

Proyecto NLP - Clasificación de libros según género literario

GRUPO 1:

Luis A. C. (alcaluis@alumni.uv.es), Rebeca C. B. (recombar@alumni.uv.es),

Marta M. M. (memuiz@alumni.uv.es), Alejandra V. (ave6@uv.es)

Universitat de València

Máster en Ciencia de Datos - 2024/2025

Roberto F. M. (roberto.fernandez@uv.es), Yolanda V.G. (yolanda.vives@uv.es)

24/05/2025

Resumen

El desarrollo del siguiente trabajo consistirá en la clasificación de libros en sus respectivos géneros dada la sinopsis de estos. Bajo este objetivo obtendremos los datos mediante recopilación automática de la web OpenLibrary, aplicaremos técnicas de limpieza y normalización de texto además de probar distintas metodologías de clasificación con modelos propios y modelos pre-entrenados.

Palabras clave: NLP, Webscraping, Clasificación de Género, Libros, Embeddings, BoW, preprocesado de texto, multi-etiqueta, TFIDF, XGBoost, Regresión Logística, SGDClassifier, Distil-BERT, ...

ÍNDICE

1. WebScraping: OpenLibrary API.....	4
2. Tratamiento de los datos.....	5
2. 1. Preprocesado.....	5
2. 2. Extracción de características.....	6
2. 3. Análisis EDA.....	7
3. Modelos.....	11
3. 1. Modelo propio de clasificación de género.....	11
3.1.1 Resultados del modelo propio.....	13
3.1.2 Análisis de los resultados del mejor modelo propio.....	16
3. 2. Modelo pre-entrenado de análisis de género.....	20
3.2.1 Resultados del modelo pre-entrenado.....	23
3.2.2 Análisis de los resultados del mejor modelo pre-entrenado.....	24
4. Conclusión, comparativa de modelos y posibles mejoras.....	27

ÍNDICE de Figuras

Figura 1. JSON resultado de consulta API.....	4
Figura 2. Ejemplo sinopsis preprocesada.....	5
Figura 3. Distribución de géneros preprocesado.....	6
Figura 4. Distribución del número de palabras por género.....	7
Figura 5.a. Palabras más frecuentes para el género “literature”	8
Figura 5.b. Palabras más frecuentes para el género “fantasy”	8
Figura 5.c. Palabras más frecuentes para el género “mystery”	8
Figura 6. Comparativa distribución géneros.....	9
Figura 7.a. Distribución embeddings BERT.....	10
Figura 7.b. Distribución embeddings Sentence Transformer.....	10
Figura 8. Matriz de confusión para los resultados del mejor modelo propio.....	16
Figura 9.a. Diagrama Upset para las etiquetas reales.....	18
Figura 9.b. Diagrama Upset para las etiquetas predichas del mejor modelo propio.....	18
Figura 10. Clasificador diseñado para los modelos pre-entrenados.....	20
Figura 11.a. Evolución del entrenamiento mediante fine-tuning basado en características.....	22
Figura 11.b. Evolución del entrenamiento mediante fine-tuning parcial.....	22
Figura 11.c. Evolución del entrenamiento mediante fine-tuning total.....	22
Figura 12. Matriz de confusión para los resultados del mejor modelo preentrenado.....	23
Figura 13. Diagrama Upset para las etiquetas predichas del mejor modelo pre-entrenado.....	24

ÍNDICE de Tablas

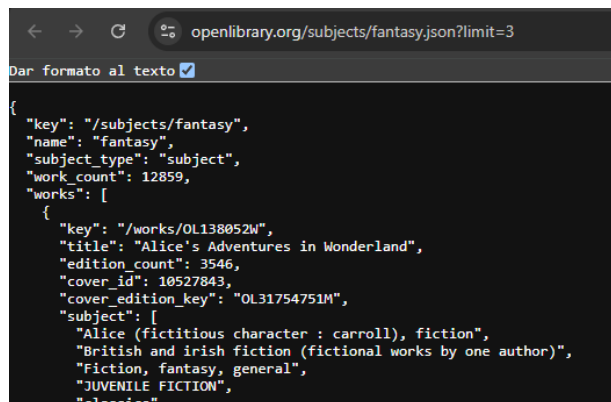
Tabla 1. Resultados del modelo propio para Bag of Words.....	14
Tabla 2. Resultados del modelo propio para TF-IDF.....	14
Tabla 3. Resultados del modelo propio con el uso de embeddings.....	15
Tabla 4. Resumen de los mejores resultados del modelo propio.....	16
Tabla 5. Resumen de los resultados de los modelos pre-entrenados.....	21

1. WebScraping: OpenLibrary API

El primer paso de nuestro proyecto fue la recolección de datos de la página web de reseñas de libros llamada “[OpenLibrary](#)”. Esta nos proporciona una API que nos permitirá descargar conjuntos de descripciones de libros en formato *.json* para agilizar el proceso.

