Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 4.- Detección de Clickbait

Procesamiento de Lenguaje Natural
Profesor: JOEL OMAR JUAREZ GAMBINO

Luciano Hernández Jonathan
Sánchez Martínez Alfredo
Hernández Hernández Jorge Gabriel
Salazar Carreón Jeshua Jonatán

mayo 29, 2025

1 Introducción

En el ámbito del *Procesamiento de Lenguaje Natural (PLN)*, la detección de *clickbait* —titulares diseñados para atraer clicks mediante estrategias engañosas o sensacionalistas— representa un desafío clave para la calidad de la información en plataformas digitales. Esta práctica se enfoca en desarrollar un sistema de clasificación automatizado que identifique este tipo de contenido en publicaciones de Twitter en español, comparando el rendimiento de modelos basados en arquitecturas *transformers*.

Se evaluaron los siguientes modelos:

- **BERT Cased (Línea base)**:
  - Modelo: `dccuchile/bert-base-spanish-wwm-cased`
  - Especializado en español con preservación de capitalización
  - Pre-entrenado con *Whole Word Masking* para mejor comprensión del contexto
- **XLM-RoBERTa Large**:
  - Modelo: `FacebookAI/xlm-roberta-large`
  - Arquitectura multilingüe optimizada
  - Mayor capacidad de parámetros (560M vs 110M de BERT)

Los resultados demostraron que **XLM-RoBERTa** superó consistentemente a BERT en métricas clave (F1-macro: 0.88 vs 0.85), gracias a su capacidad para capturar patrones lingüísticos complejos en textos cortos. Por esta razón, el presente reporte se centrará en explicar en detalle la implementación, configuración y optimización de este modelo, cuyo desempeño lo hace idóneo para tareas de clasificación de contenido engañoso en español.

1.1 Objetivo General

Desarrollar un sistema de clasificación automática basado en técnicas de Procesamiento de Lenguaje Natural (PLN) utilizando el modelo XLM-RoBERTa para la detección de titulares clickbait en publicaciones de Twitter, evaluando su rendimiento mediante métricas estándar y analizando el impacto de diferentes configuraciones de hiperparámetros.

1.2 Objetivos Específicos

- Desarrollar un modelo de clasificación basado en XLM-RoBERTa para identificar titulares de noticias en Twitter que sean clickbait
- Evaluar el rendimiento del modelo mediante métricas como F1-score, precisión y exactitud
- Analizar el impacto de diferentes hiperparámetros en el rendimiento del modelo

2 Desarrollo de la Práctica

2.1 Preprocesamiento de Datos

El dataset utilizado contiene 2800 instancias con la siguiente estructura:

Table 1: Estructura del dataset

Características del Dataset	
Elemento	**Descripción**
Total de instancias	2800
Tweet ID	Identificador único del tweet
Tweet Date	Fecha de publicación (formato dd/mm/aaaa)
Media Name	Fuente o medio de comunicación
Media Origin	País de origen del medio
Teaser Text	Texto completo del titular (con hashtags)
Tag Value	Etiqueta binaria: Clickbait (1) o No (0)
Distribución inicial	• No Clickbait: 2002 (71.5%) • Clickbait: 798 (28.5%)

