Port2Vec: Generación de *embeddings* a partir de descripciones de servicios de red para su uso en Inteligencia Artificial aplicada a la Ciberseguridad

Eñaut Genua, Mikel Iturbe, Iñaki Garitano, Xabier Etxezarreta, Aitor Aguirre y Urko Zurutuza
Mondragon Unibertsitatea
Loramendi 4, 20500 Arrasate.Mondragon
enaut.genua@alumni.mondragon.edu, {miturbe, igaritano, xetxezarreta, aaguirre, uzurutuza}@mondragon.edu

***Resumen*—En este trabajo se presenta un enfoque novedoso para la representación de información relacionada con servicios de red, para su posterior uso en aplicaciones de aprendizaje automático orientadas a la seguridad en redes de comunicaciones. La mayoría de las propuestas de investigación orientadas a generar sistemas de detección de anomalías o sistemas de detección de intrusiones en redes que se han analizado, codifican el puerto destino y/o el puerto origen de flujos o de paquetes utilizando distintas estrategias, ya sea tratándolos como variables categóricas, numéricas, o utilizando técnicas de codificación *one-hot*. En este trabajo se propone la generación de *embeddings* a partir de las descripciones de cada puerto desarrolladas por IANA, enriquecidas con descripciones de los puertos obtenidas a través de IA Generativa. Esta metodología permite capturar el significado semántico subyacente de los servicios asociados a cada puerto, proporcionando representaciones más ricas y contextualizadas para modelos de aprendizaje automático.**

***Index Terms*—Ciberseguridad, detección de intrusiones, ingeniería de características, aprendizaje automático**

Tipo de contribución: *Investigación original (límite 8 páginas)*

I. Introducción

En el ámbito de la ciberseguridad y el análisis de tráfico de red, los números de puerto juegan un papel fundamental en la identificación de servicios y protocolos utilizados en la comunicación entre dispositivos. Tradicionalmente, los modelos de aprendizaje automático que trabajan con datos de tráfico de red, ya sea analizando flujos o paquetes de red, utilizan diferentes estrategias para codificar los números de puertos. En los casos en los que se usa esta característica como una variable numérica continua [1], [2], presupone erróneamente que dos números de puertos correlativos son más parecidos o cercanos a aquellos que no lo son. A modo ilustrativo, si se tomaran los números de puerto 80, 88, y 443 asignados (pero no obligatoriamente) a los servicios HTTP, autenticación de Kerberos y HTTP sobre TLS, respectivamente, el tratamiento de esta característica como variable numérica en algoritmos de agrupación indicaría que Kerberos y HTTP son más cercanos o parecidos que el propio HTTP y HTTPS. Si se tomaran las características como variables categóricas, provocaría que cada número de puerto se considerase como única referencia para un servicio de red, obviando que, para este caso, HTTP y HTTPS pueden estar relacionados en una misma interacción entre atacante y víctima. Finalmente, en otros casos se emplean técnicas como la codificación *one-hot* [3], [4] para representar cada puerto como una nueva característica en el conjunto de datos para su entrenamiento de forma similar al uso de variables categóricas. Sin embargo, genera vectores de alta dimensionalidad, especialmente cuando se trabaja con una gran cantidad de puertos (comportamientos relacionados con escaneos de red, denegación de servicio,...), provocando la necesidad de altas capacidades de computación y multiplicando el tiempo necesario para el entrenamiento y la inferencia de los modelos de aprendizaje.

Para abordar esta problemática, proponemos **Port2Vec** (todo el código está disponible en este repositorio[1]), un enfoque basado en técnicas de Procesamiento del Lenguaje Natural (PLN) para la generación de *embeddings* a partir de las descripciones textuales de los puertos de red. En lugar de tratar los números de puerto como valores discretos sin contexto, utilizamos técnicas de representación de texto para capturar la información semántica de los servicios asociados a cada puerto. Este enfoque permite que los modelos de aprendizaje automático aprovechen una representación más informativa y contextualizada, con la ventaja añadida de que los *embeddings* se computan una sola vez y permite una transcripción directa entre cada número de puerto y su representación vectorial.

En este trabajo, exploramos la metodología de generación de *embeddings* de descripciones de puertos, analizamos su efectividad en comparación con el uso tradicional de números de puerto y codificación *one-hot*. Los resultados sugieren que la incorporación de información semántica mediante *Port2Vec* puede mejorar la capacidad de los modelos para identificar relaciones y anomalías dentro del tráfico de red, y hacerlo en tiempos de computación razonables.

II. Fundamentos

El desarrollo de *Port2Vec* se sustenta en varios conceptos fundamentales del procesamiento del lenguaje natural y la representación vectorial de información, así como en el conocimiento establecido sobre puertos de red y sus servicios asociados.

II-A. Puertos de Red y Servicios

Los puertos de red son puntos de conexión lógicos que permiten la comunicación entre aplicaciones y servicios a través de redes TCP/IP. Cada puerto se identifica mediante un número entero sin signo de 16 bits, que debido a esto, hay $2^{16}-1$ puertos, y puede estar asociado a servicios específicos, como HTTP (puerto 80), SSH (puerto 22) o DNS (puerto 53). *Internet Assigned Numbers Authority* (IANA) [5] mantiene un

[1]Repositorio: https://gitlab.danz.eus/danz/segurtasuna/lip/port2vec.git

registro de asignaciones de puertos que incluye descripciones detalladas de los servicios y protocolos asociados a cada número de puerto. Estos puertos se asignan en varios modos, basándose en tres grupos principales:

- **Puertos de Sistema:** Los puertos de sistema, o puertos bien conocidos, abarcan el rango del puerto 0 al 1023, y están asignados a servicios fundamentales como HTTP, SSH o SMTP. Su función es ser el canal predecible para comunicaciones estándar. Estos puertos son controlados y asignados por IANA, y en la mayoría de los sistemas, el uso de estos puertos está restringido para los procesos con permisos privilegiados.
- **Puertos de usuario:** Los puertos de usuario, o puertos asignados, abarcan el rango del puerto 1024 al 49151 y son utilizados por programas que han sido oficialmente asignados para operar en un puerto específico, lo que permite tener una menor probabilidad de conflicto con otros servicios básicos. IANA no puede controlar el uso de estos puertos, pero sí que registra y asigna los usos como conveniencia a la comunidad.
- **Puertos dinámicos:** Los puertos dinámicos, también conocidos como puertos privados o efímeros, abarcan el rango del puerto 49152 al 65535, y nunca son ni serán asignados a ningún servicio por IANA, ni deberían por recomendación de los mismos.

