Detector de phishing en correos electrónicos

1st Sandro Carrillo
Escuela de Ciencia de la Computación
Universidad Nacional de Ingeniería
Lima, Perú
sandro.carrillo.j@uni.pe

2nd Ariana Lopez
Escuela de Ciencia de la Computación
Universidad Nacional de Ingeniería
Lima, Perú
ariana.lopez.j@uni.pe

3rd Albert Argumedo
Escuela de Ciencia de la Computación
Universidad Nacional de Ingeniería
Lima, Perú
albert.argumedo.r@uni.pe

4th Juan de Dios Lerzundi
Escuela de Ciencia de la Computación
Universidad Nacional de Ingeniería
Lima, Perú
juan.lerzundi.r@uni.pe

***Resumen*—El phishing es una de las amenazas prevalentes en ciberseguridad. En este proyecto se implementó un sistema de detección automática basado en técnicas de aprendizaje supervisado y procesamiento de lenguaje natural.**

El pipeline consta de preprocesamiento de texto, vectorización mediante TF-IDF y la evaluación de dos clasificacodres principales: Logistic Regressión (LR) y Random Forest.

Además, se usó Multinomial Naive Bayes como modelo baseline.

***Palabras clave*—phishing detection,email classification, TF-IDF, Logistic Regression, Random Forest, Naive Bayes, NLP, machine learning.**

I. Introducción

A. Contexto y Motivación

El correo electrónico, por su naturaleza masiva, es usado en ámbitos personales, académicos y empresariales; sin embargo, es un medio ideal para ataques maliciosos que buscan engañar a las personas para obtener información sensible. Este tipo de ataque se demonina *phishing*, que se identifica mayormente por el uso de links maliciosos a sitios web que se encargan de obtener información como el número de tarjeta de crédito o credenciales de usuario [1].

B. Importancia del Problema

El correo electrónico, debido a su bajo coste de distribución, y a su adopción masiva, es uno de los principales vectores de ingeniería social. Esto implica que el riesgo de ser víctima de phishing va más allá del aspecto técnico y se centra en la capacidad del usuario para reaccionar ante estos ataques. En el pasado se emplearon filtros basados en listas negras, pero estos ya no resultan suficientes en el escenario actual, en el que los correos imitan con un mayor grado de exactitud el lenguaje natural como los dominios desde los cuales se envían.

En este contexto, el uso de técnicas de Inteligencia Artificial y de procesamiento de lenguaje natural ofrece una solución adaptable y escalable, que permite analizar el contenido y los metadatos del correo para identificar patrones complejos en el contenido del mensaje [2].

C. Objetivos del Proyecto

1) Objetivo General:

- Implementar un sistema de clasificación automática para identificar el phishing en correos electrónicos mediante técnicas de aprendizaje supervisado y una representación textual basada en TF-IDF.

2) Objetivos Específicos:

- Construir un pipeline reproducible que abarque la limpieza, normalización y vectorización del texto contenido en correos electrónicos etiquetados. [Obj. aplicación]
- Implementar un modelo baseline utilizando Multinomial Naive Bayes para establecer un punto de referencia en términos de precisión, recall y F1-score. [Obj. técnico]
- Entrenar y evaluar modelos avanzados, específicamente Logistic Regression y Random Forest. [Obj. técnico]
- Compara el desempeño de los modelos considerando métricas relevantes para el problema de phishing, priorizando la red de falsos negativos medianteel análisis de recall. [Obj. evaluación]
- Analizar las características lingüísticas más relevantes para la clasificación mediante coeficientes del modelo y la importancia de atributos. [Obj. ético]
- Identificar oportunidades de mejora y establecer posibles líneas futuras de investigación, como el uso de embeddings contextualizados o modeos basados en Transformers. [Obj. ético]

D. Restricciones

- El proyecto se centra en la detección de phishing general (no spear-phishing dirigido).

- El proyecto se limita a correos electrónicos ya etiquetados como phishing o legítimos; no se consideran metadatos como cabeceras completas o características asociadas a URLs externas. Usa el dataset en [3].
- El archivo se encuentra preprocesao inicialmente.
- La partición de conjuntos de entrenamiento, validación y prueba se realizaa partir del archivo `phishing_email.csv`, por lo que la representatividad depende de la distribución original del dataset.
- Se restringe el alcance a clasificadores supervisados clásicos como Naive Bayes, Logistic Regression y Random Forest.
- El entrenamiento se realiza en un entorno limitado como Google Colab, lo cual condiciona la elección de hiperparámetros y el tamaño de los modelos.

