Email Classification Using Deep Learning: Spam, Scam, and Safe

Atharva Sardar*
atharvasardar02@vt.edu
atharva.sardar02@gmail.com
Virginia Tech
Falls Church, Virginia, USA

Abstract

This project learns and explores the use of deep learning models for classifying emails into spam, scam, and safe categories using two datasets from Kaggle: the Spam Mails Dataset[12] and the Phishing Email Detection dataset[4]. The main objective of this project is to develop a model that can accurately identify and categorize emails into 3 categories being spam, scam and safe unlike the traditional projects which focuses on either of them to improve user security. During the course of this project, we try, apply and understand the use of LSTM [7]models and BERT [8] for text classification, giving importance to the challenges in real-world applications. We develop a model which is a combination of LSTM [7]model and BERT [8] model to reduce the loss to 0.0885.

Keywords: Scam, Spam, LSTM[7], BERT[8]

ACM Reference Format:
Atharva Sardar. 2024. Email Classification Using Deep Learning: Spam, Scam, and Safe. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Project '24).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

1.1 Motivation

Email fraud has been a big cybersecurity problem, which has been impacting a large population ranging from individuals and organizations. It poses a serious risk to personal and professional data security. While there are a lot of spam filters which can filter out most of the unwanted messages, but they are not able to detect more sophisticated scams which many a times tend to be targeted. These type of scams are many times designed specifically to force or trick the user to share their personal information which can even contain some sensitive data which can be devastating. With the development in technology and its wide spread use this scams are becoming more and more sophisticated as the day is passing. There have been traditional spam filters which used to work well but to detect planned and professional scams (e.g., phishing emails) requires more advanced models[3]. The use of this models and their accuracy in detection of scams will lead to a higher layer of security. As there are two types of fraudulent emails being spam and scam. Generally spam is referred to the bulk of messages we receive daily. These messages can be annoying but generally they tend to be safe. But scams tell a completely different story as they cause a lot of damage compared to spams. So to combine different types of fraudulent emails (spam and scam) into a single model and provide complete solution for email classification is the way to go forward.So our main aim is to develop a robust model which will not only detect spams and scams but also differentiate them.

The inspiration for this project is from a prior Open-Source Intelligence (OSINT) investigation project which I conducted as a part of my OSINT course. The project dealt with a variety of online scams ranging from refund scam, lottery scam to old age scam. [CITE HERE]. But its main focus was on job scams. The idea to choose this topic was due to the fact that I received many scam job offers which affected my job search. Job scams can be of many types ranging from fake job offers to asking for money for job offers. This investigation was mainly linked to the Open-Source Intelligence aspect of the intelligence and we were not allowed to actively work on catching the scamsters. But this investigation highlighted the urgent need for efficient detection mechanisms for us to be safe in this chaotic job market. This experience taught me that general spam filters are not as good as they should be for an individual to feel safe. They lacked the ability to evolve with the changing landscape. So by learning my lessons from this study this project aims to apply the knowledge I learned during the course of my deep learning course to develop a model which not only classifies spam but also detect scams. This has potential to enhance the cybersecurity across the globe and lead to the building of a safer digtal space.

Project '24, Dec 08, 2024, Falls Church, VA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/XXXXXXX.XXXXXXX

1.2 Difference between Spam and Scam

1.2 Difference between Spam and Scam Spam emails are usually unwanted messages which are sent in bulk and are often used for advertising or promoting products ,content or services. But generally they do not have any intent of harming the individual[13]. They can be very irritating as they tend to make our inbox a mess but are typically not directly harmful. They are generally very generic and not targeted to the individual. They can be tracked using some keywords and the sender. So the existing spam filter try to rely on this patterns and analyze them to stop them to reach the inbox of the user.

Scam emails, on the other hand, are engineered with the intent to do harm to the targeted person They make use of personalized techniques unique to this individual which make them very difficult to identify. The tactics often used include phishing to steal sensitive information or lure victims into financial traps[16] .They lure the victim into providing sensitive information like login credentials, personal detail or financial details. This might be some ways but are not limited to this. They are continuously evolving using social engineering techniques to gain trust of the user with a subtle change in language which is a hard to track and are getting better at their jobs.

Recognizing this difference is very important, because the risks are very different, and countermeasures vary. Spam filtering requires focusing on patterns in bulk messaging, while scam detection, as I have learned from my OSINT project, requires identifying social engineering tactics and subtle manipulative language.

1.3 Importance of Separate Classification

Classifying spam and scam emails differently is very important for keeping ourselves safe in this chaotic internet space and have a effective email security. So we need to develop a model that understands the difference between these two types can not only reduce false positives but also prevent genuine scams from being treated as just normal spam. This clear difference between the two lets us do a variety of actions such as handling high risk scams first by flagging them when we are working on a so called non severe spam. This capability is what sets this project apart from previous work, providing a more comprehensive email filtering system.

2 Related Work and Key Challenges

2.1 Related Work

Traditional spam filters use machine learning models like Naive Bayes and SVMs, while recent projects use deep learning approaches[5] due to the rapid development in the field. Many existing projects focus exclusively on either spam[1] or scam detection[14] but not both, which creates a gap in providing a single solution that can handle both types of threats simultaneously. Research has shown BERT's[8] effectiveness in understanding context, which can come very handy for email classification tasks. As the earlier models had a lot of limitations, such as handling complex language and context, We will be focusing on BERT[8] and LSTM [7].

