INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE
M. U. Demirezen, Vol.12, No.3, pp.29-59
https://doi.org/10.55859/ijiss.1338813

Lambda Architecture-Based Big Data System for Large-Scale Targeted Social Engineering Email Detection

Mustafa Umut Demirezen[1] , Tuğba Selcen Navruz[2]

[1]Data Products Department, UDemy Inc., CA, USA
[2]Department of Electrical and Electronics Engineering, Gazi University, Ankara, Turkey
Corresponding Author: umut@demirezen.tech

Research Paper Received: 07.09.2023 Revised: 14.09.2023 Accepted: 25.09.2023

Abstract—In this research, we delve deep into the realm of Targeted Social Engineering Email Detection, presenting a novel approach that harnesses the power of Lambda Architecture (LA). Our innovative methodology strategically segments the BERT model into two distinct components: the embedding generator and the classification segment. This segmentation not only optimizes resource consumption but also improves system efficiency, making it a pioneering step in the field. Our empirical findings, derived from a rigorous comparison between the fastText and BERT models, underscore the superior performance of the latter. Specifically, The BERT model has high precision rates for identifying malicious and benign emails, with impressive recall values and F1 scores. Its overall accuracy rate was 0.9988, with a Matthews Correlation Coefficient value of 0.9978. In comparison, the fastText model showed lower precision rates. Leveraging principles reminiscent of the Lambda architecture, our study delves into the performance dynamics of data processing models. The Separated-BERT (Sep-BERT) model emerges as a robust contender, adept at managing both real-time (stream) and large-scale (batch) data processing. Compared to the traditional BERT, Sep-BERT showcased superior efficiency, with reduced memory and CPU consumption across diverse email sizes and ingestion rates. This efficiency, combined with rapid inference times, positions Sep-BERT as a scalable and cost-effective solution, aligning well with the demands of Lambda-inspired architectures. This study marks a significant step forward in the fields of big data and cybersecurity. By introducing a novel methodology and demonstrating its efficacy in detecting targeted social engineering emails, we not only advance the state of knowledge in these domains but also lay a robust foundation for future research endeavors, emphasizing the transformative potential of integrating advanced big data frameworks with machine learning models.

Keywords—big data, lambda architecture, cybersecurity, phishing, spam, email, deep learning, transformers, BERT, NLP

1. Introduction

Modern cyberspace is characterized by a complex networked digital environment, offering new prospects and avenues for companies to engage in extroverted activities and behaviors [1]. However, the emerging cyber ecosystem encounters various fears, encompassing cybercrime, advanced persistent threats (APTs), and zero-day attacks. These advanced threats evade conventional defense strate-

gies and necessitate a comprehensive approach to managing all endeavors to exploit vulnerabilities within the system [2]. Small and medium-sized enterprises (SMEs), as well as organizations, possess an understanding of their responsibility to adhere to the General Data Protection Regulation (GDPR) and safeguard the personal data under their control. The organization allocates resources toward implementing sophisticated and intelligent measures in order to enhance its cybersecurity stance [3]. Network traffic analysis plays a crucial role in enabling organizations to promptly detect and respond to security incidents, thereby mitigating the adverse effects of cyberattacks [1]. Through the provision of real-time monitoring and threat detection capabilities, this approach facilitates the swift identification of potential threats, enabling organizations to take appropriate measures in a timely manner. However, an easy, brother-scale cyber attack, called Phishing, represents the prevailing method employed in social engineering attacks [4]. A hacker employs these sorts of attacks in an attempt to acquire sensitive information from the user, with the intention of subsequently engaging in illegal activities against the user. In contemporary times, phishing attacks have emerged as a prevailing form of social engineering attacks encountered by individuals utilizing the public internet, as well as governmental bodies and commercial enterprises [5]. Hence, there is a pressing need for enhanced phishing detection technology to address the escalating proliferation of phishing emails. This requirement will explore the employment of machine learning methods and technical remedies proposed to mitigate the issue of phishing [6]. Additionally, it will highlight the crucial awareness that users should possess to effectively identify and prevent falling victim to phishing scams [4].

1.1. Phishing Email Classification with Classical Machine Learning

Phishing attacks have undergone evolutionary changes as time goes on, using various strategies to deceive people into revealing sensitive information, including usernames and passwords, and financial information [4]. Phishing attempts have been found to create significant harm, leading to serious financial losses, instances of identity theft, and reputational damage for the organizations that are targeted. In recent times, there has been a notable rise in both the quantity and complexity of phishing attempts, as cybercriminals persistently devise novel methods to elude identification and focus on individuals who are unaware of their intentions [7]. Recently, Artificial intelligence-based phishing detection methods gained much attention. In [6], the authors presented a detection model that utilizes machine learning techniques. Their aim was to identify the intrinsic characteristics of email text and other relevant features in order to classify them as either phishing or non-phishing. Three distinct datasets were employed for this purpose. Upon conducting a comparative analysis, it was determined that the utilization of a higher number of features resulted in the attainment of more accurate and efficient outcomes. The boosted decision tree algorithm achieved accuracy rates of 0.88, 1.00, and 0.97 in consecutive order when applied to the data sets. Another study [8] introduced a novel automated framework for detecting malicious emails, employing deep-enhanced learning to analyze various components of emails, including the body, header, and attachments, and presented a demonstration of an ensemble framework consisting of deep-learning classifiers. The comprehensive evaluation of the proposed framework demonstrates its effectiveness, as evidenced by an AUC of 0.993. A recent study [9] incorporated the use of machine learning algorithms and

deep convolutional neural networks (CNNs). The outcomes of the implementation have demonstrated accuracy rates of 92%, 94%, 96%, and 98% for the K-Nearest Neighbors (KNN), Naive Bayes (NB), Bidirectional Long Short-Term Memory (BiLSTM), and Bert-Based Cards models, respectively. The convolutional neural network algorithm exhibited superior performance compared to other algorithms in the tables, achieving a higher accuracy rate of 99%. In [10], the researchers used three datasets to train and evaluate machine learning classifiers. Furthermore, a total of six machine learning classifiers have been assessed, specifically NaiveBayes, ANN, DecisionStump, KNN, J48, and RandomForest. The classifiers were trained and tested on three datasets in a two-stage process. According to the results, the artificial neural network (ANN) classifier has demonstrated superior performance, achieving an accuracy rate of 88.92% when applied to Dataset-3.

1.2. Phishing Email Classification with Classical Machine Learning and Natural Language Processing

The prevalence of phishing emails has experienced a significant surge in recent years, indicating a pressing demand for enhanced and more efficient strategies to combat this issue. A variety of techniques have been developed to mitigate the impact of phishing emails; however, a comprehensive resolution to this issue remains elusive. Several studies [11] represent a pioneering effort in the domain of using natural language processing (NLP) and machine learning (ML) methodologies to identify and recognize phishing emails. In the realm of phishing email detection, it is common to employ machine learning (ML) techniques, specifically clustering and classification methods [12]. Consequently, these techniques rely on the utilization of machine learning-based methodologies, along with machine learning-based assessment criteria. However, the dynamic nature of phishers may ultimately render evaluation outcomes obsolete over time [13]. The aforementioned factor has significantly increased the intricacy of the cybersecurity field. Nevertheless, this issue can be effectively addressed by diligently focusing on the formulation of evaluation outcomes during the development process. The current utilization of NLP technology for the detection of phishing emails, instead of employing DL techniques, fails to consider the distinctions between anti-phishing email objectives and overlooks contextual information to some extent. Consequently, this approach hampers the advancement of phishing email detection [11].

1.3. Phishing Email Classification with Transformers

The most recent breakthrough in NLP is the architecture of transformers with the attention mechanism with [14]. Applications of transformers and derived architectures gained a lot of popularity in cybersecurity applications for their representation capability and accuracy for text data. A recent study [15] presented a novel approach to developing an effective classification model for detecting spam in online SMS messages. The proposed method leverages advanced topological sentence transformer techniques. Researchers proposed a feasible and efficient integration of pre-trained natural language processing (NLP) repository models with the functionality provided by the sklearn library and utilized large-text data models from HuggingFace, specifically the roberta-base model, and then employed linguistic natural language processing (NLP) transformer techniques on NLP datasets consisting of short sentences. Their method proved that the model utilizes semantically similar paraphrase and sentence transformer methodologies to achieve

optimal F_1 scores on an SMS dataset. Another study [16] introduced a novel method for detecting spam emails, which involves the utilization of a pretrained bidirectional encoder representation from the transformer (BERT) and machine learning algorithms. The objective is to accurately classify emails as either ham or spam. The email texts were inputted into the BERT model, and the resulting BERT outputs were utilized to represent the texts. Four different classifier algorithms were utilized in the field of machine learning to categorize the characteristics of the text into either ham or spam categories, and the efficacy of the proposed model was evaluated through experimentation, utilizing two publicly available datasets. The evaluation results indicate that the logistic regression algorithm exhibited superior classification performance in both datasets. Additionally, the proponents provided a rationale for the effective capability of the proposed model in identifying spam emails. In another recent study [17], the authors presented a highly adaptable automated email classification system that utilizes a unique publicly available dataset consisting of 1447 Romanian business-oriented emails that were manually annotated. A robust foundation was established by employing pre-trained Transformer models for token classification and multi-task classification, resulting in F1 scores of 0.752 and 0.764, respectively.

1.4. Cybersecurity and Big Data Processing

The concept of "Big Data" refers to the vast and complex sets of data that are too large and intricate to be effectively managed. Cybersecurity analytics systems, specifically those pertaining to Big Data Cybersecurity Analytics (BDCA), leverage prominent big data technologies such as Apache Spark to effectively gather, retain, and scrutinize substantial quantities of security event data with the objective of identifying and thwarting cyber-attacks [18]. The exponential growth of digital data, including security event data, is leading to a significant increase in volume. The rate at which security event data is generated and inputted into a Big Data Cybersecurity Analytics (BDCA) system is indeterminate. Hence, it is imperative for a Big Data Cybersecurity Analytics (BDCA) system to possess a high degree of scalability in order to effectively handle the volatile fluctuations in the velocity of security event data. Nevertheless, conventional software systems such as relational databases and data warehouses face limitations in their ability to effectively gather, store, and analyze vast amounts of data. Hence, there is a growing utilization of big data storage and processing technologies across diverse domains. This trend is driven by the need to effectively manage the substantial magnitude, speed, and diversity of data [19].

1.5. Cybersecurity Applications with Lambda Architecture

There are several LA bases cyber security applications that exist and getting more and more popular. In [20], the authors present a novel cognitive computing Security Operations Center (SOC) that utilizes intelligence-driven approaches and relies solely on progressive, fully automated methodologies. The efficiency of the proposed λ-Architecture Network Flow Forensics Framework ($\lambda - NF3$) makes it a robust cybersecurity defense framework that effectively mitigates adversarial attacks. The implemented architecture utilizes the Lambda machine learning framework to effectively analyze a combination of batch and streaming data. The proposed methodology employs an Extreme Learning Machine neural network with a Gaussian Radial Basis Function kernel (ELM/GRBFk) for conducting batch data analysis. As a result, this tool was designed for the field of forensics in big data,

with the aim of improving the automated defense strategies employed by Security Operations Centers (SOCs) in order to effectively address the various threats encountered within their environments. Another research, [21], was focused on examining the security aspects of big data and ensuring that its performance is maintained during transmission across networks. The security of big data in an environment of networks was proposed to be enhanced by the integration of compression and splitting mechanisms with big data encryption in the proposed model. Furthermore, the utilization of user-defined ports and the implementation of multiple paths during the transmission of large datasets in a segmented manner enhance the dependability and safeguarding of big data within a networked environment. Several sectors have been working on cyber security with different perspectives. In [22], researchers focused on a significant obstacle in the financial industry, namely the need for real-time cybersecurity analytics on financial transaction data. Their solution presents a novel approach to combining supervised and unsupervised artificial intelligence models, leveraging appropriate technological tools capable of efficiently processing vast quantities of transaction data. A Lambda architecture was devised to manage both real-time and batch analytics workflows in a cohesive manner. This architectural design was specifically engineered to effectively manage large volumes of data by leveraging the capabilities of both batch and stream processing techniques. The proposed methodology aims to achieve a harmonious equilibrium between latency, throughput, and fault tolerance. This is accomplished by employing batch processing techniques to generate comprehensive and precise representations of batch data. Simultaneously, real-time stream processing is utilized to generate representations of online data in a timely manner [22]. Another research [3] presents a novel Network Traffic Analyzer, which serves as a vital element within the cyber threat intelligence information sharing architecture (CTI2SA) of the Cyber-pi project. The proposed system, which is based on the Lambda (λ) architecture, improves existing active cybersecurity methodologies for traffic analysis by integrating batch and stream processing techniques to effectively manage large volumes of data. The core module of the Network Traffic Analyzer incorporates an automated model selection mechanism, which effectively identifies and selects the machine learning model that exhibits the most superior performance when compared to its competing counterparts. The objective is to ensure the continued operational effectiveness of the architecture's overall threat identification capabilities.

