Preventing Phishing Attacks using Deep Learning

Justin Joy
Computer Science and Mathematics
Adelphi University, New York
Email: justinjoy@mail.adelphi.edu

***Abstract* – In a world of advancing technologies and many organizations relying on such technologies, phishing attacks have emerged to become a prevalent threat to cyber security. Although phishing attacks can occur in many ways, one common method is by sending deceptive emails containing malicious websites to unlawfully obtain a user's personal information and data such as usernames, passwords, and phone numbers. This increasing threat to cybersecurity requires an advanced method to counteract such sophisticated attacks. This paper proposes a method to scan and monitor inboxes to effectively detect phishing emails and URLs using different neural network-based AI (artificial intelligence) models. By comparing different models and even combining Natural Language Processing and machine learning algorithms, the most effective approach can be used for the program. This project will not only compare the efficiency of different AI models but will also return a detailed report on malicious emails for the user to analyze, allowing the user to make informed decisions when interacting with them.**

Keywords – Machine Learning, Artificial Intelligence, Cybersecurity, Deep Learning, Natural Language Processing, Phishing Detection, Long Short-Term Memory, Transformer Model, API

I. INTRODUCTION

The expected outcome of this study is to have an effective solution to mitigate phishing attacks, as well as improve the accuracy and speed of phishing detection, so that organizations and individual users can be safer on the internet. This study will investigate the effectiveness of a Long Short-Term Memory (LSTM) model, a Transformer Model, and OpenAI's GPT-3 and GPT-4 models. LSTM has been used for many use cases including Natural Language Processing, speech recognition, and time series prediction (ex. weather forecasting and stock market predictions). Phishing emails may use a specific language and an LSTM model can be used to identify such patterns. As transformer models were introduced, NLP was revolutionized as these models could focus on different parts of the input data at different times, allowing the model to understand the context and relationships between data better. OpenAI's GPT-3 and GPT-4 models, 2 highly advanced Large Language Models (LLM), may also prove useful to test. OpenAI has been actively developing its API features, allowing users to now fine-tune (add datasets to) the models. Therefore, all models in this study can be trained on the same dataset.

TABLE I
A LIST OF KEY ACRONYMS

Acronyms	Description
AI	Artificial Intelligence
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
BERT	Bidirectional Encoder Representations from Transformers
RoBERTa	Robustly Optimized BERT Approach
DistilBERT	Distilled BERT (40% Less parameters than BERT)
MLM	Masked Language Modeling
NSP	Next Sentence Prediction
GPT	Generative Pretrained Transformer
LLM	Large Language Models
API	Application Programming Interface
CPU	Central Processing Unit
GPU	Graphics Processing Unit
CUDA	Compute Unified Device Architecture
CSV	Comma Separated Values
IMAP	Internet Message Access Protocol
SMTP	Simple Mail Transfer Protocol
GUI	Graphical User Interface

II. LITERITURE REVIEW

A. Long Short-Term Memory (LSTM):

The LSTM Model is a special type of Recurrent Neural Network (RNN). RNN is a deep learning model that uses sequence data but also remembers the prior element of the sequence [8]. The main difference, however, is that LSTM can hold its information for a long time. It can remember the output of each node for a longer period to produce an output for the next node.

B. Transformer Model:

A Transformer Model is another deep learning model that was introduced in 2017 and is now mostly associated with NLP [9]. NLP is a machine learning approach giving computers the ability to interpret and comprehend human language. Unlike LSTM, which relies on sequential data processing, transformer models have "self-attention [10] or intra-attention mechanisms" [1]. This would allow the model to process the sequences of data simultaneously. Examples of transformer models are BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pretrained Transformer).

