BERT model implementation for sentiment analysis in the 2022 presidential elections in Colombia.

Pablo Andrés Pertuz Duran[1], Julián Alfonso Camacho Bertel[2], Iván Daniel Zapata Florez[3].
[1]Ingeniería Mecatrónica, Universidad Tecnológica de bolívar, Cartagena, Colombia.
[2]Ingeniería de sistemas y computación, Universidad Tecnológica de bolívar, Cartagena, Colombia.
[3]Ingeniería de sistemas y computación, Universidad Tecnológica de bolívar, Cartagena, Colombia.

[*]ppertuz@utb.edu.co

Resumen: Este trabajo se hizo con el objetivo de que a través de los modelos BERT y Bidirectional LSTM, llevar a cabo el análisis de sentimientos de la red social Twitter con respecto al debate presidencial ocurrido el pasado 23 de mayo, donde se dejó abierto un hashtag en el cual los televidentes podían dar su punto de vista de estos 3 candidatos: Gustavo Petro, Federico Gutiérrez y Rodolfo Hernández. Una vez extraídos estos tweets contenidos en el hashtag, se clasificaron manualmente siendo 0 los que tenían un mensaje negativo y 1 los que tenían un mensaje positivo, para luego pasar por todo el preprocesamiento, eliminación de caracteres especiales, links, urls, imágenes y/o videos. Para convertir estos tweets a vectores se usó la capa TextVectorization, de la librería de tensorflow y finalmente para pasar por los dos modelos; Dando como resultado un mejor rendimiento para el modelo BERT con un *accuracy* de 0.7667 y un *f1 score* de 0.8511.

1. Introducción

1.1 Contexto de la problemática abordada

En este proyecto hablaremos de como en la red social conocida como Twitter, no presenta un alto nivel de censura, y se expresa con más libertad las opiniones de las personas, en esta red social están los hashtags, que son grupos de chat donde la gente puede dar sus opiniones fomentando el deliberar. El análisis de sentimiento tiene como objetivo el tratamiento de textos de los cuales se extrae emoción, que puede ser positiva o negativa. El objetivo de este proyecto es realizar el análisis de sentimiento a las elecciones presidenciales de Colombia en este año 2022, dando como resultado quienes son aquellos candidatos que generan más sentimientos positivos o negativos.

1.2 Antecedentes

- Análisis del sentimiento político en Twitter durante las elecciones congresales 2020 en el Perú.
- Electoral prediction using a hybrid model based on sentimental analysis: presidential elections in Colombia.
- Inteligencia artificial entrega proyección electoral de más de 100.000 perfiles en redes.

- Favorabilidad en redes de los candidatos de cara a las elecciones, según la inteligencia artificial.
- Marketing político e inteligencia artificial.

1.3 Descripción del problema

El pasado lunes 23 de mayo hubo un debate presidencial entre los representantes que se están postulando a las elecciones de la presidencia de la república de Colombia, en dicho debate se abrieron encuestas y espacios donde las personas podían dar sus opiniones acerca de las propuestas presentadas por cada uno de los postulantes y de las respuestas presentadas en dicho debate.

1.4 Planteamiento de la solución propuesta

En este proyecto vamos a entrenar una red neuronal basada en BERT que determine que comentario de los que presentan las personas en las encuestas en Twitter es positivo o negativo y si el comentario da un visto bueno o malo del político, para al final reunir toda esta información y poder conocer los sentimientos de las personas acerca de los candidatos, si estos son positivos o negativos. BERT es un algoritmo de Google basado en inteligencia artificial y PLN (procesamiento del lenguaje natural). Con él, el motor de búsqueda es capaz de comprender mucho más profundamente las intenciones de búsqueda de los usuarios y los contenidos de las páginas web.

2. Trabajos relacionados

Para realizar este proyecto se tuvo en cuenta algunos artículos de referencia, los cuales sirven de evidencia de cómo funciona la red neuronal con la que trabajamos en este proyecto y así corregir errores o implementar mejoras.

Dentro de los artículos destaca el de Alva Segura y Daniel Abraham; quienes en su artículo implementaron un análisis de sentimiento en las elecciones de congresos del año del 2020 en Perú con las herramientas tecnológicas que se encontraban en el mercado que solucionen los problemas de análisis de sentimiento.