2.2 Key Challenges

Balancing model complexity and computational resources is a significant challenge, particularly when considering the limited computational power of the system being used. As we increased the complexity of the model, the demands on processing power, memory, and training time increased which in some cases was a huge issue. This can become a very big hurdle when working with systems that have limited hardware supply. This tells that it can also be an issue when trying to deploy a model efficiently in real-time environments. So to counter that we had to strike a right balance by making changes and optimizing the model's architecture so that it remains effective for our classification task while also being lightweight enough to run on the available resources. This can be often achieved by making compromises between different aspects involved mainly being the depth of the model, the number of parameters, and the cost of running the model.

In addition to these challenges, there is another important problem which is of handling data imbalance within the datasets as we are using 2 different datasets. When the dataset used for training has more cases of one type of email (e.g., spam or scam) compared to other , it can lead to biased models that perform poorly on email types with fewer samples. To address this issue, we often make use of the techniques such as oversampling, undersampling, or generating synthetic data to create a more balanced dataset. Another important aspect of this is to ensure that the model generalizes to new and unknown emails.

3 List of Contributions

- Create the datasets combining spam and phishing data sources to simulate a real-world email filtering scenario.
- Preprocess the combined dataset using different preprocessing techniques which make it relevant for our classification model
- Develop a LSTM-based approach[7] to classify emails as spam, scam, or safe.
- Develop a BERT-based[8] model to classify emails as spam, scam, or safe.
- Hypertune the BERT-based[8] model to get the parameters which help our classification model
- Develop a BERT-based[8] model with a layer of LSTM-based model[7] to classify emails as spam, scam, or safe.
- Train and apply this models on created dataset to check their efficiency.

- Provide a detailed comparison of LSTM[7], BERT[8] and other customised models in terms of accuracy, loss, precision, recall, f1 and training time and how better it can be applied in real-world.
- Using this models created, work on differentiating between spam and scam emails, taking into consideration the unique characteristics and challenges of each problem.

4 Data Description and Preprocessing

4.1 Datasets

4.1.1 Phishing Email Detection. Phishing Email Detection [4] dataset specifies the email text body the type of emails which can be used to detect phishing emails by extensive analytics of the email text and classifying those using deep learning. It contains 2 columns Email text and Email type. Total number of samples in this dataset are 18650. It contains 61 percent safe emails and 39 percent phishing emails. It shows that the dataset is not balanced.

4.1.2 Spam Mails Dataset. Spam Mails Dataset[12]gives us emails which are categorised as spam or ham. It is a great source for spam data and to train models to detect spam data. It contains 3 columns being text, label and label number. Label number was just a derived column from label. This dataset contains 5170 samples which are one-third of the Phishing Email Detection Dataset. This dataset is also highly skewed with 71 percent being ham which is the safe email and 29 percent are the spam email.

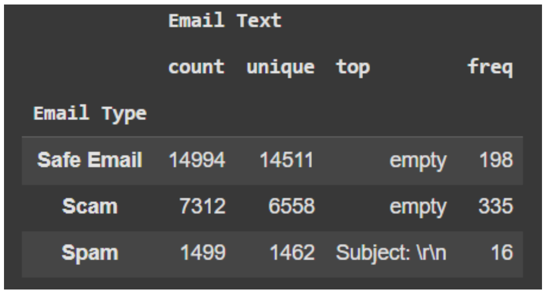
4.2 Preprocessing the Datasets

4.2.1 Combining the Dataset. The first step involves loading the datasets containing spam and phishing emails using the pandas [10] library. The code first of all reads two CSV files containing the 2 relevant datsets: spam-mails-dataset.csv for spam emails and phishingemails.csv for phishing emails. These datasets are imported into separate DataFrame objects, spamdf and phishingdf. Then we do overview of the data structure and ensure we understand the datsets. What is the size of each dataset and how that data sets can be combined with each other.

The next important task is to make the labels in both dataframes same to ensure uniformity and the data can be combined into one. For example, the spam dataset has labels like spam and ham. We then map them to Spam and Safe Email respectively. Similarly for the phishing dataset the labels used are Phishing Email and Safe Email, which are standardized to Scam and Safe Email. This process is essential as it simplifies the classification process as it is a way to reduce the mismatching of the labels during the later part of the project like the training phase. This process of renaming columns leads to consistency so that both dataframes have the same structure, with a column named Email Text for the email content and Email Type for the labels.

After this process, the two dataframes are merged using pd.concat() to create a unified dataset named combineddf. To combine the DataFrames spamdf and phishingdf, we select only the 'Email Text' and 'Email Type' columns and resetting the index with ignoreindex=True so that we get a continuous index in combineddf. This step leads to the creation of datsets where we have the type of text each of the dataset. It is like a real-world scenario where email can be anything from spam, scam or safe email and the email classifiers need to work with diverse email sources.

4.2.2 Balancing the dataset. First of all, to ensure that the dataset is good for further processing, any rows containing missing values are removed. Then the dataset is analyzed to understand if the data is skewed towards one type of email. This can be done by grouping the data by the Email Type column and generating summary statistics for each class, which provides the number of entries and other characteristics of each category i.e. Safe Email, Scam, Spam.



	Email Text			
	count	unique	top	freq
Email Type				
Safe Email	14994	14511	empty	198
Scam	7312	6558	empty	335
Spam	1499	1462	Subject: \r\n	16

Figure 1. Dataset Composition

The dataset needs to be balanced to ensure each class being Safe Email, Spam, Scam is represented equally.I considered many different ways to do this. Considering that the limitation of the computational capability I went with the downsampling[9] of the data to balance the dataset. To achieve this we make use of the resample() function perform class balancing by systematically downsampling the overrepresented classes which are safe and scam in this case to match the sample size of the minority class which being spam in this case. After that, the balanced dataset is randomly shuffled so that is there is any correlation related to the order of the samples then that is eliminated.

