Universidad del Valle de Guatemala
Facultad de Ingeniería
CC3094 – Security Data Science

Laboratorio 3 – Detección de Malware

Marzo de 2025

Adrian Rodríguez y Daniel Gómez

1. Resumen

En este trabajo, se implementaron dos modelos de aprendizaje para la detección de malware, utilizando secuencias de llamadas a APIs. El primer modelo, basado en aprendizaje automático (ML), empleó un clasificador de Random Forest con características TF-IDF. El segundo modelo, basado en aprendizaje profundo (DL), utilizó una red neuronal alimentada con embeddings generados por la API de Gemini.

La API de Gemini se utilizó con éxito para generar embeddings de alta dimensión a partir de las secuencias de llamadas a la API, sin encontrar limitaciones significativas de velocidad o uso. El preprocesamiento se centró principalmente en el modelo ML, donde se aplicaron técnicas de vectorización como TF-IDF. Para el modelo DL, el preprocesamiento fue relativamente mínimo, ya que los embeddings de Gemini se generaron directamente a partir de las secuencias de texto de las llamadas a la API.

Ambos modelos mostraron un rendimiento sólido en la identificación de software malicioso, pero se realizó un análisis detallado para determinar cuál sería más adecuado para un sistema de detección de malware en producción, como un antivirus comercial. Los resultados sugieren que aunque ambos enfoques son altamente efectivos, existen diferencias importantes en sus características operativas que los hacen más adecuados para diferentes escenarios de aplicación.

2. Introducción

En este laboratorio, implementamos dos enfoques distintos para la detección de malware basados en estas secuencias de llamadas a APIs:

1. Un modelo de aprendizaje automático tradicional utilizando técnicas de representación numérica de texto (TF-IDF) con un clasificador Random Forest.

2. Un modelo de aprendizaje profundo que utiliza embeddings generados por la API de Gemini como entrada para una red neuronal.

Este trabajo detalla la metodología empleada, los resultados obtenidos y un análisis comparativo de ambos enfoques, con el objetivo de determinar cuál es más adecuado para implementaciones reales de sistemas de detección de malware.

3. Metodología

3.1. Conjunto de Datos y Preprocesamiento

Se utilizó un conjunto de datos que contiene secuencias de llamadas a APIs extraídas de la ejecución de programas benignos y maliciosos. Este conjunto de datos fue proporcionado como parte del laboratorio y está basado en el artículo "Automated Behaviour-based Malware Detection Framework Based on NLP and Deep Learning Techniques".

Para el modelo de Random Forest, las secuencias se convirtieron en representaciones TF-IDF considerando tanto unigramas como bigramas. Para el modelo de red neuronal, se generaron embeddings de dimensión 768 utilizando la API de Gemini, que captura información semántica de las secuencias de llamadas.

Los datos se dividieron en conjuntos de entrenamiento (70 %) y prueba (30 %), manteniendo la proporción de clases mediante muestreo estratificado.

3.2. Implementación de Modelos

3.2.1. Modelo 1: Random Forest con TF-IDF

Se implementó un clasificador Random Forest con 100 estimadores, profundidad máxima de 20, criterio de división Gini, y configuraciones adicionales para optimizar su rendimiento en este problema específico.

3.2.2. Modelo 2: Red Neuronal con Embeddings de Gemini

Se diseñó una red neuronal feedforward con la siguiente arquitectura:

- Capa de entrada (768 dimensiones, correspondientes a los embeddings de Gemini)
- Primera capa densa (64 neuronas con activación ReLU)
- Capa de dropout (tasa 0.3)
- Segunda capa densa (32 neuronas con activación ReLU)
- Capa de dropout (tasa 0.2)
- Capa de salida (1 neurona con activación sigmoide)

La red se entrenó utilizando el optimizador Adam, función de pérdida de entropía cruzada binaria, y early stopping para prevenir el sobreajuste.

3.3. Evaluación y Validación

Para una evaluación robusta, se implementó validación cruzada de 10 folds, calculando métricas como precisión (accuracy), precision, recall, F1-score y AUC-ROC. También se generaron matrices de confusión y curvas ROC para un análisis más detallado del rendimiento de clasificación.