II. Marco Teórico

A. Definición y naturaleza del phishing

El phishing es una técnica de ingeniería social que emplea correos y otros canales simulando entidades legítimas para inducir a la víctima a revelar credenciales, ejecutar acciones o entregar información sensible. Sus variantes incluyen spear-phishing (ataques dirigidos), whaling (dirigido a ejecutivos) y ataques basados en URL o malware ([1] y [4]).

B. Representación del texto y fundamentos de PLN

- **Tokenización y normalización:** separación en tokens, conversión a minúsculas, normalización Unicode y tratamiento de caracteres especiales (URLs, correos, números).
- **Stopwords, lematización, stemming:** reducción de forma de palabras para disminuir sparsity.
- **Bag of Words (BoW) , n-grams:** representación básica que captura presencia/frecuencia de términos y frases cortas; útil con Naive Bayes y modelos lineales.
- **TF-IDF:** ponderación por frecuencia inversa a la frecuencia en corpus, reduce peso de términos comunes.
- **Embeddings y representaciones densas:** word2vec y GloVe representan palabras en espacios vectoriales densos; capturan similitud semántica y mejoran cuando se requiere generalización semántica ([5], [6]).
- **Representaciones contextualizadas:** Transformers (BERT, RoBERTa) generan embeddings contextuales por token o frase y suelen mejorar el rendimiento en tareas de clasificación de texto si hay datos suficientes ([7], [8]).

C. Modelos de clasificación supervisada

- **Naive Bayes (Multinomial, Bernoulli):** modelo probabilístico asumiendo independencia condicional de características; rápido y efectivo en text mining con Bag of Words y TF-IDF ([9]).
- **Árboles y ensamblados (Random Forest):** capturan interacciones no lineales entre features; Random Forest es robusto a ruido y poca sintonía de hiperparámetros.
- **SVM:** buena separación en espacios de alta dimensión, efectivo con kernels y cuando hay margen claro entre clases.
- **Redes neuronales y Transformers:** CNNs, RNNs (antes dominantes en PLN) y hoy Transformers. Requieren más datos y cómputo, pero capturan mejor semántica y contexto.
- **Evaluación y selección de modelos:** emplear cross validation, curvas ROC-PR, y reportar precision,recall y F1 por clase (especialmente recall para detectar phishing).

D. Tipos de features relevantes en detección de phishing

- **Contenido textual:** términos sospechosos como ***"password"***, ***"verify"***, patrones de lenguaje urgente, uso de imperativos, errores ortográficos.
- **Características estructurales:** presencia de URLs, número de enlaces, longitud del correo, HTML vs texto plano, uso de imágenes.
- **Características del remitente o metadatos:** dominio del remitente, coincidencia entre From y Return Path, IP de origen, registros SPF, DKIM y DMARC.
- **Características de la URL:** longitud, uso de subdominios, caracteres especiales, presencia de IP en lugar de dominio, similitud visual con dominios legítimos (homoglyphs).
- **Comportamiento y contexto:** tasa de envíos por remitente, patrones temporales, reputación del dominio.

E. Problemas técnicos centrales

- **Desbalance de clases:** los correos legítimos suelen superar en número a los phishing; requiere técnicas de muestreo (oversampling,SMOTE), o métricas robustas (PR-AUC).
- **Drift y evolución de ataques:** campañas nuevas cambian vocabulario y tácticas; necesario reentrenamiento y monitorización.
- **Adversarialidad:** atacantes pueden adaptar textos para evadir detectores; conviene estudiar adversarial ML y robustez ([10]).

III. Estado del Arte

A. Enfoques históricos y baselines

- **Reglas heurísticas y blacklists:** sistemas tempranos usaban listas negras de dominios, URLs y reglas de coincidencia de patrones; funcionan contra técnicas conocidas pero no contra evasiones.
- **Filtrado bayesiano y ML clásico:** Multinomial Naive Bayes y SVMs se convirtieron en baseline por su eficiencia y desempeño en datasets textuales ([1], [2]).

