Revista Científica Multidisciplinar G-ner@ndo ISSN: 2806-5905

Análisis de Sentimientos de las Noticias en x utilizando procesamiento de Lenguaje Natural

Sentiment Analysis of News in x using Natural Language processing

David Fernando Ramos Tomalá, Mariuxi Del Carmen Toapanta Bernabé, Katty Nancy Lino Castillo, Jairo Geovanny Veintimilla Andrade & Diana Ercilia Gallegos Zurita

PUNTO CIENCIA.

julio - diciembre, V°6 - N°2; 2025

Recibido: 01-12-2025
Aceptado: 10-12-2025
Publicado: 11-12-2025

PAIS

- Ecuador, Guayaquil
- Ecuador, Guayaquil
- Ecuador, Guayaquil
- Ecuador, Guayaquil
- Ecuador, Guayaquil

INSTITUCION

- Universidad de Guayaquil
- Universidad de Guayaquil
- Universidad de Guayaquil
- Universidad de Guayaquil
- Universidad de Guayaquil

CORREO:

✉ david.ramost@ug.edu.ec
✉ mariuxi.toapantab@ug.edu.ec
✉ katty.linoc@ug.edu.ec
✉ jairo.veintimillaa@ug.edu.ec
✉ diana.gallegosz@ug.edu.ec

ORCID:

https://orcid.org/0009-0007-2702-8926
https://orcid.org/0000-0002-4839-7452
https://orcid.org/0000-0002-0345-3246
https://orcid.org/0000-0002-2841-2344
https://orcid.org/0000-0002-7319-3443

FORMATO DE CITA APA.

Ramos, D., Toapanta, M., Lino, K., Veintimilla, J. & Gallegos, D. (2025). Análisis de Sentimientos de las Noticias en x utilizando procesamiento de Lenguaje Natural. *Revista G-ner@ndo, V°6 (N°2). Pág. 3493 – 3516.*

Resumen

El presente proyecto tiene como propósito desarrollar un componente de análisis de sentimientos de las noticias en X seleccionadas por las verificadoras de hechos en Ecuador, utilizando Procesamiento de Lenguaje Natural, que sirva de aporte a una investigación a nivel macro que busca ayudar al fact-ckecker a realizar el proceso de verificación de noticias. Para realizarlo, se emplearon técnicas de recoección de datos como la entrevista, técnicas de análisis de sentimientos y modelos de aprendizaje automático. En su implementación, se utilizó la base de datos en la nube MongoDB Atlas para obtener los datos requeridos, el lenguaje de programación Python utilizando el cuaderno colaborativo Google Colaboratory y el modelo preentrenado de BERT en español como base para entrenar un modelo que pueda realizar predicciones de sentimientos y emociones. Además, se emplearon métricas para evaluar el rendimiento del modelo entrenado. Estas herramientas mencionadas contribuyeron para que se cumplan los objetivos propuestos.

Palabras clave: X, Modelo, Procesamiento, Análisis, Sentimientos.

Abstract

The purpose of this project is to develop a sentiment analysis component of X news selected by fact-checkers in Ecuador, using Natural Language Processing, which serves as a contribution to a macro-level research that seeks to help the fact-checker to perform the news verification process. To do so, data collection techniques such as interviews, sentiment analysis techniques and machine learning models were used. In its implementation, the cloud database MongoDB Atlas was used to obtain the required data, the Python programming language using the Google Colab collaborative notebook, and the pre-trained BERT model in Spanish as the basis for training a model that can perform sentiment and emotion predictions. In addition, metrics were used to evaluate the performance of the trained model. These tools contributed to the achievement of the proposed objectives.

Keywords: X, Model, Processing, Analysis, Feelings.

Introducción

En la actualidad la mayor parte de la sociedad se aferra al consumo de Internet de tal manera que ha incrementado el uso de esta gran red. Kemp (2023) indica en su informe insignia Digital 2023 que “En los últimos meses ha existido un gran cambio en el número global de usuarios de Internet”. Adicionalmente, se menciona que el número de usuarios de internet en el mundo alcanzó los 5160 millones de personas, lo que representa al 64,4% de la población mundial (Galeano, 2023).

Es importante recalcar que los internautas tienden a pasar más tiempo en las redes sociales más que en cualquier otro sitio web (Naso et al., 2012). Por lo tanto, se puede indicar que las redes sociales son llamativas para los usuarios, por lo consecuente se conectan por bastante tiempo a estas aplicaciones, lo cual puede traer consigo ventajas y desventajas. De hecho, los usuarios de redes sociales dedican al día una media de dos horas y veinticinco minutos y se prevé que para 2024 el porcentaje de internautas a nivel mundial con perfil en redes sociales supere el 82 por ciento. (Piñero, 2021, p.317)

Las redes sociales son medios que permiten la difusión masiva, ya que tienen un gran alcance e impacto en la sociedad moderna, adicionalmente, pueden ser utilizadas tanto por personas naturales como por empresas, por lo que permiten lograr una comunicación interactiva y dinámica (Hütt Herrera, 2012). Al ser medios de propagación masiva, el usuario tiene la libertad de compartir cualquier tipo de contenido, como consecuencia en muchas ocasiones se puede proliferar información errónea, desactualizada o que no es del agrado de los usuarios.

Según Gleick (2011) el aumento impresionante que han tenido los contenidos de información en Internet repercute en la posibilidad de entenderlos, contrastarlos y sobre todo de comprobar si es información verídica. De esta forma al navegar en estas plataformas de índole social, los usuarios encuentran distintos tipos de información, visualizando noticias engañosas o

falsas conocidas también como fake news. Las noticias falsas existen desde tiempos inmemorables; sin embargo, su nivel de impacto y trascendencia es en la actualidad desmedido por varios factores, uno de ellos es el alcance y uso que se le da a la tecnología y la amplificación de los contenidos de las redes sociales. (Cusot & Palacios, 2019). Por esta razón, se ha buscado garantizar que los textos informativos estén comprobados para que las personas que naveguen en Internet, especialmente en redes sociales no sean víctimas de fake news. Actualmente, a nivel mundial existen varias entidades que están autorizadas y certificadas por la International Fact-Checking Network (IFCN) para comprobar la veracidad de una noticia y Ecuador no se queda atrás, también cuenta con verificadores de hechos como Ecuador Chequea y Ecuador Verifica.

