Movie Genre Classification

Presentado por:

Nidia Marcela Ortiz Sandoval (199713132)
Luis Jorge García Camargo (201920053)
Jairo Alberto Pedraza Corredor (201924260)
Carlos Andrés Páez Rojas (201924257)

Grupo: 6. Julio 2020.

Con el presente documento describimos los análisis y detalles de la solución obtenida.

A partir de técnicas de procesamiento de lenguaje natural, aplicación de modelos predictivos de clasificación y machine learning, se desea realizar una clasificación de género de una serie de películas a partir de su descripción, nombre y año de lanzamiento.

Se cuenta con 2 datasets correspondientes para entrenar y probar la solución planteada, con 7895 y 3383 registros respectivamente.

Análisis inicial y lectura de datos

Inicialmente realizamos una lectura y análisis general de los datos:

Sobre la variable de entrada, donde encontramos la descripción de cada película, aplicamos la función ***CountVectorizer*** para transformar el texto en un vector y tener una lectura a nivel de palabras.

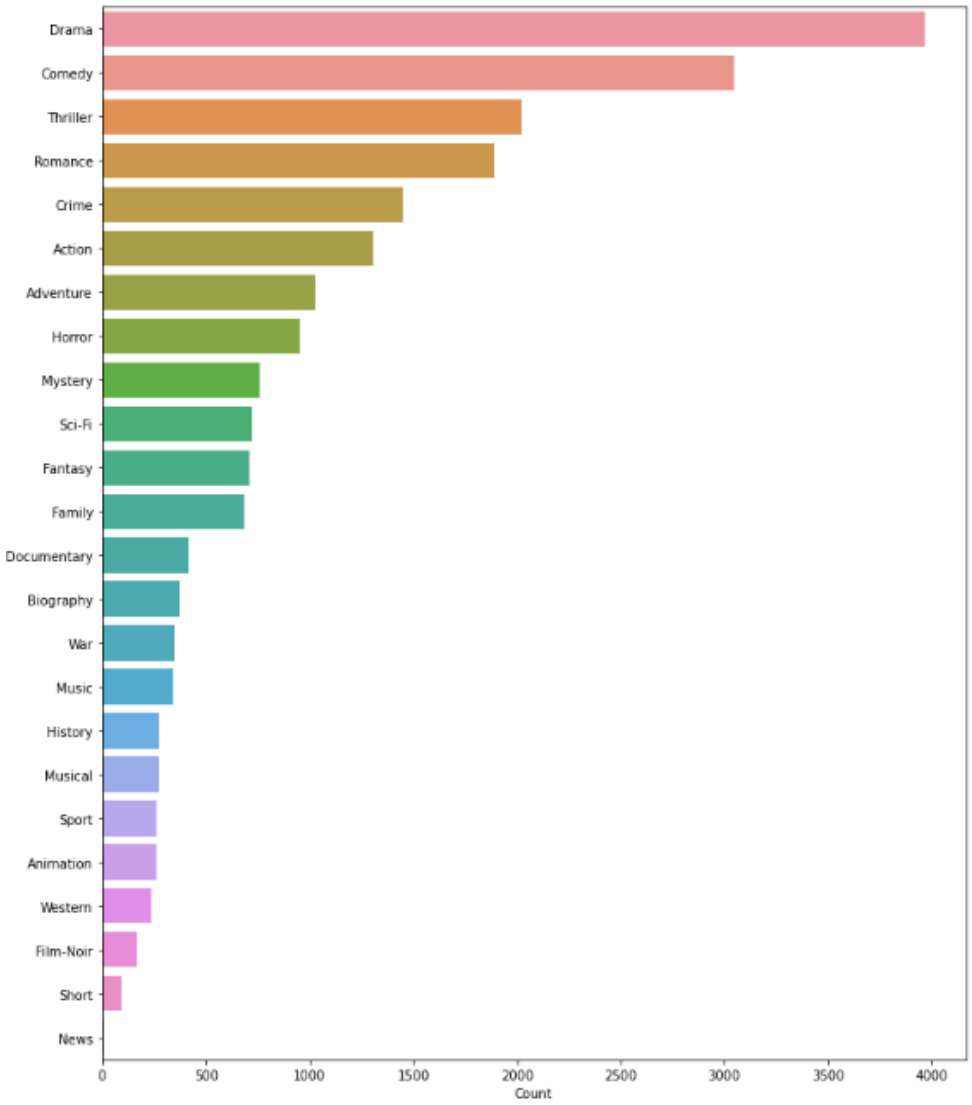
Para la variable de respuesta, que corresponde a los géneros de película posibles para clasificación, aplicamos la función ***MultiLabelBinarizer***, con el objetivo de tener la lectura de esta variable en un matriz binaria que identifique la pertenencia de una clase para cada registro.