In the study [23], the authors introduced a novel intrusion detection system (IDS) model that utilizes deep neural networks within the context of edge cloud-based Lambda architecture. The efficacy of this approach was assessed using real-time traffic data within IoT networks, resulting in a 99% detection rate for attacks. However, the efficacy of this approach was compromised in identifying novel attack strategies as a result of its reliance on a narrow set of features from benchmark instances. In another IDS-related research [24], the researchers introduced a novel intrusion detection system (IDS) that utilizes a deep ensemble-based approach within the Lambda architecture framework. The proposed system adopts a multi-pronged classification strategy. The binary classification approach employs Long Short Term Memory (LSTM) to distinguish between malicious and benign network traffic. On the other hand, the multi-class classifier utilizes an ensemble of LSTM, Convolutional Neural Network (CNN), and Artificial Neural Network (ANN) classifiers to identify the specific type of attacks. The training of the model is conducted in the batch layer, whereas the evaluation in real-time is accomplished

by making inferences using the model in the speed layer of the Lambda architecture. The proposed methodology demonstrates a remarkable level of precision exceeding 99.93% and offers significant time-saving benefits through the implementation of a multi-faceted classification strategy and the utilization of the Lambda architecture.

The concurrent processing of data streams, both in real-time and offline, has been a crucial requirement for Big Data applications over an extended period. Various technologies and corresponding Big Data architectures are determined by the specific requirements of processing batch data and real-time processing tasks. The Lambda Architecture (LA) is a widely adopted concept that has been developed and is currently in common use [25]. The architecture consists of three distinct layers that facilitate the simultaneous processing of both data in motion (DiM) and data at rest (DaR), along with a serving layer responsible for presenting the derived insights. According to the source cited, every LA layer is assigned a specific role in the processing of data with different attributes. These layers are responsible for merging the processed results and providing the combined datasets for purposes such as visualization, querying, and data mining [26]. The primary function of the speed layer is to handle the continuous flow of data in real time, commonly referred to as streaming data or real-time data. This layer is particularly susceptible to disruptions caused by delays in data transmission and the occurrence of recurring data patterns. The batch layer assumes responsibility for the processing of offline data (DaR), the computation of predetermined analytics actions, and the rectification of potential errors that may occur during the delivery of data to the speed layer. The serving layer is responsible for the ingestion of data from both the batch and speed layers, as well as the subsequent indexing and merging of the resulting data sets to facilitate the execution of analytical queries. The capacity of the serving layer should be sufficient to efficiently process and manage a high volume of real-time streaming data (DiM) as well as bulk data (DaR). It is important to emphasize that LAs are finally consistent solutions for applications that involve enormous amounts of data and may be used to resolve the CAP theorem [27]. Upon completion of data processing, the batch layer undertakes the task of rectifying any discrepancies in data digestion that may have been generated by the real-time layer. Accurate data are eventually disseminated and rendered accessible at the serving layer, thereby enabling subsequent processes to obtain pertinent information.

The major benefit of an LA-based Big Data system is that it can handle the necessity of designing a fault-tolerant architecture to prevent data loss due to hardware failures and unanticipated errors during DaR and DiM processing. It performs effectively in applications that demand low-latency read and update operations. This type of system must be able to handle ad-hoc queries, as well as be linearly scalable and extendable. According to previous studies, traditional LA should have an extra layer [28]. To enhance LA, the researchers used software engineering concepts, developed a reference model for Big Data systems, and used it as the foundation for developing a software reference architecture (SRA) for semantic-aware Big Data operations. They demonstrated that SRA could manage the most common Big Data features by adding an extra layer to the LA, the Semantic Layer. These sophisticated data processing operations, as well as providing the corrected data as precisely as possible, clearly necessitate highly coordinated and continuous operation between the speed and batch layers. The fact that LA is made up of at least three

layers allows for the flexible use of different Big Data technologies for each layer, which is a huge advantage. However, there are certain disadvantages to this. Although LA is a promising approach, its success is dependent on an effective mix of appropriate and mature technology [29]. The main issue is creating a Big Data application for each layer independently and then integrating them so that they can operate together and be interoperable. Because various technologies are used at different LA levels, each layer requires its own development and maintenance effort. If the data model or data format in the application changes or if additional new analytics capabilities are required, this big data application must be updated, tested, and deployed at all layers. A LA was proposed in a recent report for smart agricultural applications. The work [30] provides a general framework for explaining the problems of obtaining, processing, storing, and displaying massive amounts of data, including batch and real-time data. A main archetype has been demonstrated and tested on several farms with impressive results. For context-sensitive trajectory prediction, an LA-based big data platform was proposed [31]. According to its design, this platform performs batch and stream operations before combining them to perform jobs that cannot be performed by analyzing any of these layers alone. The most important aspect of the proposed platform is that it is context-neutral. Their findings demonstrated that each component of the LA is effective in achieving specific goals and that the combination of these components is important to improve the overall accuracy and performance of the platform.

As a result, classical machine learning algorithms, deep learning architectures, and novel transformer-based NLP methods are very common in the literature. According to several results shared in the studies, transformer-based models have several advantages, such as improved classification accuracy. However, they require very high computation power, and retraining this architecture from scratch is not feasible, especially with a very limited corpus. Even after the successful training and testing stage, if a transformer-based model is decided to be used, it requires a considerable amount of system resources to serve the model, and when the number of inferences per second from the model-hosted environment increases, latency issues and problems might occur. So, it has to be scaled properly, and even model compression and quantization methods can be useful. In addition, specialized hardware for deep learning models accelerator, such as Amazon inferentia [32], could sometimes be a viable solution. In reality, when the number of emails to be classified increases, processing these emails, including the attachments and classification of these emails, is not feasible with the transformers due to resource constraints. However, data processing approaches based on big data might be useful for solving this type of problem. As a state-of-the-art approach currently, the lambda architecture can provide the best opportunities to mitigate this requirement. But transformer architectures' high resource requirements again constitute an obstacle to deploying these transformers in lambda architectures.

This study aims to present an LA, a reasonably new generation of big data architecture, as an end-to-end extensive data system for a scalable, fault-tolerant, and high-accuracy targeted social engineering email (spam and phishing emails) classification with a very novel data transfer method, an innovative deep learning model, and a low resource data processing methodology. To the best of our knowledge, this is the first study to propose the deployment of a big data system based on LA implementation for targeted social engineering email classification problems in conjunction with pre-

trained language models, such as BERT transformer architectures [33], RoberTa [34], and Conv-BERT [35]. As an additional contribution and novelty, to the best of our knowledge, again, our transformer architecture separating approach is the first time it has been proposed not only for big data systems but also for other application domains. This study focuses primarily on the examination and performance evaluation of the novel method of separating a transformer model into two parts and using these two partial model pieces in a lambda architecture to overcome high resource and hardware requirements. This study not only makes a valuable contribution to the current body of knowledge on LA but also fills a notable gap in the existing literature. By providing a novel transformer model splitting methodology to overcome the resource constraints of the transformer model deployment and to demonstrate its effectiveness through empirical experiments, this study paves the way for further research in this area.

It is worth noting that in addition to the transformer splitting method as a novel approach, this work is based only on the content-based phishing/spam email detection approach, and we have not taken advantage of using the other machine learning-related features such as IP addresses, URL-based information, attachments in the emails, image-embedded email body content.

This paper is meticulously structured to provide a comprehensive insight into the domain of detecting targeted social engineering emails. In Section 1, we introduce the pressing challenges and underscore the importance of advanced detection mechanisms. Section 2 provides a comprehensive review of the literature, highlighting prevailing research and methodologies; we dive into our chosen methodologies, exploring the nuances of transformer models such as the BERT and the fastText model. Our unique approach to classifying and detecting phishing emails is presented, where we detail our architecture and its real-world deployment using the Lambda Architecture system. Section 3 also outlines our experimental setup, including data sets, training, and evaluation metrics. We present and critically analyze our findings, comparing our results with existing methodologies and discussing their implications. The paper culminates in Section 4, where we summarize our contributions, revisit our research objectives, and propose potential directions for future exploration in this domain.

2. Material and Methods

In the rapidly evolving digital era, many organizations are increasingly prioritizing the exploration and utilization of relevant Big Data technologies to enhance the accuracy, speed, and efficiency of their analytical processes [36]. The challenge of performing real-time analytics on Big Data is significant, primarily due to the vast volume of intricate data that must be efficiently distributed for processing. Big data is characterized by its substantial volume, intricate structure, and real-time performance capabilities. The primary challenge associated with big data processing is to improve both the speed and accuracy of processing.

Lambda architecture (LA) offers a solution to some of these constraints inherent in data processing frameworks. The proposed approach is based on the utilization of two distinct data processing streams within a single system. This methodology involves real-time computing, focusing on the rapid processing of data streams, and batch computing, designed to handle large workloads for delayed processing [37]. Although these two modes are not novel in themselves, LAs enable their synchronized execution to prevent interference. The allocation of resources across cloud infrastructure has a significant impact on both performance and cost. Pre-

dicting performance in advance would allow architects to make more informed decisions regarding resource allocation, thus enhancing the efficiency of system utilization. Our approach serves as a rapid evaluation tool to assist in making design choices with respect to parameters, ultimately contributing to improved architecture designs.

On the other hand, phishing and email spamming, the predominant form of social engineering attacks, involves the perpetrator formulating an email with content enticing enough to lure potential victims [38]. Understanding the mindset of attackers is crucial when formulating email defense strategies. The application of principles of persuasion is the main approach in constructing highly effective emails. There are numerous machine learning-based classification models available that can potentially address this problem. This study aims to explore the feasibility of constructing a robust classifier leveraging machine learning-based transformers, such as BERT [39] and proposes an architectural modification for its usage in an LA for distributed data processing and only considers a content-based phishing/spam email detection approach.

The current study focuses on the analysis and performance assessment of a unique method of dividing a transformer model into two parts and employing these two partial models in the lambda architecture. This approach aims to overcome the high resource and specialized hardware requirements associated with transformer models. The proposed methodology can be easily adapted to other phishing/malicious emails written in different languages, such as Turkish.

In the following sections, we present an LA as an end-to-end big data system for phishing email classification with a novel transformer-based architecture usage. It is worth noting that because we use common phishing email-based datasets, our current work is only limited to providing a solution for the English language. However, our proposed methodology can easily be applied/adapted to other phishing/malicious emails written in other languages, such as Turkish.

2.1. Problem Definition

The literature often uses traditional machine learning algorithms, deep learning architectures, and innovative transformer-based NLP techniques. Research findings suggest that transformer-based models offer a number of benefits, including increased classification accuracy. Though retraining this architecture from the start is not possible, particularly with a very small corpus, they need very high levels of computer power. A transformer-based model requires a significant amount of system resources to serve it, even after the successful training and testing stages, and when the rate of inferences per second from the model-hosted environment rises, latency issues and problems may arise. As a result, the scaling must be done correctly, and approaches like model compression and quantization may be helpful. Additionally, hardware designed specifically to accelerate deep learning models, such as Amazon inferentia [32], can sometimes be an effective remedy. In fact, due to resource limitations, processing emails, including attachments, and classifying emails becomes impractical with transformers as the volume of emails to be categorized rises. To solve this kind of problem, big data-based data processing techniques could be helpful. Lambda architecture, which is now the state-of-the-art method, may provide the greatest chance to reduce this demand. However, again, deploying these transformers in lambda systems is difficult because of the high resource needs of transformer topologies.