4.2.3 Cleaning the dataset. First of all we download the necessary Natural Language Processing (NLP) resources from the Natural Language Toolkit (NLTK)[6]. It is a popular library in Python for working with human language data. The main use of this library is to use the stopwords[15] and wordnet[11] resources which are downloaded and used to support text cleaning and preprocessing. Stopwords[15] are common words such as "the," "and," or "is" etc. and they generally do not carry important information or meaning in NLP tasks. While the wordNet[11] is a lexical database that helps in lemmatizing words, i.e., reducing words to their base or root form.

Then I create the clean text function for the processing and cleaning of the data in a proper manner. First of all, it removes any HTML tags using a regular expression, which helps strip out unnecessary formatting. Next, we replace the newline characters with spaces. This leads to a creation of the text which is uniform in nature. Generally special characters does not give any information which can help in our classification. So we remove special characters so that only alphanumeric content remains, which makes our text simple. Then we convert all the text to lowercase to reduce different types of cases like camelCase etc. which helps us avoid treating the same word as a different word due to case mismatching. The final step in the cleaning function is to apply lemmatization. It reduces words to their base form (e.g., "playing" becomes "play"). Along with this, we also filter out common stopwords using the stopwords list from NLTK. This leads to a dataset which contains the most meaningful words while irrelevant words which do not contribute to the data are removed, which helps in improving model performance along with the efficiency in the model.

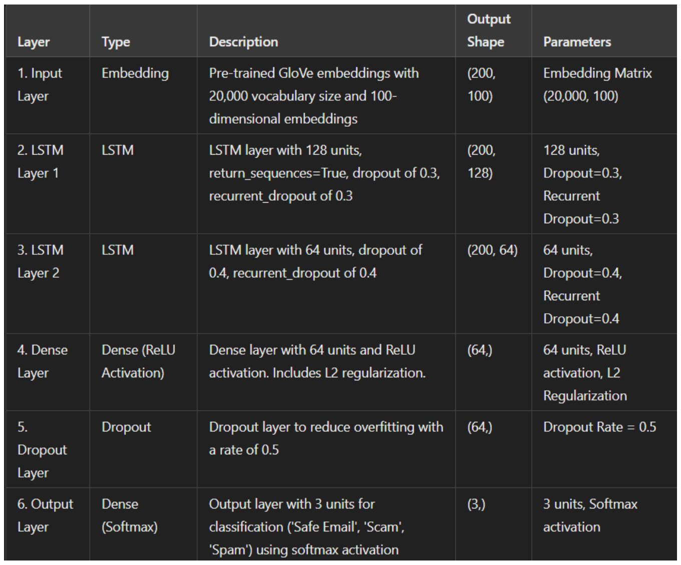
5 Model Description

5.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) model is a type of recurrent neural network (RNN). The main use of the Long Short-Term Memory as the name suggests, addresses the challenges of learning long-term dependencies in a data which is sequential. The traditional models of the RNNs often used to struggle to remember the information for a long sequence of data mainly due to the reason of vanishing gradient. LSTMs were developed to solve this vanishing gradient problem. They made use of a gating mechanism which controls the information flow in the structure, which allowed the network to pick and choose the information which it wants to save and which it wants to forget. The structure includes an input gate, a forget gate, and an output gate, which work in tandem to enable the model to capture long-range relationships in data. Forget gate decides which information from the previous step needs to be removed from the cell state. Input gate keeps the track of the new information which needs to be added to the cell state. While the output gate as the name suggests it is used to control the output of the cell state to the next layer.

For our project, we used an LSTM model to classify emails into three categories: 'Safe Email', 'Scam', and 'Spam'. After the preprocessing was done on the data, the first step we performed was splitting the dataset into training and testing subsets with the 80:20 ratio. After this we tokenized the data. Now as a result of this we were able to convert each email to a sequence of integers. We padded the sequence so that all have the same length. On top of this we applied a GloVe model which was a pre-trained model for word embeddings. This was done so that the understanding of the text gets better. The embeddings provided by the glove model was loaded matrix. This matrix was then used to initialize the embedding layer in the LSTM model, allowing it to use the pre-learned word representations.

The LSTM network architecture used in this project consists of two LSTM layers, both of the layers have dropout and recurrent dropout to stop overfitting of this model. After the LSTM layers, I added dense layers, with the final layer having three output units and a softmax activation function for multi-class classification. The model was compiled with the Adam optimizer and sparse categorical cross-entropy loss function, suitable for a multi-class classification task. After this I also employed early stopping and model checkpoint to monitor the training process, ensuring the best model was saved. This sometimes lead to model being trained for some epochs and then it just stopped because it was not improving. So for this we kept the patience of 5 which means that if it model does not make significant change even after 5 epochs then the model stops training. The model's performance was evaluated using metrics such as precision, recall, and F1-score, and the results were visualized through bar plots and heatmaps to better understand the model's effectiveness across each category. Finally, the trained model was saved for future deployment in email classification tasks.