Otro que destaco fue el Mauro Callejas y Manuel Vélez; quienes en su artículo propusieron un modelo hibrido que predecía el desenlace de la primera vuelta de elecciones presidenciales de Colombia en el año de 2018, con el objetivo de mejorar la calidad de predicción final y minimizar los errores. Donde las actividades de los usuarios de Facebook y Twitter fueron registradas y analizadas por algoritmos de inteligencia artificial obteniendo resultados precisos y coherentes.

3. Metodología

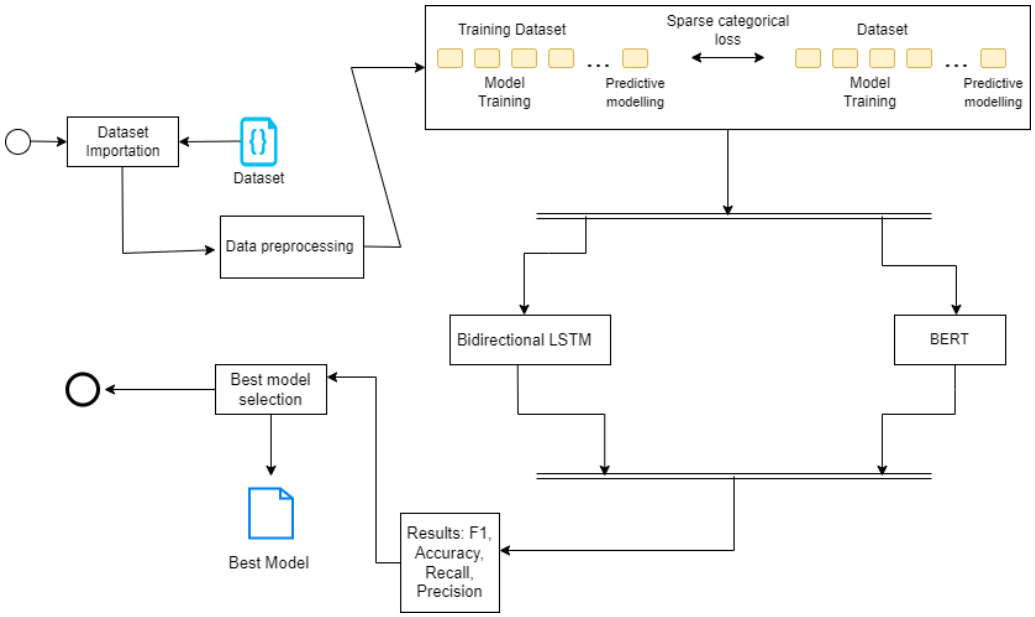

Figura 1. Pipeline de la arquitectura

3.1 Etapas del Pipeline/Arquitectura

El Dataset es importado utilizando la librería de pandas. Luego se realiza una etapa de *data exploration* y se procede a realizar el preprocesamiento de los datos. Una vez las cadenas de texto están limpias, se codifican de la forma adecuada para cada modelo (*vectorization layer* para el modelo LSTM Bidireccional y la forma de salida de la función InputExample de la librería de *transformers* para el modelo BERT preentrenado). Cada modelo es alimentado con los datos de entrenamiento y validación, y entrenado. Se usan las métricas de *Accuracy*, *Recall, Precision, y F1 Score* para decidir qué modelo es mejor, y se selecciona el mismo para futuros desarrollos.

3.2 Conjuntos de Datos utilizados.

3.2.1 API de Twitter

Mediante la API de Twitter, y usando el secret_token solicitado a el equipo de Twitter, hicimos la extracción de todos los tweets que contenían el hashtag #ElDebateDefinitivo, programa que se realizó el lunes 23 de Mayo del presente año 2022, y donde todos sus televidentes opinaban mediante ese hashtag. Usando esta API se extrajeron 2000 tweets en español, donde se hablaba de los candidatos Gustavo Petro, Federico Gutiérrez y Rodolfo Hernández. Se eliminaron los tweets duplicados, lo que dejó un total de 246 tweets únicos.

