Análisis de desempeño de redes neuronales en la clasificación de textos: IA vs. Humano

Antonio Suárez[1], Tifanny Morales[1], Ericka Saenz[1] , André Calvo[1]
jose.suarezquesada@ucr.ac.cr, tifanny.morales@ucr.ac.cr, ericka.saenz@ucr.ac.cr, andre.calvo@ucr.ac.cr

RESUMEN

El auge del texto generado por inteligencia artificial ha permitido la creación de contenidos que imitan el estilo humano con gran precisión. Es por esto que crear herramientas que logren identificar la naturaleza del texto es de suma importancia. Para esto se desarrollaron y optimizaron mediante el algoritmo Hyperband 3 modelos de redes neuronales diferentes, entre esos un modelo básico Multi-Layer Perceptron (MLP) y dos redes neuronales recurrentes (RNN) que logren clasificar oraciones según su origen. Como resultado, se obtuvieron buenas métricas para cada una de las redes donde la MLP es la red más equilibrada, y cada una de las RNN sobresalen según el tipo de texto que sea de mayor interés identificar correctamente.

PALABRAS CLAVE: Clasificación automática, evaluación de rendimiento , optimización

ABSTRACT

The rise of AI-generated text has enabled the creation of content that mimics human style with great precision. For this reason, developing tools that can identify the nature of the text is of utmost importance. To achieve this, three different neural network models were developed and optimized using the Hyperband algorithm, including a basic Multi-Layer Perceptron (MLP) model and two Recurrent Neural Networks (RNNs) designed to classify sentences based on their origin. As a result, good metrics were obtained for each network, with the MLP being the most balanced, and each RNN excelling depending on the type of text that is of greater interest to identify correctly.

KEYWORDS: Automatic classification, performance evaluation, optimization

INTRODUCCIÓN

En los últimos años, los chatbots de inteligencia artificial (IA) generativa, como Chat GPT, han experimentado un notable auge, generando debates sobre la posible sustitución de textos creados por humanos por aquellos elaborados por IA. Esta preocupación surge de casos donde los mismos bots han logrado redactar textos en un idioma específico con tal precisión que engañan a sus lectores, haciéndoles creer que dichos textos fueron escritos por una persona. No obstante, a medida que estas herramientas se popularizan, también se hacen evidentes sus limitaciones, como la falta de originalidad, la posibilidad de generar contenido tóxico, los riesgos para la privacidad, entre otros, lo que genera inquietud entre sus usuarios según (Hua, S., Jin, S. y Jiang, S. ,2024)

(Bontridder y Poullet, 2021) comentan que la proliferación del contenido generado automáticamente puede llevar a la desinformación y a la creación de noticias falsas, y además, en

[1] Estudiantes de Bachillerato en Estadística, Universidad de Costa Rica

contextos académicos y profesionales también es fundamental garantizar la autenticidad y la integridad del trabajo presentado. Debido a esto, por varias razones surge la necesidad de desarrollar herramientas que puedan identificar aquellos textos generados artificialmente, no sólo para proteger la calidad del contenido, sino que también ayuda a mantener la confianza de los usuarios que consumen información de medios digitales.

Por lo anterior mencionado, es un buen paso optar por enfoques innovadores que permitan agilizar el proceso de identificación de los textos generados artificialmente. Una posible solución a este problema es lograr clasificar el texto según su origen mediante redes neuronales artificiales. (Hassan S., 2024) utilizó un conjunto de datos de aproximadamente 100,000 oraciones para crear un modelo BERT optimizado obteniendo una precisión del 90% en lograr predecir la naturaleza de cada una de estas oraciones. De la misma forma, el objetivo de este estudio es desarrollar 3 diferentes modelos de redes neuronales para crear un sistema automatizado capaz de clasificar textos de diferentes fuentes distinguiendo entre sí es escrito por un humano o por una inteligencia artificial.

METODOLOGÍA

El conjunto de datos (Hassan S., 2024) utilizado para este análisis cuenta con un total de 105,000 oraciones en inglés, cada una etiquetada como escrita por humanos o generada por Inteligencia Artificial (IA). Cada categoría está compuesta por 52,500 oraciones lo que hace que este conjunto de datos esté completamente balanceado. Las oraciones en este conjunto de datos fueron recolectadas de diferentes fuentes para asegurar una gama diversa de temas y diferentes estilos de escritura, tal como se puede observar en el cuadro 1.

Cuadro 1

Ejemplos de oraciones dentro del conjunto de datos y sus respectivas etiquetas.

Oración	Etiqueta
The car is in our everyday use basically..	Humano - (0)
Everyone is given the same opportunity to make a difference and all votes are counted equally.	IA - (1)
It has been three straight years that cars have been banned in Colombia.	Humano - (0)
In conclusion, obtaining wisdom and experience from someone like a parent or guardian is really important.	IA - (1)
It takes major stress and anxiety of our backs.	Humano - (0)

Se inició con el preprocesamiento del texto con el objetivo de limpiar y preparar el texto para su análisis posterior. El procesamiento se encargó de: eliminar acentos en las palabras, convertir el texto en minúsculas, remover comas y puntos (conservando otros signos de puntuación), eliminación de espacios en blanco al final y al principio del texto, y también se llevó a cabo el proceso de lematización para reducir las palabras a su raíz. Además, se realizó la tokenización para dividir las

oraciones en unidades más comprensibles, seguido del padding para asegurar que todas las secuencias tengan la misma longitud.

Una vez terminado el proceso, se utilizó un tamaño de muestra de 50,000 oraciones de manera aleatoria para así dividir las oraciones en conjuntos de entrenamiento, validación y prueba, conformados por el 70%, 15% y 15% del tamaño de la muestra respectivamente. Las subsecuentes fases de construcción de las redes difieren según los modelos ya que se evaluaron diferentes formas de vectorización, arquitecturas e hiperparámetros. Se utilizó el algoritmo Hyperband (Li, L., Jamieson, K., DeSalvo, G. y Talwalkar, A., 2018) para así conseguir la mejor combinación de hiperparámetros en términos de rendimiento para la creación de los modelos. Cada uno de estos detalles está descrito más adelante.