El presente estudio es parte de una investigación a nivel macro que busca ayudar al fact-ckecker a realizar el proceso de verificación de noticias, aportando con el desarrollo de un componente que realice el análisis de sentimientos en las noticias de X que son seleccionadas por las verificadoras acreditadas en Ecuador y los comentarios de los usuarios en estas noticias (Capuano et al., 2023). Se considera importante realizar este componente porque las noticias falsas son creadas para intervenir en las emociones que pueden tener los usuarios, generar miedo, incertidumbre, infamar y desequilibrar (Santamaria, 2017), esto es posible porque se hace uso de la emocionalidad como herramienta para conseguir interés por parte de los internautas en este tipo de contenido (Ecuador Chequea, 2018).

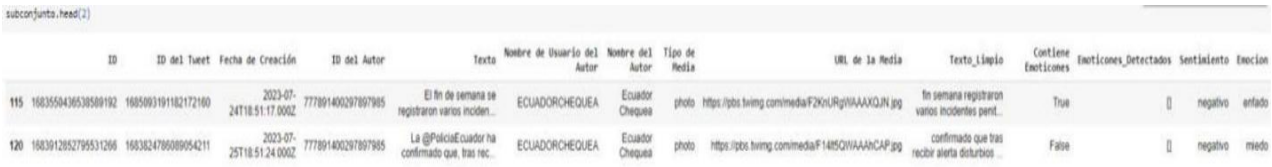
El análisis de sentimientos constituye una herramienta fundamental para comprender cómo las emociones influyen en la difusión y aceptación de las noticias, especialmente aquellas de carácter desinformativo. La identificación temprana de patrones emocionales puede aportar información relevante para los procesos de verificación, al permitir detectar contenidos potencialmente manipuladores o diseñados para generar impacto emocional en la audiencia. En este sentido, el desarrollo de un modelo basado en técnicas de PLN y en el uso de modelos preentrenados como BERT en español representa una contribución significativa al

fortalecimiento de las estrategias tecnológicas aplicadas al combate de la desinformación en el contexto ecuatoriano.

Métodos y Materiales

El modelo propuesto se basa en el modelo preentrenado de BERT en español de la librería Transformers de Hugging Face y PyTorch (Priya, 2023), en este modelo se propone identificar y etiquetar sentimientos en negativos y positivos, y etiquetar emociones como alegría, esperanza, tristeza, enfado y miedo (Romero Moreno et al., 2020). Como se muestra en la Figura 1, para proceder con el entrenamiento del modelo propuesto, se ha creado un subconjunto del conjunto de datos “df_nuevo”, el cual se denomina “subconjunto” que contiene una muestra aleatoria de 100 datos, esto se realiza con el objetivo de etiquetarlos con los sentimientos y emociones correspondientes para luego entrenar el modelo con estos datos (Liu, 2015). Este pequeño DataFrame contiene la variable “Texto_Limpio” del conjunto de datos más grande “df_nuevo”, en el mismo que ya se encuentran los textos de las noticias preprocesados.

Figura 1. *Subconjunto de datos etiquetados*



subconjunto.head(2)

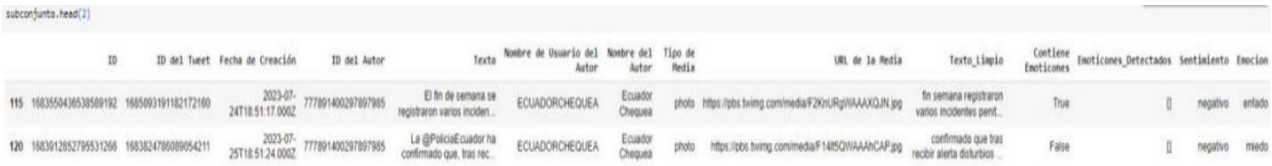
	ID	ID del Tweet	Fecha de Creación	ID del Autor	Texto	Nombre de Usuario del Autor	Nombre del Autor	Tipo de Media	URL de la Media	Texto_Limpio	Contiene Emoticones	Emoticones_Detectados	Sentimiento	Emocion
115	1683550436538589192	1685093191182172160	2023-07-24T18:51:17.000Z	777891400297897985	El fin de semana se registraron varios inciden...	ECUADORCHEQUEA	Ecuador Chequea	photo	https://pbs.twimg.com/media/F2KnURgWAAAXQJN.jpg	fin semana registraron varios incidentes penit...	True	[]	negativo	enfado
120	1683912852795531266	1683824786089054211	2023-07-25T18:51:24.000Z	777891400297897985	La @PoliciaEcuador ha confirmado que, tras rec...	ECUADORCHEQUEA	Ecuador Chequea	photo	https://pbs.twimg.com/media/F1485QIWAAAhCAP.jpg	confirmado que tras recibir alerta disturbios ...	False	[]	negativo	miedo

Nota. Sentimientos y emociones etiquetadas de una muestra aleatoria de 100 datos. Información adaptada de Google Colaboratory, 2023.

Para el entrenamiento del modelo, es necesario preparar el conjunto de datos, por esta razón, previamente se creó un conjunto de datos etiquetados para poder entrenar el modelo propuesto (Mayo, 2018). La Tokenización, que se visualiza en la Figura 2, es fundamental en el Procesamiento de Lenguaje Natural (NLP); tiene como fin convertir los textos en secuencias numéricas, conocidas como tokens, para que puedan ser entendidas en el modelo de

aprendizaje. En primer lugar, es necesario cargar el tokenizador preentrenado de BERT en español, utilizando la sentencia de código tokenizer = BertTokenizer.from_pretrained("dccuchile/bert-base-spanish-wwm-uncased"), para proceder a tokenizar los textos.

Figura 2. *Tokenización de textos en Modelo Propuest*



subconjunto.head(2)

	ID	ID del Tweet	Fecha de Creación	ID del Autor	Texto	Nombre de Usuario del Autor	Nombre del Autor	Tipo de Media	URL de la Media	Texto_Limpio	Contiene Emoticones	Emoticones_Detectados	Sentimiento	Emocion
115	1683550436538589192	1685093191182172160	2023-07-24T18:51:17.000Z	777891400297897985	El fin de semana se registraron varios inciden...	ECUADORCHEQUEA	Ecuador Chequea	photo	https://pbs.twimg.com/media/F2KnURgWAAAXQJN.jpg	fin semana registraron varios incidentes penit...	True	[]	negativo	enfado
120	1683912852795531266	1683824786089054211	2023-07-25T18:51:24.000Z	777891400297897985	La @PoliciaEcuador ha confirmado que, tras rec...	ECUADORCHEQUEA	Ecuador Chequea	photo	https://pbs.twimg.com/media/F1485QIWAAAnCAP.jpg	confirmado que tras recibir alerta disturbios ...	False	[]	negativo	miedo

Nota. Conversión de los textos en secuencias numéricas, conocidas como tokens, para que puedan ser entendidas en el modelo de aprendizaje. Información adaptada de Google Colab, 2023.