C. Bidirectional Encoder Representations from Transformers:

The BERT Model was developed by Google and released under an open-source license in 2018. A BERT model is an "encoder-only" type of transformer. BERT is used for a wide array of NLP tasks and was a dramatic improvement over older models like LSTM. Whereas prior models like LSTM are trained unidirectionally, BERT is trained bidirectionally (left and right) to understand the context of words based on the surrounding words. Models like this can help understand the subtle nuances of the human language. BERT is also different from LSTM because BERT is first pre-trained on a large dataset of text with unsupervised learning by using Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) [2]. MLM is a technique where the model must predict the original word of randomly masked words in a sentence. NSP is when the model tries to predict the next logical sentence. Then BERT can be fine-tuned on specific data.

D. Generative Pretrained Transformer:

The GPT model was developed by OpenAI and used the transformer model architecture. Similarly to BERT, GPT is also pre-trained on a large dataset of text with unsupervised learning, however, GPT is trained using language modeling. Language Modeling is when the model must predict the next word of a sentence based on the previous words. This is one reason why GPT is known mostly for generating text [3], while BERT is more focused on understanding text. GPT-4 was also fine-tuned with additional data using reinforced learning from human feedback.

E. *Similar studies on AI-based phishing detection*

Another study has been made similar to this current study [5] by Suhaima Jamal, Hayden Wimmer, and Iqbal H. Sarker. Their study was using only transformer models (BERT, RoBERTa, DistilBERT) to detect phishing, spam, and ham emails. Their study also consisted of a smaller dataset compared to this study (14,475 entries vs 18,211 entries). However, this study focuses on only phishing emails and will consider spam emails as phishing and ham emails as safe. Their dataset has some spam and ham emails as well. More specifically, this study aims to see if AI can detect the subtle indications of a phishing attempt and clearly detect safe (company marketing, official receipts, official password requests, etc.) emails. Modern day phishing emails are often indistinguishable from real emails. The goal of this study is to create a program that can be beneficial to companies and organizations (and their employees) that are susceptible to such emails – to assist them in deciding for sure if an email is phishing or not.

III. TECHNOLOGIES

This section describes the technologies used for this project. The program will be written in Python using VS Code on a Windows 11 PC. The LSTM model will be trained using an AMD Ryzen 7 5700G CPU. The BERT model will be trained with a Google Colab (Linux-based) notebook using a 40 gigabyte NVIDIA A100 GPU.

A. TensorFlow

TensorFlow is an open-source library developed by the Google Brain Team. This library has many deep learning models and tools that support AI training. TensorFlow was used to create the LSTM model and train it. TensorFlow also supports the Keras API which simplifies the process of neural network building. Compared to building a model in PyTorch, a model with Keras can be trained and evaluated in fewer lines of code.

B. Hugging Face (Transformers)

Hugging Face, a French American company based in New York City, develops many tools related machine learning. Their platform consists of an open-source community of machine learning professionals and enthusiasts collaborating on models, datasets, and other AI related applications. For this project, their Transformers library, which includes a wide variety of pre-trained models designed for NLP tasks, will be used along with TensorFlow to train and deploy the project model.

C. Sci-Kit Learning

Sci-Kit Learning (sklearn) is a machine learning library developed by David Cournapeau. It includes a wide array of tools for many machine learning tasks. For this project, its data preprocessing tools like Label Encoding (converting categorical labels into numerical values) or Vectorization (converting tokenized text into numerical values) will be used.

D. Pandas

Pandas is a data manipulation and analytics library originally developed by Wes McKinney. For this project, pandas will be used for data loading and creating data frames from the loaded CSV files (the datasets).

E. Natural Language Toolkit

The Natural Language Toolkit, or NLTK, developed by Team NLTK in 2001, is a library of tools that are used to work with human language. It has many lexical tools including wordnet, a lexical database of English words developed at Princeton University. This lexical database contains English words grouped together in "synsets" or groups of synonyms. Wordnet is a also tool used for data augmentation in machine learning.

F. StreamLit

StreamLit is an open-source library developed by a community of creators. StreamLit is used to develop web apps for python scripts. The main use of StreamLit for this case will be to create a GUI for the program. The GUI will allow the user to see a visual representation of the data, send a report to the user's email on command, or for development and testing – manually enter the email and get results.

