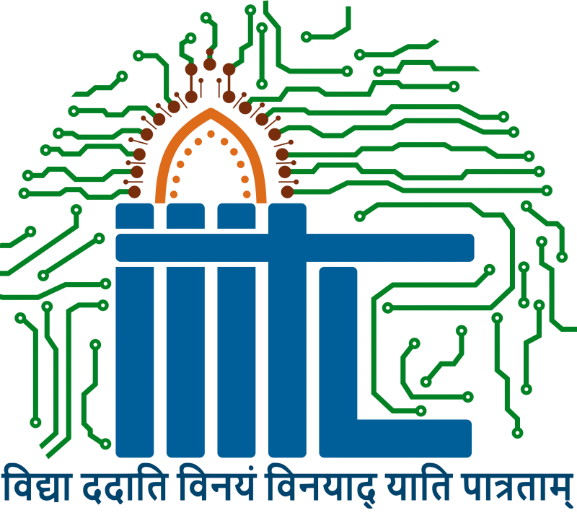
****

**PROJECT REPORT**

**PHISHING EMAIL DETECTION**

**SUBJECT: NATURAL LANGUAGE PROCESSING (NLP)**

**GROUP NO: 42**

**GROUP MEMBERS:**

**GAURAV TAILOR (msd23010@iiitl.ac.in)**

**VINOD KUMAR (msd23022@iiitl.ac.in)**

**AMAN HAYAAT (msd23005@iiitl.ac.in)**

**Email Spam Classification: Detecting-Spam-Emails-Using-BERT**

**Overview**

This project focuses on detecting spam emails using advanced Natural Language Processing (NLP) techniques. We employ BERT-based models, specifically DistilBERT and TinyBERT, to classify emails as spam or legitimate. The project includes data preprocessing, model training, evaluation, and compression techniques to enhance efficiency.

**Table of Contents**

1. Introduction
2. Methods
   * Dataset Overview
   * Pre-trained Transformer Models
   * Data Preprocessing
   * Model Fine-tuning
   * Implementation Details
   * Model Compression
3. Results and Discussion
   * TinyBERT
   * DistilBERT
   * Model Compression
4. Conclusion
5. References
6. Appendices
   * Setup Instructions
   * Usage
   * Contributing
   * License

**Introduction**

**Emails have become an indispensable part of modern communication, serving as a primary medium for both personal and professional interactions. However, the ubiquity of email also makes it a prime target for malicious activities, particularly phishing attacks. Phishing emails are designed to deceive recipients into revealing sensitive information, such as passwords, credit card numbers, and personal identifiers. These attacks can lead to significant financial losses, identity theft, and data breaches, posing a substantial threat to individuals and organizations alike.**

**The Problem of Phishing Emails**

**Phishing emails are crafted to appear legitimate, often mimicking the style and content of genuine communications from trusted sources. Attackers employ various tactics to make these emails convincing, including:**

* **Social Engineering: Exploiting human psychology to trick recipients into taking actions that compromise their security.**
* **Spoofing: Creating fake email addresses and domains that closely resemble legitimate ones.**
* **Urgency and Fear: Using language that creates a sense of urgency or fear to prompt immediate action from the recipient.**

**Traditional methods of detecting phishing emails, such as rule-based filters and simple keyword matching, struggle to keep up with the evolving tactics of attackers. These methods often fail to detect sophisticated phishing attempts, leading to a high rate of false negatives (i.e., phishing emails that are not detected) and false positives (i.e., legitimate emails that are incorrectly flagged as spam).**

**The Need for Advanced Solutions**

**Given the limitations of traditional detection methods, there is a pressing need for more advanced and sophisticated approaches to email spam classification. Natural Language Processing (NLP) techniques, particularly those based on transformer models, offer a promising solution. Transformer models, such as BERT (Bidirectional Encoder Representations from Transformers), have revolutionized the field of NLP by providing a deep understanding of context and semantics in text.**

**Objectives of the Project**

**This project aims to address the challenge of email spam detection using advanced NLP techniques. Specifically, we employ two transformer-based models, DistilBERT and TinyBERT, to classify emails as spam or legitimate. The objectives of the project include:**