4. Resultados

4.1. Rendimiento de los Modelos

Los resultados obtenidos para ambos modelos se presentan y comparan a continuación.

Métrica	**Random Forest con TF-IDF**	**Red Neuronal con Embeddings**
Precisión (Accuracy)	**0.9468**	0.9390
Precisión (Precision)	**0.9886**	0.9388
Recuperación (Recall)	0.9039	**0.9412**
Puntuación F1	**0.9444**	0.9400
AUC-ROC	**0.9829**	0.9775

Cuadro 1: Comparación de métricas entre los dos modelos en el conjunto de prueba

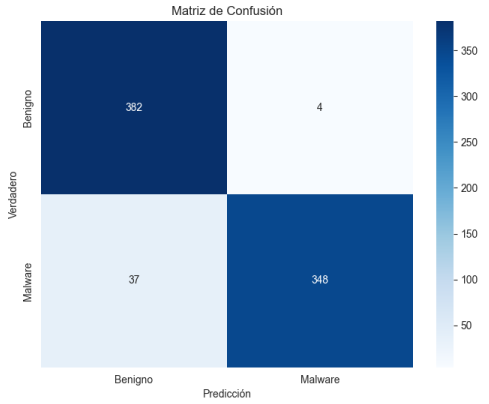
4.1.1. Validación Cruzada

Los resultados de la validación cruzada proporcionan una visión más completa del rendimiento y la estabilidad de ambos modelos:

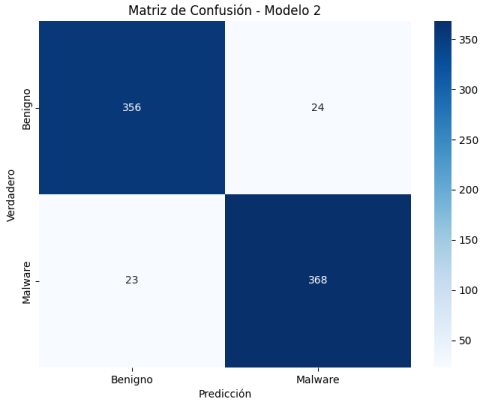
Métrica	**Random Forest con TF-IDF Media (±Desv. Est.)**	**Red Neuronal con Embeddings Media (±Desv. Est.)**
Precisión (Accuracy)	0.9355 (±0.0257)	**0.9500 (±0.0139)**
Precisión (Precision)	**0.9839 (±0.0170)**	0.9540 (±0.0199)
Recuperación (Recall)	0.8856 (±0.0445)	**0.9452 (±0.0146)**
Puntuación F1	0.9317 (±0.0281)	**0.9494 (±0.0140)**
AUC-ROC	**0.9849 (±0.0092)**	0.9834 (±0.0088)

Cuadro 2: Comparación de métricas de validación cruzada (10-folds)

Esta comparación revela un aspecto interesante: aunque el modelo Random Forest muestra un mejor rendimiento en el conjunto de prueba específico, el modelo de red neuronal con embeddings de Gemini presenta mayor estabilidad y un rendimiento promedio superior en la validación cruzada, especialmente en términos de recall (capacidad para detectar correctamente el malware).