Los pasos realizados durante el web scraping fueron los siguientes:

1. Obtención de los géneros pertenecientes a la categoría “Ficción” de la página de géneros de OpenLibrary. Los 14 géneros obtenidos son: “*Fantasy*”, “*Historical Fiction*”, “*Horror*”, “*Humor*”, “*Literature*”, “*Magic*”, “*Mystery and detective stories*”, “*Plays*”, “*Poetry*”, “*Romance*”, “*Science Fiction*”, “*Short Stories*”, “*Thriller*” y “*Young Adult*”.
2. A partir de los géneros obtenemos el *.json* de la mayor cantidad de libros posibles, alrededor de 200 en nuestro caso. Este paso lo conseguimos haciendo una petición a la API, manipulando parámetros en una petición web.



```
{
  "key": "/subjects/fantasy",
  "name": "fantasy",
  "subject_type": "subject",
  "work_count": 12859,
  "works": [
    {
      "key": "/works/OL138052W",
      "title": "Alice's Adventures in Wonderland",
      "edition_count": 3546,
      "cover_id": 10527843,
      "cover_edition_key": "OL31754751M",
      "subject": [
        "Alice (fictitious character : carroll), fiction",
        "British and irish fiction (fictional works by one author)",
        "Fiction, fantasy, general",
        "JUVENILE FICTION",
        "classics"
      ]
    }
  ]
}
```

Figura 1. [JSON resultado de consulta API](#)

3. Del *.json* obtenemos los campos relevantes para nuestro trabajo como:
 - a. title: nombre de la obra.
 - b. description: consiste en la sinopsis o descripción breve de la obra.
 - c. El género ya lo tenemos, pues hemos realizado la consulta a la API a partir del género. Tuvimos que tener en cuenta que un libro puede pertenecer a más de un género y gran parte del trabajo girará alrededor de ello.

Una vez realizada la recolección inicial de libros tenemos un dataset de 2002 libros que pasaremos a preprocesar.

2. Tratamiento de los datos

En este apartado realizaremos la limpieza del dataset realizando pasos como la eliminación de contenido no relevante, normalización del texto, lematizado y otros. Seguidamente aplicaremos algunas técnicas de extracción de características como BoW, TFIDF, obtención de embeddings, ... Finalmente, acabaremos la sección con un análisis de los datos resultantes.

2. 1. Preprocesado

Para el preprocesado utilizaremos la librería "spacy" con el modelo "en_core_web_lg", pues las descripciones se encuentran en inglés. Además eliminaremos aquellas sinopsis cuyo número de palabras final sea percentil 10 de la distribución del número de estas en los textos . El preprocesado lo aplicaremos a 3 niveles distintos:

1. **texto_limpio:** Eliminación de contenido irrelevante. Secciones adicionales a parte de la propia sinopsis, enlaces a otros recursos, elementos html. Ideal para la creación de embeddings.
2. **preprocesado:** Pasos realizados en *texto_limpio* además de eliminación de *stopwords* y signos de puntuación, conversión a minúsculas y eliminación de elementos no alfanuméricos.
3. **preprocesado_lemma:** Pasos realizados en *preprocesado* además de un lematizado.

Un ejemplo de sinopsis preprocesada es la siguiente:

```
Sinopsis original:
A very real little girl named Alice follows a remarkable rabbit down a rabbit hole and steps through a looking-;
--back cover

Contains:
- [Alice's Adventures in Wonderland](https://openlibrary.org/works/OL8193508W)
- [Through the Looking Glass, and What Alice Found There][2]

[2]: https://openlibrary.org/works/OL15298516W

Sinopsis procesada (no lema):
very real little girl named alice follows remarkable rabbit down rabbit hole steps through looking glass come f.
-----
Sinopsis procesada (lema):
very real little girl name alice follow remarkable rabbit down rabbit hole step through look glass come face fa
-----
Sinopsis procesada (limpia):
A very real little girl named Alice follows a remarkable rabbit down a rabbit hole and steps through a looking ;
back cover
```

