ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo fin de Máster

Máster Universitario en Ingeniería Informática

Análisis empírico y comparativo de técnicas de detección automática de phishing en correos electrónicos

Realizado por:
Alberto Sánchez Abad

Dirigido por:
Rafael Ceballos Guerrero
Rafael Martínez Gasca

Departamento
Lenguajes y Sistemas Informáticos

Convocatoria 2
Curso 2024/2025

Madrid, 02/07/2025

Agradecimientos

En primer lugar, me gustaría expresar mi más profundo agradecimiento a mi familia, la cual es el pilar fundamental de mi vida. A mis padres, por su apoyo y amor incondicional, por cuidarme siempre y por la educación que me han dado. A mi hermana, por ser la compañera perfecta de máster y de vida, por ser una fuente constante de inspiración y por hacerme abrir los ojos muchas veces. A mi novia, por estar a mi lado en cada paso de este proceso, por su comprensión durante las largas horas de trabajo y por ser mi refugio cuando lo he necesitado.

También me gustaría reconocer a todos mis amigos que han sido comprensivos con mis ausencias y gracias a los cuales he encontrado momentos para poder desconectar y disfrutar de momentos maravillosos.

Mi sincero agradecimiento se extiende a mis tutores, cuya orientación académica, sus valiosos consejos y su tiempo han enriquecido enormemente este proyecto. Especialmente agradezco que hayan diseñado una propuesta para mí específicamente que me motivase y que tuviese de ingredientes principales la IA y la ciberseguridad, mis dos grandes pasiones.

Finalmente, gracias a todas las personas que, de una u otra manera, han contribuido a hacer posible este logro.

Resumen

En la actualidad, el phishing por correo electrónico representa una de las amenazas de ciberseguridad más arraigadas, con más de 3.400 millones de correos enviados diariamente en todo el mundo. Además, su evolución constante es alarmante. Este trabajo desarrolla y compara distintas técnicas para detectar automáticamente estos ataques en correos electrónicos.

Se emplea una colección de 11 datasets con 218.086 registros de correos recopilados a lo largo de tres décadas distintas. La distribución es bastante equilibrada entre phishing (47,88 %) y legítimos (52,11 %). Además, esta diversidad permite verificar la verdadera capacidad de generalización de las técnicas implementadas gracias a su notable heterogeneidad.

Después de examinar las tecnologías de detección existentes, se implementan tres familias de técnicas: heurísticas basadas en reglas predefinidas, algoritmos clásicos de machine learning (SVM, Random Forest, Naive Bayes y Regresión Logística) y modelos de deep learning (BERT, LSTM y BiLSTM). Un aspecto clave es que todas funcionan de forma completamente autónoma, sin depender de servicios externos. Sin embargo, cada técnica requiere preprocesamientos específicos y optimización para alcanzar su máxima efectividad.

Los resultados revelan diferencias significativas entre los distintos enfoques. Las técnicas heurísticas presentan limitaciones importantes, mostrando una fuerte dependencia del tipo concreto de phishing analizado. Por otro lado, los algoritmos de machine learning superan claramente a las heurísticas, con métricas superiores al 95 % de efectividad. SVM destaca como el más sólido, alcanzando un 98,03 % en datasets combinados. Por su parte, las técnicas de deep learning logran el mejor rendimiento: BiLSTM obtiene los resultados más destacados (99,31 %), seguido por LSTM (99,09 %) y BERT (98,86 %).

Estos resultados sugieren que los algoritmos de machine learning tradicional ofrecen un balance ideal entre efectividad y eficiencia computacional. Resultan especialmente apropiados para entornos con recursos limitados, tanto en entrenamiento como en evaluación en tiempo real. En contraste, las técnicas de deep learning proporcionan la efectividad superior que requieren entornos críticos donde minimizar falsos negativos es prioritario, aunque demanden mayor potencia computacional.

Abstract

Currently, email phishing represents one of the most entrenched cybersecurity threats, with over 3.4 billion emails sent daily worldwide. Moreover, its constant evolution is alarming. This work develops and compares different techniques for automatically detecting these attacks in emails.

A collection of 11 datasets with 218,086 email records gathered across three different decades is employed. The distribution is fairly balanced between phishing (47.88 %) and legitimate emails (52.11 %). Additionally, this diversity allows verification of the true generalization capability of the implemented techniques thanks to its notable heterogeneity.

After examining existing detection technologies, three families of techniques are implemented: heuristics based on predefined rules, classical machine learning algorithms (SVM, Random Forest, Naive Bayes, and Logistic Regression), and deep learning models (BERT, LSTM, and BiLSTM). A key aspect is that all operate completely autonomously, without depending on external services. However, each technique requires specific preprocessing and optimization to achieve maximum effectiveness.

The results reveal significant differences between the different approaches. Heuristic techniques present important limitations, showing strong dependence on the specific type of phishing analyzed. On the other hand, machine learning algorithms clearly outperform heuristics, with metrics exceeding 95 % effectiveness. SVM stands out as the most robust, achieving 98.03 % on combined datasets. For their part, deep learning techniques achieve the best performance: BiLSTM obtains the most outstanding results (99.31 %), followed by LSTM (99.09 %) and BERT (98.86 %).

These results suggest that traditional machine learning algorithms offer an ideal balance between effectiveness and computational efficiency. They are especially appropriate for environments with limited resources, both in training and real-time evaluation. In contrast, deep learning techniques provide the superior effectiveness required by critical environments where minimizing false negatives is a priority, although they demand greater computational power.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

El phishing por correo electrónico constituye una modalidad de fraude digital donde los ciberdelincuentes envían mensajes falsos haciéndose pasar por organizaciones de confianza. Estos estafadores explotan tanto la confianza como la presión temporal para manipular a sus víctimas. Su objetivo es claro: obtener información confidencial como credenciales de acceso o datos financieros.

La mecánica es engañosamente simple. Normalmente, los atacantes incluyen enlaces maliciosos que redirigen hacia sitios web fraudulentos, réplicas casi perfectas de las plataformas originales. Una vez allí, las víctimas introducen sus datos creyendo estar en el sitio legítimo. Sin embargo, lo que realmente hace devastadores estos ataques es su apariencia de autenticidad: logotipos corporativos, diseños profesionales y un lenguaje que imita fielmente el estilo comunicativo de las empresas reales.

Cabe aclarar que, a lo largo de este trabajo, cuando se mencione el término *"phishing"* se entenderá que se refiere específicamente al phishing por correo electrónico, salvo indicación expresa en contrario.

Este capítulo presenta una introducción al proyecto. Primero, contextualiza la problemática actual del phishing, estableciendo así las razones que motivan esta investigación. Además, se definen los objetivos que se persiguen: tanto los principales como los personales, profesionales y académicos que se esperan alcanzar. Finalmente, se describe la estructura del documento, facilitando de esta manera la navegación y comprensión del contenido por parte del lector.

1.1– Contexto

El phishing se ha convertido en una de las amenazas de la ciberseguridad más comunes y peligrosas hoy en día. En 2025, los ataques han aumentado de manera alarmante, con unos 3400 millones de correos maliciosos enviados cada día en todo el mundo [1]. Además, esto se vuelve aún más preocupante por el nivel de sofisticación e ingenio que están usando los atacantes actualmente.

No solo se están enfocando en el correo electrónico, también están utilizando nuevas técnicas como el smishing, que es el phishing por SMS. Este ataque supone el 45 % de las amenazas móviles actuales con un incremento del 22 % en el último trimestre del 2024 [2]. El vishing, que involucra fraudes por llamadas de voz, ha aumentado un 442 % en la segunda mitad del año 2024 [3] debido a que está usando inteligencia artificial para crear voces muy realistas, lo cual hace que sea mucho más fácil engañar a la gente.

El daño económico del phishing sigue siendo muy alto para empresas de todos los tamaños y organizaciones públicas. Un claro ejemplo es que en marzo de 2024, el Banco Central de Bangladesh sufrió un ataque de phishing que resultó en el robo de 101 millones de dólares, cuando empleados del departamento financiero respondieron a correos electrónicos fraudulentos que simulaban comunicaciones oficiales del sistema SWIFT [4]. Este caso demuestra que hasta las instituciones financieras más seguras pueden ser vulnerables a ataques de ingeniería social sofisticados.

Los atacantes están usando tecnología avanzada para mejorar sus tácticas. Están empleando modelos de lenguaje como ChatGPT para crear mensajes que parecen personalizados, logrando tasas de aciertos del 54 % [5].

La situación se complica más con el surgimiento de plataformas de Phishing-as-a-Service (PhaaS), que ofrecen herramientas avanzadas para los atacantes. Estas plataformas tienen kits que ayudan a sortear sistemas de autenticación y aprovechar vulnerabilidades en servicios como Microsoft 365.

Recientes ataques han puesto el foco en plataformas de trabajo colaborativo. Microsoft Teams, por ejemplo, ha sido un blanco importante, donde enlaces maliciosos disfrazados de actualizaciones han afectado a 618 organizaciones en seis meses [6]. Esto muestra cómo los atacantes se están adaptando a las nuevas formas de trabajo remoto e híbrido.

La persistencia del phishing queda clara en las estadísticas: el 80 % de los incidentes de seguridad se originan en ataques de phishing, y el 74 % proviene de ataques de ingeniería social [7]. Esto resalta la urgencia de crear mejores métodos para detectar y adaptarse a esta amenaza en constante cambio.

1.2– Descripción y motivación

La detección automática de correos de phishing es un gran reto en la ciberseguridad hoy en día. Aunque hay varios métodos para afrontar esto, muchas soluciones tienen limitaciones que afectan su efectividad y uso práctico.

Los atacantes han mejorado sus métodos para evadir los sistemas de detección que conocemos. Ahora usan dominios nuevos, certificados SSL válidos y técnicas de enmascaramiento que hacen que las técnicas clásicas que suelen depender de servicios externos, como las listas negras o verificaciones de terceros, puedan cometer más errores en su diagnóstico. Además, muchas propuestas solo miran las URL o el contenido del mensaje por separado, sin tener en cuenta cómo se relacionan las distintas partes del mismo correo.

La gestión de datasets variados también es un gran desafío. Los conjuntos de datos de phishing que se pueden encontrar a menudo tienen estructuras inconsistentes y formatos diferentes, lo que complica la creación de modelos que funcionen bien en situaciones reales con datos variados.

La motivación principal de este trabajo surge de la necesidad de desarrollar un enfoque completamente autónomo que no dependa de servicios externos y que pueda procesar de manera eficaz grandes volúmenes de correos electrónicos. El auge de la inteligencia artificial representa una oportunidad para intentar superar algunas de las limitaciones actuales y mejorar las tasas de detección.

1.3– Objetivos

El objetivo principal de este trabajo es comparar y evaluar diversas técnicas para la detección automática de phishing en correos electrónicos. Se busca identificar las configuraciones más efectivas y analizar su desempeño en conjuntos de datos heterogéneos, con la meta de lograr precisiones elevadas. Esta investigación se enfocará en técnicas que permitan un sistema de detección de phishing completamente autónomo, que funcione sin depender de servicios externos (a través de Internet), basándose solo en el contenido y los metadatos de los emails. Este objetivo general se alcanzará mediante los siguientes subobjetivos:

- **Subobjetivo 1:** Seleccionar, integrar y preparar datasets heterogéneos de correos electrónicos legítimos y de phishing, desarrollando estrategias de combinación que permitan evaluar la capacidad de generalización de los modelos.
- **Subobjetivo 2:** Implementar y evaluar diferentes métodos de detección, incluyendo técnicas heurísticas basadas en reglas específicas, modelos tradicionales de machine learning y modelos de deep learning, optimizando sus parámetros para maximizar el rendimiento de detección.
- **Subobjetivo 3:** Realizar una evaluación comparativa en profundidad de todas las técnicas implementadas utilizando métricas estándar (*accuracy*, *precision*, *recall*, *F1-score*), analizando su desempeño para identificar la más efectiva.

1.3.1. Objetivos personales

En lo personal, este proyecto representa una excelente ocasión para continuar investigando en el campo de la ciberseguridad y de la inteligencia artificial, áreas que me apasionan y que están creciendo exponencialmente. Adquirir conocimientos y mejorar en machine learning y deep learning aplicados a problemas de seguridad es un objetivo importante para especializarme en un sector que tiene mucha demanda.

También, trabajar con grandes cantidades de datos me puede proporcionar habilidades valiosas en big data y analítica avanzada.

1.3.2. Objetivos profesionales

Este TFM busca desarrollar competencias que el mercado laboral actual valora enormemente. La capacidad de diseñar, implementar y evaluar sistemas de detección automática de amenazas representa una ventaja competitiva significativa. Especialmente en ciberseguridad, donde la demanda de profesionales que dominen inteligencia artificial crece constantemente.

Además, poder comparar metodologías de manera sistemática proporciona una perspectiva práctica muy enriquecedora para futuras decisiones técnicas. Esto permite elegir la solución más apropiada según los requerimientos específicos de cada organización en contextos particulares. Por otro lado, el dominio de técnicas de preprocesamiento y optimización de modelos trasciende este ámbito ya que se pueden aplicar a múltiples problemas de clasificación, independientemente del sector.

1.3.3. Objetivos académicos

A nivel académico, esta investigación integra y consolida el aprendizaje del máster. Particularmente, conecta conocimientos de asignaturas clave como Análisis de Datos en Sistemas de Información, Aprendizaje Automático, Big Data Engineering, Ciberseguridad y Deep Learning.

Con este proyecto, se desarrollan competencias fundamentales para la investigación: diseño experimental riguroso, aplicación de métricas de evaluación precisas e interpretación objetiva de resultados. Estas capacidades constituyen una base sólida tanto para estudios doctorales como para una trayectoria profesional en investigación.

1.4– Estructura de la memoria

En este primer capítulo, se presenta el problema del phishing, explicando por qué se realizó esta investigación y qué se busca con ella. También se describe cómo está organizado

el documento, el cual cuenta con seis capítulos que detallan el proceso de investigación.

El segundo capítulo se enfoca en la planificación y metodología del proyecto. Aquí se detalla cómo se repartieron las horas en el trabajo final, desde la recolección de datos hasta la escritura de la memoria. También se incluye una estimación del coste del proyecto, considerando los recursos humanos, computacionales y el software que se utilizó para implementar y evaluar las técnicas de detección.

El tercer capítulo habla sobre el estado del arte y las tecnologías que se usan, revisando distintas técnicas para detectar phishing. Se analizan los métodos tradicionales basados en reglas y las nuevas propuestas que utilizan machine learning y deep learning.

En el cuarto capítulo, se describe el desarrollo de la solución, explicando por qué se eligió ese enfoque y cómo se diseñó el pipeline, desde recolectar y preparar datos hasta las implementaciones que usan técnicas heurísticas e inteligencia artificial. También se discuten las decisiones de diseño, como la gestión de diferentes conjuntos de datos y la integración de varios algoritmos.

El quinto capítulo presenta los resultados de las evaluaciones de las técnicas implementadas, incluyendo gráficos, tablas y análisis que muestran el rendimiento de cada enfoque. Se analiza la efectividad con las métricas de accuracy, precisión, recall y F1-score en todas las configuraciones.

Finalmente, el sexto capítulo ofrece las conclusiones del trabajo y algunas ideas para el futuro. Se comparan los objetivos iniciales con lo logrado, explicando las diferencias. Las conclusiones resumen los hallazgos más importantes, señalando las técnicas más efectivas y las mejores configuraciones para detectar phishing. Además, se revisa si se cumplieron los objetivos personales, profesionales y académicos, sugiriendo áreas para seguir investigando.

CAPÍTULO 2

Planificación y metodología de trabajo

Este capítulo se centra en cómo se planificó el tiempo en la fase inicial y la metodología usada para el proyecto de investigación. Se detalla la planificación inicial propuesta para las 300 horas asignadas al TFM, describiendo las diferentes fases del proyecto desde la investigación bibliográfica hasta la redacción de la memoria final. Esta distribución del tiempo ayuda a ver el orden de las actividades para alcanzar los objetivos y administrar bien los recursos.

La metodología que se estableció fue bastante flexible para adaptarse a los hallazgos durante el proyecto. También se incluye una estimación de los costes del proyecto, considerando tanto los recursos informáticos como las herramientas de software que se necesitan para hacer la investigación de manera efectiva.

2.1– Planificación inicial

Esta planificación temporal sirvió como referencia inicial, aunque posteriormente experimentó reajustes y modificaciones durante la ejecución real del proyecto en función de las dificultades encontradas y los hallazgos obtenidos durante el desarrollo, estas desviaciones son explicadas en la Sección 6.1. La planificación inicial del proyecto se estructuró en seis fases principales, distribuyendo las 300 horas asignadas al TFM de manera equilibrada entre las siguientes actividades:

- **1. Investigación y revisión de literatura:** La primera fase, dedicada a la investigación y revisión de literatura, se planificó con una duración de 60 horas. Esta etapa contemplaba la revisión exhaustiva de papers académicos, libros especializados y documentación técnica relevante sobre técnicas de detección de phishing. El objetivo era establecer una base teórica sólida que fundamentara las decisiones técnicas posteriores y permitiera identificar las técnicas más prometedoras para su implementación

y evaluación.

- **2. Recopilación y preprocesamiento de datasets:** Esta fase se estimó en 30 horas de trabajo en la cuales se incluían la búsqueda, obtención y preparación de conjuntos de datos adecuados para el entrenamiento y evaluación de las técnicas de detección. Se contemplaba la necesidad de unificar datasets de diferentes fuentes, realizar tareas de limpieza de datos y preparar los conjuntos finales en formatos apropiados para su procesamiento posterior.
- **3. Implementación de técnicas individuales:** Constituía el núcleo del desarrollo técnico con 60 horas asignadas. Esta etapa abarcaba la implementación y ajuste de tres técnicas principales de detección de phishing, incluyendo tanto aproximaciones heurísticas como modelos de machine learning y deep learning. Se contemplaba la optimización de hiperparámetros y la adaptación de cada técnica a las características específicas de los datasets recopilados.
- **4. Evaluación y análisis de resultados:** Se planificó con 40 horas de duración. Esta etapa incluía la evaluación sistemática del rendimiento de cada técnica implementada utilizando métricas estándar como accuracy, precision, recall y F1-score. Se contemplaba la generación de análisis comparativos detallados y la interpretación de los resultados obtenidos para cada aproximación individual.
- **5. Desarrollo y evaluación de técnicas combinadas:** Esta etapa contemplaba el desarrollo de estrategias de combinación de las técnicas individuales, la implementación de enfoques ensemble y la evaluación de su rendimiento comparativo. Se planificaba explorar diferentes métodos de fusión de resultados y analizar el impacto de las combinaciones en la precisión global del sistema. Se estimaron 70 horas.
- **6. Redacción del TFM:** Esta última fase se estimó en 40 horas para la elaboración del documento final. Esta etapa incluía la estructuración del contenido, la redacción de todos los capítulos, la preparación de figuras y tablas, y las revisiones necesarias para garantizar la calidad y coherencia del documento.

2.2– Metodología de trabajo

La metodología CRISP-DM (CRoss-Industry Standard Process for Data Mining) [8] es particularmente adecuada para este proyecto porque ofrece un camino claro para trabajar con múltiples datos. Esta metodología se usa mucho en investigaciones y se adapta bien a la meta de comparar técnicas para detectar phishing y mejorar las métricas finales.

CRISP-DM consta de seis etapas: Comprensión del negocio, comprensión de los datos, preparación de datos, modelado, evaluación y despliegue.

Relacionándolo con este proyecto, se comienza por entender el negocio, lo que significa definir el problema del phishing y ver qué se puede hacer para combatirlo. La comprensión de los datos consta de la investigación de datos de correos, tanto maliciosos como legítimos que hay por Internet y entender qué campos se podrían utilizar para alcanzar el objetivo. La preparación de datos abarca la unificación de datasets heterogéneos y la definición de características relevantes. En la fase de modelado, se implementa la comparación de técnicas heurísticas, modelos clásicos y redes neuronales, siguiendo la recomendación de

CRISP-DM de probar múltiples algoritmos. En la evaluación, se analizan los resultados usando métricas que miden la efectividad de estos algoritmos y probando con datos que no se usan para entrenar, lo cual ayuda a asegurar que los experimentos sean confiables. Por último, en cuanto al despliegue, no se hace como tal en el proyecto, pero podría asemejarse a la publicación del TFM.

2.3– Estimación de costes

La estimación del presupuesto para el proyecto incluye lo siguiente:

- **Recursos humanos:** Se estima que se necesitarán unas 300 horas para trabajar en el TFM, y el salario promedio para un analista senior de ciberseguridad en España es de 25 euros por hora. Esto se calcula teniendo en cuenta que sus sueldos anuales van desde 45.000€ a 60.000€, lo cual se traduce en unos 25-34 euros por hora [9]. Así que, cogiendo el mínimo, el total sería de unos 7.500€.
- **Recursos tecnológicos:** Para el proyecto, especialmente para entrenar y evaluar los modelos de machine learning y deep learning, se usó un servidor propio valorado en unos 1.200 euros, preparado para tareas que requieren poder de cómputo con las siguientes especificaciones técnicas:
  - **CPU:** AMD Ryzen 7 5700G
  - **GPU:** NVIDIA RTX 3060 con 12GB
  - **RAM:** 16 GB DDR4
  - **Almacenamiento:** SSD NVMe de 2 TB para datasets y modelos
  - **Valor total del equipo:** 1.200€

  Para calcular la amortización del equipo, considerando el coeficiente lineal máximo del 26 % según las tablas oficiales de la Agencia Tributaria para equipos informáticos [10], y que el servidor estuvo en funcionamiento durante 600 horas (tiempo de procesamiento real, no horas dedicadas por el estudiante) equivalentes a 75 días (600h ÷ 8h/día), la amortización proporcional sería: (1.200€ × 0,26 × 75/365) = 64€.
- **Electricidad:** Respecto a la electricidad, si el servidor funcionó durante unas 600 horas, consumiendo aproximadamente 500W y con un precio de 0,14 euros el kWh en España [11], el gasto de energía sería de unos 42 euros.
- **Software y acceso a literatura:** No hay gastos adicionales porque el proyecto utiliza herramientas de código abierto como Python, scikit-learn y TensorFlow, y el acceso a la literatura científica se realizó a través de la biblioteca digital de la Universidad de Sevilla, que es gratuita para los estudiantes.

En total, sumando todos los gastos, **el coste del proyecto estimado sería de unos 7.606 euros**.

CAPÍTULO 3

Estado del arte

Este capítulo revisa cómo están las técnicas de detección de phishing hoy en día. Se habla de los métodos tradicionales que usan reglas hasta las más modernas que aplican inteligencia artificial y aprendizaje automático. Se analizan las diferentes técnicas, viendo qué hacen bien y en qué fallan. También se incluye una comparación entre las principales técnicas que se han estudiado, para ayudar a entender las decisiones técnicas tomadas en este trabajo.

