FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Trabajo Práctico 2 — Criticas cinematográficas

Organizacion de Datos
Curso Rodriguez
Primer cuatrimestre de 2023

Alumno	Padrón	gitHub
Camila Gonzalez	105661	c-gonzalez-a
Eduardo Martín Bocanegra	106028	martinboca
Mateo Cabrera	108118	m-cabrerar

1. Introducción

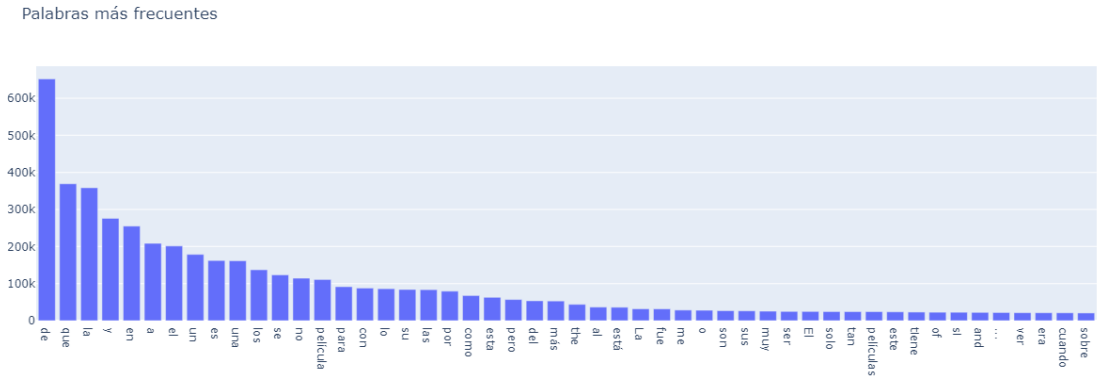
El objetivo de este proyecto será escoger y entrenar un modelo para predecir el sentimiento de una reseña cinematográfica. Para ello vamos a utilizar una colección de críticas en español dadas por la cátedra y vamos a tratar de identificarlas como positivas o negativas según corresponda, usando diferentes modelos y preprocesamientos del texto, y subiremos nuestras predicciones a una competencia en Kaggle.

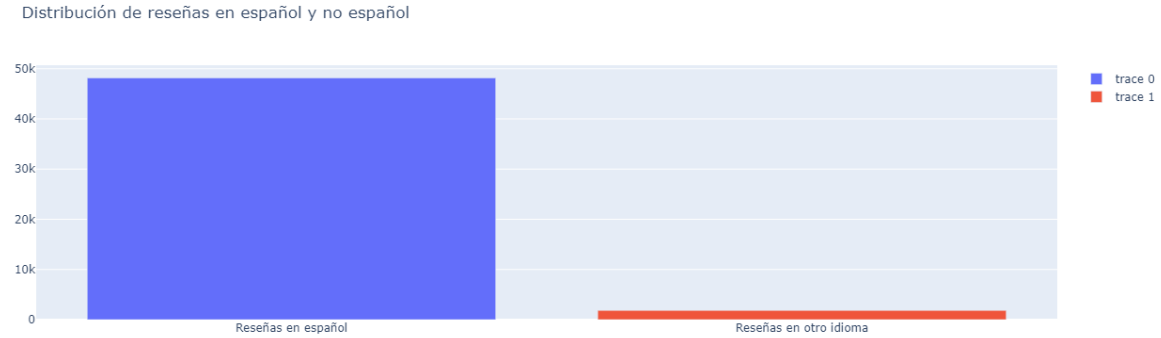
2. Resumen

En el presente informe se desarrollarán los mejores modelos obtenidos en la búsqueda del mejor clasificador de sentimientos usando Bayes Naïve, XGBoost, Random Forest, SVM, Redes Neuronales y Ensambles de modelos.
Repositorio con el código: https://github.com/m-cabrerar/7506R-1C2023-GRUPO01/tree/main/TP2

3. Exploración inicial y preprocesamiento de texto

Durante la exploración inicial, verificamos que nuestro dataset esté balanceado. Efectivamente, encontramos igual porcentajes de reseñas positivas y negativas. Además, exploramos las palabras más frecuentes y el idioma de las reseñas, información que nos será útil en el preprocesamiento de datos.





Para la resolución de la problemática, exploramos varios preprocesamientos de texto.