Figura 2. Ejemplo sinopsis preprocesada

Tras el preprocesado nos quedamos con una cantidad de 1801 libros, los cuales pueden pertenecer a uno o más géneros. La distribución de género es la siguiente:

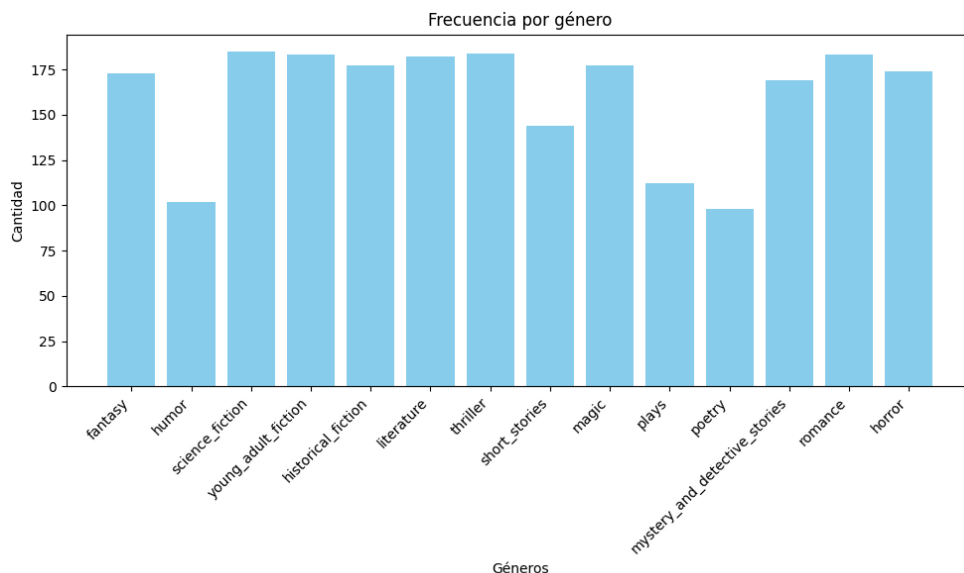


Figura 3. Distribución de géneros preprocesado

2. 2. Extracción de características

En esta sección extraemos BoW, TF-IDF (datos preparados *preprocesado* y *preprocesado_lemma*) y embeddings (del texto limpio pues requerimos de toda la información del contexto posible).

Para BoW y TF-IDF, consideraremos *n-gramas* de hasta 3 elementos y el tamaño del vocabulario ascenderá hasta 10000. Para los embeddings, hemos utilizado los modelos BERT y Sentence-Transformers (all-MiniLM-L6-v2) con un tamaño de embeddings de 768.

También preparamos la etiqueta a predecir. Como veremos más adelante los 14 géneros propuestos no son un buen objetivo de clasificación. Ya que existen géneros muy generales, otros que se resumen dentro de otros, otros que no son debidamente un género en sí ... Esto provocaría dificultades para una debida clasificación. Es por ello que proponemos agrupar las etiquetas para que tenga un resultado más coherente respecto a la realidad. Agrupando géneros similares como lo son la "Magia" y la "Fantasía" y eliminando otros los cuáles no tiene sentido comparar como "Historias Cortas". La propuesta de los nuevos géneros no implica que el problema deje de ser multietiqueta. Tras este ajuste nos quedaremos con la cantidad de 1240 libros.

Es decir, evaluaremos dos enfoques:

1. Preservar los 14 géneros originales.
2. Realizamos una agrupación inteligente de géneros.

- Eliminando los géneros: “*humor*”, “*romance*”, “*historical_fiction*”, “*young_adult_fiction*”, “*short_stories*”. Eliminaremos del dataset aquellos libros que pertenezcan exclusivamente a estos géneros.
- Agrupando “*Magic*” y “*Fantasy*” en “*Fantasy*”
- Agrupando “*Poetry*”, “*Literature*” y “*Plays*” en “*Literature*”.
- Agrupando “*Thriller*”, “*Horror*” y “*mystery_and_detective_stories*” en “*mystery*”
- Resumiendo nos quedamos con los siguientes 4 géneros: “*Fantasy*”, “*Literature*”, “*Mystery*” y “*Science Fiction*” (el cuál hemos conservado).

