

Entrega Proyecto Final – Clasificación Géneros de Películas

Jaime Andrés Unriza Vargas

Roberto Bustamante

Iván Amarillo Lozada

En este proyecto, el objetivo principal era mediante el uso de una base de datos de películas establecer (haciendo uso de sus parámetros de calibración) un modelo de predicción que fuera capaz de asignar las películas, acorde con el género al cual pertenecen. Cada observación contaba con el título de la película, su año del lanzamiento, la sinopsis de la película y los géneros a los que esta película pertenece (cada película puede pertenecer a uno o más géneros).

Para poder llevar a cabo las etapas de entrenamiento, testing y validación sobre el algoritmo entrenado, contamos con dos bases de datos diferentes una de entrenamiento y una disponible para poder someter el algoritmo a prueba y así determinar su capacidad de clasificación sobre datos que no hubiese visto antes.

Para iniciar este ejercicio, fue necesario entender el contenido de cada una de las bases de datos disponibles, siendo la base de datos de entrenamiento aquella que contaba con los géneros y así poder indicar o entrenar al modelo de acuerdo al género al que realmente pertenecía cada película y la base de testing sin esta información acerca de los géneros sobre la cual debíamos someter el algoritmo para poder identificar su capacidad de clasificación. A continuación, mostramos la estructura de cada base empleada.

| | year | | title | plot | genres | rating |
|------|------|---------------------------|---|---|--------------------|--------|
| 3107 | 2003 | | Most | most is the story of a single father who takes... | ['Short', 'Drama'] | 8.0 |
| 900 | 2008 | How to Be a Serial Killer | a serial killer decides to teach the secrets o... | ['Comedy', 'Crime', 'Horror'] | | 5.6 |
| 6724 | 1941 | A Woman's Face | in sweden , a female blackmailer with a disfi... | ['Drama', 'Film-Noir', 'Thriller'] | | 7.2 |
| 4704 | 1954 | Executive Suite | in a friday afternoon in new york , the presi... | ['Drama'] | | 7.4 |
| 2582 | 1990 | Narrow Margin | in los angeles , the editor of a publishing h... | ['Action', 'Crime', 'Thriller'] | | 6.6 |

dataTraining.head()

| | year | | title | plot |
|---|------|---------------------|--|---|
| 1 | 1999 | Message in a Bottle | | who meets by fate , shall be sealed by fate |
| 4 | 1978 | Midnight Express | | the true story of billy hayes , an american c... |
| 5 | 1996 | Primal Fear | | martin vail left the chicago da ' s office to ... |
| 6 | 1950 | Crisis | husband and wife americans dr . eugene and mr... | |
| 7 | 1959 | The Tingler | | the coroner and scientist dr . warren chapin ... |

dataTesting.head()

Preprocesamiento de Datos:

Para poder dar inicio al entrenamiento de los diferentes modelos y su comparación era necesario hacer un procesamiento de los datos, básicamente llevar los datos a un estado estandarizado que pudiera ser empleado en todos los modelos a entrenar por lo que utilizamos el `fit_transform()` sobre los géneros, así como dividir la data en fragmentos para entrenamiento y otros para testing, esto con el fin de poder realizar entonces estos análisis y estos entrenamientos de forma estandarizada. El código empleado se muestra a continuación.

```
1 # Definición de variable de interés (y)
2 dataTraining['genres'] = dataTraining['genres'].map(lambda x: eval(x))
3 le = MultilabelBinarizer()
4 y_genres = le.fit_transform(dataTraining['genres'])

1 # Separación de variables predictoras (X) y variable de interés (y) en set de entrenamiento y test usandola función train_test_split
2 X_train, X_test, y_train_genres, y_test_genres = train_test_split(X_dtm, y_genres, test_size=0.33, random_state=42)
```

