Spam Detection Using Natural Language Processing (NLP)

[1]M. Kishore Kumar, [2]S. Venu Gopal Reddy, [3]P. Rajesh Reddy, [4]C. Sri Sowkhya Reddy, [5]M S Swetha Patil
[1,2,3,4] *UG Student Dept. Of CSE,* [5]*Professor Dept. Of CSE*
[1,2,3,4,5] *Presidency University, Bengaluru - 560064*

Abstract-Spam detection has become a critical area of research due to the increasing prevalence of unsolicited and malicious emails. Leveraging Natural Language Processing (NLP) for spam detection offers powerful tools for analyzing and classifying email content with high accuracy. This paper explores the evolution of spam detection, highlighting traditional machine learning (ML) approaches and recent advancements like Bidirectional Encoder Representations from Transformers (BERT), AMALS (Alternating Minimum and Least Squares), and feature extraction techniques such as TF-IDF. The integration of NLP techniques enables models to capture contextual and semantic features in text, significantly improving classification performance. Despite these advancements, challenges such as data imbalance, evolving spam strategies, and feature optimization persist. This research discusses current methodologies, evaluates their performance, and provides insights into future directions for building robust spam detection systems.

IndexTerms-Spam Detection, Natural Language Processing (NLP), Machine Learning, Text Classification, Email Filtering, Message Filtering, Feature Extraction, Sentiment Analysis, Text Mining, Data Preprocessing

INTRODUCTION

The widespread use of email for personal and business communication has made it a primary target for spam and phishing attacks. Spam emails not only waste time and resources but also carry significant security risks, such as malware or phishing links. Traditional spam detection systems relied on rule-based filters, which lacked the flexibility to adapt to evolving spam strategies.

With the advent of NLP, spam detection systems have seen remarkable improvements. NLP enables the analysis of textual data to identify patterns and extract features that differentiate spam from legitimate emails. Techniques such as tokenization, stopword removal, lemmatization, and advanced models like BERT have revolutionized this field by incorporating semantic understanding and contextual relationships in email content.

This paper reviews the state-of-the-art in spam detection using NLP, focusing on the challenges and innovations in the field. It aims to provide a roadmap for researchers and practitioners to develop effective and scalable spam detection solutions.

Keywords-Spam detection, Natural Language Processing, BERT, TF-IDF, AMALS, deep learning,

phishing emails, feature extraction, machine learning.

LITERATURE SURVEY

Spam Email Detection Using Deep Learning Techniques Isra'a AbdulNabi et al. proposed a model leveraging BERT (Bidirectional Encoder Representations from Transformers) for detecting spam emails, achieving an accuracy of 98.67%. The research emphasized pre-trained transformers' ability to account for word semantics in context, outperforming classical models like Naïve Bayes (NB) and k-Nearest Neighbors (k-NN). The study highlighted deep learning models such as CNNs and BiLSTM for spam detection, showing substantial improvements in feature extraction and classification when paired with BERT [1].

Phishing Email Detection Using NLP Said Salloum et al. conducted a systematic review focusing on NLP applications for phishing detection. They identified TF-IDF and word embeddings as prevalent techniques for feature extraction, with support vector machines (SVM) and neural networks frequently utilized for classification. The study also noted that datasets such as the Nazario phishing corpus are commonly employed in benchmarking. A key insight was the lack of studies focusing on phishing detection in Arabic language texts, highlighting opportunities for future research [2].

A Novel Approach for Spam Detection Using AMALS Models Ruchi Agarwal et al. introduced an innovative spam detection method employing the AMALS (Approximations with Modifying Alternating Least Squares) framework. This approach tackled data sparsity by using probabilistic models and gradient descent techniques. The research reported an accuracy improvement of 98% compared to traditional TF-IDF approaches, demonstrating the efficiency of combining statistical and ML-based techniques. The study further explored the utility of machine learning models, such as Naïve Bayes and SVM, in handling big data environments for spam detection [3].

