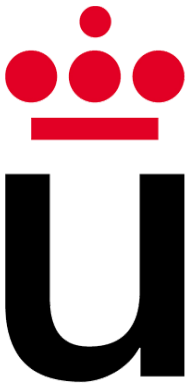

Universidad
Rey Juan Carlos

# Práctica 1: Técnicas de NLP para clasificación de géneros.

**Jorge Camacho Mejias**

**Alberto Pérez Paredes**

Grado en Inteligencia Artificial

# Índice

# 1. Introducción

El objetivo principal de esta práctica es aprender a cómo enfrentar a un problema real de *PLN* por lo que el desarrollo de esta práctica se desarrollará en los siguientes puntos:

- Elección del problema y búsqueda de los datasets.
- Exploración, armonizado y tratado de los distintos datasets.
- Entrenamiento de los modelos de *PLN* clásicos.
- Entrenamiento del los modelos basados en transformers.
- Comparación de los modelos y su rendimiento.
- Explicabilidad en los modelos.
- Conclusiones.

Además de la explicación teórica, se explicará de forma detallada que dificultades se encontraron en cada punto del desarrollo de esta práctica y la solución aplicada a cada uno de ellos.

## 2. Elección y Tratamiento del Dataset

Para comenzar el desarrollo de la práctica en cuestión decidimos optar por un tipo de problema que no habíamos resuelto nunca, decidimos hacer una clasificación de géneros cinematográficos respecto al título, año y la sinopsis de la película clasificar.

Al ponernos a buscar datasets para el problema descubrimos que este tipo de problemas de clasificación se conoce como *Genre Classification* y si bien es un problema con bastante información se suele aplicar más a géneros musicales. Es decir, existen datasets pero no en exceso.

Al tratarse de una clasificación multiclase solo fue necesario encontrar y armonizar dos datasets y elegimos:

- adrienheymans/imdb-movie-genres.
- jquigl/imdb-genres.

Ambos extraidos desde HuggingFace.

### 2.1. Exploración y Armonizado

El proceso de Exploración de Datos reveló varios desafíos iniciales:

- **Desbalanceo de clases:** Se detectó que ciertos géneros (como Drama o Comedia) estaban sobrerrepresentados frente a otros minoritarios, lo cual podría sesgar las predicciones.
- **Cantidad absurda de clases:** Siguiendo con el punto anterior, el dataset presenta una fragmentación severa con un total de 27 categorías diferentes. La distribución es extremadamente desigual, existiendo clases con menos de 100 muestras. Esto dificulta estadísticamente que el modelo aprenda patrones generalizables para las categorías minoritarias, sesgando las predicciones hacia las clases dominantes.
- **Mala calidad del dataset:** Se detectó una contaminación significativa en el cuerpo del texto derivada de una mala configuración del *web crawler* con el que probablemente se extrajeron los datos. Un porcentaje considerable de las descripciones no contiene información semántica, sino el texto residual *"View full summary"*. Esto introduce ruido puro en el entrenamiento y obliga a descartar dichas muestras, reduciendo el tamaño efectivo del dataset útil.

### 2.2. Soluciones aplicadas

Como explicamos en la introducción la memoria constará de el desarrollo de los problemas encontrados y como planteamos las soluciones.

- **Desbalanceo de clases:** Para el desbalanceo de clases aplicamos técnicas de estratificación, además de meter pesos en las clases minoritarias para que su fallo penalizara más. Aunque la forma correcta de tratar con este problema sería hacer algún tipo de aumento de datos de las clases menos representativas, esto estaba fuera del contexto de la práctica, por lo que con el siguiente punto creemos haber conseguido que el modelo infiera a través de los textos.
- **Cantidad absurda de clases:** Como se mencionó en los problemas, la alta cantidad de clases no tenía sentido ya que muchas de las clases a penas tenían representación y podían chocar entre ellas confundiendo al modelo (p.e. **Histórica** y **Bélica**) por lo que nuestra idea fue condensar estas clases en meta-clases que englobaran más contexto para que el modelo fuera capaz de inferir, quedándonos así con 6 clases.