3.1– Introducción

La detección automática de phishing ha sido un tema importante de estudio durante los últimos 20 años. Las formas de detectar phishing han cambiado a medida que los atacantes inventan nuevas estrategias para engañarnos.

El phishing puede manifestarse de varias maneras, y los cibercriminales usan distintos tipos de contenido dañino, cambiando sus métodos con frecuencia. Por eso, se han desarrollado diversas técnicas, cada una con sus pros y contras.

En las siguientes secciones se aborda cómo está la detección de phishing en estos días, agrupando las diferentes técnicas. Se enfoca en los métodos que han tenido mejores resultados y en ideas nuevas que mejoran enfoques anteriores. Esta revisión ayuda a identificar las técnicas más efectivas para emplear y comparar en este trabajo.

3.2– Tecnologías

Hoy en día, existen diferentes tecnologías para detectar intentos de phishing, cada una con características particulares que las hacen más o menos útiles dependiendo del contexto en el que se quieran aplicar.

3.2.1. Detección basada en reputación e indicadores de compromiso

Una de las formas más comunes para detectar correos electrónicos de phishing es usar información previa sobre amenazas conocidas. Esto incluye tanto las tradicionales **listas de reputación** (listas negras y blancas), como también fuentes y plataformas de **inteligencia de amenazas** (Threat Intelligence) que comparten y actualizan continuamente estos datos.

Las listas de reputación se componen de Indicadores de Compromiso (IOCs, *Indicators of Compromise*), que pueden incluir direcciones IP maliciosas, dominios utilizados en campañas previas de phishing, URLs sospechosas, hashes de archivos adjuntos, remitentes sospechosos... Estos elementos permiten identificar correos maliciosos por su coincidencia con amenazas conocidas.

Estas listas pueden gestionarse localmente por una organización, aunque su efectividad será limitada al conocimiento previamente adquirido. Lo más habitual es integrarlas con servicios externos como *VirusTotal*, *PhishTank*, *AbuseIPDB* o *Google Safe Browsing*, que ofrecen APIs para verificar en tiempo real la reputación. Estas plataformas agregan información de múltiples fuentes y, en algunos casos, realizan análisis dinámico ligero.

Por otro lado, los sistemas de *Threat Intelligence* permiten a las organizaciones acceder y compartir datos actualizados sobre amenazas emergentes, patrones de ataque y nuevos IOCs antes de que lleguen a difundirse ampliamente. Plataformas como *MISP* (*Malware Information Sharing Platform*) o servicios gestionados de inteligencia (como los ofrecidos por empresas de ciberseguridad) facilitan la integración de esta información en los sistemas de detección y respuesta.

Si bien estas técnicas ofrecen una detección rápida y de bajo coste computacional, tienen una limitación crítica: su efectividad depende de que la amenaza ya haya sido observada previamente. Como se señalan Sheng y col. [12], estas listas pueden detectar menos del 20 % de los ataques cuando se trata de campañas nuevas o previamente no identificadas. Además, según Fernando y col. [13], puede transcurrir un promedio de 16 horas desde la primera víctima de una campaña hasta que se incorporan los IOCs correspondientes a las listas negras, lo que deja a los primeros afectados completamente expuestos.

3.2.2. Protocolos de autenticación de correo electrónico

Son una estrategia complementaria para detectar y mitigar el phishing, al centrarse en verificar la legitimidad del remitente y la integridad del mensaje. Entre los más utilizados se encuentran:

- **SPF (Sender Policy Framework):** Permite al dominio del remitente especificar qué servidores están autorizados para enviar correos en su nombre. Si un servidor no autorizado intenta enviar mensajes, el receptor puede detectarlo como potencial suplantación.
- **DKIM (DomainKeys Identified Mail):** Añade una firma digital al contenido del mensaje, lo que permite verificar que el contenido no ha sido alterado y que proviene del dominio indicado.
- **DMARC (Domain-based Message Authentication, Reporting, and Conformance):** Se apoya en SPF y DKIM y permite al propietario del dominio especificar cómo deben tratarse los correos que no pasen las validaciones, además de habilitar informes para el seguimiento de posibles abusos. Este funcionamiento se muestra en la figura 3.1 [14].

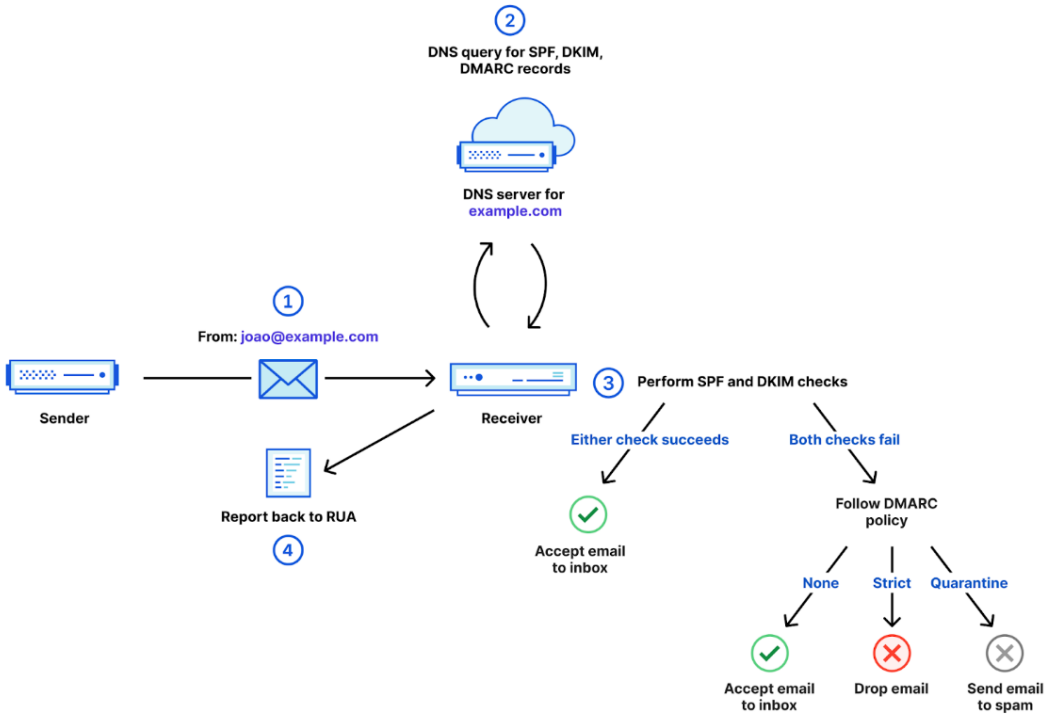

Figura 3.1: Esquema de protocolo DMARC

Estos mecanismos han demostrado una eficacia notable en entornos corporativos, especialmente para detectar intentos de suplantación directa del dominio del remitente. Según [15], países con mandatos obligatorios de DMARC han logrado reducir significativamente las tasas de phishing exitoso, como Estados Unidos, que las redujo del 69 % al 14 %.

No obstante, estas protecciones presentan limitaciones importantes frente a ataques modernos ya que hay estudios que confirman que el 89 % de los mensajes no deseados pasan las verificaciones SPF, DKIM y/o DMARC, lo que demuestra que los atacantes pueden encontrar formas de eludir la autenticación para engañar a los sistemas de correo [16].

3.2.3. Sandboxing

Esta técnica consiste en ejecutar archivos adjuntos o analizar URLs sospechosas en un entorno controlado y aislado, con el fin de observar su comportamiento en tiempo real. En concreto, el proceso de *sandboxing* funciona mediante la creación de un entorno virtual donde se carga el contenido sospechoso, se monitorea y registra su comportamiento, incluyendo sus interacciones con el sistema operativo y otros programas. Esto permite identificar amenazas que eluden análisis estáticos, como malware polimórfico (código malicioso que utiliza un motor embebido para cambiar continuamente su apariencia mediante técnicas de ofuscación y cifrado) o enlaces que redirigen a páginas maliciosas tras un retraso programado.

Aunque el *sandboxing* proporciona una inspección profunda y contextual del contenido malicioso, tiene varias limitaciones: requiere una infraestructura especializada, tanto en hardware como en software, tiene un alto coste computacional y puede introducir latencias significativas [17].

3.2.4. Heurísticas

La detección heurística se basa en la aplicación de reglas predefinidas o algoritmos simples que permiten identificar correos electrónicos sospechosos analizando patrones comunes en los ataques de phishing.

Estas reglas pueden aplicarse a distintos elementos del mensaje [18] [19] [20], como:

- **Metadatos y cabeceras:** Se analizan elementos técnicos del mensaje como:
  - Incongruencias entre el campo `From` y el dominio del servidor SMTP.
  - El dominio del campo `From` no coincide con el dominio de `Reply-To` o `Return-Path`.
  - Direcciones IP de envío que no pertenecen a rangos esperados o son de países inusuales.
  - Ausencia de autenticación SPF, DKIM o DMARC en los encabezados.
  - Presencia de Punycode en el remitente, que puede indicar un intento de suplantar un dominio. El Punycode es un sistema de codificación que permite representar caracteres Unicode (como letras con acentos o alfabetos no latinos) en nombres de dominio que solo admiten caracteres ASCII.
  - El remitente muestra un nombre completo que no coincide con el dominio real (`Banco Santander <avisos@sitiomalicioso.net>`).

- Correo enviado en fin de semana o fuera del horario laboral habitual.
- Muchos destinatarios simultáneos, lo que puede indicar una campaña masiva.

- **URLs contenidas en el mensaje:**
  - URLs visibles que no coinciden con el destino real del enlace.
  - Uso de acortadores de enlaces (`bit.ly`, `tinyurl`, etc.).
  - URLs que contienen direcciones IP en lugar de dominios (`http://192.168.1.1/login`).
  - URLs muy largas.
  - Uso excesivo de subdominios para emular legitimidad (`paypal.com.sitiomalicioso.com`).
  - Presencia de caracteres sospechosos o no ASCII en las URLs, como %, $, @ o unicode.
  - URLs con Punycode que simulan dominios legítimos con caracteres visualmente similares.

- **Contenido del cuerpo y asunto del mensaje:**
  - Uso de frases alarmistas como *"¡Último aviso!"*, *"Tu cuenta será suspendida"*, o el uso excesivo de mayúsculas.
  - Palabras clave sospechosas en el cuerpo como *"verifica"*, *"urgente"*, *"seguridad"*, *"bloqueo"*, *"clic aquí"*.
  - Términos financieros fuera de contexto como *"transferencia bloqueada"*, *"operación rechazada"* o *"saldo retenido"* en mensajes no bancarios.
  - Peticiones explícitas de credenciales o datos bancarios.
  - Archivos adjuntos con extensión peligrosa (`.exe`, `.scr`, `.zip`).
  - Inclusión de formularios HTML incrustados que solicitan información personal.
  - Falta de personalización (uso de *"Estimado cliente"* en vez del nombre real).
  - Detección de estafas conocidas (fraude nigeriano, premios falsos, sextorsión...).
  - Detección de errores ortográficos o gramaticales, incluyendo letras sustituidas con números (*c0ntraseña*).
  - Presencia de patrones de ofuscación en el texto o código HTML, como espacios intermedios, codificación en Base64 o inserciones de caracteres invisibles.

- **Estilo y estructura del mensaje:**
  - HTML mal estructurado o mezclas incoherentes de estilos.
  - Imágenes que simulan interfaces reales (botones de banca online, logos de servicios).
  - Texto con fuentes inconsistentes o mal alineado.
  - Uso excesivo de mayúsculas, colores llamativos o símbolos de alerta.
  - Inclusión de texto en imágenes (para evadir detección automática).

Estas reglas heurísticas se pueden implementar mediante **scripts personalizados** en lenguajes como Python, utilizando expresiones regulares para detectar patrones sospechosos en texto y cabeceras. Estas implementaciones pueden ser binarias (si el correo contiene

una característica concreta sospechosa, se marca como phishing) o basadas en un sistema de puntuación multicriterio, donde solo se clasifica como phishing si se supera un umbral definido tras sumar los puntos correspondientes a cada característica [21]. También es posible usar **motores de reglas** como `YARA` o `Suricata`, que permiten definir condiciones complejas que pueden activar alertas de forma binaria o probabilística, asignando un valor de confianza o puntuación basada en múltiples activaciones [22].

La principal ventaja de las heurísticas es su rapidez, simplicidad y transparencia, ya que permiten decisiones interpretables y no requieren entrenamiento previo. Sin embargo, también presentan limitaciones importantes: son sensibles a cambios en las técnicas de ataque, pueden generar falsos positivos si no se calibran bien y su efectividad disminuye ante ataques más sofisticados o personalizados. Las heurísticas por sí solas demuestran un rendimiento variable pero generalmente efectivo, con tasas de detección que oscilan entre el 89,6 % y el 98,7 % dependiendo de la complejidad de las reglas implementadas y del dataset utilizado [18] [19] [23] [24].

3.2.5. Machine Learning

Los métodos de aprendizaje automático (ML) representan una forma supervisada de identificar correos de phishing que se logra entrenando modelos con datos que ya han sido etiquetados previamente.

A diferencia de las heurísticas, estos algoritmos aprenden a reconocer patrones a partir de muchos datos, lo que les permite adaptarse mejor a nuevas formas de ataques si la variedad de datos permite generalizar a los modelos.

El proceso de detección mediante machine learning tradicional se estructura en varias etapas fundamentales:

- **1. Extracción y selección de características:** Esta etapa consiste en identificar y extraer del correo electrónico aquellos atributos o campos que resultan útiles para diferenciar entre mensajes legítimos y correos de phishing.

  Las características pueden ser tanto elementos básicos del correo (cuerpo y asunto), así como metadatos y cabeceras (remitente, destinatario, fecha, hora...). Además, muchas de las reglas heurísticas descritas anteriormente pueden traducirse en características cuantificables para los modelos de machine learning [25], como la presencia y número de enlaces, la cantidad y tipo de archivos adjuntos o indicadores de ofuscación en el texto.

- **2. Preprocesamiento de datos:** Esta etapa esencial transforma los datos brutos en un formato adecuado para el entrenamiento de modelos de machine learning. El proceso incluye múltiples técnicas especializadas que abordan diferentes aspectos de la calidad de los datos [26] [27]:

  - **Limpieza de datos:** Constituye el primer paso, eliminando registros corruptos, duplicados o inconsistentes.

- **Preprocesamiento de texto:** Se aplican técnicas de normalización como conversión a minúsculas, eliminación de signos de puntuación y caracteres especiales, tokenización para dividir el texto en unidades manejables, y eliminación de palabras vacías (*stop words*) que no aportan valor semántico.
- **Reducción morfológica:** La lematización y *stemming* reducen las palabras a sus formas raíz, permitiendo agrupar variaciones con significados similares y reducir la dimensionalidad del vocabulario.
- **Normalización de características:** El escalado y normalización de características aseguran que todas las variables numéricas tengan rangos comparables.
- **Codificación de variables:** La codificación de variables categóricas convierte datos no numéricos en formatos procesables.

- **3. Entrenamiento del modelo:** Cada algoritmo implementa diferentes estrategias de aprendizaje. Aquí están algunos de los modelos de Machine Learning más usados en detección de phishing:

  - **k-Nearest Neighbors (k-NN):** Clasifica correos comparando similitudes con ejemplos previamente etiquetados. Busca los "k" vecinos más cercanos en el espacio de características y asigna la categoría predominante. Es simple pero computacionalmente costoso con grandes datasets.
  - **Árboles de decisión:** Construyen reglas de clasificación jerárquicas mediante nodos que dividen datos según características clave. Generan decisiones interpretables pero son propensos a sobreajuste.
  - **Random Forest:** Combina múltiples árboles de decisión con votación mayoritaria para mejorar precisión y reducir sobreajuste. Destaca por manejar ruido en datos y alta dimensionalidad.
  - **Máquinas de Vectores Soporte (SVM):** Utiliza hiperplanos de separación óptimos en espacios de alta dimensionalidad, siendo particularmente efectivo con características textuales.
  - **Naive Bayes:** Basado en el teorema de Bayes, calcula probabilidades condicionales de phishing dadas las características del correo. Asume independencia entre características.
  - **Regresión Logística:** Modela probabilidades de phishing mediante una función sigmoide. Ideal cuando la relación entre características y resultado es lineal. Es eficiente computacionalmente.

La tabla comparativa 3.1 sintetiza los resultados de *accuracy* obtenidos por los principales modelos de aprendizaje automático evaluados en tres investigaciones seleccionadas [28] [29] [30].

Tabla 3.1: Comparativa de *accuracy* en modelos de ML

Modelo	**[28]**	**[29]**	**[30]**
k-Nearest Neighbors (k-NN)	—	89,00 %	98,32 %
Árboles de decisión	—	96,00 %	99,10 %
Random Forest	97,60 %	95,00 %	99,45 %
Máquinas de Vectores Soporte (SVM)	90,10 %	95,00 %	98,13 %
Naive Bayes	93,81 %	98,00 %	96,67 %
Regresión Logística	—	90,00 %	—

- **4. Optimización:** Para maximizar la efectividad de los modelos, se utilizan las siguientes técnicas:
  - **Validación cruzada:** Técnica de remuestreo que divide los datos en k particiones. Entrena el modelo con *k-1* grupos y valida con el restante, rotando hasta cubrir todos. Proporciona una evaluación robusta de la efectividad en datos no vistos, evitando sobreajuste.
  - **Búsqueda en rejilla (*Grid Search*):** Exploración sistemática de combinaciones de hiperparámetros (por ejemplo, profundidad de árboles o kernels de SVM). Evalúa cada combinación mediante validación cruzada y selecciona la configuración óptima basándose en la métrica que se elija.
  - **Manejo de desbalanceo:** Existen técnicas para equilibrar clases, lo que evita sesgos hacia clases dominantes, mejorando la detección de ataques:
    - *Oversampling* (SMOTE): Genera muestras sintéticas de la clase minoritaria.
    - *Undersampling*: Reduce muestras de la clase mayoritaria.

- **5. Evaluación:** Los modelos se evalúan mediante métricas que cuantifican diferentes aspectos de la efectividad en clasificación binaria. Las cuatro métricas esenciales son:

  1. ***Accuracy*:** Mide la proporción global de predicciones correctas:

  $$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.1}$$

     - VP = Verdaderos positivos (phishing detectado correctamente)
     - VN = Verdaderos negativos (legítimos identificados correctamente)
     - FP = Falsos positivos (legítimos marcados como phishing)
     - FN = Falsos negativos (phishing no detectado)

     Refleja la capacidad general del modelo para clasificar correos correctamente, pero puede ser engañosa en datasets desbalanceados.

  2. ***Precision*:** Cuantifica la fiabilidad de las detecciones positivas:

  $$\text{Precision} = \frac{VP}{VP + FP} \tag{3.2}$$

     Evalúa cuántos de los correos marcados como phishing son realmente maliciosos. Valores altos indican pocos falsos positivos.

  3. ***Recall*:** Mide la capacidad para detectar ataques reales:

  $$\text{Recall} = \frac{VP}{VP + FN} \tag{3.3}$$

     Cuantifica la proporción de ataques de phishing que el modelo logra identificar. Valores bajos implican que no se están detectando los ataques.

  4. ***F1-Score*:** Combina *precision* y *recall* en una métrica balanceada:

  $$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.4}$$

     Es especialmente útil en datasets desbalanceados, donde valores cercanos a 1 indican equilibrio óptimo entre detección y falsas alarmas.

El uso de machine learning para la detección de correos de phishing ofrece ventajas significativas, como la capacidad de identificar patrones complejos y evolutivos en los mensajes, adaptarse a nuevas variantes de ataques gracias al aprendizaje continuo y automatizar la clasificación a gran escala, lo que permite una respuesta rápida y eficiente ante amenazas cambiantes. Sin embargo, estos sistemas requieren grandes volúmenes de datos de calidad y representativos para su entrenamiento, son sensibles a sesgos presentes en los datos y pueden resultar difíciles de interpretar, además de exigir costes elevados en recolección, limpieza y protección de la información.

3.2.6. Deep Learning

Los métodos de aprendizaje profundo (*Deep Learning*) representan una evolución del aprendizaje automático tradicional, caracterizados por el uso de redes neuronales artificiales con múltiples capas intermedias. Estas arquitecturas permiten el aprendizaje automático de características directamente desde los datos crudos, eliminando en gran medida la necesidad de una extracción manual de características que sí es requerida en los enfoques de machine learning tradicional descritos en la sección anterior.

A diferencia de los algoritmos clásicos, los modelos de deep learning procesan información mediante transformaciones no lineales sucesivas que capturan representaciones jerárquicas de los datos. Esta capacidad puede resultar muy interesante para el análisis de correos electrónicos, donde se pueden modelar eficientemente relaciones complejas en secuencias textuales, estructuras semánticas y patrones no evidentes en metadatos.

El flujo de trabajo mantiene etapas análogas al machine learning tradicional, pero con diferencias clave en su implementación:

- **1. Preprocesamiento de datos:** A diferencia de los enfoques clásicos de Machine Learning, múltiples estudios recomiendan evitar el preprocesamiento exhaustivo para modelos Deep Learning, ya que estos pueden capturar patrones contextuales directamente del texto crudo [31]. No obstante, se aplican técnicas básicas de adaptación:
    - ***Embeddings* distribuidos:** En términos sencillos, un *embedding* es una forma de convertir cada palabra o fragmento de texto en un conjunto de números (un vector) que refleja su significado y relación con otras palabras. Así, el modelo puede captar que diferentes palabras o frases pueden expresar intenciones similares, incluso si no son idénticas, lo que resulta especialmente útil para identificar patrones de phishing que varían en la forma pero no en el fondo. Por ejemplo, *"bancos"* y *"finanzas"* se podrían relacionar con este preprocesamiento.
    - ***Tokenización* subpalabra:** En la práctica, los atacantes suelen emplear variantes mal escritas o creativas de palabras clave para evadir los filtros tradicionales, como escribir *"Payp@l"* en lugar de "PayPal". La tokenización subpalabra descompone las palabras en unidades más pequeñas y reconocibles (por ejemplo, "Pay", "p", "@", "l"), permitiendo que el modelo identifique fragmentos familiares aunque la palabra completa no haya sido vista antes. Este

enfoque es especialmente útil para manejar vocabularios ilimitados y adaptarse a términos técnicos, errores ortográficos o intentos deliberados de manipulación, tan frecuentes en el phishing.

- ***Padding* dinámico:** Es una técnica que resuelve el problema de que los correos electrónicos pueden tener longitudes muy variables. Las redes neuronales requieren que todos los ejemplos de entrada tengan el mismo tamaño, por lo que se añaden tokens de relleno al final de los mensajes más cortos, igualando su longitud a la de los más largos (o a una longitud máxima predefinida).