Estandarización y separación de la data en bases de entrenamiento y testing

Asimismo, como parte del ejercicio de preprocesamiento de los datos empleamos una función para lematizar las palabras con su respectiva posición y concatenar el título de año con la trama y poder cruzar así una matriz X lematizada durante el proceso de entrenamiento. El código se expone a continuación.

```
1 # Función para lematizar palabras con su respectiva POS
2 def lematize_with_pos(text):
3     lemmatizer = WordNetLemmatizer()
4     # Tokenizar las palabras
5     tokens = word_tokenize(text)
6     # Etiquetar POS de cada palabra
7     tagged_tokens = nltk.pos_tag(tokens)
8     # Lematizar cada palabra según su POS
9     lemmatized_tokens = [lemmatizer.lemmatize(word, pos=pos_tag[0].lower())
10                          if pos_tag[0].lower() in ['n', 'v', 'a', 'r'] else word
11                          for word, pos_tag in tagged_tokens]
12     return ' '.join(lemmatized_tokens)
13
14 # Concatenar título y año con la trama
15 dataTraining['plot_title_year'] = dataTraining['plot'] + ' ' + dataTraining['title'] + ' ' + dataTraining['year'].astype(str)
16 dataTesting['plot_title_year'] = dataTesting['plot'] + ' ' + dataTesting['title'] + ' ' + dataTesting['year'].astype(str)
17
18 # Aplicar lematización con POS a los datos de entrenamiento
19 X_train_lemmatized = [lemmatize_with_pos(text) for text in dataTraining['plot_title_year']]
20
21
22 # Crear el vectorizador CountVectorizer
23 vect = CountVectorizer(lowercase=True, strip_accents='ascii', stop_words='english', ngram_range=(1,2), max_features=10000)
24
25 # Definición de variables predictoras (X)
26 X_dtm = vect.fit_transform(X_train_lemmatized)
27 print("Shape of DTM:", X_dtm.shape)
28
29 # Definición de variable de interés (y)
30 dataTraining['genres'] = dataTraining['genres'].map(lambda x: eval(x))
31 mlb = MultilabelBinarizer()
32 y_genres = mlb.fit_transform(dataTraining['genres'])
33
```

Shape of DTM: (7895, 10000)

Función y proceso de lematización de la matriz X.

Este proceso nos permite ya tener la data lista para empezar a entrenar los modelos. La separación definitiva de la data se presenta en el siguiente código:

```

2 # Separación de variables predictoras (X) y variable de interés (y) en set de entrenamiento y test usandola función train_test_split
3 X_train, X_test, y_train_genres, y_test_genres = train_test_split(X_dtm, y_genres, test_size=0.20, random_state=42)
4
5
6 # Definición de dimensiones de entrada, variables predictoras
7 dims = X_train.shape[1]
8 print(dims, 'input variables')
9
10 # Definición de dimensiones de salida, variables de interés
11 output_var = y_train_genres.shape[1]
12 print(output_var, ' output variables')
13
14 ##### modelos

```

10000 input variables
24 output variables

Separación en fragmentos de train y test lematizados.

Modelos Entrenados, Comparación y Selección:

Para poder identificar el mejor modelo se entrena son 3 diferentes con el fin de compararlos basándonos en el ROC AUC Score y así poder determinar aquel que tuviera el mejor nivel predictivo en torno al género de cada película en la base de datos.

Modelo de Random Forest:


Nuestro primer modelo escogido es un modelo de random forest entrenado empleando la clase OneVsRestClassifier(), tomando 100 estimadores. Al entrenarlo encontramos un ROC AUC Score de 0.55 como se muestra en la imagen a continuación.

✓ Modelo de Radom Forest

```

[10] 1 rf_model = OneVsRestClassifier(RandomForestClassifier(n_estimators=100, random_state=42))
      2 rf_model.fit(X_train, y_train_genres)
      3 y_pred_rf = rf_model.predict(X_test)
      4 roc_auc_rf = roc_auc_score(y_test_genres, y_pred_rf, average='macro')
      5 print("ROC AUC Score (Random Forest):", roc_auc_rf)

```

 ROC AUC Score (Random Forest): 0.5545072093765353

Modelo de Random Forest.