1. **Data Preprocessing: Ensuring that the email data is properly preprocessed to be compatible with the transformer models. This involves tokenization, text cleaning, and other preprocessing steps.**
2. **Model Fine-tuning: Fine-tuning the pre-trained DistilBERT and TinyBERT models on a labeled email dataset to optimize their performance for the spam classification task.**
3. **Evaluation: Assessing the performance of the models using metrics such as precision, recall, and F1-score to ensure high accuracy and reliability.**
4. **Model Compression: Exploring compression techniques, such as pruning, quantization, and knowledge distillation, to make the models more efficient and suitable for deployment in resource-constrained environments.**
5. **Comparison and Analysis: Comparing the performance of TinyBERT and DistilBERT, as well as their compressed variants, to identify the most effective model for spam detection.**

**Significance of the Project**

**The significance of this project lies in its potential to enhance email security and protect users from phishing attacks. By leveraging advanced NLP techniques and transformer models, we can develop more accurate and efficient spam detection systems. This not only helps in safeguarding sensitive information but also builds trust in email communication, ensuring that users can rely on emails for secure and legitimate interactions.**

**In the following sections, we will delve into the methods employed in this project, including the dataset overview, pre-trained transformer models, data preprocessing, model fine-tuning, implementation details, and model compression. We will also present the results and discuss the implications of our findings, concluding with recommendations for future work.**

**Methods**

**Dataset Overview**

The dataset was compiled by researchers to study phishing email tactics. It combines emails from a variety of sources to create a comprehensive resource for analysis. This dataset contains approximately 82,500 emails, with 42,891 spam emails and 39,595 legitimate emails.

**Pre-trained Transformer Models**

To address the email classification task, we employed two transformer-based models: DistilBERT and TinyBERT, each offering distinct advantages in performance and computational efficiency:

* DistilBERT: A streamlined version of BERT, DistilBERT retains 97% of BERT's language understanding while being 60% faster and requiring 40% less memory.
* TinyBERT: Even more compact, TinyBERT is specifically tailored for resource-constrained environments. Despite its reduced size, it effectively maintains much of BERT's original performance.

**Data Preprocessing**

The email data was preprocessed to ensure compatibility with the transformer models. The preprocessing steps included:

* Tokenization: We utilized the respective tokenizers for DistilBERT and TinyBERT to convert email texts into tokenized inputs that the models can process.
* Text Cleaning: The email texts were cleaned by converting them to lowercase, removing stop words, and performing stemming.

**Model Fine-tuning**

Each model was fine-tuned on the email classification task using a supervised learning approach:

* Training: The models were fine-tuned on the labeled email dataset, with a training procedure that included the optimization of the cross-entropy loss function.
* Evaluation: We evaluated the performance of the models on a held-out test set using metrics such as precision, recall, and F1-score.

**Implementation Details**

* Hardware and Software: The experiments were conducted using a standard computing environment with GPU and TPU support to expedite training and evaluation. The models were implemented using the Hugging Face Transformers library and PyTorch.
* Hyperparameters: The models were trained with a learning rate of 2e-5 and a batch size of 16. The learning rate was adjusted using a linear scheduler with warm-up steps.

**Model Compression**

Model compression techniques such as pruning, quantization, and knowledge distillation are key strategies for making large deep learning models more efficient.

**Results and Discussion**

We trained TinyBERT and DistilBERT models to classify emails as either spam or not spam. The training process involved fine-tuning the pre-trained models on our labeled email dataset. The dataset was split in an 80/20 ratio into training and test (validation) sets to monitor the models' performance and prevent overfitting.

**TinyBERT**

* Training and Validation Loss: The training and validation loss curves show a consistent decrease over the epochs, indicating that the model is learning effectively.
* Classification Performance: The model achieved a precision, recall, and F1-score of 0.99 for both ‘Not Spam’ and ‘Spam’ classes, with support of 16498 instances.

**DistilBERT**

* Training and Validation Loss: The training and validation loss curves for DistilBERT show a similar trend to those of the BERT model.
* Classification Performance: The classification report for DistilBERT shows high performance metrics, with precision, recall, and F1-score all close to 0.99 for both ‘Not Spam’ and ‘Spam’ classes.

