Organización de Datos (75.06)
Trabajo Práctico 2: Críticas cinematográficas
Cuatrimestre, año y grupo: 2ºC 2023
Grupo: 11 - Los Pandas
Corrector: Mateo Suster
Fecha de entrega: 2 de diciembre de 2023
Datos de los integrantes:

Labollita, Francisco	106.861	flabollita@fi.uba.ar
Mundani Vegega, Ezequiel	102.312	emundani@fi.uba.ar
Otegui, Matías Iñaki	97.263	motegui@fi.uba.ar

Introducción

En este trabajo práctico se realiza un análisis de sentimiento de diferentes críticas cinematográficas, donde se busca entrenar un modelo que prediga si una crítica fue positiva o negativa a partir de un conjunto de entrenamiento con varias críticas en español que ya están clasificadas.

El dataset de entrenamiento es uno balanceado que tiene 50.000 críticas, teniendo misma cantidad de positivas que de negativas. El dataset de prueba tiene 8.599 críticas en español. Algo que resultó interesante es que a pesar que las críticas son en español, varias contenían palabras en otros lenguajes, incluso en otros alfabetos como del hebreo o el islandés.

Modificaciones sobre el dataset

1. Se realiza un HTML Strip (eliminar etiquetas HTML en caso que existan).
2. Se reemplazan caracteres extraños que no sean letras (con o sin tilde) o números por un espacio.
3. Se eliminan los espacios al principio y al final de cada review.
4. Se eliminan los espacios sobrantes entre palabras, no debe haber más de uno.
5. Se convierte todo a minúsculas.
6. Se eliminan las *stopwords* en español con la librería spaCy.
7. Se realiza un proceso de lematización de las palabras.
8. Se eliminan los tildes.

Al momento de decidir qué limpieza sobre el dataset realizar, primero se consideró transformar todas las mayúsculas en minúsculas. Por un lado es cierto que se perdía cierta información dado a que "HERMOSO" pareciera tener una connotación más positiva que "hermoso", aunque por otro lado se ganaba información principalmente con las palabras que iniciaban una oración, de manera que "Hermoso" y "hermoso" sí debían tener la misma connotación. Esto se comprobó al entrenar modelos con ambos datasets.

Se eliminaron los caracteres que no eran letras y se quitaron los tildes, esta última decisión principalmente porque se suelen cometer errores de ortografía.

Utilizando la librería spaCy se eliminaron las *stopwords* y se lematizó el texto. También se probó el proceso de stemming, pero no otorgó resultados favorables, por lo que se descartó.

Todas estas decisiones además de suponer que mejoran la precisión del modelo a entrenar, también reducen el tiempo que se tarda en los entrenamientos por disminuir la cantidad total de diferentes "palabras". Finalmente se exportaron dos nuevos .csv con las modificaciones hechas, uno para train, de nombre `train_clean.csv` y otro para test de nombre `test_clean.csv`.

En base a todo lo anterior, se implementó un *bag of words*, que será el conjunto de entrada de los diferentes modelos.

Cuadro de resultados para el conjunto test

Modelo	F1 Score	Precision	Recall	Accuracy	Kaggle
Bayes Naive	**0,862**	**0,850**	**0,874**	**0,860**	**0,762**
Random Forest	0,830	0,786	0,880	0,821	0,744
Red Neuronal	0,828	0,894	0,771	0,841	0,730
Ensamble	0,859	0,843	0,876	0,855	0,729
XGBoost	0,846	0,840	0,853	0,844	0,705

Cuadro 1: Tabla de resultados

Descripción de modelos

Bayes Naive

El modelo Bayes Naive fue el único modelo nuevo introducido en este TP. Está basado en el Teorema de Bayes y es utilizado generalmente para la clasificación de textos sin considerar el nombre de las palabras (por eso naive). Se experimentó combinando distintos clasificadores y vectorizadores y se encontró que la mejor combinación se lograba utilizando el Tfidf Vectorizer y MultinomialNB. Con un Grid Search se optimizaron los siguientes hiperparámetros: mnb_alpha (0.1), tfidf maxfeatures (100.000) y tfidf_ngram_range (1, 2).