Modelo de Logistic Regression:

Posteriormente, entrenamos un modelo de regresión logística tomando como un máximo de 1.000 iteraciones y resultando en un ROC AUC Score de 0.65, resultando en un mejor Score que el previamente evidenciado mediante el entrenamiento y testing del modelo de Random Forest

✓ Modelo de Logistic Regression

```
[11] 1 lr_model = OneVsRestClassifier(LogisticRegression(max_iter=1000, random_state=42))
      2 lr_model.fit(X_train, y_train_genres)
      3 y_pred_lr = lr_model.predict(X_test)
      4 roc_auc_lr = roc_auc_score(y_test_genres, y_pred_lr, average='macro')
      5 print("ROC AUC Score (Logistic Regression):", roc_auc_lr)
```

ROC AUC Score (Logistic Regression): 0.6538607677270819

Modelo de Red Neuronal:

Por último, entrenamos un modelo de red neuronal secuencial resultando en un ROC AUC Score de 0.76, este resultado acompañado de los parámetros de densidad y dropout se exponen a continuación.

Model: "sequential"

| Layer (type) | Output Shape | Param # |
|-------------------------|--------------|----------|
| dense (Dense) | (None, 3500) | 35003500 |
| dropout (Dropout) | (None, 3500) | 0 |
| dense_1 (Dense) | (None, 24) | 84024 |
| activation (Activation) | (None, 24) | 0 |

=====
Total params: 35087524 (133.85 MB)
Trainable params: 35087524 (133.85 MB)
Non-trainable params: 0 (0.00 Byte)

None
Epoch 1/5
632/632 [=====] - 683s 1s/step - loss: 0.1124 - val_loss: 0.0984
Epoch 2/5
632/632 [=====] - 695s 1s/step - loss: 0.0946 - val_loss: 0.0910
Epoch 3/5
632/632 [=====] - 703s 1s/step - loss: 0.0863 - val_loss: 0.0867
Epoch 4/5
632/632 [=====] - 697s 1s/step - loss: 0.0806 - val_loss: 0.0840
Epoch 5/5
632/632 [=====] - 681s 1s/step - loss: 0.0762 - val_loss: 0.0820
50/50 [=====] - 1s 18ms/step
ROC AUC Score (Red Neuronal): 0.7606731774374434

La comparación de los resultado arroja un mejor modelo, siendo este la red neuronal, razón por la cual escogimos el mismo para poder someterlo a competencia en la plataforma de Kaggle.

Comparación de Modelos

Red Neuronal: ROC AUC = 0.7606731774374434

Random Forest: ROC AUC = 0.5545072093765353

Logistic Regression: ROC AUC = 0.6538607677270819

✓ Selección Mejor Modelo

Analizando los resultados de las métricas de desempeño, en este caso y específicamente el ROC-AUC (Área bajo la curva) para cada modelo. El ROC-AUC es una medida de la capacidad de un modelo para distinguir entre clases. El valor más alto indica un mejor rendimiento.

Resultados de los Modelos

- Red Neuronal: ROC AUC = 0.7494
- Random Forest: ROC AUC = 0.5545
- Logistic Regression: ROC AUC = 0.6539

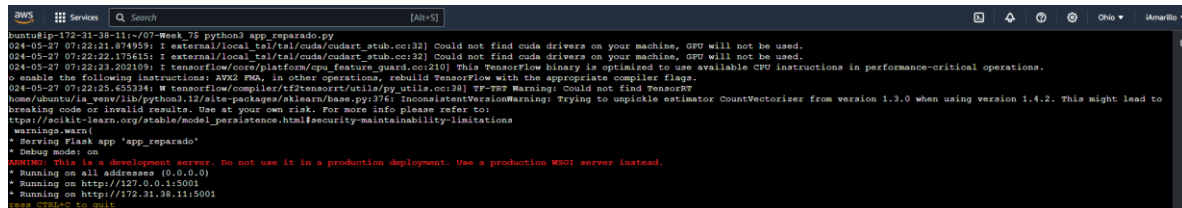
Mejor Modelo Red Neuronal (ROC AUC = 0.7494) Indica que la red neuronal tiene una buena capacidad para predecir la probabilidad de que una película pertenezca a un género en particular. Si bien es importante considerar el factor tiempo de entrenamiento, la complejidad del modelo y la interpretabilidad, el modelo Red Neuronal permite obtener un mejor desempeño significativamente superior a los otros dos modelos Random Forest y Logistic Regression.

Los resultados obtenidos se pueden considerar como esperables en el caso de problemas de NLP (Procesamiento de Lenguaje Natural), las redes neuronales y esquemas LSTM nos demuestran que son muy efectivas.

Fragmento en Jupyter del modelo escogido y justificación.