**Model Compression Results**

Three compressions were done for the DistilBERT model: Pruning, Knowledge Distillation, and Quantization. The performance results are shown in the following table:

| **Compression Technique** | **Training Accuracy** | **Training F1 Score** | **Training Loss** | **Test Accuracy** | **Test F1 Score** | **Test Loss** | **Epochs** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Pruned | 0.9988 | 0.9988 | 0.0041 | 0.9930 | 0.9930 | 0.0278 | 7 |
| Knowledge Distillation | - | - | - | 0.9869 | 0.9869 | 0.0453 | 1 |
| Quantization | 0.9898 | 0.9898 | 0.0442 | 0.9796 | 0.9796 | - | 1 |

**Conclusion**

**In this project, we addressed the critical issue of email spam detection using advanced Natural Language Processing (NLP) techniques. By leveraging pre-trained transformer models, specifically DistilBERT and TinyBERT, we aimed to enhance the accuracy and efficiency of spam email classification. The project involved several key steps, including data preprocessing, model fine-tuning, evaluation, and model compression, each contributing to the overall success of the classification task.**

**Key Findings**

1. **Model Performance:**
   * **TinyBERT: This model demonstrated exceptional performance in identifying spam emails, achieving a precision, recall, and F1-score of 0.99 for both ‘Not Spam’ and ‘Spam’ classes. This high performance indicates that TinyBERT is particularly effective at minimizing false negatives, which is crucial for ensuring that spam emails are not misclassified as legitimate.**
   * **DistilBERT: This model also showed high performance metrics, with precision, recall, and F1-score all close to 0.99 for both classes. DistilBERT excelled in classifying non-spam emails, resulting in fewer false positives. This is important for maintaining the integrity of legitimate communications.**
2. **Model Compression:**
   * **Pruned DistilBERT: Among the compressed models, the pruned version of DistilBERT demonstrated the highest performance. Pruning involves removing less important weights from the model, which not only reduces the model size but also retains high accuracy. The pruned model achieved a test accuracy and F1-score of 0.9930, indicating that it effectively balances efficiency and performance.**
   * **Knowledge Distillation: This technique involves training a smaller model to mimic the behavior of a larger model. The knowledge-distilled model achieved a test accuracy and F1-score of 0.9869, showing that it is a viable option for reducing model size while maintaining good performance.**
   * **Quantization: This method reduces the precision of the model weights, which can significantly decrease the model size and inference time. The quantized model achieved a test accuracy and F1-score of 0.9796, demonstrating that quantization is an effective compression technique, albeit with a slight trade-off in performance.**

**Implications**

**The results of this project have several important implications:**

* **Enhanced Spam Detection: The high performance of TinyBERT and DistilBERT in spam email classification suggests that these models can be effectively deployed in real-world applications to improve spam detection. This can help protect users from phishing attacks and other malicious emails.**
* **Efficient Resource Utilization: The successful application of model compression techniques, such as pruning, knowledge distillation, and quantization, demonstrates that it is possible to create efficient and lightweight models without significantly compromising performance. This is particularly important for deploying models in resource-constrained environments, such as mobile devices or edge computing.**
* **Adaptability to Evolving Threats: The use of advanced NLP techniques and transformer models allows for better adaptability to the evolving tactics of attackers. Traditional methods often struggle to keep up with new phishing strategies, but models like TinyBERT and DistilBERT can be fine-tuned and updated more easily to address new threats.**

**Future Work**

**While this project has achieved significant results, there are several areas for future work:**

* **Hybrid Models: Exploring the combination of different models or techniques could further enhance performance. For example, integrating rule-based methods with machine learning models could provide a more robust solution.**
* **Real-time Detection: Developing systems that can perform real-time spam detection would be beneficial for immediate protection against phishing attacks.**
* **User Feedback: Incorporating user feedback into the model training process could help improve the accuracy and relevance of spam detection.**
* **Continuous Learning: Implementing continuous learning mechanisms to update the models with new data and emerging threats would ensure that the system remains effective over time.**