G. PyTorch

PyTorch is a machine learning library developed by Meta AI (formerly, Facebook AI Research) and is used to build AI applications like computer vision and natural language processing. This library will be used similarly to how TensorFlow will be used, which is to build the AI model, in this case, the Transformer model. This decision was made since HuggingFace has more information and resources available

when working with PyTorch and it is easier to set up CUDA support for GPU training.

H. APIVoid

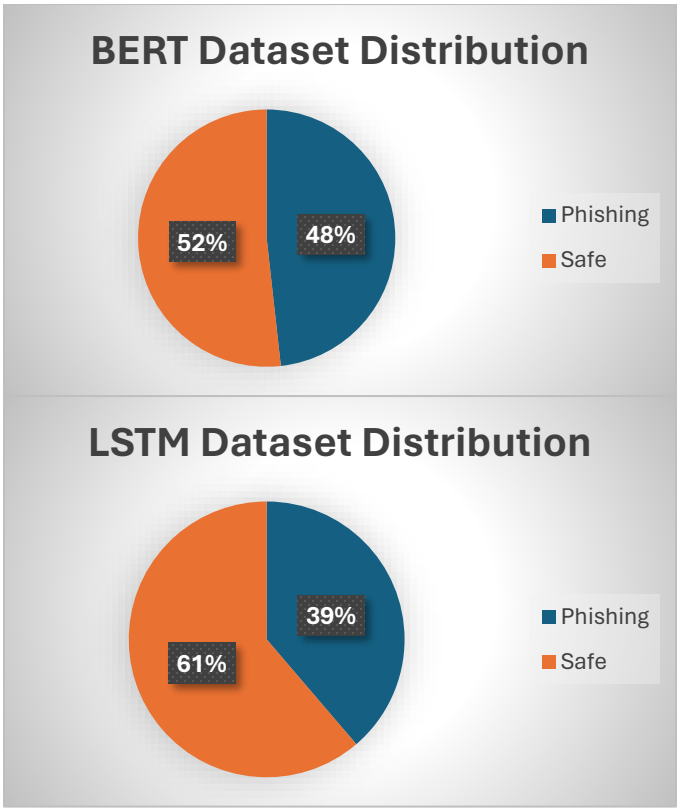
This is a service that provides email reputation and URL reputation data. The service allows a user to make API requests that will return relevant data about the sender's email address and all the URLs that were scraped from the email. Both will also return a reputation score.

III. METHODOLOGY

A. Datasets

The datasets for this project were sourced from Kaggle, Hugging Face, Google Dataset Archive, and UCI Machine Learning Repository. The datasets will be loaded in CSV format using Pandas. Along with the datasets sourced from online, some custom additional data will be added to further increase accuracy. The LSTM dataset has a total of 18,211 entries– 7,055 phishing and 11,156 safe emails. The BERT dataset has much less, 199 entries – 96 phishing emails and 103 safe emails. The reason for this choice was due to the more successful results seen by the BERT model with less data finetuned to it.

TABLE II
LSTM and BERT DATASET DISTRIBUTIONS



The LSTM and Transformer models will be trained on the dataset by following these procedures:

1. Pre-processing: The dataset needs to be readable for the model. To accomplish this, the data must be tokenized, so that a text can be a numerical vector that can be interpreted by the model.
2. Data Augmentation: This method will be used primarily to prevent the model from overfitting and give a little bit more variation. For this study, synonym replacement will be the choice of data augmentation. By using the 'wordnet' lexical database of the Natural Language Toolkit library, some words (any random 2 words per sentence) will be replaced, and a new entry will be created, not only adding some variation to the dataset, but also increasing the size of the dataset.
3. Data Splitting: The data will be split into 2 sets – training and validation/test sets. The training set is used to train the model and the test set is used to assess the model's performance on data that it has not seen before.

The OpenAI GPT-3 and GPT-4 models are very large deep learning models, pre-trained on an extensive set of data. The API will allow users to "fine-tune" or upload datasets for the model to train on. For this study, the GPT models will rely mostly on prompt engineering. By providing the model concise contextual information and instructional prompts, the AI will return more targeted and relevant outputs.

