EPiC Series in Computing

Volume 77, 2021, Pages 13–22

Proceedings of ISCA 30th International Conference on Software Engineering and Data Engineering

Developing a Google Chrome Extension for Detecting Phishing Emails

Hongkai Chen[1,2*] and Mohammad Hossain[1†]

[1] University of Minnesota Crookston, MN, USA

[2] University of California San Diego, CA, USA

hongkai@ucsd.edu, hossain@crk.umn.edu

Abstract

Phishing is a fraudulent attempt where attackers trick the victims into disclosing sensitive information under pretenses. This research project aims to develop a Google Chrome extension to detect phishing emails. We firstly collected a number of phishing email samples. Then we used text mining techniques to find out the words that are important in phishing emails. Next, we developed a classifier model that the Chrome extension will use to detect phishing emails using those words. The next step was to test the extension with phishing email samples and standard (non-phishing) email samples. After that, the evaluation metrics were collected. We found our extension was able to identify phishing emails and non-phishing emails with a relatively high degree of accuracy.

1 Introduction

Google Chrome extensions (Figure 1) are programs that can be installed into Chrome in order to change the browser's functionality [1]. The extension can provide new functionality by combining existing web browser features and making it possible for users to do many things simultaneously [2]. Some examples of Google Chrome extension functions include blocking ads from being displayed, password management, and translating the web page.

In phishing attacks, phishers (attackers) trick the victims into divulging critical information under pretenses [3]. There are several types of phishing scams, including SMS messages, email, and fake websites. This research project aims to develop a Google Chrome extension to detect phishing emails using a model or algorithm.

[*] Work was done when the first author was at the University of Minnesota Crookston.

[†] Corresponding Author

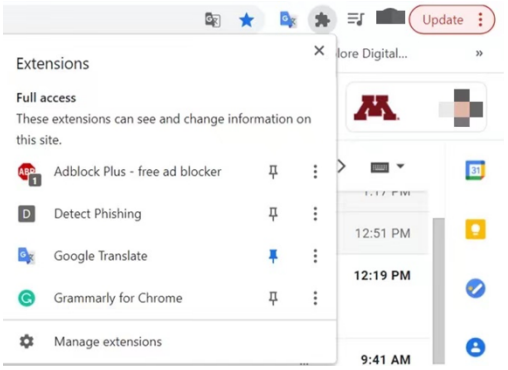


Figure 1: Google Chrome Extensions

```
Date: Thu, 15 Oct 2020 15:41:01 +0000
Subject: 2020 FACULTY EVALUATION
From: billymaxwell204@gmail.com
To: user@berkeley.edu


Google Forms

Jim Knowlton invited you to fill out a form:

2020 FACULTY EVALUATION

Designed for Microsoft and Office 365 users only

FILL OUT FORM
hxxps://docs.google.com/forms/d/e/1FAIpQLSfUCvno3DdViZI24

Create your own Google Form
```

Figure 2: Phishing Email Sample

2 Background and Why Chrome Extension

According to the Anti-Phishing Working Group (APWG), the number of phishing detections in the first quarter of 2018 increased by 46% compared with the fourth quarter of 2017 [4]. During the COVID-19 pandemic, phishing is becoming a security issue more than ever. The United States Department of Homeland Security reported that advanced persistent threat (APT) groups and cybercriminals are targeting individuals, small and medium enterprises, and large organizations with COVID-19-related scams and phishing emails in the alert [5]. With the continuous development of camouflage technology, it is becoming more and more essential to develop an effective and precise tool to detect and prevent phishing.

Google Chrome extension is suitable as a platform for development for many reasons. Firstly, the Google Chrome Extension is based on and installed inside the Chrome browser, which has a minimal effect on the use of the computer system resources. Chrome is a very popular browser around the world, and users can easily access it. Secondly, most users view and write emails via browsers, which means that the browser build-in function, like extensions, will be the most straightforward and effective way to detect phishing emails. Thirdly, the extension will be small in size and easy to install inside the browser. Google Chrome extension is safe and user-friendly because it just needs one click to install. When users want to stop it, they need one click to disable the extension [6].