3.2.2 Etiquetado manual

Para el entrenamiento del modelo, se extrajeron aquellos tweets que solo se referían a un candidato para evitar ambigüedades, luego se etiquetaron manualmente los tweets correspondiendo 79 al candidato Petro, 37 a Federico Gutiérrez y 10 a Rodolfo Hernández. Aquellos que manifestaran un sentimiento negativo se etiquetaban con un cero y los positivos con un 1. Los tweets neutrales no se consideraron, dejando la casilla de label vacía.
Finalmente, se juntaron los 3 datasets en uno único, quitando aquellos tweets con un label NaN (tweets neutrales), siendo el dataset final de 123 filas.
Se exportó el dataset a un formato CSV con codificación utf-8 usando la librería de pandas.

3.3 Análisis descriptivo de los datos

El conjunto de datos *tweets.csv* corresponde a un dataframe de forma (123,2) dónde la primera columna corresponde a los tweets extraídos (sin procesar) y la segunda columna a un label correspondiente al sentimiento (0: Negativo, 1: Positivo).

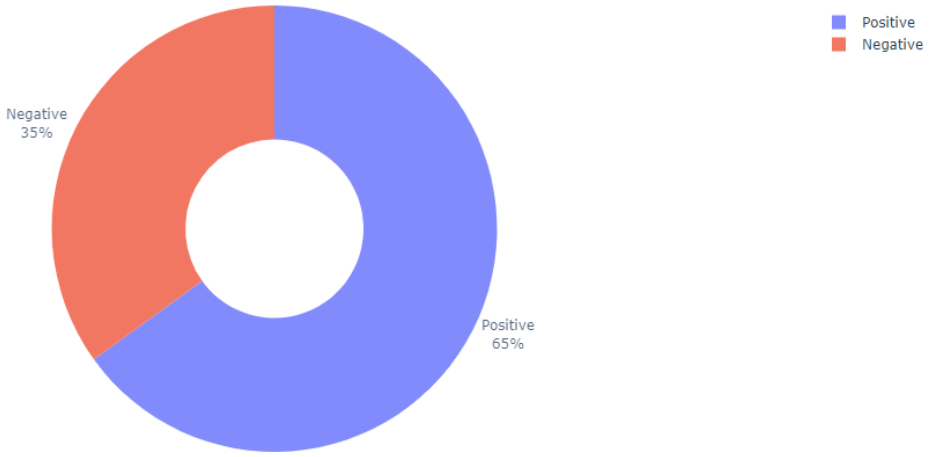

Figura 2. Distribución de clases en el dataset

En el dataframe, 43 tweets corresponden a un sentimiento negativo y 80 a un sentimiento positivo. En total el dataframe tiene 123 filas sin valores vacíos NaN.

3.4 Preprocesamiento de los datos

Para el procesamiento de los datos se utilizaron 7 funciones:
Remove links: Se utilizó para remover links incluidos en los comentarios.
Remove users: Se utilizó para remover menciones a usuarios en los comentarios.
Remove hashtags: Se utilizó para remover hashtags en los comentarios.
Remove av: Se utilizó para remover etiquetado de imágenes y/o vídeos en los comentarios.
Remove emoji: Se utilizó para remover emojis en los comentarios, basada en la librería *clean text*.
Remove punctuation: Se utilizó para remover signos de puntuación en los comentarios.
Preprocessing: Se utilizó para implementar las 6 funciones en cada cadena de texto.

Después del preprocesamiento, algunos de los tweets quedaron vacíos, estos fueron reemplazados con un valor nulo y desechados, siendo el dataframe final de 117 filas (tweets).

3.4 Extracción de características empleadas

3.4.1 Extracción para la arquitectura LSTM Bidireccional

Para convertir las palabras de los tweets a vectores, se aplicó la capa TextVectorization propio de la librería de tensorflow. TextVectorization es una capa de preprocesamiento que asigna características de texto a secuencias enteras.
Se definió el número de palabras en el vocabulario del vectorizador como 200000 (MAX_FEATURES), con una longitud máxima de 1800 palabras por tweet a vectorizar. Si el tweet no abarca el límite de 1800 palabras, el vector será completado con ceros (zero padding).
El vectorizador fue adaptado (generación del vocabulario) con los tweets procesados del dataset de entrenamiento, y fue usado para vectorizar los tweets del dataset de entrenamiento y el dataset de validación.