With a highly innovative data transmission approach, a transformer-based deep learning model,

and a state-of-the-art big data processing architecture, this research intends to present an LA as an end-to-end big data system for a scalable, fault-tolerant, and high-accuracy phishing email classification problem. The proposed LA-based big data system for phishing email classification problems in combination with pre-trained language models, such as the transformer architectures BERT [33], RoberTa [34], and Conv-BERT [35]. In order to provide a solution to the high resource and specialized hardware requirements, this work focuses, in particular, on the analysis and performance assessment of the unique way of dividing a transformer model into two parts and employing these two partial models in the lambda architecture. This study sets the way for more research in this field by offering a unique transformer model splitting approach to get beyond the resource limitations of deploying transformer models and proving its efficacy through empirical testing.

It should be noted that since we leverage widely used phishing and spam email-based datasets, our present effort is restricted to simply offering an English-language solution. For other phishing/malicious emails sent in other languages, such as Turkish, our suggested technique can be readily manipulated or altered.

2.2. Lambda Architecture based Big Data System

An LA compromises speed, batch, and serving layers in order to analyze incoming data and respond to inquiries on stored historical and newly acquired data [40]. When the serving layer receives a query request, the response is created by simultaneously interrogating both real-time and batch views simultaneously and integrating the data from these levels. Both real-time and batch databases are searched at the serving layer, and the results are combined into a single resultant data set to provide a near-real-time data set in response to the query. A scalable distributed data transmission system (data bus) allows continuous data transfer to batch and speed layers simultaneously. On the speed layer, data processing and analytics are performed in real time, while on the batch layer, they are performed offline. The LA is conceptually represented and shown in Figure 1. Incoming data from the data ingestion bus are transmitted to both the speed and batch layers, which then generate multiple views employing new and old data and store the results on the LA's serving layer. Several extant big data technologies may be utilized at each of the three phases of a LA's construction. According to LA's polyglot persistence paradigm, each available big data technology framework may be used for its specific data processing capacity to handle this type of data and support analytical activities.

The batch layer is responsible for the management, operation, and storage of units of immutable primary data sets. The incoming, very recent data is simply added to the historical data already stored in the batch layer. In the batch layer, update and remove actions are not allowed. As required, continuous data processing and analytics are executed to generate batch views from these data. A new batch view calculation operation is reexecuted sequentially and combined to generate new batch views when coordinated with the speed layer or on a predetermined number of new data arrivals. This process is continuous and unending. The batch views are made up of the immutable data sets of the batch layer. Depending on the volume of both incoming and stored historical data, batch data processing and analytical computations require an excessive amount of time. Consequently, it is uncommon to execute batch layer actions and computations to generate recent batch views. The processing and importation

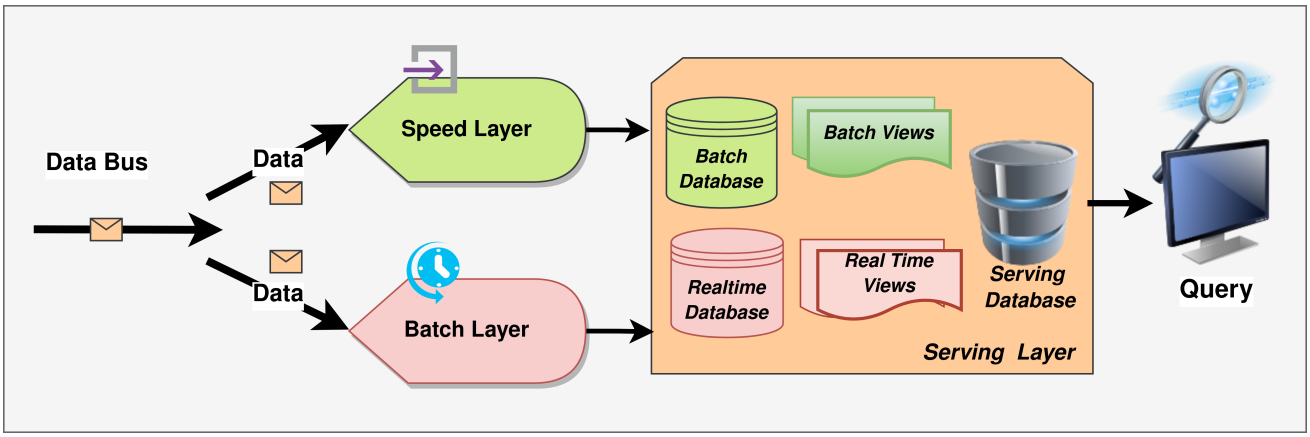

Figure 1. Abstract representation of lambda architecture.

of batch layer data must be monitored to ensure that batch view creation is complete before the speed layer becomes overloaded. The serving layer utilizes real-time and batch views generated by the speed and batch layers in order to respond to incoming queries. As a result, the serving layer must have the ability to store significant amounts of data, such as NoSQL databases with multiple features. Due to the variety of data ingestion patterns, this layer must support both real-time and aggregate data ingestion. In situations where data transmission is delayed or absent, the serving layer is vulnerable. Under these circumstances, inconsistencies may arise in data analyses and query responses, which are ultimately resolved by the batch layer alone [40].

To meet the requirements for low-latency analytics and quick queries, LA's speed layer compensates for the staleness of batch views by serving the most recently collected data that the batch layer has not yet processed. Depending on its limited capacity, the speed layer processes streaming data in real time and stores its output as real-time views in the serving layer.

The speed layer requires extensive read and write operations from the serving layer due to the nature of real-time operation requirements. Real-time views store only recent data until the batch layer completes its operation cycle one or two times (s). After the batch layer completes its data processing and analytics calculation operations, the data stored as real-time views during batch processing is deleted and removed from the serving layer. Depending on the data processing at the batch layer, some real-time views must be purged or cleared from the real-time layer upon completion of batch view generation. This procedure is essential for reducing the strain on the real-time database of the serving layer. Monitoring and acting on the resources of the speed layer depend on the utilization of resources and capacity needs, the precise coordination of the layer, and specific performance indicators at all levels. The batch view is obsolete for at least the processing time between the start and end times of the batch processes, if not longer, and if there are inappropriate conditions or faulty coordination with the speed layer. This requires meticulous coordination between the speed and batch layers. As soon as the coordinated data processing activity between the speed and batch levels has ended, the serving layer must initiate the import of large amounts of data. Data ingestion from the serving layer is complete when the final batch views are available [40].

2.3. Transformer Models

The fundamental technological innovation of BERT [33] for language modeling is the bidirectional training of a Transformer architecture. The Transformer model, in its default setup, comprises two distinct processes. The initial process is the encoder component, which operates on the input tokens derived from the text. The second component is a decoder that generates a prediction for the task by utilizing the information obtained from the encoder. The primary objective of the BERT model is to generate word embeddings for a given language. Consequently, only the encoder component of the transformer architecture is necessary, as it can be employed for various task-specific operations. The BERT model undergoes training in a simultaneous manner, utilizing both the Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) techniques. The utilization of this pretrained model is highly efficient for various downstream NLP applications, including but not limited to classification, intent detection, named entity recognition (NER), natural language inference (NLI), and others. In Table 1, several common pre-trained transformer models with their architectural information are listed.

Table 1.
Several transformer models with their parameter and layer sizes.

Transformer Model	# Layers	# Parameters
BERT-Base	12	110 M
BERT-Large	24	336 M
RoBERTa-Base	12	125 M
RoBERTa-Large	24	355 M
Mobile-BERT	24	24.5 M
ELECTRA-Large	24	330 M
GPT2-Base	12	117 M
GPT2-Large	26	774 M
GPT2-XL	48	1558 M

The BERT [33], represents Bidirectional Encoder Representations from Transformers, is a deep learning model that utilizes Transformers, a framework where each output element is interconnected with every input element. The connections between these elements are dynamically determined, and their respective weightings are calculated accordingly. In the past, language models were limited to processing text input in a sequential manner, either from left-to-right or right-to-left. However, they were unable to simultaneously process both directions of text input at a given moment. The majority of language models possess the ability to process input data in either a left-to-right or right-to-left manner. However, BERT distinguishes itself by concurrently processing input data in both directions. The phenomenon of bi-directionality is facilitated by the utilization of transformers.

The inception of Transformers can be traced back to their initial introduction by Google in the year 2017. The predominant methodologies employed in addressing NLP-related challenges encompass conventional techniques, including Convolutional Neural Networks (CNN), Recursive Neural Networks (RNN), and Long-Short Term Memory (LSTM). Due to the unidirectional nature of these models, wherein input sequences were processed solely from left to right or vice versa, a critical limitation arose: beyond a certain threshold, the input information became irretrievable, resulting in the loss of the paragraph's intended semantic interpretation. The issue at hand is effectively tackled by Transformers as a whole, with BERT specifically being a notable example.

BERT has undergone training using unannotated data, specifically a plain text corpus consisting of the English Wikipedia and a Brown corpus [38]. The utilization of a pre-trained model serves as a foundational model that can be adapted and customized to perform a range of specific tasks as re-

quired. One notable benefit of BERT is its ability to undergo fine-tuning without the need for retraining the entire model. The aforementioned methodology is commonly referred to as transfer learning. By incorporating additional layers into the foundational model, it is possible to customize and optimize the model for a particular task. Consequently, the utilization of the BERT model does not inherently require advanced hardware or an extensive dataset. In addition, the duration required for fine-tuning is significantly reduced. In the classification experiment detailed in this study, a supplementary ”classification layer” was incorporated into the BERT model to facilitate model refinement and accomplish the objective of precise classification.

Before using BERT for various downstream tasks, it is imperative to perform tokenization on the input text utilizing the WordPiece tokenizer [41]. The main rationale behind this is that each tokenizer is designed with distinct strategies to handle unknown tokens. Additionally, the vocabulary and index mapping utilized during training must be compatible. The initial step in the tokenization procedure involves the division of the text into individual words. However, when using this methodology, words that do not belong to the established vocabulary are considered ”unfamiliar.” In order to address this issue, modern NLP models employ a technique known as sub-word tokenization, which involves dividing the text into smaller units that often preserve their inherent linguistic significance, such as morphemes. Hence, despite the potential unfamiliarity of a word to the model, its meaning can still be deduced to some extent by leveraging the information stored in the model’s memory of the constituent sub-word tokens. The WordPiece technique is a commonly employed approach for sub-word tokenization, which exhibits the possibility of being expanded to various other natural language processing (NLP) models. The WordPiece algorithm initially divides the text into words by using punctuation and whitespaces as delimiters. Subsequently, it tokenizes each word into smaller units known as word pieces. The aforementioned procedure is executed using the input text provided. Following the tokenization procedure, it is possible to utilize a specific structure for our input, which involves the incorporation of distinct tokens. As a general practice, it is customary to add distinct tokens at the beginning and end of each input phrase. The tokens in question are commonly referred to as CLS and SEP tokens, respectively. In the context of classification tasks, such as Natural Language Inference (NLI), the [CLS] token, denoting ”classification,” is placed at the beginning of the provided sentence or sequence of sentences. The [SEP] token, which is commonly referred to as the separator token, is utilized to indicate the conclusion of a sentence. Furthermore, utilization of the [PAD] token, commonly referred to as padding, can be employed to effectively handle the maximum allowable input length. One of the primary obstacles in training language models involves the identification of a suitable prediction target task. Numerous models offer predictions regarding the subsequent word in a sequence, employing a directive approach that inherently limits the extent of context learning. BERT utilizes two distinct training mechanisms simultaneously in order to address this challenge: the training of the BERT model through next sentence prediction (NSP) and masked language modeling (MLM) [33]. The objective of the MLM training is to infer the underlying vocabulary of the obscured word solely based on the contextual cues provided within a given text. The selection of masked tokens is performed randomly from the tokenized text provided as input. Prior to being inputted into the BERT model, word sequences undergo a substitution process where approximately 15 percent of the words are replaced with [MASK] tokens. Prior to

inputting the word sequences into the BERT model, this preprocessing step is performed. The model utilizes the contextual information provided by the non-masked phrases in the sentence to predict the true value of the masked words.