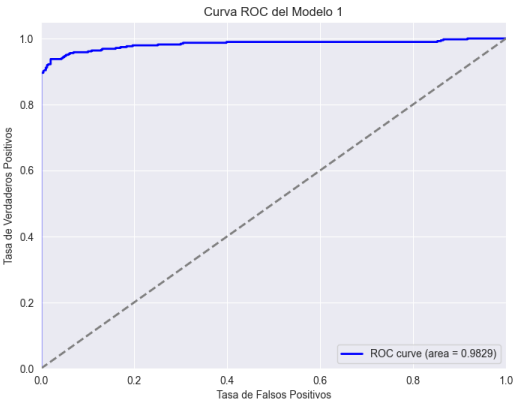


(a) Matriz de confusión del modelo Random Forest

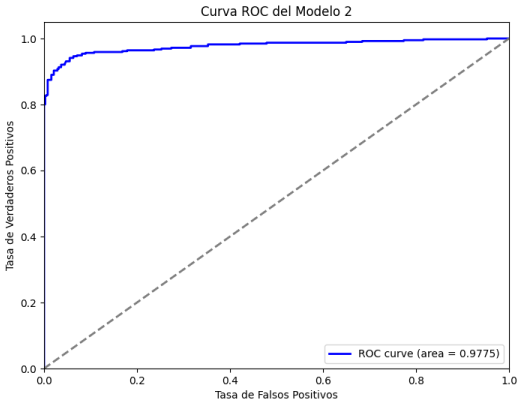


(b) Matriz de confusión del modelo de Red Neuronal

Figura 1: Comparación de matrices de confusión entre ambos modelos



(a) Curva ROC del modelo Random Forest



(b) Curva ROC del modelo de Red Neuronal

Figura 2: Comparación de curvas ROC entre ambos modelos

5. Análisis

El modelo de Random Forest destaca por su alta precisión (precision), lo que indica una baja tasa de falsos positivos. Esto significa que cuando este modelo clasifica un programa como malware, existe una alta probabilidad de que realmente lo sea. Por otro lado, el modelo de red neuronal con embeddings de Gemini muestra un mayor recall, lo que indica una mejor capacidad para identificar correctamente la mayoría de las muestras de malware, aunque con un ligero aumento en los falsos positivos.

Esta diferencia es fundamental desde una perspectiva operativa. En un sistema de detección de malware, existe un equilibrio delicado entre detectar todas las amenazas posibles

(maximizar el recall) y evitar clasificar erróneamente software legítimo como malicioso (maximizar la precisión).

5.1. Análisis de Fortalezas y Debilidades

Característica	Random Forest con TF-IDF	Red Neuronal con Embeddings
Fortalezas principales	▪ Alta precisión (precision) con mínimos falsos positivos ▪ Mayor interpretabilidad ▪ Menor dependencia de servicios externos ▪ Eficiencia computacional en inferencia	▪ Alto recall (detecta más malware) ▪ Mayor estabilidad (menor varianza) ▪ Mejor generalización potencial ▪ Captura relaciones semánticas complejas
Debilidades principales	▪ Menor recall (pierde algunas muestras de malware) ▪ Mayor variabilidad entre folds ▪ Adaptación limitada a nuevos patrones ▪ Preprocesamiento más complejo	▪ Mayor tasa de falsos positivos ▪ Menor interpretabilidad ▪ Dependencia de API externa ▪ Mayor complejidad de implementación

Cuadro 3: Análisis de fortalezas y debilidades de los modelos

5.2. Análisis de Casos Específicos

Para profundizar en el comportamiento de los modelos, analizamos casos específicos donde los modelos mostraron discrepancias en sus predicciones:

5.2.1. Casos de Falsos Negativos del Modelo Random Forest

El modelo de Random Forest tendió a clasificar erróneamente como benignas muestras de malware que exhibían comportamientos menos comunes o que utilizaban patrones de llamadas a APIs más similares a software legítimo. Estas muestras generalmente presentaban secuencias de llamadas que, individualmente, no son sospechosas, pero cuya combinación específica podría indicar comportamiento malicioso.

Este patrón sugiere una limitación en el enfoque TF-IDF, que se centra en la frecuencia de ocurrencia de tokens específicos sin capturar completamente las relaciones semánticas más complejas entre ellos.

5.2.2. Casos de Falsos Positivos del Modelo de Red Neuronal

El modelo de red neuronal, por otro lado, mostró una tendencia a generar falsos positivos en casos de software legítimo que utiliza patrones de llamadas a APIs típicamente asociados con malware, como operaciones intensivas en el registro del sistema, manipulación de procesos o ciertos patrones de networking. Estos software legítimos, a menudo herramientas de sistema o utilidades de networking, pueden compartir características comportamentales con malware sin ser maliciosos.

Esta observación sugiere que los embeddings de Gemini están capturando patrones semánticos a un nivel más profundo, pero ocasionalmente pueden asignar similitudes semánticas excesivas entre comportamientos legítimos especializados y patrones maliciosos.