3.4.2 Extracción para la arquitectura BERT

Para la extracción de características se crearon 3 funciones:

Convert data to examples: Esta función tiene como parámetros los dataframe de entrenamiento y validación, y los nombres de las columnas de la variable independiente (los tweets) y el *target* o variable objetivo (labels). Utiliza la función *InputExample* de la librería de transformers para crear ejemplos, un formato admitido por los modelos de transformers que consiste en dos piezas de texto (text_a es el tweet y text_b es vacío) y el label correspondiente.

Convert examples to TF Dataset: Esta función tiene como parámetros los ejemplos generados por la función *convert_data_to_examples,* el tokenizador inicializado (para este trabajo, el tokenizador basado en el modelo *bert-base-uncased* y un truncador de un máximo de 128 tokens (default) como entrada del modelo. Básicamente, tokeniza los ejemplos generados y los prepara para ser entradas aptas para el modelo, esto uzando el método *encode_plus*:

- Agrega tokens especiales.
- Establece una longitud máxima.
- Retorna los IDs de los tokens.
- Retorna *attention masks.*
- *Padding.*
- *Truncation.*

3.5 Método empleado para evitar el sobre entrenamiento de los modelos

Se utilizó la clase *EarlyStopping* de Keras para monitorear el entrenamiento de los modelos. Esta usa 7 parámetros:

- La métrica a monitorear.
- El incremento o decremento mínimo que debe sufrir la métrica para no detener el entrenamiento.
- El número de *epochs* a esperar antes de detener el entranamiento.
- Si mostrar o no detalles durante el entrenamiento (*verbose*).

- Establecer si se busca un decremento o incremento en la métrica a monitorear.
- Si se quiere comparar el rendimiento de un modelo con una linea base.
- Si se desea restaurar los valores del modelo (*checkpoint*) en el momento que presentó el mejor rendimiento.

Se usó un cambio delta mínimo de 0.001, una paciencia de 6 epochs, restaurando los valores correspondientes al mejor rendimiento durante el entrenamiento.

3.6 Medidas de evaluación usadas

Las métricas de evaluación que se llevaron a cabo fueron: Accuracy, Precision, Recall y F1.
Accuracy(exactitud): Mide el porcentaje de casos en el que el modelo ha acertado.
Precision: la calidad del modelo a la hora de clasificar (cantidad de positivos correctamente clasificados, sobre el total de los positivos clasificados).
Recall (Exhaustividad): Cantidad de positivos correctamente clasificados, sobre el total de positivos reales.
F1 Score: Este valor se utiliza para combinar las medidas de precisión y recall en un solo valor. Esto es practico porque se hace más fácil el poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones.

3.7 Entrenamiento

Para el entrenamiento se iteró entre la distribución de los datos de entrenamiento y validación respecto al dataset base, entre 80-20, 75-25 y 70-30. Se usaron como métricas la *accuracy*, *precision*, *recall* y *F1 score*. Además, se iteró el valor del learning rate entre 1e-5, 2e-5, 3e-5, 4e-5 y 5e-5.

4. Resultados

4.1 Iteraciones realizadas, ajustes y resultados

Para evaluar la mejor distribución de entrenamiento y validación, se utilizó una linea base con un learning rate de 1e-5 para el optimizador, función de pérdida *binary_crossentropy* y

binary_accuracy como métrica para el modelo Bidireccional, y *sparse_categorical_crossentropy* de función de pérdida u *sparse_categorical_accuracy* como para el modelo BERT. Luego, se compararon los valores más altos para cada métrica.

Train/Valid Split 80%-20% Best performance:

Tomó 100 epochs de 400ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.6667	1	**0.8**	0.6667
BERT	1	0.95	0.7917	**0.7**

Figura 3. Performance Bi-LSTM y BERT model train/valid 80-20

Se observó el comportamiento de la función de pérdida para los datos de entrenamiento. A diferencia del modelo BERT, la función de pérdida del modelo Bi-LSTM presentó un comportamiento errático.