El proceso de codificación de los textos se realiza porque permite convertir los tokens en índices numéricos, para que puedan ser comprendidos por el modelo. Esto conlleva a crear un diccionario llamado “encodings” que contiene los tokens de los textos, con opción a truncado en caso de que los textos superen la longitud máxima permitida que es de 40 tokens, respecto a esta longitud, es recomendable que se corrobore previamente el número de tokens del texto más largo del conjunto de datos. Además, como se muestra en la Figura 3, se utiliza el parámetro padding, lo que indica que, si existen textos que sean más cortos que la longitud máxima definida, se deban agregar tokens de relleno a estos textos.

Figura 3. *Codificación de textos en Modelo Propuesto*

```
# Codificar los textos
encodings = tokenizer(input_texts, truncation=True, padding=True, max_length=40)
```

Nota. Creación de un diccionario llamado “encodings” que contiene los tokens de los textos. Información adaptada de Google Colab, 2023.

Del mismo modo, como se refleja en la Figura 4, las etiquetas referentes a los sentimientos y emociones se deben convertir en índices numéricos, los que se guardan en dos listas “sentimiento_label_ids” y “emocion_label_ids” para entrenar y evaluar el modelo de aprendizaje automático.

Figura 4. *Conversión de etiquetas a números*

```
# Convertir las etiquetas a números
sentimiento_label2id = {label: idx for idx, label in enumerate(set(sentimiento_labels))}
emocion_label2id = {label: idx for idx, label in enumerate(set(emocion_labels))}
sentimiento_label_ids = [sentimiento_label2id[label] for label in sentimiento_labels]
emocion_label_ids = [emocion_label2id[label] for label in emocion_labels]
```

Nota. Conversión en índices numéricos de las etiquetas referentes a los sentimientos y emociones. Información adaptada de Google Colab, 2023.

Por otra parte, se deben convertir las variables resultantes del paso anterior en tensores de PyTorch para sustentar al modelo durante la etapa de entrenamiento y la evaluación, lo cual se plasma en la Figura 5. Las entradas input_ids y attention_mask hacen referencia a las secuencias tokenizadas y sus máscaras de atención, a diferencia de las etiquetas de sentimiento y emoción que se utilizan como objetivos para calcular la pérdida y medir el rendimiento del modelo.

Figura 5. *Creación de tensores de PyTorch*

```
# Crear tensores de PyTorch a partir de los encodings
input_ids = torch.tensor(encodings["input_ids"])
attention_mask = torch.tensor(encodings["attention_mask"])
sentimiento_labels = torch.tensor(sentimiento_label_ids)
emocion_labels = torch.tensor(emocion_label_ids)
```

Nota. Input_ids, attention_mask refieren a secuencias tokenizadas; sentimiento, emoción se usan para calcular pérdida y rendimiento del modelo. Información adaptada de Google Colab, 2023.

Para dividir los datos en conjuntos de entrenamiento y prueba, como se representa en la Figura 6, se ha utilizado la función “train_test_split” de Scikit-learn, como explica Pedregosa et

al. (2011), en este sentido la división se realiza en las secuencias tokenizadas y sus máscaras de atención en las etiquetas de sentimiento y emoción. Para determinar la proporción del conjunto de datos para prueba, se ha utilizado el parámetro “test_size” con un valor del 20% de los datos, con una aleatoriedad de la división de 42 (Sahagian, 2020), siendo un valor fijo y garantizando que la división sea reproducible, de esta manera se obtendrán los mismos resultados en cada ejecución. Los datos de entrenamiento se usarán para ajustar el modelo, y los datos de prueba se utilizarán para evaluar su rendimiento. Esta división es fundamental para evaluar la capacidad de generalización del modelo a datos no vistos durante el entrenamiento.

Figura 6. *División en conjunto de entrenamiento y prueba*

```
# Dividir los datos en conjuntos de entrenamiento y prueba
train_input_ids, test_input_ids, train_attention_mask, test_attention_mask, train_sentimiento_labels, test_sentimiento_labels, train_emocion_labels, test_emocion_labels = train_test_split(
    input_ids, attention_mask, sentimiento_labels, emocion_labels, test_size=0.2, random_state=42
)
```

Nota. Para determinar la proporción del conjunto de datos para prueba, se ha utilizado el parámetro “test_size” con un valor del 20% de los datos, con una aleatoriedad de la división de 42. Información adaptada de Google Colab, 2023.

Estos conjuntos de datos se crean en formato PyTorch haciendo uso de la clase TensorDataset de la librería “torch.utils.data”. Como se percibe en la Figura 7, se crea un conjunto de datos de entrenamiento denominado “train_dataset” con cuatro tipos de datos, “train_input_ids” que se refiere a las secuencias tokenizadas, “train_attention_mask” que son las máscaras de atención que corresponden a las secuencias, “train_sentimiento_labels” las etiquetas de sentimiento para las entradas y “train_emocion_labels” que respecta a las etiquetas de emoción para las entradas. Por otro lado, se crea el conjunto de datos de prueba llamado “test_dataset”, en el que de manera similar al conjunto de datos de entrenamiento se tienen los mismos tipos de datos, pero para prueba.

Figura 7. *Creación de conjuntos de datos de entrenamiento y prueba*

```
# Crear conjuntos de datos
train_dataset = TensorDataset(train_input_ids, train_attention_mask, train_sentimiento_labels, train_emocion_labels)
test_dataset = TensorDataset(test_input_ids, test_attention_mask, test_sentimiento_labels, test_emocion_labels)
```

Nota. Creación de datos de entrenamiento y prueba: “train_dataset” con cuatro tipos de datos: “train_input_ids”, “train_attention_mask”, “train_sentimiento_labels”, y “train_emocion_labels”. Información adaptada de Google Colab, 2023.

Después de la creación de los conjuntos de datos de entrenamiento y prueba, es indispensable cargar el modelo preentrenado de BERT para el análisis de sentimientos y emociones utilizando la arquitectura "bert-base-spanish-wwm-uncased". Por consiguiente, y como se muestra en la Figura 8, se crea una instancia del modelo preentrenado “BertForSequenceClassification”, utilizando la función “from_pretained” con el fin de cargar los pesos preentrenados del modelo utilizado.

