TP2: Críticas Cinematográficas - Grupo 08

Introducción

En este trabajo práctico vamos a utilizar una colección de críticas cinematográficas en idioma español y vamos a tratar de identificar cada crítica como positiva o negativa.

Descripción del dataset
Los datasets de train y test contaban originalmente con 50000 y 8599 filas respectivamente.

Dataset: train.csv

```
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   ID           50000 non-null  int64 
 1   review_es    50000 non-null  object
 2   sentimiento  50000 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.1+ MB
```

****

Dataset: test.csv

```
RangeIndex: 8599 entries, 0 to 8598
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   ID         8599 non-null   int64 
 1   review_es  8599 non-null   object
dtypes: int64(1), object(1)
memory usage: 134.5+ KB
```

Como vemos, en el conjunto de train tengo 3 variables (ID, review_es y sentimiento) y en el de test tengo 2 variables (ID y review_es). Todas las variables son **cualitativas**

- `ID` (nominal): ID de la reseña.
- `review_es` (ordinal): Critica emitida.
- `sentimiento` (nominal): Sentimiento que expresa la critica.

Eliminación de variables irrelevantes
Eliminamos la variable ID en el conjunto de train, dado que no cumple ninguna función. No así en el conjunto de test, dado que la necesitamos para la competencia de Kaggle.

Búsqueda de valores nulos
Al revisar ambos datasets notamos que afortunadamente no contaban con valores nulos.

Limpieza general
Realizamos una limpieza general de nuestro texto a fin de eliminar caracteres especiales, no alfabéticos, mayúsculas, etc. Para ellos realizamos las siguientes tareas:

- Convertimos el texto a minúsculas.
- Eliminamos los diacríticos.
- Reemplazamos los signos de puntuación por espacios.
- Dejamos únicamente los caracteres alfabéticos, espacios y la letra "ñ".
- Eliminamos las palabras que tengan un único carácter.
- Sustituimos los espacios en blanco múltiples por un único espacio.
- Buscamos las reseñas que estaban en otros idiomas que no sean el español y las eliminamos.

Lematización
Realizamos una lematización del dataset de train.

Tokenización y eliminación de Stopwords
Eliminamos las stopwords y tokenizamos el conjunto de train.

Resultado
De todo este trabajo de limpieza y exploración, extrajimos los siguientes datasets:

- **train_reducido** ➡️ Dataset base, se le eliminó la columna ID y la columna review_es se reemplazó por texto_codificado
- **train_limpio** ➡️ Se formateo el texto y se eliminaron las críticas en otros idiomas.
- **train_lematizado** ➡️ Se partió de train_limpio y se lematizo el texto.
- **train_tokenizado** ➡️ Se partió de train_limpio y se tokenizo el texto y se eliminaron las stopwords
- **train_final** ➡️ Se partió de train_lematizado y se tokenizo el texto y se eliminaron las stopwords

Cuadro de Resultados

Realizar un cuadro de resultados comparando los modelos que entrenaron seleccionando los que a su criterio obtuvieron la mejor performance en cada caso. Deben indicar cuál es el que seleccionaron como mejor predictor de todo el TP. Confeccionar el siguiente cuadro con esta información:

Medidas de rendimiento en el conjunto de TEST:

- F1
- Precision
- Recall
- Métrica XXX (Cualquier otra que consideren relevante)
- Resultado obtenido en Kaggle.

Modelo	F1-Test	Precision Test	Recall Test	Accuracy	Kaggle
Bayes Naive	0.840	0.845	0.840	0.84	.72649
Random Forest	0.840	0.845	0.845	0.84	.71835
XGBoost	0.820	0.820	0.820	0.820	.73870
Red Neuronal	0.870	0.870	0.870	0.870	.73987
Ensamble	0.820	0.820	0.820	0.820	.76623

Descripción de Modelos

Naive Bayes: es un clasificador que se basa en el teorema de Bayes y asume que las palabras en un texto son independientes entre sí. Utiliza esta idea para estimar la probabilidad de que un texto pertenezca a una categoría específica basándose únicamente en las palabras que contiene, sin considerar cómo se relacionan esas palabras entre sí dentro del texto. El único hiper parámetro es el alpha para Laplace Smoothing, el cual por recomendación de la cátedra se tomó como uno.

Random Forest: es un método de aprendizaje automático donde se construyen múltiples árboles de decisión no podados, cada uno entrenado con diferentes partes del conjunto de datos mediante particiones aleatorias de columnas y observaciones. Luego, se selecciona el mejor árbol y se repite el proceso para obtener varios modelos clasificadores. Cuando se presenta una nueva observación, cada modelo en el conjunto clasifica la observación y la decisión final se basa en la mayoría de las clasificaciones de los modelos individuales.

XGBoost: Es un algoritmo de aprendizaje automático basado en boosting que construye un conjunto de árboles de decisión de manera secuencial. Cada árbol

se centra en corregir los errores de los árboles anteriores. Además, incorpora términos de regularización en la función de pérdida para controlar el sobreajuste y mejorar la capacidad del modelo para generalizar a datos nuevos y no vistos.