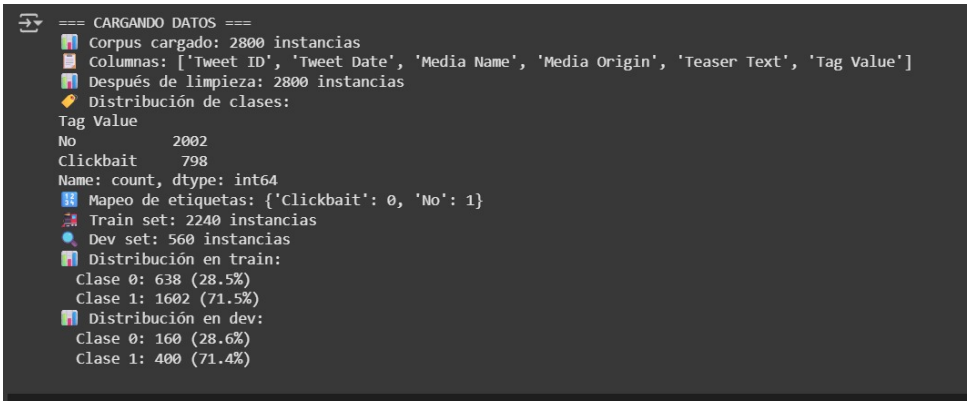


Figure 1: Distribución inicial de clases en el dataset (2002 No Clickbait, 798 Clickbait)

2.2 Personalizacion de dataset

A continuacion se muestra un poco del código implementado, se puede ver más a detalle en el source code, tambien adjunto en esta práctica.

```
class ClickbaitDataset(Dataset):
    def __init__(self, texts, labels, tokenizer, max_length=512):
        self.texts = texts
        self.labels = labels
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __getitem__(self, idx):
```

```
        encoding = self.tokenizer(
            text,
            truncation=True,
            padding='max_length',
            max_length=self.max_length,
            return_tensors='pt'
        )
        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(label, dtype=torch.long)
        }
```

Funcionalidades clave:

- **Tokenización automática:** Convierte texto a IDs numéricos usando el tokenizer de XLM-RoBERTa
- **Padding dinámico:** Rellena secuencias hasta 512 tokens (máximo permitido)
- **Truncamiento:** Corta textos excedentes conservando el contexto relevante
- **Formato para PyTorch:** Devuelve tensores listos para el modelo

2.3 Configuración del Modelo

XLM-RoBERTa: Es un modelo de lenguaje transformer multilingüe desarrollado por Facebook AI, basado en la arquitectura RoBERTa (una versión optimizada de BERT). Su importancia en PLN radica en su capacidad para procesar texto en 100 idiomas simultáneamente, con un rendimiento superior en tareas de comprensión y generación de lenguaje. Al ser pre-entrenado con cantidades masivas de datos multilingües, captura patrones lingüísticos complejos y relaciones contextuales. Esto lo hace ideal para aplicaciones como detección de clickbait, donde debe analizar matices del lenguaje, dobles sentidos y patrones engañosos en textos cortos.