La implementación y análisis de datos se realizó utilizando el lenguaje de programación Python, versión 3.10.12 (Python Software Foundation, 2023) y se utilizaron las siguiente bibliotecas para los diferentes procesos de manipulación de los datos y ajuste de los modelos planteados: numpy (Harris, CR, Millman, KJ, van der Walt, SJ., 2020), tensorflow (Abadi, M., Agarwal, A., Barham, P., et al., 2015), pandas (McKinney, W., 2010), matplotlib (Hunter, J. D., 2007), scikit-learn (Pedregosa, F., Varoquaux, G., Gramfort, A., 2011), keras (Chollet, F., 2015), keras-tuner (O'Malley, T., Bursztein, E., Long, J., 2019) y pillow (Clark, A. 2015).

MODELO 1 - RED MLP

La vectorización de esta red fue mediante el uso de Bag of Words (BoW) con cada oración representada como un vector basado en la frecuencia de las palabras dentro del vocabulario. La arquitectura de esta red es una MLP (Multi-Layer Perceptron) iniciando con una capa de entrada de 64 neuronas con activación ReLU y un kernel regularizador L2, seguido de una capa de dropout y una capa densa de 32 neuronas utilizando también la función de activación ReLU. La capa final tiene una sola neurona que utiliza la función de activación sigmoide, proporcionando probabilidades de pertenencia a una de las dos categorías. Además, se agregó un mecanismo de detención temprana en caso de que la pérdida en el conjunto de validación no lograse mejorar después de 10 epochs. La respectiva función de pérdida utilizada es entropía cruzada binaria.

Para asegurar el mejor rendimiento de esta red se evaluaron 3 optimizadores diferentes: RMSProp, Gradiente Descendiente Estocástico (SGD) y Adam, validando paralelamente diferentes tasas de aprendizaje para cada uno de estos. También, se evaluaron diferentes valores de penalización para la regularización L2 y probabilidades de desactivación para la capa de dropout. La mejor combinación de hiperparámetros para esta red son aquellos que minimicen la pérdida en la validación.

MODELO 2 - RED RNN (1)

En este caso la vectorización de las palabras se llevó a cabo utilizando embeddings preentrenados de GloVe (Pennington et al., 2018), obtenidos de la página web del grupo Stanford NLP. Cada palabra se mapeó en un vector de 100 dimensiones y se definió un vocabulario máximo de 10,000 palabras. Además, la longitud máxima de las secuencias de texto que se procesaron es de 100 palabras. La arquitectura de este modelo es de una red neuronal recurrente (RNN). Esta red inicia con una capa que transforma las entradas en vectores de embeddings utilizando la matriz de

embeddings preentrenada mencionada anteriormente. Seguido de una capa LSTM (Long Short-Term Memory) de 16 neuronas y un kernel regularizador con una fuerza de penalización de 0.01. La capa de salida es la misma que la red anterior; una capa de una sola neurona que utiliza la función de activación sigmoide. Su función de pérdida es entropía cruzada binaria. También se mantiene el mismo mecanismo de denteción temprana con parámetro de paciencia de 10 epochs en caso de que el rendimiento de la red no mejore.

La hiperparámetros a optimizar de esta red son similares a la red MLP, comparando los mismos 3 optimizadores (RMSProp, SGD y Adam), y su respectiva tasa de aprendizaje. Adicionalmente, se evaluó si actualizar o mantener fijos los embeddings preentrenados durante el entrenamiento mejoran el rendimiento del modelo. Similarmente, la mejor combinación de hiperparámetros y condiciones son aquellas que minimicen la pérdida en el conjunto de validación

MODELO 3 - RED RNN (2)

Para la última red diseñada se optó por utilizar embeddings no preentrenados, es decir, los embeddings fueron creados desde cero, manteniendo un vocabulario máximo de 10,000 palabras y 100 dimensiones. La arquitectura de este modelo es una red neuronal recurrente, iniciando con la capa de embedding. Luego, se utilizó una capa GRU bidireccional, permitiendo que la red capture información de las secuencias hacia adelante y hacia atrás. Se agrega una capa densa con 64 neuronas con función de activación ReLU, seguido de una capa de dropout y finalizando con su respectiva capa de salida con una sola neurona y función de activación sigmoide. Para esta red se utiliza el mismo mecanismo para detener el entrenamiento en caso de ser necesario y función de pérdida que las redes anteriores.