3 Collecting Phishing Email Samples

To analyze phishing emails and test the extension, we collected more than fifty phishing email samples from various sources. Most of the samples come from the Berkeley Information Security Office [7]. In addition, we collected some of them from our university email inbox. The collected phishing email samples were used as the data set. We analyzed the samples using text mining techniques and found out the suspicious words in phishing emails. Then, in the testing phase, we used those phishing email samples to test our extension.

4 Employing Text Mining to Determine the Suspicious Words

We imported the phishing email samples as a document to find out the suspicious words for detecting phishing emails. Then, we wrote a program to determine the tf-idf (term frequency-inverse

document frequency) for each word in the document. The tf-idf weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus [8]. Term Frequency (TF) measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length as a way of normalization [9]:

$$tf(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document}$$

Inverse Document Frequency (IDF) measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as "is," "of," and "that," may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scaling up the rare ones by computing the following [9]:

$$idf(t) = \log \frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}$$

$$tf\_df(t) = tf(t) \times idf(t)$$

Then we used the following code (Figure 3) in Python to implement the calculation of tf-idf.

```
In [7]:
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
import numpy as np
import pandas as pd
import re

In [8]:
sentences = list()
with open("sample.txt") as file:
    for line in file:
        for l in re.split(r"\.\s|\?\s|\!\s|\n",line):
            if l:
                sentences.append(l)

In [9]:
cvec = CountVectorizer(stop_words='english', min_df=3, max_df=0.5, ngram_range=(1,2))
sf = cvec.fit_transform(sentences)

In [10]:
transformer = TfidfTransformer()
transformed_weights = transformer.fit_transform(sf)
weights = np.asarray(transformed_weights.mean(axis=0)).ravel().tolist()
weights_df = pd.DataFrame({'term': cvec.get_feature_names(), 'weight': weights})

In [11]:
weights_df.sort_values(by='weight', ascending=False).head(50)
```

Figure 3: Code Snip for tf-idf

After executing the above code, we got the list of suspicious words. Figure 4 shows a part of the suspicious word list.

Word ID	Term	Weight	Word ID	Term	Weight
78	email	0.034288	124	link	0.014512
184	subject	0.030727	5	access	0.014212
7	account	0.025981	97	hello	0.013772
23	berkeley	0.023322	178	sincerely	0.013636
93	google	0.016028	112	information	0.013365
59	dear	0.015539	119	job	0.012687
131	mail	0.015045	195	time	0.012545
162	request	0.01475	182	student	0.011917

Figure 4: Part of the Suspicious Word List

After filtering out some unrelated words such as location words and date words, we got the final version of the word list, as shown in Figure 5.

```
var searches = ["urgent", "calls", "paid", "need", "gift", "gifts", "cards", "card"
    , "urgently", "response", "needed", "login", "expiring", "soon", "immediate"
    , "immediately", "free", "detect", "pay", "job", "access", "expire", "friend", "lowest"
    , "price", "serious", "action", "database", "winner", "refund", "files", "activate"
    , "activated", "wage", "vital", "irregular", "docs", "invited", "account", "employment"
    , "notice", "service", "bcourse", "employee", "phone", "information", "dirks"];
```

Figure 5: Final Suspicious Word List

5 Developing the Chrome Extension

To build the Chrome extension, we went through the video tutorial in reference [10]. The first step was creating the "manifest.json" file inside the extension folder. This JSON file contains the name of the extension and the version of it. And it also links to the content script and the popup HTML file. Figure 6 shows the "manifest.json" file. In line 8, it connects the content script "content.js." In line 12, it connects the popup script "popup.html."

{} manifest.json > ...

```
1  {
2      "name": "Detect Phishing",
3      "version": "1.0",
4      "manifest_version": 2,
5      "content_scripts": [
6          {
7              "matches": ["<all_urls>"],
8              "js": ["content.js"]
9          }
10     ],
11     "browser_action": {
12         "default_popup": "popup.html",
13         "default_title": "Detect Phishing"
14     }
15 }
```

Figure 6: File "manifest.json"

After creating the file "manifest.json," we went to the Chrome extensions developer page and clicked the "Load unpacked" button to upload the folder from our computer. Then the extension was visible in the Chrome browser, and it was ready to use.