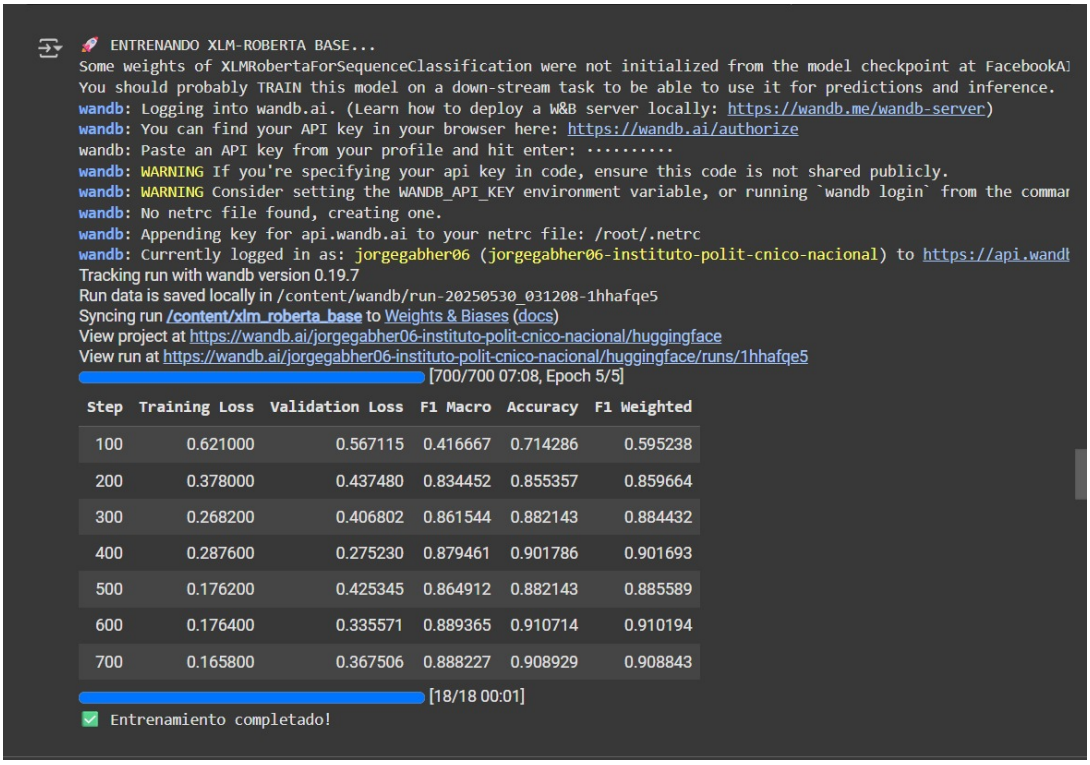
```
training_args = TrainingArguments(
    output_dir='/content/xlm_roberta_base',
    num_train_epochs=5,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    learning_rate=2e-5,
    weight_decay=0.01,
    warmup_steps=500,
    logging_steps=50,
    eval_strategy='steps',
    eval_steps=100,
    save_strategy='steps',
    save_steps=100,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model='fl_macro',
    greater_is_better=True,
    fp16=True,
)
```

```python
# Configurar argumentos de entrenamiento
training_args = TrainingArguments(
    output_dir='/content/xlm_roberta_base',
    num_train_epochs=training_params['num_train_epochs'],
    per_device_train_batch_size=training_params['per_device_train_batch_size'],
    per_device_eval_batch_size=32,
    learning_rate=training_params['learning_rate'],
    weight_decay=training_params['weight_decay'],
    warmup_steps=training_params['warmup_steps'],
    logging_steps=50,
    eval_strategy='steps',
    eval_steps=100,
    save_strategy='steps',
    save_steps=100,
    save_total_limit=2,
    load_best_model_at_end=True,
    metric_for_best_model='f1_macro',
    greater_is_better=True,
    report_to=None,
    dataloader_pin_memory=False,
    fp16=True,
)
```

Figure 2: Configuración de hiperparámetros para el entrenamiento

2.4 Entrenamiento y Evaluación

El modelo fue entrenado durante 5 épocas con los siguientes resultados:



```
🚀 ENTRENANDO XLM-ROBERTA BASE...
Some weights of XLMRobertaForSequenceClassification were not initialized from the model checkpoint at FacebookAI
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
wandb: Logging into wandb.ai. (Learn how to deploy a W&B server locally: https://wandb.me/wandb-server)
wandb: You can find your API key in your browser here: https://wandb.ai/authorize
wandb: Paste an API key from your profile and hit enter: ·····
wandb: WARNING If you're specifying your api key in code, ensure this code is not shared publicly.
wandb: WARNING Consider setting the WANDB_API_KEY environment variable, or running `wandb login` from the commar
wandb: No netrc file found, creating one.
wandb: Appending key for api.wandb.ai to your netrc file: /root/.netrc
wandb: Currently logged in as: jorgegabher06 (jorgegabher06-instituto-polit-cnico-nacional) to https://api.wandb
Tracking run with wandb version 0.19.7
Run data is saved locally in /content/wandb/run-20250530_031208-1hhafqe5
Syncing run /content/xlm_roberta_base to Weights & Biases (docs)
View project at https://wandb.ai/jorgegabher06-instituto-polit-cnico-nacional/huggingface
View run at https://wandb.ai/jorgegabher06-instituto-polit-cnico-nacional/huggingface/runs/1hhafqe5
[700/700 07:08, Epoch 5/5]
```