A pesar de la existencia de estos grupos de puertos, nada impide que un servicio como SSH, que debería operar en el puerto 22, opere en otro puerto, como puede ser el 2222.

II-B. Representación Vectorial de Texto

La representación vectorial de texto, o *word embeddings* es una técnica fundamental en el PLN que permite convertir palabras o frases en vectores numéricos densos. Estos vectores capturan relaciones semánticas y sintácticas entre palabras, permitiendo que términos con significados similares se ubiquen cerca en el espacio vectorial. Modelos como *word2vec* [6] y *GloVe* [7] han demostrado la eficacia de estas representaciones en diversas tareas del PLN.

III. Trabajos Relacionados

El uso de representaciones vectoriales ha transformado la manera de abordar problemas tanto en el PLN como en el análisis de datos de red. En el ámbito del PLN, modelos como *Word2Vec* [6] y *GloVe* [7] han demostrado que es posible capturar relaciones semánticas entre palabras al proyectarlas en un espacio vectorial. La idea central es que elementos semánticamente similares queden próximos entre ellos en dicho espacio.

En el contexto de la ciberseguridad y el análisis de tráfico de red, ha sido común utilizar técnicas tradicionales como la codificación *one-hot* para representar atributos categóricos, incluidos los números de puerto o servicios. Este enfoque ha sido empleado en numerosos estudios de detección de intrusiones y clasificación de tráfico [3], [4]. Sin embargo, la codificación *one-hot* trata cada puerto como una categoría completamente independiente, lo que resulta en vectores de alta dimensionalidad, requiriendo mayores recursos y no captura ninguna relación semántica entre ellos. Para evitar eso, Figueiredo et al. [8] presenta una manera de evitar la gran dimensionalidad del vector resultante, capando el número máximo del puerto en 4096, y así, perdiendo la capacidad de distinguir los puertos mayores a 4096.

El uso directo de valores numéricos de puerto es un enfoque alternativo, empleado en diferentes estudios [1], [2], que también presenta limitaciones significativas. El uso directo de los números de puerto como características numéricas puede ser problemático, ya que la magnitud del número de puerto no tiene una relación directa con su función o comportamiento en la red. Por ejemplo, el puerto 21 (FTP) no está funcionalmente "más cerca" del puerto 22 (SSH) que del puerto 25 (SMTP), a pesar de su proximidad numérica.

De forma similar a este trabajo, Ring et al. presenta en [9] IP2Vec, como una medida de similitud para direcciones IP basada en Word2Vec. La propuesta aprende similitudes extrayendo información contextual de datos de red, considerando similares aquellas direcciones IP que aparecen en contextos similares. La evaluación experimental en dos conjuntos de datos públicos demuestra su efectividad para agrupar direcciones IP en redes botnet y, mediante métodos de visualización, confirma la capacidad de IP2Vec para capturar similitudes basadas en patrones de comunicación. La propuesta DarkVec y su posterior evolución i-DarkVec [10], [11] construyen un espacio de vectores o *embedding* único, estático e inmutable a partir de todos los datos observados en una darknet, sin ser específicas en las relaciones entre puertos.

En [12] se presenta un esquema de aprendizaje automático para rastrear actividades de ataque en la darknet mediante extracción y búsqueda de similitudes de características de puertos de destino con FastText, reducción dimensional con UMAP [13] y agrupamiento con DBSCAN [14]. Los experimentos utilizan tráfico de un sensor darknet /16 durante un mes para demostrar la eficacia del método propuesto. Esta investigación ya identifica una problemática similar, pero una aproximación que no se basa en convertir los puertos en características con semántica, sino en modelar los propios puertos para analizar las relaciones subyacentes.

Al igual que en nuestra propuesta, están emergiendo propuestas similares que ofrecen modelos pre-entrenados de tráfico para facilitar la labor de análisis. Peng et al. proponen recientemente PTU (*Pre-trained model for network Traffic Understanding*) en [15], como el diseño de un esquema de representación de tráfico que integra el contenido estático de los paquetes y las dinámicas de red en un espacio de entrada unificado. En esta ocasión también, los modelos y *embeddings* involucran todo el comportamiento del tráfico, no permitiendo identificar características intrínsecas de la red en un modelo de detección.

El presente trabajo introduce *Port2Vec*, como una metodología que genera *embeddings* de puertos a partir de sus descripciones textuales. Al igual que en PLN, esta estrategia permite que los puertos con significados o funciones similares—por ejemplo, el puerto 80 (HTTP) y 443 (HTTPS)—se representen mediante vectores cercanos en el espacio vectorial. Esta representación no solo reduce la dimensionalidad en comparación con la codificación *one-hot*, sino que también facilita la integración de la información semántica de los puertos en modelos de aprendizaje automático.

De este modo, *Port2Vec* se posiciona como una propuesta que aprovecha avances en técnicas de representación de características en espacios vectoriales (*embedding*) para superar

las limitaciones de las representaciones tradicionales y ofrecer una visión más rica y contextualizada de la funcionalidad de los puertos en entornos de red.