Al tratarse de un problema Multi-label; una película puede tener más de un género asociado, para abordar este problema, la función ***MultiLabelBinarizer*** permitirá trabajar como un vector binario en $R24$ la variable de respuesta.

Análisis descriptivo

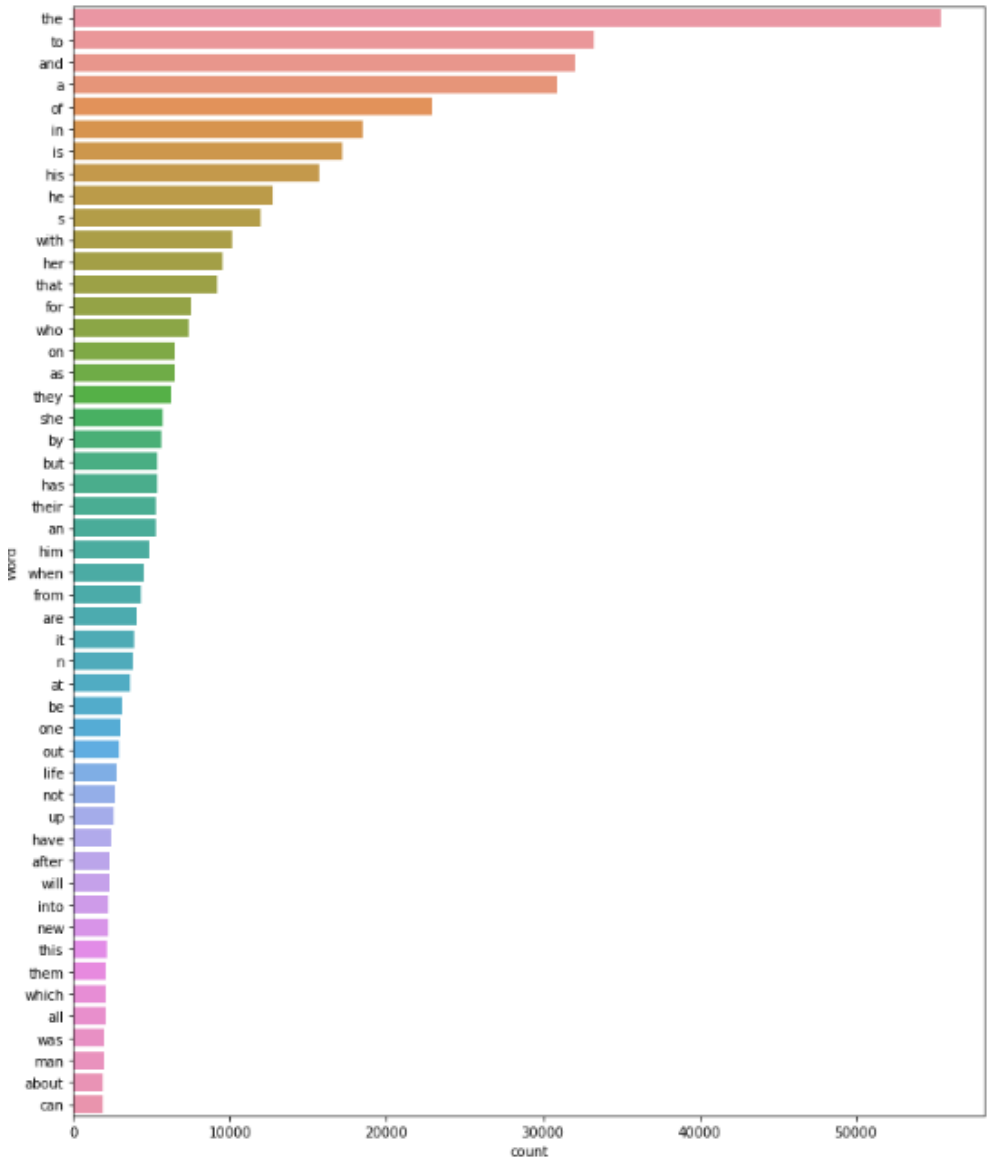
Inicialmente realizamos un análisis descriptivo de las variables de entrada y salida:

La siguiente grafica corresponde a la distribución de los registros de entrenamiento por género, donde se aprecia un mayor peso en películas de drama y comedia:



Posteriormente, realizamos limpieza y estandarización de la variable de descripción, se eliminaron caracteres especiales, números y convertimos el texto en minúsculas.

Así, gráficamente a continuación se observa la distribución de frecuencia por palabra de esta variable:



Por consiguiente, analizando la grafica anterior identificamos desde luego gran uso de artículos y conectores que no generan valor para predecir el género de una película.

Por lo cual aplicamos la función ***stopwords*** para remover dichas palabras inicialmente, así como una lematización y paso a forma raíz de las palabras.

Definición y calibración de modelos de clasificación

Realizamos la construcción de diferentes modelos de clasificación, donde evaluamos el desempeño de cada uno, utilizando la métrica AUC, sobre un 20% de los datos, y aplicando la función ***tfidf_vectorizer*** para el tratamiento del texto:

El primer modelo propuesto es una regresión logística sobre una transformación por pesos de las descripciones.

- Regresión logística
  AUC: 0.8817

Con este modelo se pudo dar cuenta la necesidad de probar diferentes metodologías Multi-label y la optimización de los parámetros en la transformación del texto a un espacio numérico; para ello se realizaron las siguientes pruebas:

Mientras la estrategia One Vs Rest classifier ajusta un modelo para cada uno de los géneros, la estrategia de ClassifierChain ajusta un modelo en cadena para cada categoría j que está asociada con las $j-1$ categorías anteriores.

- ClassifierChain
  AUC: 0.8805

LabelPowerSet: Esta estrategia de transformación de la variable respuesta, evalúa cada una de las posibles $2L$ combinaciones, en nuestro caso L=24 , transformando el problema Multi-label a un problema Single label; y para cada una de las combinaciones ajusta un modelo.