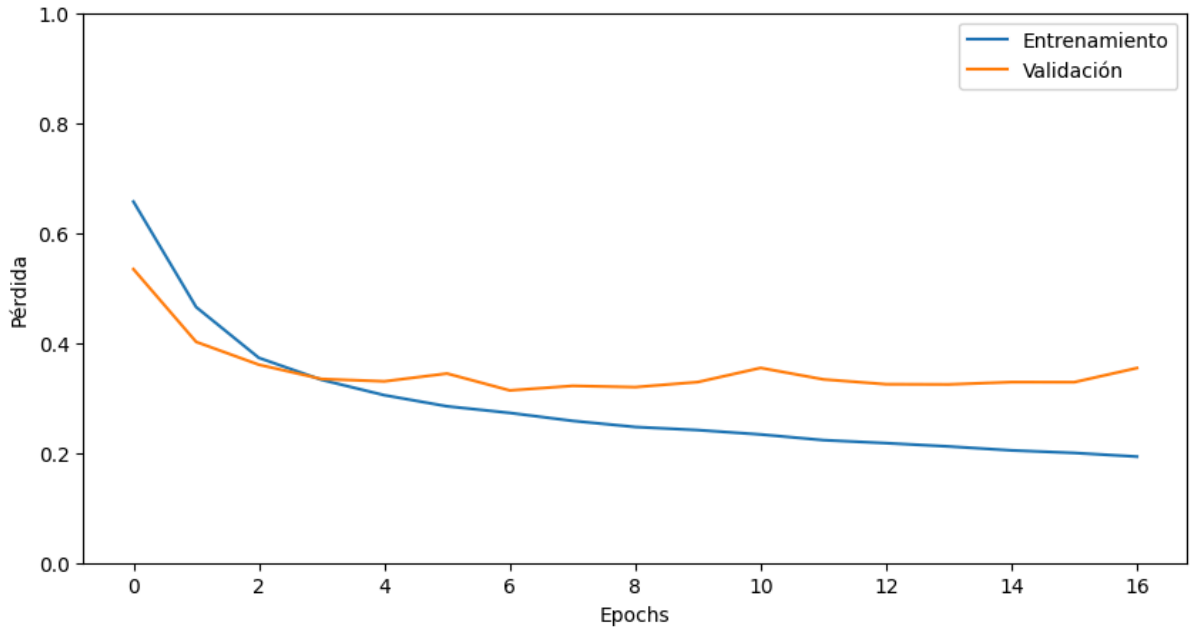
La validación de los optimizadores y tasas de aprendizaje para esta red que se llevan a cabo son solo RMSProp y Adam. También se probaron diferente cantidad de neuronas para la capa GRU bidireccional con un mínimo de 32 y un máximo de 128 con saltos de 32 unidades. Para la desactivación aleatoria de neuronas se probaron valores entre 0.2 y 0.5. Al igual que las redes anteriores, se busca la combinación de dichos hiperparámetros que minimicen la pérdida

RESULTADOS

Al analizar los diferentes valores para la red MLP, se obtuvo que el optimizador que presentó menos fluctuaciones y una mejora en la precisión fue Adam con una tasa de aprendizaje de 0.00013. Al entrenar la red, se obtuvo que la probabilidad más adecuada para la desaparición de las neuronas es de 0.2, seguido de un valor de penalización de 0.0001 definiendo la arquitectura de la red final. Observando la pérdida y precisión de esta red en la figura 1 el modelo mejora rápidamente tanto en los datos de entrenamiento como en los de validación y se obtiene una precisión sumamente alta desde el inicio, a pesar de ello, después de ciertos epochs el modelo no logra aprender más y hay un ligero sobreajuste.

Figura 1

Gráfico de la pérdida para la red MLP optimizada



Similarmente, la primera red neuronal de tipo recurrente mostró que el mejor optimizador para dicha red fue Adam con una tasa de aprendizaje de 0.007. Al evaluar si congelar los embeddings preentrenados de GloVe o dejar que se actualicen con el tiempo, se mostró que se obtuvo mejor rendimiento para esta red si se permitía que los embeddings se siguieran adaptando al conjunto de datos. El comportamiento a lo largo del entrenamiento de la red (Anexos 3, 4) fue muy similar que el de la red MLP, en los primeros epochs el modelo aprende muy rápido y su precisión es alta, pero después de un tiempo no es capaz de seguir aprendiendo.

Ahora bien, la última red al pasar por el proceso de optimización se mostró que la mejor probabilidad de desaparición de neuronas es de 0.4 para combatir el sobreajuste de los datos. La cantidad de unidades más óptima para la capa GRU bidireccional es de 32 neuronas y en términos del optimizador, el de mejor rendimiento fue el RMSProp con una tasa de aprendizaje de 0.007. El rendimiento de esta red no se aleja de las dos anteriores, obteniendo una precisión muy alta al inicio, pero sin capacidad de aprender más (Anexos 5, 6).

Una vez seleccionadas las mejores arquitecturas e hiperparámetros para cada uno de los modelos, se procedió a evaluar diferentes métricas de rendimiento utilizando el conjunto de datos de prueba. Contrastando la curva ROC para los diferentes modelos, la red MLP obtuvo un AUC de 0.94 y las dos arquitecturas de redes recurrentes obtuvieron un AUC de 0.95. Esto implica que, para los tres modelos creados, aproximadamente un 95% de las veces un texto seleccionado aleatoriamente del grupo de textos artificiales tiene mayor probabilidad para ser clasificado como texto generado por IA que como texto originado por humanos.

Por otro lado, observando otras métricas de rendimiento (ver cuadro 2) podemos concluir que la red MLP es sumamente equilibrada y consistente en todas las métricas. La primera red neuronal recurrente es excelente en capturar satisfactoriamente aquellos textos que sean generados por IA con un recall de 0.90. En contraste, la segunda red neuronal recurrente es la que tiene mayor precisión, con un valor de 0.91, dando a entender que de las tres es la que mejor logra clasificar aquel texto que realmente sea etiquetado como humano. En el cuadro 3, se pueden observar algunos casos de las diferentes predicciones que lograron realizar los diferentes modelos.

Cuadro 2

Métricas de rendimiento de los diferentes modelos optimizados

Modelo	Exactitud	Precisión	Recall	F1 Score
MLP	0.87	0.87	0.87	0.87
RNN (1)	**0.88**	0.86	**0.90**	**0.88**
RNN (2)	0.87	**0.91**	0.81	0.86

Cuadro 3

Ejemplos de las diferentes predicciones realizadas por los modelos

Oración	Etiqueta Real	Etiqueta predicha		
		MLP	RNN (1)	RNN (2)
That's why people don't get rid of their cars.	0	0	0	0
One way is to actively work on improving one's behavior and actions.	1	1	1	1
😅 I mean, , there are already so many apps and websites that can help us learn stuff.	1	0	1	1
That's how BOGOTA got started in Colombia.	0	0	0	0
Are we ready for driverless cars?	0	1	0	0

CONCLUSIONES

Los tres modelos evaluados tienen un buen desempeño, por lo que todos podrían ser útiles dependiendo del caso de uso. Si se prioriza identificar correctamente la mayor cantidad posible de textos generados por IA la primera red neuronal recurrente es la más sensible y acertada para esta situación. En el caso de que se quiera prevenir clasificar erróneamente textos humanos como IA, la segunda red neuronal recurrente es la mejor opción. En caso de que se necesite un modelo adecuado sin priorizar alguna métrica en específica la red MLP es una opción segura.