Transformer models, such as BERT, have typically undergone pre-training for a wide range of languages and subject domains, although their coverage may not be exhaustive. In instances of limited availability of resources and uncommon languages or specific domains, the creation of a customized transformer model would yield the greatest advantages. Pre-existing models are trained to utilize corpora that are specific to a particular domain or application. Research has demonstrated that training models on corpora specific to a particular domain yield better results when fine-tuning them for downstream natural language processing (NLP) tasks, such as text classification within those domains. This performance surpasses that of fine-tuning BERT, which was originally trained on the BooksCorpus and Wikipedia datasets. In the concluding phase, we proceeded to undergo training on a language modeling (LM) and next sentence prediction (NSP) task prior to further refining the text classification process. Theoretically, the performance of this approach is expected to be favorable as the model can leverage its comprehension of language semantics acquired through initial generative training. The inclusion of an additional fine-tuning step for classification tasks subsequent to language modeling enables the effective capture of long-term interdependencies within the language and facilitates the efficient integration of hierarchical relationships. The inclusion of language modeling (LM) in training can enhance the model's comprehension of language-specific semantics, which may be advantageous. Additionally, LM benefits from a vast corpus of data, unlike text classification tasks that typically have limited access to such resources.

2.4. Proposed Methodology

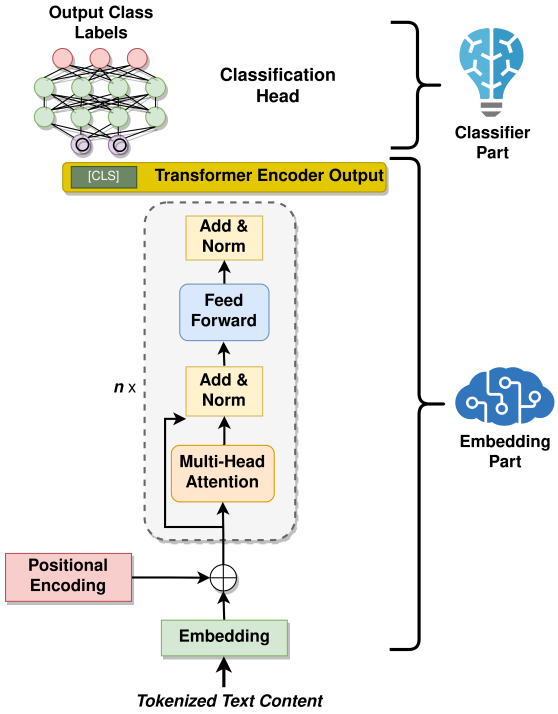

Figure 2. Transformer model for classification

In order to classify and detect phishing attacks from incoming emails, our proposed methodology is shown in Figure 3. In this methodology, several agents are connected to all email servers with different protocols in order to receive and investigate incoming emails from the outside world. By using these agents, incoming mail contexts are preprocessed and then sent to the AI embedding producer service. This service includes pretrained, fine-tuned, highly optimized, scalable, and accelerated versions of a BERT model. This deployed BERT model has a special architecture, and only half of the original model is used in this service as an embedding generator. In Figure 2, we show this novel approach in detail. Original transformer architecture can be split into 2 sub-parts, such as the Embedding part and the classifier part, after training

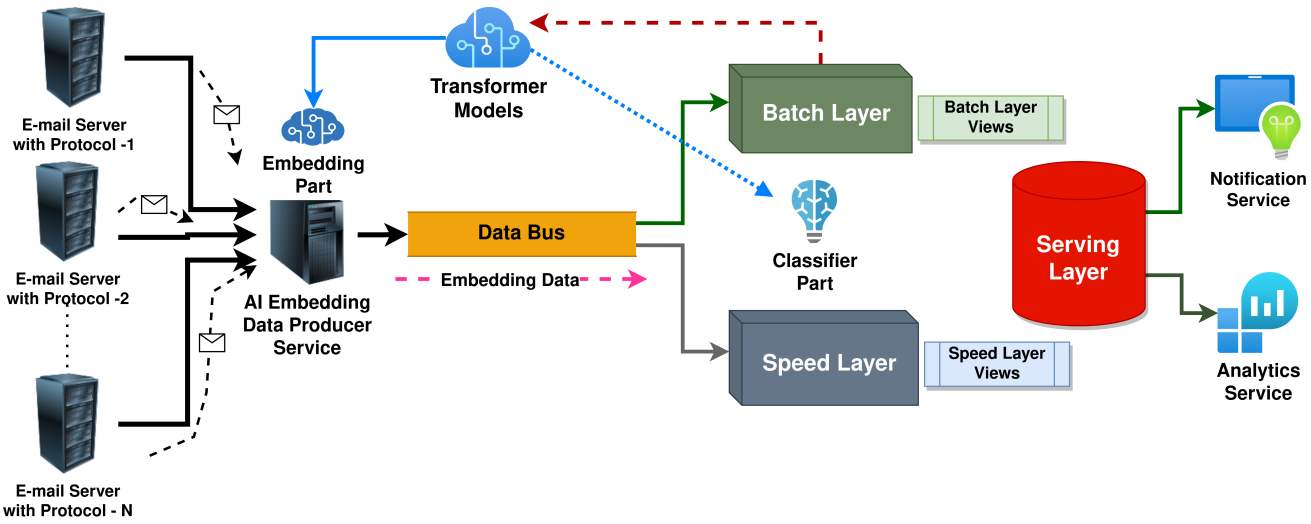

Figure 3. Lambda architecture for phishing email classification.

and fine-tuning operation. Serving the whole model is computationally very high, costly, and requires more resources. As a result, In order to mitigate these problems, we deployed the embedding part into the AI Embedding Data Producer service and the classifier part into the speed and batch layers. The classifier part consists of only a few feed-forward networks. This feature drastically reduces the required resources and computation power. In addition to this, the embedding part accepts the text-based content and then produces the fixed-size of vector as an output token. This special token ([CLS]) represents the whole text-based content in the n dimensional vector space and is completely human-unreadable. This might be interpreted as a cryptographic encryption operation. Based on this analogy, the output of the embedding part can not be understood and is not interpretable; this feature adds an additional security layer to the overall architecture. Thus, no sensitive or private data is transmitted from AI Embedding Data Producer Service to the data bus or anywhere else. This feature also helps to deploy this system to geographically distributed locations. This shows another additional advantage of the proposed methodology.

After the embedding generation step, the AI embedding data producer service sends a 768-dimensional vector representation of the email content to the data bus. The data bus (Apache Kafka) transfers this vector representation to the batch and speed layer for further processing. The 2nd part of the transformer model, namely the classifier part, is hosted in batch and speed layers. In the speed layer, the classifier part, consisting of a simple feed-forward neural network model, receives the incoming data from the data bus and directly applies a feed-forward inference step to calculate target label probabilities. As a result of this operation, the classification results from this model are sent to the serving layer to be used in notification and analytic services.

In parallel to sending the vector representation of the email text content to the speed layer, the same data is also transmitted to the batch layer at the same time. This data is not processed instantly in the batch layer. After a predefined time or predefined resource constraints, the batch layer starts working on these collected data. However, this time, by using the same classification part of the BERT architecture

(exactly the same feed-forward neural network but deployed in the batch layer), all accumulated data is classified as normal or malicious email. Due to batch processing, this operation takes a longer time, and all of the results are sent to the serving layer. When this bulk data is received in the serving layer, if the data already exists in the serving layer, then its existing records are updated. If the data are not already in the serving layer, then it is inserted as new data in the serving layer for querying operations.

As a novel approach, we introduce another task to the batch layer. Different than the classical lambda architecture, model re-training for fine-tuning the deployed transformer model is employed. The main reason for this additional task is the dataset shift and concept shift of the machine learning models [42]. Dataset shift is a prevalent issue in predictive modeling, wherein there is a disparity in the joint distribution of inputs and outputs between the model training and test phases. In addition, covariate shift refers to a specific instance of dataset shift, wherein there is a change solely in the distribution of input variables. Dataset shift is a prevalent phenomenon observed in a majority of practical applications. This occurrence can be attributed to various factors, such as the introduction of bias through experimental design or the inability to reproduce testing conditions during training [43]. One common instance pertains to the application of spam/phishing email filtering, wherein there exists a potential for the filter to exhibit inadequacy in identifying malicious messages that deviate in structure from the spam patterns upon which the automated filter has been trained. As a result, dataset and covariate shift detection and model re-training job is also added to the batch layer. After a predefined time (once a day), if any dataset or covariate shift is detected by using statistical tests and performance metrics, the batch layer immediately starts fine-tuning by using retraining of the current model. Then, the new model is split into two parts as before, and each part is redeployed in the batch layer, speed layer (classification part), and AI embedding data producer service (embedding part). By enforcing this policy, the robustness of the classification model can be improved against new phishing/malicious emails.

2.5. Experiments

In this research, the primary focus is on the development and evaluation of a novel approach to targeted social engineering email detection using Lambda Architecture (LA) and transformer-based models. The experiments were meticulously designed to address the challenges posed by the high computational demands of transformer models, especially when dealing with large-scale email data.

The study introduced a unique method of dividing a transformer model into two distinct parts, allowing for its integration into the Lambda Architecture. This division aimed to overcome the resource-intensive nature of transformer models, making it feasible to deploy them in a big data processing environment. The experiments were grounded in real-world scenarios, utilizing a dataset comprising both malicious and benign emails. This dataset underwent rigorous preprocessing, including the removal of duplicates and the addition of subject fields to the email content.

Two primary classification models were employed in the experiments: fastText and BERT. The fastText model served as the baseline, leveraging its bag of tricks strategy for email categorization. This model was particularly chosen for its efficiency in handling morphologically complex languages and its ability to generate embeddings for a wide range of languages. On the other hand, the BERT model, a transformer-based architecture, was chosen for

its state-of-the-art performance in natural language processing tasks. The experiments were designed to compare the performance of these two models in the context of the LA, especially focusing on their efficiency, accuracy, and resource consumption.

The empirical findings from these experiments were pivotal in demonstrating the effectiveness of the proposed approach. Not only did they validate the feasibility of integrating transformer models into the LA, but they also highlighted the advantages of the novel model-splitting methodology in addressing resource constraints.

The proposed LA-based system is deployed on a local cluster. For cluster hardware, 5 Dell Power Edge R730 and 2 Dell Power Edge R320 servers are used. With this cluster, over 2 TB of RAM, more than 100 cores, and more than 240 TB of disk space are available for data processing and storage.

In the following sections, we explain all the details of these experiments, including the theoretical foundations.

2.5.1 Dataset

Both spam and phishing emails are considered undesirable by the receiver and are sent using very identical methods to the victims. Hence, the primary distinction between spam and phishing emails is in their respective contents. Hence, in the context of classifying spam and phishing emails, the only consideration is the content of the email message body. Supervised learning methodologies need the use of labeled datasets, including spam-phishing and ham (normal) emails [44]. The construction of the dataset included the integration of several distinct datasets, such as the Nazario dataset, which contains a compilation of phishing emails [45], the SpamAssassin dataset, which comprises a collection of spam emails [46], and Enron dataset [16], Spam filter dataset from Kaggle [47], CSDMC2010 dataset [48], and Spam Base [49]. The datasets were accessed by extracting just the email message body and subject content, with a specific extraction code developed for each dataset. Furthermore, these emails underwent preprocessing. The email message body contents were applied to a cleanup operation, during which all HTML, CSS (cascading style sheets), JavaScript code, and special symbols were removed, resulting in unformatted text. Due to the presence of personal information, several emails were also excluded. This measure was implemented in order to prevent email messages from being linked to a particular dataset. For instance, the Nazario dataset often references the email address jose@monkey.org, whereas the Enron dataset mentions the company Enron. Consequently, in order to ensure privacy and confidentiality, any identifiable personal information such as the recipient's name, email address, and organization's name were substituted with generic terms (NAME, EMAIL, ORGANIZATION). Additionally, certain dates, including the year, were omitted from the text. The process of removing personal information from the dataset was carried out using a semi-automated approach. Initially, regular expression expressions were used to exclude some identifiable details. Subsequently, all email addresses were meticulously reviewed and amended manually. Upon conducting formatting and

Table 2.
Datasets for the experiments

	Class Labels	
Dataset ID	Malicious	Benign
[45] Nazario	4598	1000
[46] SpamAssasin	1897	4150
[16] ENRON (4,5,6)	17171	16545
[47] Spam Filter	1368	4360
[48] CSDMC2010	1378	2949
[49] Spam Base	1813	2788

removing personal information, it was discovered that there were instances of duplicated emails. Several occurrences of identical email templates were observed, resulting in the selection of distinct messages for the dataset, with all duplicated copies being eliminated. After this clean-up process, the subject field of all emails was also added at the end of each email. Finally, the new dataset includes 28225 malicious and 31792 benign emails, and the number of emails for each dataset (both malicious and normal) is listed in Table 2 for the final dataset.

