TP2: Críticas Cinematográficas - Grupo Alan Taylor

Introducción

En este caso nos pedían dada una review decidir si es positiva o negativa, el dataset contaba con 50000 registros, exactamente 50% de reseñas positivas y el otro 50% negativas, y 3 columnas, id, review y sentimiento.

Para el preprocesamiento de los datos primero "normalizamos" todas las reviews.

Este proceso de normalización consta de 4 pasos:

- Eliminar citas: Eliminamos todos los textos escritos entre comillas, ya que pueden provocar una confusión en los modelos al tratarse de una ironía o cita u otro tipo de frase que provoque confusión.
- Limpieza de comienzo con mayúsculas: Reemplazamos las letras iniciales con mayúsculas por minúsculas ya que nos interesa saber que tanto aparece una palabra y nos es indiferente si es usada al comienzo de una oración o párrafo. Eso sí, antes verificamos que no sea una palabra escrita a propósito en mayúsculas, esas las dejamos como están, o sea en definitiva: **Critica = critica**, pero **CRITICA ≠ crítica.**
- Reemplazo por comas: Reemplazamos todos los signos por comas, ya que nos interesa saber si hay signos pero no nos interesa saber cual. En definitiva: **! = ¿ = . = ; = ,**.
- Limpiar espacios: Esto lo hacemos para dejar prolijo el texto, ya que con las anteriores modificaciones pueden llegar a quedar palabras concatenadas y lo agregamos para evitar cualquier tipo de error.

(Supuestos en **negrita**)

PD: También en este apartado intentamos utilizar un método visto en clase que consta de agregar un "NO_" detrás de cada palabra en la oración en la que aparece un no dentro de la review. Este método daba resultados un poco menos efectivo que sin hacerlo, y de hecho los modelos hacen mejores predicciones sin siquiera tener la palabra no en las oraciones, dato que nos llamó la atención (esto sucedió debido a que la biblioteca de stop words contenía dicha palabra, y no fue intencional pero dio mejores resultados).

Cuadro de Resultados

Medidas de rendimiento en el conjunto de TEST:

Modelo	F1-Test	Precision Test	Recall Test	Accuracy	Kaggle
Bayes Naive	0.8340	0.8616	0.8082	0.8386	0.7869
Random Forest	0.8499	0.8346	0.8658	0.8466	0.7582
XG Boost	08369	0.8130	0.8623	0.8314	0.7216
Red Neuronal	0.8808	0.8750	0.8866	0.8796	0.6012
Voting	0.8602	0.8522	0.8683	08584	0.7582

Descripción de Modelos

Las técnicas de preprocesamiento fueron las mismas para todos los modelos (exceptuando la red neuronal), además de los ya mencionados utilizamos el método CountVectorizer de scikit-learn para armar nuestra bag of words con las 5000 palabras más frecuentes en las reviews y se utilizó .en Random Forest, XG Boost y en el ensamble Stacking, para los demás utilizamos tfidfVectorizer con un max feature de 6000. En el caso de la red neuronal utilizamos un Tokenizer directamente con el texto.

Hiperparametros:

Bayes-Naive Multinomial: alpha = 1.7,
class_prior = [0.4,0.5] **MEJOR PREDICTOR.**

Se basa en contar cuántas veces aparecen las palabras en los documentos y utiliza este recuento para calcular la probabilidad de pertenecer a una categoría específica. Es simple pero efectivo, especialmente en problemas de clasificación de texto.

XGBoost: n_estimators=500,
max_depth=13,
learning_rate=0.1,
subsample=1,
colsample_bytree=0.5,
gamma=0.2,
objective="reg:logistic",
random_state=20

.Red Neuronal: La red neuronal que hicimos está conformada por 6 capas más los dropouts. La capa embedding que se utiliza para convertir secuencias de palabras (representadas como índices) en vectores densos de valores de punto flotante. Y ayuda a representar semánticamente las palabras de manera más significativa en un espacio dimensional. Luego las capas LSTM que son capas recurrentes que procesan secuencias de datos. En nuestro modelo, utilizamos tres capas LSTM. Luego cada DropOut, que ayudan a prevenir el sobreajuste al apagar aleatoriamente algunas neuronas durante el entrenamiento. Utilizamos dropout después de cada capa LSTM para regularizar la red y mejorar la generalización. Por último las Dense que están completamente conectadas. Agregamos dos capas densas después de las capas LSTM. La primera tiene 128 neuronas con función de activación ReLU, y la segunda tiene 1 neurona con activación sigmoide, que es utilizada para problemas de clasificación binaria cómo es el caso. En este apartado queremos aclarar que a lo largo de todo el trabajo intentamos muchas formas y variantes de redes neuronales (utilizando embeddings, redes pre entrenadas, más/menos capas, más/menos epochs, etc) pero ninguna dió un buen resultado que sea efectivo en el mundo real.

Random Forest: n_estimators = 550,
bootstrap = True,
max_depth = 25,
max_features = 20,
min_samples_split = 2,
min_samples_leaf = 2,
random_state = 65.

Voting: Modelos: Bayes Naive: Bernoulli, Multinomial, Regresión Logística.

Conclusiones generales

Ya que nos dieron un dataset equilibrado al tener 50% de reviews positivas y negativas no fue necesario hacer mucho análisis exploratorio de los datos, lo que sí resultó muy favorable hacer un buen pre-procesamiento de los datos para luego obtener una buena bag of words, lo que nos permitió que los modelos dieran mejores resultados.
En cuanto a los modelos nos sorprendió que Bayes-Naive fue de los que tenía peores resultados en el test pero el que mejor resultado obtuvo en Kaggle, cabe recalcar que también fue uno de los modelos más sencillos de entrenar. Y por otro lado Stacking fue el modelo de los peores resultados en Kaggle, a pesar de ser un ensamble generalmente efectivo.

Creemos que el resultado obtenido es bastante bueno pero no sabemos si tanto como para aplicarlo en un entorno productivo, al mismo tiempo pensamos que aplicando mejor las redes neuronales y los ensambles podríamos mejorar bastante la predicción y así poder aplicarlo de forma más productiva.

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Estefano Polizzi	5 hs
Santiago Trezeguet	3 hs
Ignacio Oviedo	3 hs