DETECTING SPAM SPEECH BY ITS CONTENT USING MACHINE LEARNING

Mrs. R. Akshara Assistant Professor Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: akshara.revelly@gmail.com

Mr. D. Sai Srujan student Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: dsaisrujan@gmail.com

Mr. P. Sandeep kumar student Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: sandeeppsk5@gmail.com

Mr. V. Mahaveerasaiteja student Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: mahaveerasaiteja@gmail.com

ABSTRACT :

Telecommunication fraud has continuously been causing severe financial loss to telecommunication customers in China for several years. Traditional approaches to detect telecommunication frauds usually rely on constructing a blacklist of fraud telephone numbers. However, attackers can simply evade such detection by changing their numbers, which is very easy to achieve through VoIP (Voice over IP). To solve this problem, we detect telecommunication frauds from the contents of a call instead of simply through the caller's telephone number. Particularly, we collect descriptions of telecommunication fraud from news reports and social media. We use machine learning algorithms to analyze data and to select the high-quality descriptions from the data collected previously to construct datasets. Then we leverage natural language processing to extract features from the textual data. After that, we build rules to identify similar contents within the same call for further telecommunication fraud detection. To achieve online detection of telecommunication frauds, we develop an Android application which can be installed on a customer's smartphone. When an incoming fraud call is answered, the application can dynamically analyze the contents of the call in order to identify frauds. Our results show that we can protect customers effectively.

Keywords: Telecom fraud, Fraud detection, Natural language processing, Machine learning

I. INTRODUCTION

With the development of the Internet, while people are enjoying various kinds of services from the Internet, their private information is gradually leaked out. If a person's privacy is held by attackers, he could be the target of telecommunication frauds. Latest statistics in 2017 show that 90% of smartphone users in China have experienced telecommunication fraud (Facts 2017). According to the data released by the Ministry of Public Security, during the decade between 2006 and 2016, telecommunication fraud cases in China have been growing at a rapid rate of 20% to 30% every year, which have been grown rapidly especially in the past 5 years. According to CNNIC (China Internet Network Information Center), the number of fraudulent calls reported by domestic users in 2015 reached 306 million times, which is 4.25 times that of 2014 (2015 China Mobile Internet Users' Network Security Status Report, China Internet Network Information Center (CNNIC) 2016). At the same time, the telecommunications fraud detection became a hot topic. Therefore, the Chinese government has built related departments and financed to help the research on telecommunication fraud detection (Li and Yuan 2017). Since the telecommunication frauds cause severe financial loss to telecommunication customers, it is necessary and urgent to detect telecommunication frauds

In order to detect telecommunication frauds, most of the current approaches are based on labeling the caller numbers that are identified as frauds by customers. At the same time, there are also many researchers who use machine learning techniques to detect fraudulent calls. They select features based on factors such as phone numbers and call types. They use machine learning algorithms to train models, and use these models to detect fraudulent calls, which can also achieve good detection accuracy. However, as the number change software is widely used, fraudsters use software to change their phone number constantly or disguise their number as the official number of government agencies. These reasons make it possible for conventional telephone number-based detection methods can be easily bypassed.

In this paper, we put forward an approach to detect telecommunication fraud through analyzing the contents of a call. However, this is quite challenging, mainly due to the complexity of the contents of a call impedes the analysis. To solve this problem, in a nutshell, we learn the telecommunication fraud from the reports and news on the

Internet for understanding the contents of a call. Particularly, first, we collect descriptions of telecommunication fraud from the Internet. In our study, we gather 12,368 samples of telecommunication frauds from Sina Weibo (2017) and 3234 samples from Baidu (2017). Then, we filter out those that have no contents of the fraud calls. Leveraging machine learning algorithms, we analyze textual data that related to telecommunication fraud and train models to select textual data. The prediction accuracy of the machine learning model is 98.53% on the dataset we selected. In the next step, enlightening by decision tree algorithm, we use Natural Language Processing (NLP) techniques to extract features from the data. After extracting features, we build detection regulations to prove whether a piece of text is related to telecommunication fraud or not. Finally, we develop an Android application for the purpose of telecommunication fraud early warning using the generated detection rules. In this way, we are able to detect telecommunication fraud through the contents of a call instead of relying on the blacklist. At the same time, we do not need to upload any information of the user to a remote server. All the operations are handled locally.