El desempeño de las redes en este problema demuestra que, al tratarse de una tarea relativamente sencilla, no fue necesario utilizar arquitecturas más complejas o con más capas. Una red MLP bien configurada logró resultados comparables a los de modelos más avanzados, lo que

indica que, en este caso, la simplicidad fue suficiente. A medida de recomendación, en problemas similares es preferible optar por modelos más parsimoniosos y solo recurrir a arquitecturas más complejas cuando el problema lo requiera.

Se sugiere que el rendimiento de los modelos podría mejorar si se incorporan más textos u oraciones de fuentes más diversas o de diferentes tipos. Esto debido a que los modelos se crearon con solo un conjunto específico de datos, y ampliar la variedad de ejemplos podría ayudar a capturar otro tipo de patrones que los modelos tal vez no lograron identificar.

BIBLIOGRAFÍA

Abadi, M., Agarwal, A., Barham, P., et al. (2015). TensorFlow: aprendizaje automático a gran escala en sistemas heterogéneos . TensorFlow. https://www.tensorflow.org/

Bontridder, N. y Poullet, Y. (2021). El papel de la inteligencia artificial en la desinformación . Datos y Políticas, 3 , e32. https://doi.org/10.1017/dap.2021.20

Chollet, F., (2015). Keras: la biblioteca de aprendizaje profundo de Python . Keras. Recuperado de https://keras.io/

Clark, A. (2015). Almohada (tenedor PIL) . https://pillo

Harris, CR, Millman, KJ, van der Walt, SJ, et al. (2020). Programación de matrices con NumPy . Nature, 585 , 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hassan, S. (2024). Sentencias humanas vs IA [Conjunto de datos]. Abrazando la cara. Recuperado de https://huggingface.co/datasets/shahxeebhassan/human_vs_ai_sentences

Hua, S., Jin, S. y Jiang, S. (2024). Limitaciones y consideraciones éticas de ChatGPT . Inteligencia de Datos, 6(1) , 201-202https://doi.org/10.1162/dint_a_00243

Hunter, JD (2007). Matplotlib: Un entorno gráfico 2D . Computación en Ciencias e Ingeniería . https://doi.org/10.1109/MCSE.2007.55

Li, L., Jamieson, K., DeSalvo, G. y Talwalkar, A. (2018). Hyperband: Un nuevo enfoque basado en Bandit para la optimización de hiperparámetros .

McKinney, W. (2010). Estructuras de datos para computación estadística en Python . *Actas deActas de la 9.ª Conferencia sobre Python en la Ciencia . https://doi.org/10.25080/Majora-92bf1922-00a

O'Malley, T., Bursztein, E., Long, J., et al. (2019). Sintonizador Keras . Sintonizador Keras. Recuperado de https://keras.io/keras_tuner/

Pennington, J., Socher, R., y Manning, CD (2014). GloVe: Vectores globales para la representación de palabras.

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Aprendizaje automático en Python . Journal of Machine Learning Research, 12 , 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

Python Software Foundation. (2023). *Python Language Reference, version 3.10.12*. https://www.python.org/

