 6: n-gram scores for Monkey typed and Normal Addresses using Language Model

Sl No.	Address	Score($*10^{-3}$)	
		n=3	n=7
1	6crosswilliamtown	9.4439	11.3258
2	1floorlloydvillalloyd roadcookertown	10.7745	15.3010
3	kjnlknmlmklnmklnm	9.4387	19.4918
4	mgrdtimesol	12.6973	13.3279

4 EXPERIMENTATION AND RESULTS

Monkey typed addresses occur very rarely in a given dataset. But they are present and need to be located through a model. The ratio of normal addresses to monkey typed addresses leads to severe class imbalance. Here we consider equal number of patterns for each of the classes to demonstrate working of our proposed methods. We consider 30,000 addresses with nearly equal number of normal and monkey-typed addresses. We present case studies and their corresponding datasets in Table 7. For cases 1-3, we consider $\frac{2}{3}$:$\frac{1}{3}$ division between training and test datasets. As a first step, the data is randomly shuffled. Each time an exclusive set of training dataset of size $\frac{2}{3}$, is chosen and the remaining data is considered as test data, thus resulting in three distinct cases. For cases 4-5, the training and test data are considered as $\frac{1}{2}$:$\frac{1}{2}$.

Table 7: Case Studies and Datasets

Case No	Training dataset	Test dataset
1	1-10000	10001-15000
2	5001-15000	1-5000
3	1-5000, 10001-15000	5001-10000
4	1-7500	7501-15000
5	7501-15000	1-7500
6	1-10000	10001-15000
7	5001-15000	1-5000
8	1-5000, 10001-15000	5001-10000
9	1-7500	7501-15000
10	7501-15000	1-7500

Table 7 contains serial number of case studies and corresponding datasets. The pattern numbers presented in Column 2 and 3 of the table correspond to training and test datasets of each of the classes. For example, Case-3 indicates that from randomly shuffled training data corresponding to class-1 and class-2, {1 to 5000} and {10001 to 15000} patterns each are considered for training and the rest as test data. The training data and their corresponding labels are considered accordingly.

The data is subjected to 2-class classification using linear SVM and Naive Bayes classifier. The performance is comparable. For the sake of brevity, we present the results of the exercises using linear SVM alone for both the methods. Table 8 contains results corresponding to Approach-1. The precision, recall and F-score numbers are presented in columns 2,3 and 4 for the cases considered in Table 7.

Experiments are carried out with different character lengths for the features such as 4,8, and 16. We observe that the performance is better for 4 and 8. Further, the results with 4-character long features show better performance as evident from the table.

Performance of 4-character long features perform nearly the same for both the data splits of $\frac{2}{3}$:$\frac{1}{3}$ and $\frac{1}{2}$:$\frac{1}{2}$. The best and average classification accuracy of **98%** is obtained for features of length 4 and training-test data split of $\frac{1}{2}$:$\frac{1}{2}$.

Table 9 contains results with Approach-2 for the same cases as listed in Table 7. The performance of all the five cases is nearly the same. Approach-1 shows superior performance to Approach-2, in general.

The true positive rate and false negative rates for Case-4 of Table 7 for different proportion of monkey-typed and normal addresses are provided in Table 10. This is provided in order to indicate performance under varying natural rates of monkey typed addresses ranging from 5% to 75%.

Although Approach-1 outperforms the other approach, deploying the solution across the country requires training of different combination of k-char groups, since the language spoken, names of locality, and ethnic words are distinct in different regions. Approach-2 is immune to basic nuances of words and focus on pronounceability of words. Thus, it is a more generic pattern representation.

Table 8: Experimental Results for Approach-1

Case No	Precision(%)	Recall (%)	F-Score
8-Character long features			
1	97.75	91.30	94.42
2	97.81	90.94	94.25
3	97.39	91.00	94.09
4	97.84	90.61	94.09
5	97.86	90.33	93.95
4-Character long features			
6	97.98	97.86	97.92
7	98.35	97.74	98.04
8	98.19	97.46	97.82
9	**98.58**	**97.72**	**98.05**
10	**98.35**	**97.68**	**98.01**

Table 9: Experimental Results for Approach-2

Case No	Precision(%)	Recall (%)	F-Score
1	96.81	96.81	96.81
2	96.76	96.75	96.75
3	96.81	96.81	96.81
4	96.81	96.80	96.80
5	96.75	96.72	96.72

Percentage of monkey typed addresses in training data	FPR	FNR
5	0.002	0.655
10	0.002	0.4968
20	0.0042	0.0424
50	0.0062	0.0176
75	0.007	0.011

Table 10: FPR and FNR for different proportion of monkey typed addresses in the training data

The classification time per pattern is 38 microsec on Intel i3, 2.6Ghz, 4-core, 8GB RAM machine. This is well within the single digit millisec latency requirement of the operational system.

It is interesting to see how some failures have happened in the classification. We look at an example of a normal address that is classified as monkey typed. The address *'microwave lab Dept: ECEE, C04, IISC'*, when split into 4-char features, amounts to *'micr owav elab dept ecee c04i isc*'*. Such combination of characters, such as 'elab', 'isc*', 'owav', etc., occur in monkey typed addresses as well. This is the possible reason for its misclassification. Consider the address which is monkey typed but classified as normal address, viz., *'figb irds igjr educ tion sjfk kuit ebgo b***'*. It has mix of monkey-typed 4-char features, such as, 'igjr','sjfk', and 'edgo' and few features that occur in normal addresses, such as, 'educ', and 'tion'. This possibly resulted in its classification as a normal address.

An important aspect of the deployment is dealing with class imbalance. We provide a discussion in Section 5 on various approaches to tackle imbalance.