It is worth noting that our aim is to classify spam and phishing emails against normal (benign, ham) ones. In this work, we do not try to develop a model that classifies spam vs. phishing emails. Our concept is that both spam and phishing emails are considered undesirable by the recipient and are transmitted using very identical methods to the victims, so both spam and phishing emails will be in one group called malicious in this study. As a result, in this study, our aim is to solve a binary classification problem by distinguishing malicious emails from benign emails.

2.5.2 Classification Models

In this work, we used two different classification methods, such as fastText [50], [51] (as a base model) and a transformer model, BERT [33].

The fastText technique was chosen as the baseline model, and the bag-of-tricks strategy [51] was used to categorize emails. The fundamental idea behind fastText-based classification is that the morphological composition of a word inherently encodes crucial information pertaining to its semantic interpretation. Classic word representation and embeddings, such as Word2Vec, do not have a framework that allows multiple words. Instead, they focused on creating a single-word representation for each individual word. The significance of this matter is particularly apparent in morphologically complex languages such as Turkish, whereby a single word might have several morphemes, each of which may have limited occurrences, hence presenting challenges in effectively acquiring word representations. The fastText approach addresses this concern by seeing each word as an aggregation of its subwords, which are smaller units of characters. To ensure language independence, the subwords considered are the character-level n-grams of the whole word. A word representation refers to the consolidation of all the vectors associated with its constituent character-level n-grams. This feature offers the potential to address both the issue of out-of-vocabulary words (by calculating an embedding vector for each unknown word) and the need for individual vector representations for each word. Furthermore, fastText provides word embeddings for a total of 157 languages, which have been constructed using data from Wikipedia and Common Crawl. The aforementioned characteristics of the fastText-based classifier demonstrate its suitability for addressing our specific challenge, therefore justifying its selection as the baseline model. It is worth noting that this method is only deployed in SL and BL in LA because its architecture is not separable, and it also does not require so many system resources.

In this study, the fastText model was only trained with the dataset mentioned in Section 2.5.1. We divided all data into two 70% training and 30% test datasets by using stratified sampling to ease the adverse effects of the class imbalance problem. In order to find the best hyperparameters for the most accurate fastText model, we performed an extensive hyperparameter search by training all models with each answer sentence in the training dataset and tested the performance of the models with the test set. Hyperparameter search space for fastText model

training is given in Table 3.

Table 3.
Hyperparameter search space (fastText)

Parameters	Range	Step Size
Learning Rate (lr)	$0.01 - 1.2$	0.01
Embedding Dim. (dim)	$50 - 500$	5
Size of the context (ws)	$5 - 8$	1
Epoch	$50 - 1000$	5
Loss	ns, hs, ova, softmax	-
wordNgrams	$1 - 3$	1

The grid search approach was used for doing the hyper-parameter search. The F1 score measure was used in model selection because of the presence of class imbalance within some classes in the datasets. The learning rate parameter was adjusted within the range of 0.01 to 1, with increments of 0.01. The embedding dimension was varied between 50 and 500, with increments of 5. The size of the context window was explored within the range of 5 to 8, with increments of 1. Different loss functions, namely ns, hs, ova, and softmax, were employed to train the models. Lastly, the maximum length of the word n-grams was selected within the range of 1 to 3, with increments of 1. The learning rate update parameter for the training was set to its default value of 100. The result of the hyperparameter search process is given in Table 4. It is worth noting that this model was deployed for BL and SL in the LA system, and all of the receiving emails are sent to the data bus.

Transformer models, like BERT [33], have typically undergone pre-training for several languages and topic areas, but their coverage is not exhaustive. In instances of limited availability of resources and uncommon languages or specialized fields, the creation of a customized transformer model would provide the greatest advantages. Pre-existing models are trained by using corpora that are specialized to a certain area or application. Research has shown that

Table 4.
Best Hyperparameters for the fastText model

Parameters	Range	Step Size
Learning Rate (lr)	0.42	0.01
Embedding Dim. (dim)	300	5
Size of the context (ws)	8	1
Epoch	862	5
Loss	hs	-
wordNgrams	2	1
Train Loss	0.00182	1

training models on corpora that are specialized to a certain domain leads to better performance when fine-tuning them for downstream natural language processing (NLP) tasks, such as text classification within those domains. This performance is seen to be higher compared to fine-tuning BERT, which is pre-trained on a combination of BooksCorpus and Wikipedia. In the concluding phase, we proceeded to undergo training on a language modeling (LM) and next sentence prediction (NSP) challenge prior to improving the text categorization process. Theoretically, the performance of this approach is expected to be favorable due to the model's ability to use its comprehension of language semantics acquired from initial generative training. The inclusion of an additional fine-tuning stage for classification tasks subsequent to language modeling enables the effective capture of long-term interdependencies within the language and facilitates the efficient integration of hierarchical connections. The inclusion of language modeling (LM) in the training process might enhance the model's comprehension of the specific semantic nuances present in the taught language. This is due to the fact that LM has access to a large corpus of data, which is often not available in text classification tasks. In this work, we implemented the text classification natural language processing (NLP) pipeline described before using BERT.

Subsequently, the pre-trained model [52] was utilized to carry out a final fine-tuning process. This involved adapting the pre-trained BERT model to a binary-class text classification task using our datasets, which had a limited corpus for malicious emails. To achieve this, an additional layer of neurons was added to the final layer, which had not been previously trained. The modified architecture was then retrained specifically for our target classification task. As stated above, BERT utilizes the WordPiece tokenizer [41]. Additionally, we conducted training on a tokenizer using a comprehensive data corpus. However, it is important to note that this corpus is not precisely aligned with the linguistic domain of our specific challenge. The data used in this study originate from many email domains and have been sourced from various spam and phishing email datasets. The WordPiece tokenization method autonomously manages the handling of out-of-vocabulary terms. The use of this approach confers a substantial benefit over traditional natural language processing (NLP) pipelines, which often encounter unfamiliar tokens, resulting in a notable loss of information. This methodology enables us to obtain exceptional outcomes despite the constraints imposed by our restricted corpus and datasets, surpassing the performance of the fastText technique.

Using the pre-trained BERT-Base model [52], we performed a final fine-tuning task to classify the incoming email classification. During this stage, the BERT-Base model that had been previously trained was used, with the addition of an additional layer to its architecture, specifically for the purpose of performing the classification job (shown in Figure 2). The embedding vector produced by the [CLS] token is a crucial component in the BERT-Base model. This implies that the classifier is provided with a 768-dimensional embedding vector derived from the [CLS] token as an input. Consequently, the classifier generates an output vector with a dimensionality equivalent to the number of classes in the classification task at hand. Utilizing this embedding as an input to our classifier in the context of a text classification task is satisfactory, rendering additional outputs of the BERT model unnecessary. The search was limited to identifying the optimal learning rate to train the classification model within the range of 1e-6 to 5e-5, using a step size of 1e-6 as a hyperparameter. The batch size was selected as 32, the weight decay was set at 0.01, and all other parameters were left at their default levels. We again split all data into two 70% training and 30% test datasets by using stratified sampling to ease the adverse effects of the class imbalance problem. The model was trained for a limited number of epochs, namely 10, using text data and its labels within the training dataset. The majority of models exhibited signs of overfitting after four epochs, prompting us to implement an early-stopping approach. We trained the network using a binary cross-entropy loss function given in Equation 1. The loss L is defined by the output of our model $f(\mathbf{x};\theta)$ for input $\mathbf{x}$ and the label $y \in \{0,1\}$ and the model parameter θ.

$$\begin{aligned} L(\mathbf{x}, y; \theta) = &-y \log(f(\mathbf{x};\theta)) \\ &+ (1-y) \log(1 - f(\mathbf{x};\theta)) \end{aligned} \tag{1}$$

For fine-tuning the pre-trained BERT model, we solve for $\hat{\theta}$ the optimal set of parameters that minimize the loss over the dataset given in Equation 2:

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} L\left(\mathbf{x_i}, y_i; \theta\right) \tag{2}$$

Where n is the number of samples in our dataset, and $\mathbf{x_i}$ and y_i are the feature vector of the i^{th} training sample and the label respectively. This first part of this model was used in the embedding generation step, where the AI embedding data producer service sends a 768-dimensional vector representation of the

email content to the data bus. The data bus (Apache Kafka) transfers this vector representation to the batch and speed layers for further processing. The second part of the transformer model, namely the classifier part, is hosted in batch and speed layers. In the speed layer, the classifier part, consisting of a simple feedforward neural network model, receives the incoming data from the data bus and directly applies a feedforward inference step to calculate the probabilities of the target label. As a result of this operation, the classification results from this model are sent to the serving layer to be used in the notification and analytic services. From this section onwards, we will name the proposed model as Separated BERT (Sep-BERT) in all remaining sections and subsections.

2.5.3 The Lambda Architecture System

In this study, we build an LA system with different big data technologies. Selected technologies are listed in Table 5. For email transmission, we deployed an email server [53] to send and receive email messages. An email-sending agent was written to send bulk emails to the email server using the previously explained datasets. We deployed 10 of these email-sending agents in parallel to produce serious email traffic. In addition, another email agent receives incoming emails and sends them to the AI embedding data producer service. During the experiments, we used 10 of these receiving agents in parallel to process incoming emails and produce very heavy email traffic. All agents are developed in Python programming language and deployed in docker containers.

Apache Kafka is used as the primary data bus in this LA and is connected to AI embedding data producer service through TCP connections. Spark Streaming + Kafka Integration receiverless technique is used to connect the data bus to the speed layer. Over Apache Kafka, raw data from the AI embedding data producer service are combined in DStreams, followed by ETL and filtering activities at the speed layer. HDFS is also used to save the raw data from Kafka.

Table 5.
Selected Big Data technologies for the lambda architecture

LA Component Name	Selected Framework
Batch Layer	Hadoop / Apache Spark
Speed Layer	Spark Streaming
Serving Layer	Druid
Data Bus	Apache Kafka
Email Server	mailcow:dockerized

In the batch layer, Spark is used to perform ETL and filtering operations, and the results are written again in the HDFS. Following this, Druid's [54] batch data ingestion process is begun using Druid's indexing services. Both the Hadoop indexer task and Druid's indexing service are used to import offline data at the batch layer. The Druid indexing service is utilized if the data are smaller than a few Gigabytes; otherwise, a coordination agent starts the Hadoop indexer task for efficient data ingestion to Druid as a serving layer. After ETL processes, the coordination agent is primarily in charge of monitoring and activating batch ingestion activities based on the size of the data offline. This agent starts an indexing process concerning offline data size when a specific amount of offline data reaches a preset value, and the ETL operation is performed. New segments are created on a regular basis, and query-focused views are created at the serving layer.