ANEXOS

Figura 2

Gráfico de la precisión para la red MLP optimizada

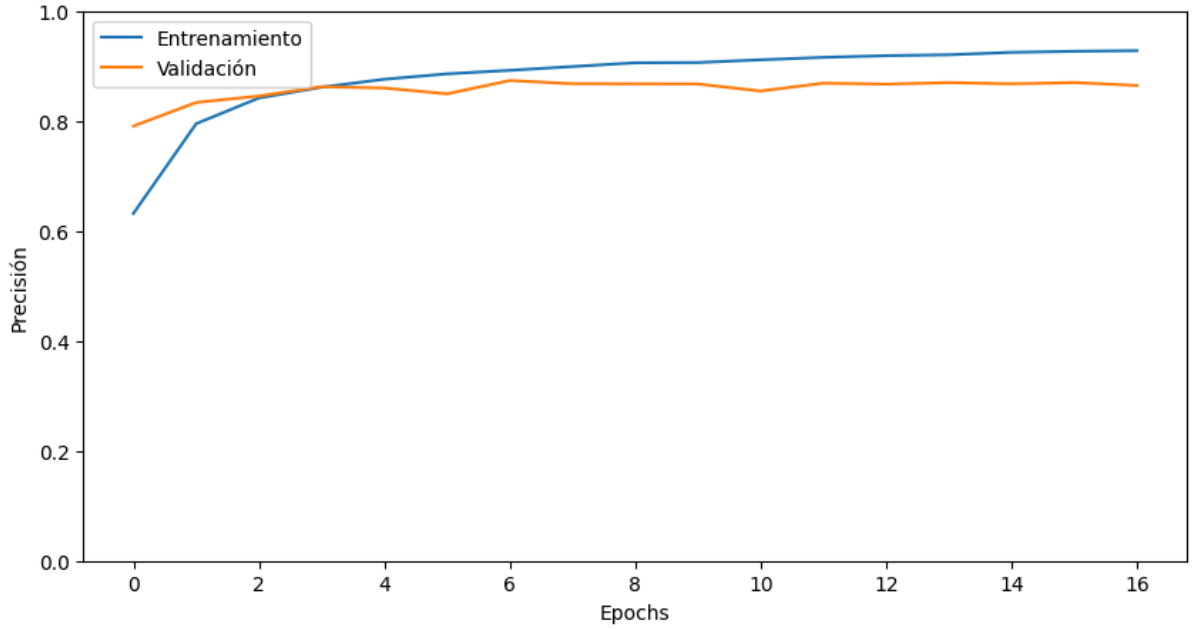


Figura 3

Gráfico de la pérdida para la primera red RNN optimizada

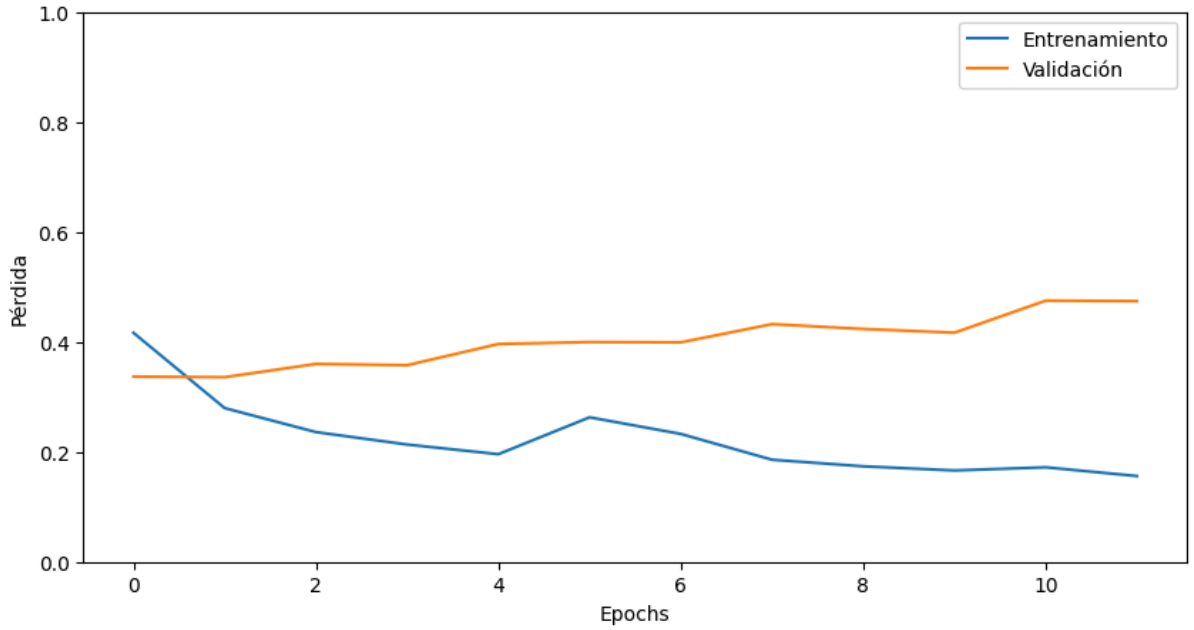


Figura 4