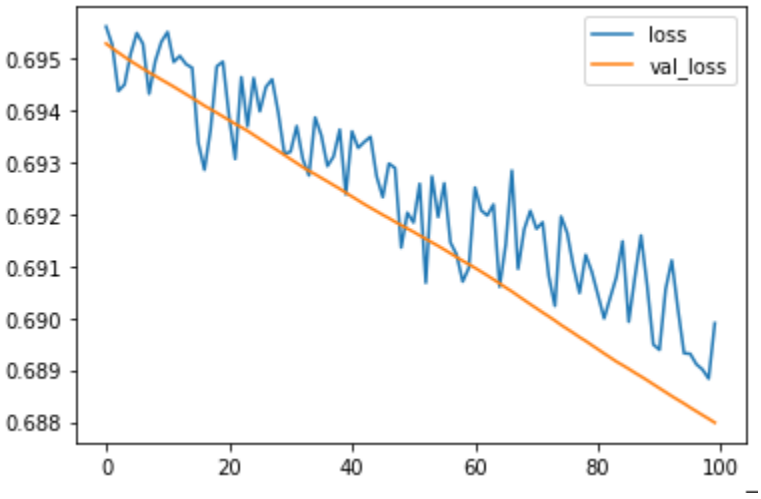

Figura 4. Loss Function Bi-LSTM train/valid 80-20

Train/Valid Split 75%-25% Best performance:

Tomó 100 epochs de 400ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.6667	1	0.8	0.6667

BERT	0.8333	1	**0.8163**	**0.7333**

Figura 5. Performance Bi-LSTM y BERT model train/valid 75-25

Train/Valid Split 70%-30% Best performance:

Tomó 100 epochs de 450ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.7188	1	0.8360	0.6944
BERT	0.8333	1	**0.8511**	**0.7667**

Figura 6. Performance Bi-LSTM y BERT model train/valid 70-30

Se evidenció que la distribución 70%-30% presentó el mejor rendimiento en cuanto al a*ccuracy/F1 score*. Se usará esta para las siguientes comparaciones.

Model	80%-20%	75%-25%	70%-30%
Bi-LSTM	0.6667	0.6667	0.6944
BERT	**0.7**	**0.7333**	**0.7667**

Figura 7. Accuracy Bi-LSTM y BERT model in all train/valid splits

Model	80%-20%	75%-25%	70%-30%
Bi-LSTM	**0.8**	0.8	0.8360
BERT	0.7918	**0.8163**	**0.8511**

Figura 8. F1 score Bi-LSTM y BERT model in all train/valid splits

Resultados para un learning rate de 1e-5:

Tomó 100 epochs de 400ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.7188	1	0.8360	0.6944
BERT	0.8333	1	**0.8511**	**0.7667**

Figura 9. Performance Bi-LSTM y BERT model learning rate 1e-5

Resultados para un learning rate de 2e-5:

Tomó 100 epochs de 381ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.6667	1	0.8	0.6667
BERT	1	1	**0.8182**	**0.7667**

Figura 10. Performance Bi-LSTM y BERT model learning rate 2e-5

Resultados para un learning rate de 3e-5:

Tomó 100 epochs de 390ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.6667	1	0.8	0.6667
BERT	0.8125	1	**0.8163**	**0.7333**

Figura 11. Performance Bi-LSTM y BERT model learning rate 3e-5

Resultados para un learning rate de 4e-5:

Tomó 100 epochs de 390ms cada uno para Bi-LSTM.

Tomó 12 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.6667	1	0.8	0.6667
BERT	1	0.95	**0.7895**	**0.6667**

Figura 12. Performance Bi-LSTM y BERT model learning rate 4e-5

Resultados para un learning rate de 5e-5:

Tomó 100 epochs de 390ms cada uno para Bi-LSTM.

Tomó 11 epochs de 5s cada uno para el modelo BERT.

Model	Precision	Recall	F1	Accuracy
Bi-LSTM	0.6667	1	0.8	0.6667
BERT	1	0.95	**0.7917**	**0.7333**

Figura 13. Performance Bi-LSTM y BERT model learning rate 5e-5

Se comparó la función de pérdida para el modelo Bi-LSTM con un learning rate de 1e-5, 2e-5, 3e-5, 4e-5 y 5e-5 y una distribución de 70%-30% entre los datos de entrenamiento y validación.

