Implementation of a Spam Detector in Python for Corporate Companies

Abstract: Many companies, while not categorizing spam as a form of attack, may still consider it a significant threat. When referring to spam, it represents an excessive volume of unwanted and unsolicited electronic mail sent to company email accounts [1]. Although it is often used for marketing purposes, in many cases, it is also employed by malicious actors for detrimental purposes against the company. For instance, spam can be used to propagate viruses or infiltrate corporate computers [1] [2]. Furthermore, spammers may send emails with the aim of directing users to phishing web pages, attempting to obtain confidential information from authorized personnel within the company. In general, spam is, at its simplest, an annoying form of mail, and in its worst cases, it serves as an enabler for spyware, viruses, intrusion, and phishing threats [3].

Keywords: Spam, protection methods, informational resources, unwanted email, malicious.

I. INTRODUCTION

Spam can be identified based on the following characteristics:

- Mass email distribution: In this case, electronic mail is directed towards multiple and diverse targets, rather than a specific individual.
- Anonymity: In this scenario, the true identity of the sender or senders is kept confidential.
- Unwanted email: The sent electronic messages are either unnecessary for the company or not anticipated by any entity within the organization [2] [3].

Now, let's consider the implementation of the system in the Python environment that identifies these harmful electronic emails sent to the company as spam.

Firstly, messages characterized as spam are presented as shown below. Their total count is 5728. This data is of particular importance for the system in terms of how it classifies messages as spam (Figure 1).

II. METHODS

As shown in the figure, these data are marked as 0 and 1. Where 0 denotes that the email is not spam, and 1 characterizes it as spam and unsafe. After instructing the system with this data, it can automatically determine which emails are spam without human intervention [4]. The code used to create this system is presented in main.py (Figure 2).
After importing the Python packages and libraries to be used, the dataset is incorporated into the project. Based on the existing datasets, visualization is performed. Visualization is a method that represents the quantity of spam within the data that the system has learned with clarity and numerical values [3] [4]. This is illustrated as follows (Figure 3).

The orange quantity represents the amount of spam, while the blue quantity indicates the count of regular emails. Since the computer environment recognizes words as numbers, this information is later converted into numbers through a vectorization process. All words within the emails are included here. Subsequently, within the transferred dataset, the system learns in the background how often each word is used in the email data [5].

After the execution of these processes, the converted data into numbers are divided into two parts: test data and training data. A necessary model is built using the training data. Once this model is established, it must be validated using the test data. It is important to note that the data used for testing should be previously unseen by the established model. Otherwise, the model will not function accurately. To prevent such a situation, test data is kept separate from the model [6].

**Spam frequency**

![Figure 3. A visual representation of the amount of spam](image-url)
Once the model is created and configured, it is stored in memory. Subsequently, the same model is loaded. Then, we use the test data to evaluate the established model [6] [7]. An example of the code for checking spam in the model is shown as follows (Figure 4).

```python
# Example email to classify
email = "Subject: unbelievable new homes made easy by wanting to show you this homeowner you have a 10% home loan at a 3.72 fixed rate. This offer is being extended to you unconditionally and take advantage of this limited time opportunity. All we ask is that you visit our website and consider look forward to hearing from you, Dorcas Pittman"
```

**Figure 4.** Test data

The resulting output is displayed below (Figure 5). If the submitted data is not spam, then the output will indicate that the data is a safe message and not spam [7] [10].

```python
# Print the prediction
if prediction[0] == 1:
    print("The email is predicted as spam.")
else:
    print("The email is predicted as not spam.")
```

**Figure 5.** Spam email detection

In the presented figures, in addition to the mentioned elements, the codes incorporating the prediction condition are also provided (Figure 6).

```python
def process_email(text):
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)
    clean = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]
    return clean

preprocessed_email = process_email(email)

# Vectorize the preprocessed email
vectorized_email = vectorizer.transform([' '.join(preprocessed_email)])

# Make prediction
prediction = model.predict(vectorized_email)

# Print the prediction
if prediction[0] == 1:
    print("The email is predicted as spam.")
else:
    print("The email is predicted as not spam.")
```

**Figure 6.** Data recognition through a vectorizer

After the model is constructed, it is essential to assess the extent to which it makes accurate and precise decisions. To achieve this, the following code example should be executed (Figure 7).
As evident from the figure, the accuracy metric of the created model is 99%. This implies that it can accurately classify 99 out of every 100 incoming emails.

Furthermore, it is possible to create a matrix that describes how accurately the model identified the normal and safe emails, how many spam emails were misclassified as non-spam, how many normal emails were identified as spam, and how many spam emails were correctly classified as spam among all the test data entered into this model [7] [8] [9]. The mentioned matrix is presented below (Figure 8).

III. RESULTS

The protection of informational resources in companies holds significance not only for potential users but also for the companies themselves. When it comes to safeguarding databases and information sources, it is not merely achieved through software solutions. Hardware infrastructure establishment and utilization should also be considered. Corporations and organizations that ensure complete security safeguard their reputations and stand out sharply from their active competitors.

Each security model possesses distinct characteristics, and consideration should be given to their component composition when utilizing these models. Furthermore, to protect the company's assets from potential computer threats, security measures, such as maintaining and regularly updating signatures, should be diligently upheld. In the Python environment, spam protection detectors can effectively mitigate the problems encountered by companies. These detectors facilitate a more dynamic and robust defense against relevant attacks, thereby enhancing the overall security of the company.
REFERENCES