State-of-the-Art Methods in Email Spam Filtering McMahan et al. reviewed email spam detection systems, comparing ML-based classifiers like multilayer perceptrons, Naïve Bayes, and SVM against traditional rule-based systems. The study underscored the importance of feature selection techniques like N-gram analysis in enhancing classifier accuracy. Challenges such as linguistic complexity and concept drift were highlighted, emphasizing the need for adaptive systems in dynamic spam environments [4].

Security Challenges in Phishing Email Detection Bhuiyan et al. provided a comprehensive analysis of contemporary spam filtering technologies, particularly in IoT contexts. The research examined the economic and ethical concerns surrounding spam emails and presented various ML-based solutions, with a focus on combining Naïve Bayes and SVM for higher precision. The study concluded that while traditional ML algorithms provide robust solutions, advancements in deep learning offer promising improvements in handling

unstructured data and multimedia content [5].

BENEFITS

Spam and phishing email detection offers numerous benefits to organizations and individuals. The primary advantages include:

Enhanced Cybersecurity: By detecting and filtering spam and phishing emails, organizations can protect sensitive data from unauthorized access, thereby reducing the risk of cyberattacks and financial fraud.

Time Efficiency: Automated systems save significant time for users by filtering irrelevant and harmful emails, allowing them to focus on important communications.

Resource Optimization: Effective spam detection reduces the burden on server resources, preventing issues like slow response times and memory overload.

User Awareness: Advanced spam filters educate users about potential threats by flagging suspicious content, promoting digital literacy and caution.

Scalability: Modern spam detection systems are capable of handling vast email volumes, making them suitable for large-scale enterprises and email service providers.

PRACTICAL EXAMPLE

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("darkgrid")
%matplotlib inline

import string
import nltk
from nltk.corpus import stopwords

from wordcloud import WordCloud
from sklearn.feature_extraction.text import CountVectorizer
from nltk.stem import WordNetLemmatizer

from sklearn.model_selection import train_test_split
from sklearn import metrics

messages = pd.read_csv('spam.csv',encoding = 'latin-1')
messages.head()

messages = messages.drop(labels = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis = 1)
messages.columns = ["label", "message"]

messages['length'] = messages['message'].apply(len)
messages.head()

messages['message'].value_counts().rename_axis(['message']).reset_index(name='counts').head()

messages["label"].value_counts().plot(kind = 'pie',explode=[0, 0.1],figsize=(6, 6),autopct='%1.1f%%',shadow=True)
plt.title("Spam vs Ham")
plt.legend(["Ham", "Spam"])
plt.show()

plt.figure(figsize=(12,6))
messages['length'].plot(bins=100, kind='hist') # with 100 length bins (100 length intervals)
plt.title("Frequency Distribution of Message Length")
plt.xlabel("Length")
plt.ylabel("Frequency")

messages[messages['length'] == 910]['message'].iloc[0]

messages.hist(column='length', by='label', bins=50,figsize=(12,4))

def text_preprocess(mess):
    """
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
    2. Remove all stopwords
    3. Returns a list of the cleaned text
    """
    # Check characters to see if they are in punctuation
    nopunc = [char for char in mess if char not in string.punctuation]

    # Join the characters again to form the string.
    nopunc = ''.join(nopunc)
    nopunc = nopunc.lower()

    # Now just remove any stopwords and non alphabets
    nostop=[word for word in nopunc.split() if word.lower() not in stopwords.words('english') and word.isalpha()]

    return nostop

spam_messages = messages[messages["label"] == "spam"]["message"]
ham_messages = messages[messages["label"] == "ham"]["message"]
print("No of spam messages : ",len(spam_messages))
print("No of ham messages : ",len(ham_messages))

# This may take a while....
spam_words = text_preprocess(spam_messages)

spam_wordcloud = WordCloud(width=600, height=400).generate(' '.join(spam_words))
plt.figure( figsize=(10,8), facecolor='k')
plt.imshow(spam_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

print("Top 10 Spam words are :\n")
print(pd.Series(spam_words).value_counts().head(10))

ham_words = text_preprocess(ham_messages)

ham_wordcloud = WordCloud(width=600,
```

```
height=400).generate(' '.join(ham_words))
plt.figure( figsize=(10,8), facecolor='k')
plt.imshow(ham_wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()

print("Top 10 Ham words are :\n")
print(pd.Series(ham_words).value_counts().head(10))

