TP2: Críticas Cinematográficas - Grupo 23

Introducción

El objetivo de este trabajo práctico es buscar la predicción más precisa posible acerca de si una crítica cinematográfica dada en español es positiva o negativa mediante el procesamiento del lenguaje natural.
El dataset consiste de unas 50000 reseñas de películas en castellano marcadas con su sentimiento, que podía ser negativo o positivo. Una particularidad de este dataset es que varias muestras estaban en idioma inglés.
Para este trabajo práctico entrenamos diversos modelos, siendo estos Bayes Naive (Multinomial, Bernoulli, Gausseano), árbol de decisión, regresión logística, redes neuronales, random forest, SVM, XGBoost y ensambles de tipo voting y stacking. Decidimos experimentar con una amplia variedad de modelos ya que como no conocíamos el dominio del problema, no sabíamos de antemano cuáles eran los que mejor iba a rendir. Por ejemplo, si nos hubiéramos acotado solo a los pedidos por el enunciado, íbamos a perder modelos con gran performance como la regresión logística o SVM.
Por otro lado, probamos diversas técnicas como identificar el idioma de una review, traducir las reviews en inglés, pasar emojis a texto, analizar si existían patrones en las reviews, eliminado de outliers y duplicados y preprocesamiento de texto.
Para el procesamiento del texto:

- limpiar texto = Elimina los caracteres que no sean un carácter alfanumérico y los reemplaza por espacios en blanco. Elimina las palabras que tienen solo un carácter alfabético y las reemplaza por espacios en blanco. Sustituye múltiples espacios en blanco por uno solo. Convierte todos los caracteres del texto a minúsculas.
- filtrar stopwords y números = elimina los números y las stopwords que son palabras comunes sin significado específico. Estas stopwords carecen de información distintiva y no contribuyen significativamente al significado del texto.
- stemear palabras = reducimos cada palabra su raíz (o "stem"). Por ejemplo, correría y corre se reducirán a corr.

- tokenizar = realiza una serie de operaciones de preprocesamiento (las mencionadas anteriormente y también elimina los sustantivos propios) y devuelve el texto tokenizado y procesado.

Además, se eliminaron las reviews duplicadas, las que estaban un idioma que no sea castellano y outliers marcados por el z-score.
Utilizando lo anterior ya teniamos todo el texto preprocesado, pero falta tokenizarlo de cierta manera para que nuestro modelos puedan utilizar esa informacion. Para ello utilizamos el Tfidf Vectorizer, el cual brinda una representación numérica de los documentos que captura la importancia relativa de las palabras en el contexto del conjunto de documentos.
Cabe destacar que el vectorizador que utilizamos aportó gran performance a todos los modelos teniendo de hiperparametros max_features=25000 (en algunos modelos menos), ngram_range=(1,2) (utiliza unigramas y bigramas), min_df=5, max_df=0.7, binary=True (en algunos modelos no)

- min df: establece el umbral mínimo para la frecuencia de un término en los documentos.
- max df: establece el umbral máximo para la frecuencia de un término en los documentos.

Cuadro de Resultados

Medidas de rendimiento en el conjunto de TEST:

- F1
- Precision
- Recall
- Métrica X (F1 score del conjunto train)
- Resultado obtenido en Kaggle.

Modelo	F1-Test	Precision Test	Recall Test	Metrica X	Kaggle
Bayes Naive	0,8760	0,8528	0,9005	0,8945	0,7790
Random Forest	0,8289	0,7999	0,8600	0,8484	0,7085
XgBoost	0,8685	0,8596	0,8776	0,9230	0,7186
Red Neuronal	0,8476	0,8267	0,8691	0,9977	0,7131

Ensamble	0,8886	0,8815	0,8956	0,9034	0.7699
SVM	0,8891	0,8781	0,9005	0,9095	0,7541
Regresion Logistica	0,8861	0,8752	0,8974	0,9075	0,7569

Descripción de Modelos

Modelos utilizados (incluimos los que mejor rindieron)