- **2. Entrenamiento:** Cada modelo implementa diferentes estrategias de aprendizaje. Destacan cuatro enfoques principales en detección de phishing:

  - **Redes Neuronales Convolucionales (CNN):** Utilizan filtros que se desplazan sobre los datos para detectar patrones locales. En el contexto de correos, estos filtros identifican combinaciones de palabras o frases (*n-grams*) sospechosas, independientemente de su posición en el texto. Esto es útil para reconocer patrones de phishing que pueden estar dispersos en el cuerpo del mensaje.
  - **Transformers y BERT:** Los *Transformers* emplean un mecanismo de autoatención que analiza todas las palabras del mensaje simultáneamente, asignando un peso a cada una según su relevancia para el contexto global. Esto permite capturar relaciones complejas entre palabras distantes. Un modelo destacado es BERT (*Bidirectional Encoder Representations from Transformers*), que procesa el texto en ambas direcciones (izquierda-derecha y derecha-izquierda) para comprender el contexto completo de cada palabra, mejorando la detección de sutilezas engañosas. Existen variantes optimizadas de este modelo, como RoBERTa (más robusto y entrenado con más datos) o DistilBERT (una versión más ligera y rápida, pero con rendimiento comparable).
  - **Redes Neuronales Recurrentes (RNN) y Memoria a Largo Plazo (LSTM y BiLSTM):** Las RNN procesan secuencias de datos (como el texto de un correo) manteniendo un estado interno que actúa como memoria. Sin embargo, las RNN estándar tienen dificultades para recordar información a largo plazo. La variante LSTM soluciona esto mediante una estructura de celdas de memoria que pueden retener información relevante durante más tiempo. Esto permite al modelo relacionar palabras separadas por varias frases, capturando patrones temporales que pueden indicar intenciones fraudulentas. Además, una versión bidireccional de LSTM (BiLSTM) procesa la secuencia tanto hacia adelante como hacia atrás, lo que mejora la comprensión del contexto al tener en cuenta tanto las palabras anteriores como las siguientes.
  - **Arquitecturas híbridas:** Combinan dos o más modelos de deep learning. Por ejemplo, una CNN puede emplearse para extraer patrones locales del texto, que luego se pasan a una LSTM o a un Transformer para analizar dependencias contextuales más amplias. Estas combinaciones permiten capturar tanto características superficiales como relaciones semánticas complejas aunque no tienen por qué tener mejor rendimiento que los modelos individuales.

La tabla comparativa 3.2 sintetiza los resultados de *accuracy* obtenidos de varias investigaciones que evalúan la efectividad de los modelos de Deep Learning [31] [32] [33] [34].

Tabla 3.2: Comparativa de *accuracy* en modelos de Deep Learning

Modelo	**[31]**	**[32]**	**[33]**	**[34]**
CNN y variaciones				
CNN	98,07 %	97,00 %	—	—
Transformers				
BERT	98,65 %	—	—	99,11 %
DistilBERT	98,74 %	—	—	98,99 %
RoBERTa	99,03 %	—	—	99,43 %
RNN y variaciones				
RNN	95,14 %	—	96,00 %	—
LSTM	97,43 %	—	96,00 %	—
Arquitecturas híbridas				
CNN+RNN	—	96,00 %	—	—
BERT+CNN	—	—	97,50 %	—

- **3. Optimización:** Para maximizar la efectividad de los modelos de deep learning en detección de phishing, se emplean técnicas avanzadas de ajuste y optimización que difieren de los métodos tradicionales de machine learning:
  - **Fine-tuning (Ajuste fino):** Técnica fundamental en deep learning que adapta modelos preentrenados en grandes conjuntos de datos a la tarea específica de detección de phishing. Se puede aplicar el enfoque de ajuste completo donde se reentrenan todos los parámetros del modelo con nuevos datos (correos), lo que ofrece máximo rendimiento pero requiere recursos computacionales elevados. Por otra parte, está la opción de realizar un ajuste parcial, donde se actualizan solo capas específicas (generalmente las finales) o se añaden capas adicionales entrenables, manteniendo congeladas las capas iniciales que capturan características generales. Este último enfoque reduce costes computacionales y previene sobreajuste en conjuntos pequeños.
  - **Búsqueda de mejores hiperparámetros:** El objetivo es el mismo que lo explicado en la sección de machine learning, pero adaptándolo a los hiperparámetros típicos del deep learning (tasa de aprendizaje, tamaño de capas, funciones de activación). Grid Search se podría aplicar ya que garantizaría encontrar la mejor combinación, sin embargo, resulta computacionalmente prohibitivo para redes profundas con muchos hiperparámetros. Por ello, se utilizan métodos como Random Search (muestreo aleatorio) u optimización bayesiana, que permiten explorar espacios amplios con menor costo, siendo más viables en escenarios prácticos.
  - **Regularización avanzada:** Técnicas como Dropout (desactivación aleatoria de neuronas durante el entrenamiento) y Early Stopping (interrupción anticipada al detectar sobreajuste) mejoran la generalización de modelos complejos. Para problemas de desbalanceo de clases, se podrían emplear funciones de pérdida ponderadas que penalizan más los errores en la clase minoritaria.
- **4. Evaluación:** Utiliza las mismas métricas fundamentales que el machine learning (*Accuracy, Precision, Recall, F1-Score*).

La aplicación de deep learning en detección de phishing ofrece la capacidad única de descubrir patrones complejos y evolutivos sin depender de reglas predefinidas, adaptándo-

se dinámicamente a nuevas tácticas de ataque gracias a su aprendizaje jerárquico. Estos modelos requieren conjuntos de datos extensos y balanceados para un entrenamiento efectivo, presentan mayor demanda computacional durante su operación, y su naturaleza de caja negra dificulta la interpretación de las decisiones, lo que puede complicar la revisión de falsos positivos. Además, son sensibles a manipulaciones adversarias en el contenido del correo que pueden engañar al modelo.

3.3– Análisis de soluciones

En esta sección se presenta un análisis comparativo de las técnicas de detección de phishing existentes. El análisis se estructura en dos componentes principales: una comparativa técnica y un análisis cualitativo de ventajas e inconvenientes.

3.3.1. Comparativa técnica de las soluciones

La tabla 3.3 presenta una evaluación sistemática de las técnicas de detección de phishing según cinco criterios fundamentales que determinan su viabilidad práctica en entornos reales.

Tabla 3.3: Comparativa de las técnicas de detección de phishing

Técnica	**Velocidad detección**	**Coste computacional**	**Efectividad**	**Recursos necesarios**	**Adaptabilidad**
Reputación e IOCs	Muy alto	Bajo	Bajo	Dependencia de servicios externos y mantenimiento de bases de datos.	Muy bajo
Protocolos de autenticación	Muy alto	Bajo	Medio	Configuración DNS, certificados digitales, infraestructura PKI	Bajo
Sandboxing	Bajo	Muy alto	Alto	Infraestructura especializada y aislada	Alto
Heurísticas	Alto	Bajo	Alto	Desarrollo de reglas, calibración manual	Medio
Machine Learning	Alto	Medio	Alto	Datasets etiquetados grandes, infraestructura de entrenamiento	Alto
Deep Learning	Medio-Alto	Alto	Muy alto	Datasets etiquetados grandes, GPUs/TPUs	Muy alto

La evaluación revela patrones claros en el comportamiento de las diferentes técnicas. Las soluciones basadas en conocimiento previo, como las listas de reputación e IOCs y los protocolos de autenticación (SPF, DKIM, DMARC), destacan por su rapidez y bajo coste computacional, pero presentan limitaciones importantes en adaptabilidad y en la detección de ataques novedosos. En contraste, las técnicas de deep learning ofrecen la mayor efectividad y capacidad de adaptación, aunque requieren grandes volúmenes de datos etiquetados y alta potencia computacional, lo que incrementa su coste y complejidad. Una alternativa más equilibrada son los modelos tradicionales de machine learning, que mantienen altos niveles de efectividad con menores requisitos computacionales, aunque siguen dependiendo de conjuntos de datos representativos. Por otro lado, las técnicas heurísticas permiten una detección eficiente mediante reglas definidas manualmente, con buena efectividad, aunque

su capacidad de adaptación depende del mantenimiento constante de dichas reglas. Finalmente, el sandboxing ofrece una alta tasa de detección al analizar el comportamiento de los elementos en entornos aislados, pero su velocidad es baja y su implementación requiere una infraestructura compleja y costosa.

3.3.2. Análisis crítico de ventajas e inconvenientes

A continuación, se proporciona un análisis detallado recopilando las fortalezas y debilidades de cada técnica, facilitando así la toma de decisiones según los requisitos específicos del contexto de implementación.

Listas de reputación e IOCs

Ventajas:

- Detección instantánea de amenazas conocidas.
- Muy bajo coste computacional.
- Fácil integración y mantenimiento.
- No requiere entrenamiento previo.

Inconvenientes:

- Ineficaz ante ataques nuevos o desconocidos (menos del 20 % de efectividad en campañas inéditas).
- Dependencia de la actualización de bases de datos externas.
- Puede haber un retraso de hasta 16 horas en la incorporación de nuevas amenazas.

Protocolos de autenticación

Ventajas:

- Verificación criptográfica robusta del remitente y la integridad del mensaje.
- Reducción significativa de suplantaciones directas.
- Estandarización y soporte en grandes plataformas.

Inconvenientes:

- Limitado frente a ataques sofisticados (el 89 % pasa los filtros).
- Requiere configuración y mantenimiento técnico.
- Su efectividad depende de la adopción global.

Sandboxing

Ventajas:

- Permite analizar el comportamiento real de archivos y enlaces.
- Detecta amenazas que eluden análisis estáticos.
- Útil para análisis forense y campañas avanzadas.

Inconvenientes:

- Muy alto coste computacional y de infraestructura.
- Introduce latencia significativa en la detección.
- Puede ser evadido si el malware detecta el entorno de sandbox.

Heurísticas

Ventajas:

- Respuesta inmediata y bajo coste.
- Transparencia e interpretabilidad de las reglas.
- No requiere grandes volúmenes de datos.

Inconvenientes:

- Sensibles a cambios en las técnicas de ataque.
- Generan falsos positivos si no se calibran bien.
- Requieren ajuste y mantenimiento continuo.

Machine Learning

Ventajas:

- Detecta patrones complejos y evolutivos.
- Adaptable a nuevas variantes de phishing.
- Automatización y escalabilidad.

Inconvenientes:

- Requiere datasets grandes y etiquetados.
- Sensible a sesgos y calidad de los datos.
- Difícil de interpretar y revisar.

Deep Learning

Ventajas:

- Máxima efectividad y capacidad de adaptación.
- Aprende características automáticamente.
- Detecta patrones no evidentes en texto y metadatos.

Inconvenientes:

- Altos requisitos computacionales y de datos.
- Difícil de interpretar y auditar.
- Vulnerable a ataques de evasión adversaria.

CAPÍTULO 4

Desarrollo de la solución

Este capítulo presenta el desarrollo de la implementación de varias técnicas para la detección de phishing. Se han seleccionado metodologías representativas de diferentes enfoques (heurístico, machine learning y deep learning) para validar empíricamente su efectividad en datasets masivos y diversos, con el objetivo de verificar la escalabilidad y capacidad de generalización reportadas en la literatura.

4.1– Introducción

El desarrollo experimental se centra en implementar y contrastar técnicas clave identificadas en el análisis comparativo previo. Se han seleccionado tres tecnologías que representan distintos equilibrios entre efectividad, coste computacional y adaptabilidad:

- Un sistema heurístico basado en reglas.
- Modelos clásicos de Machine Learning (SVM, Random Forest, Naive Bayes y Regresión Logística)
- Arquitecturas avanzadas de Deep Learning (BERT, LSTM y BiLSTM).

Cada técnica se implementará con preprocesamientos específicos optimizados para sus características, utilizando datasets masivos y diversos para evaluar su capacidad de generalización en escenarios reales.

4.2– Justificación

La selección de técnicas responde a los hallazgos clave del análisis comparativo realizado en el estado del arte [3] y su idoneidad para los objetivos experimentales:

- **Heurísticas:** Se incluyen por su bajo coste computacional y rapidez de detección, características esenciales para entornos con restricciones de recursos. Su implementación permitirá establecer una línea base de rendimiento y validar su efectividad reportada en datasets reales. La selección específica de reglas heurísticas se basó en la literatura existente, priorizando aquellas que han demostrado mayor efectividad en estudios previos.
- **Machine Learning:** Estos modelos ofrecen un equilibrio óptimo entre efectividad alta, coste computacional medio y adaptabilidad alta. La selección cubre:
  - **Random Forest:** Fue seleccionado por su buen rendimiento en múltiples estudios, alcanzando precisiones (*accuracy*) de hasta 99,45 %, combinado con su robustez ante ruido en los datos y capacidad de manejo de características de alta dimensionalidad.
  - **SVM:** Se incluyó por ser eficaz en espacios de alta dimensionalidad, como lo es el texto (asunto y cuerpo de correos).
  - **Naive Bayes:** Se incorporó por su rapidez, bajo consumo de recursos y rendimiento competitivo (hasta un 98 % de efectividad).
  - **Regresión Logística:** Se añadió como método de referencia debido a su facilidad de interpretación y proporciona un punto de referencia para la comparación con métodos más complejos.
- **Deep Learning:** Seleccionados por su efectividad y adaptabilidad. Su inclusión responde a:
  - **LSTM:** Fue seleccionado por su capacidad probada para capturar dependencias temporales en secuencias textuales. Su arquitectura especializada en el manejo de información secuencial lo hace particularmente adecuado para analizar la estructura narrativa de correos electrónicos de phishing.
  - **BiLSTM:** Se incluyó para evaluar si el procesamiento bidireccional mejora la detección al permitir que el modelo considere tanto el contexto anterior como el posterior de cada elemento en la secuencia. Esta capacidad es especialmente relevante para identificar patrones de engaño que pueden manifestarse de forma no lineal en el texto.
  - **BERT:** Fue incluido por su capacidad para generar representaciones contextuales bidireccionales y su entrenamiento previo en conjuntos de datos masivos (millones de palabras), además de por su buen rendimiento en los estudios analizados (3.2).

La exclusión de otras técnicas se fundamenta en:

- **Técnicas de listas de reputación e IOCs:** A pesar de su amplia adopción en entornos productivos, fueron excluidas del desarrollo experimental debido a su efec-

tividad inferior al 20 % y su latencia de actualización de hasta 16 horas frente a campañas de phishing previamente no identificadas. Además, esto las hace inadecuadas para un estudio centrado en la capacidad de generalización de varios datasets. Su dependencia crítica de servicios externos con límites de consultas gratuitas introducen variables incontrolables que comprometerían la reproducibilidad del experimento.

- **Protocolos de autenticación:** De forma similar, los protocolos de autenticación (SPF, DKIM, DMARC) fueron descartados debido a su limitado alcance funcional, ya que el 89 % del spam consigue eludir estas verificaciones según la literatura analizada. Su efectividad se circunscribe principalmente a la detección de suplantación directa de dominios, lo que no abarca el espectro completo de técnicas de phishing modernas que emplean dominios alternativos. Además, su evaluación rigurosa en entornos científicos resulta compleja, al depender de factores externos como la configuración real de los servidores emisores y receptores, así como del cumplimiento adecuado de estándares por parte de los dominios implicados.
- **Sandboxing:** Aunque presenta alta efectividad de detección, fue excluido por su alto coste computacional y la latencia significativa que introduce en el procesamiento. Estas características lo hacen inviable para evaluaciones sistemáticas sobre conjuntos de datos masivos, además de requerir una infraestructura especializada que excede el alcance de este trabajo experimental.

La elección de estas técnicas responde a su carácter completamente autónomo, ya que todas permiten abordar el caso de uso en el que la solución puede implementarse y ejecutarse localmente en un servidor de correo, sin necesidad de realizar peticiones externas a Internet ni depender de servicios de terceros, garantizando así tanto la privacidad de los datos como la ausencia de demoras contundentes en la llegada de correos legítimos.

Esta selección permite comparar tres técnicas distintas. Los resultados obtenidos con diferentes conjuntos de datos permiten verificar si los modelos, junto con los preprocesamientos específicos aplicados en este trabajo, mantienen o incluso mejoran el rendimiento observado en estudios previos mostrados en el estado del arte [3].

4.3– Diseño y aspectos relevantes de la implementación

Esta sección explica cómo se implementan las técnicas elegidas para detectar phishing. El trabajo se organiza en cuatro etapas principales que van desde preparar los datos hasta implementar estos métodos como se muestra en la figura 4.1.

El proceso arranca seleccionando y preparando varios conjuntos de datos. Después viene una etapa de limpieza y normalización para asegurar que los datos tengan buena calidad y sean consistentes. Posteriormente, se agrupan los datasets según sus campos. Por último, se implementan las tres familias de técnicas que se eligieron: heurísticas basadas en reglas, algoritmos de machine learning tradicional y modelos de deep learning. Cada familia de técnicas necesita sus propias estrategias de preprocesamiento optimizadas para obtener el mejor rendimiento posible, además de métodos de entrenamiento y validación que se adapten a sus características específicas.

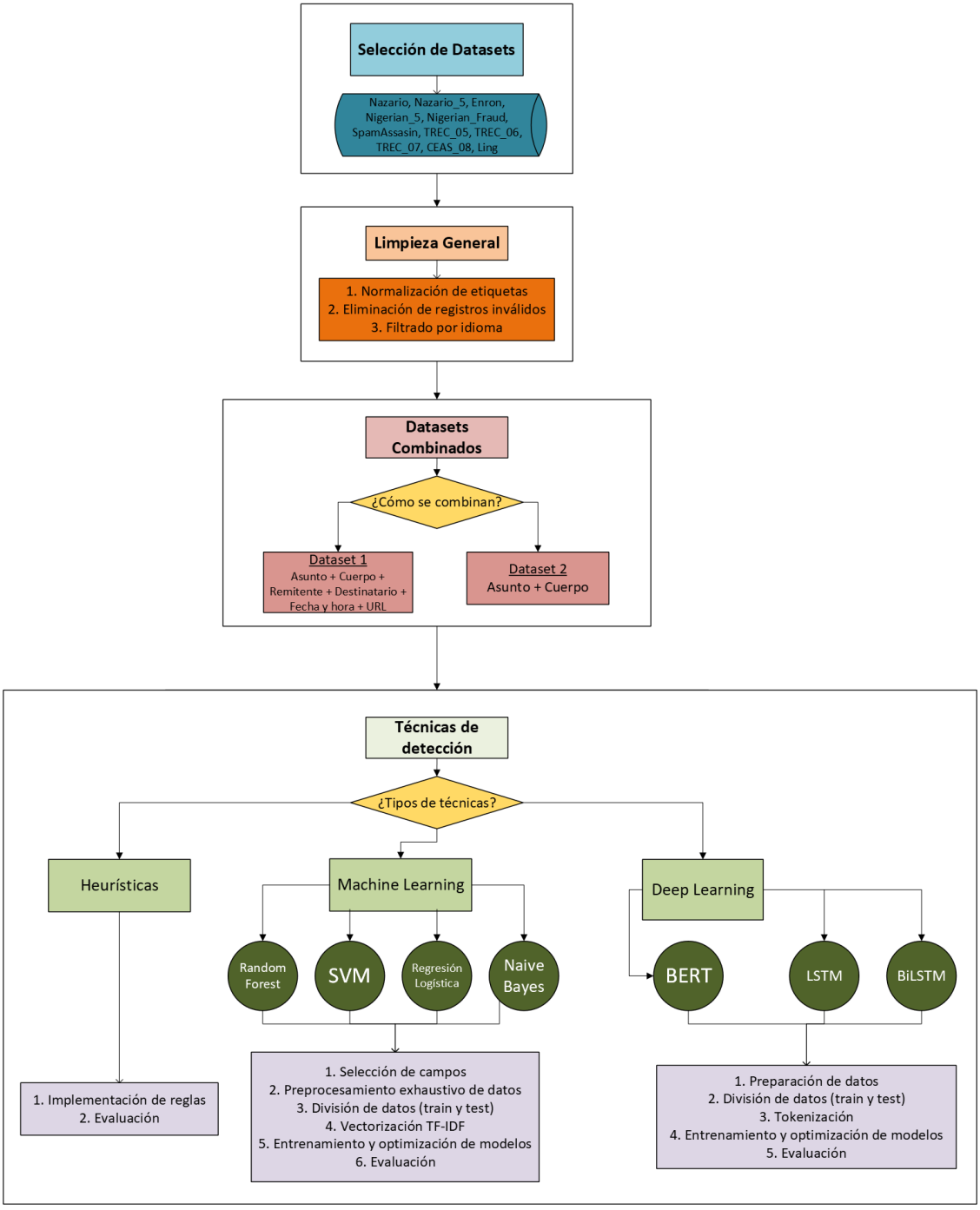

Figura 4.1: Proceso de implementación de las técnicas de detección de phishing

4.3.1. Selección de datasets

Los conjuntos de datos utilizados en este trabajo han sido obtenidos del repositorio público de Zenodo, específicamente de la colección "Phishing Email Curated Datasets" desarrollada por Champa y col. [35]. Esta colección representa una compilación de 11 datasets que abarcan un período temporal desde 1998 hasta 2022, proporcionando una perspectiva histórica y evolutiva de las técnicas de phishing. Estos 11 datasets juntos contienen un total de 218.086 registros de correos con una distribución bastante balanceada de un 47,88 % de phishings y un 52,11 % de legítimos.

La colección incluye los siguientes conjuntos de datos con sus características específicas, según se detalla en los trabajos de referencia [35] [36]:

- **CEAS_08**: Procedente del *CEAS 2008 Challenge Lab Evaluation Corpus*, que contiene 39.154 correos electrónicos. Este dataset incluye 17.312 correos legítimos y 21.842 correos de phishing, con una ratio legítimo:phishing de 44:56. Los correos fueron recopilados en 2008 y representan una mezcla diversa de comunicaciones corporativas y phishing de la época.
- **Enron**: Conjunto de datos derivado del famoso corpus de correos electrónicos de Enron, que incluye 29.767 emails (15.791 legítimos y 13.976 phishing) con un ratio de 53:47. Este dataset procede del año 2006 y proporciona comunicaciones corporativas auténticas que sirven como base sólida para correos legítimos, ofreciendo diversidad en estilos de comunicación empresarial.
- **Ling**: Dataset que contiene 2.859 correos electrónicos recopilados del año 2000 de la *Linguist List online email communication*. Presenta una distribución altamente desbalanceada con 2.401 correos legítimos y 458 phishing (ratio 84:16), caracterizado por un lenguaje más formal y estructurado típico de comunicaciones académicas.
- **Nazario**: Colección pública de correos de phishing recopilados por José Nazario entre 2015-2022, que incluye 1.565 correos de phishing auténticos. Este dataset se centra exclusivamente en phishing y representa diversas técnicas de ingeniería social contemporáneas.
- **Nazario_5**: Versión expandida que combina el dataset *Nazario* con 1.500 correos legítimos seleccionados aleatoriamente de otros datasets, creando un conjunto balanceado de 3.065 correos con una ratio de 49:51. Esta extensión permite su uso en tareas de clasificación binaria.
- **Nigerian_Fraud**: Colección específica de 3.332 correos de fraude nigeriano recopilados entre 1998 y 2007 de la *CLAIR collection of fraud email*. Contiene exclusivamente correos fraudulentos de este tipo específico de ataque.
- **Nigerian_5**: Versión expandida que combina el dataset *Nigerian_Fraud* con casi 3.000 correos legítimos seleccionados aleatoriamente de otros datasets, creando un conjunto balanceado de 6.331 correos con una ratio de 47:53. Esta extensión permite su uso en tareas de clasificación binaria.
- **SpamAssasin**: Corpus ampliamente reconocido del *Apache Software Foundation* (2002-2006) que contiene 5.809 correos electrónicos con 4.091 legítimos y 1.718

phishing (ratio 70:30). Es utilizado como referencia en múltiples estudios de la literatura.