B. Métodos basados en características (features)

- Investigaciones tempranas mostraron que combinar features de contenido, URL y metadatos mejora la detección. [4] y [11] analizaron heurísticas de URL y contenido.
- Modelos de ensamblado (Random Forest, Gradient Boosting) demostraron robustez frente a ruido y capacidad para priorizar features relevantes (importancia de variables).

C. Deep learning y representaciones modernas

- **CNN, RNN:** capturan patrones locales y secuencias, clasifican correos.
- **Transformers (BERT y variantes):** han mostrado mejoras marcadas en clasificación de texto y tareas de seguridad cuando se dispone de datos de calidad o se hace fine tuning ([8]).
- **Enfoques híbridos:** combinar embeddings BERT con features manuales (URL, encabezados) suele ser altamente efectivo, aprovecha semántica de texto y reglas estructurales.

D. Datasets y benchmarks comunes

- **Enron Email Dataset:** corpus grande de correos empresariales (usado para spam o filtrado [12]).
- **SpamAssassin public corpus:** colección etiquetada de spam y ham [13].
- **Phishing corpora y repositorios:** conjuntos públicos con muestras de phishing ([14] y [15]).
- Investigaciones recientes también construyen datasets a partir de correos reales anotados y de campañas de phishing actuales; calibrar modelos en datos recientes es crucial.

E. Evaluación práctica y métricas

- **Recall prioritario:** en detección de phishing, disminuir falsos negativos suele ser más importante que minimizar falsos positivos.
- **PR-AUC vs ROC-AUC:** cuando la clase positiva es rara, PR-AUC refleja mejor la capacidad del clasificador para identificar positivos relevantes.
- **Explicabilidad y análisis de errores:** LIME, SHAP son herramientas para entender decisiones y depurar falsos positivos o negativos.

F. Resumen de hallazgos empíricos

- Los modelos que combinan features manuales (URL, encabezados) con representaciones semánticas (embeddings o Transformers) tienden a obtener los mejores resultados.
- Random Forest y Gradient Boosting son fuertes competidores cuando los recursos son limitados; Transformers dominan cuando hay datos y cómputo suficientes.
- La robustez a nuevas campañas requiere pipelines de re-entrenamiento y detección de deriva.

IV. Metodología

El sistema consta de los siguientes módulos:

A. Vision General del pipeline

El pipeline implementado tiene una serie de pasos reproducibles y modular, consta de las siguientes etapas:

1) Adquisición y estandarización del dataset.
2) Limpieza y normalización del texto.
3) División del dataset en subconjuntos (train, validation y test) o uso de particiones ya provistas.
4) Extracción de características textuales mediante TF-IDF (n-grams).
5) Entrenamiento de modelos supervisados: Multinomial Naive Bayes (baseline), Logistic Regression (modelo lineal) y Random Forest (modelo de ensamblado).
6) Evaluación detallada con métricas (precision, recall, F1-score, ROC-AUC, PR-AUC), análisis de la matriz de confusión y curvas ROC y PR.
7) Interpretabilidad: análisis de coeficientes (LR) e importancia de features (RF).
8) Persistencia de artefactos (vectorizador, modelos) y documentación de metadatos para reproducibilidad.

B. Dataset y particionado

1) Origen y formato: El conjunto de datos principal es el archivo CSV preprocesado (en el notebook: `data/processed/all_emails.csv` y los splits `data/processed/train.csv`, `val.csv`, `test.csv`). Cada fila contiene al menos las columnas: text (contenido del correo) y label (0 = legítimo, 1 = phishing). Adicionalmente se mantiene una columna source para identificar la procedencia ([3]).

2) Estandarización: Se ejecuta una función `build_standard_dataframe(df, source_name)` para homogeneizar nombres de columna, rellenar valores faltantes y asegurar el esquema text, label, source. Se eliminan entradas con text vacío y duplicados exactos (`subset=['text','label']`).

3) Particionado: El notebook utiliza un conjunto de splits cargados mediante `preprocessing.load_splits()`. Si los splits no están provistos, la práctica estándar es dividir `all_emails.csv` en `train/validation/test` con estratificación por etiqueta (stratify=y) para preservar la proporción de clases.

C. Preprocesamiento del texto

Usaos para reducir el ruido y mejorar la calidad de las representaciones.

Se realiazonr las siguientes operaciones:

- Eliminación de HTML y etiquetas: stripping de etiquetas <...> y contenido HTML residual.
- Normalización de espacios y caracteres: normalización Unicode, conversión a minúsculas, eliminación de saltos de línea y múltiples espacios.