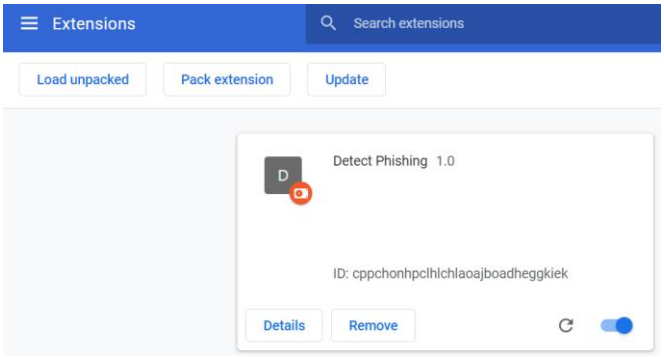
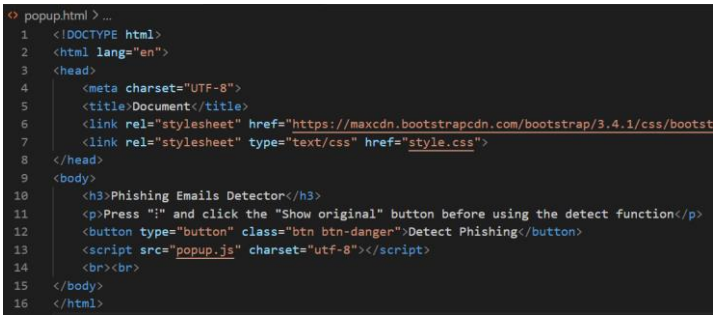


Figure 7: Extension Page on the Browser

We created the "popup.html" file that will run when users click the extension button. This HTML file contains a button and some instructions. The user needs to follow the instruction and click the button for the detecting function to start. Figure 8 shows the source code of the "popup.html" file. Figure 9 shows the popup window after clicking the extension.



```
popup.html > ...
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootst
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <h3>Phishing Emails Detector</h3>
    <p>Press ":" and click the "Show original" button before using the detect function</p>
    <button type="button" class="btn btn-danger">Detect Phishing</button>
    <script src="popup.js" charset="utf-8"></script>
    <br><br>
</body>
</html>
```

Figure 8: File "popup.html"

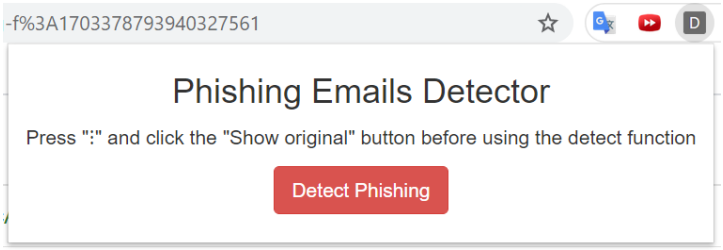


Figure 9: Popup Window for the Extension

We created the "popup.js" file. This file will work as a response to the "Detect Phishing" button. The Javascript code will use an event listener to wait for the detection counter result from the "content.js" file. If the counter is greater than six, the program will display the warning message "This is a phishing email" using the appendChild method to the HTML body. If the counter is less or equal to six, the program will display the message "This is not a phishing email." Figure 10 shows the source code of the "popup.js" file.

```
popup.js > ...
1  document.addEventListener('DOMContentLoaded', function(){
2      document.querySelector('button').addEventListener('click',
3      onclick, false)
4
5      function onclick() {
6          chrome.tabs.query({currentWindow: true, active: true},
7          function (tabs){
8              chrome.tabs.sendMessage(tabs[0].id, 'hi', setCount)
9          })
10     }
11
```

```
12     function setCount (res) {
13         if (res.count > 6) {
14             const div = document.createElement('div')
15             div.textContent = `This is a phishing email!`
16             div.style = `color: red; font-size: 18px;`
17             document.body.appendChild(div)
18         }
19         else {
20             const div = document.createElement('div')
21             div.textContent = `This is not a phishing email`
22             div.style = `color: green; font-size: 18px;`
23             document.body.appendChild(div)
24         }
25         const br = document.createElement('br')
26         document.body.appendChild(br)
27     }
28 }, false)
```

Figure 10: File "popup.js"

6 Main Function for Detection

We created the "content.js" file for the main function for detection. Figure 11 shows the source code of the "content.js" file. This file will read the web page and execute the detecting function. A for loop was used to go through all the words in the suspicious list. The program can do a match for the web page with a word each time. The innerHTML.match() function can check if the web page contains the word or not. In line 13 of Figure 11, we created a variable "matches" to store all matched words each time. If the variable "matches" is not equal to null, the web page contains the word. Then the counter will increase by one. After the program completes the matches for all words, it will send the counter to the "content.js" file. Based on the experience, we set the boundary condition as six. If the counter is greater than six, it will report phishing. That is the detection result. If the boundary condition is more than six, we get more false negatives. If the boundary condition is less than six, we get more false positives.