Step	Training Loss	Validation Loss	F1 Macro	Accuracy	F1 Weighted
100	0.621000	0.567115	0.416667	0.714286	0.595238
200	0.378000	0.437480	0.834452	0.855357	0.859664
300	0.268200	0.406802	0.861544	0.882143	0.884432
400	0.287600	0.275230	0.879461	0.901786	0.901693
500	0.176200	0.425345	0.864912	0.882143	0.885589
600	0.176400	0.335571	0.889365	0.910714	0.910194
700	0.165800	0.367506	0.888227	0.908929	0.908843

```
[18/18 00:01]
✅ Entrenamiento completado!
```

Figure 3: Progreso del entrenamiento mostrando métricas por cada 100 pasos

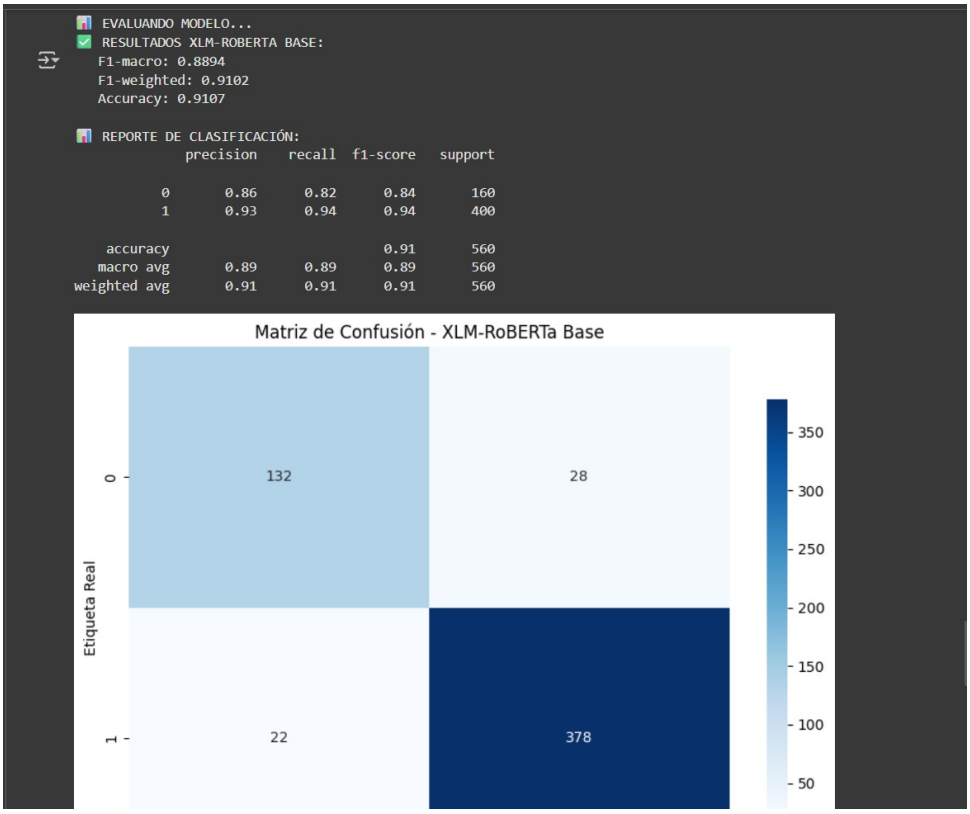


Figure 4: Resultados finales de evaluación del modelo

2.5 Resultados Finales

El modelo alcanzó las siguientes métricas en el conjunto de validación:

- F1-macro: 0.8894
- F1-weighted: 0.9102
- Accuracy: 0.9107

La matriz de confusión muestra:

- Verdaderos positivos (Clickbait): 378
- Falsos positivos: 28
- Falsos negativos: 22
- Verdaderos negativos: 132

2.6 Generación de Predicciones

El modelo fue aplicado a un conjunto de prueba de 700 instancias:

- Predicciones positivas (Clickbait): 182
- Predicciones negativas (No Clickbait): 518