Layer	Type	Description	Output Shape	Parameters
1. Input Layer	Embedding	Pre-trained GloVe embeddings with 20,000 vocabulary size and 100-dimensional embeddings	(200, 100)	Embedding Matrix (20,000, 100)
2. LSTM Layer 1	LSTM	LSTM layer with 128 units, return_sequences=True, dropout of 0.3, recurrent_dropout of 0.3	(200, 128)	128 units, Dropout=0.3, Recurrent Dropout=0.3
3. LSTM Layer 2	LSTM	LSTM layer with 64 units, dropout of 0.4, recurrent_dropout of 0.4	(200, 64)	64 units, Dropout=0.4, Recurrent Dropout=0.4
4. Dense Layer	Dense (ReLU Activation)	Dense layer with 64 units and ReLU activation. Includes L2 regularization.	(64,)	64 units, ReLU activation, L2 Regularization
5. Dropout Layer	Dropout	Dropout layer to reduce overfitting with a rate of 0.5	(64,)	Dropout Rate = 0.5
6. Output Layer	Dense (Softmax)	Output layer with 3 units for classification ('Safe Email', 'Scam', 'Spam') using softmax activation	(3,)	3 units, Softmax activation

Figure 2. LSTM Model

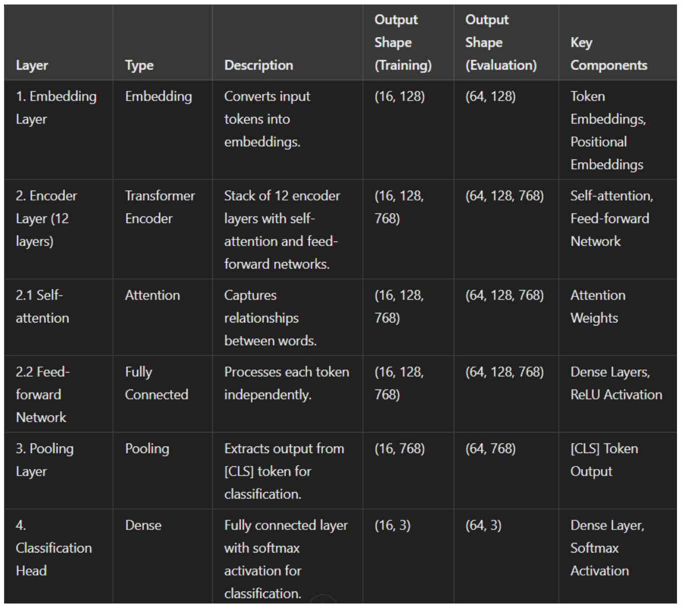
5.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT (Bidirectional Encoder Representations from Transformers) is a deep learning model designed for natural language processing tasks, developed by Google. When we compare BERT to previous models, BERT is pre-trained using a large set of texts in a bidirectional manner, which means that it can understand the context of a word by viewing the words which come before and after the word in a given sentence. Due to this special ability, BERT can capture representations of text in a much better way with the meaning of the text. This making it highly effective for a variety of tasks, such as text classification, named entity recognition, and question answering. The model uses transformers, a type of neural network architecture that excels in handling sequential data and understanding long-range dependencies in text.

5.2.1 BERT Model Implementation.

For our project, we used an BERT model to classify emails into three categories: 'Safe Email', 'Scam', and 'Spam'. In this project, we use the Hugging Face Transformers library to implement a BERT model for email classification. The first step we perform is to encode the labels of the dataset into numerical values using LabelEncoder. The dataset is then split into training and text sets with the ratio of 80:20. But the problem was how will the bert understand the email text. So next, we use BERT's tokenizer (BertTokenizer)[18] to preprocess the text data to converting raw email text into tokenized inputs that the model can process. Then we create a custom dataset class, EmailDataset, to structure the text and label pairs. It ensures the text is tokenized and padded to a consistent length, preparing it for input into the BERT model.

The pre-trained bert-base-uncased model is then loaded, which is fine-tuned for sequence classification by adding a classification head on top. BERT makes use of the transformer architecture which is made up of many layers which are essential to capture contextual information. First of all is the Embedding Layer. It is used to represent words as vectors in a continuous space by conterting tokenized text into embeddings as suggested by the name. After this BERT includes multiple Encoder Layers i.e. 12 in the case of bert-base-uncased. Each of this layer has two key layers inside it. The first one is the Self-attention Mechanism. It captures relationships between words in a sentence in both ways to understand them. The second layer is a Feed-forward Neural Network. This layer processes each token without the interference of other token by using two fully connected layers with a ReLU activation. After this, Pooling Layer extracts the output which is in relation with the CLS token. This CLS token is used to represent a full sentence. So finally by connecting to it we can use this to classify our emails. Finally to predict the labels for the input we add a fully connected Classification Head (Output Layer). As we have a multi-class classification, so we had to add a softmax layer to generate the class probabilities. The training is controlled by the Trainer class. After training, the model is evaluated on the test set, and performance metrics such as precision, recall, and F1-score are calculated using the classification report function to assess the model's ability to classify emails correctly. Finally, the trained model and tokenizer are saved, enabling future use for email classification tasks.



Layer	Type	Description	Output Shape (Training)	Output Shape (Evaluation)	Key Components
1. Embedding Layer	Embedding	Converts input tokens into embeddings.	(16, 128)	(64, 128)	Token Embeddings, Positional Embeddings
2. Encoder Layer (12 layers)	Transformer Encoder	Stack of 12 encoder layers with self-attention and feed-forward networks.	(16, 128, 768)	(64, 128, 768)	Self-attention, Feed-forward Network
2.1 Self-attention	Attention	Captures relationships between words.	(16, 128, 768)	(64, 128, 768)	Attention Weights
2.2 Feed-forward Network	Fully Connected	Processes each token independently.	(16, 128, 768)	(64, 128, 768)	Dense Layers, ReLU Activation
3. Pooling Layer	Pooling	Extracts output from [CLS] token for classification.	(16, 768)	(64, 768)	[CLS] Token Output
4. Classification Head	Dense	Fully connected layer with softmax activation for classification.	(16, 3)	(64, 3)	Dense Layer, Softmax Activation

Figure 3. Bert Model

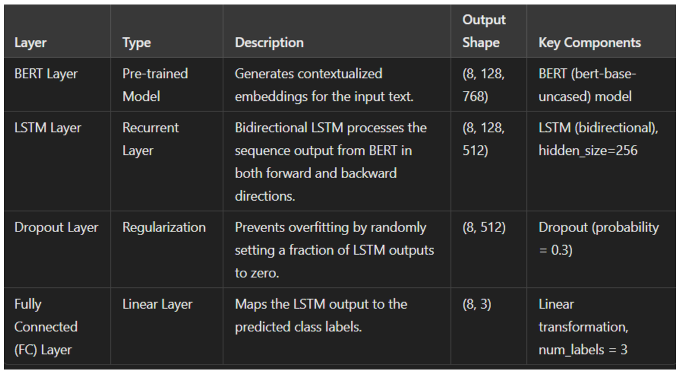
5.2.2 Hyper-tuning the BERT parameters.