Gráfico de la precisión para la primera red RNN optimizada

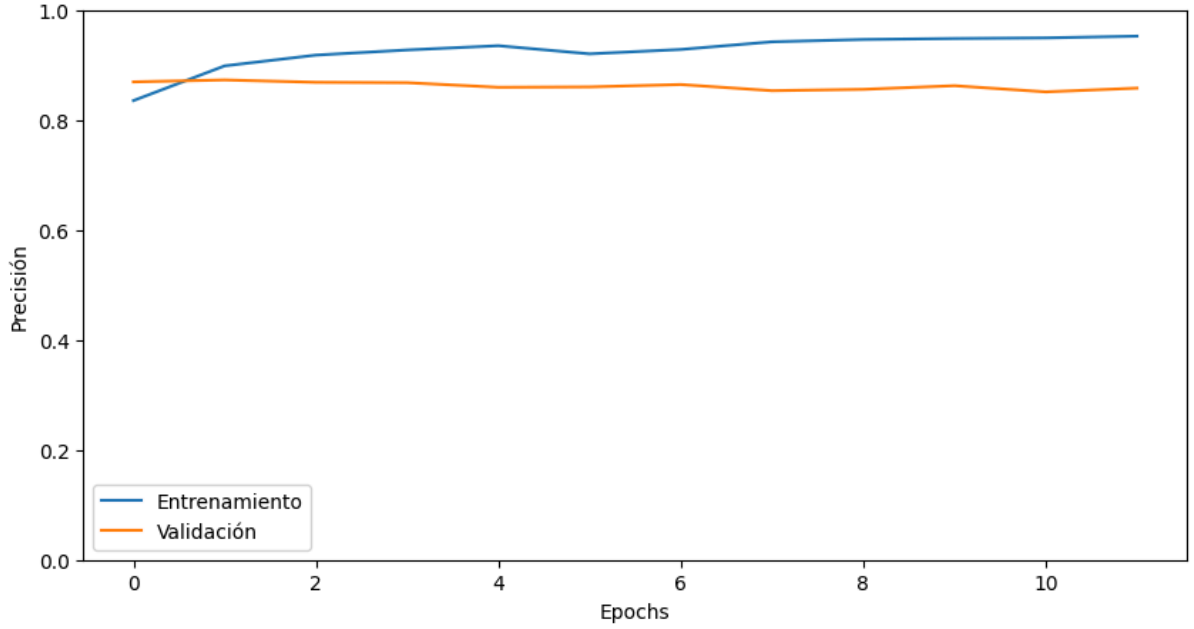


Figura 5

Gráfico de la pérdida para la segunda red RNN optimizada

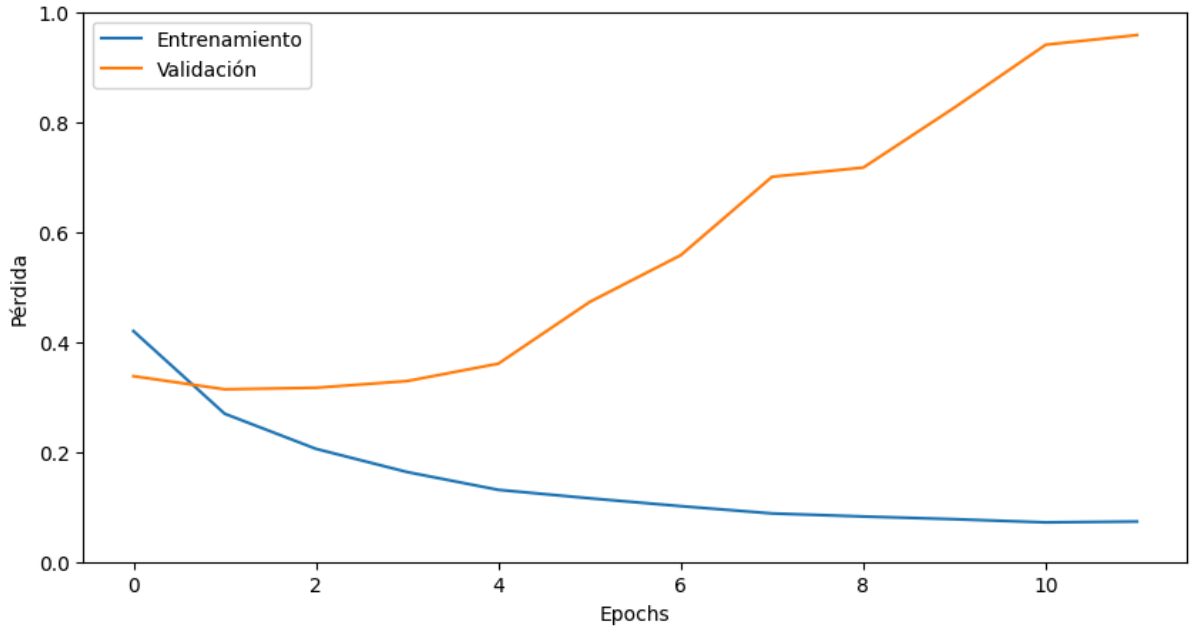


Figura 6

Gráfico de la precisión para la segunda red RNN optimizada