Disponibilización del modelo en API:

Al intentar desplegar el microservicio, dado el tamaño del modelo seleccionado, el cual se carga empleando la lectura de 'model.h5', no logra cargar la página (se pasa el tiempo de espera y retorna un error por tiempo de ejecución). Sin embargo a continuación, exponemos el código y el runtime en AWS para garantizar que está funcionando, al parecer Chrome retorna error por tiempo y por ello no se logra desplegar en producción.




```
aws
Services
Search
[Alt+S]
buntub@ip-172-31-38-11:~/07-Week 78$ python3 app_reparado.py
024-05-27 07:22:21.874959: I external/local_tsl/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
024-05-27 07:22:22.175615: I external/local_tsl/tsl/cuda/cudart_stub.cc:32] Could not find cuda drivers on your machine, GPU will not be used.
024-05-27 07:22:23.202109: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
024-05-27 07:22:25.655394: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
/home/buntub/.local/lib/python3.12/site-packages/alembic/base.py:176: InconsistentVersionWarning: Trying to unpickle estimator CountVectorizer from version 1.3.0 when using version 1.4.2. This might lead to
breaking code or invalid results. Use at your own risk. For more info please refer to:
https://mckit-learn.org/stable/model_persistence.html#security-maintainability-limitations
WARNING:warn()
* Serving Flask app 'app_reparado'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5001
* Running on http://172.31.38.11:5001
Press CTRL+C to quit
```

```

1 from flask import Flask, request, jsonify
2 import tensorflow as tf
3 from keras.models import load_model
4 import pickle
5
6 app = Flask(__name__)
7
8 # Cargar el modelo y el vectorizador
9 model = load_model('model.h5')
10 with open('vectorizer.pkl', 'rb') as f:
11     vect = pickle.load(f)
12
13 @app.route('/predict', methods=['POST'])
14 def predict():
15     data = request.json
16     plot_title_year = data['plot_title_year']
17
18     # Preprocesamiento
19     X = vect.transform([plot_title_year])
20
21     # Predicción
22     prediction = model.predict(X)
23
24     # Crear respuesta
25     response = {
26         'predictions': prediction.tolist()
27     }
28     return jsonify(response)
29
30 if __name__ == '__main__':
31     app.run(debug=True)
32

```



































 * Serving Flask app '__main__'
 * Debug mode: on
 INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on <http://127.0.0.1:5000>
 INFO:werkzeug:Press CTRL+C to quit
 INFO:werkzeug: * Restarting with stat

Conclusiones:

A manera de conclusión y a pesar de seleccionar el modelo con el mejor indicador dentro de los tres modelos entrenados, no logramos una posición alta en la competencia, sugiriendo que nuestro modelo en relación a los otros grupos no predice de forma adecuada. Se necesita calibrar parámetros y probablemente incluir en el análisis otros algoritmos más avanzados al igual que probar con otras aproximaciones de clasificación para conseguir mejores resultados.

Desde el trabajo en grupo hemos identificado que debemos reforzar el proceso de calibración de los modelos, que escoger modelos complejos como una red neuronal por si sola, no garantiza un alto poder predictivo y de clasificación. Trabajaremos en ello para robustecer nuestro criterio de calibración a futuros modelos a desarrollar.

A la fecha de elaboración del presente documento, el top 10 de grupos dentro de la competencia superan scores sobre 0.89.

| # | Team | Members | Score | Entries | Last | Join |
|----|--------------------------|---|---------|---------|------|------|
| 1 | Grupo_20 |     | 0.92869 | 7 | 1d | |
| 2 | Team 4 |    | 0.91754 | 22 | 21m | |
| 3 | Grupo 6 |     | 0.91077 | 22 | 14h | |
| 4 | Team 7 |     | 0.90976 | 7 | 3d | |
| 5 | Equipo 3 |     | 0.90685 | 4 | 1d | |
| 6 | Juan Sebastián Hernández |  | 0.90553 | 4 | 1d | |
| 7 | Team 29 |     | 0.89400 | 21 | 32m | |
| 8 | Team 12 |     | 0.89349 | 22 | 1h | |
| 9 | grupo 32 |     | 0.89306 | 2 | 6d | |
| 10 | Grupo 18 |   | 0.89306 | 11 | 7h | |