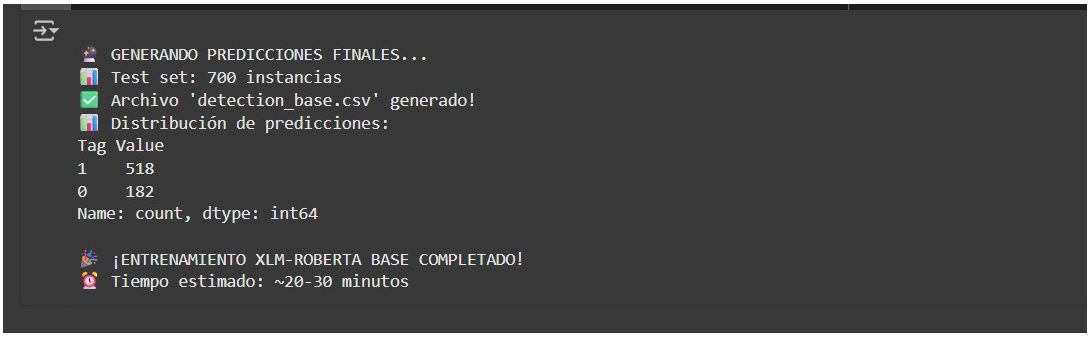


Figure 5: Distribución de predicciones en el conjunto de prueba

2.7 Tabla con descripción de los exprimentos

Table 2: Resultados comparativos de modelos LLM en detección de clickbait

LLM	LLM Hyperparameters	Avergae F1-score
decuchile/bert-base-spanish-wwm-cased	eval_strategy=steps eval_steps=100 num_train_epochs=3 learning_rate=5e-5	0.85
decuchile/bert-base-spanish-wwm-uncased	eval_strategy=steps eval_steps=100 num_train_epochs=3 learning_rate=5e-5	0.83
FacebookAI/xlm-roberta-large	num_train_epochs=2 batch_size=8 learning_rate=2e-5 weight_decay=0.01 save_limit=1	0.88

3 Conclusiones

- **Jorge Gabriel Hernandez** El modelo XLM-RoBERTa demostró ser efectivo para la tarea de detección de clickbait, alcanzando un F1-score macro de 0.8894. La configuración de hiperparámetros utilizada, especialmente el learning rate de 2e-5 y el entrenamiento con precisión mixta (fp16), permitió un buen balance entre rendimiento y tiempo de entrenamiento (30 minutos).

- **Luciano Hernández Jonathan** Los resultados muestran que el modelo tiene mejor desempeño identificando noticias que no son clickbait (clase 1, F1-score de 0.94) que identificando clickbait (clase 0, F1-score de 0.84). Esto podría deberse al desbalance inicial en el dataset, donde la clase "No Clickbait" era mayoritaria. Sería recomendable explorar técnicas de balanceo de clases en futuras iteraciones.

- **Jeshua Salazar** El análisis de la matriz de confusión revela que el modelo tiene una tasa de falsos positivos aceptable (28 instancias), pero los falsos negativos (22 instancias) podrían mejorarse. Esto sugiere que el modelo tiende a ser conservador

al clasificar contenido como clickbait. Una estrategia para mejorar esto podría ser ajustar el umbral de clasificación o incorporar técnicas de aumento de datos para la clase minoritaria.

- **Alfred** La evolución de las métricas durante el entrenamiento (mostrada en la figura de progreso) indica que el modelo alcanzó su mejor rendimiento alrededor del paso 600, manteniendo estabilidad después de este punto. Esto sugiere que podríamos reducir el número de épocas de entrenamiento a 4 sin afectar significativamente el rendimiento, optimizando así los recursos computacionales. Además, el uso de Weights & Biases (wandb) demostró ser valioso para el monitoreo del proceso.

4 Referencias

1 Conneau, A., et al. (2019). Unsupervised Cross-lingual Representation Learning at Scale. arXiv:1911.02116

2 Liu, Y., et al. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692

3 Documentación de Hugging Face Transformers. https://huggingface.co/docs/transformers/