After applying the BERT model I felt the need the to hyper-tune the parameters[17]. So I made use of the Optuna[2] for the hypertuning of the parameters. To do this first of all I split the dataset into training and test sets, and a custom EmailDataset class is created to tokenize the text using BERT's tokenizer which is similar to the process I performed earlier. Similarly the 'bert-base-uncased' model is initialized and a custom metric function is defined to calculate accuracy, precision, recall, and F1-score. The hyperparameters which I need to tune are learning rate, batch size, number of epochs, and weight decay, which are optimized using Optuna. The number of trials which I set was two due to the limited resources i.e. computational capability. The Training Arguments and Trainer classes from Hugging Face's Transformers library handle the training and evaluation of the model. The best hyperparameter combination is determined by maximizing the F1-score as it is a parameter which leads to a balance between recall and precision, and the resulting best configuration is printed after the optimization process.

5.3 BERT With LSTM

Now after I had implemented the LSTM model and then followed it by BERT model and then hypertuning the model, I thought what I can do which can lead to even better results. Till now BERT was giving the best result so I thought of combining the two so that I get the best of both the models. The BertLSTMClassifier integrates a pre-trained BERT model from Hugging Face to generate contextual embeddings of the input text. BERT processes the text to understand the context of each word, producing hidden states for each token. These embeddings are then passed through a bidirectional LSTM layer, which captures both forward and backward dependencies in the sequence. The LSTM's output corresponding to the [CLS] token, which represents the entire sentence, is used for classification. This output is further processed by a dropout layer for regularization and a fully connected layer that maps it to the final class labels.

The dataset is managed using a custom EmailDataset class that handles tokenization and formatting, preparing the data for BERT input. This class tokenizes the text using BERT's tokenizer, ensuring the text is padded or truncated to 128 tokens. The data is then split into training and test sets and passed into the CustomTrainer class, which inherits from Hugging Face's Trainer. This custom trainer overrides the loss computation to calculate cross-entropy loss during training. The model is trained for four epochs, with parameters such as learning rate, batch size, and number of epochs being fine-tuned for optimal performance. By combining the powerful language representations from BERT with the sequence-learning capability of LSTM, this hybrid model aims to achieve better results in tasks like email classification.



Layer	Type	Description	Output Shape	Key Components
BERT Layer	Pre-trained Model	Generates contextualized embeddings for the input text.	(8, 128, 768)	BERT (bert-base-uncased) model
LSTM Layer	Recurrent Layer	Bidirectional LSTM processes the sequence output from BERT in both forward and backward directions.	(8, 128, 512)	LSTM (bidirectional), hidden_size=256
Dropout Layer	Regularization	Prevents overfitting by randomly setting a fraction of LSTM outputs to zero.	(8, 512)	Dropout (probability = 0.3)
Fully Connected (FC) Layer	Linear Layer	Maps the LSTM output to the predicted class labels.	(8, 3)	Linear transformation, num_labels = 3

Figure 4. BERT-LSTM Model

6 Evaluation Plan

The evaluation of the deep learning models will focus on assessing their ability to accurately classify emails into three categories: spam, scam, and safe. The plan includes different evaluation metrics and approaches.

- **Accuracy:** Measures the overall percentage of correctly classified emails across all categories.
- **Precision, Recall, and F1-Score:** These will be calculated for each class (spam, scam, safe):
  - **Precision:**

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

    To reduce false positives, we need to verify that the email which is flagged as spam or scam is really a malicious mail.
  - **Recall:**

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

    This will help in measuring how well the model identifies scams and spam, aiming to minimize false negatives.
  - **F1-Score:**

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

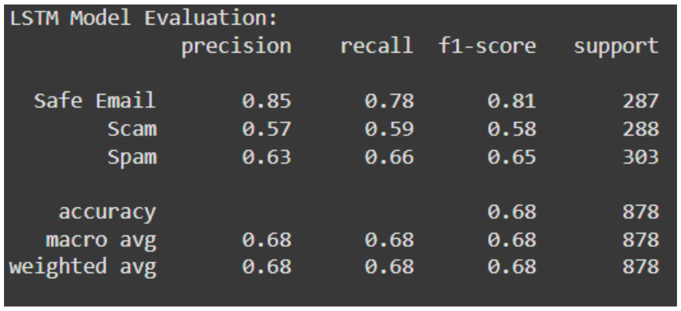
    This is used to strike a balance between precision and recall to provide a better measure of model performance.
- Confusion Matrix: Will be used to visualize how many emails of each class are correctly and incorrectly classified, giving details about specific areas where the model may be misclassifying.
- Loss Function The `compute_loss` method computes the **Cross-Entropy Loss** between the model's predictions and the true labels. Cross-Entropy Loss is used in classification tasks to measure the dissimilarity between the predicted probability distribution and the true distribution. Mathematically, it is defined as:

$$\text{Loss} = -\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i)$$

  where y_i is the true label (typically one-hot encoded), $\hat{y}_i$ is the predicted probability of class i, and N is the number of classes. The loss function penalizes incorrect predictions more heavily when the predicted probability for the true class is low. During training, the goal is to minimize this loss, driving the model's predictions closer to the true labels.