B. Training and Evaluation (LSTM)

The data for this model will be split into 80% for training and 20% for testing. The training set is used to teach the model. The testing set is used after the training to evaluate how well the model can generalize unseen data. The training of an LSTM model involves layers that process the data fed to the model. The LSTM model structure for this project is as follows:

- Embedding Layer – will take in an input of word indices and turn them into dense fixed size vectors.
- First LSTM Layer – this layer processes the input sequence. This layer will pass the sequence of output vectors to the next layer. This is because there is a second LSTM layer expecting a sequence input.
- First Dropout Layer – This layer will deactivate or as the name says, "drop" a percentage of neurons at random. This decreases the model's dependence on a few specific weights, which will then increase the model's ability to generalize unseen data.
- Second LSTM Layer – this layer processes the previous output sequence. This is the last LSTM output sequence, so it reduces everything to a single vector.
- First Dense Layer – This layer takes the single vector and transforms it with the ReLU activation function, allowing the model to learn non-linear relationships.
- Second Dropout layer – just like the first dropout layer, this "drops" a percentage of neurons at random, to prevent overfitting – since the first dense layer has just made the model more complex.
- Final Dense Layer – This makes the output of this layer into a single unit with the Sigmoid activation function and gives a value between 0 (phishing) to 1 (safe).

C. Training and Evaluation (BERT)

The data for this model will be split into 90% for training and 10% for testing. This decision was made due to several factors including the small size of the dataset and the BERT

model's complexity. Training the full BERT model with 18,000 entries would be very resource intensive and expensive, so it will be trained only once with the full dataset as well as a benchmark for the smaller dataset.

D. Other Analysis and Report Generation

APIVoid offers URL and Email Reputation analysis APIs. The URL API will process the URL and check through a database of phishing URLs, like OpenPhish.com and will check through blacklisted emails to give a reputation score for both the URL and emails. For this report, only specific results of the URL and email analysis will be included, for example:

- *URL:*
  - IP
  - Risk Score
  - SSL Certificate Validity
- *Email:*
  - Username
  - Domain
  - Suspicious Username?
  - Suspicious Domain?
  - Is Spoof-able?
  - Recommended to Block?
  - Score

For the user to view these results and make an educated decision, the report will include:

- Date and Time the email was sent.
- Email Subject and Content
- OpenAI Phishing Detection Result
- LSTM Phishing Detection Result
- BERT Phishing Detection Result
- Danger Word Detection
- Unsubscribe Option Detection
- URL Reputation Analysis Results
- Email Reputation Analysis Results

Finally, a detailed report is generated and will be sent from report@antiphishmachine.com, which is sent using the SMTP protocol. The addition of a report will allow the user to make an educated decision based on the AI's predictions and the analysis results. Sometimes, the AI can be wrong. The use of three AI models can not only benefit as a benchmark for each, but also to help decide if an email is safe or a phishing attempt based on the predictions of all three models. For example, if a phishing email is sent and the OpenAI GPT predicts that the email is safe, however both the LSTM and BERT models predict that the email is phishing, the user can assume that the email is a phishing attempt based on the majority of predictions.

E. Email Monitoring and GUI

The user can have their email monitored by connecting to their email's IMAP server. This will allow the program to access the user's email. By using the IMAPClient's IDLE feature, the email server will receive any latest email as soon as it is received. This will allow for continuous monitoring without the user having to manually request the latest email. By using StreamLit, a GUI was also implemented for manually requesting the latest email and manually entering text (emails) for the AI to analyze. The StreamLit GUI is mostly for presentation and debugging purposes, where ideally a business would have the program continuously monitor the emails.

IV. EXPERIMENTATION AND RESULTS

A. Training and Evaluation Results (LSTM)

The training of the LSTM model will be done within 7 epochs, with no early stopping. When training the LSTM model using the large dataset, the model's training accuracy started at 61.21% and peaked at 99.53%, averaging 88.64% accuracy. The model's testing accuracy began with 61.07% accuracy and peaked at 97.61% accuracy, averaging 90.81% accuracy.