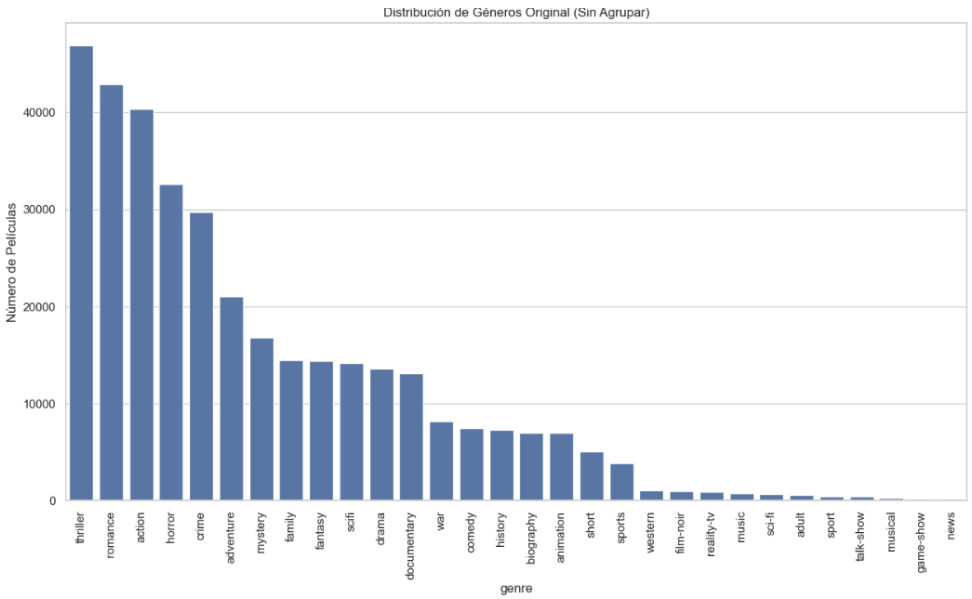


Figura 1: Distribución de clases en el dataset antes del armonizado.

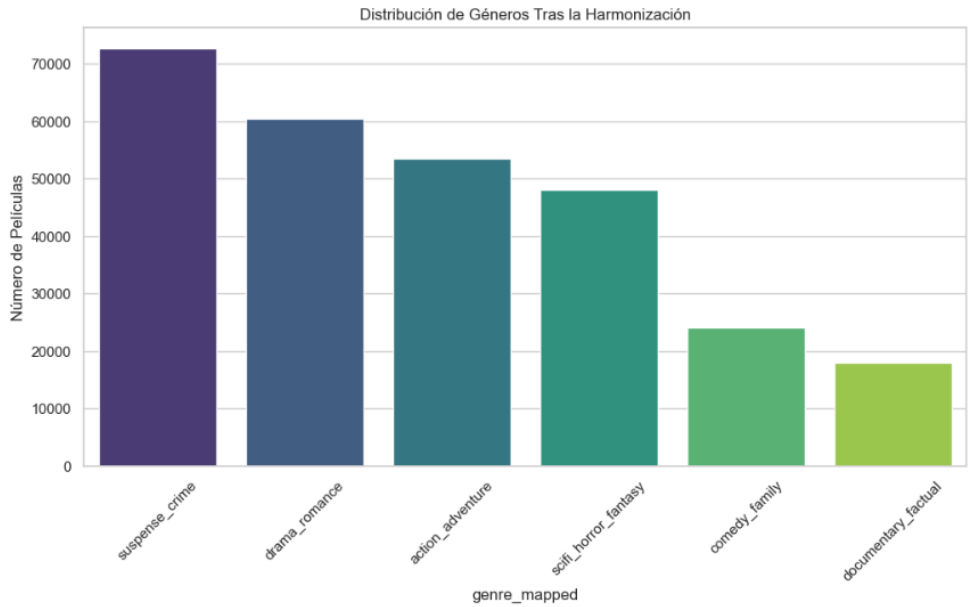


Figura 2: Distribución de clases en el dataset tras el armonizado.