Figura 8. *Definición del modelo preentrenado de BERT*

```
# Definir el modelo preentrenado de BERT para análisis de sentimientos y emociones
model = BertForSequenceClassification.from_pretrained("dccuchile/bert-base-spanish-wwm-uncased", num_labels=len(sentimiento_label2id))
```

```
Downloading pytorch_model.bin: 100% 440M/440M [00:20<00:00, 22.6MB/s]
```

Nota. Creación de una instancia del modelo preentrenado “BertForSequenceClassification”, utilizando la función “from_pretained” con el fin de cargar los pesos preentrenados del modelo utilizado. Información adaptada de Google Colab, 2023.

La selección de parámetros e hiperparámetros en un modelo de aprendizaje automático con Procesamiento de Lenguaje Natural (PLN), es relevante porque controlan algunos aspectos del proceso de entrenamiento y la inferencia dentro del modelo, es por esta razón, que es necesario realizar una elección adecuada para obtener resultados óptimos en la tarea del análisis de sentimientos. Los hiperparámetros se establecen antes de comenzar el proceso de entrenamiento del modelo, es decir que son valores que no son aprendidos directamente del conjunto de datos como los parámetros. En este caso, y como se visualiza en la Figura 9, se han

implementado dos hiperparámetros importantes, el “batch_size” que representa el tamaño del lote, esto se refiere al número de ejemplos de entrenamiento que se utilizarán en cada interacción durante el entrenamiento, en este caso se utilizará un tamaño de lote de 20. En el mismo tema de hiperparámetros, se tiene el “num_epochs”, es decir el número de veces que el modelo recorrerá todo el conjunto de datos de entrenamiento mientras se esté entrenando, en este caso se trabajará con 5 épocas, es importante definir este hiperparámetro porque controla la cantidad de veces que el modelo ajusta sus pesos en función de los datos de entrenamiento.

Figura 9. *Hiperparámetros de entrenamiento*

```
# Definir hiperparámetros de entrenamiento
batch_size = 20  # Tamaño del lote
num_epochs = 5  # Número de epochs o épocas que se va a repetir el proceso de entrenamiento
```

Nota. “Batch_size” son los ejemplos que se utilizarán en cada interacción durante el entrenamiento. “Num_epochs” son las veces que el modelo recorrerá todo el conjunto de datos de entrenamiento mientras se esté entrenando. Información adaptada de Google Colab, 2023.

Es importante crear los cargadores de datos para el conjunto de entrenamiento y el conjunto de prueba, ya que forman parte del flujo de trabajo en PyTorch; esto se vislumbra en la Figura 10. Los cargadores de datos van a permitir iterar de una forma eficiente sobre los datos de entrenamiento y prueba en lotes y esto es fundamental para entrenar el modelo y por consiguiente las evaluaciones.

Figura 10. *Creación de cargadores de datos*

```
# Crear cargadores de datos
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

Nota. Creación de los cargadores de datos para el conjunto de entrenamiento y el conjunto de prueba. Información adaptada de Google Colab, 2023.

Para definir la función de pérdida o criterio, como se puede ver en la Figura 11, se ha utilizado la función “CrossEntropyLoss”, que se utiliza regularmente en los problemas de

clasificación, en el presente caso, se utiliza para medir la discrepancia entre las predicciones del modelo y las etiquetas verdaderas conocidas como “ground truth”, con esto se puede guiar el proceso de ajuste de los parámetros del modelo. En el presente entrenamiento, se ha utilizado el optimizador AdamW que pretende evitar el sobreajuste, además se utiliza el “model.parameters()” que indica cuáles son los parámetros que deben ser actualizados durante el entrenamiento, determinando la tasa de aprendizaje para poder controlar cuánto se ajusta los parámetros en función de la magnitud del gradiente, de esta manera se afina el rendimiento del modelo.

Figura 11. *Función de pérdida y optimizador*

```
# Definir función de pérdida y optimizador
criterion = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.AdamW(model.parameters(), lr=2e-5)
```

Nota. En entrenamiento se ha utilizado el optimizador AdamW para evitar el sobreajuste, además de “model.parameters()” que indica cuáles son los parámetros a actualizar durante el entrenamiento. Información adaptada de Google Colab, 2023.

El proceso de entrenamiento del modelo se visualiza en la Figura 12. Comienza con la iteración a través de las épocas o epochs, es decir que se repite el proceso de entrenamiento por el número determinado de épocas, luego se debe poner el modelo en modo de entrenamiento, con el fin de que se habiliten capas, tales como: dropout y batch normalization para que actúen acorde al entrenamiento. En este proceso se itera por medio de los lotes de datos de entrenamiento, posteriormente se procede con la optimización, se restablecen los gradientes del optimizador a cero para evitar acumulación en iteraciones previas, para las salidas se pasa un lote de datos de entrenamiento al modelo y se obtienen las predicciones, luego se calcula la pérdida entre las predicciones del modelo y las etiquetas verdaderas para ese lote. Además, se calculan los gradientes de la pérdida en contexto de parámetros del modelo y se actualizan los parámetros del modelo.

Figura 12. *Entrenamiento del modelo*

```
# Entrenar el modelo
for epoch in range(num_epochs):
    model.train()
    total_loss = 0
    for batch in train_loader:
        input_ids, attention_mask, sentimiento_labels, emocion_labels = batch
        optimizer.zero_grad()
        outputs = model(input_ids, attention_mask=attention_mask, labels=sentimiento_labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f"Epoch {epoch+1}, Loss: {total_loss / len(train_loader)}")
```

Nota. Iterar los lotes de datos de entrenamiento; optimizar; restablecer gradientes a cero; para salidas, pasar un lote de datos de entrenamiento al modelo y predecir; calcular la pérdida entre predicciones y etiquetas verdaderas para el lote. Información adaptada de Google Colab, 2023.

Para finalizar el proceso de entrenamiento, se acumula la pérdida total en la variable "total_loss" y al final de cada época se imprime el promedio de la pérdida a lo largo de todos los lotes de entrenamiento. Como se puede observar en la Figura 13, al completar la quinta época de entrenamiento se obtiene como resultado una pérdida promedio de 2.3856729285398615e-06. Por esta razón, se puede corroborar que el modelo se está ajustando de forma precisa a los datos de entrenamiento y está logrando relacionarse de manera correcta con las etiquetas verdaderas, cabe recalcar que para asegurarse que el modelo tiene un buen rendimiento es importante evaluarlo y revisar sus métricas, tales como precisión, entre otras.

Figura 13. *Pérdida promedio del modelo*