IV. Metodología

En este trabajo, se propone una metodología para generar representaciones vectoriales densas (*embeddings*) para el espacio completo de puertos de red.

Esta metodología transforma cada puerto en un vector de dimensión controlada que codifica sus características semánticas. El proceso (Figura 1) se estructura en cuatro fases secuenciales:

1. Adquisición de datos.
2. Preprocesamiento de datos sobre servicios de red.
3. Generación de representaciones vectoriales mediante modelos de *embeddings* neuronales.
4. Reducción de dimensionalidad preservando relaciones semánticas.

Adicionalmente, se implementa una variante aumentada utilizando *LLM*s (Modelos del Lenguaje de Gran Escala) para enriquecer el contenido semántico de los *embeddings*.

Un aspecto crítico de la metodología es su naturaleza agnóstica respecto a modelos específicos, permitiendo la sustitución de componentes individuales con implementaciones alternativas o modelos mejorados sin alterar el marco general. Esta característica garantiza la extensibilidad de la metodología de *Port2Vec* ante avances futuros en tecnologías de *embeddings* semánticos.

IV-A. Adquisición de Datos y Preprocesamiento

Como fuente de datos primarios se utilizó el registro oficial "*IANA Service Names and Port Numbers*" [5], que constituye la referencia autoritativa para la asignación global de puertos de red. Esta fuente proporciona metadatos estructurados para cada puerto.

La evaluación preliminar de los datos reveló heterogeneidad significativa en la completitud y calidad de los metadatos disponibles. Específicamente, se identificaron varios desafíos: ausencia de descripciones, falta de asignación de nombre de servicio, representaciones mediante rangos numéricos, y ausencia total de asignaciones para puertos dinámicos (49152-65535).

Para solventar dichas limitaciones, se implementó un método de procesamiento de los registros, estructurado en cuatro etapas:

1. **Normalización sintáctica:** Se aplicaron transformaciones determinísticas para estandarizar las representaciones textuales y numéricas, incluyendo resolución de valores nulos en protocolos mediante la asignación del designador "*any*", sustitución de nombres de servicio ausentes basado en patrones léxicos, y estandarización de variaciones ortográficas y sintácticas.
2. **Complementación de los datos:** Se añadieron entradas para los puertos sin representación explícita en el registro IANA, particularmente el segmento dinámico/privado (49152-65535), incluyendo la información asociada. Esta información asociada se ha metido manualmente, poniendo como protocolo el valor "*any*" y como descripción el valor "*Dynamic and/or Private Port*". Este procedimiento garantiza la exhaustividad del conjunto de datos, condición necesaria para la generación completa de *embeddings*.
3. **Expansión de rangos:** Se implementó una función de expansión que transforma rangos de puertos (e.g. "49152-65535") en registros individuales, generando entradas discretas para cada puerto mientras se preservan los metadatos asociados.
4. **Generación de características textuales:** Se diseñó un esquema de representación textual unificada que integra los atributos discretos en una descripción estructurada mediante una plantilla consistente que incluye nombre del servicio, número de puerto, protocolo, clasificación

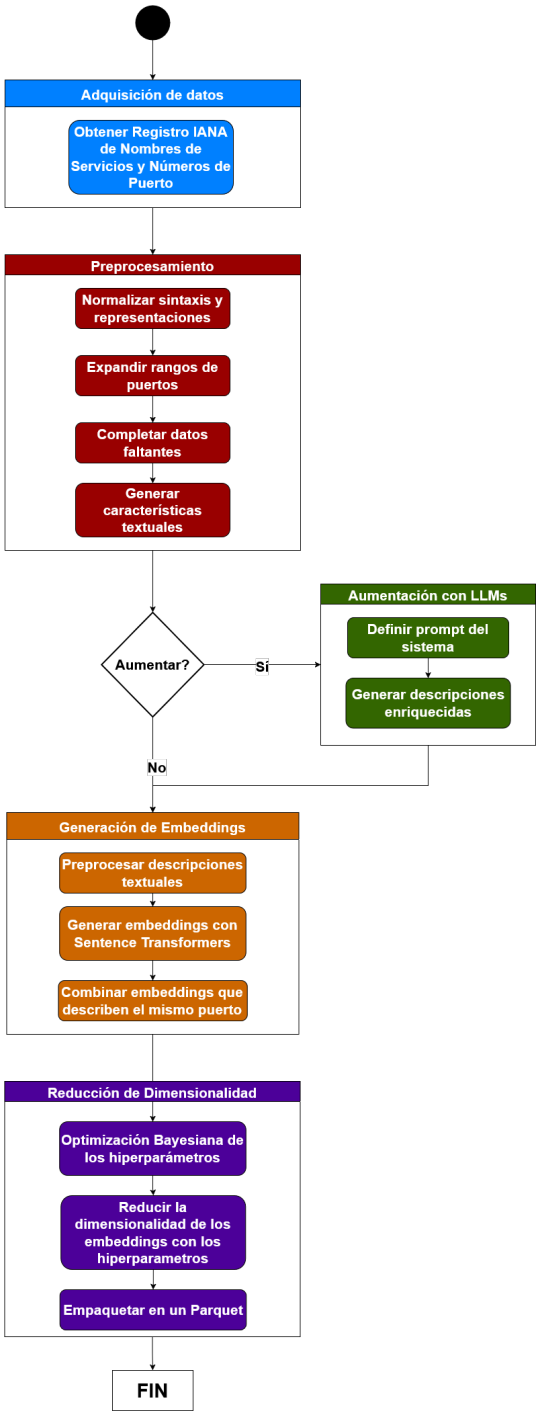

Figura 1. Diagrama de flujo de la metodología

(puerto de sistema/usuario/dinámico) y descripción original. Como ejemplo, una de las características textuales finales del servicio SSH es la siguiente: "*The 'ssh' service runs on network port 22 using the TCP protocol (system port). Description: The Secure Shell (SSH) Protocol*"

La verificación de la exhaustividad del conjunto de datos procesado se realizó mediante la comprobación de la representación de los 65.536 puertos posibles y la unicidad de cada puerto en el conjunto final.

