TP2: Críticas Cinematográficas - Grupo 12

Introducción

La problemática a resolver era dada una crítica cinematográfica (documento de texto), lograr poder predecir mediante un análisis de sentimientos, si la crítica era positiva o negativa.

El dataset de entrenamiento está compuesto por 50000 registros y 3 columnas que son un ID, la reseña y sentimiento (su etiqueta). Sobre la copia de este dataset, lo que hicimos fue aplicar la técnica de stemming, para reducir las palabras de los textos a su forma raíz, con el objetivo de eliminar el ruido y mejorar la coincidencia. Luego utilizamos técnicas de BoW(bag of words), para el preprocesamiento del texto a vectores, como CountVectorizer/TfIdfVectorizer/HashingVectorizer, y también se quitaron las palabras vacías (stopwords) utilizando el conjunto disponible en la librería nltk.

Para los modelos de predicción de sentimientos se utilizaron: un clasificador Naive Bayes, RandomForest, XGBoost, una red neuronal (utilizando una capa de Embedding normalizado) y un ensamble del tipo Voting, donde, exceptuando el modelo Naive Bayes, para los demás modelos se transformó la variable cualitativa target (sentimiento) en una variable binaria (1 para positivo, 0 para negativo).

Cuadro de Resultados

Modelo	F1-Test	Presicion Test	Recall Test	Accuracy	Kaggle
Bayes Naive	0.86	0.87	0.86	0.87	0.74413
Random Forest	0.86	0.95	0.79	0.88	0.71777
XgBoost	0.85	0.83	0.87	0.84	0.68385
Red Neuronal	0.87	0.87	0.87	0.87	0.7484
Ensamble	0.87	0.86	0.89	0.87	0.74859

Descripción de Modelos

- *Bayes naive:* Se construyó un modelo con el clasificador Naive Bayes al que se le realizó una búsqueda de mejores hiperparámetros con RandomizedSearchCV. También se realizó un stemming a los datos antes de la construcción.

- *Random Forest*: Se construyó un modelo con Random Forest al que se le realizó una búsqueda de mejores hiperparámetros con RandomizedSearchCV. También se realizó un stemming a los datos antes de la construcción.

- *XGBoost*: Se construyó un modelo con XGBoost al que se le realizó una búsqueda de mejores hiperparámetros con RandomizedSearchCV.
-
- Red neuronal: Se construyó una arquitectura de red neuronal cuyos hiperparámetros eran: cantidad de capas ocultas y de neuronas (hidden_layers y neurons) y el learning_rate para el optimizador SGD, donde estos fueron optimizados (junto a los hiperparámetros de entrenamiento) para obtener una mejor métrica de f1-score .

  Esta arquitectura contenía una capa inicial de embedding normalizado, seguida de otra capa con regularizador L2, y además por cada capa oculta se utiliza la técnica de regularización Dropout.

  Esta arquitectura fue elegida por consecuencia de construir varias arquitecturas y evaluar la performance de sus mejores modelos, donde por ejemplo las arquitecturas optimizadas sin regularizadores ó sin la capa de embedding, performaban peor que las que sí contenían estas capas.

- Ensamble: Se construyó un ensamble del tipo voting con los mejores modelos obtenidos de Bayes Naive, Random Forest y XGBoost.

  El modelo que mejor rindió fue el modelo de ensamble Voting.

Conclusiones generales

- **¿Fué útil realizar un análisis exploratorio de los datos?**
  Si, consideramos que siempre es útil realizar un análisis exploratorio antes de comenzar a entrenar cualquier modelo, nos permite tener una primera visualización de los datos, como también nos permite detectar si hubiese algún dato faltante. En nuestro caso, al aplicar stemming para determinados modelos, al reducir las palabras a sus raíces, nos hizo mejorar nuestros modelos.

- **¿Las tareas de preprocesamiento ayudaron a mejorar la performance de los modelos?**
  Si, principalmente quitar las stopwords nos permitió obtener una mejora considerable en las predicciones, así como también la aplicación de la técnica de stemming.

- **¿Cuál de todos los modelos obtuvo el mejor desempeño en TEST?**
  El ensamble nos dió el mejor desempeño en test.

- **¿Cuál de todos los modelos obtuvo el mejor desempeño en Kaggle?**
  El modelo de ensamble de tipo Voting fue el que mejor desempeño tuvo en Kaggle.

- **¿Cuál fue el modelo más sencillo de entrenar y más rápido? ¿Es útil en relación al desempeño obtenido?**
  El modelo construído con el clasificador Bayes Naive fue el modelo más sencillo de entrenar, aunque el modelo con Random Forest fue apenas más rápido pero requirió un poco más de preprocesamiento en comparación. Consideramos que es bastante útil, ya que no tiene una gran diferencia de métricas con el ensamble.

- **¿Cree que es posible usar su mejor modelo de forma productiva?**
  Para este tipo de problema en particular parece ser bastante adecuado considerando las métricas obtenidas, sin embargo, habría que ver su desempeño en un caso dónde la entrada no sea tan consistente.

- **¿Cómo podría mejorar los resultados?**

Se puede mejorar realizando una búsqueda aún más exhaustiva de hiperparámetros como puede hacerse con GridSearchCV ya que prueba todas las combinaciones posibles, sin embargo, este requeriría más tiempo de entrenamiento .También se puede tratar de construir modelos basados en transformers como lo son BERT o GPT, que probablemente agilicen el entrenamiento y mejoren la performance.

Tareas Realizadas

Integrante	Promedio Semanal (hs)
Gaston Sabaj	8hs
Juan Yago Pimenta	6hs.