Identifying fraud tech support calls by using LSTM, NLTK, and Word2Vec techniques

Akshat Agarwal

Received July 09, 2023
Accepted August 09, 2023
Electronic access September 30, 2023

Tech support scams have become increasingly common over the past few years with the advent of technology and the constant expansion of internet users. It is now more important than ever to fight against such scams to prevent people around the world from falling victim to such schemes. Although research has been conducted into NLP being used for spam/scam detection, there is a need for the implementation of such techniques to fight specifically against tech support scams. In this paper, the severity of the problem of tech support scam calls and the feasibility of applying NLP to combat it is discussed. Three popular methods for NLP, LSTM, NLTK, and Word2Vec, have been implemented to a custom-built dataset and their accuracies have been compared to decide which type of model would be most suitable for the problem at hand. The structures and innate workings behind all of these models have also been presented. Rationales for the misclassifications of the models have also been provided, giving more insight as to how exactly these models work and why they are suitable/not suitable for solving the problem of fraud tech support calls.

Introduction

The phrase "Tech Support Scams"[1] refers to a widely experienced issue in today's world. Scammers, over a phone call, use scare tactics to convince the receivers of their calls to employ their usually expensive technical support services to supposedly fix issues with their victims' electronic devices that, in actuality, do not exist. The victim can receive a direct phone call from scammers who identify themselves as tech firm personnel. They might even spoof the caller ID so that it shows a genuine customer service number from a reputable business. They'll probably want the victim to download software so they can access their device remotely. These skilled con artists then pass off common system alerts as errors by using remote access. Scammers may also contact the victim by tricking them into calling their scam organizations by posting phony error messages and assistance numbers on the websites the victim may visit. Additionally, they might make the victim's browser run in full-screen mode and show persistent pop-up ads, ostensibly locking it. The purpose of these fictitious trouble warnings is to frighten the victim into phoning their "technical assistance hotline."[2]

The best-case scenario is that these scammers are looking to extort their victims for small amounts of money usually in the form of non-traceable payment methods such as gift cards. The worst-case scenario is that these scammers could frequently install malware, ransomware, or other unwanted programs that can steal the victim's information or harm their data or device. Such programs could lead to more fake pop-ups prompting the victim to call the scam organization once again to fix the "problem", continuing the cycle. In some cases, the scammer convinces their victim to log into their online bank account to "rightfully" pay for their service. After the victim pays, they use inspect element (a technique to modify the values displayed on a webpage) to alter the contents of the webpage to act as if the user accidentally paid too much. Out of their "good faith", they offer to refund the extra money back to the victim. They use a fake transaction program, acting as if that is how their company usually refunds its consumers, and make the victim manually enter the refund amount. Using

Fig. 1 Example of a fake virus pop-up[1]

their quick typing abilities, they add a few zeroes to the end of the number before the victim enters the amount. They then use inspect element again to trick the victim into that they did in fact refund too much. The victim then withdraws thousands of dollars from their bank account, thinking it is the organization's money. However, they are withdrawing money from their life savings.[3] In such cases, a simple scam can lead to the loss of one's livelihood. The main targets of tech support scams are usually senior citizens as most of them are individuals who are not as familiar with how new technology works. However, millennials are also just as susceptible to falling for such scams.[4] This has been attributed to their tendency or desire to explore the internet more, leading to more opportunities for them to experience false pop-ups on various websites.

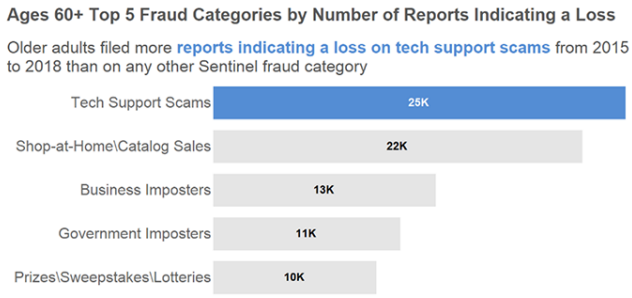

Fig. 2 Top fraud categories for adults aged 60+[5]

Although phone scams take place around the world, no country experiences as many scam calls per person as India. According to a survey conducted by Microsoft in 2021, 7 out of 10 of the Indians who participated in the survey had experienced tech support scams in the previous year. As high as 31% of them who continued with the interaction lost money[4].

