**PIPELINE DE DISEÑO DE MODELO**

Se pidió realizar un modelo para una clasificación binaria de texto. Antes de realizar cualquier paso se hizo un análisis exploratorio de dato el cual arrojó los siguientes resultados:

1. Como era de esperarse los datos se encuentran sesgados hacia la clase “Orgánico” puesto que hay más tweets orgánicos que inorgánicos en twitter.
2. A pesar de que en el overview de la competencia se dice que son tweets existen 1182 tweets que tienen más de 280 caracteres. Ésto podría representar un problema en producción ya que los tweets no tienen más de 280 caracteres, esto afectaría la distribución entre entorno de producción y de entrenamiento y no tener un buen performance en producción.
3. Los tweets inorgánicos tienden a ser más largos, mientras que los orgánicos tienden a ser más cortos.
4. También hay en el dataset tweets en inglés, aunque si el modelo está expuesto a tweets en otros idiomas la cantidad de datos en dicho lenguaje no es suficiente.

Para remediar la situación se utilizó un ponderador de clases de la librería sklearn para enmendar el desbalance de clases.

Los modelos testeados fueron, Regresión Logística, SVM, Decision Trees, Random Forest y XGboots, los cuales reportaron performance entre 0.74 y 0.85 siendo XGboost el que arrojó mejor performance, además se puso a prueba un modelo de Fasttext muy fácil y rápido de crear el cual obtuvo un 0.86 en el score de test, debido a lo interesante de este último experimento se dejó en el notebook.

Con respecto a la longitud de los tweets se truncó la cantidad de tokens al momento de tokenizar el texto a 256 tokens, tweets más sufrieron padding, más largos truncados.

Se utilizó un modelo multi lenguaje por el hecho de manejar tweets en inglés y español, y tokens en varios idiomas.

Con respecto a los embeddings, no se empleó ningún embedding pre entrenado. La razón:

Los embedding suelen ser pre entrenados en un corpus particular, para tareas particulares. Esto podría representar un problema dado la particularidad de nuestro problema, el pre procesamiento podría no ser el mismo. Por tal motivo nuestro modelo tendrá que entrenar también su capa de embeddings.

Se realizó un baseline, con un modelo simple creado a partir de un LSTM y un clasificador con un performance usual.

Luego se realizó el preprocesamiento de los datos:

Remover los caracteres especiales, usuales en twitter además de emojis, números, y caracteres no alfanuméricos, esto deterioró el performance del modelo.

Se hizo ajuste de hyperparameters como Learning Rate, Batch Size, Epochs, Gradient Clip, sin mejora en el rendimiento del modelo.

Se puso a prueba otro modelo más complejo “ROBERTA-BASE” Con un performance igual de bajo.

Tomando en cuenta la literatura actual de NLP se tomaron en cuenta varias consideraciones para la prueba de hyperparameters y así reducir el tiempo de búsqueda:

* Como se tenían secuencias largas el número máximo de tokens fue de 256.
* Ya que nuestro número de tokens es el alto el batch size debe ser bajo 8 (previene el overfitting)
* Como optimizador se utilizó AdamW con learning rates [1e-6 3e-6 5e-6 3e-5 1e-5 1e-4 3e-4] concluyendo en 1-e4 para la primera fase de entrenamiento y 3e-6 para la segunda parte del entrenamiento. variando esto se consigue un cambio de hasta 10% en el performance del modelo.
* El número de épocas se puso desde 12 a 30 epocas resultando en 30 el mejor número.
* Función de coste CategoriacalCrossentropy con clases ponderadas por el desbalance.

El entrenamiento del modelo final fue el siguiente: Ya que se estaba trabajando con un modelo pre entrenado al cual le agregamos un clasificador, se entrenó primero el clasificador, congelando los pesos del modelo de “roberta”

Modelo: “roberta-base”

Épocas: 6

Learning Rate: 1e-4

Batch size: 8

Gradient Clip: 1

Weight Decay para el clasificador: 0.0001

Weight Decay para roberta: 0

Una vez finalizado nuestro “calentamiento” como se decidió llamar a esta fase del entrenamiento de todo el modelo, esta vez con un learning rate más bajo puesto que se descongelarán los pesos del modelo más grande.

Modelo: “roberta-base”

Épocas: 6

Learning Rate: 3e-6

Batch size: 8

Gradient Clip: 1

Weight Decay para el clasificador: 0.0001

Weight Decay para roberta: 0

Para hacer el la división de los datos validación se hizo en ratio 90% entrenamiento - 10% Validación. Sin embargo para el submission final cuando se garantizó el rendimiento del modelo se entrenó el modelo en el 100% de los datos y así el modelo pueda ver la mayor cantidad de datos posibles, esto aumentó un 1% el performance del modelo.

**CONCLUSIONES FINALES Y COMENTARIOS PERSONALES.**

Es una utopía como data scientist el querer tener los datos completamente perfectos, sin embargo al ser ellos el backbone de los modelos de Machine Learning se requieren unos estándares mínimos para ellos. Sin embargo, lo más curioso de todo es el desarrollo del modelo con los datos sin pre procesar. Hasta el día de hoy no tengo una respuesta clara de porqué esto ocurrió pero estaré trabajando en ello, aun después se haya acabado el challenge para descubrir la razón por la cual esto sucedió.