IV-B. Aumentación con LLMs

Para enriquecer las características textuales del conjunto final, se implementó un procedimiento de aumento semántico utilizando modelos de lenguaje de gran escala (LLM). Este componente complementa la generación de características textuales mediante la incorporación de conocimiento técnico contextual no explícitamente presente en el registro de IANA, manteniendo la independencia del modelo específico utilizado. Por ejemplo, la descripción del puerto 153 (SGMP) en el registro dice: "*SGMP*", pero con el aumento semántico conseguimos: "*Simple Gateway Monitoring Protocol used for monitoring and managing network gateways*".

El procedimiento de aumento semántico sigue los siguientes pasos:

1. **Definición del *prompt* del sistema:** Se diseñó un *prompt* específico para guiar al modelo en la generación de descripciones técnicamente precisas y semánticamente enriquecidas, con directrices para preservar la información original mientras se añade contexto relevante.
2. **Generación de descripciones aumentadas:** Para cada puerto, se genera una descripción enriquecida utilizando la descripción original como contexto.

Aunque el modelo usado es *deepseek-ai/DeepSeek-R1-Distill-Llama-70B* [16], la arquitectura permite la sustitución del modelo de lenguaje utilizado para el aumento, adaptándose a la futura actualización a nuevos modelos con mayor capacidad de razonamiento técnico o conocimiento específico del dominio.

IV-C. Generación de Embeddings

La codificación de las representaciones textuales en vectores numéricos se realizó mediante un marco agnóstico de modelos, diseñado para funcionar con cualquier modelo de *embeddings* para frases, basándose en la tecnología de *Sentence Transformers* [17]. Esta aproximación desacopla la metodología de implementaciones específicas, permitiendo la integración fluida de avances futuros en modelos de *embeddings* sin modificaciones fundamentales al marco general. Esta flexibilidad es particularmente valiosa, considerando la rápida evolución del campo de los modelos de lenguaje y representaciones vectoriales.

El proceso de generación de *embeddings* sigue un flujo estandarizado que abstrae los detalles específicos del modelo subyacente:

1. **Preprocesamiento textual:** La descripción estructurada de cada puerto se estandariza en un formato común para todos los puertos, que incluye la información recolectada del conjunto de datos de IANA.

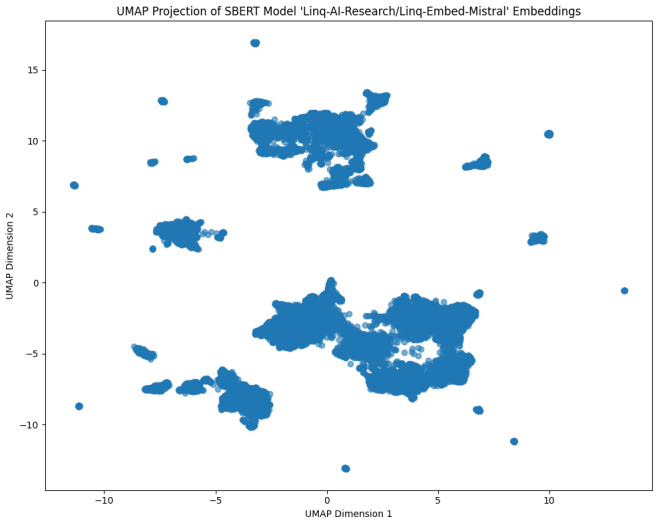

Figura 2. Projección con UMAP [13] de los *embeddings* generados por *Linq-AI-Research/Linq-Embed-Mistral* [18] con aumentación.

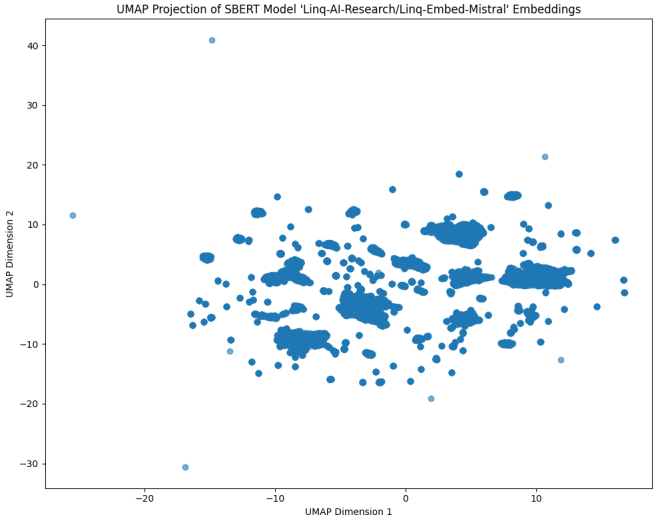

Figura 3. Projección con UMAP [13] de los *embeddings* generados por *Linq-AI-Research/Linq-Embed-Mistral* [18] sin aumentación.

2. **Generación de *embeddings*:** Cada puerto se procesa mediante el modelo seleccionado para obtener su representación vectorial en la dimensionalidad nativa del modelo.
3. **Agregación para puertos multifuncionales:** Para puertos con múltiples entradas en el registro (debido a registros duplicados con distintos protocolos), se aplica una suma vectorial seguida de normalización para obtener una representación unificada.

Este procedimiento garantiza la consistencia en la generación de *embeddings* independientemente del modelo subyacente, facilitando la actualización futura con modelos mejorados.