7 Evaluation and Experimental Results

7.1 LSTM Results



```
LSTM Model Evaluation:
              precision    recall  f1-score   support

  Safe Email       0.85      0.78      0.81       287
        Scam       0.57      0.59      0.58       288
        Spam       0.63      0.66      0.65       303

    accuracy                           0.68       878
   macro avg       0.68      0.68      0.68       878
weighted avg       0.68      0.68      0.68       878
```

Figure 5. LSTM Model Results

The LSTM model evaluation demonstrates its effectiveness in classifying emails into the categories of Safe Email, Scam, and Spam. The Safe Email category showed a strong performance, with a precision of 0.85, recall of 0.78, and an F1-score of 0.81, indicating that the model was effective in identifying non-spam emails. In contrast, the Scam category had lower scores, with precision at 0.57, recall at 0.59, and an F1-score of 0.58, showing that the model had difficulty accurately detecting scam emails. The Spam category achieved moderate results with a precision of 0.63, recall of 0.66, and an F1-score of 0.65, suggesting some success in classifying spam but also highlighting areas for enhancement. Overall, the model's accuracy was 0.68, meaning it correctly classified 68 percent of the test data. The macro average, which treats each class equally, aligned with the overall accuracy at 0.68, while the weighted average, which reflects the proportion of each class in the data, also stood at 0.68, indicating a balanced but not optimal performance across all categories.

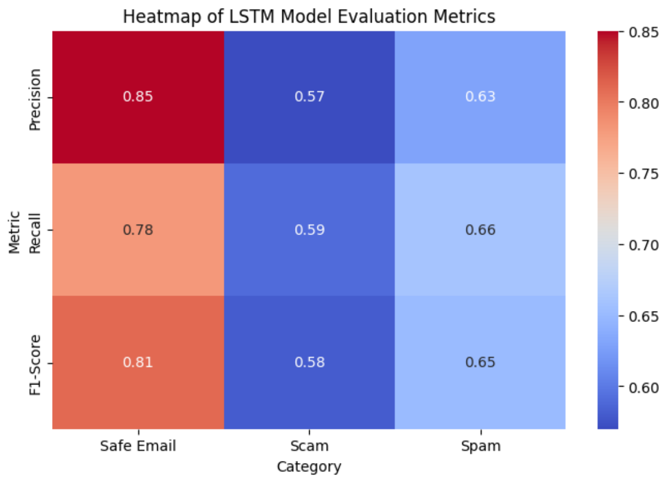
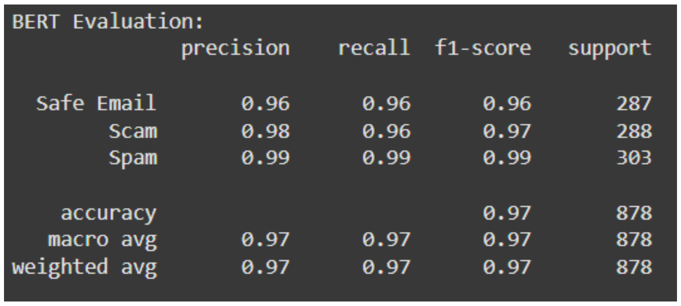


Figure 6. LSTM Model Evaluation Matrix

7.2 BERT Results



```
BERT Evaluation:
              precision    recall  f1-score   support

  Safe Email       0.96      0.96      0.96       287
        Scam       0.98      0.96      0.97       288
        Spam       0.99      0.99      0.99       303

    accuracy                           0.97       878
   macro avg       0.97      0.97      0.97       878
weighted avg       0.97      0.97      0.97       878
```

Figure 7. BERT Model Results

The BERT model evaluation indicates exceptional performance in classifying emails into the categories of Safe Email, Scam, and Spam. The Safe Email category showed outstanding results with a precision of 0.96, recall of 0.96, and an F1-score of 0.96, suggesting that the model effectively identified non-spam emails with high accuracy. The Scam category also demonstrated strong performance, with a precision of 0.98, recall of 0.96, and an F1-score of 0.97, indicating that the model was able to accurately detect scam emails. The Spam category exhibited the highest scores, with a precision of 0.99, recall of 0.99, and an F1-score of 0.99, reflecting the model's superior ability to identify spam emails. The overall accuracy was 0.97, meaning that 97 percent of the test data was correctly classified. The macro average, which treats each class equally, and the weighted average, which accounts for the distribution of classes in the dataset, both reported an impressive score of 0.97, showing that BERT maintained consistent performance across all categories with minimal imbalance. A loss of 0.112998 indicates how well the model performs on

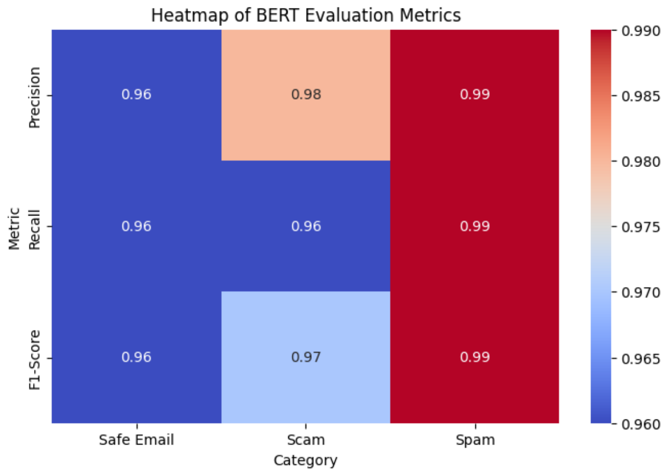
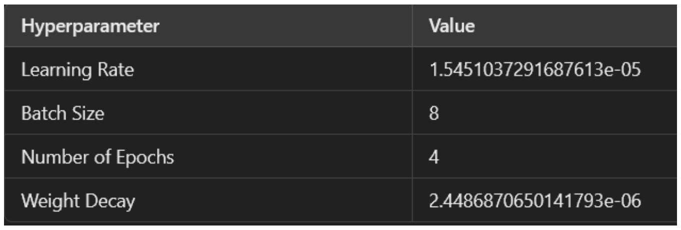


Figure 8. BERT Model Evaluation Matrix

unseen data during the training process. A lower value suggests better generalization, meaning the model has learned to make accurate predictions, but it should be balanced with the training loss to avoid overfitting.