```
Epoch 1, Loss: 1.1531936934261466e-05
Epoch 2, Loss: 7.069084063004993e-06
Epoch 3, Loss: 4.798161739927309e-06
Epoch 4, Loss: 3.36765646125059e-06
Epoch 5, Loss: 2.3856729285398615e-06
```

Nota. Ya en la 5ta. época de entrenamiento se obtiene de pérdida: 2.38e-06, corroborando que el modelo se está ajustando preciso a los datos de entrenamiento, relacionándose de manera correcta con las etiquetas verdaderas. Información adaptada de Google Colab, 2023.

La evaluación es una fase que se debe realizar después de entrenar un modelo, esto se realiza con el objetivo de verificar el rendimiento real del modelo y corroborar si el modelo generaliza bien a nuevos ejemplos. Para evaluar el modelo entrenado, como se muestra en la Figura 14, en primer lugar, se debe colocar el modelo en modo evaluación para que no realice cambios en los pesos durante la evaluación, luego se utilizan las listas "sentimiento_predicted_labels" y "emocion_predicted_labels" para almacenar las etiquetas predichas por el modelo, y para almacenar las etiquetas verdaderas se utilizan las listas "true_sentimiento_labels" y "true_emocion_labels". Es importante mantener un seguimiento del número total de predicciones correctas para los sentimientos y emociones y llevar un registro del número total de ejemplos en el conjunto prueba. Al finalizar la evaluación, se tiene el número total de predicciones correctas para sentimientos y emociones, de la misma forma se tiene el número total de ejemplos en el conjunto de prueba.

Figura 14. *Evaluación del modelo*

```
# Evaluar el modelo
model.eval()
sentimiento_predicted_labels = []
emocion_predicted_labels = []
true_sentimiento_labels = []
true_emocion_labels = []
correct_sentimiento = 0
correct_emocion = 0
total = 0
with torch.no_grad():
    for batch in test_loader:
        input_ids, attention_mask, sentimiento_labels, emocion_labels = batch

        # Predicciones para sentimientos
        sentimiento_outputs = model(input_ids, attention_mask=attention_mask)
        sentimiento_predicted_batch_labels = torch.argmax(sentimiento_outputs.logits, dim=1)
        sentimiento_predicted_labels.extend(sentimiento_predicted_batch_labels.cpu().numpy())
        true_sentimiento_labels.extend(sentimiento_labels.cpu().numpy())
        correct_sentimiento += (sentimiento_predicted_batch_labels == sentimiento_labels).sum().item()

        # Predicciones para emociones
        emocion_outputs = model(input_ids, attention_mask=attention_mask)
        emocion_predicted_batch_labels = torch.argmax(emocion_outputs.logits, dim=1)
        emocion_predicted_labels.extend(emocion_predicted_batch_labels.cpu().numpy())
        true_emocion_labels.extend(emocion_labels.cpu().numpy())
        correct_emocion += (emocion_predicted_batch_labels == emocion_labels).sum().item()
        total += sentimiento_labels.size(0)
```

Nota. Modo evaluación para no cambios en pesos; "sentimiento_predicted_labels", "emocion_predicted_labels" para predicciones; etiquetas verdaderas: "true_sentimiento_labels", "true_emocion_labels". Información adaptada de Google Colab, 2023.

Posteriormente, para facilitar la comparación y generación de métricas de evaluación, se convierten las listas de etiquetas de predicción y etiquetas verdaderas en matrices NumPy, como se ve en la Figura 15. Las métricas son necesarias para poder evaluar el modelo entrenado, entre las métricas utilizadas se tienen las siguientes: La precisión: Es una métrica utilizada para evaluar el rendimiento de un modelo de clasificación y se calcula dividiendo la cantidad de predicciones correctas entre el total de ejemplos, con la sentencia de código accuracy = correct / total.

Figura 15. *Precisión general del modelo*

```
# Calcular el accuracy
accuracy = correct / total
print(f"Accuracy o precisión general en el conjunto de prueba: {accuracy:.2f}")

Accuracy o precisión general en el conjunto de prueba: 0.70
```

Nota. Precisión es una métrica utilizada para evaluar el rendimiento de un modelo de clasificación y se calcula dividiendo la cantidad de predicciones correctas entre el total de ejemplos. Información adaptada de Google Colab, 2023.

Teniendo como resultado una precisión general de 0.70 en el conjunto de prueba, lo que significa que el modelo ha realizado predicciones correctas para aproximadamente el 70% de los ejemplos. Por lo tanto, este valor de precisión indica que el modelo está acertando de manera correcta aproximadamente el 70% de los casos en el conjunto de prueba.

La precisión, recall y F1-score para sentimientos: Aunque se tiene un modelo con un 70% de precisión general, en la Figura 16 se ha realizado un reporte de clasificación para la etiqueta sentimiento con la sentencia de código.

Figura 16. *Calcular métricas en sentimientos*

```
# Calcular el classification report para sentimiento
sentimiento_report = classification_report(true_sentimiento_labels, sentimiento_predicted_labels,
                                           target_names=sentimiento_label2id.keys())
```

Nota. Reporte de clasificación para la etiqueta sentimiento con la sentencia de código. Información adaptada de Google Colab, 2023.

En la Figura 17, en donde se evalúan las métricas de precisión, recall y F1-score, respecto a la precisión indica la proporción de predicciones positivas que son realmente positivas, para la clase "positivo", la precisión es 0.60, lo que significa que del total de predicciones que el modelo hizo como "positivo", aproximadamente el 60% de ellas eran correctas. El recall o sensibilidad, hace referencia a la proporción de ejemplos reales de la clase que el modelo ha identificado correctamente, para la clase "positivo", el recall es 1.00, lo que significa que el modelo ha identificado todos los ejemplos reales de la clase "positivo". En cuanto al F1-score, combina la precisión y recall, brindando una medida equilibrada del rendimiento del modelo, para la clase "positivo", el F1-score es 0.75, lo que indica que el modelo tiene un buen equilibrio entre precisión y recall para esta clase.

Figura 17. *Reporte de clasificación para sentimientos*

Classification Report para Sentimiento:

	precision	recall	f1-score	support
positivo	0.60	1.00	0.75	6
negativo	1.00	0.71	0.83	14
accuracy			0.80	20
macro avg	0.80	0.86	0.79	20
weighted avg	0.88	0.80	0.81	20

Nota. De predicciones, aproximadamente el 60% de ellas eran correctas. Recall: 1.00, el modelo ha identificado todos los ejemplos reales de la clase "positivo". F1-score: 0.75, un buen equilibrio entre precisión y recall. Información adaptada de Google Colab, 2023.

La precisión, recall y F1-score para emociones: De la misma manera que para los sentimientos, se visualiza en la Figura 18 que en las emociones también se hace un reporte de clasificación para la etiqueta emoción con la siguiente sentencia de código.

Figura 18. *Calcular métricas en emociones*

