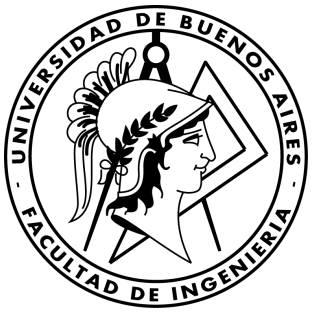

Trabajo Práctico 2
Críticas de cine

[75.06] Organización de datos
Primer cuatrimestre 2023
Grupo 19: Sudanalytics

Integrantes

Nombre	Padrón	Mail
Re, Adrián Leandro	105025	are@fi.uba.ar
Lorenzo, Luciano Andrés	108951	llorenzo@fi.uba.ar
Toulouse, Alan	105343	atoulouse@fi.uba.ar
Tonizzo, Nicolas	107820	ntonizzo@fi.uba.ar

Fecha de entrega: 29/06/2023

Índice

1. Introducción

En este proyecto, se abordó el desafío de construir modelos de clasificación capaces de analizar porciones de texto en lenguaje natural y detectar el sentimiento presente, en particular, en forma binaria (positivo o negativo). Se implementaron diferentes enfoques de modelado, como Bayes Naïve, Random Forest, XGBoost, una red neuronal utilizando Keras y Tensor Flow, y un ensamble de modelos.

Se evaluaron las métricas de precisión, recall y medida F1 en el conjunto de datos de prueba local para cada modelo. Estas métricas brindaron información sobre la capacidad de los modelos para identificar correctamente los sentimientos en el texto.

2. Desarrollo

Durante el transcurso de este trabajo, se entrenaron 4 modelos distintos, y un último en donde se ensamblaron los modelos previos. Previo a este proceso, hubo un preprocesamiento del texto el cual solo utilizamos en la red neuronal, en donde se pasaron todos los textos a minúsculas, se reemplazaron caracteres especiales como letras con tildes y la ñ, se eliminaron las stopwords, que se consideran palabras que no aportan información al sentimiento que se analiza y por ultimo detectamos que en el dataset existian criticas en idioma ingles las cuales decidimos eliminar.

2.1. Bayes Naive

Para este modelo no usamos los datos preprocesados manualmente, ya que contamos con el método `CountVectorizer()` que incluye su propio preprocesamiento. La salida de este se procesa una vez mas mediante el método `TfidfTransformer()` calcula las puntuaciones TF-IDF y determina la importancia de cada término en los documentos comparándolos con el resto de la colección. Una vez realizada esta conversión, se entrena un modelo Bayes Naive de tipo multinomial, y para realizar las predicciones, debemos convertir los datos de igual manera como hicimos con el conjunto de entrenamiento.

2.2. Random Forest

Para Random Forest utilizamos un preprocesamiento idéntico al realizado en Bayes Naive usando los métodos `CountVectorizer()` Y `TfidfTransformer()`. Una vez hecho esto, se entreno un modelo con sus hiperparametros por defecto para conocer sus resultados en train y validación. Teniendo en cuenta estos hiperparametros y el tiempo que consumen en realizar el entrenamiento es que elegimos el set de hiperparametros y sus valores para utilizar en GridSearchCV y obtener la mejor combinación posible.

2.3. XGBoost

Para XGBoost utilizamos un preprocesamiento idéntico al realizado en Bayes Naive usando los métodos `CountVectorizer()` Y `TfidfTransformer()`. Después de esto, se entrenaron varios modelos con diferentes hiperparámetros, pero en todos los casos, los scores de train y test quedaban muy diferentes. Finalmente utilizamos RandomForestCV para encontrar los hiperparámetros que mejor ajusten al modelo de XGBoost y den un score de validación lo mas alto posible.