El análisis de las visualizaciones UMAP muestra los beneficios del aumento mediante LLM en la generación de representaciones vectoriales de puertos de red. La comparación entre las dos proyecciones revela que las representaciones aumentadas por LLM (Figura 2) presentan agrupaciones notablemente más definidas que las representaciones no au-

mentadas (Figura 3). En la Figura 3, los puntos aparecen dispersos con límites difusos entre grupos, indicando una menor coherencia semántica donde puertos con funciones relacionadas pueden no estar consistentemente posicionados cerca entre sí. En contraste, la Figura 2 muestra clústeres claramente delimitados y concentrados, sugiriendo una organización más significativa basada en similitudes funcionales, donde puertos pertenecientes a servicios relacionados quedan agrupados coherentemente.

IV-D. Reducción de Dimensionalidad

Las representaciones nativas generadas por los modelos son vectores, siendo típicamente de alta dimensionalidad y son subóptimas para aplicaciones prácticas debido al coste computacional que pueden producir. Debido a esto, se implementó un marco agnóstico de técnicas de reducción dimensional, que permite la integración o sustitución de antedichas técnicas, adaptándose a diferentes requisitos de preservación topológica. Esta flexibilidad facilita la adaptación a diferentes casos de uso con requisitos específicos de preservación de estructura o eficiencia computacional.

La búsqueda de los mejores hiperparámetros para reducir la dimensionalidad de los *embeddings*, se realizó mediante un procedimiento bayesiano utilizando el algoritmo Tree-structured Parzen Estimator (TPE). Este enfoque modela la distribución de hiperparámetros condicionada al rendimiento observado, adaptando la estrategia de búsqueda según los resultados anteriores.

Un aspecto crítico de la metodología es la inclusión del algoritmo de reducción dimensional como un hiperparámetro dentro del espacio de búsqueda. Esto permite la selección automática del método más adecuado junto con sus parámetros específicos. Para cada técnica de reducción, se definió un subespacio de búsqueda de parámetros relevantes.

El procedimiento de optimización implementó varias estrategias para aumentar la eficiencia:

1. **Búsqueda inicial aleatoria:** Las primeras iteraciones exploran el espacio de hiperparámetros de manera aleatoria para construir un mapa inicial de los resultados.
2. **Búsqueda adaptativa:** Las iteraciones siguientes utilizan el conocimiento acumulado del mapa inicial para buscar hiperparámetros en regiones prometedoras del espacio de búsqueda.
3. **Detección automática de convergencia:** El proceso monitoriza la mejora relativa entre iteraciones, con un umbral de mejora mínima y un mecanismo de paciencia que detiene la optimización cuando no se observan mejoras significativas durante un número predeterminado de iteraciones.

IV-E. Evaluación de Calidad de Representación

La evaluación cuantitativa de la calidad de las representaciones se realizó mediante dos métricas complementarias que evalúan aspectos diferenciados de la preservación de la estructura:

1. ***Trustworthiness:*** Cuantifica la preservación de vecindades locales, evaluando en qué medida los puntos que son vecinos cercanos en el espacio reducido también lo son en el espacio original. Esta métrica penaliza las relaciones de vecindad "falsas" introducidas por la reducción, siendo una vecindad "falsa" aquella donde un punto aparece entre los k vecinos más cercanos en el espacio reducido pero no figura entre los k vecinos más cercanos en el espacio original. El parámetro k define el umbral que determina cuándo se considera que dos puntos son vecinos.
2. ***Shepard Goodness of Fit:*** Evalúa la preservación global de la estructura de distancias mediante el coeficiente de correlación entre las distancias por pares en ambos espacios.

Como métrica de distancia se usó la distancia del coseno, debido a que es la dirección lo que determina el carácter semántico del *embedding* y no la magnitud.

IV-F. Artefactos Resultantes

Los artefactos principales de este marco metodológico comprenden matrices de *embeddings* para cada combinación de modelo y dimensionalidad. Estos artefactos se generaron manteniendo la separación entre los derivados de las características textuales aumentadas y las originales.

Los *embeddings* finales se codificaron en formato Apache Parquet [19] con compresión Zstandard [20], estructurados como arrays multidimensionales de números de punto flotante con precisión de 32 bits, indexados por el número de puerto. Esta representación permite la fácil integración en otras aplicaciones de aprendizaje automático y análisis de redes.

La implementación final proporciona una API programática que facilita la lectura de los *embeddings* de los archivos Apache Parquet.

V. Entorno de Experimentación

La evaluación de los *embeddings* se realizó mediante un diseño experimental centrado en su aplicación práctica para tareas de clasificación de flujos de red. Este enfoque permite evaluar directamente la utilidad de los *embeddings* en escenarios reales de análisis de tráfico y seguridad.

En esta sección se describen las configuraciones y recursos utilizados para la evaluación experimental de los embeddings de puertos generados mediante Port2Vec. Se detallan tanto los aspectos técnicos del entorno computacional como las arquitecturas de aprendizaje automático empleadas para los experimentos de clasificación.

V-A. Infraestructura Computacional

Los experimentos se realizaron en una estación de trabajo con las siguientes especificaciones:

- CPU: AMD EPYC 7773X
- RAM: 130 GB
- GPU: NVIDIA RTX A6000 (48GB VRAM)
- SO: Ubuntu 22.04.5 LTS

Esta configuración permitió el procesamiento de los 65.535 puertos y la generación de *embeddings*, así como el entrenamiento paralelo de modelos de clasificación para la evaluación comparativa.

V-B. Conjunto de Datos para la Evaluación

Para esta evaluación se utilizó el conjunto de datos CIC-IDS-2017 mejorado [21]. Este conjunto proporciona flujos de red capturados durante varios días de actividad normal

y ataques simulados, con etiquetas detalladas para diferentes tipos de comportamiento malicioso.