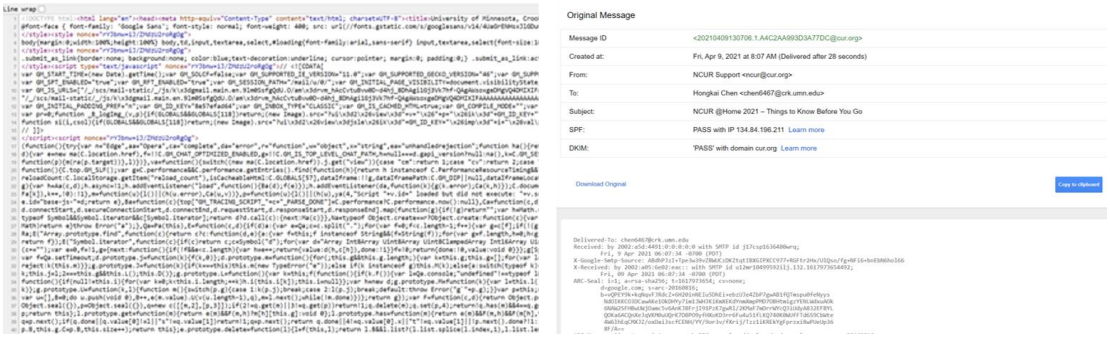
```
content.js > ...
1  chrome.runtime.onMessage.addListener(
2      function(request, sender, sendResponse) {
3          var searches = ["urgent", "calls", "paid", "need", "gift", "gifts", "cards", "card"
4              , "urgently", "response", "needed", "login", "expiring", "soon", "immediate"
5              , "immediately", "free", "detect", "pay", "job", "access", "expire", "friend", "lowest"
6              , "price", "serious", "action", "database", "winner", "refund", "files", "activate"
7              , "activated", "wage", "vital", "irregular", "docs", "invited", "account", "employment"
8              , "notice", "service", "bcourse", "employee", "phone", "information", "dirks"];
9
10         var totalPoints = 0;
11         for (search of searches) {
12             let re = new RegExp(search, 'gi')
13             let matches = document.documentElement.innerHTML.match(re);
14             if (matches != null) {
15                 totalPoints = totalPoints + 1;
16             }
17         }
18
19         sendResponse({count: totalPoints})
20     }
21 );
```

Figure 11: File "content.js"

7 The Main Challenge

We used Gmail as the email service. The main challenge of Gmail is that the HTML source code is encrypted. If we run the extension directly on the email page, the extension cannot access the email content. Figure 12(a) shows the encrypted HTML code.

To solve this problem, we added an instruction asking users to click the "show original" button before pressing the detect button. Figure 12(b) shows the Original Message page. On this page, the extension can read the HTML for the email message since the HTML is not protected.



(a) **(b)**

Figure 12: (a) Encrypted HTML Code (b) Original Message Page

8 Experimental Results

This section presents the testing result of the extension on different email services. We sent the phishing email samples to the email inbox and ran the extension on each sample email. The output results of the extension were recorded. We used the following metrics to evaluate the performance of the extension.

Accuracy: Accuracy shows how many of the predictions are correct [11].

$$Accuracy = \frac{Number\ of\ true\ positives + Number\ of\ true\ negatives}{Number\ of\ phishing\ samples + Number\ of\ Standard\ samples}$$

Precision: Precision measures how good our model is when the prediction is positive [11].

$$Precision = \frac{Number\ of\ true\ positives}{Number\ of\ true\ positives + Number\ of\ false\ positives}$$

Recall: Recall measures how good our model is at correctly predicting positive classes [11].

$$Recall = \frac{Number\ of\ true\ positives}{Number\ of\ true\ positives + Number\ of\ false\ negatives}$$

F1 Score: F1 score is the weighted average of precision and recall. It is a more useful measure than accuracy for problems with uneven class distribution because it takes into account both false positives and false negatives [11].