2.4. Red neuronal

Para este modelo realizamos un preprocesamiento distinto a los anteriores utilizando un modelo de tipo `Bag Of Words`, `Tokenizer`, para llevar las criticas a secuencias y `PadSequence` para que estas tengan una misma longitud. Hicimos pruebas con diferentes arquitecturas, activaciones y regularizaciones. En cuanto a la arquitectura, intentamos utilizar sucesivas `Dense Layers` obteniendo resultados muy pobres y elevados tiempos de entrenamiento. Encontramos una gran

mejoría al utilizar una capa de entrada de tipo `Embedding`,donde se convierten las palabras a vectores y se clusterizan aquellas similares, junto con una segunda capa de tipo `LSTM` y una tercera y cuarta capa de tipo `Dense Layers` . Una vez obtenida esta red, fuimos incluyendo diferentes tipos de regularizaciones y activaciones en las capas para obtener la mejor combinación posible. Se utilizaron regularizaciones del tipo `Drop Out`, `L2` y `Early Stopping` para evitar el sobreajuste ademas de regularizaciones L2. Consideramos que hubiese sido mejor tener una matriz de embeddings preentrenada, pero no encontramos ninguna en español. Creemos que la regularizacion mas significativa en cuanto a reducción del sobreajuste fue el uso de `DropOut` en la capa `LSTM`.

2.5. Ensamble

Para esta parte realizamos un ensamble de tipo voting combinandolos modelos Bayes Naive, Random Forest y XGBoost ya que fueron estos los modelos que mejores resultados nos dieron.

Como no pudimos mejorar los resultados, también probamos Bagging en un intento de encontrar un mejor modelo. Para ello combinamos muchos modelos de naive bayes (`MultinomialNB`) utilizando la herramienta de bagging que proporciona sklearn: `BaggingClassifier`. Creemos que naive bayes era la mejor opción para probar esto porque es un modelo muy rápido de entrenar.

3. Resultados

En el set de entrenamiento los resultados que obtuvimos para cada modelo:

Modelo	F1	Precision	Recall
Bayes naive	0.86439	0.87593	0.85426
Random forest	0.85069	0.82575	0.87718
XGBoost	0.85183	0.83903	0.86502
Red neuronal	0.80252	0.80119	0.80385
Ensamble	0.79282	0.87961	0.72163

En la competencia de Kaggle, este fue el mejor f1 score que conseguimos para cada modelo:

- Bayes naive: 0.71467
- Random forest: 0.72165
- XGBoost: 0.71215
- Red neuronal: 0.73929
- Ensamble: 0.72165

4. Conclusiones

Este proyecto ha demostrado la viabilidad y eficacia de diversos enfoques de modelado en la tarea de análisis de sentimientos en lenguaje natural. Los modelos desarrollados pueden ser utilizados como herramientas útiles para analizar y comprender el sentimiento en los documentos.

En base a los resultados obtenidos, se concluye que los modelos de clasificación implementados demostraron una buena capacidad para detectar el sentimiento en el texto. Cada uno de los modelos presentó un desempeño sólido en términos de precisión, recall y medida F1 en el conjunto de prueba local. Todos los modelos dieron resultados similares, siendo la red neuronal un poco mejor que los otros modelos, y XGBoost un poco peor que el resto.

Así como en el trabajo previo tuvimos mejores resultados con los modelos de árboles de decisión (en concreto XGBoost), en este el modelo con las mejores métricas fue la red neuronal. Creemos que esto se debe a la naturaleza del problema, ya que este set de datos tenía muchas menos dimensiones que el anterior, por lo que el modelo de red neuronal supo adaptarse de una forma

más óptima que el resto.
Concluimos entonces que las redes neuronales son un mejor modelo para el procesamiento de lenguaje natural.

5. Referencias

- Documentación oficial de Scikit-learn: Entrenamiento de modelo Bayes Naive
- Documentación oficial de Scikit-learn: Entrenamiento de clasificador Random Forest
- Página oficial de Keras
- Documentación oficial de Scikit-learn: Ensamblado de modelos de tipo voting