- **Mala calidad del dataset:** Por último, el punto más complicado al que nos enfrentamos fue la baja calidad de los datasets y, si bien nos planteamos el cambiar de datasets y volver a hacer tanto el armonizado como la exploración de datos, creímos que es importante también mostrar que muchas veces en *PLN* el problema no viene en el hecho de que no estés aplicando el modelo correcto si no que los datasets pueden no ser fiables, por ello decidimos eliminar los datos no válidos pasándose así de más de 800.000 datos en entrenamiento a poco más de 200.000, es decir 3/4 partes del dataset tuvo que ser eliminado al tratarse de datos de baja calidad, valorando así la calidad frente a la cantidad.

# 3. Modelos Clásicos de NLP (Baseline)

Para establecer una línea base (*baseline*) y verificar si realmente es necesario el uso de redes neuronales profundas, implementamos modelos clásicos basados en frecuencias. Utilizamos **TF-IDF** (*Term Frequency - Inverse Document Frequency*) para vectorizar el texto.

## 3.1. Problemas y Soluciones

- **Pérdida de contexto semántico:** El principal problema teórico de modelos como Naive Bayes o SVM con TF-IDF es que funcionan bajo el enfoque *Bag-of-Words*. Pierden el orden de las palabras y el contexto.
- **Dimensionalidad:** Al tener un vocabulario extenso, la matriz dispersa generada era enorme.
- **Solución aplicada:** Aceptamos estas limitaciones asumiendo estos modelos solo como punto de comparación. Se entrenaron cuatro variantes: **Logistic Regression** (Modelo linear por defecto), **Naive Bayes** (referencia probabilística), **Linear SVM** (bueno en alta dimensionalidad) y **Random Forest** (bagging), obteniendo resultados que rondaban el 45-50 % de accuracy, lo cual definió nuestro objetivo a batir.

# 4. Modelos Basados en Transformers

Esta es la sección crítica de la práctica. Decidimos utilizar **DeBERTa V3** debido a su superioridad frente a BERT. Al principio decidmos utilizar el modelo más potente que encontramos, para darnos cuenta los problemas que puede generar esto y luego cambiar a un modelo que se ajustara de una forma más coherente a nuestro problema.

## 4.1. El problema con DeBERTa-v3-large

- **Demasiado pesado para la GPU:** Todos los modelos entrenados en esta práctica fueron entrenados con una NVIDIA RTX 4070 Laptop (8GB VRAM). Sin embargo al intentar entrenarla con nuestros datos nos dimos cuenta del tiempo excesivo que tardaba (unas 50 horas por época) esto nos extrañó dado que aunque los datos empezaban a ser medianamente pesados estos modelos han sido entrenados con muchos más.
  Tras investigar nos dimos cuenta de que el problema no estaba en los datos, si no en el hardware que estábamos utilizando junto al modelo, ya que aunque una GPU de ordenador portátil te diga que dispone de 16 GB de memoria estos se dividen en 8 GB de VRAM (memoria de la GPU) y 8 GB de RAM compartida por tu ordenador, el problema es que la RAM es unas 10 veces más lenta que la VRAM lo que crea un cuello de botella y hace que la GPU no trabaje como debería.
  Por lo que al intentar cargar el modelo grande, junto a varios datos (batch size alto) y una precisión por defecto (FP32) hacía que la VRAM no pudiera con todo y tuvieramos que usar RAM compartida.

- **Sobreajuste del Modelo Large (8 % Accuracy):** Tras arreglar el entrenamiento y hacer que el modelo entrenara con nuestros datos nos dimos cuenta de que al tratarse de un modelo tan grande era capaz de sobreajustar cada uno de los datos dados en el entrenamiento y haciendo asi que alcanzara un 100 % de accuracy sobre los datos de entrenamiento mientras que en la parte de inferencia solo acertara el 8 %, haciendo que todas las predicciones fueran de la clase 1.