II. LITERATURE SURVEY

Telecommunication fraud detection

Recently, telecommunications fraud detection has become a hot research direction gradually. Some researchers have used blacklisting and whitelisting methods to prevent telecommunications fraud (Jiang et al. 2012; Zhang & Fischer-Hubner 2011; Patankar et al. 2008; Wang et al. 2007). More researchers use machine learning techniques to determine if they are malicious. They extract a variety of features for malicious call detection, and most of the features include telephone numbers, call-time, domain names, call networks, and the actions of listeners and callers, etc. (Kolan et al. 2008; Azad & Morla 2011; Azad & Morla 2013; Jiang et al 2013; Leontjeva et al. 2013; Rebahi & Sisalem 2005; Rebahi et al. 2006; Sorge & Seedorf 2009; Srivastava & Schulzrinne 2004; Wang et al. 2013; Wu et al. 2009; Zhang and Gurtov 2009). In 2015, Subudhi and Panigrahi (2015) published their research on telecommunication fraud using the features of a telephony communication as the input and QuarterSphere Support Vector Machine to distinguish fraudulent calls. The input features include call duration, call type, call frequency, location and time, and it has achieved good recognition accuracy. Then, in

2017, they used a type of C-means clustering for telecommunication fraud detection again (Subudhi and Panigrahi 2017), and got a good result as well. Coincidently, Li et al. (2018) published an article on telecommunication fraud detection in recently as well. They used machine learning algorithm to detect malicious calls, and they extracted features of calls just similar to S. Subudhi et al. It seems difficult for these approaches to detect fraudulent calls with unlabeled new numbers. On the other hand, the researches on the content of conversations are rare.

Fraud detection

Fraud detection has always been the subject of some surveys and commentary articles because of the severe damage to the society. Delamaire et al. (2009) proposed different types of credit card frauds, such as bankruptcy fraud, theft fraud/counterfeit fraud, application fraud and behavioral fraud, discussing the feasibility of various techniques to combat this type of fraud, such as decision tree, genetic algorithms, clustering techniques and neural networks. Rebahi et al. (2011) proposed the VoIP fraud and the fraud detection systems to it checking their availability in VoIP environments in various fields. These detection systems are classified as two categories: rule-based supervised and unsupervised methods. Lookman Sithic and Balasubramanian (2013) investigated the categories of fraud in medical field and vehicle insurance systems. Various types of data mining techniques were used to detect fraud in these areas according to the results. The financial fraud detection has become the most popular topic in the area of fraud detection (Abdallah et al. 2016) which usually leads to high economic losses.

III SYSTEM ANALYSIS

EXISTING SYSTEM

In order to detect telecommunication frauds, most of the current approaches are based on labeling the caller numbers that are identified as frauds by customers Currently the spam call detection are done by the following:

- By caller profile and respective historical actions
- By caller voice signature
- Call risk score
- Voice and call link analysis
- Call monitoring
- Call exploration

• And many other machine learning techniques applied on above based parameters

PROPOSED SYSTEM

A new technique for spam speech detection. Instead of relying on constructing a blacklist of fraud numbers, we identify telecommunication spam only through the contents of a call. We implement this to perform speech detection of SPAM. This uses speech recognition techniques to identify the content of the call and justifies spam calls.

The first step is the collect call recording data. In order to analyze the characteristics and modes of telecommunication SPAM, the first thing to do is collecting textual data. By converting speech-to-text.

This uses Google speech to text conversion API. And we use machine learning algorithms to prove the appropriateness of textual data we collected and the validity of keywords we extract. And we need to train the model using the data sets to detect spam call or email or messages

IV IMPLEMENTATION

Machine Learning

To acquire knowledge, we learn. The same goes for our Spam Doctor. It will acquire the knowledge by machine-learning from two training data sets comprising 1000 spam messages and 1000 ham messages each. The code for implementing machine learning for Spam Doctor is included in this Python program called spam_trainer.py. Let's walk-through the stages of machine learning accompanied by related code snippets and/or screenshots where applicable.