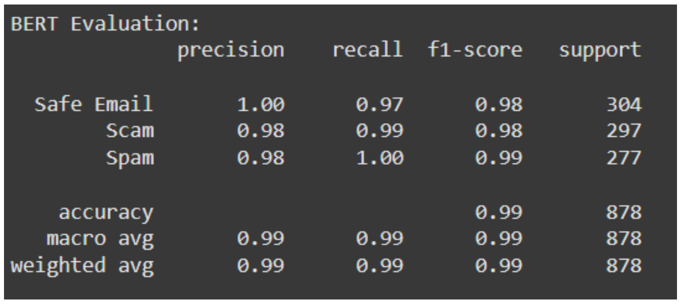
7.3 Hypertuned BERT Results



Hyperparameter	Value
Learning Rate	1.5451037291687613e-05
Batch Size	8
Number of Epochs	4
Weight Decay	2.4486870650141793e-06

Figure 9. Hypertuned BERT Model Parameters

The BERT model evaluation indicates exceptional performance in classifying emails into the categories of Safe Email, Scam, and Spam. The Safe Email category showed outstanding results with a precision of 0.96, recall of 0.96, and an F1-score of 0.96, suggesting that the model effectively identified non-spam emails with high accuracy. The Scam category also demonstrated strong performance, with a precision of 0.98, recall of 0.96, and an F1-score of 0.97, indicating that the model was able to accurately detect scam emails. The Spam category exhibited the highest scores, with a precision of 0.99, recall of 0.99, and an F1-score of 0.99, reflecting the



```
BERT Evaluation:
              precision    recall  f1-score   support

  Safe Email       1.00      0.97      0.98       304
        Scam       0.98      0.99      0.98       297
        Spam       0.98      1.00      0.99       277

    accuracy                           0.99       878
   macro avg       0.99      0.99      0.99       878
weighted avg       0.99      0.99      0.99       878
```

Figure 10. Hypertuned BERT Model Results

model's superior ability to identify spam emails. The overall accuracy was 0.97, meaning that 97 percent of the test data was correctly classified. The macro average, which treats each class equally, and the weighted average, which accounts for the distribution of classes in the dataset, both reported an impressive score of 0.97, showing that BERT maintained consistent performance across all categories with minimal imbalance.

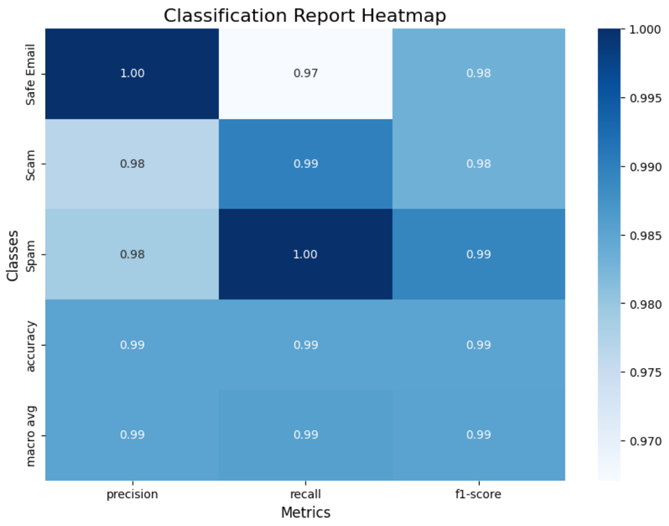
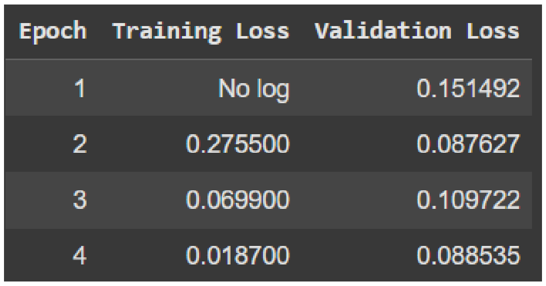


Figure 11. Hypertuned BERT Model Evaluation Matrix

The loss of 0.09825 for the hypertuned BERT model is significantly lower than the original BERT model's loss of 0.1129, indicating that the hyperparameter tuning improved the model's performance. This reduction in loss suggests that the fine-tuning process optimized the training process more effectively, allowing the hypertuned model to achieve better generalization and accuracy on unseen data.

7.4 BERT-LSTM Model

A loss of 0.0885 indicates that the hybrid BERT-LSTM model performs well, with a relatively low error rate during validation. This suggests that the model's combined architecture is effectively capturing and learning the important features from the data, resulting in a more accurate and robust performance in classifying email content. This improved loss value compared to the individual models shows the potential benefit of integrating BERT's contextual understanding with LSTM's sequential processing to create a model that excels in complex text classification tasks.