El proceso de preprocesamiento fue exhaustivo para garantizar la calidad de los datos. Inicialmente, se eliminaron todos los registros que contenían valores nulos o presentaban inconsistencias que pudieran afectar el análisis. A continuación, se procedió con un filtrado riguroso para eliminar duplicados y valores atípicos extremos que podrían distorsionar los resultados. Los flujos clasificados como ataques intentados fueron reclasificados como benignos, ya que no representan amenazas reales al sistema.

Las variables categóricas presentes en el conjunto de datos fueron debidamente codificadas para su uso en los modelos de aprendizaje automático. Se realizó también una cuidadosa selección de características para reducir la dimensionalidad y mejorar la eficiencia del modelo. Un aspecto importante del preprocesamiento fue la eliminación deliberada de ciertos tipos de ataques, específicamente los escaneos de puertos y ataques de denegación de servicio (DoS), permitiendo así que el análisis se centrara en la detección de patrones de tráfico más sutiles y sofisticados.

Después de esto se generaron tres conjuntos de datos, y cada uno tenía una estrategia de preprocesamiento de los puertos diferente:

- Sin procesamiento especial (variable numérica continua).
- Codificación *one-hot*.
- Uso de los *embeddings* generados.

V-C. Arquitectura del Clasificador

Se implementó un clasificador de flujos basado en redes neuronales con una arquitectura diseñada para optimizar la detección de patrones en el tráfico de red. La estructura comienza con una capa de entrada adaptada específicamente a la dimensionalidad de las características extraídas durante el preprocesamiento. Para estabilizar el proceso de entrenamiento y acelerar la convergencia, se incorporó una capa de normalización por lotes inmediatamente después de la entrada.

El núcleo de la red está formado por cuatro capas ocultas densamente conectadas, con una configuración descendente de neuronas (512, 256, 128, 64) que permite una abstracción progresiva de los patrones del tráfico. Para preservar el carácter semántico de los *embeddings* a través de las diferentes capas de procesamiento, se implementaron conexiones de salto que mantienen la información esencial durante la propagación hacia adelante.

La robustez del modelo se mejoró mediante la incorporación de regularización por desactivación aleatoria de neuronas con una probabilidad de 0.2, reduciendo así el sobreajuste a los datos de entrenamiento. Finalmente, la arquitectura culmina con una capa de salida específicamente adaptada para la tarea de clasificación binaria, permitiendo discriminar eficientemente entre tráfico normal y malicioso.

V-D. Configuración del Entrenamiento

Para el entrenamiento del modelo se implementó una configuración robusta que permitiera capturar eficientemente los patrones distintivos del tráfico de red malicioso. Se seleccionó el optimizador *AdamW* [22] con una tasa de aprendizaje inicial de 0.01, aprovechando su capacidad para desacoplar la regularización del peso de los pasos de adaptación del gradiente, lo que resulta particularmente beneficioso en redes neuronales profundas.

La tasa de aprendizaje se gestionó dinámicamente mediante un planificador *CosineAnnealingWarmRestarts* [23], permitiendo ciclos de calentamiento que facilitan la exploración del espacio de parámetros y ayudan a escapar de mínimos locales subóptimos. Para abordar adecuadamente el desbalance inherente entre las clases de tráfico normal y malicioso, se implementó una función de pérdida *BCEWithLogitsLoss* con ponderación de clases, asegurando que ambas categorías recibieran la atención apropiada durante la optimización.

Los datos fueron distribuidos siguiendo un esquema estándar de división, asignando el 70 % al conjunto de entrenamiento, 15 % para validación y el 15 % restante para el test. El proceso de entrenamiento se estableció en 10 épocas y *PyTorch* [24] se configuró para que todas las operaciones fuesen reproducibles, para garantizar condiciones experimentales uniformes entre todas las estrategias.

VI. Resultados

Para la evaluación experimental se generaron *embeddings* en tres dimensionalidades: 16, 32 y 64 dimensiones. Estas configuraciones representan diferentes compromisos entre compacidad y capacidad representativa, permitiendo analizar el impacto de la dimensionalidad en el rendimiento. Aunque la metodología es agnóstica respecto al modelo, para estos experimentos se utilizó un único modelo de *embeddings*, *Linq-AI-Research/Linq-Embed-Mistral* [18], que se encuentra entre los modelos abiertos de mejor rendimiento según el benchmark MMTEB (*Massive Multilingual Text Embedding Benchmark*) [25] para inglés en HuggingFace. Se generaron con características textuales aumentadas y sin aumentar.

VI-A. Resultados de la Clasificación

Dada la naturaleza altamente desequilibrada del conjunto de datos, donde el tráfico legítimo supera significativamente al tráfico malicioso, se seleccionaron métricas específicas que proporcionan una evaluación robusta en escenarios de desequilibrio de clases:

- **F1-Score:** Representa la media armónica entre precisión y sensibilidad, proporcionando un equilibrio entre ambas. Esta métrica penaliza tanto los falsos positivos como los falsos negativos, siendo crucial en detección de amenazas donde tanto las falsas alarmas como los fallos de detección tiene costes significativos.
- **Average Precision:** Calcula el área bajo la curva precisión-sensibilidad, capturando el rendimiento del modelo a través de múltiples umbrales de decisión. Al ser independiente de un umbral específico, proporciona una evaluación más completa del rendimiento del clasificador en contextos desequilibrados.

Los resultados de clasificación para cada estrategia de representación de puerto se presentan en la Tabla I.

VI-B. Importancias de las Características

Para complementar el análisis del rendimiento de clasificación, se realizó un estudio detallado de la importancia de las características utilizadas por el modelo, con especial atención al impacto de los *embeddings* de puertos en la toma de decisiones del clasificador.

