Trabajo Práctico 2
Críticas Cinematográficas

[75.06/95.58] Organización de Datos
Primer cuatrimestre de 2023

ALUMNO	PADRON	CORREO
BARTOCCI, Camila	105781	cbartocci@fi.uba.ar
LOURENGO, Lucía	104880	llourengo@fi.uba.ar
PATIÑO, Franco	105126	fpatino@fi.uba.ar

Preprocesamiento del conjunto de entrenamiento

El preprocesamiento aplicado sobre el dataset de entrenamiento incluye la eliminación de stopwords y la transformación a minúsculas de todos los caracteres.
En todos los casos, hemos realizado los entrenamientos de los modelos y las predicciones con las etiquetas de sentimiento 'negativo' y 'positivo' como 0 y 1.
Por otro lado, para la representación de bag of words utilizamos la técnica de Count Vectorizer y también probamos con Term Frecuency-Inverse Document Frecuency (TF-IDF) para convertir el texto en características numéricas, teniendo en cuenta la importancia relativa de las palabras en el conjunto de documentos. Adicionalmente, utilizamos n-gramas para capturar relaciones entre secuencias de palabras. En general, obtuvimos mejores métricas utilizando TF-IDF junto con unigramas, bigramas y trigramas.
Por último, aplicamos las técnicas de lematización y de stemming sobre el dataset de entrenamiento y el de test, generando nuevos datasets con estos preprocesamientos. Decidimos probar con ellos para reducir la variabilidad de las palabras, agrupando variantes a la forma base y reduciendo la dimensionalidad del espacio de características y la interpretación y análisis de los textos. En particular, obtuvimos mejores resultados con lematización (exceptuando en Redes Neuronales, donde funcionó mejor el dataset original).

Bayes Naïve

El mejor modelo Bayes Naïve obtenido fue implementado con MultinomialNB y tiene los siguientes hiperparámetros:

- alpha = 1 (suavizado Laplace)
- force_alpha = False
- fit_prior = True

En cuanto a las métricas en entrenamiento, obtuvimos:

- Accuracy de 0.878
- Precisión de 0.878
- Recall de 0.878
- F1 de 0.877

En Kaggle, la metrica obtenida es 0.71699.

Random Forest

El mejor modelo Random Forest obtenido tiene los siguientes hiperparámetros:

- n_estimators = 300
- min_samples_split = 20
- min_samples_leaf = 1
- max_depth = 35
- criterion = entropy

En cuanto a las métricas en entrenamiento, obtuvimos:

- Accuracy de 0.8531
- Precisión de 0.8534

- Recall de 0.8529
- F1 de 0.8530

En Kaggle, la metrica obtenida es 0.73773.

XGBoost

El modelo XGBoost predeterminado nos dio los mejores resultados frente a aquellos con hiperparámetros optimizados. Algunos de los hiperparámetros son:

- max_depth = 6
- learning_rate = 0.3
- n_estimators = 100
- gamma = 0
- subsample = 1

En cuanto a las métricas en entrenamiento, obtuvimos:

- Accuracy de 0.8444
- Precisión de 0.8444
- Recall de 0.8443
- F1 de 0.8443

En Kaggle, la metrica obtenida es 0.71099.

Ensamble de Modelos

Hemos probado ensambles híbridos con distintas combinaciones de modelos y usando tanto las técnicas de Voting como de Stacking. Las mejores métricas tanto en entrenamiento como en la competencia las obtuvimos con un ensamble Stacking compuesto por un SVM Lineal, un Random Forest y un Bayes Naïve como modelos base y una Regresión Logística como modelo predictor. Los mejores hiperparámetros obtenidos son:

- alpha = 1 (Bayes Naïve)
- n_estimators = 300 (Random Forest)
- max_depth = 10 (Random Forest)
- C = 1 (SVM Lineal)
- C = 1 (Regresión Logística)

En cuanto a las métricas en entrenamiento, obtuvimos:

- Accuracy de 0.9017
- Precisión de 0.9018
- Recall de 0.9016
- F1 de 0.9016

En Kaggle, la metrica obtenida es 0.74239.

Red Neuronal

Hemos entrenado distintas arquitecturas de Redes Neuronales, probando capas densas, recurrentes y convolucionales y combinaciones entre ellas. Para la métrica a optimizar, utilizamos nuevamente la función 'get_f1' del TP1 para poder optimizarla en el entrenamiento (basada en la implementación que se muesta en este artículo).
En cuanto al entrenamiento de las redes, notamos que performaba mejor con el dataset de entrenamiento original (es decir, sin haber aplicado lemmatización ni stemming). Por otro lado, probamos utilizar el preprocesamiento hecho para los demás modelos pero los tiempos de entrenamiento eran excesivamente grandes (hasta 2 horas por época), probablemente debido al tamaño del vocabulario, por lo que decidimos aplicar otro preprocesamiento para el entrenamiento de redes neuronales. Las tareas realizadas en el preprocesamiento previos a la tokenización fueron eliminar stopwords, acortar las críticas a 600 caracteres (ya que generalmente es posible conocer el sentimiento de una crítica en las primeras oraciones) y por último hicimos una reducción del vocabulario, conservando las 10000 palabras con más ocurrencias en las críticas para así simplificar los datos.
La red neuronal que mejor performó tiene la siguiente arquitectura:

- Capa de Embedding: la utilizamos para convertir las palabras en vectores de longitud fija. Al utilizar una dimensión de 128, se permite que el modelo capture relaciones semánticas y contextuales entre las palabras. Además, indicamos el tamaño del vocabulario (10000 palabras) que contiene las palabras más utilizadas como se explicó anteriormente
- Capa de Dropout espacial: la añadimos para reducir el sobreajuste aplicando dropout a la secuencia de embeddings. Al eliminar aleatoriamente conexiones entre las palabras durante el entrenamiento, se evita que el modelo se vuelva dependiente de algunas características mejorando la generalización
- Capa GRU bidireccional: la agregamos para que capture información de palabras anteriores y posteriores a una palabra específica ya que puede influir en la interpretación del sentimiento
- Capa GRU: agregarla luego de la GRU bidireccional puede permitir una mayor capacidad de extracción de características
- Capa densa: con una función de activación RELU para proporcionar transformación no lineal a las características extraídas anteriormente y el modelo aprenda relaciones más complejas
- Capa de dropout: agregamos otra capa de dropout con una tasa de 0.5 dado que sobreajustaba demasiado con bastante velocidad
- Capa densa de salida: con función de activación sigmoide para la clasificación binaria de las críticas

En cuanto a la configuración del modelo, se utilizó la función de pérdida binary_crossentropy y optimizador Adam con una tasa de aprendizaje de 0.001. Por último, incluimos EarlyStopping para ayudar a evitar el problema del sobreajuste deteniendo el entrenamiento cuando no se observa una mejora significativa del F1 score. En cuanto a las métricas en entrenamiento, obtuvimos:

- Accuracy de 0.8367
- Precisión de 0.8389
- Recall de 0.8362
- F1 de 0.8363

En Kaggle, la metrica obtenida es 0.55398.

Modelo de clasificación secuencial basado en Redes Neuronales

Con el objetivo de mejorar las métricas de rendimiento y dado que encontramos dificultades para generalizar los resultados a otro conjunto de datos, decidimos probar algo distinto y entrenar una red neuronal con boosting secuencial. Tomamos un enfoque que combina múltiples clasificadores de redes ponderados y ajusta los pesos para mejorar la generalización y adaptabilidad a diferentes datos. Elegimos una arquitectura recurrente, utilizando capas LSTM y densas, además de aplicar regularización. Si bien en entrenamiento obtuvimos un F1 score de 0.8153 y métricas similares a las obtenidas con las demás redes, en Kaggle notamos una mejoría, obteniendo una métrica de 0.58712.

Métricas obtenidas

A continuación, se muestra una tabla con las métricas obtenidas para los modelos entrenados.

MODELO	ACCURACY	PRECISIÓN	RECALL	F1
Bayes Naïve con CountVectorizer	0.8805	0.8811	0.8806	0.8804
Bayes Naïve con CountVectorizer Optimizado	0.8799	0.8805	0.8800	0.8798
Bayes Naïve con TfidfVectorizer	0.8780	0.8781	0.8780	0.8779
Bayes Naïve con TfidfVectorizer Optimizado	0.8869	0.8869	0.8869	0.8868
Random Forest	0.8526	0.8526	0.8526	0.8525
Random Forest Optimizado	0.8531	0.8534	0.8529	0.8530
XGBoost	0.8444	0.8444	0.8443	0.8443
XGBoost Optimizado	0.8473	0.8476	0.8471	0.8472
Voting	0.8807	0.8807	0.8806	0.8806
Voting Optimizado 1	0.8899	0.8902	0.8897	0.8898
Voting Optimizado 2	0.8655	0.8668	0.8652	0.8653
Stacking 1	0.9033	0.9034	0.9032	0.9032
Stacking 2	0.8868	0.8869	0.8867	0.8867
Stacking 1 Optimizado 1	0.9017	0.9018	0.9016	0.9016
Stacking 2 Optimizado	0.8933	0.8933	0.8932	0.8932
Stacking 1 Optimizado 2	0.9058	0.9059	0.9057	0.9057
Red Neuronal 1	0.8329	0.8331	0.8327	0.8328
Red Neuronal 2	0.8366	0.8377	0.8362	0.8363
Red Neuronal 3	0.8220	0.8234	0.82166	0.8217
Red Neuronal 4	0.8426	0.8455	0.8421	0.8421
Red Neuronal 5	0.8233	0.8307	0.8224	0.8220
Red Neuronal 6	0.8367	0.8389	0.8362	0.8363
Red con Boosting	0.7868	0.7256	0.9302	0.8153

Tabla de resultados