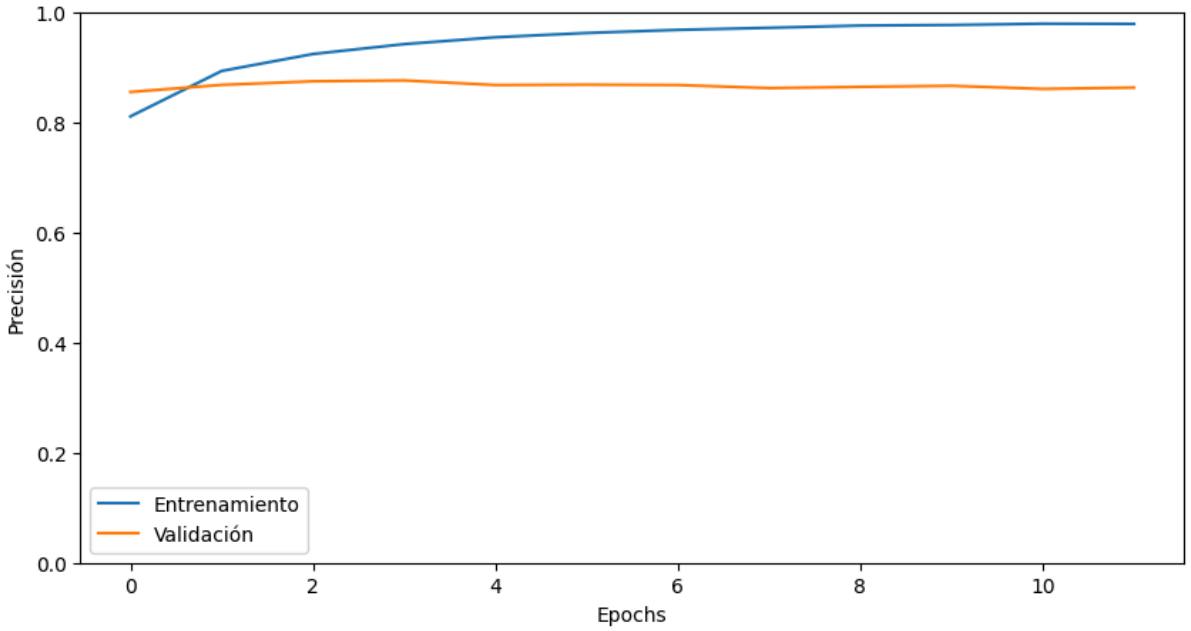


Figura 7

Curva ROC para el modelo MLP

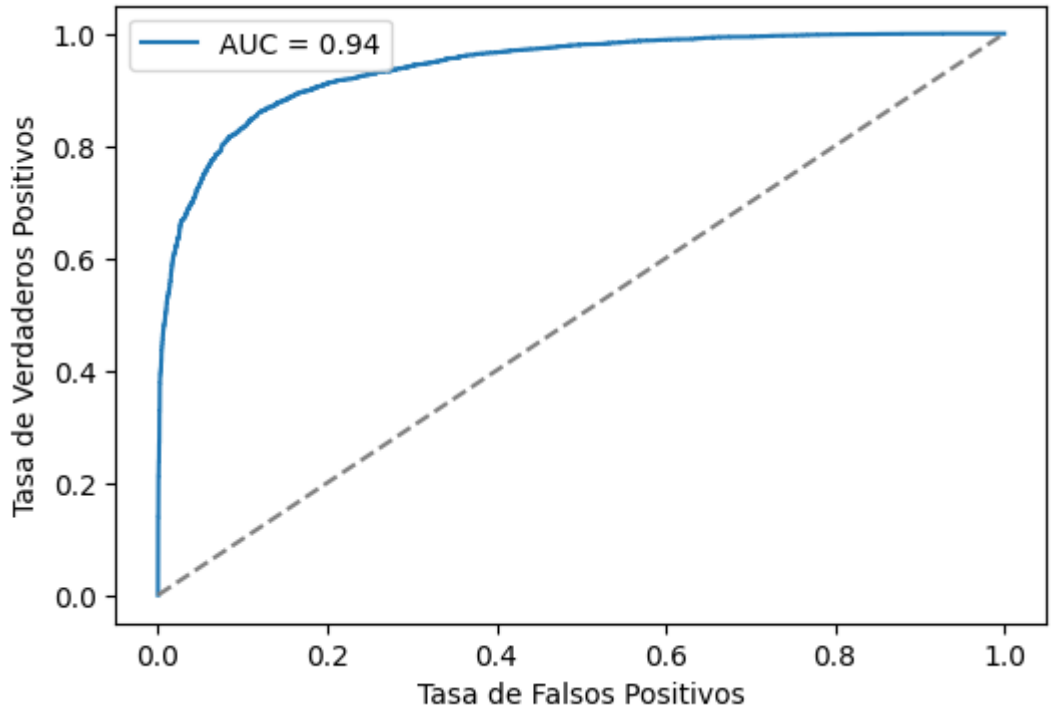


Figura 8

Curva ROC para la primera RNN

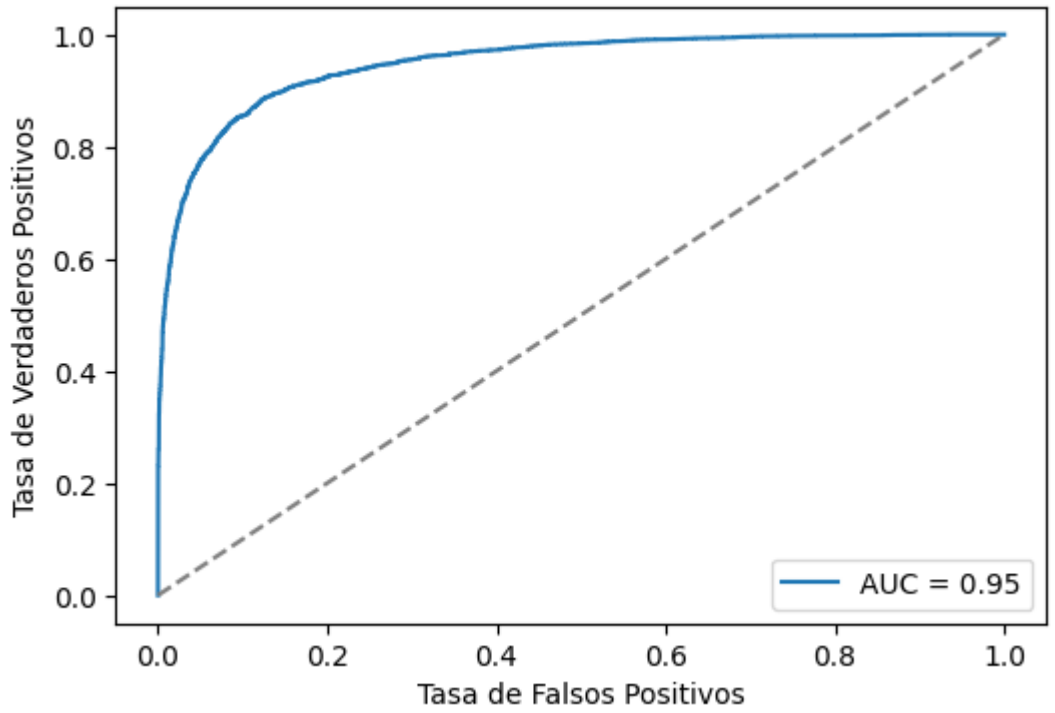


Figura 9

Curva ROC para la segunda RNN

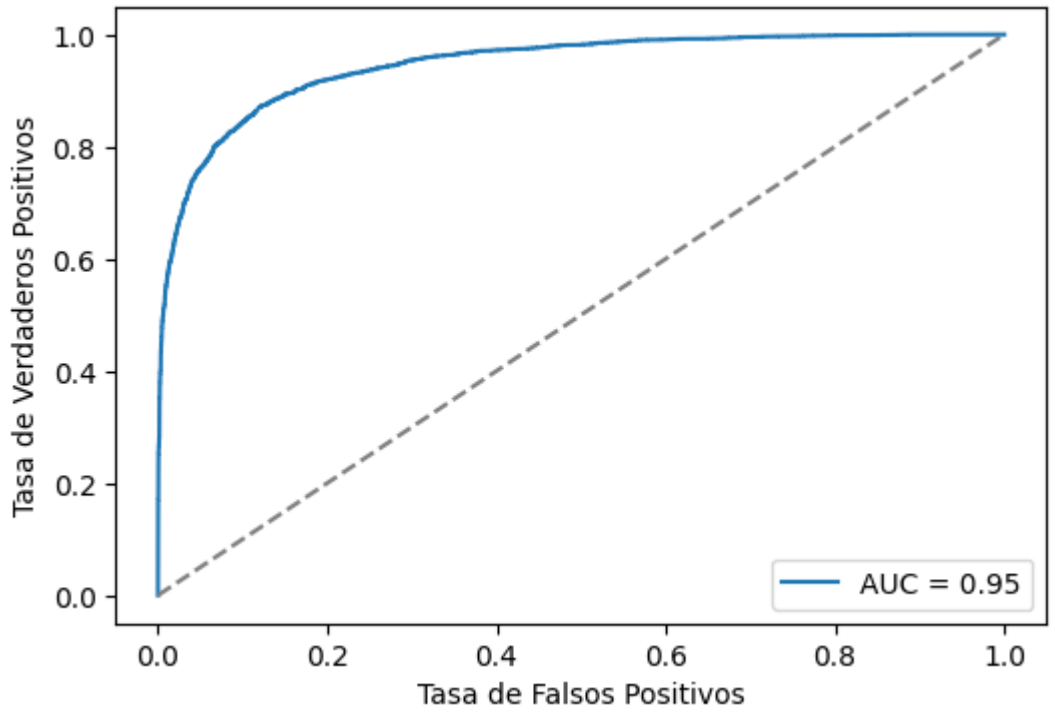