Epoch	Training Loss	Validation Loss
1	No log	0.151492
2	0.275500	0.087627
3	0.069900	0.109722
4	0.018700	0.088535

Figure 12. BERT-LSTM Model Loss

8 Broader Impacts

The broader impact of this email classification project is on a variety of levels starting from small email use to a large company which provides electronic mail services and finally to the society as a whole. First, by leveraging deep learning models such as BERT and LSTM, the project aims to improve the accuracy of spam and scam detection, creating a safer email environment. This reduces the risks of phishing, identity theft, and data breaches, helping individuals and organizations protect sensitive information and reduce operational costs which are incured due to manual email filtering.

On a societal level, the project contributes to enhancing cybersecurity awareness by providing an effective tool for detecting harmful emails. As cyber threats evolve, this system can help users stay informed and more vigilant in their online interactions, thus contributing to a more secure digital ecosystem. Additionally, the accessibility of this technology allows smaller businesses or individuals without technical expertise to benefit from sophisticated email filtering systems.

Economically, the project can increase trust in email communications and e-commerce, leading to more secure online transactions and fostering a safer environment for digital interactions. Furthermore, the model's ability to filter malicious content in real time could enable companies to allocate resources more efficiently, mitigating financial losses due to scams or fraud.

9 Work Distribution

All work was done by me i.e. Atharva Sardar in the entirety. I followed 3 sprints to complete the project. First sprint was searching for a proper model which would work well with the project objectives. Then the implementing LSTM and BERT models was another big task in this sprint.Sprint two dealt with the fine-tuning and hypertuning of the BERT model. I explored different ways to do this. After a lot of trial and error I reached to this outcome.Finally in the third sprint I tried a variety of different combinations of the models to check which gives the bast result. With my limited computation power I was able to zero down this model.

10 Conclusion

The project dealt with different models ranging from a base model of the LSTM to the BERT model and then the hyperparameter tuning and finally the BERT-LSTM model which gave the best performance for this project. In conclusion, this project successfully demonstrates the power of deep learning, particularly the combination of BERT and LSTM, to classify emails into categories such as "Safe Email", "Scam", and "Spam". By leveraging pre-trained models and advanced natural language processing techniques, the system effectively improves email security, reduces the risks of phishing, and enhances user experience.

Acknowledgments

Thanking professor Chandan Reddy for giving opportunity to work on this project

I would like to thank the reviewers for their detailed feedback on the project proposal. Based on the feedback provided, I have made several improvements to the report and project. Specifically, I have added more technical detail regarding the data preprocessing steps and the integration of datasets. Additionally, I have addressed the concerns regarding model tuning, hyperparameter optimization, and dataset imbalance. Furthermore, I clarified the handling of evaluation metrics and refined the overall presentation to enhance clarity and coherence. As suggested I have included some visual methodologies for the evaluation.

References

[1] Isra'a AbdulNabi and Qussai Yaseen. 2021. Spam Email Detection Using Deep Learning Techniques. *Procedia Computer Science* 184 (2021), 853–858. https://doi.org/10.1016/j.procs.2021.03.107 The 12th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 4th International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[3] Faisal S. Alsubaei, Abdulwahab Ali Almazroi, and Nasir Ayub. 2024. Enhancing Phishing Detection: A Novel Hybrid Deep Learning Framework for Cybercrime Forensics. *IEEE Access* 12 (2024), 8373–8389. https://doi.org/10.1109/ACCESS.2024.3351946

[4] Subhadeep Chakraborty. 2023. Phishing Email Detection. https://doi.org/10.34740/KAGGLE/DSV/6090437

[5] Emmanuel Gbenga Dada, Joseph Stephen Bassi, Haruna Chiroma, Shafi'i Muhammad Abdulhamid, Adebayo Olusola Adetunmbi, and Opeyemi Emmanuel Ajibuwa. 2019. Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon* 5, 6 (2019), e01802. https://doi.org/10.1016/j.heliyon.2019.e01802

[6] Nitin Hardeniya, Jacob Perkins, Deepti Chopra, Nisheeth Joshi, and Iti Mathur. 2016. *Natural language processing: python and NLTK*. Packt Publishing Ltd.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[8] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, Vol. 1. Minneapolis, Minnesota, 2.

[9] Wonjae Lee and Kangwon Seo. 2022. Downsampling for binary classification with a highly imbalanced dataset using active learning. *Big Data Research* 28 (2022), 100314.

[10] Wes McKinney et al. 2011. pandas: a foundational Python library for data analysis and statistics. *Python for high performance and scientific computing* 14, 9 (2011), 1–9.

[11] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[12] Author Name or Organization. 2023. Spam Mails Dataset. https://www.kaggle.com/datasets/venky73/spam-mails-dataset/data Accessed: YYYY-MM-DD.

[13] Justin M. Rao and David H. Reiley. 2012. The Economics of Spam. *Journal of Economic Perspectives* 26, 3 (September 2012), 87–110. https://doi.org/10.1257/jep.26.3.87

[14] Said Salloum, Tarek Gaber, Sunil Vadera, and Khaled Shaalan. 2022. A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access* 10 (2022), 65703–65727.

[15] Serhad Sarica and Jianxi Luo. 2021. Stopwords in technical language processing. *Plos one* 16, 8 (2021), e0254937.

[16] Ashok Yadav, Atul Kumar, and Vrijendra Singh. 2023. Open-source intelligence: a comprehensive review of the current state, applications and future perspectives in cyber security. *Artificial Intelligence Review* 56, 11 (2023), 12407–12438.

[17] Yichun Yin, Cheng Chen, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2021. Autotinybert: Automatic hyper-parameter optimization for efficient pre-trained language models. *arXiv preprint arXiv:2107.13686* (2021).

[18] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. 2021. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832* (2021).