- Sustitución de tokens estructurales: reemplazo de patrones por tokens especiales para evitar vocabulario explosivo:
  - URLs → `<URL>`
  - direcciones de correo → `<EMAIL>`
  - números → `<NUM>`
- Eliminación de caracteres no imprimibles y puntuación irrelevante (según criterios dependientes del idioma).
- Tokenización (implícita en el `TfidfVectorizer` de ***scikit-learn***) y stopword removal.
- Lematización y stemming : el notebook centraliza el preprocesamiento vía `preprocessing.py`.

D. Extracción de características (vectorización)

Se emplea

```
sklearn.feature_extraction.text.TfidfVectorizer
```

para convertir textos a vectores numéricos esparcidos o sparse matrices. Los parámetros y decisiones principales son:

- `ngram_range=(1,2)`: unigramas y bigramas para capturar palabras sueltas y frases cortas como "verify account".
- `max_features=20000` : limitar el vocabulario para controlar dimensionalidad y ruido; ajustar según tamaño del corpus.
- `sublinear_tf=True`: usar escala sublineal en la frecuencia $(1 + log(tf))$.
- `use_idf=True`, `smooth_idf=True`: comportamiento por defecto que favorece raras pero discriminativas palabras.
- `stop_words='english'` o lista personalizada, según idioma del dataset.
- `norm='l2'`: normalización de vectores.

La salida después de la extracción de características es:

- **Matrices dispersas**: `X_train`, `X_val` y `X_test`
- **vectores** y_⋆

Que se guardan y se reusan para entrenamiento y despliegue. El vectorizados persiste en la línea:

```
joblib.dump('models/tfidf_vectorizer.pkl')
```

E. Modelos y configuración

Se implementan tres modelos con los hiperparámetros usados en el notebook:

1) Baseline Multinomial Naive Bayes:

- **Clase**: `sklearn.naive_bayes.MultinomialNB`
- **Hiperparámetro principal**: alpha=1.0 (Laplace smoothing) como valor por defecto.
- **Ventajas**: eficiente, rápido, buen baseline en tasks de texto.
- **Uso en notebook**: `baseline.py` contiene la función `train_baseline()` que entrena y guarda el modelo.

2) Logistic Regression (modelo lineal):

- **Clase**: `sklearn.linear_model.LogisticRegression`
- **Parámetros usados**: `max_iter=1000`, `random_state=42`.
- **Interpretabilidad**: coeficientes asociados a features posibilitan identificar tokens que aumentan o disminuyen la probabilidad de phishing.
- **Notas**: `max_iter` elevado para asegurar convergencia con vectores de alta dimensión.

3) Random Forest:

- **Clase**: `sklearn.ensemble.RandomForestClassifier`
- **Parámetros usados**: `n_estimators=100`, `random_state=42`, `n_jobs=-1`.
- **Salida de interés**: `feature_importances_` para análisis de importancia de tokens.
- **Ventajas**: captura interacciones no lineales entre features, robusto ante ruido.

F. Procedimiento de entrenamiento y validación

1) Entrenamiento reproducible:

1) Cargar `X_train`, `y_train` y el vectorizador ya ajustado.
2) Ajustar el modelo con `model.fit(X_train, y_train)`.
3) Evaluar en `X_val` y finalmente en `X_test` (hold-out) para estimación final.

2) Validación y tuning:

- Para selección de hiperparámetros se recomienda **GridSearchCV** o **RandomizedSearchCV** sobre la partición de validación o mediante cross validation estratificada.
- Rango de búsqueda sugerido:
  - Logistic Regression: `C = [0.01, 0.1, 1, 10]`, `penalty = ['l2']`.
  - Random Forest: `n_estimators = [100, 200, 500]`, `max_depth = [None, 10, 50, 100]`, `min_samples_split = [2, 5, 10]`.
  - Naive Bayes: `alpha = [0.1, 0.5, 1.0]`.

G. Descripción del Dataset

Para el desarrollo y evaluación de los modelos propuestos, se utilizó el "Phishing Email Dataset" obtenido del repositorio de Kaggle ([3]). Este conjunto de datos es una compilación robusta que integra múltiples fuentes de referencia en el ámbito de la ciberseguridad, incluyendo el corpus de Enron para correos corporativos legítimos, el dataset de SpamAssassin y colecciones de correos fraudulentos como el Nigerian Fraud y Nazario.