## 4.2. Soluciones aplicadas

Para solventar estos bloqueos técnicos, aplicamos ingeniería de optimización:

- **Optimización de Memoria (FP16):** . Esto redujo el peso del modelo a la mitad sin perder rango dinámico, solucionando el cuello de botella de la VRAM.

```
training_args = TrainingArguments(
    fp16=True,                          # Uso de float16
    per_device_train_batch_size=2,      # Batch minimo
    gradient_accumulation_steps=8,      # Simular batch de 16
)
```

Listing 1: Argumentos de entrenamiento optimizados

- **Cambio de Arquitectura (DeBERTa-Large → RoBERTa-Base):** Dado que el modelo *Large* sufría de inestabilidad con pocos datos por lote, la solución definitiva fue migrar a `RoBERTa-base`. Al ser más ligero, permitió estabilizar los gradientes y el entrenamiento fluyó correctamente, subiendo la precisión drásticamente.

# 5. Resultados y Comparación

A continuación, se presentan los resultados finales comparando las métricas obtenidas por los modelos clásicos frente a los modelos transformer.

Cuadro 1: Tabla Comparativa de Resultados (Accuracy y Macro F1)

| ID | Modelo | Accuracy | Macro F1 |
|---|---|---|---|
| 4 | **RoBERTa-base** | **0.605903** | **0.621853** |
| 1 | LogReg | 0.529224 | 0.534691 |
| 0 | Naive_Bayes | 0.528791 | 0.505870 |
| 2 | Linear_SVM | 0.522022 | 0.522958 |
| 3 | Random_Forest | 0.482473 | 0.480092 |
| 5 | deberta-v3-large* | 0.087004 | 0.026680 |

*Nota: Como se puede obervar, deberta-v3-large no ha conseguido converger.

**Análisis de los resultados:**

- **Limitación de los Modelos Clásicos:** Como se puede observar, todos los modelos básicos presentan un rendimiento muy similar y un estancamiento alrededor del 52–53 % de accuracy. Esto nos indica que este es el máximo que se puede obtener con representaciones de texto básicas en este dataset concreto.
- **Supremacía del modelo Transformer:** El modelo *RoBERTa-base* fue claro ganador, superando a los modelos clásicos con un margen bastante significativo. Esto nos demuestro una vez mas que la capacidad de los Transformers para capturar el contexto semántico y las relaciones entre palabras es bastante superior a los enfoques basados en frecuencia (TF-IDF) de los modelos tradicionales.
- **A veces lo mas potente no es lo mejor:** Como podemos observar, existe un importante fallo en la convergencia del modelo deberta-v3-large. Este, muestra un accuracy mas bajo incluso que el azar.
- **Reproducibilidad y ajuste:** Como es lógico, para que esta comparativa tenga sentido, todos los modelos básicos han sido entrenado con el mismo procesamiento y tratando de encontrar su máximo potencial mediante un ajuste exhaustivo de loa parámetros. Con los modelos transformer igual, ambos han sido entrenados durante tres épocas, y buscando su máximo rendimiento mediante el ajuste de hipermetamorfosis.

## 5.1. Matriz de Confusión de mejor modelo (RoBERTa-base)

Cuadro 2: Matriz de Confusión para *RoBERTa-base*

| Real \Predicción | Act/Adv | Com/Fam | Doc/Fact | Dra/Rom | Sci/Hor | Sus/Cri |
|---|---|---|---|---|---|---|
| **action_adventure** | **5171** | 1053 | 183 | 865 | 1633 | 1802 |
| **comedy_family** | 307 | **3158** | 215 | 583 | 418 | 139 |
| **documentary_factual** | 39 | 100 | **3244** | 185 | 19 | 23 |
| **drama_romance** | 1202 | 893 | 712 | **7690** | 603 | 989 |
| **scifi_horror_fantasy** | 750 | 827 | 38 | 364 | **6471** | 1178 |
| **suspense_crime** | 2009 | 343 | 98 | 1006 | 3257 | **7833** |