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Sensitivity: Sensitivity is also known as the true positive rate (TPR). It is the same as recall. Hence, it measures the proportion of positive class that is correctly predicted as positive [11].

Specificity: Specificity (true negative rate) is similar to sensitivity but focused on the negative class. It measures the proportion of negative class that is correctly predicted as negative [11].

$$Specificity = \frac{Number\ of\ true\ negatives}{Number\ of\ true\ negatives + Number\ of\ false\ positives}$$

False positive rate (FPR): The false positive rate is defined as the probability of falsely rejecting the null hypothesis [12].

$$False\ positive\ rate = \frac{Number\ of\ false\ positives}{Number\ of\ false\ positives + Number\ of\ true\ negatives}$$

We tested the extension with thirty phishing email samples and twenty standard (non-phishing) email samples. Figure 13 summarizes the findings of our extension. By default, Gmail has a phishing and spam protection feature. It will report phishing or spam warning to users when detecting suspicious emails. We also tested the Gmail protection feature with the phishing email samples and standard email samples, summarized in Figure 14.

Figure 13: Distribution of TP, FP, FN, and TN (Our Extension)

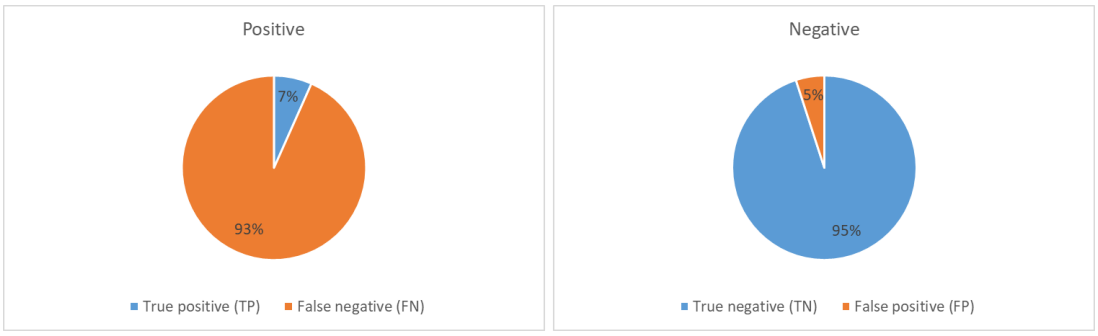


Figure 14: Distribution of TP, FP, FN, and TN (Gmail Protection Feature)

After calculation, we got the following figure showing all measure values as a clustered column chart. The results show that our extension has higher overall accuracy and higher sensitivity compared with the Gmail protection feature, while the Gmail protection feature has a higher specificity and lower false positive rate. Those results suggest that our extension has significantly better performance when detecting phishing emails.