- **TREC_05, TREC_06, TREC_07**: Conjuntos de datos del *TREC Public Corpus*, creados mediante la aplicación de filtros automáticos de spam seguidos de evaluación humana. TREC-05 contiene 55.990 correos (ratio 58:42), TREC-06 incluye 16.457 correos (ratio 76:24), y TREC-07 comprende 53.757 correos (ratio 45:55). Estos datasets representan diferentes años y evoluciones en las técnicas de phishing.

Los datasets fueron sometidos a un proceso de curación básico llevado a cabo por Champa y col. [35] antes de ponerlos de manera pública en el repositorio que incluyó:

- **Decodificación**: Los correos codificados/encriptados fueron decodificados utilizando técnicas específicas para cada tipo de codificación, manejando diferentes conjuntos de caracteres según las cabeceras Content-Type.
- **Extracción de texto plano:** Los correos en formato HTML fueron convertidos a texto plano, eliminando etiquetas de formato pero preservando el contenido esencial.

Los datasets se organizan en dos categorías estructurales: aquellos que contienen únicamente asunto y cuerpo (Ling y Enron), y aquellos que incluyen seis campos completos: remitente, destinatario, fecha y hora, asunto, cuerpo y presencia de URLs (que son todos los demás). Como se puede apreciar, estos datasets no contienen más cabeceras ni adjuntos que podrían dar más información a las técnicas de detección sobre si el correo realmente es phishing o legítimo, y además están en su mayoría en inglés. En relación a esto, cabe destacar que no se han encontrado datasets lo suficientemente reseñables en Internet que contengan todo el email completo sin preprocesar (con todas las cabeceras y adjuntos), ni tampoco datasets en español.

La elección de esta colección de datasets se fundamenta en varias ventajas críticas que la distinguen significativamente de las aproximaciones utilizadas en estudios previos de la literatura:

- **1. Diversidad temporal y representatividad**: Esta selección abarca casi tres décadas de evolución del phishing. Esta perspectiva temporal permite evaluar la capacidad de los modelos para detectar tanto técnicas clásicas como emergentes, proporcionando una evaluación más robusta de la generalización.
- **2. Volumen y diversidad de fuentes**: Con un total de 449,3 MB de datos distribuidos en 11 datasets distintos, la colección supera significativamente en volumen y diversidad a estudios como *Advancing Phishing Email Detection* [37] que utiliza únicamente 6.428 correos de entrenamiento o *Evaluation of Deep Learning Algorithms in Comparison for Phishing Email Identification* [32] que utiliza 18.650 emails en total. Esta limitación de escala compromete la capacidad de los modelos para aprender patrones complejos y generalizar efectivamente a nuevos tipos de ataques.
- **3. Especialización en tipos de phishing**: La inclusión de datasets especializados como Nigerian_Fraud y Nazario, combinados con corpus corporativos como Enron y

académicos como TREC, proporciona una cobertura exhaustiva de diferentes vectores de ataque y contextos comunicativos.

- **4. Desafío de generalización como fortaleza**: La heterogeneidad de la colección presenta un desafío significativo que, paradójicamente, constituye su mayor fortaleza. Esta exigencia metodológica, aunque puede resultar en métricas de efectividad inicialmente inferiores, proporciona una evaluación más realista de la capacidad de generalización en entornos operacionales reales.

La selección de esta colección de datasets representa, por tanto, una aproximación metodológicamente rigurosa que prioriza la validez externa y la capacidad de generalización sobre la optimización de métricas en contextos controlados, estableciendo un estándar más exigente pero realista para la evaluación de técnicas de detección de phishing.

4.3.2. Limpieza General

Una vez seleccionados los conjuntos de datos, se implementa un proceso integral de limpieza y normalización. Este se aplica de manera individualizada a cada uno de los 11 datasets. El proceso se organiza en varias etapas consecutivas, todas diseñadas para asegurar la calidad de los datos antes de que las diferentes técnicas de detección los utilicen:

- **1. Normalización de etiquetas:** La primera etapa se enfoca en estandarizar las etiquetas de clasificación. Los datasets originales mostraban inconsistencias notables en el formato: algunos contenían valores como 0.0, 1.0, mientras otros usaban '0', '1', e incluso valores numéricos no binarios. Por ello, se desarrolla un proceso de conversión que unifica todas las etiquetas al formato entero binario (0 para correos legítimos, 1 para correos de phishing). Durante este paso, además se eliminan registros con etiquetas inválidas, ambiguas o ausentes que podrían comprometer la integridad del proceso de clasificación.
- **2. Eliminación de registros inválidos:** En esta fase se localizan y descartan registros con deficiencias estructurales críticas. Esto incluye campos esenciales vacíos (cuando tanto asunto como cuerpo están ausentes simultáneamente), inconsistencias de formato, o datos corrompidos. Sin embargo, esta etapa va más allá, también incorpora la detección y eliminación de registros duplicados mediante un sistema de tres niveles. Primero, duplicados exactos basados en la concatenación completa de asunto y cuerpo; segundo, duplicados identificados únicamente por contenido idéntico del cuerpo del mensaje; y tercero, duplicados completos que consideran la totalidad de campos disponibles. La eliminación de duplicados resulta crucial para evitar sesgos tanto en el entrenamiento como en la evaluación de los modelos.
- **3. Filtrado por idioma:** La fase final implementa un filtrado lingüístico para conservar exclusivamente correos electrónicos redactados en inglés. De esta manera se garantiza la homogeneidad necesaria para aplicar las técnicas de detección. Se emplea detección automática de idioma sobre el texto concatenado de asunto y cuerpo de cada correo, estableciendo un umbral mínimo de 10 caracteres para asegurar la fiabilidad de la clasificación lingüística.

El proceso incorpora controles de calidad que verifican la consistencia de la estructura de datos, la validez de tipos de datos y la coherencia de las distribuciones de clases tras cada transformación. Los resultados detallados se presentan en la sección 5.1. Cabe destacar que el proceso de limpieza prioriza la calidad sobre el volumen, aplicando un enfoque conservador. Solo se eliminan aquellos registros que realmente comprometerían la integridad del entrenamiento o la evaluación posterior. Además, cada transformación se documenta exhaustivamente mediante reportes automatizados, lo que permite la trazabilidad completa del proceso y garantiza la reproducibilidad de los resultados.

4.3.3. Combinación de datasets

Tras completar el proceso de limpieza individual, se procede a la creación de dos conjuntos de datos combinados que agrupan los datasets según sus características estructurales. Esta estrategia de agrupación permite evaluar el impacto de la disponibilidad de metadatos adicionales en la efectividad de las técnicas de detección implementadas, constituyendo una aproximación metodológica fundamental para validar la robustez y capacidad de generalización de los modelos desarrollados.

- **Dataset Combinado 1 (Campos completos):** Este conjunto incluye únicamente los 9 datasets que proporcionan información completa de metadatos: remitente (sender), destinatario (receiver), fecha (date), asunto (subject), cuerpo (body) y presencia de URLs (urls), además de la etiqueta de clasificación (label). Con 180.270 registros totales, presenta una distribución bastante equilibrada con 52,65 % de correos legítimos (94.919 registros) y 47,35 % de correos de phishing (85.351 registros). Se excluyen los datasets Ling y Enron debido a que no disponen de los metadatos adicionales requeridos para esta configuración.
- **Dataset Combinado 2 (Campos básicos):** Este conjunto agrupa todos los 11 datasets utilizando únicamente los campos fundamentales comunes: asunto (subject), cuerpo (body) y etiqueta de clasificación (label). Con 212.250 registros totales, mantiene un equilibrio similar con 53,18 % de correos legítimos (112.884 registros) y 46,82 % de correos de phishing (99.366 registros). Esta configuración garantiza la máxima compatibilidad entre todos los conjuntos de datos y permite evaluar la efectividad de las técnicas basándose exclusivamente en el contenido textual de los correos electrónicos.

Ambos datasets combinados mantienen distribuciones de clases naturalmente equilibradas sin necesidad de aplicar técnicas de balanceo artificial como *SMOTE* o *undersampling*. Esta característica elimina el riesgo de introducir sesgos sintéticos en los datos y garantiza que los resultados obtenidos reflejen la efectividad real de los modelos sobre patrones auténticos de phishing y correos legítimos.

La principal ventaja de esta estrategia de combinación está en la creación de datasets masivos (180.270 y 212.250 registros) y heterogéneos que representan múltiples décadas de evolución del phishing (1998-2022), diferentes vectores de ataque (fraude nigeriano, phishing corporativo, ataques genéricos) y diversos contextos comunicativos (académico, corporativo). Esta diversidad temporal y tipológica permite evaluar la capacidad real de

los modelos para generalizar entre diferentes tipos de amenazas, proporcionando una validación más rigurosa que la obtenida mediante datasets individuales homogéneos. Además, el volumen considerable de ambos datasets combinados facilita el entrenamiento robusto de modelos de deep learning que requieren grandes cantidades de datos para alcanzar su potencial máximo.

Esta estrategia de combinación dual permite evaluar sistemáticamente si la incorporación de metadatos estructurales mejora significativamente la capacidad de detección de las técnicas implementadas, proporcionando evidencia empírica sobre la importancia relativa del contenido textual frente a la información contextual en la detección de phishing.

4.3.4. Detección por heurísticas

La implementación de técnicas heurísticas se estructura en múltiples iteraciones experimentales que exploran diferentes enfoques de combinación y ponderación de reglas de detección. El desarrollo se fundamenta en la creación de un conjunto extenso de reglas basadas en patrones identificados en la literatura y características comunes en correos de phishing.

El proceso de desarrollo heurístico se inicia con un **enfoque binario simple**, donde la detección de cualquier patrón específico resulta en la clasificación inmediata del correo como phishing. Este método implementa reglas categóricas como la presencia de dominios acortadores, palabras clave críticas en el asunto, o patrones de ofuscación evidentes. La simplicidad de este enfoque permite una implementación rápida pero presenta limitaciones significativas en términos de precisión debido a la rigidez de las decisiones binarias.

Posteriormente, se desarrolla un **sistema de puntuación ponderada** que asigna valores numéricos específicos a cada regla detectada, acumulando una puntuación total que se compara contra un umbral de clasificación. Este enfoque permite mayor flexibilidad al considerar la combinación de múltiples indicadores débiles que, en conjunto, pueden señalar un correo sospechoso.

Se exploran también **variaciones de umbralización adaptativa** que ajustan los límites de clasificación según las características específicas de cada grupo de datasets. Los datasets con únicamente campos de asunto y cuerpo (Grupo 2) utilizan un umbral menor que aquellos con metadatos completos (Grupo 1) para compensar la información adicional disponible.

El conjunto de reglas heurísticas implementadas abarca múltiples categorías de análisis:

- **Análisis de contenido textual:**
  - Detección de palabras clave sospechosas en asunto (peso: 4 puntos)
  - Frecuencia de términos alarmistas en cuerpo (peso: 1 punto por término)
  - Identificación de frases específicas de estafas (peso: 4 puntos)
  - Presencia de terminología financiera sospechosa (peso: 2 puntos)

- **Análisis de URLs:**
  - Detección de servicios de acortamiento de URLs (peso: 3 puntos)
  - Identificación de dominios frecuentemente utilizados en phishing (peso: 5 puntos)
  - Análisis de longitud excesiva de dominios (más de 30 caracteres, peso: 2 puntos)
  - Evaluación de estructura de subdominios (peso: 2-4 puntos según complejidad)
  - Detección de caracteres especiales en URLs (peso: 1 punto)
  - Identificación de codificación Punycode (peso: 2 puntos)
- **Análisis lingüístico:**
  - Detección de palabras mal escritas mediante diccionario inglés (peso: 1-2 puntos)
  - Identificación de patrones de ofuscación textual (peso: 1 punto)
  - Análisis de uso excesivo de mayúsculas (peso: 2 puntos)
- **Análisis de metadatos (aplicable únicamente al Grupo Combinado 1):**
  - Presencia de nombre completo en campo remitente (peso: 1 punto)
  - Detección de Punycode en dirección de remitente (peso: 2 puntos)
  - Identificación de dominios sospechosos en remitente (peso: 5 puntos)
  - Análisis de múltiples destinatarios (peso: 1 punto)
  - Detección de envío en fin de semana (peso: 2 puntos)
  - Identificación de envío fuera de horario laboral (peso: 1 punto)

Se establecen umbrales diferenciados según la disponibilidad de metadatos:

- **Datasets Grupo 1** (6 campos): Umbral de 6 puntos
- **Datasets Grupo 2** (2 campos): Umbral de 5 puntos

El sistema se diseña con arquitectura modular que permite la activación/desactivación individual de reglas y el ajuste dinámico de pesos, facilitando la experimentación con diferentes combinaciones y la optimización empírica de parámetros.

4.3.5. Detección por Machine Learning

La implementación de técnicas de aprendizaje automático se estructura en tres fases secuenciales que optimizan el procesamiento de datos textuales para la clasificación binaria de correos electrónicos: selección de campos, preprocesamiento especializado y entrenamiento con optimización de modelos.

a) Selección de campos y justificación

Para la implementación de algoritmos de aprendizaje automático se utiliza exclusivamente la información textual contenida en los campos 'subject' y 'body' de los correos electrónicos. Esta decisión se fundamenta en las limitaciones inherentes de los algoritmos de ML tradicionales para procesar eficientemente metadatos heterogéneos como direcciones de correo, fechas y URLs sin transformaciones complejas adicionales. Por tanto, se procesan los datasets iniciales utilizando únicamente los campos textuales esenciales, además del grupo combinado 2 para ver la capacidad de generalización que se puede llegar a tener.

b) Preprocesamiento específico para Machine Learning

El preprocesamiento para algoritmos de aprendizaje automático requiere transformaciones específicas que difieren significativamente del procesamiento aplicado a técnicas heurísticas. Se implementa un pipeline de procesamiento de lenguaje natural que maximiza la calidad de las características extraídas del texto.

- **1. Expansión de contracciones:** Se aplica normalización lingüística que convierte formas contraídas del inglés a su forma expandida (ejemplo: "don't"→ "do not", "we're"→ "we are"). Esta transformación garantiza consistencia en el vocabulario y evita que variaciones ortográficas de la misma expresión se traten como términos diferentes.
- **2. Limpieza de contenido no textual:** Se eliminan sistemáticamente elementos que no aportan valor semántico para la clasificación: URLs completas, direcciones de correo electrónico, y caracteres especiales no alfabéticos. Esta limpieza reduce el ruido en el vocabulario y mejora la eficiencia computacional del procesamiento posterior.
- **3. Tokenización y etiquetado gramatical:** Se aplica tokenización avanzada que segmenta el texto en palabras, seguida de etiquetado gramatical (POS tagging) que identifica la categoría morfosintáctica de cada palabra (sustantivo, verbo, adjetivo...). Esta información contextual es esencial para la fase posterior de lematización.
- **4. Lematización contextual:** Se implementa lematización que reduce las palabras a su forma canónica considerando su contexto gramatical. A diferencia de la *stemming* simple, la lematización utiliza las etiquetas POS para determinar la forma base correcta (ejemplo: *"running"* como verbo → *"run"*, *"running"* como sustantivo → *"running"*).
- **5. Eliminación de palabras vacías:** Se aplica filtrado de stopwords posterior a la lematización, eliminando palabras de alta frecuencia pero bajo valor discriminativo (*"the"*, *"and"*, *"is"*, etc.). Esta fase se ejecuta después de la lematización para evitar la eliminación inadecuada de palabras que podrían tener valor semántico en contextos específicos.
- **6. Combinación de campos:** Se concatenan los campos asunto y cuerpo en un único campo de texto preprocesado, manejando robustamente valores nulos y tipos de datos inconsistentes. Esta unificación permite el procesamiento homogéneo independientemente de la estructura original del dataset.

- **7. División estratificada de datos:** Se establece una división estratificada 80/20 para entrenamiento y evaluación, manteniendo las proporciones originales de clases en ambos conjuntos. Esta división se realiza con semilla fija (*random_state=42*) para garantizar reproducibilidad de resultados.
- **8. Vectorización TF-IDF:** La transformación de texto preprocesado a representaciones numéricas se realiza mediante Term Frequency-Inverse Document Frequency (TF-IDF), configurado específicamente para optimizar el rendimiento en tareas de detección de phishing. Para la configuración de parámetros TF-IDF, se establecen valores que balancean la riqueza del vocabulario con la eficiencia computacional del entrenamiento: limitación a 5.000 características más relevantes, eliminación de términos que aparecen en menos de 2 documentos para reducir ruido, y exclusión de términos presentes en más del 95 % de documentos para eliminar palabras demasiado comunes. Se incluyen tanto unigramas como bigramas para capturar tanto palabras individuales como secuencias de dos palabras que pueden tener significado específico en contextos de phishing. Esta configuración permite detectar frases características como *"urgent action"* o *"verify account"*. Se aplica un filtrado de tokens que mantiene solo palabras de al menos 3 caracteres alfanuméricos, eliminando abreviaciones muy cortas que típicamente aportan poco valor discriminativo pero incrementan la dimensionalidad del espacio de características (como *"to"*, *"a"* o *"up"*).

Para ilustrar el proceso de preprocesamiento (hasta la división de datos), se presenta la transformación de un correo aleatorio de phishing real del dataset:

Asunto original: "Never agree to be a loser"

Cuerpo original: "Buck up, your troubles caused by small dimension will soon be over! Become a lover no woman will be able to resist! http://whited*ne.com/ come. Even as Nazi tanks were rolling down the streets, the dreamersphilosopher or a journalist. He was still not sure.I do the same."

Texto preprocesado resultante: "never agree loser buck trouble cause small dimension soon become lover woman able resist come even nazi tank roll street dreamersphilosopher journalist still sure"

Esta transformación evidencia la eliminación de URLs, signos de puntuación, la lematización de palabras (troubles → trouble, caused → cause), y la concatenación efectiva de asunto y cuerpo en una representación textual limpia y normalizada.

c) Entrenamiento y optimización de modelos

Se implementan los cuatro algoritmos de aprendizaje automático comentados en la Justificación de este capítulo [4.2]: Regresión Logística, Random Forest, Support Vector Machine y Naive Bayes.

La fase de entrenamiento implementa una metodología rigurosa que combina validación cruzada con búsqueda exhaustiva de hiperparámetros para cada algoritmo seleccionado:

- **Búsqueda de hiperparámetros:** Se aplica Grid Search exhaustivo para cada algoritmo, explorando espacios de parámetros específicamente diseñados para cada técnica. Para Regresión Logística se optimizan el parámetro de regularización C, el número máximo de iteraciones y el solver; para Random Forest se ajustan el número de estimadores, el tipo de asignación de pesos a las clases y la profundidad máxima; para SVM se exploran diferentes kernels, se ajusta un parámetro de regularización (C) y otro que define la influencia de cada punto en kernels no lineales (gamma); para Naive Bayes se optimiza el parámetro de suavizado alpha.
- **Validación cruzada:** Se implementa validación cruzada k-fold con k=5, garantizando evaluación robusta de la efectividad y reduciendo el riesgo de sobreajuste. La validación cruzada se aplica durante la búsqueda de hiperparámetros, utilizando F1-score como métrica de optimización para manejar adecuadamente datasets con distribuciones de clases desbalanceadas.
- **Persistencia de modelos:** Se implementa un sistema de almacenamiento que guarda tanto los modelos entrenados como los vectorizadores TF-IDF, permitiendo la reutilización eficiente y la aplicación posterior sobre nuevos datos sin necesidad de re-entrenamiento.

4.3.6. Detección por Deep Learning

La implementación de técnicas de aprendizaje profundo se centra en la evaluación de modelos de última generación para la detección de phishing, aprovechando las capacidades avanzadas de procesamiento de lenguaje natural que ofrecen las arquitecturas neuronales modernas. Debido a los elevados requisitos computacionales y temporales que caracterizan el entrenamiento de estos modelos, especialmente en términos de recursos GPU y tiempo de procesamiento, la evaluación se limita estratégicamente a los dos datasets combinados que proporcionan la mayor representatividad y volumen de datos.

Esta decisión permite concentrar los recursos computacionales disponibles en la evaluación sobre conjuntos de datos que maximizan la diversidad y el volumen, garantizando resultados más significativos y generalizables que los obtenibles mediante evaluaciones fragmentadas sobre los datasets individuales.

4.3.6.1. Detección con BERT

La implementación de BERT para la detección de phishing se desarrolla mediante un pipeline especializado que aprovecha las capacidades de representación contextual bidireccional del modelo. El proceso se estructura en dos fases principales: preprocesamiento específico para la arquitectura transformer y optimización exhaustiva de hiperparámetros mediante búsqueda automática.

a) Preprocesamiento específico para BERT

El preprocesamiento para BERT requiere transformaciones particulares que difieren

significativamente de las aplicadas en técnicas de Machine Learning tradicional, adaptándose a los requerimientos específicos de la arquitectura transformer:

- **1. Preparación diferenciada por grupo:** Se implementa un sistema de concatenación adaptativo que estructura la información según la disponibilidad de metadatos. Para el Grupo Combinado 1, se concatenan metadatos estructurados mediante tokens especiales: "[SENDER]", "[RECEIVER]", "[DATE]", "[URL]" y "[TEXT]", creando una secuencia enriquecida que permite a BERT procesar tanto contenido textual como información contextual. Para el Grupo Combinado 2, se mantiene únicamente el contenido textual (subject + body).
- **2. División estratificada de datos:** Al igual que en Machine Learning, se implementa división estratificada 80/20 para entrenamiento y evaluación, manteniendo las proporciones originales de clases en ambos conjuntos. Esta división utiliza semilla fija (*random_state=42*) para garantizar reproducibilidad y comparabilidad de resultados entre diferentes configuraciones de hiperparámetros.
- **3. Tokenización con BertTokenizer:** Se aplica tokenización específica utilizando el tokenizador pre-entrenado *'bert-base-uncased'* (versión de BERT sin distinción de mayúsculas), que segmenta el texto en subpalabras según el vocabulario *WordPiece* de BERT. Esta tokenización preserva la compatibilidad con las representaciones aprendidas durante el pre-entrenamiento del modelo.
- **4. Configuración de longitud máxima:** Se establece una longitud máxima de 512 tokens, correspondiente al límite arquitectónico de BERT. Los textos que exceden esta longitud se truncan, mientras que los textos más cortos se completan con tokens de padding para homogeneizar las dimensiones de entrada.
- **5. Generación de máscaras de atención:** Se crean máscaras de atención que indican a BERT qué tokens corresponden a contenido real y cuáles son padding, permitiendo que el modelo ignore las posiciones artificiales durante el procesamiento.

b) Entrenamiento y optimización de modelo

Para configurar el modelo, se utiliza *TFBertForSequenceClassification*, un modelo de inteligencia artificial preentrenado basado en 'bert-base-uncased', configurado para clasificación binaria, como es este caso (phishing o legítimo). El modelo se somete a un proceso de *fine-tuning*, donde se aprovecha el conocimiento lingüístico general que BERT adquirió durante su preentrenamiento en grandes corpus de texto, y se adapta específicamente para la tarea de detección de phishing mediante el entrenamiento con el dataset de emails etiquetados. Durante este proceso, todos los pesos del modelo BERT se actualizan para optimizar el rendimiento en la clasificación binaria, permitiendo que el modelo combine su comprensión contextual del lenguaje con patrones específicos de emails maliciosos.