```
# Calcular el classification report para emocion
emocion_report = classification_report(true_emocion_labels, emocion_predicted_labels,
                                       labels=list(emocion_label2id.values()),
                                       target_names=emocion_label2id.keys())
```

Nota. Reporte de clasificación para la etiqueta emoción con la siguiente sentencia de código. Información adaptada de Google Colab, 2023.

En este caso, la precisión varía para los diferentes tipos de emociones, se puede observar que para la emoción “tristeza”, la precisión es de 0.40, lo que significa que el 40% de las predicciones de este tipo son correctas, es importante mencionar que en el presente estudio se asocia más a los sentimientos negativos, por lo que es un modelo factible para lo que se requiere.

Figura 19. *Reporte de clasificación para emociones*

Classification Report para Emoción:

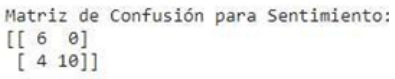
	precision	recall	f1-score	support
esperanza	0.20	1.00	0.33	2
tristeza	0.40	0.80	0.53	5
miedo	0.00	0.00	0.00	5
enfado	0.00	0.00	0.00	4
alegría	0.00	0.00	0.00	4
accuracy			0.30	20
macro avg	0.12	0.36	0.17	20
weighted avg	0.12	0.30	0.17	20

Nota. La precisión varía para los diferentes tipos de emociones: para la emoción “tristeza”, la precisión es de 0.40, lo que significa que el 40% de las predicciones de este tipo son correctas. Información adaptada de Google Colab, 2023.

Respecto al recall o sensibilidad, para el tipo “esperanza”, el recall es 1.00, lo que significa que el modelo ha identificado correctamente todos los ejemplos reales de la clase "esperanza" y para “tristeza” el recall es 0.80, indicando que el modelo ha identificado el 80% de los ejemplos reales de la clase "tristeza". En cuanto al F1-score, para “esperanza” es 0.33, lo que indica un equilibrio razonable entre precisión y recall para esta clase y para “tristeza” es 0.53, indicando un mejor equilibrio que la clase "esperanza".

Para realizar la matriz de confusión, mostrada en la Figura 20, se ha utilizado “confusion_matrix” de Sklearn para visualizar el rendimiento del modelo al comparar las etiquetas reales de las muestras con las predichas por el modelo.

Figura 20. *Matriz de confusión para sentimiento*



Nota. Visualización del rendimiento del modelo comparando las etiquetas reales de las muestras con las predichas por el modelo. Información adaptada de Google Colab, 2023.

En la matriz de confusión respecto a las predicciones de los sentimientos, se tienen 6 verdaderos positivos, lo que indica que el modelo identificó 6 casos de sentimiento “positivo” de manera correcta. En cuanto a los falsos positivos, el valor es 0, por lo que se interpreta que no hubo casos donde el modelo haya errado al predecir “negativo”. Respecto a los verdaderos negativos, se observa un valor de 10, es decir que el modelo identificó 10 casos de sentimiento “negativo” correctamente.

En cuanto a la matriz de confusión para las emociones, que se muestra en la Figura 21, contiene cinco filas que son representadas por las emociones de la siguiente manera: Fila 1: Representa la clase "esperanza"; Fila 2: Representa la clase "alegría"; Fila 3: Representa la clase "tristeza"; Fila 4: Representa la clase "enfado"; Fila 5: Representa la clase "miedo". Entre las representaciones más relevantes, se tienen: Fila 1: Tiene 2 en la columna 1, que hace referencia a la esperanza y ceros en las demás columnas, es decir que hay 2 muestras de la clase esperanza que fueron clasificadas correctamente como esperanza (verdaderos positivos) y ninguna fue clasificada como ninguna de los otros tipos de emociones. Fila 2: Representa la emoción "alegría": Tiene 1 en la columna 1 (esperanza) y 4 en la columna 2 (alegría), lo que significa que hay 1 muestra de "esperanza" clasificada como "esperanza" y 4 muestras de "alegría" clasificadas como "alegría" (verdaderos positivos). No hubo muestras clasificadas como ninguna de las otras clases.

Figura 21. *Matriz de confusión para emoción*

```
Matriz de Confusión para Emoción:
[[2 0 0 0 0]
 [1 4 0 0 0]
 [2 3 0 0 0]
 [1 3 0 0 0]
 [4 0 0 0 0]]
```

Nota. Fila 1: Representa la clase "esperanza"; Fila 2: Representa la clase "alegría"; Fila 3: Representa la clase "tristeza"; Fila 4: Representa la clase "enfado"; Fila 5: Representa la clase "miedo". Información adaptada de Google Colab, 2023.

Análisis de Resultados

Para poder corroborar los resultados del presente análisis de sentimientos, es necesario guardar el modelo entrenado, por consiguiente, como se ve en la Figura 22, se ha utilizado Google Drive para almacenar el modelo dentro de un objeto denominado “ruta_modelos_entrenados” y el modelo tiene por nombre “modelo_propuesto_sentimientos_emociones”.

Figura 22. *Almacenamiento del modelo entrenado*

```
# Guardar el modelo
ruta_modelo_guardado = f"{ruta_modelos_entrenados}modelo_propuesto_sentimientos_emociones"
model.save_pretrained(ruta_modelo_guardado)
```

Nota. Almacenamiento del modelo dentro de: “ruta_modelos_entrenados”; el modelo tiene por nombre: “modelo_propuesto_sentimientos_emociones”. Información adaptada de Google Colab, 2023.

Posteriormente, y como se muestra en la Figura 23, para realizar las predicciones o inferencias de los sentimientos y emociones, se debe cargar el modelo entrenado definiendo la ruta en la que se almacenó el modelo, utilizando la clase “BertForSequenceClassification” del framework Transformers de Hugging Face.

Figura 23. *Carga del modelo entrenado*

```
# Definir la ruta del modelo guardado
ruta_modelo_guardado = "/content/drive/My Drive/modelos_entrenados_colab/modelo_propuesto_sentimientos_emociones"