Figura 10

Matriz de confusión para la red MLP

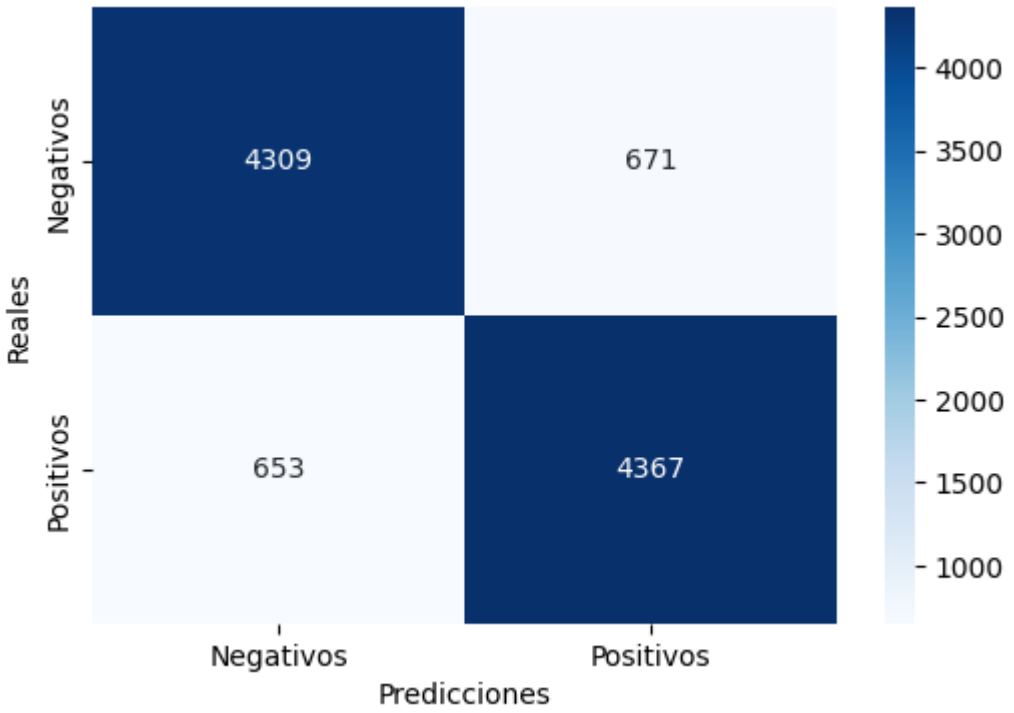


Figura 11

Matriz de confusión para la primera red RNN

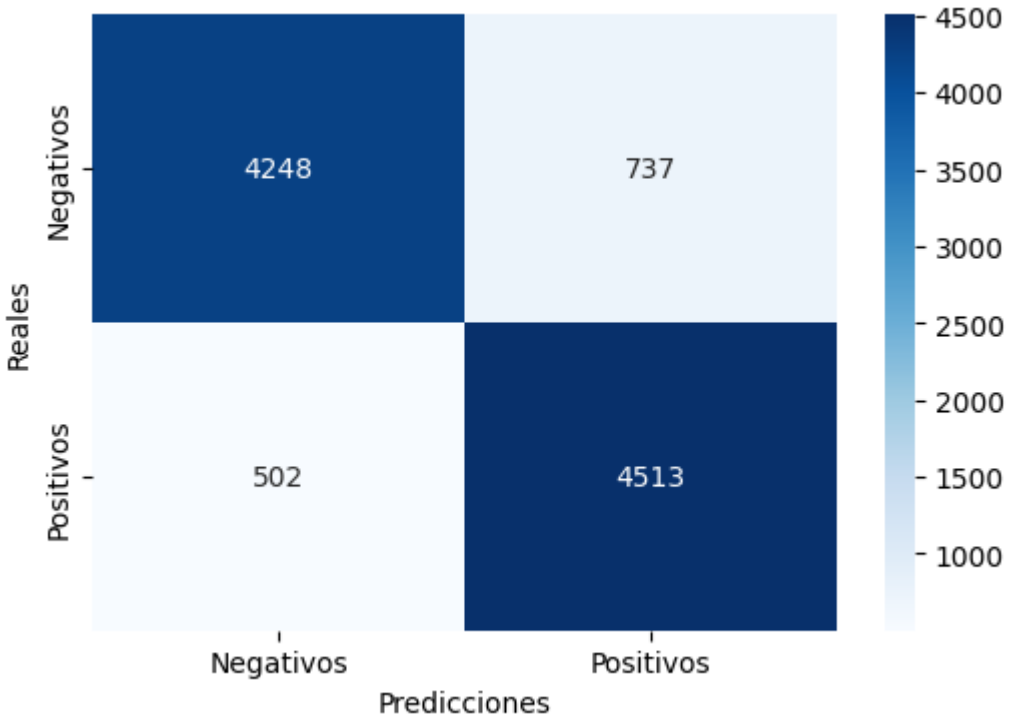


Figura 12

Matriz de confusión para la segunda red RNN