#Creating the bag of words
vectorizer = CountVectorizer()
bow_transformer = vectorizer.fit(messages['v2']) # Assuming 'v2' contains your text data

print("20 Bag of Words (BOW) Features: \n")
print(vectorizer.get_feature_names_out()[20:40])

print("\nTotal number of vocab words : ", len(vectorizer.vocabulary_))


# fit_transform : Learn the vocabulary dictionary and return term-document matrix.
bow4 = bow_transformer.transform([message4])
print(bow4)
print(bow4.shape)

print(vectorizer.get_feature_names_out()[5945])


messages_bow = bow_transformer.transform(messages['v2'])


from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer().fit(messages_bow)

tfidf4 = tfidf_transformer.transform(bow4)
print(tfidf4)

tfidf4 = tfidf_transformer.transform(bow4)
print(tfidf4)

feature_names = bow_transformer.get_feature_names_out()

print(feature_names[5945])
print(feature_names[3141])


print(tfidf_transformer.idf_[bow_transformer.vocabulary_['say']])

messages_tfidf = tfidf_transformer.transform(messages_bow)
print(messages_tfidf.shape)

from sklearn.feature_extraction.text import TfidfVectorizer

vec = TfidfVectorizer(encoding="latin-1", strip_accents="unicode", stop_words="english")
features = vec.fit_transform(messages['v2']) # Assuming 'v2' contains the text data
print(features.shape)
print(len(vec.vocabulary_))

from sklearn.model_selection import train_test_split

msg_train, msg_test, label_train, label_test = \

    train_test_split(features, messages['v1'], test_size=0.2, random_state=42)


print("train dataset features size : ",msg_train.shape)
print("train dataset label size", label_train.shape)

print("\n")

print("test dataset features size", msg_test.shape)
print("test dataset lable size", label_test.shape)

from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()
spam_detect_model = clf.fit(msg_train, label_train)

from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Assuming you have defined and trained your classifier
classifier = SVC()
classifier.fit(msg_train, label_train)

# Obtain predictions for training data
predict_train = classifier.predict(msg_train)

# Print classification report
print("Classification Report:\n", classification_report(label_train, predict_train))

# Print confusion matrix
print("\nConfusion Matrix:\n", confusion_matrix(label_train, predict_train))

# Print accuracy score
accuracy_train = accuracy_score(label_train, predict_train)
print("\nAccuracy of Train dataset: {0:.3f}".format(accuracy_train))


print(messages.head())


# Drop unnecessary columns if they exist
columns_to_drop = ["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"]
columns_to_drop = [col for col in columns_to_drop if col in messages.columns]  # Filter out non-existing columns
if columns_to_drop:
    messages = messages.drop(columns=columns_to_drop)

# Rename the remaining columns
messages.columns = ["label", "message"]

# Print the updated column names
print(messages.columns)
```

```
from sklearn.model_selection import
train_test_split from sklearn.naive_bayes import
MultinomialNB from sklearn import metrics

# Split the data into training and testing sets
msg_train, msg_test, label_train, label_test =
train_test_split(features, messages['label'], test_size=0.2,
random_state=42)

# Create and train the Naive Bayes classifier
clf = MultinomialNB()
spam_detect_model = clf.fit(msg_train, label_train)

# Make predictions on the training set
predict_train = spam_detect_model.predict(msg_train)

# Evaluate the model on the training set
print("Classification Report on Training Set:\n",
metrics.classification_report(label_train, predict_train))
print("\nConfusion Matrix on Training Set:\n",
metrics.confusion_matrix(label_train, predict_train))
print("\nAccuracy on Training Set:
{:.3f}".format(metrics.accuracy_score(label_train,
predict_train)))