# Cargar el modelo previamente entrenado
modelo_cargado = BertForSequenceClassification.from_pretrained(ruta_modelo_guardado)
```

Nota. Definición de la ruta en la que se almacenó el modelo, utilizando la clase "BertForSequenceClassification" del framework Transformers de Hugging Face. Información adaptada de Google Colab, 2023.

Antes de utilizar el modelo previamente entrenado, es necesario preparar los datos de entrada, como se visualiza en la Figura 24, en este caso los textos de las noticias seleccionadas en X por las verificadoras de hecho en Ecuador deben encontrarse preprocesados, posteriormente se debe cargar el tokenizador de Bert con el modelo preentrenado de Bert en español para tokenizar y codificar los textos, los cuales se han almacenado en el objeto "textos_entrada".

Figura 24. *Tokenización y codificación de textos de entrada para las predicciones*

```
# Tokenizar y codificar los textos de entrada
encodings_textos_entrada = tokenizer(textos_entrada, truncation=True, padding=True, max_length=40, return_tensors="pt")
```

Nota. Carga del tokenizador de Bert con el modelo preentrenado de Bert en español para tokenizar y codificar los textos, los cuales se han almacenado en el objeto "textos_entrada". Información adaptada de Google Colab, 2023.

Para realizar predicciones de los sentimientos y emociones de los datos de entrada se utiliza el modelo previamente cargado, como se ve en la Figura 25, luego se determina "encodings_textos_entrada.input_ids" como las entradas de ID de tokens y "encodings_textos_entrada.attention_mask" como la máscara de atención correspondiente para los encodings de los textos de entrada y los resultados se almacenan en "emocion_outputs", posteriormente se utiliza la variable "emocion_id2label" para convertir los índices de etiquetas predichos en etiquetas legibles y comprensibles para las emociones y se utiliza "torch.argmax" para conseguir el índice de la etiqueta predicha con la puntuación más alta para cada ejemplo.

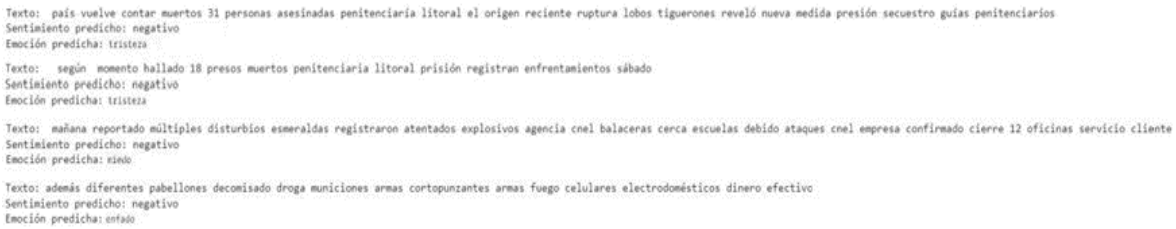
Figura 25. *Conversiones índices de etiquetas predichos en etiquetas legibles para las emociones*

```
# Realizar predicciones para emociones
with torch.no_grad():
    emocion_outputs = modelo_cargado(encodings_textos_entrada.input_ids, attention_mask=encodings_textos_entrada.attention_mask)
    emocion_predicho_indices = torch.argmax(emocion_outputs.logits, dim=1)

# Convertir los índices de etiquetas en etiquetas legibles para emociones
emocion_predicha_etiquetas = [emocion_id2label[idx] for idx in emocion_predicho_indices.tolist()]

# Imprimir las predicciones para sentimientos y emociones
for texto, sentimiento, emocion in zip(textos_entrada, sentimiento_predicho_etiquetas, emocion_predicha_etiquetas):
    print(f"Texto: {texto}")
    print(f"Sentimiento predicho: {sentimiento}")
    print(f"Emoción predicha: {emocion}")
    print()
```

Nota. “Encodings_textos_entrada.input_ids”, “encodings_textos_entrada.attention_mask”: textos entrada; almacenados: “emocion_outputs”; “emocion_id2label”: conversión etiquetas legibles; “torch.argmax”: puntuación más alta. Información adaptada de Google Colab, 2023.

Al finalizar, como se muestra unos ejemplos en la Figura 26, se obtiene como salida: el texto, el sentimiento y la emoción predichos.

Figura 26. *Predicciones de sentimientos y emociones*



```
Texto:  país vuelve contar muertos 31 personas asesinadas penitenciaría litoral el origen reciente ruptura lobos tiguerones reveló nueva medida presión secuestro guías penitenciarios
Sentimiento predicho: negativo
Emoción predicha: tristeza

Texto:   según  momento hallado 18 presos muertos penitenciaria litoral prisión registran enfrentamientos sábado
Sentimiento predicho: negativo
Emoción predicha: tristeza

Texto:  mañana reportado múltiples disturbios esmeraldas registraron atentados explosivos agencia cnel balaceras cerca escuelas debido ataques cnel empresa confirmado cierre 12 oficinas servicio cliente
Sentimiento predicho: negativo
Emoción predicha: miedo

Texto: además diferentes pabellones decomisado droga municiones armas cortopunzantes armas fuego celulares electrodomésticos dinero efectivo
Sentimiento predicho: negativo
Emoción predicha: enfado
```

Nota. Como salida: el texto, el sentimiento y la emoción predichos. Información adaptada de Google Colab, 2023.

Posteriormente, como se observa en la Figura 27, se crea un DataFrame con las predicciones, en este caso representadas en las columnas “Sentimiento predicho” y “Emoción predicha”, también las columnas: ID del tweet, ID del Autor, “Nombre del Autor” y “Texto”, denominado “df_predicciones” y se guardan en un archivo JSON utilizando la siguiente sentencia de código.

Figura 27. *Almacenamiento de predicciones en archivo JSON*

```
import json

# Se define la ruta del archivo json
ruta_archivo_json = "ruta/del/archivo.json"

# Convertir el DataFrame a formato JSON
json_data = df_predicciones.to_json(orient="records", lines=True)

# Guardar el JSON en un archivo
with open(ruta_archivo_json, "w") as archivo:
    archivo.write(json_data)
```

Nota. DataFrame con predicciones: “Sentimiento predicho”, “Emoción predicha”; columnas: ID del tweet, ID del Autor, “Nombre del Autor”, “Texto”, denominado “df_predicciones”. Guardadas en archivo JSON usando sentencia de código. Información adaptada de Google Colab, 2023.

A continuación, en la Figura 28, se muestra un ejemplo del cómo se visualiza un documento guardado en formato JSON.

Figura 28. *Documento almacenado en formato JSON*