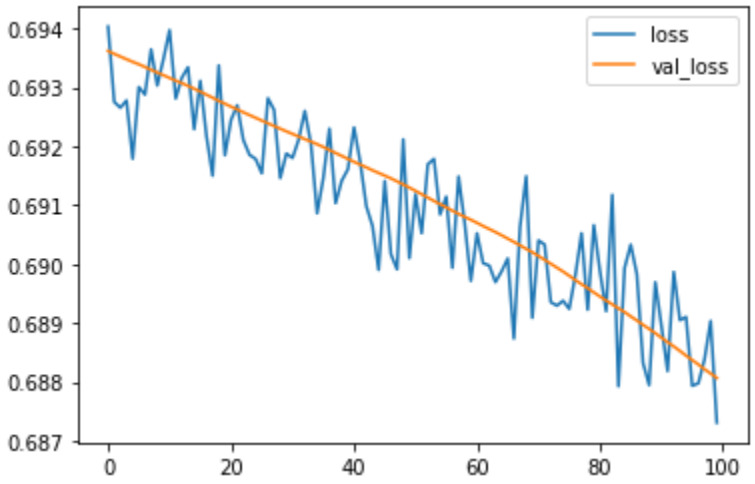

Figura 14. Loss Function Bi-LSTM train/valid 70-30, learning rate = 1e-5

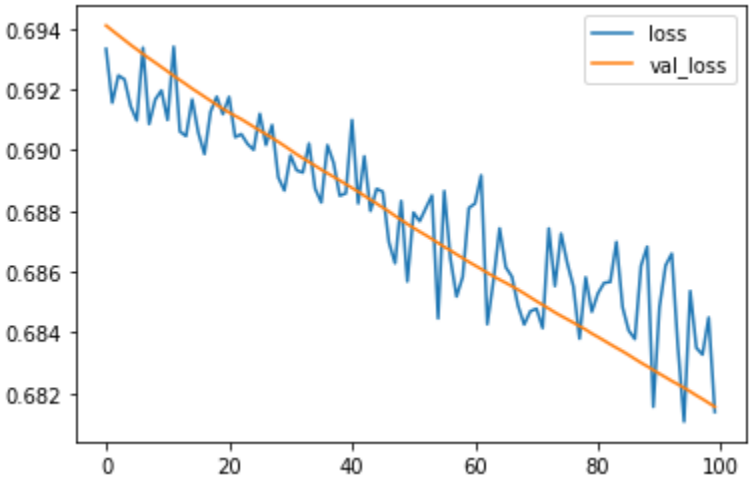

Figura 15. Loss Function Bi-LSTM train/valid 70-30, learning rate = 2e-5

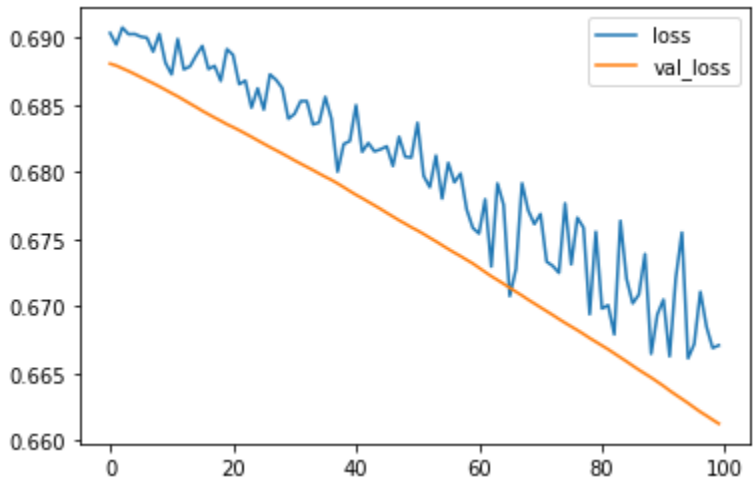

Figura 16. Loss Function Bi-LSTM train/valid 70-30, learning rate = 3e-5

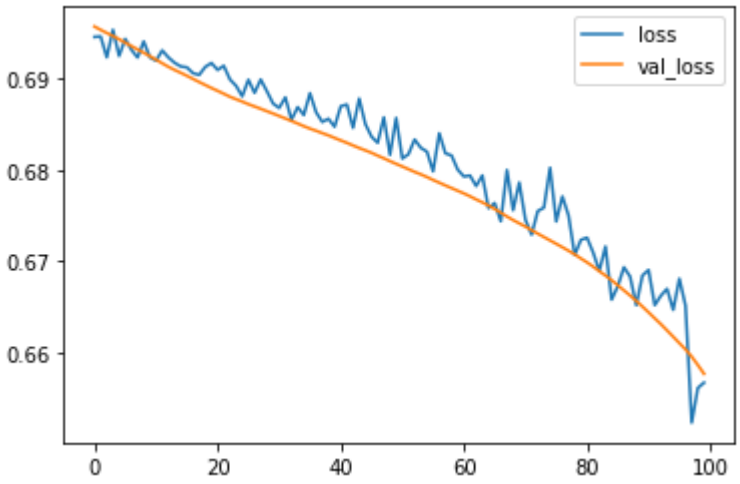

Figura 17. Loss Function Bi-LSTM train/valid 70-30, learning rate = 4e-5

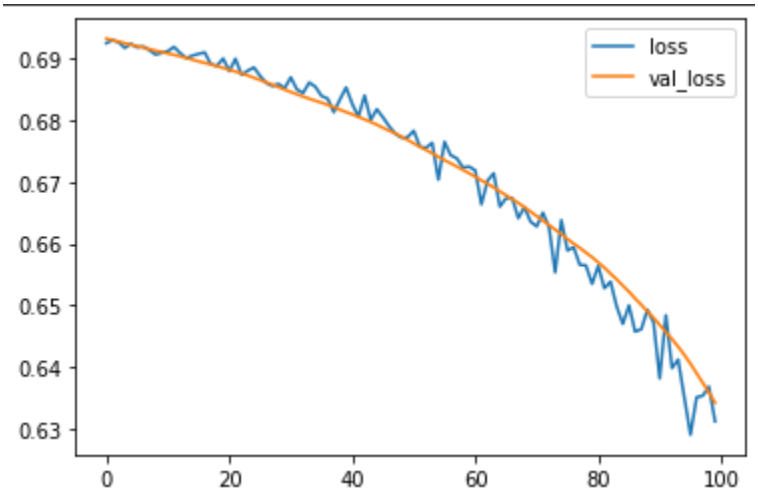

Figura 18. Loss Function Bi-LSTM train/valid 70-30, learning rate = 5e-5

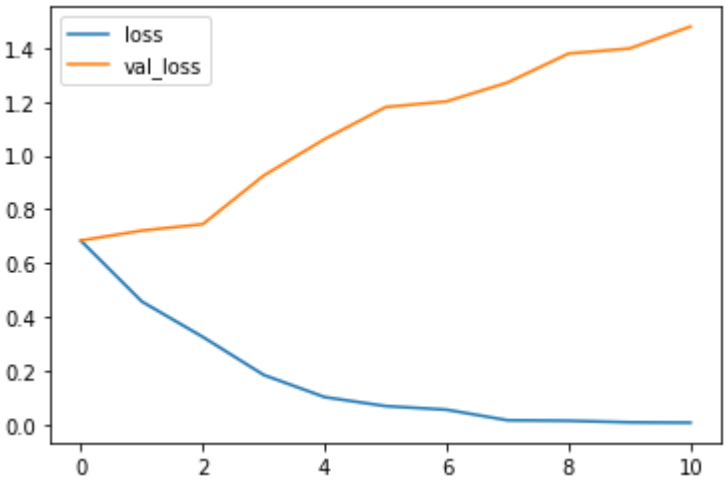

Figura 19. Loss Function BERT model train/valid 70-30, learning rate = 1e-5

4.2 Selección de mejor modelo

La **figura 18** indica el mejor valor para la *accuracy* y *f1 score* que obtuvo cada modelo durante las iteraciones, seguida de la distribución de datos de entrenamiento y validación, y el valor de *learning rate* (a) correspondiente a la iteración.

Model	Best accuracy	Best F1 score
Bi-LSTM	**0.6944** (70%-30%, a=1e-5)	**0.8360** (70%-30%, a=1e-5)
BERT	**0.7667** (70%-30%, a=1e-5)	**0.8511** (70%-30%, a=1e-5)

Figura 20. Best model scores

Para escoger el mejor modelo consideraremos 2 supuestos [7]:

- *Accuracy* funciona mejor cuando los verdaderos positivos y los verdaderos negativos son más importantes, mientras que *f1 score* funciona mejor cuando los falsos positivos y falsos negativos son cruciales.
- *Accuracy* funciona mejor cuando la distribución de clases es similar, mientras que *f1 score* es mejor cuando las clases están imbalanceadas.