**Análisis de la Matriz:**

- **Diagonal:** Se puede observar una alta densidad en la diagonal principal, lo que confirma que el modelo clasifica correctamente la mayoría de los ejemplos, podemos destacar los aciertos en las clases *Drama/Romance* y *Suspense/Crime*.
- **Confusión Semántica:** Existe una notable dispersión de errores entre *Action*, *Sci-Fi* y *Suspense*, esto, es debido a que comparten un vocabulario común (violencia, tensión, muerte).
- **Distinción:** La categoría *Documentary/Factual* es la más limpia y aislada, mostrando muy poca confusión con los géneros de ficción, si lo analizamos bien, vemos que es porque su vocabulario muy distinto al resto.

# 6. Explicabilidad

Hemos dividido esta sección en dos bloques diferenciados —Modelos Clásicos y Modelos Transformers— ya que sus arquitecturas requieren técnicas de interpretación distintas. Para realizar este análisis, hemos seleccionado una muestra de 10 predicciones correctas y 10 incorrectas, correspondientes al mejor modelo de cada tipo: Regresión Logística (clásico) y RoBERTa-base (Transformer).

## 6.1. Modelos Clásicos (Regresión Logística)

Para el modelo de Regresión Logística hemos utilizado la librería **LIME**. Esta herramienta nos ha proporcionado la siguiente información:

- **Peso de las palabras:** No ha ayudado a identificar qué palabras otorgaron más peso a una categoría u otra.
- **Direccionalidad:** Muestra si una palabra aporta probabilidad positiva es decir, apoya la predicción o negativa, contradice la predicción.
- **Incertidumbre del modelo:** Esto es muy importante, porque nos dice con qué seguridad el modelo ha predicho cada una de las etiquetas.


Figura 3: Ejemplo de Acierto con Regresión Logística (LIME).

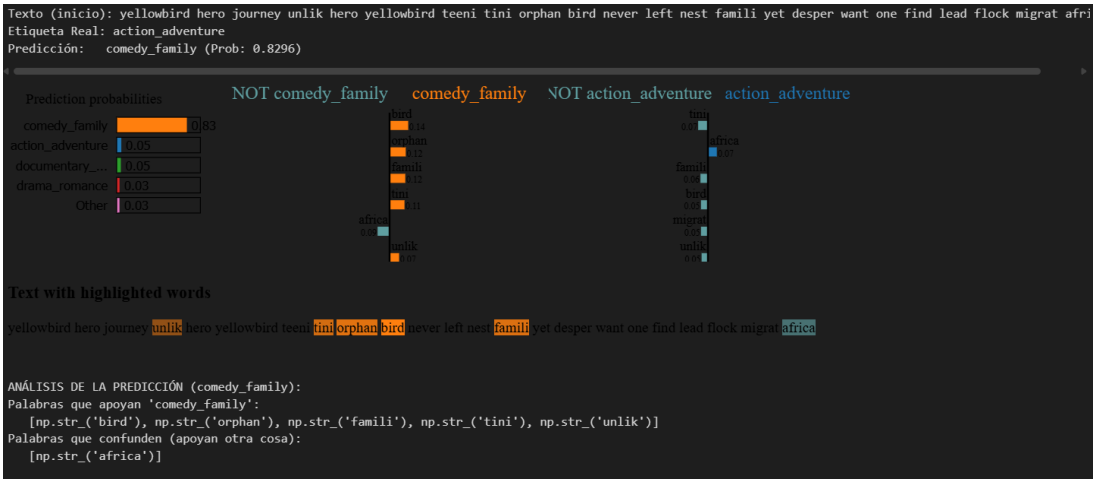


Figura 4: Ejemplo de Error con Regresión Logística (LIME).