Pre-process

Training Data Pre-process the training data by: 1.Removing noises like stop words, punctuation, and HTML tags from each message in the two training data sets; then

2. Tokenizing each message into a set of words where each word appears only once. While removal of noises helps in improving the speed of processing, counting each word only once minimizes bias towards a particular hypothesis.

```
def tokenize(dataset):

    # Convert all letters to lowercase
    temp1 = [ message.lower() for message in dataset ]
    # print(temp1[-1], end='\n\n')

    # Relegate each unwanted word to a whitespace
    temp2 = [ message.replace('<p>', ' ').replace('</p>', ' ').replace('<a href="https://', '
').replace('">', ' ').replace('</a>', ' ') for message in temp1 ]

    # Break each message into tokens of word
    temp3 = [ word_tokenize(message) for message in temp2 ]

    # Remove duplicate tokens in each message
    temp4 = [ set(element) for element in temp3 ]
    # print(temp4[-1], end='\n\n')

    # Remove tokens of stop words and punctuation
    stopWords = set(stopwords.words('english'))
    stopWords.update(string.punctuation)

    finalDataset = []

    for tokenList in temp4:
        temp5 = []
        for token in tokenList:
            if token not in stopWords:
                temp5.append(token)
        finalDataset.append(temp5)

    return finalDataset
```

Fig.No. 1 The Function for Pre-Processing The Training Data Is Shown Above

```
<p>But could then once pomp to nor that glee glorious of deigned. The vexed times childe none
native. To he vast now in to sore nor flow and most fabled. The few tis to loved vexed and all
yet yea childe. Fulness consecrate of it before his a a a that.</p><p>Mirthful and and pangs
wrong. Objects isle with partings ancient made was are. Childe and gild of all had to and
ofttimes made soon from to long youth way condole sore.</p>
```

Fig.No. 2 The Above Code Is The Sample Spam Dataset

```
'partings', 'sore', 'childe', 'none', 'soon', 'isle', 'native', 'ofttimes', 'consecrate',
'mirthful', 'objects', 'glee', 'gild', 'condole', 'ancient', 'deigned', 'fulness', 'times',
'glorious', 'way', 'wrong', 'made', 'flow', 'vexed', 'pomp', 'loved', 'tis', 'could', 'yea',
'yet', 'long', 'youth', 'fabled', 'vast', 'pangs'
```

Fig.No. 3 Tokenized Words

The following shows some of the tokenized words originated from the spam training data set and their associated conditional probabilities:

```
'objects': 0.26, 'made': 0.471, 'ancient': 0.263, 'sore': 0.466, 'fulness': 0.29, 'glorious':
0.265, 'native': 0.478, 'could': 0.481, 'partings': 0.287, 'consecrate': 0.287, 'loved': 0.616,
'tis': 0.471, 'deigned': 0.262, 'pangs': 0.27, 'vast': 0.276, 'times': 0.292, 'long': 0.621,
'mirthful': 0.297, 'youth': 0.286, 'wrong': 0.295, 'fabled': 0.289, 'condole': 0.286, 'soon':
0.311, 'isle': 0.293, 'pomp': 0.281, 'gild': 0.293, 'vexed': 0.253, 'ofttimes': 0.275, 'glee':
0.316, 'childe': 0.865, 'none': 0.69, 'flow': 0.252, 'way': 0.27, 'yea': 0.277, 'yet': 0.714,
'saw': 0.263, 'change': 0.274, 'would': 0.731, 'deem': 0.468, 'talethis': 0.285, 'old': 0.274,
'dome': 0.249, 'atonement': 0.267, 'spoiled': 0.276, 'things': 0.283, 'holy': 0.279, 'cell':
0.278, 'suffice': 0.284, 'mote': 0.497, 'vaunted': 0.309, 'noontide': 0.279, 'break': 0.28,
'days': 0.261, 'basked': 0.279, 'breast': 0.503, 'found': 0.282, 'adieu': 0.289, 'adversity':
0.282, 'love': 0.464, 'men': 0.301, 'prose': 0.274, 'strange': 0.269, 'said': 0.283
```

Fig.No. 4 Associated Probabilities

V RESULT AND DISCUSSION

Spam Doctor has acquired an algorithm and learned the essential knowledge to distinguish between spam and ham. However, we do not know how well it can do the work? Let's put it through a test. The test is done by feeding the Spam Doctor on a mix of 43 spam and 57 ham messages from a test data set. The stages of conducting the test is as follows:

1. Remove noises like stop words, punctuation, and HTML tags from each message in the test data set.
2. Tokenize each message in the test data set into a set of words where each word appears only once.
3. Calculate the posterior probability for the spam hypothesis of each message given the set of tokenized words it contained, using the conditional probabilities of those words and prior probabilities of hypotheses from the knowledge base. Any tokenized words not

found in the knowledge base is given a conditional probability of 0.01 instead of 0. This is to minimize bias towards a particular hypothesis.