Redes Neuronales Recurrentes: es una arquitectura especializada en el procesamiento de secuencias temporales. Estas redes están diseñadas con conexiones que permiten retroalimentación, lo que significa que pueden recordar y utilizar información procesada anteriormente. Esto las hace ideales para problemas donde el orden de los elementos es fundamental, como en la comprensión de sentimientos en textos o la generación de texto coherente.
Para el trabajo utilizamos una red GRU (Gated Recurrent Unit) la cual es un tipo de red neuronal recurrente que se utiliza principalmente en problemas de procesamiento de secuencias, como el análisis de texto. Se diferencia de otras redes recurrentes como las LSTM por ser más simple y eficiente, computacionalmente.

Cabe mencionar que al realizar el proceso de búsqueda de los mejores hiperparamétos para este modelo, el proceso tomó 2041 minutos (más de 34 horas) con lo cual no pudimos hacer demasiadas pruebas jugando con los parámetros de la grilla.

Debajo adjuntamos el summary de las redes que entrenamos:

Model: **Red Neuronal sin optimizar**

_____

Layer (type)	Output Shape	Param #
embedding_5 (Embedding)	(None, None, 128)	1920000
gru_10 (GRU)	(None, None, 128)	99072
gru_11 (GRU)	(None, 128)	99072
dropout_5 (Dropout)	(None, 128)	0
dense_5 (Dense)	(None, 1)	129

Total params: 2,118,273
Trainable params: 2,118,273
Non-trainable params: 0

Model: **Red Neuronal optimizada**

_____

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	(None, None, 128)	1920000
gru_4 (GRU)	(None, None, 64)	37248
gru_5 (GRU)	(None, 64)	24960
dropout_2 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 1,982,273
Trainable params: 1,982,273
Non-trainable params: 0

Model: **Red Neuronal semi compleja con datos sin procesar**
Model: "sequential"

_____

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 384)	56711808
dense_1 (Dense)	(None, 1)	385

Ensamble : Votación equitativa (bagging) compuesta por el modelo de cada tipo con mayor puntaje en Kaggle, es decir:

1. Bayes naive entrenado con dataset final preprocesado
2. Random Forest con dataset sin preprocesar
3. XG Boost con dataset lematizado (el modelo de cada tipo con mejor puntaje en Kaggle).

La misma se implementó por medio de una función en Python debido a la imposibilidad de utilizar modelos pre entrenados con los ensambles de votación de Scikit learn.

Conclusiones generales

¿Fue útil realizar un análisis exploratorio de los datos? Sí, de hecho, más que útil, consideramos que fue fundamental para poder resolver el problema.

¿Las tareas de preprocesamiento ayudaron a mejorar la performance de los modelos? Sí, observamos mejores resultados en los modelos que fueron entrenados con datasets mejor procesados.

Aunque cabe destacar que utilizar la información en crudo no resultó para nada mal (de hecho, si se busca una clasificación rápida y de bajo costo, es lo más recomendable). Los resultados sin limpieza estuvieron entre .02 y .04 puntos por debajo de todos los demás.

¿Cuál de todos los modelos obtuvo el mejor desempeño en TEST? La red neuronal fue la que obtuvo los mejores resultados.

¿Cuál de todos los modelos obtuvo el mejor desempeño en Kaggle? De todos los modelos entrenados, el ensamble fue lejos el mejor de todos. Nuestra hipótesis al respecto es que logró una mejor cobertura del problema debido a la diferencia de aprendizaje entre los distintos modelos.

¿Cuál fue el modelo más sencillo de entrenar y más rápido? ¿Es útil en relación al desempeño obtenido? Bayes Naive fue por lejos el más sencillo de entrenar, debido a que tiene un solo hiper parámetro que se tomó en su valor predeterminado debido a recomendaciones de la cátedra respecto a Laplace Smoothing.

Por el contrario, las redes neuronales fueron naturalmente las más complejas. El entrenamiento es computacionalmente costoso y lograr una optimización que mejore el fit fue complicado.
La optimización de hiper parámetros no resultó de gran utilidad, en ciertos casos los modelos mejoraron y en algunos empeoraron (sensiblemente, no en gran magnitud). Creemos que se debe al grado de overfit alcanzado por los modelos (el ensamble es el que peor califica en la prueba local, pero con mayor puntaje en Kaggle). Nos resultó sumamente llamativa la diferencia entre la performance en Kaggle y la prueba local.

¿Cree que es posible usar su mejor modelo de forma productiva? No.
Consideramos que nuestros modelos han sido entrenados con una cantidad escasa de datos, con lo cual no serían funcionales en un entorno de producción real.

¿Cómo podría mejorar los resultados? Aumentando la cantidad de datos de training. Ademas, agregar modelos más diversos podría resultar en una reducción

de la desviación y overfit. También podríamos mejorar las redes con más entrenamiento y/o modificando la arquitectura tal que tenga una serie de capas ocultas dedicadas al procesamiento y luego la “pirámide” que genera la salida.

Tareas Realizadas

Teniendo en cuenta que el trabajo práctico tuvo una duración de 9 (nueve) semanas, le pedimos a cada integrante que indique cuántas horas (en promedio) considera que dedicó semanalmente al TP

Integrante	Principales Tareas Realizadas	Promedio Semanal (hs)
Pratto, Federico	Preprocesamiento XGBoost Random Forest Red neuronal optimizada Realización del informe	6
Testa, Santiago	Bayes Naive Random Forest Red neuronal básica Ensambles Realización del informe	6