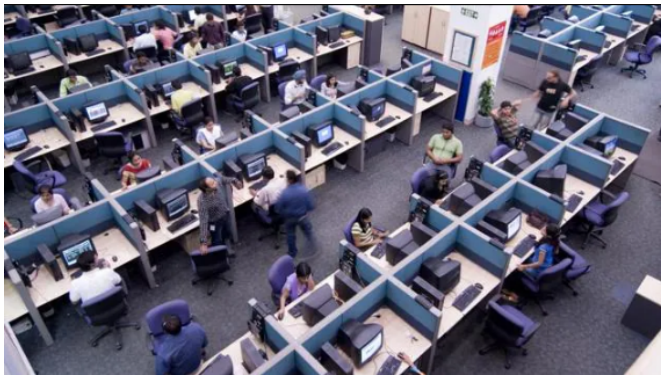

Fig. 3 A scam call center in India[6]

AI thus should be implemented in such a situation as it would provide an efficient way to screen calls that possible

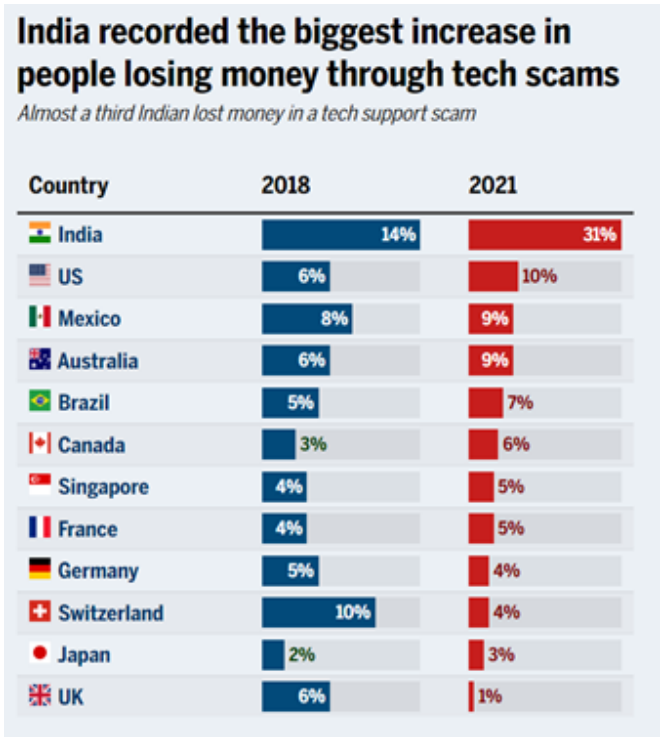

Fig. 4 Percent of people to lose money to tech support scams by country[4]

victims could have with scammers to alert them of any suspected foul play that could be occurring in real time.

Fig. 5 Usual timeline for a scam call[7]

NLP[8] refers to the branch of Artificial Intelligence that teaches computers to understand the meaning and nuances behind spoken words in the same way humans can. NLP blends statistical, machine learning, and deep learning models with computational linguistics. Using these technologies, computers are now able to process human language in the form of text or audio data and fully "understand" what is being said or written, including the speaker's or writer's intentions and sentiments. Common uses of NLP include voice-activated GPS devices, digital assistants, dictation software, and chatbots for customer service. NLP could be implemented to screen and analyze the transcripts of ongoing calls to determine whether the call will lead the user to allow a stranger to gain remote

access to their computer, reveal sensitive information, and pay someone for "fixing" a problem that did not exist in the first place. Here is some of the research that has already been conducted on the topics of scams and NLP:

A. Scams

Various literature reviews and other projects about other types of scams were helpful to use as a reference when working on my project. The first work that I looked at was a paper published by Michelle Chen and Ashish Sur, students at Berkeley[7]. The paper discusses how NLP could be used to fight against phone scammers in the US. It explores NLP techniques such as detecting authority and superiority in the scammer's speech patterns. This project focused mainly on false warrants, lawsuits, and jury duty calls. This paper provides a framework on how to build a dataset from scratch, which I used for this project. An incredibly useful paper I found was one made by Najmeh Miramirkhani, Oleksii Starov, and Nick Nikiforakis from the Department of Computer Science at Stony Brook University[9]. The paper focused on finding patterns in tech support scam calls and included a few examples of call transcripts at the end of the paper which I referenced when typing out my dataset. A paper was written by Sathvik Prasad, Elijah Bouma-Sims, Athishay Kiran Mylappan, and Bradley Reaves at North Carolina State University in August 2020[10]. The paper discusses how NLP could be used to characterize Robocalls by analyzing audio from the calls. This paper provided a framework of how I could apply NLP to tech support calls instead of Robocalls.

