TP2: Críticas Cinematográficas - Grupo 33

Introducción

Teníamos un dataset con tres columnas: ID, review_es y sentimiento. Con ese dataset de entrenamiento teníamos que entrenar un modelo para que pueda reconocer si una crítica de película era positiva o negativa. Dicho dataset tenía 50mil líneas mientras que el de testeo tenía alrededor de 8500 líneas.

Una de las complicaciones que vimos fue que en el df de entrenamiento habían ciertas filas que estaban en inglés mientras que en el de testeo estaban todas en español. Además de que en los dos datasets habían demasiados signos de puntuación que podrían dificultarle a los modelos en el momento de entrenar.

Para empezar decidimos pasar el sentimiento a numero binario, los positivos a 1 y los negativos a 0. Además de eso, pasamos todas las críticas a minúscula. Luego implementamos algunas técnicas de procesamiento:

- Identificacion de lenguajes de cada critica con langdetect (https://pypi.org/project/langdetect/)
- Stemmizar con Snowball

Y finalmente, utilizamos unos diccionarios de lexicon y modelos pre-entrenados para ver qué sucedía:

- Bert (versión Beto que admite espanol) (https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased
- Sentileak (https://pypi.org/project/SentiLeak/)/

Cuadro de Resultados

Modelo	F1-Test	Presicion Test	Recall Test	Accuracy	Kaggle
Bayes Naive	0.8457927049476346	0.8491660623640319	0.8424460431654677	0.8465408805031447	0.73037
Random Forest	0.8269497640854844	0.8395322626091857	0.8147388569866011	0.8274645451400899	0.68482

XgBoost	0.8410195769281303	0.821272773928663	0.8617393645665051	0.8386666666666667	0.70013
Red Neuronal	0.8142	0.8147	0.8189	0.8147	0.71118
Stacking	0.8381026539484783	0.8122248773739152	0.8656836461126005	0.8336666666666667	0.73386
Bert/Beto (Extra)	0.8862	0.8443	0.9324	0.8795	0.83271

Dentro de los modelos pedidos, Stacking fue el que mejor rindió, lo cual es una obviedad ya que uno de su modelos de entrenamiento y su meta modelo era Bayes Naive, el cual consideramos óptimo para este tipo de problemas.

Descripción de Modelos

Modelo	**Hiperparametros**
Bayes Naive	force_alpha = False, fit_prior = True, class_prior = None, alpha = 1
Red Neuronal	Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=250), LSTM(100), Dense(64, activation='relu'), Dropout(0.5), Dense(1, activation='sigmoid') optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'] epocas = 5 batches = 10

Stacking	Modelos: Bayes Naive, Random Forest, XgBoost. Meta Modelo: Bayes Naive.
tfidfVectorizer	ngram_range = (1,2), min_df = 0.01, max_df = 0.5, lowercase = True

Utilizamos el mismo vectorizer para todos los modelos excepto para XgBoost que utilizamos los parámetros en default. XgBoost y Random Forest terminaron con los parámetros default ya que no pudimos optimizarlos.

Para los modelos Bayes Naive y Random Forest, decidimos eliminar las filas en inglés del dataset de entrenamiento y vimos que mejoraron los porcentajes. Para el resto de los modelos decidimos no hacerlo porque no ayudaba en las predicciones en Kaggle.

Hicimos los ensambles Stacking y Voting con los mismos modelos e hiperparametros, y nos quedamos con el Stacking ya que dio mejor en Kaggle (0.73386 vs. 0.70323). En Stacking utilizamos los mejores modelos de Random Forest, Bayes Naive y XgBoost, y como meta modelo usamos otro Bayes Naive con los mismo hiperparametros.

Nuestra red neuronal está hecha con tensor flow. La primera capa es una capa de embedding y la segunda es una LSTM, lo cual tuvo un impacto mayor en mejorar los resultados. Tiene la función sigmoide en la última capa ya que al pasar los sentimientos a 0s y 1s, la decisión era binaria. Además de todo esto, utilizamos solo 25000 líneas para entrenar, ya que nos dimos cuenta que si usabamos el dataset entero, el modelo terminaba overfitteando, por ende bajaban las métricas en Kaggle.

Finalmente, nuestro modelo extra de Beto surgió de nuestro deseo de utilizar un diccionario de sentimientos. Al principio probamos con Sentileak para hacerle un análisis de sentimiento a cada crítica, pero como tardaba demasiado, pasamos a Beto. Beto es un modelo pre-entrenado de la misma forma que Bert pero con léxico en español. Nosotros utilizamos el modelo y luego lo seguimos entrenando con nuestro dataset de entrenamiento. Si no lo entrenabamos con el train.csv los resultados eran bastante bajos, por lo cual decidimos seguir mejorandolo con nuestros datos, otro gran detalle es que con el 100 % de los registros del archivo tardaba demasiado y tuvimos que reducirlo a solo utilizar las primeras 10 mil líneas del train.csv.

Conclusiones generales

Nuestro análisis exploratorio fue útil hasta cierto punto. Nos ayudó a darnos cuenta que había líneas en inglés, así como también investigar si había caracteres especiales en uso o emoticones, o si quizás había entradas en blanco.

Nos encontramos con que el preprocesamiento de datos ayudaba en ciertos casos y en otros no. Particularmente con los modelos más robustos, como la red neuronal, el análisis previo hecho no nos aportaba en las predicciones. También intentamos utilizar Snowball para stemmizar las críticas y poder sacar los caracteres especiales, pero luego al correr las pruebas los resultados empeoraban entonces decidimos no utilizarlo.

Si dejamos de lado a Beto, Bayes Naive tuvo el mejor desempeño en TEST, pero tuvo un bajo desempeño en Kaggle, por lo cual seguramente el modelo terminó overfitteando. En realidad, esto ocurre con todos nuestros modelos (excepto Beto), los resultados de Kaggle comparados a los del test son mucho menores.

Por amplia diferencia, la implementación de Bert fue la que mejor desempeño tuvo incluso habiendo entrenado con menos datos

De todos los modelos, Bayes Naive fue el más sencillo de entrenar y el más rápido. Esto se debe a que tiene pocos hiperparametros para optimizar y son intuitivos, especialmente en relación a modelos más complejos como lo son las redes neuronales. Además, este fue uno de los pocos que le pudimos aplicar un Random Search para optimizar los hiperparametros, ya que con muchos de los otros modelos no se nos resultó posible por falta de recursos, por otro lado exceptuando nuestra experimentación con Bert, Bayes no solo fue el más fácil sino el que mejor métrica nos dio, e incluso estamos seguros que se puede mejorar más, investigando mejores maneras de preprocesamiento de los datos, en nuestra opinión y por lejos Bayes Naive es el mejor modelo para este tipo de problemas.

Si bien Bert no era uno de los modelos explícitos que se pedían, nos pareció una oportunidad de aprendizaje intentar un modelo que puede mejorar las predicciones mucho más avanzado que los utilizados pero que a su vez se mencionó en clases y nos pareció buena idea investigar e intentarlo, y si bien el modelo performa por encima de los demás dándonos una métrica muy superior, valoramos mucho más nuestros logros con los otros modelos donde con muchas pruebas logramos que mejoren su funcionamiento, algo que quizá se nos quedó fuera del alcance fue poder incorporar este modelo en uno de los ensambles conocidos, así como intentar un ensamble más avanzado como lo es el de cascading

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Lucas Nahuel Raimondi	15
Manuel Davila	12
Dolores Levi	9