5.3. Implicaciones para Diferentes Escenarios de Implementación

En un sistema antivirus tradicional destinado a usuarios finales, la prioridad suele ser minimizar los falsos positivos, ya que estos pueden resultar en la cuarentena o eliminación de archivos legítimos, causando interrupciones significativas en la experiencia del usuario. En este contexto, el modelo de Random Forest con TF-IDF sería más adecuado debido a su alta precisión.

Ahora bien, en entornos corporativos con equipos de seguridad dedicados, especialmente en sectores altamente sensibles como finanzas o infraestructura crítica, la prioridad puede inclinarse hacia maximizar la detección de todas las posibles amenazas, incluso a costa de un aumento en los falsos positivos. En estos contextos, los falsos positivos son generalmente revisados por analistas de seguridad antes de tomar acciones definitivas.

5.4. Análisis de Eficiencia Computacional

Otro aspecto relevante para la implementación práctica es la eficiencia computacional. Aquí, ambos modelos presentan diferentes perfiles:

Aspecto	**Random Forest con TF-IDF**	**Red Neuronal con Embeddings**
Tiempo de Inferencia	Más rápido para inferencia una vez entrenado	Ligeramente más lento para inferencia, pero la diferencia es marginal en hardware moderno
Preprocesamiento	Computacionalmente intensivo, especialmente para vectorización TF-IDF de nuevas muestras	Requiere llamadas a API externa para generación de embeddings, introduciendo latencia y dependencias
Escalabilidad	El espacio vectorial crece con el vocabulario de APIs	Dimensionalidad fija (768) independientemente del vocabulario
Requisitos de Memoria	Mayor para almacenar la matriz TF-IDF dispersa	Menor para almacenar embeddings densos

Cuadro 4: Comparación de eficiencia computacional

Esta comparación de eficiencia es particularmente relevante para implementaciones en dispositivos con recursos limitados o para sistemas que requieren análisis en tiempo real de grandes volúmenes de datos.

6. Conclusiones

6.1. Hallazgos Principales

1. Ambos modelos muestran un rendimiento excepcional en la detección de malware basada en secuencias de llamadas a APIs, con métricas generales por encima del 93 % en todas las dimensiones evaluadas.

2. El modelo de Random Forest con TF-IDF destaca por su alta precisión y baja tasa de falsos positivos, haciéndolo particularmente adecuado para sistemas antivirus destinados a usuarios finales donde las interrupciones causadas por falsos positivos deben minimizarse.

3. El modelo de red neuronal con embeddings de Gemini muestra un mayor recall y estabilidad, sugiriendo una mejor capacidad para detectar una gama más amplia de malware, incluyendo potencialmente variantes menos comunes o más nuevas.

4. La elección entre ambos modelos depende fundamentalmente del contexto de implementación y del equilibrio deseado entre minimizar falsos positivos y maximizar la detección de amenazas.

5. Los embeddings generados por la API de Gemini demuestran ser una representación poderosa para capturar relaciones semánticas en secuencias de llamadas a APIs, ofreciendo una alternativa prometedora a enfoques tradicionales como TF-IDF.

6. La interpretabilidad del modelo Random Forest proporciona ventajas significativas para el análisis forense y la comprensión de los patrones de comportamiento del malware.

7. Referencias

1. "Automated Behaviour-based Malware Detection Framework Based on NLP and Deep Learning Techniques Referencia del artículo base para la construcción del dataset.

2. Documentación de la API de Gemini para embeddings de clasificación: https://ai.google.dev/gemini-api/docs/embeddings

3. Documentación sobre tipos de tareas para embeddings de Gemini: https://ai.google.dev/api/embeddings#v1beta.TaskType

4. Tutorial sobre clasificadores de texto con embeddings: https://github.com/google/generative-ai-docs/blob/main/site/en/gemini-api/tutorials/text_classifier_embeddings.ipynb

5. Recursos adicionales sobre embeddings y puntuaciones de similitud: https://www.kaggle.com/code/markishere/day-2-embeddings-and-similarity-scores