The loss is highest at the first epoch, at 0.64 and decreases to 0.02 by the seventh epoch. The testing loss starts at 0.61 and decreases to 0.11 by the seventh epoch. Although the accuracy dips and loss creeps up at epoch 6, it is only slight and quickly becomes stable at epoch 7. This shows that there are most likely no signs of overfitting.

The model trained on the larger dataset showed a significant improvement compared to the model trained on smaller dataset. With the smaller dataset, the model is generalizing data poorly compared to the larger dataset. The loss was an average 0.63, which is relatively high, showing that the loss (for the smaller data set) had not decreased as much.

TABLE III
LSTM TRAINING LOSS AND ACCURACY

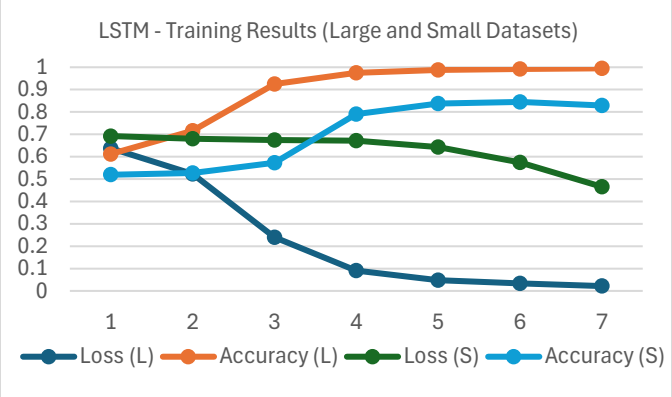


TABLE IV
LSTM TRAINING LOSS AND ACCURACY PRECISE VALUES

LSTM Training Results (Large and Small Datasets)				
	Loss (L)	Accuracy (L)	Loss (S)	Accuracy (S)
1	0.6379	61.21%	0.6925	51.94%
2	0.5225	71.58%	0.68	52.71%
3	0.2396	92.49%	0.6751	57.36%
4	0.0901	97.56%	0.6722	79.07%
5	0.0473	98.86%	0.6428	83.72%
6	0.0338	99.22%	0.5746	84.50%
7	0.0216	99.53%	0.4657	82.95%
avg.	0.227543	88.64%	0.62899	70.32%

TABLE V
LSTM TESTING LOSS AND ACCURACY

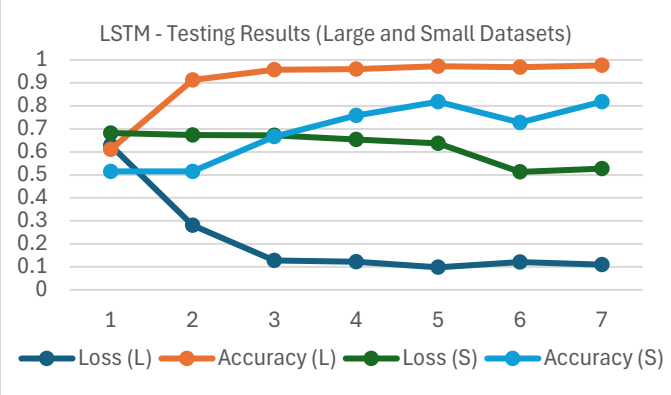


TABLE VI
LSTM TESTING LOSS AND ACCURACY PRECISE VALUES

LSTM Testing Results (Large and Small Datasets)				
	Loss (L)	Accuracy (L)	Loss (S)	Accuracy (S)
1	0.6288	61.07%	0.6822	51.52%
2	0.2807	91.26%	0.6731	51.52%
3	0.1274	95.66%	0.6723	66.67%
4	0.1216	96.02%	0.6532	75.76%
5	0.098	97.23%	0.6378	81.82%
6	0.1212	96.81%	0.5128	72.73%
7	0.1098	97.61%	0.5267	81.82%
avg.	0.2125	90.81%	0.62259	68.83%

B. Training and Evaluation Results (BERT)

The training of the BERT model will be done in 5 epochs, with no early stopping. The training accuracy started off at 70.09% accuracy and peaked at 97.78% accuracy, averaging 89.35% accuracy. The testing accuracy began at 77.5% accuracy and peaked at 97.5% accuracy, averaging 88.5% accuracy. The loss was steadily decreasing as well, starting at 0.58 and ending at 0.09 during training. During testing, the loss started at 0.47 and dropped to 0.03 by the last epoch.