B. NLP

NLP has been widely researched ever since the idea of processing language using AI was introduced. It was imperative that I learn more about NLP and the various techniques associated with it before building my models. Said Sallouma, Tarek Gaber, Sunil Vadera, and Khaled Shaalan[11] published a paper at the 5th International Conference on AI in Computational Linguistics. This paper talks about the currently ongoing research projects being conducted using NLP and Machine Learning techniques. The project talks about how various algorithms and toolkits such as Doc2Vec, XGBoost, RF, etc. theoretically could be applied to detect phishing emails. Ignacio Palacio[12] discusses the difference in performance between classic and modern NLP analysis techniques when they were used to fight against misinformation spreading on social media. It references techniques such as TF-IDF, NLTK, Word2Vec, etc. Rui Zhong, Xiaocen Dong, Rongheng Lin, and Hua Zou from the Beijing University of Posts and Telecommunications[13] wrote a paper about how they created a model that used TF-IDF to detect fraudulent phone calls. They used data obtained from China itself for their project. The problem with these papers when it comes to using them as a reference to fight fraud tech support calls is that some of them do not talk specifically about tech support calls. Tech support calls contain different types of language and buzzwords than other types of fraud calls. The papers that do talk about tech support calls do not give specific details about the models used and do not explain the rationale behind the values of their accuracy. This paper includes all these details as well as provides a framework for future improvements to the project. The ideal model from this research project should be able to distinguish between a tech support scam call and a legitimate call when applied in the context of a custom-built dataset with an accuracy of at least 80% with a higher false positive rate than a false negative rate. My solution to this problem is to explore the effectiveness of 3 models built using different NLP techniques, LSTM, NLTK, and Word2Vec, identifying which model is the most efficient when applied to the problem of detecting tech support scams. This paper discusses the results of each model, explaining the rationale behind the accuracies and misclassifications of each, discusses the future work that could be implemented to the project and takes a deeper dive into the structures and workings of each model and algorithm that has been used in the project.

Results

The split of the data that I used for all models was 80% training data (19 examples) and 20% testing data (5 examples). The accuracies of each model don't tell much about how they differ in their performances. This is to be expected when using a small dataset. Thus, to see where exactly each model succeeds and fails, we need to look compare the predictions of each model with the actual labels manually. This can be done using confusion matrices.

Table 1 Accuracies of each model

Model	Accuracy Score
Simple LSTM	60%
NLTK with Random Forest	80%
Word2Vec with Random Forest	80%

A. Simple LSTM Model

By looking at the confusion matrix, we can see that the LSTM model, most of the time, classified the input given to it as a scam[2]. The matrix shows that the LSTM is unreliable when non-scam input is given to the model. This data tells us that LSTM is not ideal to use in situations where small and spe-

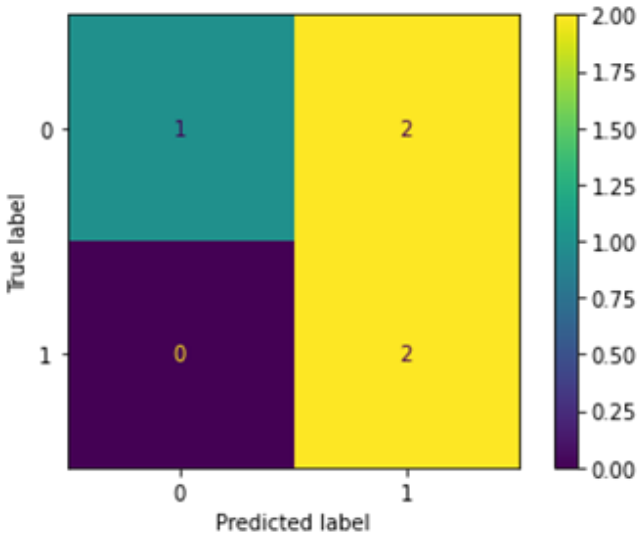

Fig. 6 LSTM Model Confusion Matrix

cific phrases completely alter the meaning and nuance of a sentence. Thus, for scam call detection, this model is not ideal.

B. NLTK with Random Forest

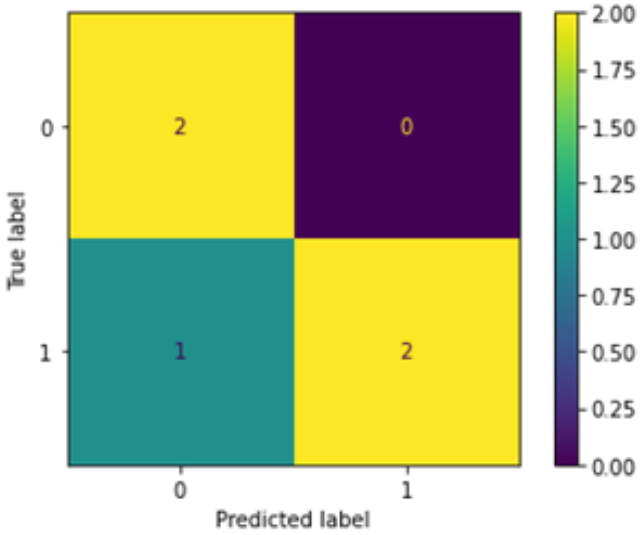

Fig. 7 NLTK Model Confusion Matrix

As we can see from the results of the NLTK model, using the list of stopwords and PorterStemmer, the problem experienced with the LSTM model is eliminated. All the filler words from the sentences are eliminated allowing the model to only focus on the main information in the sentence. From the confusion matrix, it is evident that the model excels at differentiating a normal call from a scam call. The model, however, misclassifies a scam call example as ham. To identify exactly what the model is misinterpreting, I examined the example that the model misclassified.