## 6.2. Modelos Transformers (RoBERTa-base)

Para RoBERTa-base, al tratarse de un modelo de 'caja negra' más complejo, hemos utilizado la librería **Captum**, en la que usamos una técnica llamada *Integrated Gradients* sobre los *embeddings* del texto. Gracias a esto hemos podido recopilar la siguiente información:

- **Contribución de cada token:** Sabemos que peso tiene cada *token* en la decisión final del modelo.
- **Mapa de Calor:** En verde, podemos ver las palabras que empujan al modelo a elegir ese género, y en rojo las palabras que confunden al modelo o le sugieren un género distinto.
- **Interpretación:** Gracias a esto, podemos concluir que el modelo no se limita a memorizar palabras sueltas, sino que detecta conceptos abstractos (como *muerte*, *amor* o *viaje*) e ignora las palabras de relleno o stopwords (como *el*, *de*, *un*) para clasificar la película. No obstante, se observa que en algunos ejemplos resalta nombres propios de personas, que no deberían de influir en la predicción salvo casos muy concretos, esto podría dificultar la generalización.

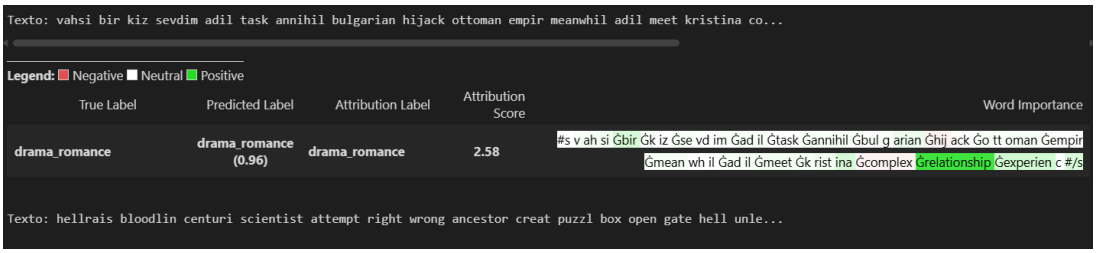


Figura 5: Ejemplo de Acierto con RoBERTa (Integrated Gradients).


Figura 6: Ejemplo de Error con RoBERTa (Integrated Gradients).

# 7. Conclusiones

El desarrollo de esta práctica nos ha permitido extraer diferentes conclusiones sobre las aplicaciones de *PLN*:

1. **Calidad de los Datos:** Como hemos ido resaltando a lo largo de la memoria, poseer datos de calidad es una parte fundamental para la resolución del problema. En nuestro caso no contábamos con ellos, pero la armonización de estos y de las etiquetas han sido claves. Ningún modelo, por complejo que fuera, lograba aprender correctamente sin esta limpieza y reducción de clases. Aun así, la mala calidad de los textos nos ha limitado a la hora de conseguir un accuracy competitivo.

2. **Superioridad Arquitectónica:** Hemos confirmado que los modelos basados en Transformers (*robert-base*) superan la barrera del 52 % de *accuracy* en la que los modelos clásicos se estancan. Aun así, estos son un muy buen punto de partida y una muy buena solución si no disponemos del tiempo y de los recursos computacionales necesarios.

3. **La solución no es siempre la fuerza bruta:** El fracaso del modelo *Large* (8 % de acierto) frente al éxito del modelo *Base* (60 %) nos ha demostrado que no siempre lo mas potente es lo que mejores resultados nos va a dar.

4. **Interpretabilidad:** Hemos aplicado técnicas de explicabilidad vistas en clase, que nos han ayudado a entender como 'piensa' el modelo por dentro.

# 8. Enlaces a conversaciones con Gemini:

- https://gemini.google.com/share/d06a2b163ab0
- https://gemini.google.com/share/fb45d9756812
- https://gemini.google.com/share/349c2e453a5b