Batch ingestion may result in the replacement of some real-time views with batch views that include more up-to-date data. Instances of data delays and recurrent data reception circumstances are often

seen. The use of a data retention policy by Druid enables the deletion of delayed data after a certain time period has elapsed. In addition, any missing data may be repaired by using batch layer ingestion. The integration of a Redis key/value store with the Spark Streaming framework occurs at the speed layer. The procedure to update the classification results involves the use of a publish-subscribe mechanism. In the context of classification visualization, this database is used to store and provide more up-to-date data to freshly connected clients, with a buffer period of 2 minutes. Finally, a Node.js service establishes a connection with Redis and uses WebSocket technology to facilitate the transmission of data to clients. An alternative online service might be used for the purpose of retrieving and examining previous data. Hadoop HDFS is used within a cluster environment to provide fault tolerance and enhance high availability. In the context of Druid, it further functions as a repository for long-term storage. In a clustered computing environment, YARN is used for resource management, whereas ZooKeeper is utilized for coordinating purposes.

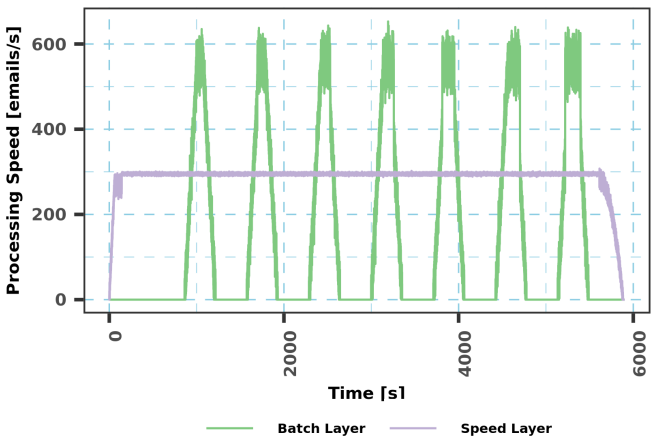

Figure 4. Data processing speed under normal conditions

The data processing rate under full data load conditions is shown in Figure 4. By deploying this design, the batch and speed layers are seen to work as expected in a complementary manner. It is worth noting that the batch layer data processing rate includes both model inference and training. However, the speed layer's data processing rate only includes model inference.

2.5.4 Performance Metrics

In this work, to measure the performance of the binary classification model, we used precision, recall, F_1 score, and accuracy metrics. Precision, which is also known as positive predictive value, refers to the proportion of relevant instances that are included in the retrieved instances. Recall, alternatively referred to as sensitivity, denotes the proportion of pertinent instances that were successfully retrieved. When used independently, precision and recall metrics do not offer significant utility. For example, it is feasible to achieve flawless memory retrieval by retrieving each individual item. Similarly, it is feasible to achieve a high level of precision by opting for a limited quantity of highly probable items. The accuracy metric assesses the average proportion of emails that have been properly categorized throughout the email collection. Matthew's correlation coefficient (MCC) [55] measures the quality of both multi-way and binary classification. It takes into consideration true and false positives and negatives and is generally regarded as a balanced measure that can be used regardless of the classes' varying sizes. The MCC value ranges from -1 to 1. A coefficient of +1 indicates a faultless prediction, 0 is a random prediction of average quality, and -1 is an inverse prediction. In the case of imbalanced data, MCC is preferred over accuracy and F1 score, particularly in binary classification [55]. The mathematical definitions of these metrics are given in Equations 3, 4, 5, 6, and 7.

$$Accuracy = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \quad (3)$$

$$Precission = \frac{T_P}{T_P + F_P} \quad (4)$$

$$Recall = \frac{T_P}{T_P + F_N} \quad (5)$$

$$F_1\ Measure = \frac{2 \cdot T_P}{2T_P + F_P + F_N} \quad (6)$$

$$MCC = \frac{T_P \cdot T_N - F_P \cdot F_N}{\sqrt{(T_P + F_N)(T_P + F_P)(T_N + F_N)(T_N + F_P)}} \quad (7)$$

In these equations, T_P, T_F, F_P, F_N are true positive, true negative, false positive, and false negative values in the specified time window, respectively.

3. Results

In this section, we present the results of our experiments with different classification models for malicious email detection problems by using the LA system. In the first part of the experiment, we investigated the fastText model in the LA system after we trained it with the training part of the data. In the second experiment, we tested and evaluated the accuracy and performance of the LA system by deploying our novel methodology with a pre-trained and fine-tuned BERT model in different sections of the LA. In the final part of our experiments, we share the results of the performance analysis for our LA-based system in order to show the resource efficiency of the proposed approach. We have tested our system under several workload and data ingestion conditions and present numeric values for several metrics, such as CPU usage, RAM usage, and inference time of both BERT and our proposed approach, Separated-BERT (Sep-BERT) models.

3.1. Model Classification Results

After we trained the fastText model with data sampled from 70% of the newly generated dataset, in order to measure the performance of the trained model, we sent these training data as emails to a user's email address created in the mail server. After these emails were processed by LA, the results and performance metrics were calculated in the serving layer. In Table 6, we show the performance metrics for the fastText model.

Table 6.
The fastText classifier results for training data

Labels	Precision	Recall	F_1-Score
Malign	0.8615	0.8684	0.8649
Benign	0.8823	0.8760	0.8791
Accuracy	0.8724		
MCC	0.7441		

The confusion matrix was also calculated by the LA system with the training data for the fastText model and shown in Table 7. As seen from Table 6, the fastText model's precision value was around 0.86, the recall value was near 0.87, and the F_1 score was 0.864 for malign emails. For benign emails, the fastText model managed to reach the precision value of 0.88, the recall value was near 0.88, and the F_1 score was 0.88 for malign emails. Because the dataset has a class imbalance, this model had problems classifying the malign emails against the benign ones. Finally, Matthew's correlation coefficient value was 0.744. As previously stated, in the context of binary classifications, it is preferable to prioritize the ranking derived from the Matthews Correlation Coefficient (MCC). This metric yields a high score when the classifier successfully predicts the majority of positive and negative data instances. As seen from Table 6, even though F_1 score and the other metrics are high, MCC shows us the trained model is not perfect for classifying the malign

emails in the training dataset. After model training,

Table 7.
The fastText classifier confusion matrix for training data

ACTUAL \ PREDICTION	Malicious	Benign
Malicious	**17157**	2601
Benign	2759	**19495**

we sampled a test set from 30% of the newly generated dataset, and we sent these test data as emails to a user's email address created in the mail server. After these emails were processed by LA, results and performance metrics were calculated in the serving layer. In Table 8, we show the performance metrics for the fastText model. The

Table 8.
The fastText classifier results for test data

Labels	Precision	Recall	F_1-Score
Malign	0.8551	0.8573	0.8562
Benign	0.8731	0.8711	0.8720
Accuracy	0.8646		
MCC	0.7283		

confusion matrix was also calculated by the LA system with the training data for the fastText model and shown in Table 9. As seen from Table 8, the trained fastText model's precision value was around 0.85, the recall value was near 0.86, and the F_1 score was 0.85 for malign emails. For benign emails, the fastText model managed to reach the precision value of 0.87, the recall value was near 0.87, and the F_1 score was 0.87 for malign emails. Because the dataset has a class imbalance, this model had problems classifying the malign emails against the benign ones. Finally, Matthew's correlation coefficient value was 0.73. As previously stated, in the context of binary classifications, it is preferable to prioritize the ranking derived from the Matthews Correlation Coefficient (MCC). This metric yields a high score when the classifier successfully predicts the majority of positive and negative data instances. As seen from Table 8, even though F_1 score and the other metrics are high, MCC shows us the trained model is not perfect for classifying the malign emails in the test dataset. In the subsequent phase

Table 9.
The fastText classifier confusion matrix for test data

ACTUAL \ PREDICTION	Malicious	Benign
Malicious	**7259**	1208
Benign	1230	**8307**

of our experiment, we delved into the application of the BERT model, which was uniquely split into two parts: the embedding generator and the classification part. This division was strategically implemented to seamlessly integrate with the Lambda Architecture, ensuring reduced resource consumption. The BERT model, renowned for its deep bidirectional transformers, was pretrained and then fine-tuned to suit the specific requirements of our study.

Following the fine-tuning process of the BERT model, we proceeded to train it using a subset of the newly produced dataset, namely 70% of the data. These training data were then sent through email to a user's email account, which was established inside the mail server. Once the emails were processed by the LA system, calculations were performed at the serving layer to determine the results and performance metrics. The performance indicators for the BERT model are shown in Table 10. The Lambda Architecture (LA) system computed the confusion matrix by using the training data for the BERT model and then displayed it in the serving layer. The results are shown in Table 11. According to the data shown in Table 10, the precision metric

INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE
M. U. Demirezen, Vol.12, No.3, pp.29-59
https://doi.org/10.55859/ijiss.1338813

Table 10.
The BERT classifier results for training data

Labels	Precision	Recall	F_1-Score
Malign	0.9973	0.9993	0.9984
Benign	0.9994	0.9976	0.9985
Accuracy	0.9984		
MCC	0.9969		

for the BERT model was approximately 0.99, the recall metric was close to 0.99, and the F_1 score was calculated as 0.99 for malignant emails. The BERT model achieved a precision value of 0.99, a recall value close to 0.99, and a F_1 score of 0.99 for benign emails. Due to the presence of a class imbalance within the dataset, the model had no difficulty in accurately identifying malignant emails from benign ones. The obtained result for the Matthew correlation coefficient was 0.996. As mentioned above, when considering binary classifications, it is more advantageous to give priority to the ranking obtained from the Matthews Correlation Coefficient (MCC). This statistic shows a significant score when the classifier effectively predicts the majority of positive and negative data occurrences. As seen in Table 10, despite the high values of the F_1 score and other metrics, the MCC measure reveals that the trained model is flawless in its classification of malignant emails within the training dataset and supported the results of classical metrics. Following

Table 11.
The BERT classifier confusion matrix for training data

		PREDICTION	
		Malicious	Benign
ACTUAL	Malicious	**19745**	12
	Benign	53	**22201**

the completion of model training, we proceeded to extract a test set comprising 30% of the recently formed dataset. Subsequently, we sent these test data to the user's designated email address established inside the mail server. Once the emails were processed by the LA system, the serving layer was used to compute the results and performance data. Table 12 presents the performance characteristics of the BERT model in the test data set. The LA

Table 12.
The BERT classifier results for test data

Labels	Precision	Recall	F_1-Score
Malign	0.9985	0.9990	0.9988
Benign	0.9991	0.9987	0.9989
Accuracy	0.9988		
MCC	0.9978		

system computed the confusion matrix for the BERT model using the test data, and the results are shown in Table 13. According to the data shown in Table 12, the precision metric of the BERT model in the test dataset was around 0.998, the recall metric was close to 0.999, and the F_1 score was measured at 0.998 specifically for malignant emails. The BERT model achieved a precision value of 0.999, a recall value close to 0.998, and a F_1 score of 0.998 for benign emails. Although there was a class imbalance within the dataset, the model had no difficulty accurately categorizing malignant emails as opposed to benign ones. The obtained result for the Matthew correlation coefficient was 0.998. As mentioned before, when considering binary classifications, it is more advantageous to give precedence to the ranking obtained from the Matthews Correlation Coefficient (MCC). The aforementioned measure shows a significant value when the classifier effectively predicts the majority of positive and negative data occurrences. As seen in Table 12, in addition to the high values of the F_1 score and other metrics, the MCC metric reveals that the trained model is flawless in its classification of malignant emails in the test data set.

Table 13.
The BERT classifier confusion matrix for test data

ACTUAL \ PREDICTION	Malicious	Benign
Malicious	**8459**	8
Benign	12	**9525**

In the realm of targeted social engineering email detection, comparative analysis between the fastText and BERT models reveals distinct performance differences. The fastText model for malign emails demonstrated a precision of approximately 0.85, a recall of 0.86, and a F_1 score of 0.85. For benign emails, the precision, recall, and F_1 score were around 0.87. The Matthews Correlation Coefficient (MCC) for the fastText model stood at 0.73, indicating a good quality binary classification but with room for improvement.