La optimización se realiza mediante *Optuna*, una herramienta que automáticamente encuentra la mejor configuración del modelo usando búsqueda bayesiana (un método inteligente que aprende de intentos anteriores) que explora eficientemente el espacio de hiperparámetros. El espacio de búsqueda incluye *learning rate*, *batch size*, *número de épocas* y *decay rate*. Se utiliza *F1-score* como métrica de optimización, con *Early Stopping*,

que significa que se para el entrenamiento cuando el modelo deja de mejorar y guardado automático de la mejor versión durante el entrenamiento.

La optimización de hiperparámetros mediante búsqueda bayesiana representa un componente crítico para maximizar la efectividad de los modelos de deep learning en detección de phishing, ya que permite alcanzar el equilibrio óptimo entre capacidad de aprendizaje y capacidad de generalización. Este proceso automatizado evita tanto el *underfitting* como el *overfitting* mediante la exploración inteligente del espacio de hiperparámetros, utilizando la información de evaluaciones previas para mejorar.

La prevención del *overfitting* se logra mediante la identificación automática de configuraciones que maximizan el rendimiento en datos de validación no vistos durante el entrenamiento, mientras que la configuración de *early stopping* detiene el proceso cuando se detecta degradación en las métricas de validación. Por el contrario, el *underfitting* se evita explorando configuraciones de mayor complejidad (learning rates más altos, más épocas, arquitecturas más profundas) cuando el modelo muestra capacidad de aprender patrones adicionales sin comprometer la generalización.

Además, la optimización automatizada reduce sustancialmente el tiempo total de desarrollo al eliminar la necesidad de experimentación manual extensiva. Mientras que un enfoque tradicional de prueba y error podría requerir semanas de experimentación, la búsqueda bayesiana converge típicamente hacia configuraciones óptimas en cuestión de horas o días, concentrando los recursos computacionales en las configuraciones más prometedoras identificadas algorítmicamente.

La metodología implementada garantiza que BERT opere en condiciones óptimas, aprovechando tanto su capacidad de representación contextual como la optimización automática de hiperparámetros para maximizar la efectividad en la tarea de detección de phishing.

El proceso de entrenamiento y clasificación de BERT se puede visualizar de manera resumida en la figura 4.2, donde los correos de entrenamiento del dataset combinado 1 se tokenizan y realizan *fine-tunning* sobre el modelo preentrenado *TFBertForSequenceClassification*, buscando los mejores hiperparámetros con *Optuna*. Tras entrenar y validar este modelo, cuando entra un nuevo correo de test con el formato requerido, este se tokeniza y el modelo infiere si es legítimo o phishing.

4.3.6.2. Detección con LSTM y BiLSTM

La implementación de modelos LSTM y BiLSTM para la detección de phishing se desarrolla mediante un pipeline especializado que aprovecha las capacidades de procesamiento secuencial de estas arquitecturas neuronales. El proceso se estructura en tres fases principales: preprocesamiento específico para modelos recurrentes, implementación diferenciada de ambas arquitecturas para el entrenamiento y optimización exhaustiva de hiperparámetros mediante Keras Tuner.

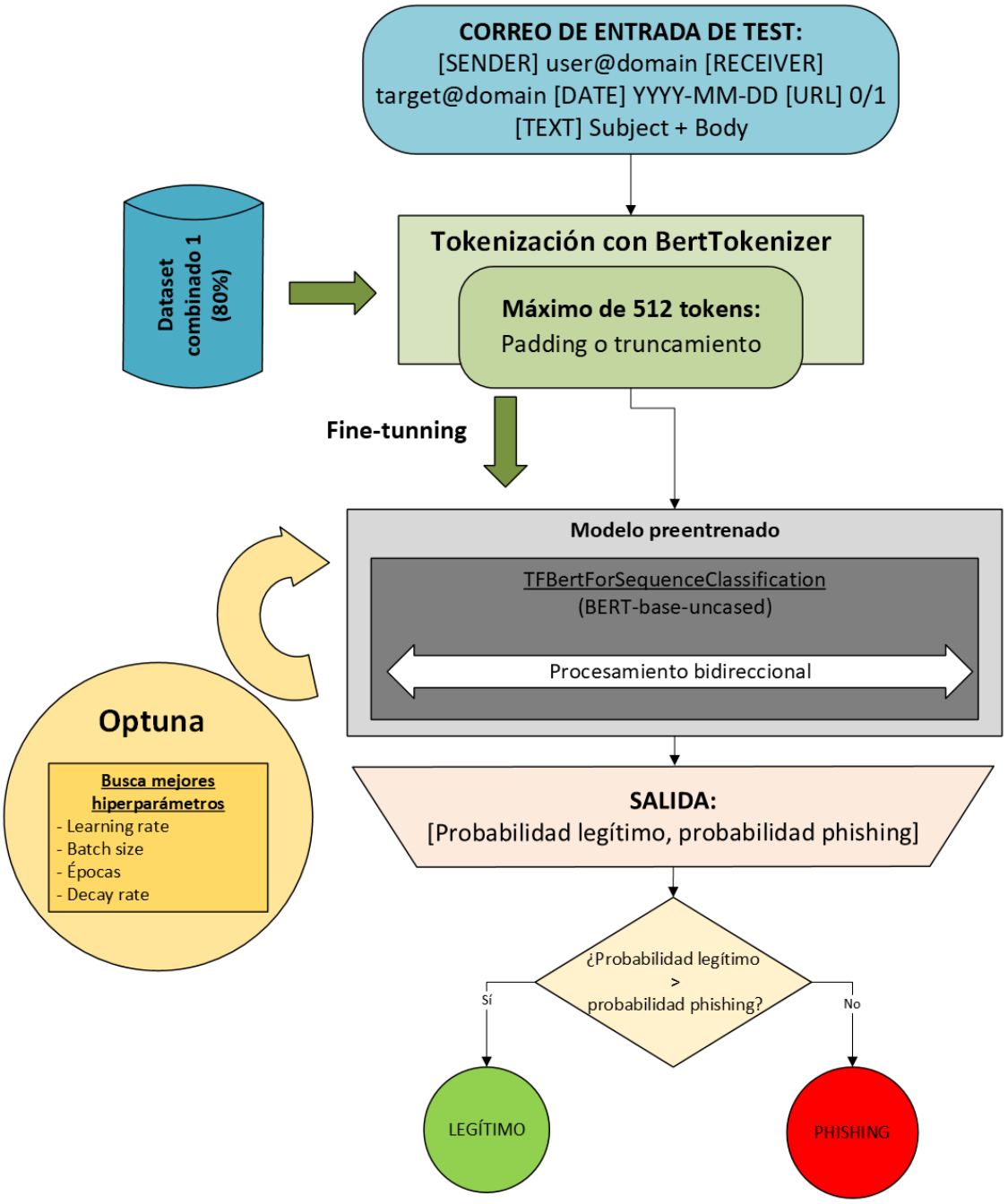

Figura 4.2: Proceso de entrenamiento y clasificación de BERT

a) Preprocesamiento específico para modelos recurrentes

El preprocesamiento para LSTM y BiLSTM requiere transformaciones específicas que difieren un poco de las aplicadas en BERT, adaptándose a los requerimientos de entrada secuencial de las redes neuronales recurrentes:

- **1. Preparación diferenciada por grupo:** Se implementa un sistema de estructuración de datos igual que en BERT. Para el Grupo Combinado 1, se concatenan metadatos mediante tokens estructurales específicos: "[SENDER]", "[RECEIVER]", "[DATE]", "[URL]" y "[TEXT]", creando secuencias enriquecidas que permiten a los modelos procesar información contextual junto al contenido textual. Para el Grupo Combinado 2, se mantiene únicamente el contenido textual preprocesado (subject + body).
- **2. División estratificada de datos:** Se aplica la misma división que en BERT, train/test (80/20).
- **3. Tokenización con vocabulario limitado:** Se utiliza una herramienta llamada *Keras Tokenizer* para dividir el texto en palabras, pero solo se consideran las 20.000 palabras más comunes que aparecen en los datos de entrenamiento. Además, se incluye un marcador especial para representar aquellas palabras que no están dentro de esas 20.000, de modo que el sistema pueda manejar palabras nuevas o poco frecuentes sin problemas.
- **4. Conversión a secuencias numéricas:** Los textos preprocesados se transforman a secuencias de enteros donde cada número representa una palabra específica del vocabulario. Esta representación es esencial para el procesamiento por parte de las capas de *embedding*.
- **5. Cálculo dinámico de longitud máxima:** Se determina la longitud máxima de secuencia basándose en el percentil 95 de las longitudes del conjunto de entrenamiento, así se conserva la información importante de correos largos sin desperdiciar recursos computacionales en secuencias innecesariamente extensas.
- **6. Aplicación de padding:** Se homogeneizan las longitudes de secuencia mediante padding posterior (*post-padding*) y truncamiento posterior (*post-truncating*), garantizando dimensiones consistentes para el procesamiento en lotes.
- **7. Persistencia de recursos:** Se almacenan tanto el tokenizador entrenado (modelo que transforma el texto en números) como los datos procesados en formato comprimido (joblib), permitiendo volver a usarlos fácilmente en el futuro y asegura que los experimentos se puedan repetir exactamente igual.

b) Entrenamiento y optimización de los modelos

Las implementaciones de LSTM y BiLSTM siguen arquitecturas distintas:

- **Arquitectura LSTM:** Se implementa mediante una arquitectura secuencial simplificada que incluye capa de embedding, *dropout* para regularización, capa LSTM unidireccional y capa densa de salida. Se exploran diferentes optimizadores para la compilación con *learning rates* logarítmicamente distribuidos.

- **Arquitectura BiLSTM:** Se desarrolla una arquitectura más compleja que incorpora procesamiento bidireccional mediante capas *Bidirectional LSTM*. La implementación incluye capa de embedding, una primera capa BiLSTM, dropout para regularización, segunda capa BiLSTM, otra de dropout, una capa densa intermedia con activación ReLU para aumentar la capacidad representacional junto con dropout final y la capa densa final de salida. Al igual que la arquitectura LSTM, también se exploran diferentes optimizadores para la compilación con *learning rates* logarítmicamente distribuidos.

Ambas arquitecturas están optimizadas para *cuDNN* (biblioteca especializada de *NVIDIA*), garantizando máximo rendimiento al usar tarjetas gráficas GPU para acelerar los cálculos.

La optimización de hiperparámetros en arquitecturas recurrentes presenta desafíos específicos relacionados con el procesamiento secuencial de texto, donde la configuración inadecuada puede resultar en problemas de gradiente desvanecido o explosivo que comprometen el aprendizaje de dependencias a largo plazo. La implementación utiliza *Keras Tuner* con algoritmo *Hyperband*, una estrategia de optimización especialmente eficiente para modelos de deep learning que combina búsqueda aleatoria con terminación temprana inteligente, eliminando configuraciones poco prometedoras sin desperdiciar recursos computacionales.

El algoritmo *Hyperband* resulta particularmente valioso para modelos recurrentes debido a su capacidad para evaluar múltiples configuraciones de hiperparámetros en paralelo, incluyendo dimensiones de *embedding*, número de unidades LSTM, tasas de *dropout* y optimizadores, identificando rápidamente aquellas combinaciones que muestran convergencia prometedora. La identificación automática de la dimensionalidad óptima de embedding permite que el modelo capture relaciones semánticas complejas entre palabras sin sobredimensionar el espacio de representación, lo cual es particularmente relevante para detectar patrones de ofuscación y manipulación textual comunes en correos de phishing.

Se configura *early stopping* que detiene el entrenamiento automáticamente cuando el modelo deja de mejorar en los datos de validación, evitando así el sobreajuste que podría comprometer la capacidad de generalización ante nuevas variantes de ataques. El ajuste del *dropout* resulta especialmente crítico en estas arquitecturas, ya que un valor insuficiente puede llevar a memorización de patrones específicos del conjunto de entrenamiento, mientras que un valor excesivo puede impedir que el modelo aprenda las secuencias temporales que caracterizan las técnicas de ingeniería social utilizadas en phishing. Además, se implementa *model checkpointing* que guarda automáticamente las mejores versiones, garantizando que se conserve la configuración óptima.

El proceso de entrenamiento y clasificación de ambos modelos se puede visualizar de manera resumida en la figura 4.3, donde los correos de entrenamiento del dataset combinado 1 se tokenizan y se introducen en la red neuronal para entrenarla, buscando los mejores hiperparámetros con *Keras Tuner*. Tras entrenar y validar este modelo, cuando entra un nuevo correo de test con el formato requerido, este se tokeniza y el modelo infiere si es legítimo o phishing.

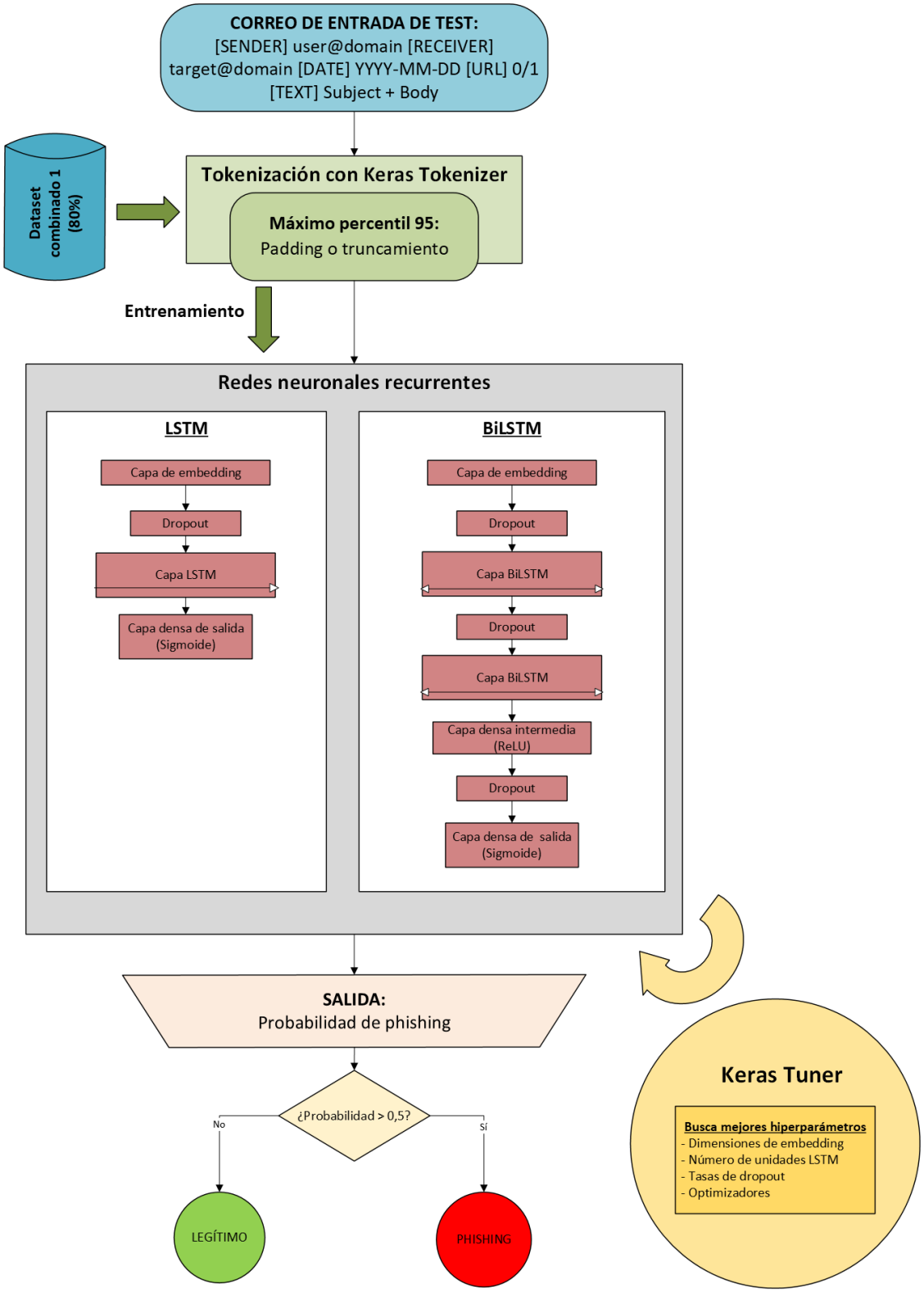

Figura 4.3: Proceso de entrenamiento y clasificación de modelos recurrentes

CAPÍTULO 5

Resultados

Este capítulo presenta los resultados obtenidos durante el desarrollo e implementación de las técnicas de detección de phishing seleccionadas. Los resultados se estructuran en múltiples secciones que abarcan desde el proceso de preparación de datos hasta la evaluación comparativa de la efectividad de cada familia de técnicas implementadas.

5.1– Resultados de limpieza general

La figura 5.1 presenta un resumen comparativo del proceso de limpieza aplicado a la colección completa de datasets, mostrando la evolución desde los datos iniciales hasta la versión final procesada.

RESUMEN TOTAL - Comparativa General

Resumen consolidado de todos los datasets procesados

Versión	Archivos Procesados	Total Registros	Distribución Labels	Total Duplicados	Duplicados (Subject+Body)	Duplicados (Body)	Total Campos Faltantes
iniciales	11	**218,086**	1: 47.88%, 0: 52.11%	105	616	641	16,271
preprocesamiento_general	11	**217,314**	1: 47.91%, 0: 52.09%	0	0	0	12,540
solo_ingles	11	**212,250**	1: 46.81%, 0: 53.18%	0	0	0	11,676

Figura 5.1: Resumen de Limpieza General realizada

El proceso completo de limpieza resultó en una reducción controlada del volumen de datos, pasando de 218.086 registros iniciales a 212.250 registros finales, representando una tasa de conservación del 97,32 %. Esta alta tasa de retención evidencia la calidad inicial

relativamente buena de la colección curada, validando la selección de fuentes de datos especializadas.

La **normalización de etiquetas** identificó y corrigió 772 registros con inconsistencias de formato, eliminando valores inválidos como etiquetas decimales no binarias, cadenas de texto no interpretables y valores nulos. Esta fase redujo el conjunto de 218.086 a 217.314 registros, manteniendo una distribución de clases estable.

La **eliminación de registros inválidos** procesó 1.362 duplicados identificados mediante el sistema de detección multinivel implementado. Los duplicados se concentraron principalmente en los datasets TREC, sugiriendo posibles errores en la recopilación original o inclusión múltiple de correos circulares. La reducción de campos faltantes de 16.271 a 12.540 evidencia la efectividad de esta fase en la mejora de la calidad estructural.

El **filtrado por idioma** constituyó la transformación más significativa, eliminando 5.064 registros (2,33 % del total) que no cumplían el criterio de contenido en inglés. Esta reducción era esperada dado el carácter internacional de algunos datasets, particularmente aquellos que incluían comunicaciones corporativas multinacionales.

El proceso de limpieza mantuvo una distribución de clases relativamente equilibrada, con una ligera variación desde la proporción inicial de 47,88 % phishing y 52,11 % legítimos hasta 46,81 % phishing y 53,18 % legítimos en la versión final. Esta estabilidad en la distribución es crucial para evitar sesgos en el entrenamiento de los modelos de clasificación.

El análisis de integridad de campos reveló mejoras significativas en la completitud de datos. Los campos críticos (subject, body, label) alcanzaron una completitud superior al 99,5 % en todos los datasets tras la limpieza. Los campos auxiliares mantuvieron patrones específicos: el campo 'receiver' presentó 11.676 valores faltantes concentrados principalmente en datasets especializados, mientras que los campos 'sender' y 'date' conservaron alta completitud en la mayoría de conjuntos.

El análisis detallado por dataset individual, incluyendo estadísticas específicas de transformaciones y distribuciones de campos faltantes, se presenta en el Anexo [7.1].

5.2– Resultados de detección por heurísticas

La evaluación de las técnicas heurísticas se realiza sobre los 11 datasets individuales y los dos grupos combinados, aplicando las configuraciones de reglas y umbrales desarrolladas durante la fase de implementación.

La tabla 5.1 presenta los resultados de evaluación de las técnicas heurísticas aplicadas a cada dataset individual y a los grupos combinados, mostrando las métricas de efectividad fundamentales.

Tabla 5.1: Resultados de detección heurística por dataset y grupo

Dataset	**Grupo**	***Accuracy***	***Precision***	***Recall***	***F1-score***
CEAS_08	1	0,403	0,427	0,213	0,285
CEAS_08	2	0,417	0,422	0,129	0,198
Enron	2	0,550	0,847	0,038	0,072
Ling	2	0,844	0,672	0,090	0,158
SpamAssasin	1	0,636	0,418	0,665	0,514
SpamAssasin	2	0,681	0,459	0,576	0,511
Nazario	1	0,386	1,000	0,386	0,557
Nazario	2	0,252	1,000	0,252	0,403
Nazario_5	1	0,515	0,519	0,386	0,443
Nazario_5	2	0,509	0,517	0,252	0,339
Nigerian_5	1	0,752	0,734	0,825	0,777
Nigerian_5	2	0,793	0,805	0,798	0,802
Nigerian_Fraud	1	0,825	1,000	0,825	0,904
Nigerian_Fraud	2	0,798	1,000	0,798	0,888
TREC_05	1	0,606	0,469	0,396	0,430
TREC_05	2	0,642	0,544	0,271	0,362
TREC_06	1	0,669	0,331	0,424	0,372
TREC_06	2	0,711	0,350	0,294	0,319
TREC_07	1	0,404	0,430	0,280	0,339
TREC_07	2	0,406	0,385	0,146	0,212
Grupo Combinado 1	1	0,514	0,482	0,349	0,405
Grupo Combinado 2	2	0,539	0,520	0,211	0,300

A pesar de las múltiples iteraciones y refinamientos aplicados al sistema de puntuación ponderada, los resultados obtenidos revelan limitaciones significativas en la capacidad de detección de las técnicas heurísticas cuando se enfrentan a la diversidad y complejidad de los datasets evaluados.