Tabla I
RENDIMIENTO DEL CLASIFICADOR DE FLUJOS CON DIFERENTES REPRESENTACIONES DE PUERTOS

Estrategia	***F1 Score***	***Average Precision***	***Precision***	***Recall***	**Tiempo de Entrenamiento**
Sin Procesar	0.92	0.81	0.89	0.96	17m 51.123s
One-hot	0.93	0.83	0.89	0.98	1h 37m 47.742s
Port2Vec 16D	0.96	0.95	0.94	0.98	17m 53.663s
Port2Vec 16D LLM	**0.98**	0.97	0.97	0.98	17m 57.751s
Port2Vec 32D	0.97	0.91	0.95	0.99	17m 58.595s
Port2Vec 32D LLM	0.96	**0.98**	0.93	0.98	17m 56.196s
Port2Vec 64D	0.97	0.97	0.95	0.98	17m 58.753s
Port2Vec 64D LLM	0.96	0.87	0.94	0.99	18m 4.568s

La evaluación de importancias se realizó mediante valores SHAP (*SHapley Additive exPlanations*) [26], una técnica fundamentada en la teoría de juegos cooperativos que permite cuantificar la contribución marginal de cada característica a la predicción final del modelo. Para el cálculo específico de las importancias, se siguió un proceso sistemático que permitió la comparabilidad entre las distintas estrategias de representación:

1. Se generaron muestras de fondo utilizando un subconjunto aleatorio del 0.5 % de los datos para cada estrategia.
2. Se seleccionó un conjunto de explicación correspondiente al 0.05 % de los datos para evaluar las contribuciones de las características.
3. Se calcularon los valores SHAP para cada característica en cada estrategia de representación.

En los casos donde los puertos se representan como vectores multidimensionales, la importancia se calculó mediante la suma de las importancias individuales de cada componente del mismo. Se utilizó la propiedad de *local accuracy* (precisión local), la cual asegura que la suma de las contribuciones individuales de las variables equivale a la predicción del modelo. Esto permite que, al descomponer una variable en varios subcomponentes, la suma de sus importancias refleje la importancia global de la variable original.

Los las importancias de los puertos para cada estrategia de representación de puerto se presentan en la Tabla II.

Tabla II
EL PORCENTAJE DE LA CONTRIBUCIÓN DE LOS PUERTOS

Estrategia	**Importancia de los Puertos (%)**
Sin Procesar	0.86 %
One-hot	21.73 %
Port2Vec 16D	45.45 %
Port2Vec 16D LLM	37.05 %
Port2Vec 32D	37.97 %
Port2Vec 32D LLM	41.84 %
Port2Vec 64D	51.96 %
Port2Vec 64D LLM	47.84 %

VII. CONCLUSIONES

Los resultados experimentales obtenidos en este estudio proporcionan evidencia sustancial sobre la efectividad de los *embeddings* de puertos de red para tareas de clasificación de flujos.

VII-A. Efectividad de los embeddings

Los *embeddings* demuestran un rendimiento superior en comparación con los enfoques tradicionales. Mientras que la representación directa sin procesamiento muestra un rendimiento base aceptable (F1: 0.92, AP: 0.81), queda claramente superada por los *embeddings* en todas sus variantes. Es de destacar que el *Average Precision* es la métrica que más se ve mejorada con el uso de los *embeddings*.

La codificación *one-hot*, a pesar de mostrar un rendimiento competitivo (F1: 0.93, AP: 0.83), conlleva una carga computacional excesiva debido a su alta dimensionalidad, como refleja su tiempo de procesamiento significativamente mayor.

VII-B. Impacto de la Dimensionalidad

El análisis de diferentes dimensionalidades revela patrones interesantes en el compromiso entre compacidad y capacidad representativa. Los resultados muestran que los *embeddings* de 16 dimensiones logran un rendimiento muy competitivo (F1: 0.96, AP: 0.95 para la versión estándar). Este hallazgo es particularmente relevante, pues demuestra que los *embeddings* muy compactos pueden capturar información suficiente para realizar eficazmente la tarea de clasificación de tráfico malicioso.

Al examinar configuraciones más amplias, se observa que los *embeddings* de 16 dimensiones alcanzan el rendimiento más alto del estudio, especialmente en su variante enriquecida con LLM, que consigue métricas casi perfectas (F1: 0.98, AP: 0.97). En los *embeddings* de 32 y 64 dimensiones se observa un rendimiento parecido al de los 16 dimensiones, incluso en la variante enriquecida con LLM para los de 32 dimensiones, pero en los de 64 dimensiones se puede observar un deterioro en las métricas. Esto puede indicar que el rendimiento puede ser impactado por la *maldición de la dimensión* en dimensiones altas, como también se puede ver en la variante de la codificación *one-hot*.

Estos resultados proporcionan una valiosa orientación práctica, indicando que en la selección de dimensionalidad se debe considerar cuidadosamente. El estudio demuestra que es posible obtener resultados altamente competitivos incluso con *embeddings* muy compactos, lo que podría ser crucial en entornos con limitaciones de recursos computacionales.

VII-C. Valor del Aumento Semántico Mediante LLM

La comparación entre las variantes estándar y aumentadas con LLM revela beneficios significativos del enriquecimiento semántico.

Las variantes aumentadas con LLM muestran consistentemente mayor *Average Precision* que sus contrapartes estándar, indicando una mejor capacidad para priorizar correctamente las instancias positivas. La mejora es particularmente notable en las representaciones de 32 dimensiones, donde el AP aumenta de 0.91 a 0.98, evidenciando que el conocimiento

contextual incorporado por el LLM proporciona información adicional. También es de destacar que en la variante de 64 dimensiones, el AP cae de 0.97 a 0.87, indicando que las altas dimensiones pueden hacer que el enriquecimiento semántico deteriore el rendimiento.