However, the BERT model, with its intricate architecture, showcased superior metrics. For malign emails, the precision was 0.9985, the recall was 0.9990, and the F_1 score was 0.9988. For benign emails, the precision was 0.9991, the recall was 0.9987, and the F_1 score was 0.9989. The MCC for the BERT model was impressive, 0.9978, suggesting an almost perfect binary classification.

When the overall results were evaluated, we discovered that several factors contributed to the superiority of the BERT model over the fastText model: noitemsep,topsep=0pt

- **Contextual Understanding:** BERT's bidirectional transformers capture the context of words in a sentence more effectively than the fastText model. This deep understanding of context is crucial for detecting the nuances in emails, which can be the difference between identifying a benign and a malicious one.
- **Fine-tuning Capabilities:** BERT's architecture allows for fine-tuning specific tasks, making it adaptable to the unique challenges of email detection. This adaptability ensures that the model can be optimized for the best performance.
- **Resource Efficiency:** By splitting the BERT model into parts for the generation and classification of the embedding, it aligns well with the Lambda Architecture. This ensures efficient real-time processing without compromising performance.
- **Robustness and Generalization:** BERT's extensive training on diverse datasets ensures that it can handle a variety of email structures and contents, making it more reliable in real-world scenarios.

While the fastText model offers commendable performance, the BERT model, with its advanced architecture and fine-tuning capabilities, stands out as the more robust choice for targeted social engineering email detection. Its superior metrics, especially the MCC value, highlight its prowess in accurately classifying emails, making it an invaluable tool in the realm of email security.

3.2. Performance Analysis of The System

In order to investigate the resource consumption of the proposed approach, we have tested our system under several workload conditions and present numeric values for several metrics, such as CPU usage, RAM usage, and inference time of both BERT and our proposed approach, Separated-BERT (Sep-BERT) models according to the email size and ingestion speed. For this experiment, we sampled various emails according to their lengths, between 100 and 1400 words. In addition, we also adjusted the data ingestion speed to range from 0 to 50,000 email/s (0, 1000, 10,000, and 50,000). When the experiment ran all cases for different email sizes and ingestion speeds, we recorded the numeric values for several metrics, such as CPU usage, RAM usage,

and inference time of both BERT and our proposed approach, Separated-BERT (Sep-BERT) models. It is worth noting that this experiment does not aim to measure the models' classification performance or accuracy for the dataset; instead, it aims to measure the hardware and resource performance of the proposed approach. In Figure 5, the memory consumption comparison analysis results for both approaches are shown.

Figure 5 depicts the relationship between email size and memory consumption for both BERT and Sep-BERT models, grouped by ingestion speed: BERT's memory consumption seems to increase with the size of the email. Different ingestion speeds don't show significant variation in memory consumption for the given email sizes. However, Sep-BERT's memory consumption is relatively stable and significantly lower than BERT's across all email sizes. Again, the ingestion speeds don't cause significant variation in memory consumption across the email sizes. In summary, while BERT's memory usage increases with email size, Sep-BERT remains consistent and uses less memory overall. The ingestion speed doesn't appear to have a pronounced effect on memory consumption for either model.

In Figure 6, CPU utilizations and model inference times comparison analysis results for both approaches are shown. In other words, these plots show the relationship between email size and CPU usage, and model inference times for both BERT and Sep-BERT models, grouped by ingestion speed. As the email size increases, the CPU usage for BERT also seems to increase, suggesting a positive correlation. Different ingestion speeds don't show significant variation in CPU usage for the given email sizes. The CPU usage for Sep-BERT also tends to increase as the email size grows. However, the increment seems less pronounced than that of BERT. Just like with BERT, the ingestion speeds don't demonstrate a significant variation in CPU usage across email sizes. As the email size increases, the inference time for BERT also seems to rise. Different ingestion speeds don't display a pronounced variation in inference times for the given email sizes. Sep-BERT's inference times are considerably lower than BERT's across all email sizes. The ingestion speeds also don't manifest a significant variation in inference times across the email sizes. In summary, while BERT's inference time increases with the email size, Sep-BERT processes emails faster and in a more consistent time frame. The ingestion speed doesn't seem to significantly influence inference time for either model.

When all results are evaluated, we see that Sep-BERT maintains a more efficient CPU profile, especially as email sizes grow. Sep-BERT emerges as a more memory-efficient model, making it potentially more scalable and cost-effective for large-scale applications. Despite the growth in email size, Sep-BERT consistently boasts faster inference times than BERT. Sep-BERT, with its swifter processing capabilities, offers advantages in scenarios where real-time or rapid responses are paramount. For both models, varying ingestion speeds didn't introduce significant fluctuations in CPU usage, memory consumption, or inference time. This indicates that both models maintain stable performances irrespective of the rate at which emails are ingested. Finally, our experiment results show that while both BERT and Sep-BERT are robust models capable of processing emails, Sep-BERT demonstrates superior efficiency, especially in terms of CPU usage, memory consumption, and inference time. For applications that prioritize efficiency, scalability, and rapid response, Sep-BERT offers a compelling advantage over BERT.

4. Conclusion

In this research, we delved deep into the realm of cybersecurity, specifically targeting the detection

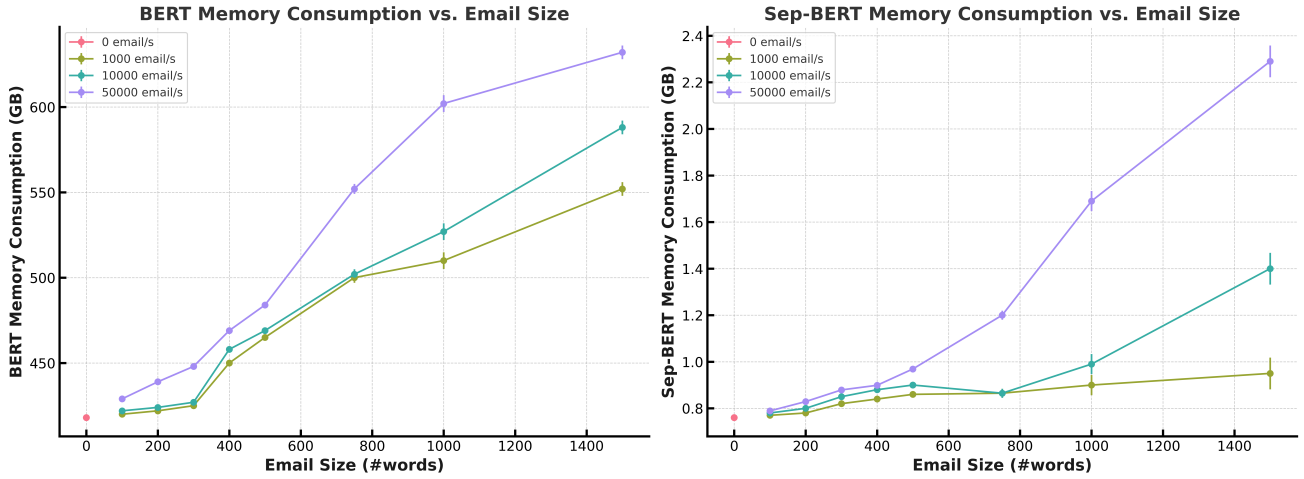

Figure 5. Memory consumption [Giga Bytes] comparisons for BERT and Sep-BERT models.

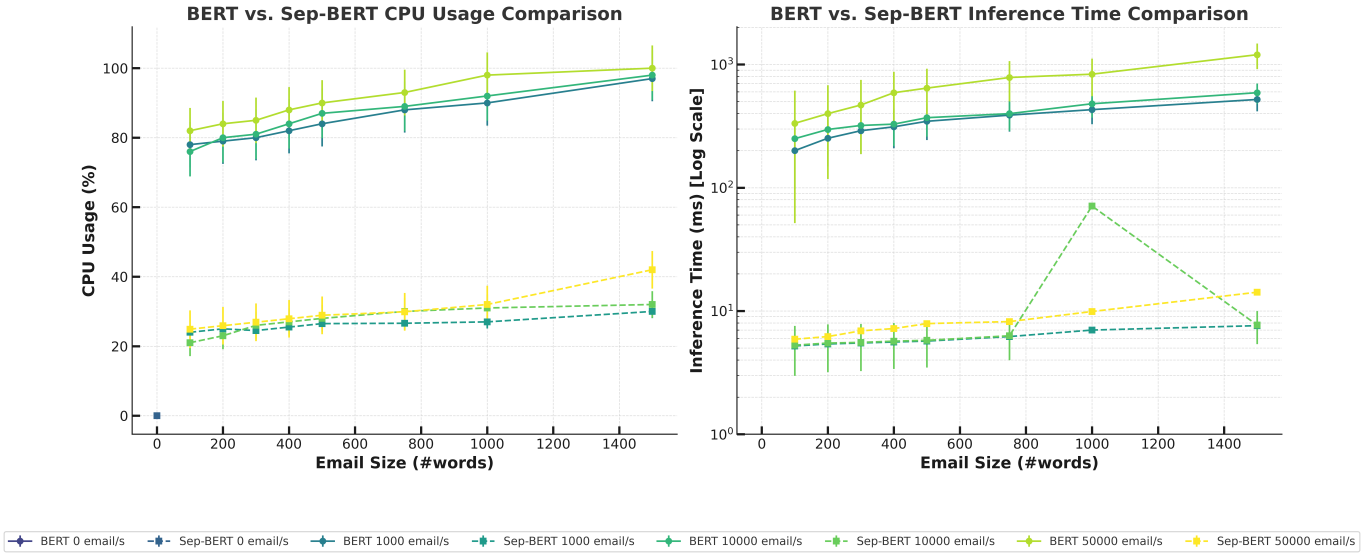

Figure 6. CPU usage (%), and inference time [ms] comparisons for BERT and Sep-BERT.

of social engineering emails using a combination of lambda architecture and transformer-based models. The novelty of this study lies in its unique approach to the problem at hand. Firstly, the introduction of Lambda Architecture (LA) as a comprehensive solution for scalable, fault-tolerant, and high-accuracy classification of targeted social engineering emails is groundbreaking. Secondly, the novel architectural separation of the BERT model into two distinct components, namely the embedding generator part and the classification part, is a pioneering move. This architectural segmentation not only ensures efficient integration within the lambda architecture but also significantly reduces resource consumption, leading to improved system efficiency.

From the empirical findings, the fastText model for malignant emails showed a precision of approximately 0.85, a recall of 0.86, and an F1 score of 0.85. For benign emails, the precision, recall, and F1 score hovered around 0.87. The Matthews Cor-

relation Coefficient (MCC) for the fastText model was 0.73. In contrast, the BERT model exhibited superior performance metrics. For malign emails, the precision was 0.9985, the recall was 0.9990, and the F1 score was 0.9988. For benign emails, the precision was 0.9991, the recall was 0.9987, and the F1 score was 0.9989. The MCC for the BERT model was an impressive 0.9978. These numerical values underscore the superiority of the BERT model over the fastText model in detecting targeted social engineering emails.

In addition, our comprehensive performance analysis revealed the distinct advantages of the Separated-BERT (Sep-BERT) model over the traditional BERT model. Notably, Sep-BERT demonstrated superior efficiency in terms of CPU and memory consumption, especially as email sizes increased. This efficiency translates to faster inference times, making Sep-BERT particularly advantageous for applications requiring real-time responses. Furthermore, Sep-BERT's consistent performance across varying ingestion speeds underscores its potential for scalability and cost-effectiveness in large-scale applications. In contrast, while BERT remains a robust model, its increased resource consumption with growing email sizes may pose challenges in efficiency-driven scenarios. Overall, for applications prioritizing efficiency, scalability, and rapid processing, Sep-BERT emerges as a compelling alternative to BERT.

This research has made significant strides in the domains of big data and cybersecurity. The innovative approach of splitting the transformer model, specifically BERT, and its integration with the Lambda Architecture addresses the high resource requirements of transformer models, making it feasible for real-world applications. However, it is worth noting that this work primarily focuses on the content-based phishing/spam email detection technique, sidelining other machine learning-related features such as IP addresses, URL-based information, and email attachments.