When trained with the larger data, the model was steady and learning effectively as both the loss and accuracy started high, the loss was decreasing, and the accuracy was increasing. With the smaller dataset, the model also learned it effectively, however, with the help of a higher dropout rate. The HuggingFace BERT model already implements a dropout of 10% probability, however it was increased to 30% to prevent overfitting [7]. Although a model like BERT should perform better with more data, like the large dataset (which also had a higher accuracy average), it actually performed poorly with real life emails. Rather, the model trained on the smaller dataset made more correct predictions.

TABLE VII
BERT TRAINING LOSS AND ACCURACY

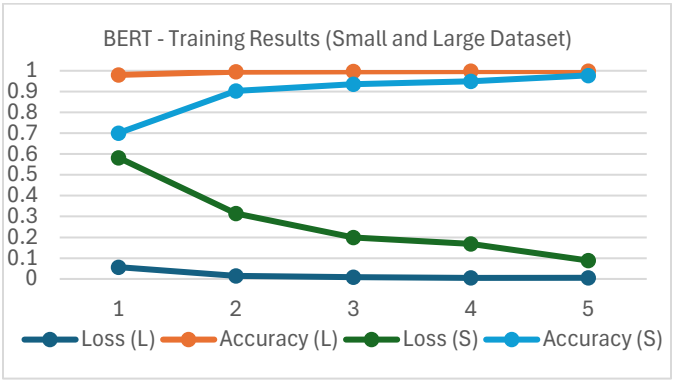


TABLE VIII
BERT TRAINING LOSS AND ACCURACY PRECISE VALUES

BERT Training Results (Large and Small Datasets)				
	Loss (L)	Accuracy (L)	Loss (S)	Accuracy (S)
1	0.0568	98.02%	0.5826	70.09%
2	0.0145	99.54%	0.3138	90.28%
3	0.008	99.76%	0.1988	93.61%
4	0.005	99.83%	0.169	95.00%
5	0.0058	99.81%	0.0886	97.78%
avg.	0.01802	99.39%	0.27056	89.35%

TABLE IX
BERT TESTING LOSS AND ACCURACY

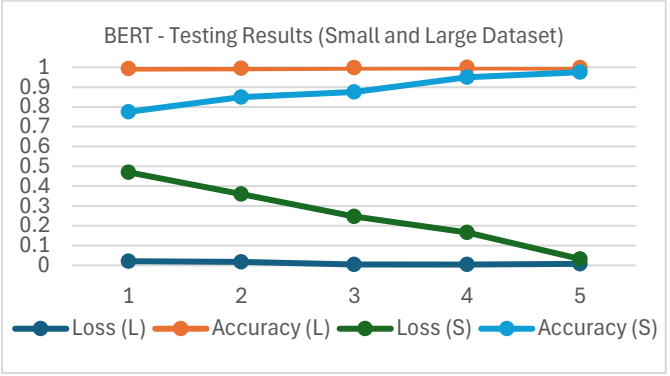


TABLE X
BERT TESTING LOSS AND ACCURACY PRECISE VALUES

BERT Testing Results (Large and Small Datasets)				
	Loss (L)	Accuracy (L)	Loss (S)	Accuracy (S)
1	0.0211	99.31%	0.4702	77.50%
2	0.017	99.45%	0.3593	85.00%
3	0.0055	99.75%	0.2469	87.50%
4	0.0044	99.89%	0.166	95.00%
5	0.0081	99.84%	0.0328	97.50%
avg.	0.01122	99.65%	0.25504	88.50%

The loss ends up creeping up after the fourth epoch and is generally flat all throughout. Also, the accuracy plateaus at 0.85. This is considered to show signs of overfitting.