El dataset final consolidado consta de un total de 82,486 registros únicos. Una ventaja significativa de este recurso es su balance de clases, reduciendo el sesgo natural que suele existir en la detección de anomalías: contiene 42,891 correos etiquetados como phishing (aproximadamente el 52%) y 39,595 correos legítimos (48%). Los datos se presentan en dos columnas principales: `text_combined`,

que contiene la concatenación del asunto y el cuerpo del mensaje sin metadatos de cabecera complejos, y la etiqueta binaria `label` (0 para legítimo, 1 para phishing).

El autor solicitó hacer mención al artículo *Novel Interpretable and Robust Web-based AI Platform for Phishing Email Detection* ([3]).

H. Modelos

- **Multinomial Naive Bayes:** `sklearn.naive_bayes.MultinomialNB(alpha=1.0)`
- **Random Forest:** `sklearn.ensemble.RandomForestCl (n_estimators=200, max_depth=None, n_jobs=-1` usado para capturar interacciones entre tokens y es robusto a features ruidosos.
- **Validación:** 5-fold cross-validation y evaluación sobre test holdout (80/20). Se reportan métricas promedio y desviación estándar.

I. Métricas de evaluación

- **Precision, Recall, F1-score:** para la clase positiva (phishing).
- **ROC-AUC y PR-AUC:** dado que la clase positiva puede ser minoritaria, PR-AUC es útil para valorar el rendimiento en detección.
- **Matriz de confusión:** análisis de falsos positivos y falsos negativos (importante por coste distinto de ambos errores).

V. Resultados

Para evaluar los resultados de los modelos entrenados usamos las siguientes métricas estándar para clasificación binaria:

- **Accuracy:** Indica el porcentaje de emails correctamente clasificados
- **Precision:** Indica de los marcados como phishing, cuántos realmente lo son
- **Recall:** Indica de todos los phishing reales, cuántos detectamos
- **F1-Score:** Cálcula la media armónica entre Precision y Recall
- **Matriz de Confusión:** Detalla los errores por tipo

A. Tabla comparativa de métricas

TABLE I
Comparación de métricas entre Logistic Regression y Random Forest

Métrica	Logistic Regression	Random Forest
Accuracy	0.9830	0.9780
Precision	0.9823	0.9782
Recall	0.9852	0.9751
F1-Score	0.9850	0.9766