- Preprocesamiento 1: Fue el primero que hicimos, vectorizamos todo el conjunto de reviews sin ningún filtro. Dentro de este, hay dos que quitan stop words en español utilizando la lista proporcionada por la librería NLTK (Natural Language Toolkit).
- Preprocesamiento 2: En este intento, se vectorizaron las criticas sin stop words españolas ni inglesas, para ello se descargó una lista externa que descargamos de github (link adjuntado en la sección referencias). También se armo un vocabulary que incluya solo aquellas palabras que aparecían tanto en el test.csv como en el train.cvs, con intenciones de reducir la cantidad de palabras.
- Preprocesamiento 3: Al haber poco porcentaje de reseñas en inglés, en este preprocesamiento las eliminamos, junto con las stopwords, y convertimos todo el texto a minúscula.
- Preprocesamiento 4: Convertimos el texto a minúscula, eliminamos las reseñas en inglés, las URLs que existían en algunas reseñas, signos de puntuación, tildes, y símbolos que no pertenecían al alfabeto español.
- Preprocesamiento 5: Mismo que el preprocesamiento 4, pero además eliminando stopwords.
- Preprocesamiento 6: Mismo que el procesamiento 5, pero agregando lematización a las palabras en español.

4. Modelos estudiados

4.1. Bayes Naïve

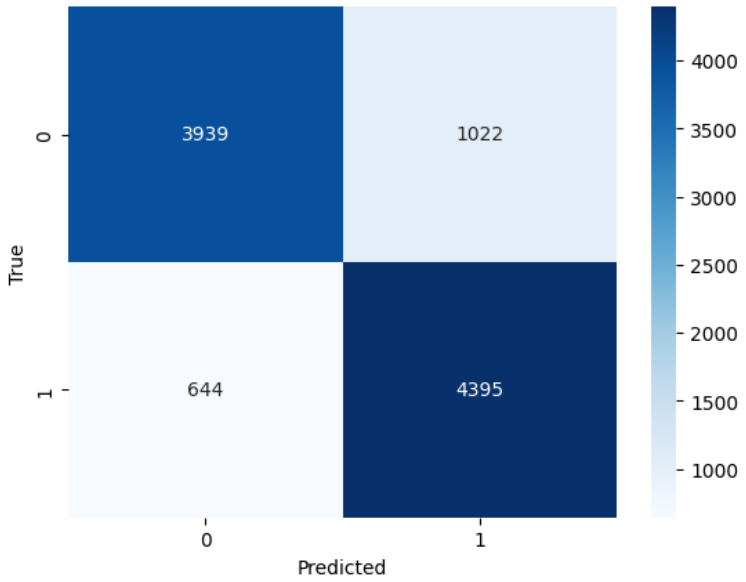
Como primer modelo, entrenamos un Bayes Naïve utilizando una Bernoulli Binomial. Obtuvimos buenos resultados con los parámetros por defecto, pero decidimos buscar mejores hiperparámetros para mejorar el resultado en la competencia. Las métricas observadas fueron:

- Accuracy: 0.8466
- Precision: 0.8670157068062827
- Recall: 0.8215915856320699
- F1 Score: 0.8436926839209293
- Kaggle: 0.73056

4.2. Random Forest

En el caso del modelo de Random Forest, se realizaron experimentos utilizando los enfoques de preprocesamiento de datos 1 y 2 propuestos. Se encontró que la variante más efectiva del preprocesamiento 1 consistió en eliminar las stop words en español utilizando la lista proporcionada por NLTK. Además, se llevó a cabo una búsqueda de hiperparámetros para optimizar el desempeño del modelo. Las métricas obtenidas fueron:

- Accuracy: 0.8334
- Precision: 0.8113346870961787
- Recall: 0.8721968644572335
- F1 Score: 0.8406656465187453
- Kaggle: 0.73773



4.3. XGBoost

Para el modelo de XGBoost, se realizaron pruebas utilizando diferentes enfoques de preprocesamiento, incluyendo el Preprocesamiento 1 y sus variantes, así como el Preprocesamiento 2. Sin embargo, se encontró que el mejor rendimiento se obtuvo utilizando el primer enfoque, y buscando los mejores hiperparámetros.
Métricas obtenidas:

- Accuracy: 0.8722
- Precision: 0.8643673706645999
- Recall: 0.8852947013296288
- F1 Score: 0.8747058823529412
- Kaggle: 0.73095

4.4. Support Vector Machines

Si bien no era requerimiento, decidimos incluir modelos SVM, por su buen desempeño a la hora de predecir sentimiento. Comenzamos probando 4 distitos kernels para observar sus métricas iniciales.

Cuadro 1: Métricas obtenidas con Support Vector Machines

Kernel	**Precisión**	**Recall**	**F1-Score**	**Kaggle**
Linear	0.89	0.89	0.89	0.7232
Polinómico	0.85	0.85	0.85	0.72824
Radial Basis Function	0.89	0.89	0.89	0.74355
Sigmoide	0.89	0.89	0.89	0.46927

En base a estas métricas decidimos seguir entrenando un SVM con kernel RBF, buscando hiperparámetros logramos mejorar las métricas aún más, obteniendo en kaggle un f1-score de 0,74355.