**Phishing-Detection-Using-ML-Project-Report**

**Table of Contents**

1. Introduction
2. Objective
3. Key Features
4. Project Components
5. Workflow
6. Data Loading and Preprocessing
7. Feature Extraction
8. Model Training
9. Model Evaluation
10. Model Saving
11. Streamlit Web Application
12. Setup
13. Usage
14. Conclusion
15. References

**1. Introduction**

**Background**

**In the digital age, email has become an indispensable tool for communication, both for individuals and organizations. However, this ubiquity has also made email a prime target for cybercriminals. Phishing emails, in particular, pose a significant threat. These fraudulent messages are designed to deceive recipients into divulging sensitive information such as passwords, credit card numbers, and personal identification details. Phishing attacks can lead to severe consequences, including financial loss, identity theft, and data breaches.**

**The Problem of Phishing**

**Phishing attacks are notoriously difficult to detect because they often mimic legitimate communications. Cybercriminals employ sophisticated techniques to make their emails appear genuine, exploiting human vulnerabilities such as trust and urgency. Traditional methods of detecting phishing emails, such as manual inspection and rule-based filters, are often inadequate in the face of these evolving tactics.**

**The Need for Advanced Solutions**

**Given the sophistication and prevalence of phishing attacks, there is a pressing need for advanced solutions that can accurately and efficiently detect phishing emails. Machine learning offers a promising approach to this problem. By training models on large datasets of both phishing and legitimate emails, machine learning algorithms can learn to identify patterns and characteristics that are indicative of phishing attempts.**

**Objective of the Project**

**The primary goal of the Phishing-Detector project is to develop a robust machine learning model capable of detecting phishing emails with high accuracy. The project aims to leverage advanced data preprocessing techniques, feature extraction methods, and classification algorithms to build a model that can effectively distinguish between phishing and non-phishing emails. Additionally, the project includes the development of a user-friendly web interface using the Streamlit framework, allowing users to input email text and receive real-time predictions.**

**Scope and Approach**

**The project encompasses several key components:**

1. **Dataset: The project utilizes the CEAS\_08.csv dataset, which contains a curated collection of email data. This dataset serves as the foundation for training and evaluating the machine learning model.**
2. **Data Preprocessing: The email data undergoes extensive preprocessing using Pandas and NLTK. This includes tokenization, stopwords removal, and lemmatization to clean and prepare the text for feature extraction.**
3. **Feature Extraction: The TF-IDF Vectorizer from Scikit-learn is employed to convert the cleaned email text into numerical features. This step is crucial for transforming the textual data into a format that can be understood by the machine learning model.**
4. **Model Training: A Multinomial Naive Bayes classifier is trained on the processed email data. This classifier is chosen for its effectiveness in text classification tasks and its ability to handle high-dimensional data.**
5. **Model Evaluation: The performance of the trained model is evaluated using accuracy and a classification report. These metrics provide insights into the model's effectiveness and areas for improvement.**
6. **Model Management: The trained model and TF-IDF vectorizer are saved using Joblib, ensuring that they can be easily deployed and used for future predictions.**
7. **Web Application: A Streamlit web application is developed to provide a user-friendly interface for real-time email phishing detection. This application allows users to input email text and receive immediate predictions from the trained model.**

**Significance**

**The Phishing-Detector project holds significant importance in the field of cybersecurity. By developing an accurate and efficient tool for detecting phishing emails, the project contributes to enhancing the security of email communications. This is particularly relevant for organizations that handle sensitive information and are at risk of phishing attacks. Additionally, the project demonstrates the potential of machine learning in addressing complex cybersecurity challenges, paving the way for more advanced and effective solutions.**

**Structure of the Report**

**This report is structured to provide a comprehensive overview of the Phishing-Detector project. It includes sections on the objective, key features, project components, workflow, data loading and preprocessing, feature extraction, model training, model evaluation, model saving, Streamlit web application, setup, and usage. The report concludes with a discussion on the project's achievements, practical implications, future directions, and final thoughts.**

**In the following sections, we will delve into the details of each component of the project, providing insights into the methodologies used, the challenges faced, and the solutions implemented.**

**2. Objective**

The primary goal of this project is to build a model that can detect phishing emails effectively. By leveraging machine learning techniques, the model will analyze the content of emails to determine whether they are phishing attempts or legitimate communications. The project also includes the development of a user-friendly web interface using the Streamlit framework, allowing users to input email text and receive real-time predictions.