- **Regresión Logística:** Decidimos agregar el modelo ya que luego de una investigación por nuestra cuenta encontramos que podía dar buenos resultados. Efectivamente fue nuestro segundo modelo, donde al igual que con Bayes Naive, mejoró considerablemente al optimizar el TFIDF. Los hiperparámetros del ultimo nombrado fueron iguales a los del Bayes Naive, y los hiperparametros específicos de la regresión fueron penalty L2, un C de 0.3, un solver LibLinear, un máximo de iteraciones de 100, tol = 1e-4 y L1 Ratio=None
- **XG Boost:** Es un modelo un tanto complejo de entrenar, el mismo no es el optimizado por hiperparametros ya que el optimizado dio peores resultados. Los hiperparametros seleccionados son: random_state=42, colsample_bytree=0.6, subsample=0.6, gamma=0.3, max_depth=5, learning_rate=0.089, min_child_weight=10, n_estimators=700
- **Red Neuronal:** Para la red neuronal decidimos realizar una red neuronal recurrente con una capa de Embedding de 128, una Convolucional de 64 filtros, una GRU de 64, luego una Dense de salida y finalmente la Activación sigmoidea. También le agregamos un early stopping con un min delta de 0.0001. Decidimos realizar una red neuronal recurrente ya que las mismas son muy útiles para análisis de texto, debido a su capacidad de memoria, donde puede tener en cuenta no solo palabra a palabra.
- **Bayes Naive:** fue por lejos nuestro mejor modelo. Utilizamos un bayes naive de bernoulli. El único parámetro que lleva es el alpha=0.5. Lo destacable es que localmente no era el mejor pero en kaggle dio el mejor resultado.
- **SVM:** otro algoritmo adicional que probamos que dio muy buenos resultados localmente pero luego en kaggle no era tan efectivo.
- **RandomForest:** Es bastante simple a la hora de entrenar y predecir pero, no dio buenos resultados en este tipo de problemas. Los hiperparametros

elegidos para este modelo son: max_depth=20, min_samples_leaf=50 y n_estimators=900.

- **Ensamble:** el ensamble elegido consiste de un tipo voting 'hard' utilizando los siguientes modelos: xgboost, random forest, regresión logística y ambos modelos de bayes naive, multinomial y bernoulli.

Conclusiones generales

Teniendo todo en cuenta, podemos afirmar que fue útil realizar un análisis exploratorio de los datos para luego realizar el preprocesamiento pertinente, por ejemplo el idioma de las reviews, las cuales solo mantuvimos las que estaban en español, o analizar la cantidad de palabras, las cuales eliminamos los casos atípicos con un z-score. Estos dos casos nombrados anteriormente fueron los que tuvieron más impacto en los resultados, los cuales mejoraron considerablemente, hubieron otros preprocesamientos que no tuvieron tanto impacto como transformar los emojis.

El modelo que mejor puntaje nos trajo en local fue SVM, sin embargo en Kaggle no fue así, ya que el mejor de la competencia fue Naive Bayes, con aproximadamente 0,02 puntos por encima del SVM. A la hora de analizar los tiempos de entrenamiento, los modelos más complejos fueron SVM y la Red Neuronal, ambos tardando aproximadamente una hora y media, y los que más rápido se entrenaron fueron Naive Bayes y Regresión Logística tardando alrededor de 10 segundos en ejecutarse. Un detalle no menor es que los dos modelos que más rápido se entrenaban fueron de los mejores que resultaron en Kaggle, por lo que estos dos modelos son altamente recomendables en términos de relación costo-desempeño.

No consideramos que nuestros modelos estén a la altura requerida para ser utilizados de forma productiva, principalmente porque nuestros resultados locales y los de la competencia de kaggle difieren muchísimo. Esto puede ser debido a varios motivos, uno de ellos es nuestro entrenamiento. Si tuviéramos mayor volumen de datos entonces nuestro modelo podría realizar una mejor generalización y además tendríamos mayor volumen de datos para el testing local. Tambien tendriamos que tener en cuenta de donde se extraen los datos, ya que nuestro dataset podría estar sesgado a la jerga local de donde se extraen los datos, entonces a la hora de realizar una predicción con datos de otra región el mismo no va a poder predecir con certeza.

Para mejorar nuestro modelo como nombramos anteriormente, consideramos que tener un dataset de train de mayor volumen y diversidad podría beneficiar mucho a nuestro modelo. Además tener una mayor capacidad de cómputo también nos hubiese beneficiado para probar más modelos. Por ejemplo, intentamos realizar un modelo detallado en un Paper sobre reseñas de un hotel de Arabia, sin embargo no logramos completar el entrenamiento por el largo tiempo que requería dicha red neuronal.

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Juan Pablo Carosi Warburg	15
Mateo Daniel Vroonland	15