2. 3. Análisis EDA

Para el análisis exploratorio de los datos vamos a evaluar la distribución de las palabras en los distintos géneros a parte de la distribución de los propios géneros. Para un análisis conciso compararemos los géneros presentes en ambos enfoques (el propuesto en la sección anterior y el original). Comenzando inicialmente con la distribución del número de palabras por géneros. Vemos en la figura que las densidades alrededor de las 50 palabras han aumentado en el nuevo enfoque. Hecho que tiene sentido pues hemos aumentado la cantidad de ejemplos de esos géneros. Por lo general vemos que las formas se han mantenido y entre los propios géneros no nos servirá para distinguir entre ellos.

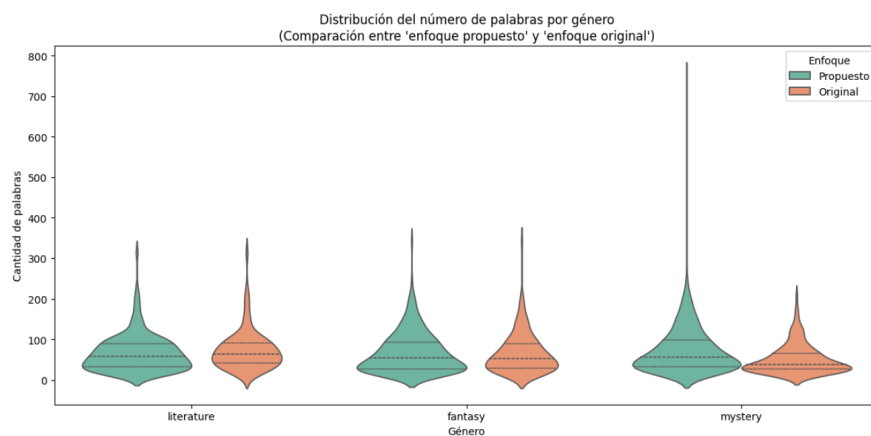
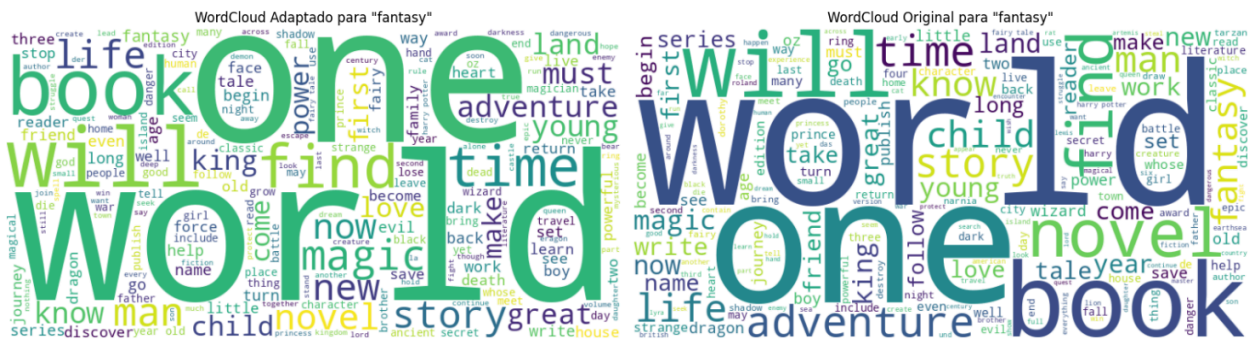
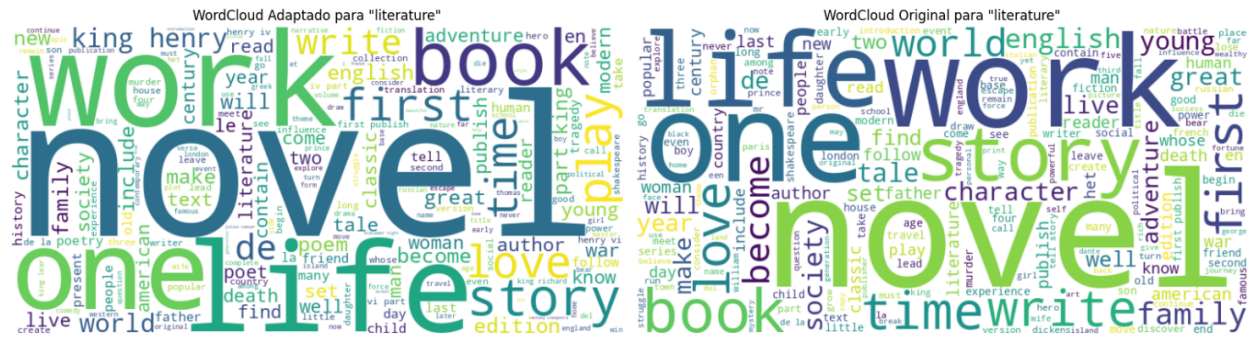