5.2.1. Análisis de efectividad por tipología de dataset

Para facilitar la interpretación de los resultados, se organiza el análisis según las características estructurales y de distribución de los datasets evaluados.

Datasets especializados en phishing (100 % maliciosos): Los datasets Nazario y Nigerian_Fraud, que contienen exclusivamente correos de phishing, exhiben *precision* perfecta (1,000) debido a la ausencia de falsos positivos, pero revelan limitaciones significativas en *recall*. Nazario muestra *recall* de 0,386 en Grupo 1 que se reduce drásticamente a 0,252 en Grupo 2, indicando que las reglas heurísticas detectan únicamente entre el 25 % y 39 % de los ataques presentes. Nigerian_Fraud presenta mejor capacidad de detección con *recall* de 0,825 en Grupo 1 y 0,798 en Grupo 2, sugiriendo que las reglas están mejor calibradas para patrones específicos de fraude nigeriano que para técnicas generales de phishing.

Datasets con distribución equilibrada (40-60 % de phishing): Los datasets CEAS_08, Nigerian_5, Nazario_5, TREC_05 y TREC_07 permiten una evaluación más robusta de efectividad real, ya que la *accuracy* no se ve sesgada por clases predominantes. CEAS_08 muestra la efectividad más deficiente con *accuracy* de 0,403 y *recall* de 0,213 en Grupo 1, evidenciando las limitaciones de las heurísticas en contextos corporativos diversos. En contraste, Nigerian_5 alcanza la mejor efectividad global con *f1-score* de 0,777 en Grupo 1, confirmando la eficacia de las reglas para detectar fraudes específicos. Los datasets TREC presentan capacidades heterogéneas: TREC_05 muestra *accuracy* moderada (0,606 en Grupo 1), mientras que TREC_07 exhibe la efectividad más baja (*accuracy* de 0,404).

5.2.2. Comparación entre grupos

La comparación entre Grupo 1 (6 campos) y Grupo 2 (2 campos) revela patrones inconsistentes. En algunos casos como Nigerian_5 y Nigerian_Fraud, el Grupo 2 supera al Grupo 1 en *f1-score*, sugiriendo que los metadatos adicionales pueden introducir ruido en lugar de información útil. En contraste, datasets como TREC_05 y SpamAssasin muestran mayor efectividad en Grupo 1, indicando que ciertos tipos de phishing se benefician del análisis de metadatos.

5.2.3. Efectividad en datasets combinados

Los datasets combinados muestran una mala efectividad, con el Grupo Combinado 1 alcanzando *accuracy* de 0,514 y *f1-score* de 0,405, mientras que el Grupo Combinado 2 presenta *accuracy* de 0,539 pero *f1-score* inferior (0,300). El *recall* particularmente bajo en ambos grupos combinados (0,349 y 0,211 respectivamente) confirma las limitaciones de las técnicas heurísticas para generalizar entre diferentes tipos de datasets.

5.3– Resultados de detección por Machine Learning

La evaluación de los algoritmos de aprendizaje automático se realiza sobre los 11 datasets individuales (excepto Nazario y Nigerian_Fraud que al ser todos los correos phishing, no van a aportar nada al modelo por separado) y el dataset combinado del Grupo 2, aplicando la metodología de optimización de hiperparámetros y validación cruzada descrita.

La tabla 5.2 presenta las métricas de efectividad del mejor modelo para cada dataset, seleccionado según el *f1-score* obtenido tras la optimización con Grid Search.

Tabla 5.2: Resultados de detección por Machine Learning - Mejor modelo por dataset

Dataset	**Mejor Modelo**	***Accuracy***	***Precision***	***Recall***	***F1-score***
CEAS_08	SVM	0,9954	0,9975	0,9942	0,9959
Enron	SVM	0,9887	0,9846	0,9912	0,9879
Ling	Naive Bayes	0,9964	0,9891	0,9891	0,9891
SpamAssasin	Regresión Logística	0,9826	0,9785	0,9608	0,9696
Nazario_5	SVM	0,9950	0,9967	0,9933	0,9950
Nigerian_5	Regresión Logística	0,9968	1,0000	0,9939	0,9970
TREC_05	SVM	0,9808	0,9729	0,9759	0,9744
TREC_06	Regresión Logística	0,9780	0,9542	0,9503	0,9523
TREC_07	SVM	0,9922	0,9892	0,9966	0,9929
Grupo Combinado 2	SVM	0,9816	0,9817	0,9790	0,9803

Los resultados obtenidos demuestran la efectividad superior de estas técnicas comparado con enfoques heurísticos.

5.3.1. Análisis de efectividad por algoritmo

Support Vector Machine emerge como el algoritmo de mejor efectividad general, alcanzando la máxima efectividad en 6 de los 11 datasets evaluados. Sus *f1-scores* oscilan entre 0,9652 (SpamAssasin) y 0,9959 (CEAS_08), demostrando robustez excepcional entre diferentes tipologías de correos electrónicos. La capacidad de SVM para encontrar hiperplanos óptimos de separación en espacios de características de alta dimensionalidad generados por TF-IDF le confiere particular efectividad para manejar la variabilidad inherente en los patrones de phishing, especialmente cuando utiliza kernels RBF y lineales según las características específicas de cada dataset.

Regresión Logística ocupa la segunda posición en términos de efectividad general, destacando particularmente en *Nigerian_5* donde alcanza el mejor resultado (*f1-score* de 0,9970) y *SpamAssasin* (*f1-score* de 0,9696). La regresión logística muestra consistencia notable con *f1-scores* típicamente superiores a 0,95 en la mayoría de los datasets, evidenciando su naturaleza probabilística y eficiencia computacional en el entrenamiento que la convierte en una alternativa atractiva para implementaciones que requieren interpretabilidad de resultados y tiempos de respuesta rápidos.

Random Forest presenta eficacia competitiva posicionándose como tercera opción general, aunque mantiene *f1-scores* superiores a 0,93 en todos los casos evaluados. Su naturaleza de ensemble le permite manejar efectivamente la diversidad de patrones, combinando múltiples árboles de decisión para generar predicciones estables, especialmente destacable en datasets como *TREC_07* donde alcanza un *f1-score* de 0,9910.

Naive Bayes muestra la efectividad más modesta del conjunto, aunque alcanza el mejor resultado en el dataset *Ling* con un *f1-score* de 0,9891. A pesar de sus asunciones simplificadoras de independencia condicional, demuestra efectividad sorprendente en ciertos contextos de clasificación de texto, confirmando su utilidad como línea base robusta para tareas de detección de phishing, especialmente considerando sus tiempos de

entrenamiento extremadamente reducidos (típicamente inferiores a 3 segundos).

El análisis detallado de todas las métricas por dataset y algoritmo, incluyendo tiempos de entrenamiento y configuraciones óptimas de hiperparámetros, se presenta en el Anexo II del trabajo [7.2].

5.3.2. Efectividad en dataset combinado

El Grupo Combinado 2, que integra múltiples datasets con más de 212.000 registros, alcanza *f1-score* de 0,9803 con SVM como mejor modelo. Las métricas obtenidas (*precision* de 0,9817 y *recall* de 0,9790) indican un balance adecuado entre la capacidad de detectar correos de phishing y la precisión en las clasificaciones positivas, minimizando tanto falsos negativos como falsos positivos. Este equilibrio es crucial para implementaciones prácticas donde tanto la detección completa de amenazas como la minimización de interrupciones a usuarios legítimos son objetivos críticos.

La *accuracy* de 0,9816 sobre un dataset balanceado de esta escala y diversidad evidencia robustez operacional para implementaciones reales, confirmando la capacidad de generalización de los algoritmos de ML entre diferentes tipologías de phishing y contextos comunicativos.

5.4– Resultados de detección por Deep Learning

BERT, LSTM y BiLSTM han mostrado resultados realmente prometedores en la detección de phishing. Sin embargo, estos modelos exigen recursos computacionales considerablemente mayores que los métodos tradicionales, lo que plantea un interesante dilema entre precisión y eficiencia.

5.4.1. BERT

BERT ha destacado especialmente en la detección de phishing cuando se evalúa en ambos datasets combinados. Esto confirma algo que ya se podía intuir: los modelos transformer pre-entrenados pueden adaptarse brillantemente a tareas específicas de clasificación mediante *fine-tuning*.

Efectividad obtenida

La tabla 5.3 recoge las métricas alcanzadas por BERT después de una optimización exhaustiva de hiperparámetros usando búsqueda bayesiana.

Tabla 5.3: Resultados de BERT con hiperparámetros optimizados

Dataset	***Accuracy***	***Precision***	***Recall***	***F1-score***
Grupo Combinado 1	0,9893	0,9967	0,9807	0,9886
Grupo Combinado 2	0,9860	0,9939	0,9760	0,9849

BERT consigue *f1-scores* superiores a 0,98 en ambos datasets combinados, lo cual es bastante impresionante. El Grupo Combinado 1, que incluye metadatos adicionales, alcanza un *f1-score* de 0,9886, mientras que el Grupo Combinado 2 logra 0,9849. Esta diferencia revela que BERT puede extraer información discriminativa suficiente del contenido textual, aunque los metadatos adicionales sí aportan una mejora incremental.

Lo que más llama la atención es la *precision* excepcionalmente alta (0,9967 en Grupo 1, 0,9939 en Grupo 2). Esto significa que BERT prácticamente elimina los falsos positivos, algo crucial cuando se habla de implementaciones reales donde interrumpir a usuarios legítimos debe ser la excepción, no la regla. Por otro lado, el *recall* algo más moderado (0,9807 en Grupo 1, 0,9760 en Grupo 2) sugiere que aproximadamente un 2-2,4 % de correos de phishing podrían pasar desapercibidos. No obstante, este nivel de riesgo sigue siendo considerablemente bajo para aplicaciones prácticas.

Configuración óptima de hiperparámetros

Optuna identificó las configuraciones específicas que se detallan en la tabla 5.4, optimizando la efectividad para cada dataset particular.

Tabla 5.4: Hiperparámetros óptimos identificados para BERT

Dataset	**Learning Rate**	**Batch Size**	**Epochs**	**Decay**	**Best Epoch**
Grupo Combinado 1	3,85e-05	4	3	0,0101	2
Grupo Combinado 2	3,93e-05	8	3	0,0114	2

Proceso de entrenamiento

Las figuras 5.2 y 5.3 muestran las curvas de entrenamiento para ambos datasets y revelan patrones de convergencia rápida y estable.

El análisis de estas curvas muestra que BERT encuentra su configuración óptima ya en la segunda época para ambos datasets. Esto demuestra cómo aprovecha eficientemente su conocimiento pre-entrenado para adaptarse rápidamente a esta tarea específica. Sin embargo, a partir de la tercera época se observa que la pérdida de validación comienza a subir ligeramente mientras la de entrenamiento sigue bajando. Esta divergencia es una señal clara de que el modelo empieza a sobreajustarse (*overfitting*). Por ello, implementar *early stopping* en la segunda época no solo evita la degradación del rendimiento, sino que además valida la robustez de la configuración de hiperparámetros.

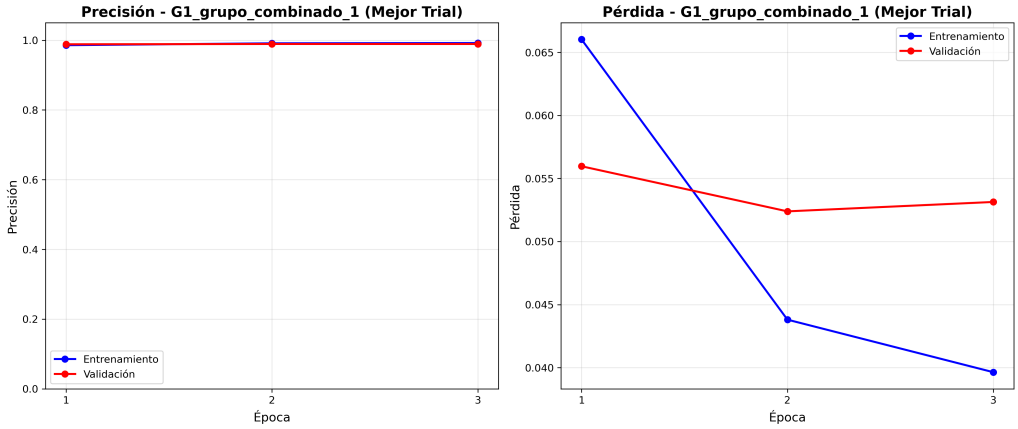

Figura 5.2: Curvas de entrenamiento y validación para BERT - Grupo Combinado 1

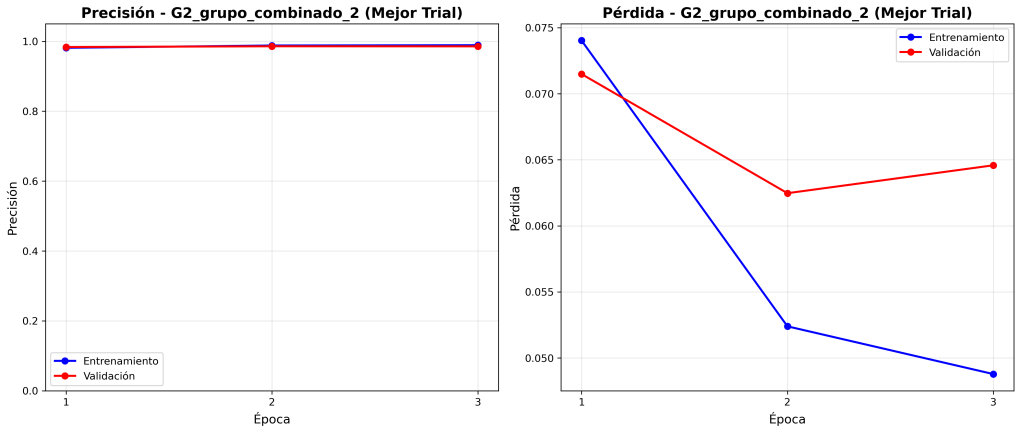

Figura 5.3: Curvas de entrenamiento y validación para BERT - Grupo Combinado 2

5.4.2. LSTM

LSTM demuestra capacidades sólidas para el procesamiento secuencial, aunque su rendimiento varía según el dataset que se analice.

Efectividad obtenida

La tabla 5.5 presenta las métricas conseguidas tras optimizar exhaustivamente los hiperparámetros mediante *Keras Tuner*.

Tabla 5.5: Resultados de detección con LSTM

Dataset	***Accuracy***	***Precision***	***Recall***	***F1-score***
Grupo Combinado 1	0,9914	0,9933	0,9885	0,9909
Grupo Combinado 2	0,9862	0,9850	0,9855	0,9852

LSTM también alcanza una efectividad notable en ambos datasets, con *f1-scores* por encima de 0,98. El Grupo Combinado 1 consigue un *f1-score* de 0,9909, superando ligeramente al Grupo Combinado 2 con su 0,9852. Esta diferencia revela que LSTM también se beneficia moderadamente de tener metadatos adicionales, ya que puede aprovechar el contexto para distinguir mejor entre correos legítimos y de phishing.

La *precision* es consistentemente alta (0,9933 en Grupo 1, 0,9850 en Grupo 2), lo que confirma que LSTM también minimiza eficazmente los falsos positivos. Además, el *recall* (0,9885 en Grupo 1, 0,9855 en Grupo 2) indica una capacidad sólida para detectar la gran mayoría de correos de phishing presentes en los datasets.

Configuración óptima de hiperparámetros

Keras Tuner identificó las configuraciones específicas que se detallan en la tabla 5.6, maximizando la efectividad para cada dataset.

Tabla 5.6: Hiperparámetros óptimos identificados para LSTM

Dataset	**Embedding Dim**	**LSTM Units**	**Dropout Rate**	**Optimizer**	**LR**	**Batch Size**	**Best Epoch**
Grupo Combinado 1	64	32	0,5	Adam	0,00497	64	10
Grupo Combinado 2	64	32	0,5	Adam	0,00497	64	8

Proceso de entrenamiento

Las figuras 5.4 y 5.5 ilustran las curvas de entrenamiento para ambos datasets. Aquí se ven patrones de convergencia estable, aunque con ligeras diferencias en la velocidad de aprendizaje.

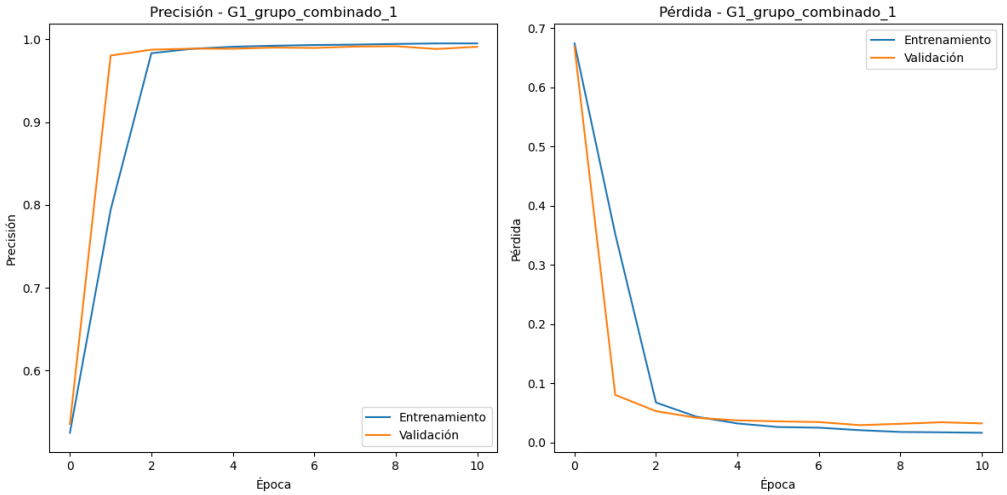

Figura 5.4: Curvas de entrenamiento y validación para LSTM - Grupo Combinado 1

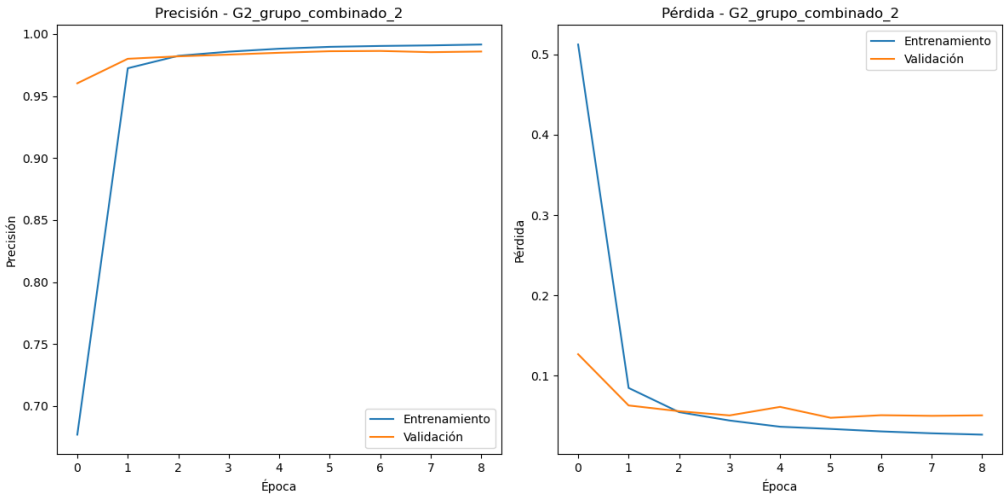

Figura 5.5: Curvas de entrenamiento y validación para LSTM - Grupo Combinado 2

El análisis revela que la convergencia óptima se produce en la época 10 para el Grupo Combinado 1 y en la época 8 para el Grupo Combinado 2. La configuración optimizada identifica hiperparámetros bastante conservadores que priorizan la generalización por encima de la capacidad de memorización. Esto se evidencia en la estabilidad de las curvas de validación, que no muestran signos pronunciados de sobreajuste.

5.4.3. BiLSTM

BiLSTM aprovecha el procesamiento bidireccional para capturar dependencias contextuales en ambas direcciones de las secuencias de texto.

Efectividad obtenida

La tabla 5.7 presenta las métricas obtenidas tras optimizar esta arquitectura más compleja.

Tabla 5.7: Resultados de detección con BiLSTM

Dataset	***Accuracy***	***Precision***	***Recall***	***F1-score***
Grupo Combinado 1	0,9935	0,9926	0,9936	0,9931
Grupo Combinado 2	0,9850	0,9881	0,9799	0,9840

BiLSTM también alcanza una efectividad muy alta, con *f1-scores* de 0,9931 en el Grupo Combinado 1 y 0,9840 en el Grupo Combinado 2. Su superioridad sobre LSTM unidireccional en el Grupo Combinado 1 confirma el valor del procesamiento bidireccional. Esto tiene sentido ya que los patrones sospechosos en correos de phishing pueden manifestarse tanto al inicio como al final de los mensajes, y BiLSTM puede capturar estas señales complejas.

El equilibrio entre *precision* y *recall* es excepcional (0,9926 y 0,9936 respectivamente en Grupo 1). Esto indica que BiLSTM logra detectar prácticamente todos los correos de phishing mientras mantiene una tasa mínima de falsos positivos.

Configuración óptima de hiperparámetros

Al igual que con LSTM, *Keras Tuner* identificó las configuraciones específicas que se detallan en la tabla 5.8, optimizando la efectividad para cada dataset.

Tabla 5.8: Hiperparámetros óptimos identificados para BiLSTM

Dataset	**Embed Dim**	**LSTM1 Units**	**LSTM2 Units**	**Dropout**	**Dense Units**	**Dense Dropout**	**Opt**	**LR**	**Batch Size**	**Best Epoch**
Grupo Comb. 1	64	128	32	0,3	32	0,3	Adam	0,001	128	4
Grupo Comb. 2	64	64	64	0,4	32	0,3	Adam	0,001	128	2

Proceso de entrenamiento

Las figuras 5.6 y 5.7 muestran las curvas de entrenamiento para ambos datasets. Revelan una convergencia más rápida que LSTM, gracias a la mayor capacidad representacional de la arquitectura bidireccional.

