TP2: Críticas Cinematográficas - Grupo 03

Introducción

Acá, se nos encarga determinar si un listado de textos, cuyo origen es el de reseñas cinematográficas, corresponde a un sentimiento "positivo" o "negativo". Para ello *vectorizamos* las palabras del texto de cada reseña en un estilo del algoritmo *bag of words*.
El dataset en sí es simple: consta de 3 columnas (el **id** de la reseña, el **texto** de la reseña en español, y el **sentimiento** que le corresponde), y 50000 filas que siguen este patrón.

Al vectorizar las palabras, encontramos que el resultado excedía por mucho lo que el hardware a mano permitía como poder de cálculo, por lo que recortamos el dataset de la siguiente manera:

- Sólo tomamos en cuenta las palabras que aparecen como **mínimo** en el *0.1%* del total de textos. Palabras demasiado poco frecuentes sobre poblaban los datos con información que no resultaba tanto la pena al final.
- También sólo consideramos las palabras que sólo aparecen como **máximo** en el 75% de los documentos. Si son *demasiado* frecuentes, entonces no son un buen factor para decidir para empezar.
- En caso de que todavía demasiados datos entrasen en los parámetros de arriba, sólo se tomarían en cuenta las **25000** palabras más frecuentes.

Ahora, hemos de suponer que las reseñas no tuvieron sutilezas o sarcasmos que el algoritmo no supo detectar.

Cuadro de Resultados

Modelo	F1-Test	Precision	Recall	Accuracy	Kaggle	Hiperparámetros	Retoques al Dataset
Bayes Naive	0.852702	0.844179	0.8614	0.8512	0.73638	**alpha=**1.0 **force_alpha=**False	**min_df=**0.001 **max_df=**0.8 **max_features=**25000
Random Forest	0.832105	0.828639	0.8356	0.8314	0.72203	**n_estimators=**70 **min_samples_split=**12 **min_samples_leaf=**2 **max_features=**"log2" **criterion=**"gini" **random_state=**1	**min_df=**0.001 **max_df=**0.75 **max_features=**25000
XGBoost	0.859719	0.851786	0.8678	0.8584	0.71215	**n_estimators=**100 **max_leaves=**0 **max_depth=**19 **learning_rate=**0.22857143 **reg_lambda=**1.314789474 **gamma=**3.6734693877551 **reg_alpha=**1.05263157895	**min_df=**0.001 **max_df=**0.75 **max_features=**25000
Red Neuronal	0.864525	0.862460	0.8666	0.8642	0.77805	*(Capas)* **Dense(**150, "relu"**)** **Dense(**15, "relu"**)** **Dense(**1, "sigmoid"**)** *(Compilación)* **optimizer=**Adagrad **loss=**"binary_crossentropy" **metrics=**["AUC"]	**min_df=**0.001 **max_df=**0.75 **max_features=**25000
Ensamble	0.88999	0.880579	0.8996	0.8888	0.72901	**meta_modelo=**LogisticRegressionCV() **modelos=**[rf, xgb, rn, rf_2, xgb_2, rn_2] **passthrough=**True **cv=**KFold(**n_splits=**5)	**min_df=**0.001 **max_df=**0.75 **max_features=**25000

Descripción de Modelos

Como se puede ver, nuestro mejor modelo fue una iteración de una red neuronal. Los mejores resultados fueron dados por pocas capas, y éstas a su vez no tuvieron una cantidad considerable de neuronas por capa. Particularmente, al entrenarlo, se hizo con los parámetros **epochs=**150 y **batch_size=**100, por lo que puede que pocas iteraciones hayan prevenido el *overfitting*: es entonces que se siguió probando con valores bajos.

Conclusiones generales

Como se explicó anteriormente, la única tarea de preprocesamiento hecha fue la de recortar el dataset de manera que se dejaron afuera las palabras demasiado o demasiado poco frecuentes. Lo mismo demostró dar mejores resultados también: un ensamble de tipo *"stacking"* obtuvo el mejor desempeño en el dataset de testeo local (rozando el 0.89), mientras que el mejor en Kaggle fue la red neuronal (con un valor cerca de 0.78).
Somos conscientes de que hay una diferencia no despreciable entre el desempeño local y el evaluado en Kaggle; esto es claro indicio de *overfitting*, y se logró reducir hasta cierto punto con los recortes antes mencionados.

Similarmente, dichas optimizaciones permitieron acelerar el tiempo de cálculo en los entrenamientos: el modelo de Bayes Naïve tenía problemas de rendimiento de esa índole que fueron así tratados (si bien después no resultó ser el modelo con mejores resultados).

Aun así, creemos decente el desempeño de los modelos y capaces de ser utilizados en un ámbito productivo. Bien es cierto que se podría haber mejorado tratando de explorar posibles outliers en las palabras vectorizadas o encontrar el balance de recortes óptimo probando otros valores.

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Franco Lighterman Reismann	2.5
Marcos García Neira	2
Martín Andrés Maddalena	2