Scam	Thank you for calling technical support, how can I help you? Uh, good morning. Um, I think I may have a problem with my computer, because I was just looking at the internet and websites and suddenly told me that I am infected. And it asked me to call you. If you can, sir, can you just read the error message which you are getting on the screen? So it said, 'Warning, you may have been infected with a virus' and it said no, don't attempt to remove it by yourself, not that I would know how. Um, and then it asked me to call this number, and then I think the browser closed. It said, you know, 'Does not respond, do you want to close?' And I closed.

Even though this text contains a lot of buzzwords that should have been picked up by the model, it wasn't. This is an example of the shortcomings of a small dataset as well as the absence of Word2Vec embeddings. TF-IDF only allows the program to calculate the number of occurrences of a certain word in a transcript. Since a scammer might not repeat the exact words that the program identified as common in the training data, it could lead to misclassification as shown above. Word2Vec, however, builds connections between words that could be used in similar contexts. This allows a model to then identify a scam call even if the scammer does not use the exact terminology.

C. Word2Vec with Random Forest

We can see that the model performed exceptionally when it came to correctly classifying the scam examples but misclassified one of the non-scam examples as a scam. This shows the effect that the word embeddings performed by the model had a profound effect by making it easier to identify the keywords in a scam call. Using the same method from the previous section, I identified the misclassified example. Let's take a look at that example:

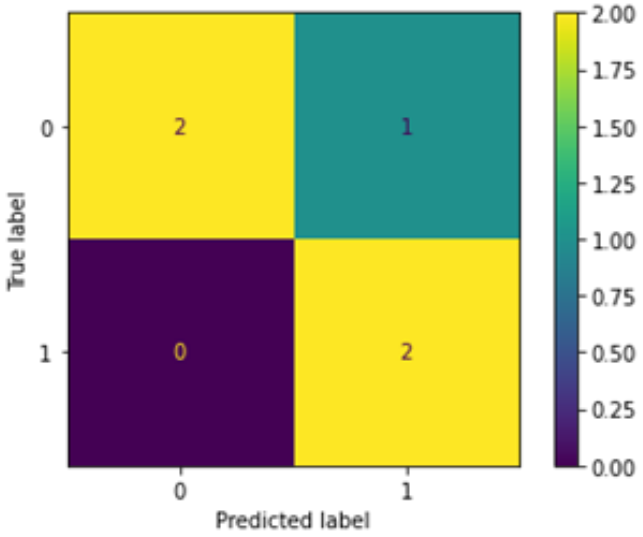

Fig. 8 Word2Vec Model Confusion Matrix

ham	Let me just open , yes , the notepad. Ok now could you repeat all the problems you want me to look at. I want you to go talk to the security guy, figure out how and why I got a notification on my phone last night about the office being compromised. Also call the wifi guy to see why our network has been acting up lately. Alright I got security being compromised and network jam. Anything else boss? No, update me on these topics later once you've finished.

In this case, the model could have seen words like "wifi", "network", "security", "compromised", etc. Since these words all are related to tech support and could thus be used by scammers, the model misclassified the example. This is another situation that could easily have been resolved if I had access to a larger and more detailed dataset.

Discussion

After looking at the results of each of the models and analyzing the strengths and weaknesses of each model, I concluded that the best model to use in a real-world setting would be the Word2Vec model. Although the model tended to misclassify ham examples, it is better to have false positives rather than false negatives as seen in the other models. Users would much rather have an alert pop up telling them about a suspected scam and then allow them to decide than for an actual scam call to get past the detection system. Implementing an effective detection system in a real-world setting requires much more than just identifying the best model for the problem. Many factors need to be considered before the project can be used by the public. The project would require a Speech to Text API to be implemented along with it because the model takes in input in the form of text and not audio. This problem can easily be countered using the Google Speech-to-Text API[14]. Next comes the problem of privacy. A censorship system also needs to be implemented that would black out sensitive information – names, bank details, etc. – as the transcripts of each conversation would have to be inputted into the model to produce results in real time. Users must be well informed about how the software works and provide their consent for the project to record the data from their calls. A system should also be put in place that enables users to decide when the AI model will analyze their calls to look for scams. The model should not be allowed to record every single call and should only be implemented at the user's request. It is also imperative that we inform the caller's side about the use of such a model as their data is being recorded. This might also add an unintended extra layer of protection against these scams. The scammers may get intimidated by the fact that their actions are being recorded by an "anti-scam" software and might reconsider even trying to perform the scam in the first place. This is also an interesting aspect that needs to be explored. Users should be allowed to decide whether the transcripts from their conversations can be used to further improve the model or whether they should be deleted from the database. Because the different models explored in this paper had different strengths and weaknesses, the possibility of combining various models and algorithms also needs to be explored to optimize the performance of a tech support scam detection model.