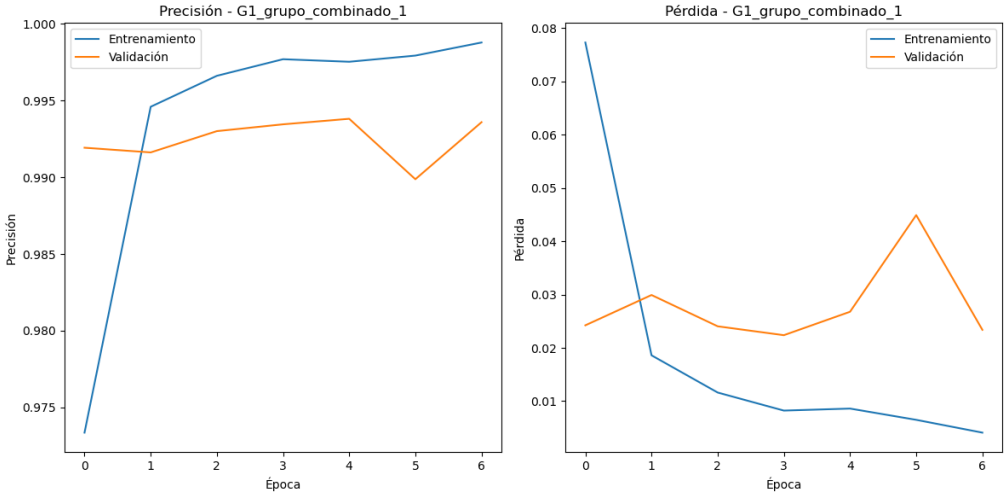

Figura 5.6: Curvas de entrenamiento y validación para BiLSTM - Grupo Combinado 1

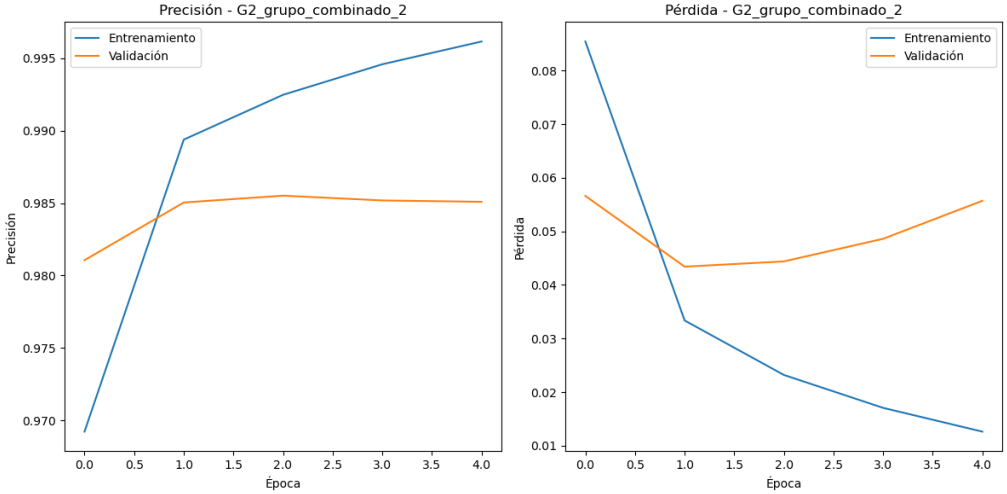

Figura 5.7: Curvas de entrenamiento y validación para BiLSTM - Grupo Combinado 2

Esta convergencia temprana refleja la eficiencia de la arquitectura bidireccional para extraer características discriminativas. Además, la configuración optimizada utiliza arquitecturas asimétricas interesantes: 128-32 unidades para Grupo 1 y 64-64 para Grupo 2, adaptándose a las diferentes complejidades de cada dataset.

5.5– Análisis comparativo de resultados

Los resultados obtenidos revelan diferencias significativas en la efectividad entre las tres familias de técnicas implementadas. Las técnicas heurísticas muestran las limitaciones

más pronunciadas, con f1-scores que oscilan entre 0,072 (Enron) y 0,904 (Nigerian Fraud), evidenciando una alta dependencia del tipo específico de phishing y la calibración de reglas. La efectividad particularmente deficiente en datasets corporativos como CEAS 08 (f1-score de 0,285) y la incapacidad para generalizar efectivamente en los grupos combinados (f1-scores de 0,405 y 0,300) confirman las limitaciones de los enfoques basados en reglas predefinidas frente a la diversidad evolutiva de las técnicas de phishing modernas.

Los algoritmos de Machine Learning demuestran una superioridad notable y consistente, alcanzando f1-scores superiores a 0,95 en prácticamente todos los datasets evaluados. Support Vector Machine emerge como el algoritmo más robusto, obteniendo la mayor efectividad en 6 de los 11 datasets individuales y logrando un f1-score de 0,9803 en el Grupo Combinado 2. La consistencia de estos resultados, con variaciones mínimas entre diferentes tipologías de correos (desde 0,9523 en TREC 06 hasta 0,9970 en Nigerian 5), evidencia la capacidad superior de estos modelos para capturar patrones complejos mediante el aprendizaje automático de características discriminativas a partir de representaciones TF-IDF optimizadas.

Las técnicas de Deep Learning alcanzan la mayor efectividad, con BERT, LSTM y BiLSTM superando consistentemente el umbral de 0,98 en f1-score para ambos grupos combinados. BiLSTM obtiene los mejores resultados globales con f1-scores de 0,9931 (Grupo Combinado 1) y 0,9840 (Grupo Combinado 2), seguido muy de cerca por LSTM (0,9909 y 0,9852) y BERT (0,9886 y 0,9849). La diferencia marginal entre estas arquitecturas sugiere que todas aprovechan efectivamente las representaciones contextuales profundas, aunque BiLSTM demuestra una ligera ventaja al procesar información bidireccional que captura patrones tanto al inicio como al final de los mensajes. Esta comparativa se muestra en la figura 5.8, donde además se puede observar que claramente la efectividad del grupo combinado 1 (con metadatos) es superior a la del grupo combinado 2 (solo asunto y cuerpo) en los modelos de deep learning.

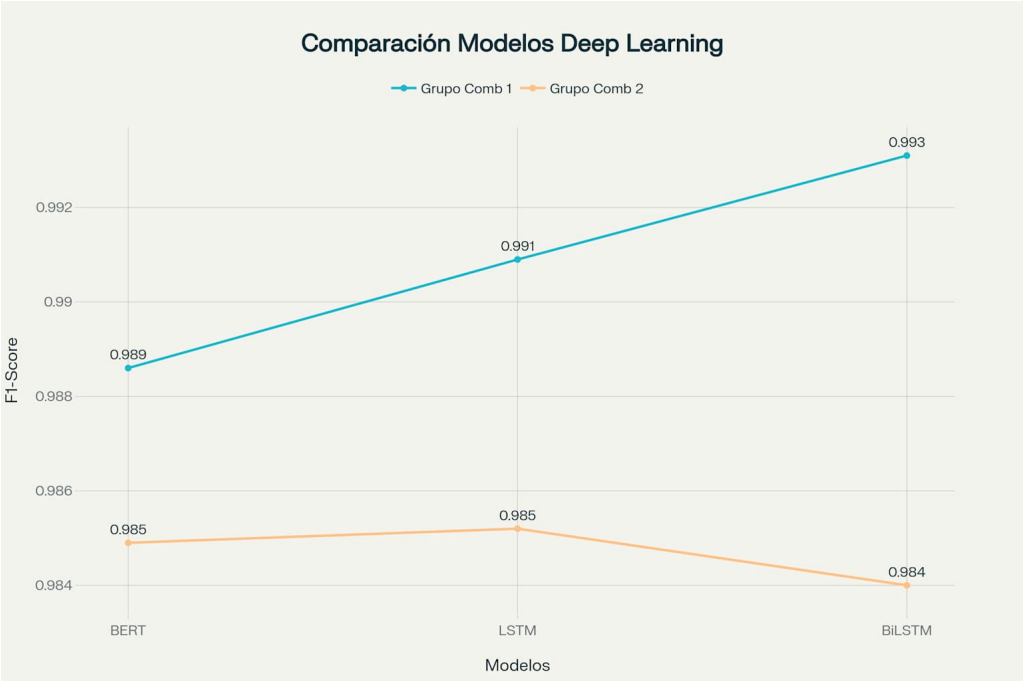

Figura 5.8: Comparación de F1-Score en modelos de Deep Learning

CAPÍTULO 6

Conclusiones y Trabajo Futuro

Este capítulo final presenta las conclusiones del estudio comparativo sobre técnicas de detección de phishing y analiza las desviaciones respecto al plan inicial del proyecto. La investigación evaluó tres enfoques principales: técnicas heurísticas basadas en reglas, algoritmos tradicionales de machine learning y modelos de deep learning, determinando su efectividad relativa y el cumplimiento de los objetivos técnicos, personales y académicos planteados.

A partir de las limitaciones experimentales identificadas y las oportunidades de mejora detectadas, el trabajo establece futuras líneas de investigación que podrían ampliar la efectividad y el alcance de estos sistemas de detección.

6.1– Planificación final, desviación y motivos

La ejecución real del proyecto experimentó desviaciones significativas respecto a la planificación inicial, superando en aproximadamente 30 horas la estimación original de 300 horas. Estas variaciones responden a la naturaleza exploratoria de la investigación y a la complejidad de los procesos experimentales implementados.

- La fase de **investigación y revisión de literatura** requirió aproximadamente 20 horas adicionales sobre las 60 inicialmente planificadas. Esta extensión se debió a que la revisión bibliográfica no se limitó únicamente a la fase inicial, sino que se mantuvo de forma continua durante todo el desarrollo del proyecto. La necesidad de consultar literatura especializada surgió iterativamente conforme se identificaban desafíos técnicos específicos en la implementación de cada familia de técnicas.
- Por el contrario, la **recopilación y preprocesamiento de datasets** se completó con aproximadamente 10 horas menos de las estimadas inicialmente. La abundante información disponible en la literatura sobre conjuntos de datos especializados

en phishing facilitó la identificación rápida de la colección de Champa y col. [35], reduciendo significativamente el tiempo de búsqueda y evaluación de alternativas.

- La **implementación de técnicas individuales** constituyó la desviación más pronunciada, requiriendo aproximadamente 50 horas adicionales sobre las 60 planificadas. Esta extensión se distribuyó principalmente en dos áreas críticas: las múltiples iteraciones experimentales de las técnicas heurísticas, que demandaron 30 horas adicionales en la búsqueda de configuraciones efectivas de reglas y umbrales sin lograr los resultados esperados, y el desarrollo de preprocesamientos especializados para cada familia de técnicas junto con la implementación de pipelines robustos con gestión de fallos, que requirió unas 20 horas adicionales.

- La fase de **evaluación y análisis de resultados** se extendió en aproximadamente 10 horas sobre la planificación inicial debido a la complejidad de los modelos de deep learning con datasets masivos. Aunque los tiempos de ejecución de hasta 6 días para ciertos modelos no se contabilizan como horas de trabajo efectivo, la planificación de experimentos, el análisis de resultados intermedios y las modificaciones iterativas de configuraciones demandaron tiempo adicional significativo.

- La fase de **desarrollo y evaluación de técnicas combinadas**, inicialmente planificada con 70 horas, fue omitida tras constatar en la literatura que los enfoques *ensemble* algunas veces no superan el rendimiento de las técnicas individuales optimizadas. Esta decisión se reforzó al verificar que ya se había excedido el tiempo asignado al proyecto, priorizando la calidad de la implementación y evaluación de las técnicas individuales.

- Finalmente, la **redacción del TFM** requirió aproximadamente 10 horas adicionales sobre las 40 planificadas inicialmente. La dimensión final del documento superó las estimaciones iniciales debido a la necesidad de documentar exhaustivamente las múltiples configuraciones experimentales, los análisis detallados de resultados por dataset y la inclusión de descripciones que garantizan la reproducibilidad de la investigación.

6.2– Conclusiones

Este trabajo ha desarrollado y evaluado comparativamente diferentes técnicas para la detección automática de phishing en correos electrónicos, implementando un enfoque completamente autónomo basado únicamente en el contenido y metadatos de los correos.

6.2.1. Hallazgos principales

La evaluación experimental revela una jerarquía de efectividad entre las técnicas implementadas. Las **técnicas heurísticas** muestran limitaciones pronunciadas con F1-scores que oscilan entre 0,072 y 0,904, evidenciando alta dependencia del tipo de phishing, y si se implementan reglas específicas para detectar el tipo de phishing concreto. Los **algoritmos de Machine Learning** tradicionales demuestran superioridad notable y consistente respecto a las heurísticas, alcanzando F1-scores superiores a 0,95 en prácticamente todos los datasets, con Support Vector Machine emergiendo como el algoritmo más robusto.

Las **técnicas de Deep Learning** alcanzan la mayor efectividad, con BiLSTM, LSTM y BERT superando consistentemente el umbral de 0,98 en F1-score, siendo BiLSTM el modelo con mayor efectividad global (F1-score de 0,9931).

6.2.2. Implicaciones para la implementación práctica

Los hallazgos tienen implicaciones directas para la implementación operacional de sistemas de detección de phishing. Los algoritmos de Machine Learning tradicional, particularmente SVM, ofrecen un equilibrio óptimo entre efectividad (F1-score de 0,9803) y eficiencia computacional, siendo especialmente adecuados para entornos con recursos computacionales limitados para el entrenamiento. Las técnicas de Deep Learning, aunque requieren mayor infraestructura, proporcionan la efectividad superior necesaria para entornos de alta criticidad donde la minimización de falsos negativos es prioritaria. La consistencia de efectividad observada entre diferentes tipos de datasets sugiere que estos modelos mantienen efectividad operacional ante la evolución natural de las técnicas de phishing, proporcionando una base sólida para sistemas de detección adaptativos y resilientes.

6.2.3. Diferenciación y aportes respecto a la literatura existente

Aunque la literatura existente sobre detección de phishing presenta trabajos con algunas de las características que se desarrollan a continuación, ningún estudio previo combina simultáneamente muchas de ellas o todas, como es el caso de este trabajo:

- **Volumen de datos**: Este trabajo utiliza 218.086 registros distribuidos en 11 datasets, lo que permite entrenar modelos más robustos y evaluar su capacidad de generalización de manera más confiable.
- **Perspectiva histórica**: La cobertura temporal de los datasets de 24 años (1998-2022) permite evaluar la evolución del phishing y la capacidad de los modelos para detectar tanto técnicas clásicas como emergentes, proporcionando una evaluación más realista de la adaptabilidad temporal de las soluciones propuestas.
- **Heterogeneidad y capacidad de generalización**: Mientras otros estudios priorizan datasets homogéneos o más simples para optimizar métricas, este trabajo busca la heterogeneidad como desafío metodológico que refleja mejor entornos operacionales reales, estableciendo un estándar más exigente para la validación de técnicas en el que se busca que estas sean capaces de generalizar.
- **Estrategia de combinación**: La agrupación de datasets según disponibilidad de metadatos (Grupo 1 vs Grupo 2) permite evaluar sistemáticamente el impacto de incluir o no esos metadatos en la efectividad de las técnicas de detección.
- **Comparación experimental controlada**: Evaluación sistemática de heurísticas, machine learning tradicional y deep learning bajo las mismas condiciones experimentales, eliminando variables confusas presentes en comparaciones entre estudios diferentes y proporcionando métricas directamente comparables.

- **Autonomía completa**: A diferencia de muchos estudios que dependen de APIs de reputación, listas negras o servicios de terceros, esta implementación es completamente autónoma, eliminando variables externas incontrolables.
- **Preprocesamiento especializado**: Aunque se ha realizado una primera fase de preprocesamiento o limpieza general para todas las técnicas, posteriormente se han llevado a cabo estrategias específicas para cada familia de técnicas, intentando maximizar la efectividad individual de cada enfoque.
- **Optimización**: Implementación de búsqueda de mejores hiperparámetros tanto en machine learning como en deep learning, garantizando configuraciones óptimas para cada técnica y documentando el proceso de optimización para facilitar la reproducibilidad.
- **Reproducibilidad**: Se añaden muchas pautas para la reproducibilidad del trabajo. La autonomía total del sistema elimina dependencias de servicios externos, permitiendo la replicación de los experimentos.
- **Efectividad excepcional**: BiLSTM alcanza 99,31 % F1-score con evidencia empírica sólida, confirmando el potencial de deep learning mediante experimentación rigurosa.

6.3– Objetivos cumplidos

El desarrollo experimental ha cumplido el **objetivo principal** establecido implementando y contrastando empíricamente múltiples enfoques para la clasificación de correos electrónicos maliciosos, determinando así las arquitecturas y configuraciones de mayor efectividad mediante una evaluación sistemática sobre conjuntos de datos heterogéneos, llegando a conseguir precisiones muy altas, especialmente con los modelos de deep learning. Además, las técnicas desarrolladas son completamente autónomas, no necesitan de conexión a Internet para funcionar ni dependen de servicios externos.

6.3.1. Subobjetivos principales:

El **primer subobjetivo** se ha completado exitosamente mediante la recopilación y procesamiento de 11 datasets heterogéneos, totalizando 212.250 registros tras el proceso de limpieza. Se desarrollaron estrategias de combinación que crearon dos grupos diferenciados según disponibilidad de metadatos, permitiendo evaluar la capacidad de generalización de los modelos.

El **segundo subobjetivo** se ha cumplido completamente mediante la implementación y optimización de tres familias de técnicas distintas. Se desarrolló un sistema heurístico basado en reglas con puntuación ponderada y umbrales adaptativos, se implementaron cuatro algoritmos de machine learning (SVM, Random Forest, Naive Bayes, Regresión Logística) con optimización de hiperparámetros mediante Grid Search, y se evaluaron tres arquitecturas de deep learning (BERT, LSTM, BiLSTM) con optimización automática mediante Optuna y Keras Tuner.

El **tercer subobjetivo** se ha alcanzado mediante una evaluación comparativa exhaustiva que utilizó las métricas estándar especificadas. Los resultados demuestran que las técnicas de deep learning obtienen la efectividad superior, seguidas por los algoritmos de machine learning, mientras que las técnicas heurísticas muestran limitaciones significativas.

6.3.2. Objetivos personales

Se han conseguido realizando una exploración en ciberseguridad e inteligencia artificial donde se han utilizado técnicas avanzadas de machine learning y deep learning aplicadas a problemas de ciberseguridad, así como la experiencia adquirida en el manejo de datasets masivos y diversos.

6.3.3. Objetivos profesionales

Se han alcanzado mediante el desarrollo de competencias especializadas en diseño, implementación y evaluación de sistemas de detección automática, además de la experiencia metodológica en comparación sistemática de técnicas que resulta transferible a múltiples contextos de clasificación y análisis de datos.

6.3.4. Objetivos académicos

Se han cumplido mediante la aplicación de conocimientos de las asignaturas del máster especificadas, desarrollando competencias de diseño experimental riguroso, evaluación sistemática y análisis objetivo de resultados que establecen bases sólidas para poder realizar futuras actividades de investigación.

6.4– Continuidad para trabajos futuros

Los resultados obtenidos en este trabajo abren múltiples líneas de investigación que pueden ampliar significativamente la efectividad y el alcance de los sistemas de detección de phishing. Las limitaciones identificadas y las oportunidades emergentes sugieren direcciones específicas para futuras investigaciones.

6.4.1. Diversificación lingüística

La incorporación de conjuntos de datos multilingües constituye una prioridad inmediata para mejorar la aplicabilidad global de las técnicas desarrolladas. Los datasets actuales se limitan al inglés, restringiendo la generalización a contextos hispanohablantes y otras

regiones lingüísticas donde el phishing presenta características culturales específicas. El desarrollo de corpus especializados en español, con patrones de ingeniería social adaptados a contextos locales, permitiría evaluar la transferibilidad de las arquitecturas optimizadas y desarrollar modelos específicamente calibrados para amenazas regionales.

6.4.2. Integración de información estructural completa

La disponibilidad limitada de metadatos en los datasets actuales representa una oportunidad significativa de mejora. Futuras investigaciones deberían incorporar cabeceras completas de correo electrónico, incluyendo información de enrutamiento, autenticación SPF/DKIM/DMARC, y análisis de archivos adjuntos. Esta expansión permitiría desarrollar sistemas híbridos que combinen análisis textual profundo con verificación técnica de autenticidad, potencialmente mejorando las tasas de detección mientras reducen falsos positivos.

6.4.3. Arquitecturas híbridas

El desarrollo de arquitecturas híbridas que integren las fortalezas complementarias de BERT para comprensión contextual, BiLSTM para análisis secuencial y SVM para separación en espacios de alta dimensionalidad representa una línea prometedora. Estas implementaciones requerirían metodologías sofisticadas de fusión de decisiones y calibración de confianza entre modelos heterogéneos, aunque no asegura que se puedan conseguir mejores resultados.

6.4.4. Detección de ataques adversarios

La investigación futura debe abordar la vulnerabilidad de los modelos de deep learning ante ataques adversarios específicamente diseñados para evadir detección automática. El desarrollo de técnicas de entrenamiento adversario y métodos de detección de manipulación textual constituye un área crítica, especialmente considerando la sofisticación creciente de los atacantes y su acceso a herramientas de inteligencia artificial generativa de última generación como GPT-4 o Claude.

6.4.5. Sistemas adaptativos y aprendizaje continuo

La naturaleza evolutiva de las técnicas de phishing demanda el desarrollo de sistemas que se adapten automáticamente a nuevas amenazas. La implementación de pipelines de aprendizaje continuo que actualicen los modelos con nuevos datos de phishing en tiempo real sin requerir reentrenamiento completo podría mantener la efectividad ante campañas emergentes.

6.4.6. Extensión a smishing y vectores de ataque emergentes

La evolución de las técnicas de phishing hacia plataformas móviles y aplicaciones de mensajería instantánea presenta una oportunidad significativa para expandir las metodologías desarrolladas. Futuras investigaciones podrían adaptar las arquitecturas de deep learning optimizadas en este trabajo para analizar mensajes de texto cortos, contenido multimedia y patrones de comunicación en plataformas como WhatsApp, Telegram y redes sociales.

6.4.7. Evaluación en entornos reales

La validación de las técnicas desarrolladas en entornos operacionales reales constituye el paso natural siguiente. Esta implementación piloto requeriría el despliegue de los modelos optimizados en servidores de correo empresariales, donde los mensajes clasificados como sospechosos por el sistema serían dirigidos automáticamente a una cuarentena especializada. En esta fase intermedia, analistas de seguridad realizarían verificaciones manuales de cada clasificación, proporcionando retroalimentación etiquetada que permitiría cuantificar métricas.

Referencias

[1] Ainhoa Carpio-Talleux. Descifrar el ataque de phishing, ¡para no picar el anzuelo!, 2025. https://www.appvizer.es/revista/it/seguridad-informatica/ataque-de-phishing.

[2] cmitsolutions. Smishing & phishing in 2025: The invisible threats draining smbs—and how cyber insurance is fighting back, 2025. https://cmitsolutions.com/piscataway-nj-1178/blog/smishing-phishing-in-2025-the-invisible-threats/.

[3] Josh Howarth. Ai cybersecurity: How ai is revolutionizing cyber defense, 2024. https://explodingtopics.com/blog/ai-cybersecurity.

[4] ISACA Journal. Lessons learned from the bangladesh bank heist. *ISACA Journal*, 6, 2023. https://www.isaca.org/resources/isaca-journal/issues/2023/volume-6/lessons-learned-from-the-bangladesh-bank-heist.