# Make predictions on the testing set
predict_test = spam_detect_model.predict(msg_test)

# Evaluate the model on the testing set
print("\nClassification Report on Testing Set:\n",
metrics.classification_report(label_test, predict_test))
print("\nConfusion Matrix on Testing Set:\n",
metrics.confusion_matrix(label_test, predict_test))
print("\nAccuracy on Testing Set:
{:.3f}".format(metrics.accuracy_score(label_test,
predict_test)))


print(metrics.classification_report(label_test,
label_predictions))
print(metrics.confusion_matrix(label_test,  label_predictions))

print("Accuracy of the model:
{0:0.3f}".format(metrics.accuracy_score(label_test,
predict_test)))
```

CHALLENGES

Data Scarcity

One of the most significant challenges in spam detection is the scarcity of high-quality, diverse datasets. Many models designed for spam detection rely on large datasets to train machine learning algorithms effectively. However, the lack of diversity in these datasets can limit their ability to generalize to new types of spam or phishing attempts. For instance, if a model is trained predominantly on a dataset containing spam emails in English, it may struggle to detect spam in other languages or dialects. The absence of labeled data from diverse sources—ranging from different cultures, languages, and types of communication—limits the development of more universal detection models.

Evolving Threats

Cybercriminals and spammers continuously adapt their tactics and methods to bypass spam filters. These evolving threats make it challenging for detection systems to maintain accuracy and relevance over time. The continuous emergence of new techniques, such as disguised phishing attempts, obfuscation of malicious links, and social engineering tactics, means that spam filters must be updated frequently. Failure to adapt to these evolving tactics can result in decreased detection rates and, consequently, a higher risk of successful attacks on users.

Multilingual Complexity

Multilingual spam poses another significant challenge. While spam detection systems are becoming more advanced in detecting emails in English, they still struggle with languages that are less studied in computational linguistics. Many spam detection models, particularly those based on Natural Language Processing (NLP), have been primarily designed for English-language content. This leaves them vulnerable to spam emails written in other languages, particularly in regions where the spam content might be localized to a specific culture or language. Developing models that can handle multilingual spam detection with high accuracy remains a substantial challenge.

Concept Drift

The phenomenon of "concept drift" refers to the changing nature of spam over time. As spammers change their tactics, the characteristics of what constitutes a spam email evolve as well. For instance, an email that was once considered spam due to certain features, such as certain phrases or word patterns, may no longer be classified as spam after spammers adapt. Concept drift poses a challenge for traditional spam detection models, as they may become less accurate over time unless they are updated to reflect new trends. This requires the development of adaptive algorithms that can adjust to these changes in the spam landscape.

False Positives

Another significant challenge in spam detection is the issue of false positives—legitimate emails mistakenly classified as spam. Overly aggressive filtering algorithms can misclassify important emails, which undermines user trust in the system. A user might miss critical business correspondence, personal emails, or even transactional notifications if a system incorrectly marks them as spam. Balancing the detection of malicious emails with the need to avoid false positives is a delicate task, as too many false positives can lead to user frustration and decreased confidence in the system's reliability.

DATA INTEGRATION AND ACCURACY

Effective spam detection relies not only on sophisticated algorithms but also on the ability to integrate and process data from diverse sources accurately. The integration of data ensures a comprehensive approach to spam detection, but it also introduces several technical challenges.

Heterogeneous Data Sources

Email data comes in a variety of formats, and spam detection systems must be capable of handling this diversity. Emails can vary greatly in terms of structure, such as plain text, HTML, or even attachments. Each format may contain different types of data, including the message body, metadata (sender information, timestamp), subject line, and attachments. This heterogeneity makes preprocessing and feature extraction tasks more complicated, as the system must be designed to effectively handle all types of content while maintaining accuracy in detection.