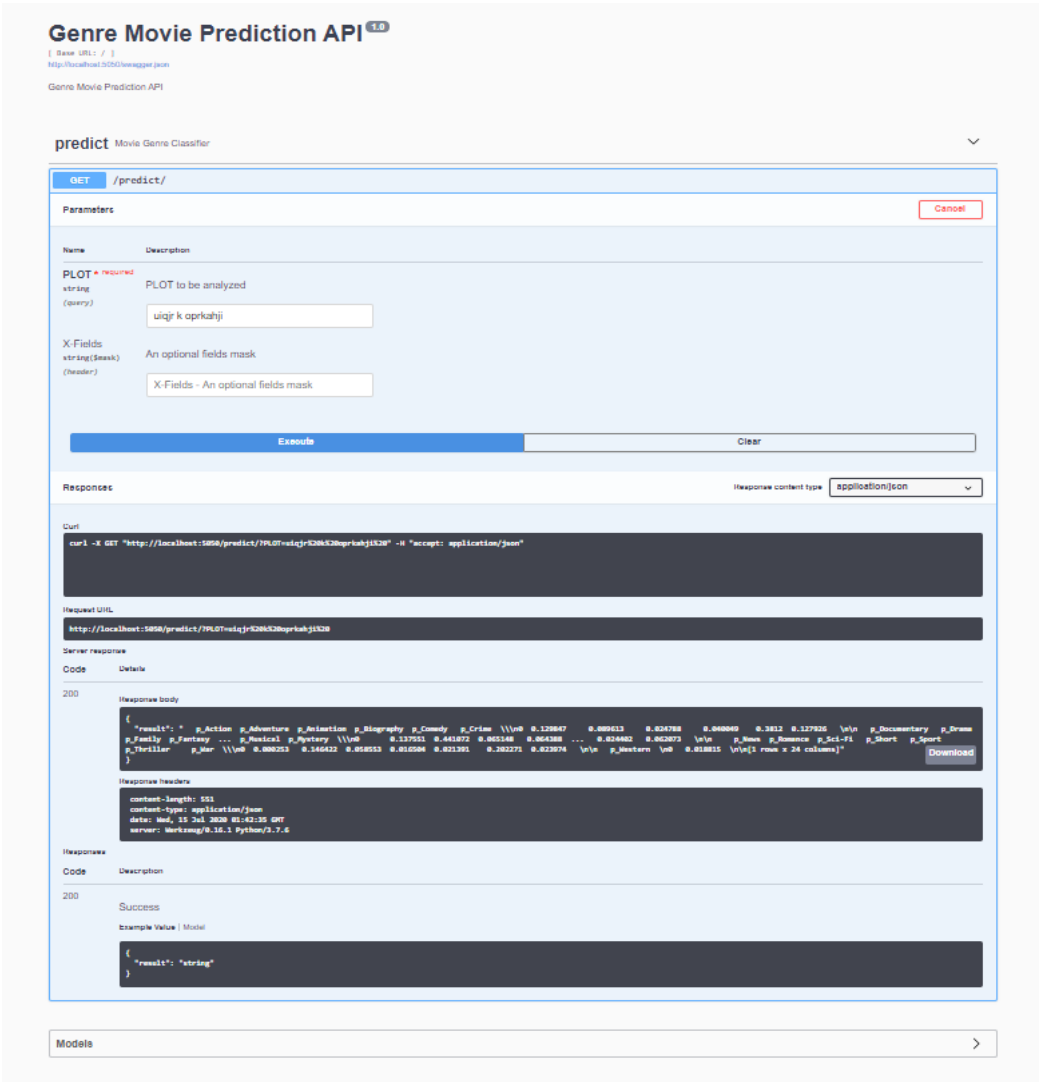
- LabelPowerset
  AUC: 0.7845

Construcción del API

Posterior al primer análisis de los resultados de los modelos de clasificación, realizamos la construcción del API, utilizando el modelo propuesto ***RandomForestClassifier,*** considerando facilidad de aplicación, buen rendimiento en ejecución, y no por su mejor desempeño.

Debido al peso del mejor modelo y como avanzamos en paralelo con el montado de la API, esta está ajustada con un modelo de Ramdonforest con 100 árboles (n_estimators=100) y una profundidad máxima de 10 (max_depth=10), igual manteniendo el OneVsRestClassifier para el Multilabel de la variable respuesta

URL: http://ec2-3-128-176-1.us-east-2.compute.amazonaws.com:5000/



Combinación de estimadores

Como alternativa útil de clasificación se propone el uso de combinación secuencial de estimadores "***Pipeline",*** a continuación se describen los modelos utilizados y sus parámetros de calibración, así como la medida de la métrica de desempeño AUC.

1. ***Bagging con Logistic***

   Se aplica TdidfVectorizer para vectorización de los datos con parámetros de calibración:
   max_df=0.25
   ngram_range=(1, 1)

   AUC: 0.8775

2. *Bagging con SVM*

Se aplica TdidfVectorizer para vectorización de los datos con parámetros de calibración:
max_df=0.25
ngram_range=(1, 1)

AUC: 0.8726

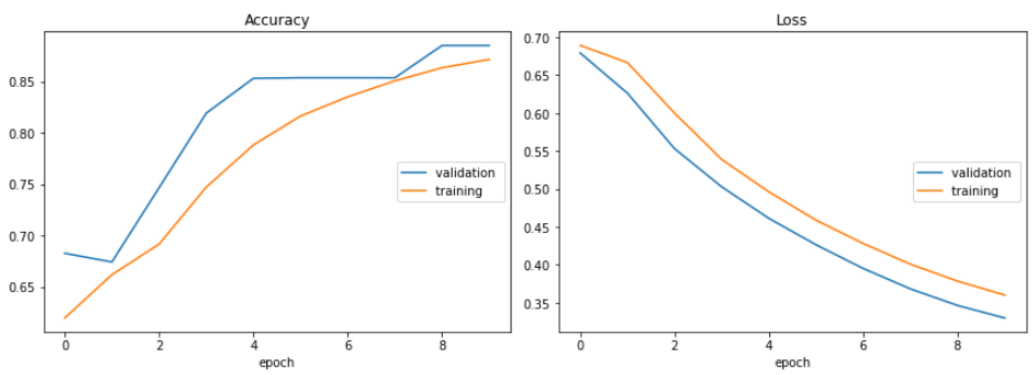
3. *MPL (Multi-layer Perceptron)*

Se aplica TdidfVectorizer para vectorización de los datos con parámetros de calibración:
max_df=0.25
ngram_range=(1, 1)

AUC: 0.8309

Calibración y entrenamiento de redes neuronales

Con el objetivo de lograr una métrica de mayor desempeño en el momento de clasificar el género de una película, se realiza el análisis de ajuste de parámetros para el entrenamiento de redes neuronales.