Methods

A. Dataset Description

I required a series of examples of both real-life tech support scam calls and regular calls that could be made on a day-to-day basis. Due to privacy laws, such examples are not made public by telephone service providers. Thus, I had to build a dataset from scratch, a difficulty also faced by the Berkeley students when starting their project[7]. I resorted to surfing the internet to find as many examples of clips of recordings as I could find to transcribe them. I used the literature review written by students at Stony Brook University who included the transcripts of 3 entire conversations with tech support scammers that they acquired when they had to collect data[9]. Using these 3 transcripts and examples I could find on YouTube[15], I constructed a dataset that consisted of 24 conversations, each containing a few back-and-forths between the caller and the receiver (about five to ten utterances per speaker). in total.

Half of the conversations represented a case in which the call was a tech support scam (labeled as a scam) and the other half were examples of real-life conversations that could take place but in the same structure as the scam calls (labeled as ham).

Table 2 Table 1. Examples from the dataset

Category	Transcript
ham	Wait so what was her name again? Ya it was M- as in Mumbai- A- as in Argentina - R- as in Russia and Y - as in Yugoslavia Mary? Her name was Mary? Yes, I can't wait to see her again. Me too.
scam	You have to press and hold the Windows button there. You have to press and hold that button. And then press R. R as in Richard, simultaneously. OK. I did it, yes. What do you see now? I see this little window that says " run." All right. Type in there W, W, W, dot. L as in lemon . M as in Mary . I as in indigo . Number one. Dot com. And should I click OK? OK. Yeah. OK. What do you see now? If you see restore option do not click on restore, OK? OK. I see a page; there is a support connection. All right, I am going to generate a six - digit code, OK? You have to type that code into that box. Plus, you have to write it in a paper because this code will be your case number for Microsoft, all right?

B. Models

1. Simple LSTM

Recurrent Neural Networks, or RNNs for short, are common neural networks for NLP applications[16]. They have been shown to be reasonably accurate and effective when used to create language models and perform speech recognition tasks[16,17]. An RNN can recognize dependencies, but it can only use recently available data. This issue can be resolved with the aid of an LSTM (Long Short-Term Memory) because it can comprehend both the context and recent reliance. As a result, LSTMs are a particular type of RNN where being aware of context might be helpful. The main distinction between LSTM networks and RNNs is the replacement of hidden layer updates with memory cells. Because of this, they become more adept at identifying long-range relationships in data. Thus, I chose to use an LSTM instead of a basic RNN for my preliminary model.

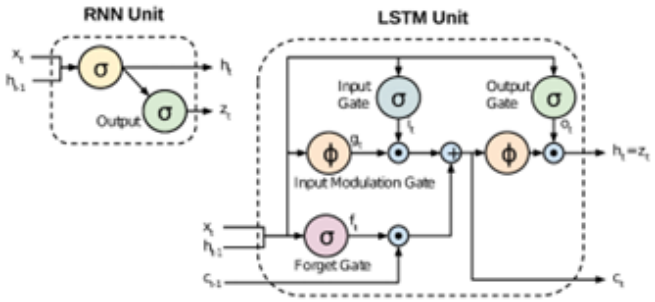

Fig. 9 Distinction between RNN and LSTM structures[6]

After tokenizing and padding the data, $num_words = 50$ (the maximum number of words to keep, based on word frequency. Only the most common words will be kept) and $max_length = 100$ (maximum length of all sequences), I constructed the model.

Note: - I tuned the parameters for the neural network based on the accuracy of my training set and then used the best-performing values to act on the test set.

2. NLTK Model with Classifier

The Natural Language Toolkit, commonly known as NLTK, is a top platform for creating programs using human language data[18]. It offers a collection of text-processing libraries with wrappers for powerful industrial-strength categorization, tokenization, stemming, tagging, parsing, and semantic reasoning tools. I used NLTK due to its ability to efficiently preprocess the textual data before it is analyzed using a machine learning algorithm. I used NLTK's PorterStemmer and stopwords list to preprocess my data. The PorterStemmer is used to "stem" the words in the text. Stemming is the process by which words are reduced to their root word or their "stem" – "programs", "programmer", and "programming" are all reduced to their stem word "program". After removing the stopwords and stemming the data, I used the CountVectorizer from the $sklearn.feature_extraction.textlibrary$. The count vectorizer converts textual data to a matrix of token counts i.e., it converts a given text into its vector form using the frequency of the occurrence of words in the text. Because the number of words in each example was around 100 to 150, I used a CountVectorizer with $max_features = 70$.