```
{
    "ID":"1683968410609233921",
    "ID del Autor":"777891400297897985",
    "Nombre del Autor":"Ecuador Chequea",
    "Texto":"\ud83d\udea8#URGENTE | La @FiscaliaEcuador inform\u00f3 que hasta ahora hay 31 muertos y 14 heridos, tras los enfrentamientos registrados desde el
    "Sentimiento predicho":"negativo",
    "Emoción Predicha":"tristeza"
}
```


Nota. Visualización del documento guardado en formato JSON. Información adaptada de Google Colab, 2023.

Para finalizar, estos documentos se almacenan en una nueva colección denominada “predicciones” en la base de datos “Twitter principal” de Mongo DB Atlas.

Discusión

Al visualizar los resultados de las predicciones de los sentimientos y las emociones identificadas en las noticias de las verificadoras de hecho en Ecuador aplicando técnicas de Procesamiento de Lenguaje Natural (PLN), se podría indicar que se tienen resultados alentadores, brindando una visión de cómo pueden llegar a percibirse estas noticias.

Utilizando como modelo base el modelo preentrenado de Bert en español junto con datos previamente etiquetados, se ha podido entrenar un modelo que ha logrado identificar y categorizar un alto nivel de precisión los sentimientos y emociones asociadas al contenido de las noticias seleccionadas por Ecuador Chequea y Ecuador Verifica.

Es importante mencionar que, al verificar los resultados de las predicciones de los sentimientos, se pudo corroborar que existen más noticias categorizadas por el sentimiento negativo, esto se puede interpretar que es porque las noticias contienen información de los últimos sucesos que se han presentado en el país, los cuales generan sentimientos negativos y de tristeza a las personas en el Ecuador.

Conclusiones

En el transcurso de este estudio, se ha podido verificar a profundidad el funcionamiento de un análisis de sentimientos en el Procesamiento de Lenguaje Natural (PLN), se han analizado varias técnicas de procesamiento de texto y varios modelos de clasificación, seleccionando el modelo preentrenado de BERT en español de la librería Transformers de Hugging Face, se realizaron pruebas con este modelo sin previo entrenamiento para determinar su funcionamiento en el análisis de sentimientos, se verificó que puede identificar y categorizar sentimientos en positivos y negativos, pero en el presente caso de estudio, se pretendía identificar las emociones que se perciben en las noticias seleccionadas por las verificadoras de hechos del Ecuador.

Por consiguiente, se entrenó el modelo con datos etiquetados con sentimientos en negativos y positivos y las emociones esperanza, alegría, tristeza, enfado y miedo, procediendo posteriormente con una evaluación del modelo en donde se obtuvo una precisión del 70% respecto a las predicciones del conjunto de prueba, verificando así un comportamiento adecuado para el análisis de sentimientos en las noticias.

Es importante mencionar, que el uso del modelo preentrenado de BERT, ha demostrado una precisión favorable en la clasificación o categorización de los sentimientos en las noticias, adicionalmente, un punto relevante en este estudio es que al tener la categorización de los sentimientos en un formato JSON, esto podría ser utilizado posteriormente para verificar si una notica cuando se etiqueta con el sentimiento negativo tiende a ser falsa.

Referencias bibliográficas

Capuano, N., Fenza, G., Loia, V., & Nota, F. D. (2023). Content-Based Fake News Detection With Machine and Deep Learning: a Systematic Review. Neurocomputing, 530 (ISSN 0925-2312), 91-103. doi:https://doi.org/10.1016/j.neucom.2023.02.005

Cusot, G., & Palacios, I. (2019). Las FAKE NEWS y las estrategias de verificación del discurso público: Caso Ecuador Chequea. 3, págs. 88-107. doi: https://doi.org/10.18272/pd.v3i1.1558

Ecuador Chequea. (17 de octubre de 2018). Ecuador Chequea by Fundamedios. Obtenido de Ecuador Chequea frente a los hechos ocurridos en Posorja: https://ecuadorchequea.com/editorial-ecuadorchequea-posorja-linchamiento- fakenews/

Galeano, S. (26 de enero de 2023). El número de usuarios de internet en el mundo crece un 1,9% y alcanza los 5.160 millones (2023). Obtenido de Marketing 4 Ecommerce: https://marketing4ecommerce.net/usuarios-de-internet-mundo/

Gleick, J. (2011). La información: historia y realidad. (koothrapali, Ed.)

Google. (2023). Google Colaboratory. Retrieved April 18, 2023, from https://colab.research.google.com/

Hütt Herrera, H. (2012). LAS REDES SOCIALES: UNA NUEVA HERRAMIENTA DE DIFUSIÓN. Reflexiones, 91(2), 121-128.

Kemp, S. (26 de enero de 2023). DataReportal - Global Digital Insights. Obtenido de DataReportal -Global Digital Insights: https://datareportal.com/reports/digital- 2023-global-overview-report

Liu, B. (2015). Sentiment Analysis. New York: Cambridge University Press.

Mayo, M. (3 de mayo de 2018). Preprocesamiento de datos de texto: un tutorial en Python. Obtenido de https://medium.com/datos-yciencia/preprocesamiento-de-datos-de- texto-un-tutorial-en-python-5db5620f1767

Naso, F., Balbi, M. L., Di Grazia, N., & Peri, J. A. (2012). La importancia de las Redes sociales en el ámbito educativo. Universidad Nacional del Noroeste de la Provincia de Buenos Aires, 1-8.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., & Thirion, B. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825- 2830.

Piñero, M. S. (2021). MARKETING EN REDES SOCIALES. Revista de Estudios Empresariales (2), 317-319. doi: https://doi.org/10.17561/ree.n2.2022.7150

Priya, B. (31 de marzo de 2023). Los 12 mejores cuadernos colaborativos de ciencia de datos [alternativas de Jupyter]. Recuperado el 8 de junio de 2023, de Geekflare: https://geekflare.com/es/best-data-science-notebooks/

Romero Moreno, F. Y., Sanchez Martelo, C. A., Breed Yeet, A. C., Sanchez Cifuentes, J. F., & Ospina López, J. P. (2020). Técnicas para la Clasificación de Sentimientos en Redes Sociales como Apoyo en el Marketing Digital. RISTI (35), 167-186.

Sahagian, G. (30 de noviembre de 2020). What is Random State? And Why is it Always 42? Obtenido de Medium: https://grsahagian.medium.com/what-is-random-state-42-d803402ee76b

Santamaria, P. (16 de enero de 2017). Antropología social: La diferencia y relación entre posverdad y las noticias falsas. Merca2.0. Obtenido de Merca2.0: https://goo.gl/agXAMR.