La grafica anterior corresponde a los primeros resultados obtenidos, Accuracy & Loss para el set de datos de entrenamiento, comparados con el set de validación.

Bidirectional Gated Recurrent Neural Network

Se probaron diferentes modelos de redes neuronales como:

1) Red neuronal simple con TfidVectorizer sobre el texto
2) LSTM Red Neuronal con Embedding de Count Vectorizer
3) CNN con TfidVectorizer
4) LSTM y CNN model
5) Bidirectional Gated Recurrent Neural Network
6) Bidirectional Gated Recurrent Neural and Convolutional Network

Siendo esta última, la que mejores resultados presentó. La idea de usar una red bidireccional es que la red aprendiera tanto de las palabras anteriores como de las siguientes para la clasificación en el género de la película. Además, utilizar GRU como estrategia de aprendizaje, tiene su explicación en que esta metodología es muy similar a LSTM, pero adopta un mecanismo de activación para rastrear el estado de la secuencia en lugar de usar una unidad de almacenamiento separada, lo que simplifica la estructura. Además, aplicar una convolución en la semántica del texto previamente, sintetiza la relación de algunas palabras sobre la elección de los géneros.

Adicionalmente, para el embedding de los datos, se usaron unos pesos pre-entrenados de las palabras por un grupo de investigación de Stanford que se encuentran disponibles en su sitio Web *http://nlp.stanford.edu/*

Resultados de entrenamiento de la red:

```
Train on 6316 samples, validate on 1579 samples
Epoch 1/50
6316/6316 [==============================] - 123s 20ms/step - loss: 0.3268 - auc_17: 0.6849 - val_loss: 0.2702 - val_auc_17: 0.7602
Epoch 2/50
6316/6316 [==============================] - 122s 19ms/step - loss: 0.2577 - auc_17: 0.7898 - val_loss: 0.2447 - val_auc_17: 0.8120
Epoch 3/50
6316/6316 [==============================] - 121s 19ms/step - loss: 0.2600 - auc_17: 0.8277 - val_loss: 0.2346 - val_auc_17: 0.8398
Epoch 4/50
6316/6316 [==============================] - 121s 19ms/step - loss: 0.2239 - auc_17: 0.8492 - val_loss: 0.2265 - val_auc_17: 0.8566
Epoch 5/50
6316/6316 [==============================] - 122s 19ms/step - loss: 0.2146 - auc_17: 0.8632 - val_loss: 0.2215 - val_auc_17: 0.8685
Epoch 6/50
6316/6316 [==============================] - 123s 19ms/step - loss: 0.2079 - auc_17: 0.8732 - val_loss: 0.2183 - val_auc_17: 0.8773
Epoch 7/50
6316/6316 [==============================] - 122s 19ms/step - loss: 0.2016 - auc_17: 0.8812 - val_loss: 0.2166 - val_auc_17: 0.8843
Epoch 8/50
6316/6316 [==============================] - 123s 19ms/step - loss: 0.1980 - auc_17: 0.8873 - val_loss: 0.2155 - val_auc_17: 0.8899
Epoch 9/50
6316/6316 [==============================] - 123s 20ms/step - loss: 0.1922 - auc_17: 0.8923 - val_loss: 0.2156 - val_auc_17: 0.8946
Epoch 10/50
6316/6316 [==============================] - 122s 19ms/step - loss: 0.1879 - auc_17: 0.8968 - val_loss: 0.2129 - val_auc_17: 0.8987
Epoch 11/50
6316/6316 [==============================] - 123s 20ms/step - loss: 0.1898 - auc_17: 0.9007 - val_loss: 0.2150 - val_auc_17: 0.9022
Epoch 12/50
6316/6316 [==============================] - 123s 19ms/step - loss: 0.1831 - auc_17: 0.9038 - val_loss: 0.2148 - val_auc_17: 0.9051
Epoch 13/50
6316/6316 [==============================] - 123s 19ms/step - loss: 0.1798 - auc_17: 0.9066 - val_loss: 0.2129 - val_auc_17: 0.9078
Epoch 14/50
6316/6316 [==============================] - 124s 20ms/step - loss: 0.1755 - auc_17: 0.9092 - val_loss: 0.2119 - val_auc_17: 0.9103
Epoch 15/50
6316/6316 [==============================] - 124s 20ms/step - loss: 0.1723 - auc_17: 0.9116 - val_loss: 0.2140 - val_auc_17: 0.9126
Epoch 16/50
6316/6316 [==============================] - 123s 20ms/step - loss: 0.1690 - auc_17: 0.9138 - val_loss: 0.2129 - val_auc_17: 0.9147
Epoch 17/50
6316/6316 [==============================] - 123s 19ms/step - loss: 0.1656 - auc_17: 0.9158 - val_loss: 0.2136 - val_auc_17: 0.9167
Epoch 18/50
6316/6316 [==============================] - 123s 19ms/step - loss: 0.1626 - auc_17: 0.9177 - val_loss: 0.2138 - val_auc_17: 0.9185
```