On this list of tokens, I utilized the TF-IDF transformer. TF-IDF stands for term frequency-inverse document frequency. It is a statistical measure that checks how relevant a certain word is to a text. It is calculated using the following formula:

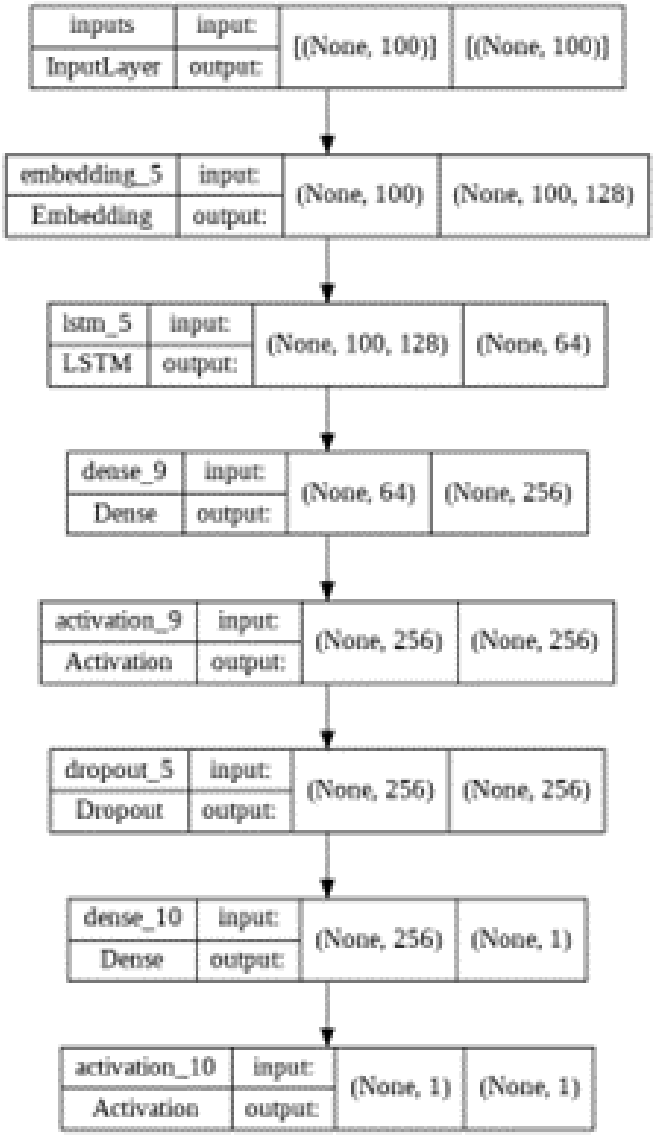

Fig. 10 Diagram of LSTM Model

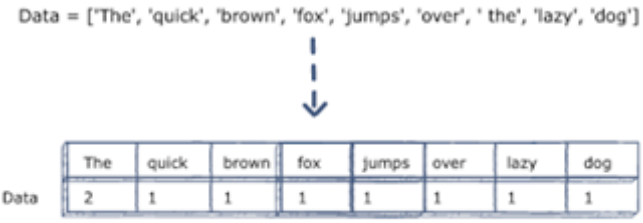

Fig. 11 Example of CountVectorizer being used on certain data

I trained a Random Forest Classifier with the resulting data to produce the results of the model with a default value of

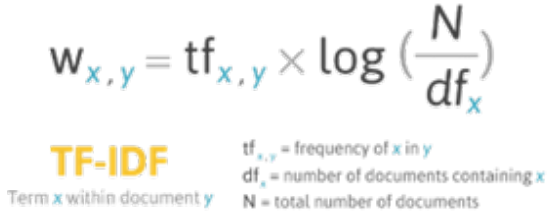

Fig. 12 Formula used to calculate a word's TF-IDF value

$n_estimators$ (100 – the number of decision trees that the classifier will use).

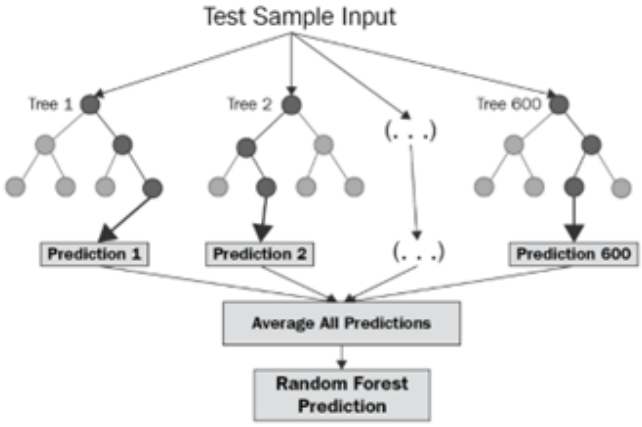

Fig. 13 Simple diagram of a Random Forest Classifier

3. Word2Vec

Word2Vec[19] is a word embedding method that can extract the idea of similarity between words or items. It can identify semantic similarity, synonyms, and concept classification patterns. This ability to detect the relationships between words would be useful in detecting words or phrases used by a scammer that could be similar to the speech used in the dataset examples because tech support scammers all tend to follow the same structure when performing a scam.