Accuracy in Feature Selection

Feature selection is a crucial step in developing an effective spam detection model. The quality of the features chosen—such as specific keywords, metadata, or behavioral patterns—directly affects the performance of the model. Selecting the wrong features can lead to lower accuracy, as the model may fail to capture the most important signals indicative of spam. On the other hand, incorporating too many irrelevant features can increase the complexity of the model, making it harder to train and prone to overfitting. Identifying and selecting the most relevant features for spam classification is an ongoing challenge in the field of NLP and machine learning.

Real-Time Processing

Spam detection systems need to operate in real-time without sacrificing detection accuracy. This is particularly important for user-facing applications, such as email clients, where emails must be processed quickly and accurately to avoid delays in communication. Achieving real-time processing in spam detection systems requires optimizing algorithms and architectures to ensure efficient computation while maintaining high levels of accuracy. Additionally, real-time systems must be adaptive to handle new and emerging types of spam efficiently.

CONCLUSION

The field of spam and phishing email detection has made significant strides due to the integration of machine learning (ML) and natural language processing (NLP) techniques. Advanced models, such as BERT (Bidirectional Encoder Representations from Transformers) and AMALS (Adaptive Machine Learning Spam detection), have set high benchmarks for accuracy and efficiency, improving the overall reliability and safety of email communication.However, despite these advancements, several challenges persist, including data scarcity, evolving threats, multilingual complexity, concept drift, and the risk of false positives. These challenges highlight the need for continued innovation and research in the domain of spam detection. Future work should focus on developing adaptive, context-aware models capable of addressing the limitations of current approaches. Furthermore, attention should be given to improving data diversity, feature selection methods, and real-time processing capabilities to enhance the robustness and accuracy of spam detection systems across different environments and languages.

By focusing on these areas, researchers and practitioners can build more resilient spam detection systems that offer greater security and usability, protecting users from malicious threats while preserving the integrity of legitimate communications.

REFERENCES

[1] AbdulNabi, I., & Yaseen, Q. (2021). Spam Email Detection Using Deep Learning Techniques. Procedia Computer Science, 184, 853-858. doi:10.1016/j.procs.2021.03.107

[2] Salloum, S., Gaber, T., Vadera, S., & Shaalan, K. (2022). A Systematic Literature Review on Phishing Email Detection Using Natural Language Processing Techniques. IEEE Access, 10, 65703-65730. doi:10.1109/ACCESS.2022.3183083

[3] Agarwal, R., Dhoot, A., Kant, S., et al. (2024). A Novel Approach for Spam Detection Using Natural Language Processing With AMALS Models. IEEE Access, 12, 124298- 124302. doi:10.1109/ACCESS.2024.3391023

[4] McMahan, B., et al. (2024). Comparative Survey of Email Spam Detection Techniques. Journal of Machine Learning Research, 18(4), 352-374

[5] Saleh, H., et al. (2023). Smart Spam Email Detection: A Comprehensive Review. International Journal of Computer Applications, 20(6), 245-260

[6] Sun, M., et al. (2022). Learning-Based Methods for Spam Filtering: A Survey. IEEE Transactions on Information Security, 32(1), 120-145

[7] Bhuiyan, K., et al. (2023). Advanced Techniques in Email Spam Filtering. Procedia Engineering, 25, 1101-1115

[8] Ferrag, M., et al. (2024). Deep Learning Techniques for Spam Detection and Intrusion Systems. Computers & Security, 45, 98-120

[9] Vyas, R., et al. (2022). Supervised Machine Learning for Spam Email Filtering. Journal of AI and Cybersecurity, 14(7), 320-340

[10] Hassanpur, M., & Egozi, A. (2020). Email Spam Detection Using Word2Vec and Deep Neural Networks. Advances in Computing Research, 15(3), 245-258

[11] Soni, R. (2021). THEMIS: RCNN-Based Phishing Detection Model. Proceedings of the International Conference on Security, 38(5), 245-268

[12] Seth, A., et al. (2021). Hybrid CNN Models for Spam Classification. Information Processing Journal, 12(4), 322-335

[13] Ezpeleta, J., et al. (2020). Bayesian Filtering and Spam Detection. Procedia Information Sciences, 34(9), 1001-1020

[14] Awad, N., et al. (2020). Comparative Study of Spam Detection Systems Using ML. Data Science Review, 13(3), 95- 110

[15] Saab, H., et al. (2019). Machine Learning in Spam Detection: A Practical Approach. Neural Computing and Applications, 17(8), 1101-1130