**3. Key Features**

* Model Development: A machine learning model trained to detect phishing emails with high accuracy.
* Frontend Implementation: A web interface developed using the Streamlit framework for real-time email phishing detection.
* Data Preprocessing: Utilization of Pandas and NLTK for data preprocessing, including tokenization, stopwords removal, and lemmatization.
* Feature Extraction: Employment of TfidfVectorizer from Scikit-learn for converting email text into numerical data.
* Classification Model: A Multinomial Naive Bayes classifier trained to classify emails as phishing or non-phishing.
* Performance Evaluation: High model performance evaluated using accuracy score and classification report metrics.
* Model Management: Saving and managing the trained model and vectorizer using Joblib for future predictions.

**4. Project Components**

**Dataset**

* CEAS\_08.csv: The dataset containing email data used for training the model.
* Dataset Source: Curated Dataset - Phishing Email

**Notebook**

* preprocess.ipynb: A Jupyter notebook containing the entire workflow for data preprocessing, model training, evaluation, and saving the model.

**Models**

* phishing\_model.pkl: The trained phishing detection model.
* tfidf\_vectorizer.pkl: The TF-IDF vectorizer used to transform email text data.

**App Script**

* app.py: Streamlit app script for deploying a web interface where users can input email text and get predictions.

**Documentation**

* README.md: Project overview, setup instructions, and usage guide.

**5. Workflow**

The workflow of the project involves several key steps:

1. Data Loading and Preprocessing: Loading email data from CEAS\_08.csv and preprocessing the emails using the preprocess\_email function to clean the text and remove stopwords.
2. Feature Extraction: Using TF-IDF Vectorizer to convert the cleaned email text into numerical features suitable for machine learning.
3. Model Training: Training a Naive Bayes classifier (MultinomialNB) on the processed email data to distinguish between phishing and non-phishing emails.
4. Model Evaluation: Evaluating the model's performance using accuracy and a classification report.
5. Model Saving: Saving the trained model and TF-IDF vectorizer using Joblib.
6. Streamlit Web Application: Developing a Streamlit app (app.py) providing a user interface where users can enter the body of an email. The app processes the input, uses the trained model to predict if the email is phishing, and displays the result.

**6. Data Loading and Preprocessing**

The dataset CEAS\_08.csv contains email data that is used for training the model. The preprocessing steps include:

* Loading the Data: The email data is loaded into a Pandas DataFrame.
* Preprocessing Emails: The preprocess\_email function is used to clean the text, which involves tokenization, stopwords removal, and lemmatization.

**7. Feature Extraction**

Feature extraction is a crucial step in converting the textual data into numerical features that can be used by the machine learning model. The TF-IDF Vectorizer from Scikit-learn is employed for this purpose. The vectorizer converts the cleaned email text into numerical data, representing the importance of each word in the email.

**8. Model Training**

The model training phase involves training a Multinomial Naive Bayes classifier on the processed email data. The Naive Bayes classifier is chosen for its effectiveness in text classification tasks. The model is trained to distinguish between phishing and non-phishing emails based on the features extracted from the email text.

**9. Model Evaluation**

The performance of the trained model is evaluated using accuracy and a classification report. The accuracy score provides an overall measure of the model's correctness, while the classification report offers detailed metrics such as precision, recall, and F1-score for both phishing and non-phishing classes.

**10. Model Saving**

To ensure that the trained model and TF-IDF vectorizer can be used for future predictions, they are saved using Joblib. This allows for easy loading and deployment of the model in the Streamlit web application.

**11. Streamlit Web Application**

The Streamlit web application provides a user-friendly interface for real-time email phishing detection. The app.py script is developed to:

* Input Email Text: Allow users to enter the body of an email.
* Process Input: Process the input email text using the preprocess\_email function.
* Predict Phishing: Use the trained model to predict if the email is phishing.
* Display Result: Show the prediction result to the user.