After using the same preprocessing methods from the NLTK model (removing stopwords and stemming), I imported Word2Vec from the gensim library. When creating the mode, I initialized the size of the vector as 20. This value would usually be much higher in other problems, such as recommender systems, but as the number of words in each of my examples is comparatively less, I reduced the size of each vector assigned to the words. I initialized the "workers" attribute as 2. This attribute refers to the number of independent threads doing simultaneous training in the Word2Vec model. I used the effective_n_jobs(-1) function from gensim.utils to determine the correct number of threads to use for my model.

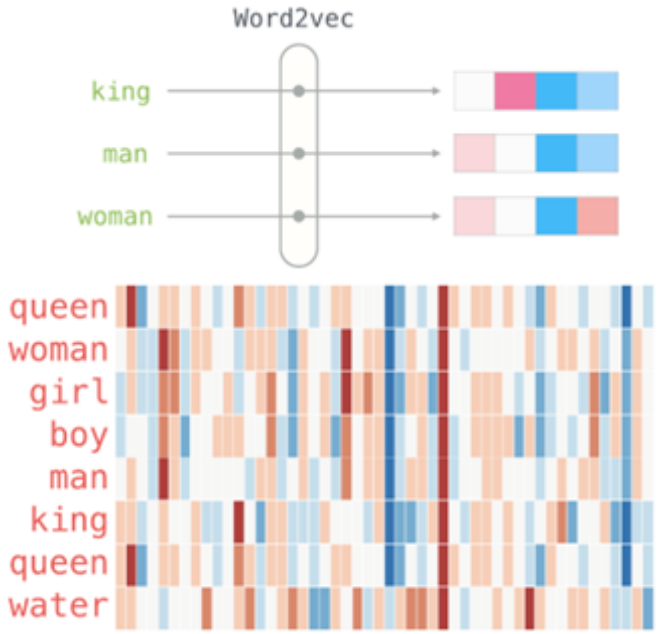

Fig. 14 Basic illustrations of Word2Vec Embedding

I used the Average Word2Vec technique to embed my data. Word2Vec itself only embeds each word separately whereas Average Word2Vec allows for each sentence to be given an embedded value by averaging the vectors of each word in the sentence.

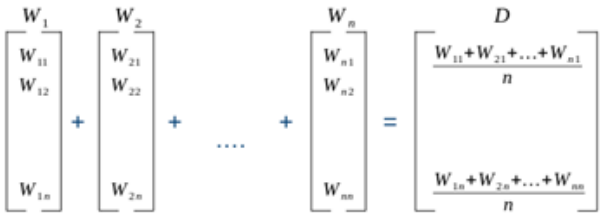

Fig. 15 Depiction of how Average Word2Vec is conducted

I used the Random Forest Classifier, then, to act on my embedded data.

C. Metrics

The metric I will be using to analyze the performance of the different models is accuracy. The formula of accuracy is derived from a confusion matrix as shown below:

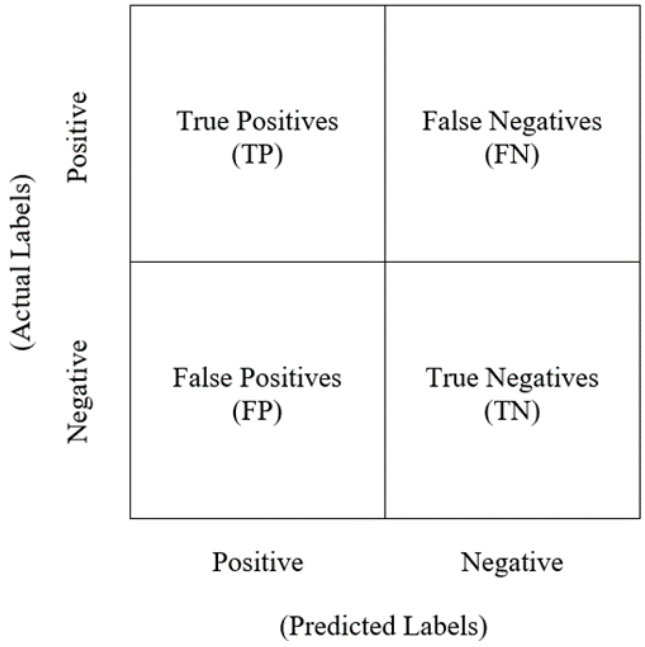

Fig. 16 A confusion matrix

$$Accuracy = \frac{CorrectPrediction}{TotalCases} \times 100\%$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \times 100\%$$

Formula derived from a confusion matrix **Fig. 16.**

Acknowledgement

Thank you Alex Cabral from Harvard University for your guidance in the development of this research paper.