5 CLASS IMBALANCE AND MITIGATION

Like in many fraud or spam classification problems, monkey-typed addresses form a very small percentage of combined class dataset leading to severe class imbalance [4, 9, 11, 14]. The problem formulation, feature selection, approaches to solve the problem and the experimental results indicate that the chosen approaches lead to effective solution to the problem. As a next step, we need to make it operational under class imbalance too. The costs of misclassification [8, 12] of a monkey typed address and normal address are not same. We discuss class imbalance problem and some important approaches to mitigate this problem.

Consider a 2-class classification scenario, where the number of patterns belonging to one class far exceeds those belonging to second class. Such data is termed imbalanced between the classes. Finding an effective classification model in such a scenario is challenging, since many classification models favour the majority class. In many practical problems, the minority class represents fraudulent patterns. Misclassification of fraudulent patterns as genuine ones is expensive. Some conventional approaches to solve the problem are under-sampling majority class till balance is obtained, over-sampling the minority class in similar manner and choice of unequal classification costs. In case of under-sampling, when representative patterns intra-class variations are eliminated, the classification accuracy suffers. In case of over-sampling of existing patterns of minority class, it is not guaranteed to arrive at inter-class discriminative boundary. In the last 15 years, a number of intelligent sampling methods, boosting and combination of sampling and boosting, are studied to effectively overcome these issues [4, 14]. Synthetic Minority Over-sampling Technique (SMOTE) [4] marks intelligent sampling approach where a combination of over-sampling and under-sampling is resorted to, to demonstrate its superior performance using area under Receiver Operating Characterstic (ROC) curve. Subsequent extensions of SMOTE and many variants of boosting are outlined in [14], where RusBoost is shown to outperform the rest of the approaches.

Adapting the above discussed approaches would help mitigating class imbalance problem for monkey-typed address classification.

6 SUMMARY AND FURTHER WORK

We consider a practical problem that affects e-commerce companies where potentially fraudulent addresses containing a single or multiple strings of randomly typed addresses occur. We refer to them as *monkey-typed addresses*. When such addresses remain undetected, they would amount to loss of revenue and cause strain on resources at various stages of processing, starting from order checkout to logistics. The problem is challenging. Through an initial study, we observed that conventional language modeling approaches do not generalise well. We present two novel solutions (a) by forming fixed-length features and (b) by exploiting inter-vowel distance, after subjecting the addresses to a number of preprocessing steps. We present the data challenges, preprocessing steps, proposed approaches and feature generation. We carry out experiments for feature lengths of 4 and 8 for Approach-1 and features marking vowels for Approach-2. The performance is similar in both the approaches.

Further work includes reducing the number of features through optimal feature selection approaches, experimenting with models that could work across different geographical regions within a large country and adapting to class imbalance.

ACKNOWLEDGMENTS

The authors would like to thank Aditya Rachakonda for his helpful comments.

REFERENCES

[1] Leman Akoglu, Rishi Chandy, and Christos Faloutsos. 2013. Opinion Fraud Detection in Online Reviews by Network Effects. In *ICWSM*. 2–11.
[2] T. Ravindra Babu, A. Chatterjee, S. Khandeparker, A.V. Subhash, and S. Gupta. 2015. Geographical address classification without using geolocation coordinates. In *Proceedings of the 9th Workshop on Geographic Information Retrieval*. ACM.
[3] S. Bhattacharyya, Jha, K. Tharakunnel, and J.C. Westland. 2011. Data mining for credit card fraud: A comparative study. *Decision Support Systems* 50, 3 (2011), 602–613.
[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.
[5] R.A. Cole, Y. Yan, B. Mak, M. Fanty, and T. Bailey. 1996. The contribution of consonants versus vowels to word recognition in fluent speech. In *Proccedings of IEEE Conf. on Acoustics, Speech, and Signal Processing, 1996. ICASSP-96*, Vol. 2. 853–856.
[6] Sharon Curry. 2000. An inside look at e-commerce fraud - Prevention and solutions. *http://www.scambusters.org/ecommercefraud.pdf* (2000).
[7] C. A. Davis-Jr. and F. T. Fonseca. 2007. Assessing the Certainty of Locations Produced by an Address Geocoding System. *Geoinformatica* 11 (2007), 103–129.
[8] C. Elkan. 2001. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd.
[9] H. He and E. A. Garcia. 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering* 21, 9 (2009), 1263–1284.
[10] J. Hollmen. 2000. *User profiling and classification for fraud detection in mobile communications networks*. Helsinki University of Technology.
[11] N. Japkowicz. 2000. The class imbalance problem: Significance and strategies. In *In Proc. of the International Conf. on Artificial Intelligence*.
[12] M. Maloof. 2003. Learning when data sets are imbalanced and when costs are unequal and unknown. In *ICML-2003 workshop on learning from imbalanced data sets II*, Vol. 2.
[13] Graham Neubig. [n. d.]. Kylm - The kyoto Language Modeling Toolkit. ([n. d.]). http://www.phontron.com/kylm/
[14] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, and Amri Napolitano. 2010. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 40, 1 (2010), 185–197.
[15] N. Shah, Alex Beutel, Bryan Hooi, Leman Akoglu, Stephan Gunnemann, Disha Makhija, Mohit Kumar, , and Christos Faloutsos. 2016. EdgeCentric: Anomaly Detection in Edge-Attributed Networks. In *In Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*. 327–334.
[16] Y. Zhang, J. Bian, and W. Zhu. 2013. Trust fraud: A crucial challenge for China's e-commerce market. *Electronic Commerce Research and Applications* 12, 5 (2013), 299–308.