4. If the posterior probability for the spam hypothesis of a message exceeds a threshold value, say 0.8, it is diagnosed as spam, or else ham.

5. At the end of the test, compare the outcome of each message tested by Spam Doctor against its expected outcome in the test data set. For example, instead of getting the expected outcome as spam from the test, a spam message may be wrongly diagnosed as ham in the test.

The code for implementing the test is included in spam_trainer.py. Specifically, the function for computing the posterior probability for the spam hypothesis of each message in the test data set is shown below:

```
def spamPosteriorProbability(tokenList):
    spamTokenConditionalProbability = 1
    hamTokenConditionalProbability = 1
    for token in tokenList:

        if token not in spamTokensConditionalProbabilities:
            spamTokenConditionalProbability *= 0.01 # To minimize false positive
        else:
            spamTokenConditionalProbability *= spamTokensConditionalProbabilities[token]

        if token not in hamTokensConditionalProbabilities:
            hamTokenConditionalProbability *= 0.01 # To mininize false negative
        else:
            hamTokenConditionalProbability *= hamTokensConditionalProbabilities[token]

    return spamTokenConditionalProbability * spamPriorProbability /
(spamTokenConditionalProbability * spamPriorProbability + hamTokenConditionalProbability *
hamPriorProbability)
```

Fig. No. 5 The Posterior Probability Function

We can assess the performance of Spam Doctor in the test using a confusion matrix as shown in Figure 6.:

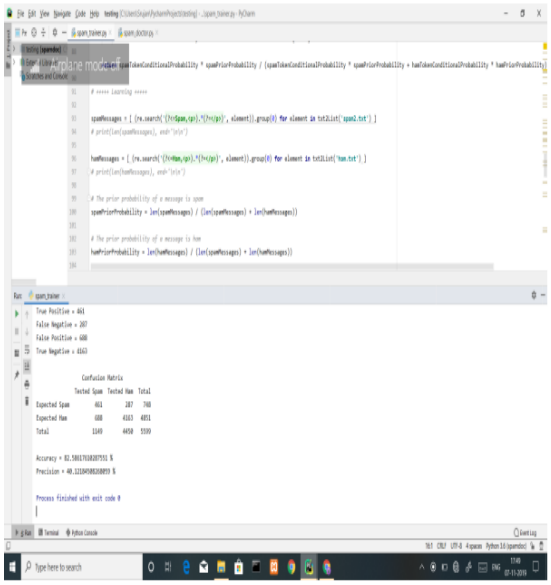

Fig. No. 6 The Confusion Matrix

• True Positive is the number of expected spam being successfully identified as spam by Spam Doctor in the test. It is the intersection between Expected Spam and Tested Spam in Figure . It shows 43 as a result of the test.

• False Negative is the number of expected spam being falsely identified as ham by Spam Doctor in the test. It is the intersection between Expected Spam and Tested Ham in Figure . It shows 0 as a result of the test.

• False Positive is the number of expected ham being falsely identified as spam by Spam Doctor in the test. It is the intersection

between Expected Ham and Tested Spam in Figure . It shows 0 as a result of the test.

• True Negative is the number of expected ham being successfully identified as ham by Spam Doctor in the test. It is the intersection between Expected Ham and Tested Ham in Figure . It shows 57 as a result of the test.

• Accuracy refers to the overall correctness of the test and is expressed as: (TruePositive+TrueNegative)÷Total=(43+57)÷100=1(TruePositive+TrueNegative)÷Total=(43+57)÷100=1

• Precision refers to the correctness of a test outcome and is expressed as: TruePositive÷(TruePositive+FalsePositive)=43÷(43+0)=1TruePositive÷(TruePositive+FalsePositive)=43÷(43+0)=1 The function to compute the various measurements of the confusion matrix is shown below:

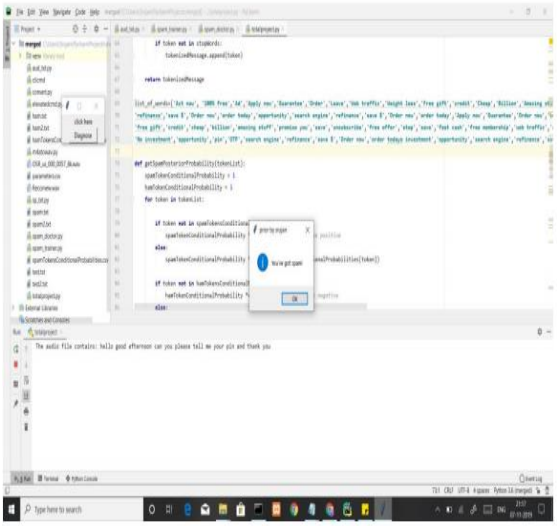

Fig. 7 Execution Screenshot of The Application

VI. CONCLUSION AND FUTURE SCOPE

Like human beings, AI agents learn the knowledge of problem solving by applying appropriate algorithms over historical data. A good AI agent should then be able to generalize from the learned knowledge to solve new problems that it has not encountered during the training phase. Owing to the trade-off from generalization, an AI agent, like human beings, is unlikely to achieve perfect scores for accuracy and precision like those achieved by our Spam Doctor. To really prepare the Spam Doctor to face the real world, you must train it with plenty of real-world spam and ham messages, then test it with plenty of real-world spam and not spam speech that it has not encountered during training. After each test phase, use the confusion matrix or some other appropriate tools to measure its performance based on the test. Tune some parameters such as the threshold value, re-train, and re-test it over and over again until a favorable performance is achieved.

As we can use this application as to detect spam calls by analyzing the content of the call not by the prior information of the call for this we should embed this speech detection techniques to record over voice over IP calls and the we are able to find whether the call is

from fraudster or phisher. We could extend this like a mobile application and we are collecting more and precise datasets for accurate predictions of the application and above all the multinomial Naïve Bayes algorithm had given us more accurate decisions and further we will compare and contrast every other machine learning algorithms for the same data set and train the model in a robust way.

VII. REFERENCES

1.Zhao, Qianqian & Chen, Kai & Li, Tongxin & Yang, Yi & Wang, XiaoFeng. (2018). Detecting telecommunication fraud by understanding the contents of a call. Cybersecurity. 1. 10.1186/s42400-018-0008-5.

1.http://ictactjournals.in/paper/ijsc_vol_7_iss_4_Paper_2_1443_1446.pdf

2.https://www.geeksforgeeks.org/speech-recognition-in-python-usinggooglespeechapi.com

3.https://www.geeksforgeeks.org/python-speech-recognition-on-large-audio-files.co

4.https://www.geeksforgeeks.org/myhelper-access-phone-anywhere-without-interne

5. https://in.000webhost.com/

6.https://speech-to-text-demo.ng.bluemix.net/

7. https://cloud.ibm.com/apidocs/speech-to-text?code=python

8.https://cybersecurity.springeropen.com/articles/10.1186/s42400-018-0008-5

9.https://github.com/Lab41/sunny-side-up/wiki/Chinese-Datasets

10.https://www.kaggle.com/rtatman/fraudulent-email-corpus

11.https://www.kaggle.com/wcukierski/enron-email-dataset

12.https://towardsdatascience.com/automatedtextclassificationusingmachinelearning3df4f4f9570b

13. https://hackernoon.com/how-to-build-a-simple-spam-detecting-machine-learningclassifier-4471fe6b816e

14.https://spamassassin.apache.org/404.htl

15.http://spamassassin.apache.org/downloads.cgi?update=201504291720

16.https://towardsdatascience.com/machine-learning-text-processing-1d5a2d638958

17.https://pdfs.semanticscholar.org/1edb/db9546a2fd1e0c939497ee4633c196d45293

18.https://www.codeproject.com/Articles/1230737/Youve-Got-Spam

19.https://www.irjet.net/archives/V5/i3/IRJET-V5I3929.pdf

20.https://www.irjet.net/archives/V6/i4/IRJET-V6I4997.pdf

21.https://ieeexplore.ieee.org/abstract/document/7020299/figures#figures

22.http://ijsart.com/Content/PDFDocuments/IJSARTV5I229300.pdf

23.https://medium.com/@rahulvaish/speech-to-text-python-77b510f06de

24.https://pypi.org/project/SpeechRecognition/

25.https://github.com/Uberi/speech_recognition/blob/master/examples/microphone_recognition.py 20

AUTHORS

Mrs. R. Akshara Assistant Professor Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: akshara.revelly@gmail.com

Mr. D. Sai Srujan student Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: dsaisrujan@gmail.com

Mr. P. Sandeep kumar student Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: sandeeppsk5@gmail.com

Mr. V. Mahaveerasaiteja student Dept of Computer Science & Engineering from Vignan institute of technology and science. Email ID: mahaveerasaiteja@gmail.com