Although the results are promising, there are certain limitations. The study is based solely on the content-based phishing/spam email detection strategy, excluding other potential machine learning features such as IP addresses, URL-based information, and embedded email content. Future research directions could explore the integration of these features into a more holistic email detection system. Additionally, the potential of implementing URL-based features and other email-related machine-learning features could further enhance the system's accuracy and reliability.

In conclusion, this study has not only filled a significant gap in the existing literature but also paved the way for future research in this domain. The novel approach of separating a transformer model and its integration with the Lambda architecture offers a scalable and efficient solution to the ever-growing challenge of targeted social engineering email detection. The empirical results further validate the efficacy of the proposed methodology, marking it as a significant contribution to the fields of big data and cybersecurity.

Acknowledgments

This study is a novel approach to big data application in the cybersecurity domain, and this research is an adapted and technically redesigned version of the previous work of the authors in [40]. The lambda architecture of the previous work has been redesigned, improved, and applied to the "Large-Scale Targeted Social Engineering Email Detection" problem with a novel transformer separation approach as a cybersecurity application.

References

[1] A. Papanikolaou, A. Alevizopoulos, C. Ilioudis, K. Demertzis, and K. Rantos, "A blockchained automl network traffic analyzer

to industrial cyber defense and protection," *Electronics*, vol. 12, no. 6, 2023.

[2] G. Manogaran, C. Thota, D. Lopez, and R. Sundarasekar, "Big data security intelligence for healthcare industry 4.0," *Cybersecurity for Industry 4.0: Analysis for Design and Manufacturing*, pp. 103–126, 2017.

[3] A. Papanikolaou, A. Alevizopoulos, C. Ilioudis, K. Demertzis, and K. Rantos, "An automl network traffic analyzer for cyber threat detection," *International Journal of Information Security*, pp. 1–20, 2023.

[4] Y. Wang, W. Ma, H. Xu, Y. Liu, and P. Yin, "A lightweight multi-view learning approach for phishing attack detection using transformer with mixture of experts," *Applied Sciences*, vol. 13, no. 13, 2023.

[5] J. Ramprasath, S. Priyanka, R. Manudev, and M. Gokul, "Identification and mitigation of phishing email attacks using deep learning," in *2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)*, 2023, pp. 466–470.

[6] A. Mughaid, S. AlZu'bi, A. Hnaif, S. Taamneh, A. Alnajjar, and E. A. Elsoud, "An intelligent cyber security phishing detection system using deep learning techniques," *Cluster Computing*, vol. 25, no. 6, pp. 3819–3828, 2022.

[7] B. Naqvi, K. Perova, A. Farooq, I. Makhdoom, S. Oyedeji, and J. Porras, "Mitigation strategies against the phishing attacks: A systematic literature review," *Computers & Security*, vol. 132, p. 103387, 2023.

[8] T. Muralidharan and N. Nissim, "Improving malicious email detection through novel designated deep-learning architectures utilizing entire email," *Neural Networks*, vol. 157, pp. 257–279, 2023.

[9] S. T. Singh Surinder Pal Singh, M. D. Gabhane, and C. Mahamuni, "Study of machine learning and deep learning algorithms for the detection of email spam based on python implementation," in *2023 International Conference on Disruptive Technologies (ICDT)*, 2023, pp. 637–642.

[10] S. A. Khan, W. Khan, and A. Hussain, "Phishing attacks and websites classification using machine learning and multiple datasets (a comparative analysis)," in *Intelligent Computing Methodologies: 16th International Conference, ICIC 2020, Bari, Italy, October 2–5, 2020, Proceedings, Part III 16*. Springer, 2020, pp. 301–313.

[11] S. Salloum, T. Gaber, S. Vadera, and K. Shaalan, "Phishing email detection using natural language processing techniques: A literature survey," *Procedia Computer Science*, vol. 189, pp. 19–28, 2021, aI in Computational Linguistics.

[12] A. Livara and R. Hernandez, "An empirical analysis of machine learning techniques in phishing e-mail detection," in *2022 International Conference for Advancement in Technology (ICONAT)*, 2022, pp. 1–6.

[13] P. Mehdi Gholampour and R. M. Verma, "Adversarial robustness of phishing email detection models," in *Proceedings of the 9th ACM International Workshop on Security and Privacy Analytics*, ser. IWSPA '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 67–76.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[15] H. Milner and M. Baron, "Establishing an optimal online phishing detection method: Evaluating topological nlp transformers on text message data," *Journal of Data Science & Intelligence system.*, pp. 1–17, 2023.

[16] Y. Guo, Z. Mustafaoglu, and D. Koundal, "Spam detection using bidirectional transformers and machine learning classifier algorithms," *Journal of Computational and Cognitive Engineering*, vol. 2, no. 1, pp. 5–9, 2023.

[17] A. Dima, S. Ruseti, D. Iorga, C. K. Banica, and M. Dascalu, "Multi-task romanian email classification in a business context," *Information*, vol. 14, no. 6, 2023.

[18] F. Ullah and M. A. Babar, "On the scalability of big data cyber security analytics systems," *Journal of Network and Computer Applications*, vol. 198, p. 103294, 2022.

[19] M. A. Amanullah, R. A. A. Habeeb, F. H. Nasaruddin, A. Gani, E. Ahmed, A. S. M. Nainar, N. M. Akim, and M. Imran, "Deep learning and big data technologies for iot security," *Computer Communications*, vol. 151, pp. 495–517, 2020.

[20] K. Demertzis, N. Tziritas, P. Kikiras, S. L. Sanchez, and L. Iliadis, "The next generation cognitive security operations center: Adaptive analytic lambda architecture for efficient defense against adversarial attacks," *Big Data and Cognitive Computing*, vol. 3, no. 1, 2019.

[21] B. Bansal, V. N. Jenipher, R. Jain, R. Dilip, M. Kumbhkar, S. Pramanik, S. Roy, and A. Gupta, *Big Data Architecture for Network Security*. John Wiley & Sons, Ltd, 2022, ch. 11, pp. 233–267.

[22] M. Aschi, S. Bonura, N. Masi, D. Messina, and D. Profeta, *Cybersecurity and Fraud Detection in Financial Transactions*. Cham: Springer International Publishing, 2022, pp. 269–278.

[23] R. Alghamdi and M. Bellaiche, "A deep intrusion detection system in lambda architecture based on edge cloud computing for iot," in *2021 4th International Conference on Artificial Intelligence and Big Data (ICAIBD)*, 2021, pp. 561–566.

[24] R. Alghamdi and M. a. Bellaiche, "An ensemble deep learning based ids for iot using lambda architecture," *Cybersecurity*, vol. 6, no. 1, p. 5, 2023.

[25] N. Martz and J. Warren, *Big Data Principles and Best Practices Of Scalable Realtime Data Systems*. New York, CA: Manning, 2015.

[26] N. Spangenberg, M. Wilke, and B. Franczyk, "A big data architecture for intra-surgical remaining time predictions," *Procedia computer science*, vol. 113, pp. 310–317, 2017.

[27] B. Twardowski and D. Ryzko, "Multi-agent architecture for realtime big data processing," *ACM International Joint Conferences of Web Intelligence and Intelligent Agent Technologies (IAT)*, pp. 333–337, 2014.

[28] S. Nadal, V. Herrero, O. Romero, A. Abelló, X. Franch, S. Vansummeren, D. Valerio *et al.*, "A software reference architecture for semantic-aware Big Data systems," *Information and Software Technology*, vol. 90, pp. 75–92, 2017.

[29] D. S. Terzi, M. U. Demirezen, and S. Sagiroglu, "Evaluations Of Big Data Processing," *Services Transactions on Big Data*, vol. 3, no. 1, pp. 44–52, 2016.

[30] A. Roukh, F. N. Fote, S. A. Mahmoudi, S. Mahmoudi *et al.*, "Big Data Processing Architecture for Smart Farming," *Procedia Computer Science*, vol. 177, pp. 78–85, 2020.

[31] V. Psomakelis, K. Tserpes, D. Zissis, D. Anagnostopoulos, T. Varvarigou *et al.*, "Context agnostic trajectory prediction based on λ-architecture," *Future Generation Computer Systems*, vol. 110, pp. 531–539, 2020.

[32] H. Zheng, S. Oh, H. Wang, P. Briggs, J. Gai, A. Jain, Y. Liu, R. Heaton, R. Huang, and Y. Wang, "Optimizing memory-access patterns for deep learning accelerators," *ArXiv*, vol. abs/2002.12798, 2020. [Online]. Available: https://arxiv.org/abs/2002.12798

[33] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[34] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *ArXiv*, vol. abs/1907.11692, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[35] Z.-H. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, and S. Yan, "Convbert: Improving bert with span-based dynamic convolution," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 837–12 848, 2020.

[36] A. Sanla and T. Numnonda, "A Comparative Performance of Real-time Big Data Analytic Architectures," in *2019 IEEE 9th International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2019, pp. 1–5.

[37] M. Garriga, G. Monsieur, and D. Tamburri, "Big data architectures," in *Data Science for Entrepreneurship: Principles and Methods for Data Engineering, Analytics, Entrepreneurship, and the Society*, W. Liebregts, W.-J. van den Heuvel, and A. van den Born, Eds. Springer International Publishing, 2023, pp. 63–76.

[38] B. Karki, F. Abri, A. S. Namin, and K. S. Jones, "Using transformers for identification of persuasion principles in phishing emails," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022, pp. 2841–2848.

[39] B. Gogoi and T. Ahmed, "Phishing and fraudulent email detection through transfer learning using pretrained transformer models," in *2022 IEEE 19th India Council International Conference (INDICON)*, 2022, pp. 1–6.

[40] M. U. Demirezen, "Büyük veri uygulamaları için bir lamda mimari geliştirilmesi / developing a lambda architecture for big data processing applications,," Ph.D. dissertation, Gazi University, Ankara, 2015.

[41] X. Song, A. Salcianu, Y. Song, D. Dopson, and D. Zhou, "Fast wordpiece tokenization," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 2089–2103.

[42] J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[43] F. Jáñez-Martino, R. Alaiz-Rodríguez, V. González-Castro, E. Fidalgo, and E. Alegre, "A review of spam email detection: analysis of spammer strategies and the dataset shift problem," *Artificial Intelligence Review*, vol. 56, no. 2, pp. 1145–1173, 2023.

[44] J. Rastenis, S. Ramanauskaitė, I. Suzdalev, K. Tunaitytė, J. Janulevičius, and A. Čenys, "Multi-language spam/phishing classification by email body text: Toward automated security incident investigation," *Electronics*, vol. 10, no. 6, 2021.

[45] J. Nazario, "The online phishing corpus," 2004, (accessed Jun. 1, 2023). [Online]. Available: https://monkey.org/~jose/phishing/

[46] T. Gangavarapu, C. Jaidhar, and B. Chanduka, "Applicability of machine learning in spam and phishing email filtering: review and approaches," *Artificial Intelligence Review*, vol. 53, pp. 5019–5081, 2020.

[47] I. AbdulNabi and Q. Yaseen, "Spam email detection using deep learning techniques," *Procedia Computer Science*, vol. 184, pp. 853–858, 2021.

[48] P. Liu and T.-S. Moh, "Content based spam e-mail filtering," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*. IEEE, 2016, pp. 218–224.

[49] M. Hopkins, E. Reeber, G. Forman, and J. Suermondt, "Spambase data set," *Hewlett-Packard Labs*, vol. 1, no. 7, 1999.

[50] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *ArXiv*, vol. abs/1607.01759, 2016. [Online]. Available: https://arxiv.org/abs/1607.01759

[51] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.

[52] Hugging Face, "Hugging Face Pretrained BERT Model," Accessed: 01.06.2023. [Online]. Available: https://huggingface.co/bert-base-uncased

[53] Mailcow dockerized, "Mailcow dockerized open source mail server," Accessed: 12.05.2023, 2023. [Online]. Available: https://github.com/mailcow/mailcow-dockerized

[54] F. Yang, E. Tschetter, G. Merlino, N. Ray, X. Léauté, D. Ganguli, H. Singh *et al.*, "Druid: a real-time analytical data store," *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pp. 157–168, 2014.

[55] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, 2020.