Random Forest

El clasificador Random Forest construye múltiples arboles de decisión en paralelo durante el entrenamiento y luego combina predicciones para obtener un solo resultado. Junto a un Count Vectorizer, se utilizó un Grid Search para optimizar los siguientes hiperparámetros: min_samples_leaf (1), criterio (entropy), min_samples_split (10), n_estimators (300), max_depth (12) y ccp_alpha (0,001). Muchos de estos hiperparámetros fueron estudiados y analizados en el trabajo práctico anterior.

XGBoost

El modelo XGBoost utiliza el método de boosting para construir un modelo predictivo a partir de varios modelos débiles de manera secuencial, enfocándose en corregir los errores de predicción en cada etapa. Mediante un Grid Search se encontró que los siguentes hiperparámetros mejoraban al modelo: colsample_bytree (1), learning_rate (0,4), max_depth (6), n_estimator (200) y subsample (1).

Redes neuronales

Se experimentó con dos Redes Neuronales, una con solamente una capa escondida y otra con 3 capas escondidas. Para la primera, se utilizó una capa Embedding, LSTM y finalmente una Dense. En la de 3 capas, se agregaron dos capas Dropout antes y después de la capa LSTM para prevenir el sobreajuste en el modelo. En ambos casos se utilizó un Grid Search y para el mejor modelo se obtuvo: batch_size (64), epochs (3), model_dropout_rate (0.3) y model_optimizer (adam).

Ensamble Voting

Finalmente, se decidió utilizar el ensamble tipo Voting que combina las predicciones de varios modelos para obtener una predicción final. Para esto, se combinaron tres modelos anteriores: Bayes Naive, Random Forest y XGBoost. Se utilizaron los hiperparámetros hallados anteriormente.

Conclusiones generales

El análisis exploratorio inicial fue importante, pero no tuvo la misma relevancia que en el TP 1, ya que había menos variables para analizar y comparar entre sí. Aunque vale aclarar que sí mejoraron los resultados del modelo predictivo. La eliminación de símbolos no pertenecientes al alfabeto español y otros ajustes que uniformizaron los datos brindaron una mejora en las predicciones.

El modelo Bayes Naive fue el mejor para la resolución de este problema. Brindó mejores resultados en el Test y en el Kaggle y no surgieron dificultades durante su creación y optimización. Esto es esperable, ya que se utiliza con frecuencia para la clasificación de textos.

Se podría haber experimentado con diferentes librerías para el procesamiento de los datos. Por dar un ejemplo, se encontró la librería TextBlob que analiza el sentimiento de textos en valores numéricos, otorgando un rango de positividad a la review en vez de clasificarla solamente como “positiva” o “negativa”. El problema de utilizar esta librería fue que a veces predecía reviews con valores numéricos negativos, cuando el sentimiento del train daba positivo (o viceversa), probablemente porque se utilizó otro método para clasificar las reviews de train. Sin embargo, vale la pena destacar que algunas reviews pueden tener un significado más bien neutral y no siempre serán negativas o positivas, ya que existen grados de positividad/negatividad.

Otra estrategia que se podría haber utilizado es realizar un one-hot encoding donde cada atributo sería la palabra y el valor de la celda si aparece o no.

El modelo entrenado podría ser utilizado de manera productiva en el contexto de predicción de si una crítica cinematográfica es positiva o no, pero no se ve que haya un lugar en el mercado para una aplicación de esta envergadura, dado a que hoy en día las críticas tienen un puntaje asociado a la película, ya sea sobre 5, sobre 10, sobre 100 o con cantidad de estrellas o butaquitas; por lo que no tendría sentido predecir algo que ya se conoce.

Distribución de tareas

Integrantes	Promedio semanal (hs)
Labollita, Francisco	16
Mundani Vegega, Ezequiel	16
Otegui, Matías Iñaki	16