References

1 M. Labs, *Tech support scams: help and resource page*, https://blog.malwarebytes.com/tech-support-scams/, Retrieved 5 August 2022, from.

2 *Protect yourself from tech support scams*, https://support.microsoft.com/en-us/windows/protect-yourself-from-tech-support-scams-2ebf91bd-f94c-2a8a-e Retrieved 5 August 2022, from.

3 Refund and R. Scams, https://consumer.ftc.gov/articles/refund-recovery-scams, Retrieved 5 August 2022, from.

4 B. News and I. News, *Why Indians lose the most money in tech scams - Times of India*, https://timesofindia.indiatimes.com/business/india-business/why-indians-lose-most-money-in-tech-scams/articleshow/84637261.cms, Retrieved 5 August 2022, from.

5 *Older adults hardest hit by tech support scams*, https://www.ftc.gov/news-events/

data-visualizations/data-spotlight/2019/03/
older-adults-hardest-hit-tech-support-scams,
Retrieved 5 August 2022, from.
6 *5 Indian BPOs indicted in multimillion-dollar call center scam in US*, https://www.hindustantimes.com/india-news/15-people-5-indian-bpos-indicted-in-multimillion-dollar-call-centre-scam-in-us/story-Rwj2EGGLpATHTRBxYULVpM.html, Retrieved 5 August 2022, from.
7 *Sounds Phishy: Protecting Consumers Against Phone Phishing*, https://www.ischool.berkeley.edu/projects/2019/sounds-phishy-protecting-consumers-against-phone-phishing, Retrieved 5 August 2022, from.
8 *IBM*.
9 https://s3.documentcloud.org/documents/3515987/Dialonescammers.pdf, Retrieved 5 August 2022, from.
10 https://www.usenix.org/system/files/sec20-prasad.pdf, Retrieved 5 August 2022, from.
11 https://reader.elsevier.com/reader/sd/pii/S1877050921011741?token=32D0B046F395B996E45177DF82255ABBA23F91BE590BBDD46A2EA413A33915944852421A70C4102BC832F655C667B34E&originRegion=eu-west-1&originCreation=20220805170816, Retrieved 5 August 2022, from.
12 I. Palacio, *Natural language processing for scam detection. Classic and alternative analysis techniques*, https://deliverypdf.ssrn.com/delivery.php?ID=735105085006007012091073126004067106120009055009062036075099119085093108127079092094009013096038009044113125095125085
EXT=pdf&INDEX=TRUE, Retrieved 5 August 2022, from.
13 *An Incremental Identification Method for Fraud Phone Calls Based On*, ieeexplore.ieee.org/document/8947271/., Accessed 25 Aug. 2022.
14 S. to Text, *Automatic Speech Recognition — Google Cloud*, https://cloud.google.com/speech-to-text/?utm_source=google&utm_medium=cpc&utm_campaign=emea-ae-all-en-dr-bkws-all-all-trial-p-gcp-1011340&utm_content=text-ad-none-any-DEV_c-CRE_544053352009-ADGP_Hybrid%20%7C%20SKWS%20-%20PHR%20%7C%20Txt%20~%20AI%20%26%20ML%20~%20Speech-to-Text%23v39-KWID_43700066203492321-kwd-475797931654-userloc_1000013&utm_term=KW_convert%20voice%20to%20text-NET_g-PLAC_&&gclsrc=aw.ds&gclid=EAIaIQobChMI8KmL9tav-QIVD9Z3Ch1fDgeqEAAYASAAEgIcO_D_BwE, Retrieved 5 August 2022, from.
15 L. Tech, *Tech Support Scammer Uses Each and Every Trick in the Book! [Video*, https://www.youtube.com/watch?v=EqSqrlbYmm8&list=PL7UZeBkn6-yIp_laO-qTc-adQY-NnVK3m&index=7, Retrieved from.
16 I. Education, *What are Recurrent Neural Networks?*, https://www.ibm.com/cloud/learn/recurrent-neural-networks, Retrieved 5 August 2022, from.
17 *Understanding How Recurrent Neural Networks Model Text*, http://www.cs.toronto.edu/~guerzhoy/tmp/understand-rnn/, Retrieved 25 August 2022, from.
18 N.L.T.K., *Natural Language Toolkit*, https://www.nltk.org, Retrieved 5 August 2022, from.
19 *word2vec — TensorFlow Core*, https://www.tensorflow.org/tutorials/text/word2vec, Retrieved 5 August 2022, from.