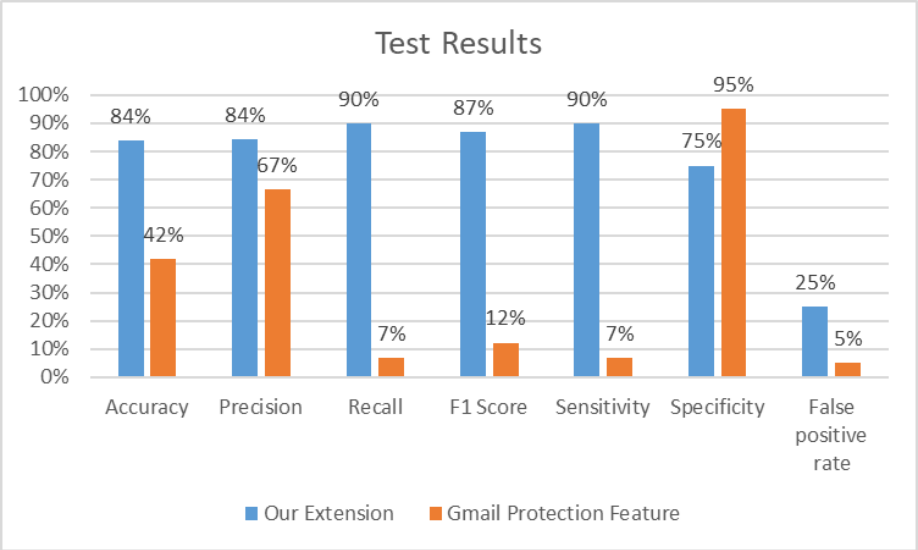


Figure 15: Measure Metrics for the Extension

9 Future Work

In order to improve the accuracy and reduce the false positive rate, we will find some phrases and short sentences that are important in phishing email samples. Then, we will add those phrases and short sentences into the suspicious word list and test the extension again. We will also employ sentiment analysis, which is the field of study that examines human opinions and sentiments towards different entities [13], to find the neutral, positive, and negative polarities in the sentences. According to those polarities, we can improve the detecting function by weighing up the segments that have negative polarities.

Currently, the “Detect Phishing” button needs to be clicked to activate the extension and run the detecting function. In the future, we will try to make an automatic detecting feature. With this feature, the extension will scan the web page first. If the web page is an email service page, the extension will automatically detect the email. Once the extension completes the detection in the background, it will send a warning popup window if the email is phishing-likely.

In addition, we will try to improve the extension so that it can access the encrypted Gmail page directly. With this improvement, users do not need to click the “show original message” button to let the extension access the unencrypted email HTML source code, improving the extension's usability. In the future, we will also develop the detecting function for languages other than English, such as Spanish, Chinese, and Arabic, in order to recognize emails in different languages.

10 Conclusion

The above testing results show that our Chrome extension has higher accuracy and precision when detecting phishing emails. Meanwhile, it has a relatively lower false positive rate. Those results indicate that the extension has a good ability to identify phishing emails and non-phishing emails.

Due to the large number of emails received every day, it is not easy for users to identify those phishing emails. Our extension can notify users of the existence of phishing emails and reduce the chance of users being scammed.

References

[1] L. Abrams, "What are Google Chrome Extensions?," April 2017. [Online]. Available: https://www.bleepingcomputer.com/tutorials/understanding-google-chrome-extensions/.

[2] P. Mehta, "Introduction to Google Chrome Extensions," in *Creating Google Chrome Extensions*, Apress, Berkeley, CA, 2016. [Online]. Available: https://doi.org/10.1007/978-1-4842-1775-7_1.

[3] R. Aravindhan, R. Shanmugalakshmi, K. Ramya and Selvan C., "Certain investigation on web application security: Phishing detection and phishing target discovery," 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), 2016, pp. 1-10, doi: 10.1109/ICACCS.2016.7586405.

[4] "Phishing Activity Trends Report 1st Quarter 2018," July 2018. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2018.pdf.

[5] "Alert (AA20-099A) COVID-19 Exploited by Malicious Cyber Actors," April 2020. [Online]. Available: https://www.us-cert.gov/ncas/alerts/aa20-099a.

[6] CodeIT, "The Importance of Chrome Extension Development," April 2018. [Online]. Available: https://medium.com/@codeit_llc/the-importance-of-chrome-extension-development-b8cbb9405bf2.

[7] "Phishing Examples Archive," [Online]. Available: https://security.berkeley.edu/education-awareness/phishing/phishing-examples-archive.

[8] A. Mishra and S. Vishwakarma, "Analysis of TF-IDF Model and its Variant for Document Retrieval," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), 2015, pp. 772-776, doi: 10.1109/CICN.2015.157.

[9] "What does tf-idf mean?," [Online]. Available: http://www.tfidf.com/.

[10] K. R. Young, "How To Make Chrome Extensions," April 2018. [Online]. Available: https://youtu.be/Ipa58NVGs_c.

[11] S. Yildirim, "How to Best Evaluate a Classification Model," March 2020. [Online]. Available: https://towardsdatascience.com/how-to-best-evaluate-a-classification-model-2edb12bcc587.

[12] "False Positive Rate," [Online]. Available: https://www.split.io/glossary/false-positive-rate/.

[13] H. Soong, N. B. A. Jalil, R. Kumar Ayyasamy and R. Akbar, "The Essential of Sentiment Analysis and Opinion Mining in Social Media: Introduction and Survey of the Recent Approaches and Techniques," 2019 IEEE 9th Symposium on Computer Applications & Industrial Electronics (ISCAIE), 2019, pp. 272-277, doi: 10.1109/ISCAIE.2019.8743799.