[5] Startup Defense. Catching up on cloud attack paths with cloud threat specialist sebastian walla, 2025. https://www.startupdefense.io/blog/catching-up-on-cloud-attack-paths-with-cloud-threat-specialist-sebastian-walla.

[6] Redacción Cyberlideria. Los 10 ataques cibernéticos más impactantes de 2024 y 2025: lecciones de vulnerabilidad y resiliencia digital, 2024. https://cyberlideriamgzn.es/los-10-ataques-ciberneticos-mas-impactantes-de-2024-y-2025-lecciones-de-vulnerabilidad-y-resiliencia-digital/.

[7] Keepnet Labs. Phishing attacks in 2024: The global threat landscape, 2024. https://keepnetlabs.com/blog/top-phishing-statistics-and-trends-you-must-know.

[8] Peter Chapman. Crisp-dm 1.0: Step-by-step data mining guide. 2000. https://api.semanticscholar.org/CorpusID:59777418.

[9] Universidad Francisco de Vitoria. ¿cuánto cobra un experto en ciberseguridad al mes?, 2025. https://www.ufv.es/cuanto-cobra-un-experto-en-ciberseguridad-al-mes-preguntas-grados/.

[10] Agencia Estatal de Administración Tributaria. Tabla de amortización simplificada, 2025. https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/folleto-actividades-economicas/3-impuesto-sobre-renta-personas-fisicas/3_5-estimacion-directa-simplificada/3_5_4-tabla-amortizacion-simplificada.html.

[11] Tarifa Luz Hora. Precio del kwh de luz hoy, 2025. https://tarifaluzhora.es/info/precio-kwh.

[12] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Faith Cranor, Jason Hong, and Chengshan Zhang. An empirical analysis of phishing blacklists. In *Proceedings of the Sixth Conference on Email and Anti-Spam*. Carnegie Mellon University, 2009. https://kilthub.cmu.edu/articles/journal_contribution/An_Empirical_Analysis_of_Phishing_Blacklists/6469805.

[13] Matheesha Fernando, Abdun Naser Mahmood, Mohammad Jabed Morshed Chowdhury, and Zhen He. Phishlex: A real-time machine learning model for zero-day phishing detection by systematizing url techniques. Jan 2025. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5205908.

[14] João Sousa Botto. Dmarc management. *The Cloudflare Blog*, 2023. https://blog.cloudflare.com/dmarc-management/.

[15] EasyDMARC Research Team. Easydmarc 2025 dmarc adoption report. *EasyDMARC*, 2025. https://easydmarc.com/blog/ebook/easydmarc-dmarc-adoption-report-2025.

[16] Juliette Cash Elaine Dzuba. Introducing cloudflare's 2023 phishing threats report. *Cloudflare Security Report*, 2023. https://blog.cloudflare.com/2023-phishing-report/.

[17] G. Hunt. Email sandboxing and message delivery delays. *SpamTitan Blog*, 2025. https://www.spamtitan.com/blog/email-sandboxing-and-message-delivery-delays/.

[18] Melad Mohamed Al-Daeef, Nurlida Basir, and Madihah Mohd Saudi. Evaluation of phishing email classification features: Reliability ratio measure. In *Proceedings of the World Congress on Engineering*, volume 1, pages 236–240, 2017. https://www.iaeng.org/publication/WCE2017/WCE2017_pp236-240.pdf.

[19] Varun Vyas, Aditya Nair, and Allan Lopes. Heuristic based malicious url detection. *International Journal for Research in Engineering Application & Management*, 6(1):267–271, 2020. http://ijream.org/papers/IJREAMV06I0161083.pdf.

[20] Eint Sandi Aung, Chaw Thet Zan, and Hayato Yamana. A survey of url-based phishing detection. In *DEIM Forum*, pages 1–8, 2019. https://db-event.jpn.org/deim2019/post/papers/201.pdf.

[21] Pavithra Madushanka and A Hanees. Phishing e-mail filtering mechanism using heuristic technique. 05 2021. https://www.seu.ac.lk/researchandpublications/asrs/2016/ASRS%202016-%20Conference%20Proceeding%20%20-%20Page%20261-271.pdf.

[22] Nitin Naik, Paul Jenkins, Nick Savage, et al. Embedded yara rules: strengthening yara rules utilising fuzzy hashing and fuzzy rules. *Complex Intell*, 2021. https://doi.org/10.1007/s40747-020-00233-5.

[23] Fernando Sanchez and Zhenhai Duan. A sender-centric approach to detecting phishing emails. *Florida State University Technical Report*, (TR-121106), 2012. https://www.cs.fsu.edu/files/reports/TR-121106.pdf.

[24] Trevor Wood, Vitor Basto-Fernandes, Eerke Boiten, and Iryna Yevseyeva. Systematic literature review: Anti-phishing defences and their application to before-the-click phishing email detection. *arXiv preprint arXiv:2204.13054*, 2022. https://arxiv.org/ftp/arxiv/papers/2204/2204.13054.pdf.

[25] Shafaizal Shabudin, Nor Samsiah Sani, Khairul Akram Zainal Ariffin, and Mohd Aliff. Feature selection for phishing website classification. *International Journal of Advanced Computer Science and Applications*, 11(4):587–595, 2020. https://thesai.org/Downloads/Volume11No4/Paper_77-Feature_Selection_for_Phishing_Website.pdf.

[26] Sudhir Satish Patil. Enhancing email fraud detection using svm: A comparative analysis with other machine learning models. *International Journal for Multidisciplinary Research*, 7(2), 2025. https://www.ijfmr.com/papers/2025/2/40462.pdf.

[27] Sanjeev Singh and Shashank Singh. Enhancing email security: A machine learning approach for robust phishing detection. *International Journal of Creative Research Thoughts*, 2025. https://www.ijcrt.org/papers/IJCRT2505171.pdf.

[28] Sachidanand Chaturvedi and Ravindra Gupta. A comparative analysis of naive bayes, support vector machines, and random forest for email spam detection: A supervised machine learning approach. *International Journal of Creative Research Thoughts*, 11(7), 2023. https://ijcrt.org/papers/IJCRT2307372.pdf.

[29] Jhoseph Alberto Molina Salavarría and Kevin Joel Monteros González. Detección de ataques de phishing utilizando procesamiento de lenguaje natural y modelo oculto de markov, 2022. https://repositoriobe.espe.edu.ec/server/api/core/bitstreams/0886d322-85f0-4cfd-8923-52fbccbc3415/content.

[30] Yoga Shri Murti and Palanichamy Naveen. Machine learning algorithms for phishing email detection. *Journal of Logistics, Informatics and Service Science*, 10(2):249–261, 2023. https://www.aasmr.org/liss/Vol.10/No.2%202023/Vol.10%20No.2.17.pdf.

[31] Abeer Alhuzali, Ahad Alloqmani, Manar Aljabri, and Fatemah Alharbi. In-depth analysis of phishing email detection: Evaluating the performance of machine learning and deep learning models across multiple datasets. *Applied Sciences*, 15(6), 2025. https://www.mdpi.com/2076-3417/15/6/3396.

[32] Mohamed Hassan. Evaluation of deep learning algorithms in comparison for phishing email identification. *Applied Mathematics on Science and Engineering*, 1(1):21–35, 2024. https://furthersci.org/journal-admin/uploads/articles/amse113.pdf.

[33] Brij B. Gupta, Akshat Gaurav, Varsha Arya, Razaz Waheeb Attar, Shavi Bansal, Ahmed Alhomoud, and Kwok Tai Chui. Advanced bert and cnn-based computational model for phishing detection in enterprise systems. *Computer Modeling in Engineering & Sciences*, 141(3):2166–2183, 2024. https://doi.org/10.32604/cmes.2024.056473.

[34] René Meléndez, Michal Ptaszynski, and Fumito Masui. Comparative investigation of traditional machine-learning models and transformer models for phishing email detection. *Electronics*, 13(24):4877, 2024. https://doi.org/10.3390/electronics13244877.

[35] Arifa I Champa, Md Fazle Rabbi, and Minhaz F Zibran. Curated datasets and feature analysis for phishing email detection with machine learning. In *Proceedings of the 3rd IEEE International Conference on Computing and Machine Intelligence (ICMI)*, pages 1–7. IEEE, 2024. https://doi.org/10.1109/ICMI60790.2024.10585821.

[36] Arifa I Champa, Fazle Rabbi, and Minhaz F Zibran. Why phishing emails escape detection: A closer look at the failure points. In *Proceedings of the 2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–6. IEEE, 2024. https://doi.org/10.1109/ISDFS60797.2024.10527344.

[37] Najwa Altwaijry, Isra Al-Turaiki, Reem Alotaibi, and Fatimah Alakeel. Advancing phishing email detection: A comparative study of deep learning models. *Sensors*, 24(7):2077, 2024. https://doi.org/10.3390/s24072077.

CAPÍTULO 7

ANEXOS

7.1– Anexo I: Análisis detallado del proceso de limpieza por dataset

Este anexo proporciona un análisis granular del proceso de limpieza aplicado a cada uno de los 11 datasets, detallando las transformaciones específicas, patrones identificados y resultados obtenidos en cada fase del pipeline de procesamiento.

7.1.1. Resultados de normalización de etiquetas

La fase de normalización identificó inconsistencias significativas en dos datasets principales. **TREC_05** presentó 715 registros con etiquetas en formato decimal (0.0, 1.0), representando el 1,28 % de su contenido total. **TREC_06** mostró 57 registros similares (0,35 % de su contenido). Estos patrones sugieren diferencias en las metodologías de etiquetado original entre las diferentes ediciones del corpus TREC. Los restantes datasets mantuvieron consistencia en el formato de etiquetas, evidenciando procesos de curación previos más rigurosos.

7.1.2. Análisis de duplicados por dataset

La detección de duplicados reveló patrones específicos por fuente de datos. **TREC_05** concentró el mayor número de duplicados con 676 registros (1,21 % de su contenido), distribuidos en 576 duplicados por contenido de cuerpo y 102 duplicados exactos por asunto+cuerpo. **TREC_06** presentó 43 duplicados (0,26 % de su contenido), mientras que **TREC_07** no mostró duplicados detectables.

Los datasets especializados (Nazario, Nigerian_Fraud, Nigerian_5) no presentaron duplicados, indicando procesos de curación más rigurosos en colecciones temáticas específicas.

7.1.3. Impacto del filtrado por idioma

El filtrado lingüístico mostró variaciones significativas entre datasets, reflejando sus contextos de recopilación originales. **TREC_05** experimentó la mayor reducción con 3.714 registros eliminados (6,6 % de su contenido), seguido por **Enron** con 594 eliminaciones (2,0 % de su contenido).

Los datasets académicos como **Ling** mostraron reducciones mínimas (52 registros, 1,8 %), consistente con su origen en comunicaciones académicas predominantemente en inglés. Los datasets especializados en fraude nigeriano mantuvieron altas tasas de conservación (98,8 % para Nigerian_Fraud), sugiriendo que estos ataques se dirigían principalmente a audiencias anglófonas.

7.1.4. Calidad de metadatos por dataset

El análisis de campos faltantes reveló patrones específicos relacionados con las metodologías de recopilación originales. Los datasets **Nigerian_5** y **Nigerian_Fraud** presentaron el mayor número de campos faltantes en metadatos (1.336 y 1.306 respectivamente para el campo 'receiver').

Los datasets corporativos como **Enron** mantuvieron alta completitud en metadatos estructurales, con únicamente 194 campos 'subject' faltantes de 29.173 registros (0,66 %). Esta completitud refleja la naturaleza formal de las comunicaciones corporativas donde los metadatos son esenciales para el funcionamiento organizacional.

7.1.5. Distribuciones finales por dataset

Las distribuciones de clases finales mantuvieron las características temáticas de cada dataset. La estabilidad de estas distribuciones tras el proceso de limpieza confirma que las transformaciones aplicadas no introdujeron sesgos sistemáticos hacia ninguna clase específica, preservando la representatividad original de cada conjunto de datos. A continuación, se muestran todos estos datos en forma de tablas para cada uno de los datasets:

CEAS_08.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	39154	0	0	0	1: 55.78%, 0: 44.22%
preprocesamiento_general	39154	0	0	0	1: 55.78%, 0: 44.22%
solo_ingles	39032	0	0	0	1: 55.65%, 0: 44.35%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	462	462	462
body	0	0	0
subject	28	28	28
label	0	0	0
subject+body	0	0	0
date	0	0	0
urls	0	0	0
sender	0	0	0

Figura 7.1: Limpieza General de CEAS_08

Enron.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	29767	0	0	0	0: 53.05%, 1: 46.95%
preprocesamiento_general	29767	0	0	0	0: 53.05%, 1: 46.95%
solo_ingles	29173	0	0	0	0: 53.53%, 1: 46.47%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
subject+body	0	0	0
label	0	0	0
body	0	0	0
subject	198	198	194

Figura 7.2: Limpieza General de Enron

Ling.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	2859	0	0	0	0: 83.98%, 1: 16.02%
preprocesamiento_general	2859	0	0	0	0: 83.98%, 1: 16.02%
solo_ingles	2807	0	0	0	0: 83.68%, 1: 16.32%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
subject+body	0	0	0
label	0	0	0
body	0	0	0
subject	62	62	60

Figura 7.3: Limpieza General de Ling

Nazario.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	1565	0	0	0	1: 100.0%
preprocesamiento_general	1565	0	0	0	1: 100.0%
solo_ingles	1499	0	0	0	1: 100.0%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	96	96	90
body	0	0	0
subject	4	4	3
label	0	0	0
subject+body	0	0	0
date	1	1	0
urls	0	0	0
sender	0	0	0

Figura 7.4: Limpieza General de Nazario

Nazario_5.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	3065	0	0	0	1: 51.06%, 0: 48.94%
preprocesamiento_general	3065	0	0	0	1: 51.06%, 0: 48.94%
solo_ingles	2999	0	0	0	0: 50.02%, 1: 49.98%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	113	113	107
body	0	0	0
subject	50	50	49
label	0	0	0
subject+body	0	0	0
date	3	3	2
urls	0	0	0
sender	2	2	2

Figura 7.5: Limpieza General de Nazario_5

Nigerian_Fraud.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	3332	0	0	0	1: 100.0%
preprocesamiento_general	3332	0	0	0	1: 100.0%
solo_ingles	3292	0	0	0	1: 100.0%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	1324	1324	1306
body	0	0	0
subject	39	39	39
label	0	0	0
subject+body	0	0	0
date	482	482	474
urls	0	0	0
sender	331	331	323

Figura 7.6: Limpieza General de Nigerian_Fraud

Nigerian_5.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	6331	0	0	0	1: 52.63%, 0: 47.37%
preprocesamiento_general	6331	0	0	0	1: 52.63%, 0: 47.37%
solo_ingles	6291	0	0	0	1: 52.33%, 0: 47.67%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	1354	1354	1336
body	0	0	0
subject	106	106	106
label	0	0	0
subject+body	0	0	0
date	490	490	482
urls	0	0	0
sender	338	338	330

Figura 7.7: Limpieza General de Nigerian_5

SpamAssasin.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	5809	0	0	0	0: 70.43%, 1: 29.57%
preprocesamiento_general	5809	0	0	0	0: 70.43%, 1: 29.57%
solo_ingles	5738	0	0	0	0: 71.12%, 1: 28.88%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	210	210	210
body	1	1	0
subject	16	16	16
label	0	0	0
subject+body	0	0	0
date	0	0	0
urls	0	0	0
sender	0	0	0

Figura 7.8: Limpieza General de SpamAssasin

TREC_05.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	55990	102	576	598	0.0: 58.49%, 1.0: 41.51%
preprocesamiento_general	55275	0	0	0	0: 58.49%, 1: 41.51%
solo_ingles	51561	0	0	0	0: 62.59%, 1: 37.41%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	2218	1780	1601
body	577	1	0
subject	1896	1319	1310
label	715	0	0
subject+body	576	0	0
date	1795	1331	769
urls	715	0	0
sender	16	16	16

Figura 7.9: Limpieza General de TREC_05

TREC_06.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	16457	3	40	43	0.0: 75.68%, 1.0: 24.32%
preprocesamiento_general	16400	0	0	0	0: 75.68%, 1: 24.32%
solo_ingles	16131	0	0	0	0: 76.91%, 1: 23.09%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	537	512	497
body	42	1	0
subject	377	336	335
label	57	0	0
subject+body	41	0	0
date	494	469	467
urls	57	0	0
sender	268	268	266

Figura 7.10: Limpieza General de TREC_06

TREC_07.csv

Métricas Principales

Versión	Registros	Duplicados Totales	Duplicados (Subject+Body)	Duplicados (Body)	Distribución Labels
iniciales	53757	0	0	0	1: 54.69%, 0: 45.31%
preprocesamiento_general	53757	0	0	0	1: 54.69%, 0: 45.31%
solo_ingles	53727	0	0	0	1: 54.68%, 0: 45.32%

Evolución de Campos Faltantes

Campo	iniciales	preprocesamiento_general	solo_ingles
texto_completo	0	0	0
receiver	325	325	324
body	0	0	0
subject	449	449	449
label	0	0	0
subject+body	0	0	0
date	23	23	23
urls	0	0	0
sender	0	0	0

Figura 7.11: Limpieza General de TREC_07

7.2– Anexo II: Resultados detallados de Machine Learning por dataset

Este anexo presenta los resultados exhaustivos de la evaluación de los cuatro algoritmos de Machine Learning implementados sobre cada dataset individual. Las tablas incluyen las métricas completas, los hiperparámetros óptimos identificados mediante Grid Search, y los tiempos de entrenamiento registrados para cada configuración.

Los resultados se organizan por dataset, comparando la efectividad de Regresión Logística (LR), Random Forest (RF), Support Vector Machine (SVM) y Naive Bayes (NB). Los hiperparámetros presentados corresponden a la configuración que maximizó el F1-score durante la validación cruzada.

7.2.1. CEAS_08

Tabla 7.1: Resultados detallados de Machine Learning - CEAS_08

Algoritmo	*Accuracy*	*Precision*	*Recall*	*F1-score*	Tiempo (s)	Hiperparámetros óptimos
LR	0,9944	0,9954	0,9945	0,9949	7,33	C=10, max_iter=1000, solver=liblinear
RF	0,9932	0,9972	0,9906	0,9939	85,71	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9954	0,9975	0,9942	0,9959	383,85	C=1, gamma=scale, kernel=rbf
NB	0,9539	0,9985	0,9185	0,9568	0,28	alpha=1.0

7.2.2. Enron

Tabla 7.2: Resultados detallados de Machine Learning - Enron

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9870	0,9849	0,9871	0,9860	3,52	C=10, max_iter=1000, solver=liblinear
RF	0,9810	0,9790	0,9801	0,9795	61,64	class_weight=balanced, max_depth=None, n_estimators=100
SVM	0,9887	0,9846	0,9912	0,9879	303,78	C=1, gamma=scale, kernel=rbf
NB	0,9794	0,9726	0,9834	0,9780	0,24	alpha=0.1

7.2.3. Ling

Tabla 7.3: Resultados detallados de Machine Learning - Ling

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9858	1,0000	0,9130	0,9545	1,70	C=10, max_iter=1000, solver=liblinear
RF	0,9947	0,9785	0,9891	0,9838	3,03	class_weight=balanced, max_depth=10, n_estimators=200
SVM	0,9929	0,9889	0,9674	0,9780	3,89	C=1, gamma=scale, kernel=linear
NB	0,9964	0,9891	0,9891	0,9891	0,17	alpha=0.1

7.2.4. SpamAssasin

Tabla 7.4: Resultados detallados de Machine Learning - SpamAssasin

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9826	0,9785	0,9608	0,9696	0,97	C=10, max_iter=1000, solver=saga
RF	0,9782	0,9666	0,9578	0,9622	9,00	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9800	0,9696	0,9608	0,9652	12,33	C=1, gamma=scale, kernel=linear
NB	0,9756	0,9606	0,9548	0,9577	0,17	alpha=0.1

7.2.5. Nazario_5

Tabla 7.5: Resultados detallados de Machine Learning - Nazario_5

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9917	0,9966	0,9867	0,9916	3,13	C=10, max_iter=1000, solver=liblinear
RF	0,9867	0,9771	0,9967	0,9868	3,97	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9950	0,9967	0,9933	0,9950	3,26	C=1, gamma=scale, kernel=linear
NB	0,9900	0,9900	0,9900	0,9900	0,16	alpha=1.0

7.2.6. Nigerian_5

Tabla 7.6: Resultados detallados de Machine Learning - Nigerian_5

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9968	1,0000	0,9939	0,9970	2,31	C=10, max_iter=1000, solver=saga
RF	0,9944	1,0000	0,9894	0,9947	8,39	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9952	1,0000	0,9909	0,9954	13,93	C=1, gamma=scale, kernel=linear
NB	0,9936	1,0000	0,9879	0,9939	0,17	alpha=0.1

7.2.7. TREC_05

Tabla 7.7: Resultados detallados de Machine Learning - TREC_05

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9755	0,9606	0,9743	0,9674	5,74	C=10, max_iter=1000, solver=liblinear
RF	0,9726	0,9651	0,9614	0,9633	233,14	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9808	0,9729	0,9759	0,9744	1069,56	C=1, gamma=scale, kernel=rbf
NB	0,9535	0,9526	0,9215	0,9368	2,34	alpha=0.1

7.2.8. TREC_06

Tabla 7.8: Resultados detallados de Machine Learning - TREC_06

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9780	0,9542	0,9503	0,9523	2,10	C=10, max_iter=1000, solver=liblinear
RF	0,9693	0,9628	0,9020	0,9314	31,74	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9768	0,9666	0,9315	0,9487	58,37	C=1, gamma=scale, kernel=rbf
NB	0,9588	0,9595	0,8577	0,9057	0,20	alpha=0.1

7.2.9. TREC_07

Tabla 7.9: Resultados detallados de Machine Learning - TREC_07

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9885	0,9875	0,9915	0,9895	6,77	C=10, max_iter=1000, solver=liblinear
RF	0,9901	0,9910	0,9910	0,9910	163,70	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9922	0,9892	0,9966	0,9929	979,75	C=1, gamma=scale, kernel=rbf
NB	0,9625	0,9879	0,9430	0,9649	2,38	alpha=0.1

7.2.10. Grupo Combinado 2

Tabla 7.10: Resultados detallados de Machine Learning - Grupo Combinado 2

Algoritmo	***Accuracy***	***Precision***	***Recall***	***F1-score***	**Tiempo (s)**	**Hiperparámetros óptimos**
LR	0,9702	0,9674	0,9690	0,9682	31,97	C=10, max_iter=1000, solver=liblinear
RF	0,9736	0,9812	0,9620	0,9715	2636,51	class_weight=balanced, max_depth=None, n_estimators=200
SVM	0,9816	0,9817	0,9790	0,9803	14943,76	C=1, gamma=scale, kernel=rbf
NB	0,9375	0,9535	0,9108	0,9317	3,25	alpha=0.1