Sin embargo, este modelo en la parte de validación no mostró mejores resultados que los modelos probados anteriormente. Por ello, se vio en la necesidad de entrenar un modelo con mayor coste computacional.

La siguiente tabla corresponde a la optimización de parámetros de calibración del tfidf_vectorizer:

Después de hacer la limpieza a los textos, eliminar stop_words, caracteres especiales, y lematizar las palabras, realizamos una optimización del TF-IDF sobre los parámetros max_df (Cota superior de pesos considerados en el TF-IDF), min_df (Cota inferior de pesos considerados en el TF-IDF), max_feature (longitud máxima de tokens considerados en la matriz de pesos)y ngram_range(combinando de tokens, unigramas, bigramas,etc).

auc	max_df	max_feature	min_df	n_gram
0.8844	0.50	40000	0.0000	1,2
0.8844	0.50	46000	0.0001	1,2
0.8844	0.50	45000	0.0001	1,2
0.8843	0.50	40000	0.0001	1,2
0.8842	0.50	35000	0.0001	1,2
0.8842	0.50	30000	0.0000	1,2
0.8842	0.50	30000	0.0000	1,2
0.8842	0.50	30000	0.0000	1,2
0.8842	0.55	30000	0.0000	1,2
0.8842	0.50	35000	0.0000	1,2
0.8842	0.50	30000	0.0000	1,2
0.8842	0.50	30000	0.0000	1,2
0.8841	0.50	30000	0.0001	1,2
0.8841	0.55	30000	0.0001	1,2
0.8839	0.50	46000	0.0000	1,2
0.8839	0.50	45000	0.0000	1,2
0.8822	0.50	30000	0.0001	1,1

Después de realizar la optimización del TF-IDF, e ir acotando los límites de cada parámetro los mejores parámetros son max_df= 0.5 (Cota superior de pesos considerados en el TF-IDF), min_df=0.0 (Cota inferior de pesos considerados en el TF-IDF), max_feature=40000 (longitud máxima de tokens considerados en la matriz de pesos)y ngram_range= (1,2), es decir combinando unigramas y bigramas.

SVM Con optimización de parámetros

AUC: 08712

Finalmente, fue este modelo el que mejores resultados presentó sobre la muestra de Test de la competencia; sin embargo, es computacionalmente muy pesado para montar la API por ello se decide montar otro modelo con buen desempeño.

Referencias

- Santanu Pattanayak. Bangalore, Karnataka, India.
  Pro Deep Learning with TensorFlow.

- Long Cheng.Andrew Chi Sing Leung. Seiichi Ozawa (Eds.)
  Neural Information Processing. Springer.

- Josep M. Sopena, Pedro J. Ramos, Joan López-Moliner & Elizabeth Gilboy
  Composicionalidad, cómputo de estructura y redes neuronales

- Grupo de investigación de Stanford - *http://nlp.stanford.edu/*