C. OpenAI API – GPT

OpenAI's Assistant API was used to obtain a GPT "benchmark" for the other two models as well as give valuable information for the user's report. The GPT was usually the tiebreaker for most contradictions between the LSTM and BERT models. The API works by creating an Assistant, then creating a new thread of the assistant for every new request, creating a message, creating a run, and retrieving the message back from the AI.

D. Real-World Application and Testing

When testing with real emails (both phishing and safe), both models performed reasonably well. Although the BERT model seemed to be consistent with overfitting, the BERT model outperformed the LSTM model. The GPT model allowed for a tiebreaker if the LSTM and BERT models gave two different results. With the tiebreaker (or majority vote of models), a correct prediction was made. Another observation was that although the BERT model that was trained on a large dataset showed no significant signs of overfitting, it still performed poorly compared to the model trained with less data. This could be due either to the model's pretrained nature and its concept of transfer learning [6] or simply the quality of the datasets. Most of the emails in the large dataset were spam/ham emails, with a few marketing and company-based emails. Whereas most emails today are rather filled with hyperlinks and images which may have confused the model.

The BERT Model had a validation rate of 90% (correctly predicting 18 of 20 emails). The LSTM model had a validation rate of 80% (correctly predicting 16 of 20 emails). The GPT model had a validation rate of 95% (correctly predicting 19 of 20 emails). Overall, when calculating the majority vote of each email, the models achieved a validation rate of 95% (correctly predicting 19 of 20 emails).

IV. CONCLUSION

A. Summary of Findings

From the data and results that we can view from this study, the LSTM and BERT models showed successful predictions, and it was capable of detecting the tones in marketing/commercial emails vs. phishing/fraud emails. Even when types of emails had a similar selling point or context, the AI models were able to detect the subtle patterns to differentiate phishing emails from marketing ones.

We can see that using AI to monitor email inboxes can be beneficial to users and can help prevent phishing attacks. Users will be more aware of their emails, and they can analyze the contents of their emails more easily and effectively. Along with the regular analysis, AI can help detect unusual tones, which would help reduce human error and the potential for oversight.

B. Future Research and Development

In terms of futureproofing or updating the model, with a more developed dataset (especially with more phishing emails) the AI can learn more patterns that would potentially help increase the accuracy of the models. Many organizations already have options in their enterprise emails to "Report Phishing". When a user reports a new phishing email, these emails can be added to a database to periodically train the models on new data. This would be extremely useful for the BERT model as its data set is currently relatively small for a highly complex model.

V. REFERENCES

[1] A. Vaswani *et al.*, 'Attention is all you need', *Advances in neural information processing systems*, vol. 30, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810. 04805, 2018.*

[3] OpenAI *et al.*, 'GPT-4 Technical Report', *arXiv [cs.CL]*. 2024.

[4] T. B. Brown *et al.*, 'Language Models are Few-Shot Learners', *arXiv [cs.CL]*. 2020.

[5] S. Jamal and H. Wimmer, 'An Improved Transformer-based Model for Detecting Phishing, Spam, and Ham: A Large Language Model Approach', *arXiv [cs.CL]*. 2023.

[6] N. Houlsby *et al.*, 'Parameter-Efficient Transfer Learning for NLP', *arXiv [cs.LG]*. 2019.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting', *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[8] "What are recurrent neural networks?," IBM, https://www.ibm.com/topics/recurrent-neural-networks (accessed May 2, 2024).

[9] "What is a transformer model?," IBM, https://www.ibm.com/topics/transformer-model (accessed May 2, 2024).

[10] R. Merritt, "What is a transformer model?," NVIDIA Blog, https://blogs.nvidia.com/blog/what-is-a-transformer-model/ (accessed May 2, 2024).