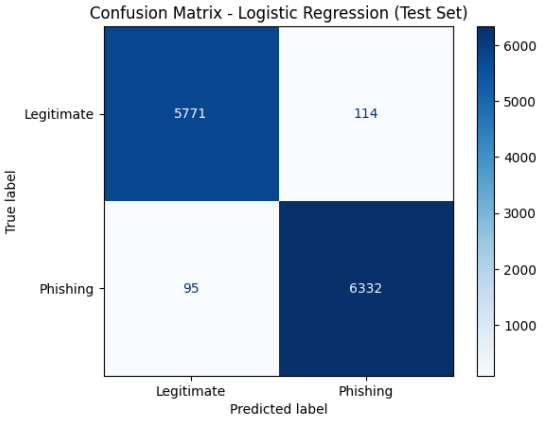

Fig. 1. Matriz de confusión del modelo Logistic Regression

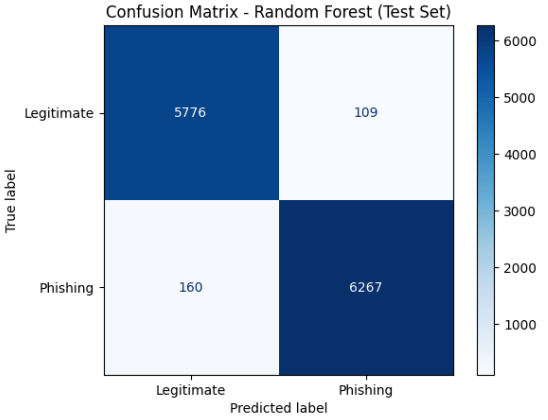

Fig. 2. Matriz de confusión del modelo Random Forest

B. Matrices de confusión

VI. Discusión

A. Consideraciones Éticas

- El análisis de correos electrónicos implica acceso a información potencialmente sensible (mensajes privados, datos personales). Es obligatorio aplicar principios de minimización de datos, anonimización/pseudonimización y políticas de retención.
- La sobre-representación de ciertos idiomas puede traducirse en mayor tasa de falsos positivos para grupos específicos, siendo idiomas minoritarios, esto añade un sesgo en los datos de entrenamiento.
- Para tratar el sesgo en los datos se recomendaría evaluar métricas por subgrupos, usar muestreo estratificado y técnicas de debiasing.
- La aparición de falsos positivos puede bloquear comunicaciones legítimas y afectar operaciones como pérdida de informacióno demora. Ante esto se deberían aplicar políticas de cuarentena, notificaciones o vías de apelación, umbrales ajustables según el criterio del usuario u organización.

VII. Conclusiones

A. Conclusiones Principales

1) Logistic regression resulta ser más adecuado para resolver el problema de clasificación del proyecto: phishing vs no phishing
2) El uso de stop words ayudó a evitar sesgos, sin embargo, se debió tener en consideración más palabras para esta lista
3) Los predictores de phishing con mayor peso son: "account", "money", "click", "replica"
4) Los predictores de ligitimidad con mayor peso son: "thanks", "opensuse", "university"

B. Trabajo a Futuro

1) Añadir análisis de URLs embebidas
2) Hacer entrenamiento y validación con datasets más modernos
3) Implementar el modelo en aplicaciones o plugins de detección para sistemas de correo electrónico

References

[1] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proceedings of the 16th International Conference on World Wide Web (WWW '07)*, Association for Computing Machinery, New York, NY, USA, 2007, pp. 649–656. doi: 10.1145/1242572.1242660.

[2] S. Abu-Nimeh, D. Nappa, X. Wang, and S. Nair, "A comparison of machine learning techniques for phishing detection," in *Proceedings of the Anti-Phishing Working Groups 2nd Annual eCrime Researchers Summit (eCrime '07)*, Association for Computing Machinery, New York, NY, USA, 2007, pp. 60–69. doi: 10.1145/1299015.1299021.

[3] Naser Abdullah Alam. (2024). Phishing Email Dataset [Data set]. Kaggle. https://doi.org/10.34740/KAGGLE/DS/5074342

[4] Garera, S., Provos, N., Chew, M., & Rubin, A. D. (2007). A framework for detection and measurement of phishing attacks. En *Proceedings of the 2007 ACM Workshop on Recurring Malcode (WORM '07)* (pp. 1–8). Association for Computing Machinery. https://doi.org/10.1145/1314389.1314391

[5] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781. https://arxiv.org/abs/1301.3781

[6] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. En Moschitti, A., Pang, B., & Daelemans, W. (Eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://aclanthology.org/D14-1162/

[7] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention is all you need*. arXiv preprint arXiv:1706.03762. https://arxiv.org/abs/1706.03762

[8] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805. https://arxiv.org/abs/1810.04805

[9] Manning, C. D., Raghavan, P., & Schütze, H. (2009). Introduction to Information Retrieval (Online edition). Cambridge University Press. Disponible en PDF

[10] Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition, 84*, 317–331. https://doi.org/10.1016/j.patcog.2018.07.023

[11] Bergholz, A., De Beer, J., Glahn, S., Moens, M.-F., Paaß, G., & Strobel, S. (2010). New filtering approaches for phishing email. *Journal of Computer Security, 18*(1), 7–35.

[12] Enron Corp y Cohen, W. W. (2015). *Conjunto de datos de correos electrónicos de Enron*. Comisión Reguladora Federal de Energía de los Estados Unidos, comp. [Filadelfia, PA: William W. Cohen, MLD, CMU]. Recurso electrónico. Recuperado de https://www.loc.gov/item/2018487913/

[13] Beato, A. (s.f.). *SpamAssassin Public Corpus*. Kaggle. https://www.kaggle.com/datasets/beatoa/spamassassin-public-corpus (Corpus público de SpamAssassin del proyecto Apache).

[14] PhishTank. (s.f.). *Join the fight against phishing*. Recuperado de https://www.phishtank.com/

[15] R. Sood, *Phishing email dataset Nazario_5 and TREC07*. Kaggle, 2023. [En línea]. Disponible: https://www.kaggle.com/datasets/rohansood98/phishing-email-dataset-nazario-5-and-trec07