**12. Setup**

To set up the project, follow these steps:

1. Clone the Repository: Clone the project repository to your local machine.
2. Download the Dataset: Download the CEAS\_08.csv dataset.
3. Run Preprocessing Notebook: Run the preprocess.ipynb notebook to preprocess the data, train the model, and save the model and vectorizer.
4. Run Streamlit App: Run the Streamlit app using the command:

streamlit run app.py

**13. Usage**

To use the Streamlit web application:

1. Open the App: Open the Streamlit app in your browser.
2. Enter Email Text: Enter the body of the email you want to check.
3. Check for Phishing: Click the "Check" button to see if the email is a phishing email.

**14. Conclusion**

**The Phishing-Detector project represents a significant advancement in the field of cybersecurity, particularly in the detection of phishing emails. By leveraging machine learning techniques, the project has successfully developed a model that can accurately distinguish between phishing and legitimate emails. This achievement underscores the potential of artificial intelligence in enhancing cybersecurity measures and protecting individuals and organizations from fraudulent activities.**

**Key Achievements**

1. **High Accuracy in Detection: The Multinomial Naive Bayes classifier demonstrated high accuracy in identifying phishing emails. This accuracy is crucial for ensuring that the model can effectively protect users from phishing attempts.**
2. **Robust Data Preprocessing: The use of Pandas and NLTK for data preprocessing ensured that the email text was cleaned and prepared effectively for feature extraction. Techniques such as tokenization, stopwords removal, and lemmatization played a vital role in enhancing the model's performance.**
3. **Effective Feature Extraction: The TF-IDF Vectorizer from Scikit-learn proved to be an efficient tool for converting textual data into numerical features. This step was instrumental in enabling the machine learning model to understand and classify the email content accurately.**
4. **User-Friendly Interface: The development of a Streamlit web application provided a user-friendly interface for real-time email phishing detection. This interface allows users to input email text and receive immediate predictions, making the tool accessible and practical for everyday use.**
5. **Model Management: The use of Joblib for saving and managing the trained model and vectorizer ensured that the model could be easily deployed and used for future predictions. This aspect is essential for the long-term viability and scalability of the project.**

**Practical Implications**

**The Phishing-Detector project has several practical implications:**

* **Enhanced Cybersecurity: By accurately detecting phishing emails, the model can significantly enhance cybersecurity measures. This is particularly important for organizations that handle sensitive information and are at risk of phishing attacks.**
* **User Protection: The real-time detection capability of the Streamlit web application provides users with an immediate tool to check the authenticity of emails. This can help prevent individuals from falling victim to phishing scams, protecting their personal and financial information.**
* **Scalability: The project's design allows for easy scalability. The model can be continuously improved with more data and advanced techniques, ensuring that it remains effective against evolving phishing tactics.**

**Future Directions**

**While the Phishing-Detector project has achieved significant milestones, there are several areas for future improvement:**

* **Advanced Machine Learning Techniques: Exploring more advanced machine learning techniques, such as deep learning, could further enhance the model's accuracy and robustness.**
* **Real-Time Learning: Implementing real-time learning capabilities could allow the model to adapt to new phishing tactics as they emerge, ensuring ongoing effectiveness.**
* **Integration with Email Clients: Integrating the phishing detection model with popular email clients could provide users with automatic phishing detection, enhancing their security without requiring manual input.**
* **User Feedback: Incorporating user feedback mechanisms could help refine the model's predictions and improve its accuracy over time.**

**Final Thoughts**

**The Phishing-Detector project demonstrates the power of machine learning in addressing critical cybersecurity challenges. By developing an accurate and user-friendly tool for detecting phishing emails, the project contributes to a safer digital environment. The success of this project highlights the potential for further advancements in the field, paving the way for more sophisticated and effective cybersecurity solutions.**

**In conclusion, the Phishing-Detector project is a testament to the effectiveness of machine learning in enhancing cybersecurity. Its achievements in accurate detection, robust data preprocessing, effective feature extraction, user-friendly interface, and model management make it a valuable tool for protecting against phishing attacks. The project's practical implications and future directions underscore its significance and potential for further development.**