4.5. Redes Neuronales

Para este modelo, realizamos muchas pruebas con redes con una capa de Embedding y capas GRU o LSTM. Probamos variar el tamaño del output de la capa de Embedding, alternar entre capas GRU y LSTM (tambien entre cantidad de capas), cambiar la cantidad de neuronas por capa y hacer el entrenamiento con más o menos épocas variando el learning rate. Siempre entrenamos la red usando EarlyStopping (con la opción de guardar los mejores pesos), para frenar el entrenamiento si los resultados dejan de mejorar, y usamos Dropout y regularizacion L2 en cada capa de la red para reducir el overfitting.

Si bien solo probamos redes recurrentes, también nos pareció interesante la opción de utilizar redes convolucionales. Según investigamos, dan muy buenos resultados al utilizarse para el análisis de sentimientos.

La red neuronal que mejor performance en cuanto a entrenamiento y resultados en la competencia dió, fue una red recurrente con de 3 capas GRU, y 50 neuronas por capa. Con esta red, realizamos entrenamientos de 1, 20, 50, 70 y 85 épocas. El entrenamiento que mejor resultados dio en la competencia fue el de 70 épocas con un leraning-rate de 1e-6. Si bien la red podía seguir mejorando resultados sobre el dataset de validación, decidimos cortar el entrenamiento en 70 épocas porque con más épocas dió peores resultados en Kaggle.

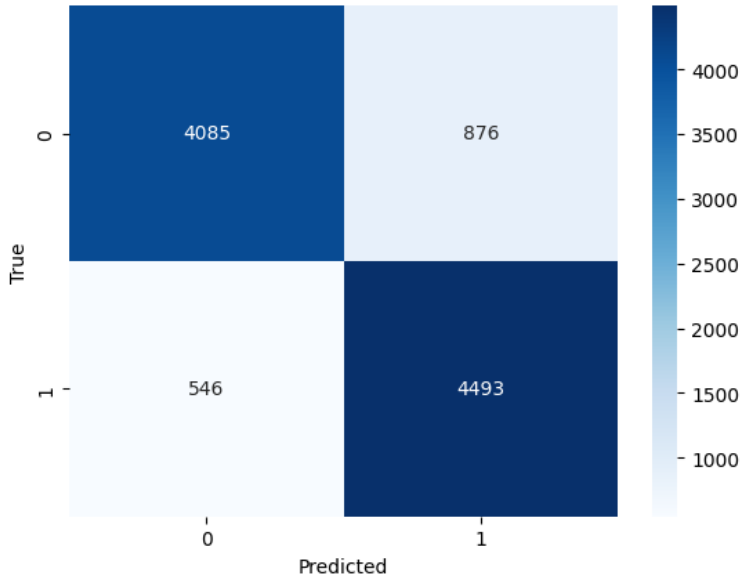
- Accuracy: 0.87
- Precision: 0.87
- Recall: 0.87
- F1 Score: 0.87
- Kaggle: 0.74607

4.6. Ensamble de modelos

Hicimos un solo ensamble de modelos utilizando la técnica de voting. En este ensamble, combinamos los mejores modelos de XGBoost, Random Forest y SVM con kernel radial. Para el preprocesamiento de los datos, utilizamos el método de preprocesamiento 2 propuesto, ya que este método resultó en una menor dimensionalidad en la vectorización de las palabras.

A continuación, se presentan las métricas obtenidas con este modelo:

- Accuracy: 0.8578
- Precision: 0.8368411249767181
- Recall: 0.8916451676920024
- F1 Score: 0.8633743274404305
- Kaggle: 0.72378



5. Conclusiones y mejor modelo final

Como conclusión nos dimos cuenta que el preprocesamiento del texto es un factor muy importante a la hora de entrenar cualquier modelo. En este trabajo práctico nos encontramos con que todos los modelos fueron bastante , y lo que mas influyó no fueron los hiperparametros, la arquitectura de la red o los ensambles elegidos, sino el preprocesamiento del dataset. Sin embargo, al comparar las métricas con distintos preprocesamientos, nos dimos cuenta que remover stopwords tendía a empeorar los modelos. Esto se puede deber al contexto que aportan.

Las Redes Neuronales recurrentes, Support Vector Machines y Random Forest fueron los modelos con los que obtuvimos mejor desempeño. Más así, el mejor modelo fue la red neuronal descrita en el apartado 4.5, con un resultado de 0.747 en Kaggle, entrenada con el dataset generado por el Preprocesamiento 4.

6. Referencias

- https://github.com/pysentimiento/pysentimiento
- https://github.com/Alir3z4/stop-words