Para nuestra aplicación los falsos positivos y falsos negativos son cruciales, y las clases están imbalanceadas, por lo cual escogeremos el modelo que haya presentado mejor *f1 score*, el cual fue el modelo BERT entrenado con un *learning rate* de 1e-5.

4.3 Evaluación

Se evaluó el mejor modelo escogido utilizando 6 comentarios. Clasificó el sentimiento de los comentarios correctamente en todos los casos, pero su decisión no fue tan segura en uno de los comentarios (score = 0.5991) evidenciando que tiene un gran margen de mejora.

Comentario	Sentimiento Real	Sentimiento Clasificado	Score
Sujeto estas lleno de odio	**NEGATIVO**	**NEGATIVO**	**0.7145**
Sujeto es el cambio verdadero	**POSITIVO**	**POSITIVO**	**0.8562**

Sujeto se va a robar el dinero del país	**NEGATIVO**	**NEGATIVO**	**0.7558**
Sujeto nos va a solucionar nuestros problemas	**POSITIVO**	**POSITIVO**	**0.8680**
Sujeto es un corrupto y nos va a hundir	**NEGATIVO**	**NEGATIVO**	**0.5991**
Sujeto es bueno administrando y traerá bien en su gobierno	**POSITIVO**	**POSITIVO**	**0.9425**

Figura 21. Best model Results

5. Conclusiones

- La distribución de datos de entrenamiento y validación de 70%-30% demostró ser el equilibrio perfecto para los datos. Al ser un dataset pequeño, ninguno de los lados podía perder información o ser menos representativo.
- Debido al pequeño tamaño del dataset, el modelo LSTM Bidireccional presentó un comportamiento errático a la hora de ajustarse a una representación adecuada, como se puede apreciar en la Figua 14. El modelo basado en BERT[8] en cambio pudo ajustarse con suavidad como se aprecia en la Figura 19, demostrando que es mejor extrayendo características.
- Las Figuras 14, 15, 16, 17 y 18 ponen en evidencia que un *learning rate* menor a 5e-5 puede no ser adecuado para el modelo LSTM Bidireccional. El modelo converge de forma más suave con *learning rates* superiores a 4e-5.

6. Enlace de los experimentos realizados

Los experimentos realizados se pueden encontrar en el siguiente repositorio: https://github.com/ppertuzduran/BERT_Sentiment_Classifier.git

7. Bibliografía

[1] Alva-Segura, D. A. (2021, 19 mayo). Análisis del Sentimiento Político en Twitter durante las Elecciones Congresales 2020 en el Perú. Reunir. https://reunir.unir.net/handle/123456789/11364Cuervo, M. C. (2019).

[2] PREDICCIÃ“N ELECTORAL USANDO UN MODELO HÃ• BRIDO BASADO EN ANÃ• LISIS SENTIMENTAL: ELECCIONES PRESIDENCIALES DE COLOMBIA. Redalyc. https://www.redalyc.org/journal/6078/607867636009/

[3] Editorial La República S.A.S. (2019, 2 agosto). Marketing Político e Inteligencia Artificial. Diario La República. https://www.larepublica.co/internet-economy/marketing-politico-e-inteligencia-artificial-2892638Gil, V. M. G. (2022a, marzo 12).
[4] Inteligencia artificial entrega proyección electoral de más de 100.000 perfiles en redes. W Radio. https://www.wradio.com.co/2022/02/15/el-voto-de-mas-100000-colombianos-inteligencia-artificial-entrega-proyeccion-electoral/Gil, V. M. G. (2022b, abril 5).

[5] Favorabilidad en redes de los candidatos de cara a las elecciones, según la inteligencia artificial. W Radio. https://www.wradio.com.co/2022/04/04/favorabilidad-en-redes-de-los-candidatos-de-cara-a-las-elecciones-segun-la-inteligencia-artificial/

[6] Heras, J. M. (2020, 9 octubre). Precision, Recall, F1, Accuracy en clasificación. IArtificial.net. https://www.iartificial.net/precision-recall-f1-accuracy-en-clasificacion/

[7] Huilgol, P. (2021, 11 diciembre). Accuracy vs. F1-Score - Analytics Vidhya. Medium. https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

[8] https://github.com/sagorbrur/codeswitch