Estos resultados validan el valor del enfoque de aumento semántico, especialmente cuando se requiere alta precisión en la identificación de tráfico malicioso.

VII-D. Importancia de los Embeddings

El análisis de importancia de características revela hallazgos significativos sobre el valor informativo de los puertos en la clasificación. La Tabla II muestra que la representación sin procesar apenas contribuye a la toma de decisiones del modelo (0.86 %), mientras que la codificación *one-hot* mejora sustancialmente esta cifra (21.73 %).

Los *embeddings*, por otro lado, elevan considerablemente la importancia relativa de los puertos (37.05 %-51.96 %), indicando que capturan información semántica mucho más relevante para el modelo. Sin embargo, se observa que no existe una correlación directa entre la importancia del puerto y el rendimiento del clasificador. Los *embeddings* de 16 dimensiones aumentados con LLM alcanzan la mejor puntuación F1 (0.98) con una importancia relativa de 37.05 %, mientras que los de 32 dimensiones con LLM logran el máximo *Average Precision* (0.98) con 41.84 % de importancia.

Estos resultados evidencian que la calidad de los *embeddings* es más determinante que su mera contribución cuantitativa.

VII-E. Eficiencia computacional

La evaluación de eficiencia computacional presenta implicaciones prácticas importantes:

- La codificación one-hot resulta prohibitivamente costosa (1h 37m 47.742s) en el momento de entrenar el clasificador.
- Los *embeddings* ofrecen un balance superior entre rendimiento y eficiencia, con tiempos de procesamiento significativamente menores que la codificación one-hot.

Esta ventaja en eficiencia, combinada con su superior rendimiento predictivo, posiciona a Port2Vec como una solución práctica y efectiva para sistemas de análisis de tráfico de red y detección de intrusiones.

AGRADECIMIENTOS

Esta publicación es parte del proyecto de I+D+i PLEC2024-011222, financiado por AEI/10.13039/501100011033. Mikel Iturbe, Iñaki Garitano, Xabier Etxezarreta, Aitor Aguirre y Urko Zurutuza son miembros del Grupo de Investigación IT1676-22, financiado por el Departamento de Ciencia, Universidades e Innovación del Gobierno Vasco.

REFERENCIAS

[1] O. Belarbi, A. Khan *et al.*, "An intrusion detection system based on deep belief networks," in *International Conference on Science of Cyber Security*. Springer, 2022, pp. 377–392.

[2] D. Pujol-Perich, J. Suárez-Varela *et al.*, "Unveiling the potential of graph neural networks for robust intrusion detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 49, no. 4, pp. 111–117, 2022.

[3] M. Al-Qatf, Y. Lasheng *et al.*, "Deep learning approach combining sparse autoencoder with svm for network intrusion detection," *IEEE Access*, vol. 6, pp. 52 843–52 856, 2018.

[4] P. Wu and H. Guo, "Lunet: a deep neural network for network intrusion detection," in *2019 IEEE symposium series on computational intelligence (SSCI)*. IEEE, 2019, pp. 617–624.

[5] "Service Name and Transport Protocol Port Number Registry — iana.org," https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[6] T. Mikolov, K. Chen *et al.*, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[7] J. Pennington, R. Socher *et al.*, "GloVe: Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.

[8] J. Figueiredo, C. Serrão *et al.*, "Deep learning model transposition for network intrusion detection systems," *Electronics*, vol. 12, no. 2, 2023.

[9] M. Ring, A. Dallmann *et al.*, "Ip2vec: Learning similarities between ip addresses," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 657–666.

[10] L. Gioacchini, L. Vassio *et al.*, "Darkvec: automatic analysis of darknet traffic with word embeddings," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 76–89.

[11] ——, "i-darkvec: Incremental embeddings for darknet traffic analysis," *ACM Trans. Internet Technol.*, vol. 23, no. 3, Aug. 2023. [Online]. Available: https://doi.org/10.1145/3595378

[12] S. Ishikawa, S. Ozawa *et al.*, "Port-piece embedding for darknet traffic features and clustering of scan attacks," in *Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 23–27, 2020, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, 2020, p. 593–603. [Online]. Available: https://doi.org/10.1007/978-3-030-63833-7_50

[13] L. McInnes, J. Healy *et al.*, "Umap: Uniform manifold approximation and projection for dimension reduction," 2020. [Online]. Available: https://arxiv.org/abs/1802.03426

[14] M. Ester, H.-P. Kriegel *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.

[15] L. Peng, X. Xie *et al.*, "Ptu: Pre-trained model for network traffic understanding," in *2024 IEEE 32nd International Conference on Network Protocols (ICNP)*, 2024, pp. 1–12.

[16] DeepSeek-AI, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," 2025. [Online]. Available: https://arxiv.org/abs/2501.12948

[17] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[18] J. Kim, S. Lee *et al.*, "Linq-embed-mistral:elevating text retrieval with improved gpt data through task-specific control and quality refinement," Linq AI Research Blog, 2024. [Online]. Available: https://getlinq.com/blog/linq-embed-mistral/

[19] "Parquet — parquet.apache.org," https://parquet.apache.org/.

[20] Y. Collet and M. Kucherawy, "Zstandard Compression and the 'application/zstd' Media Type," RFC 8878, Feb. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc8878

[21] L. Liu, G. Engelen *et al.*, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *2022 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2022, pp. 254–262.

[22] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[23] ——, "Sgdr: Stochastic gradient descent with warm restarts," 2017. [Online]. Available: https://arxiv.org/abs/1608.03983

[24] J. Ansel, E. Yang *et al.*, "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, Apr. 2024. [Online]. Available: https://pytorch.org/assets/pytorch2-2.pdf

[25] K. Enevoldsen, I. Chung *et al.*, "MMTEB: Massive multilingual text embedding benchmark," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=zl3pfz4VCV

[26] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 4765–4774.