Figura 4. Distribución del número de palabras por género

A continuación observaremos las palabras más frecuentes por género mediante *wordclouds*. Teniendo a la izquierda nuestro enfoque de géneros propuesto y a la derecha el enfoque original.



Las conclusiones que extraemos de los *wordclouds* son:

- En general, las palabras más repetidas están presentes en todos los géneros, por lo que no permiten diferenciarlos. Esto sugiere que el uso de TF-IDF será más práctico que BoW."
- Además las palabras, excepto las excepciones como *murder* y *mystery* en "*mystery*", son bastante generales. Viendo que la perspectiva del uso de las palabras aisladas y su frecuencia no se encontrará favorecida en la clasificación respecto al uso del contexto mediante embeddings.

- Analizando las diferencias entre los géneros originales y los propuestos, observamos ligeros cambios en palabras poco frecuentes, lo que indica que la combinación de géneros es adecuada. Pareciera que la cantidad de libros por géneros originales estaba relativamente balanceada y no ha supuesto grandes cambios en su repertorio de palabras.

Continuaremos con el análisis de las distribuciones de los géneros para las dos perspectivas.

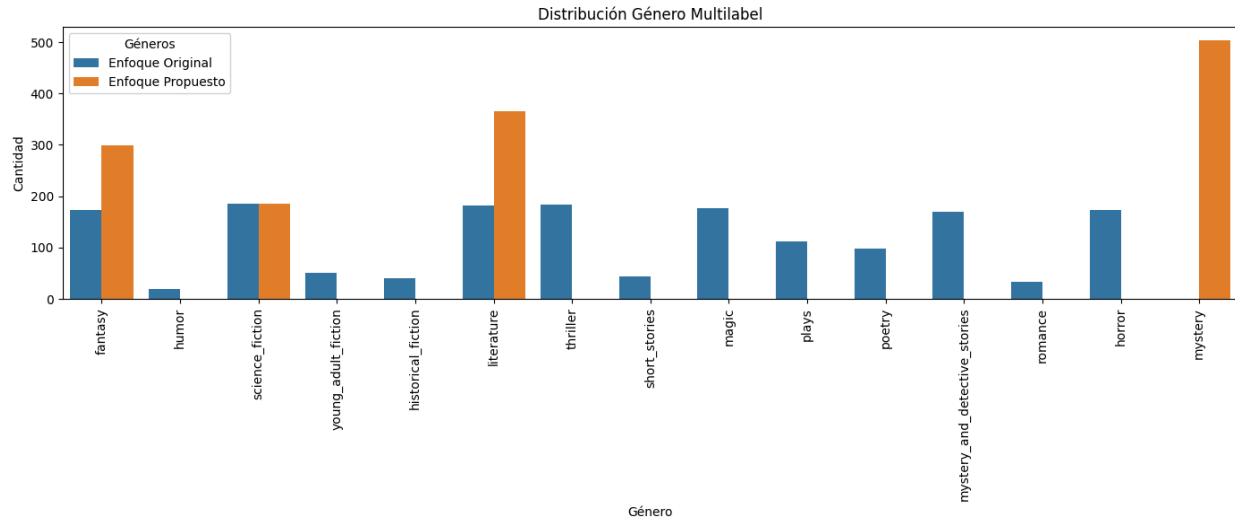


Figura 6. Comparativa distribución géneros

Dado que se trata de un problema multietiqueta, un libro puede pertenecer a varios géneros, por lo que el gráfico muestra la frecuencia de aparición de cada etiqueta.

Como se observa en la Figura 6, tras la reducción y reagrupación de géneros, la distribución de etiquetas cambió notablemente. Géneros como *Mystery* y *Literature*, que agrupan varias categorías originales, concentran ahora más asignaciones. En contraste, *Science Fiction* y *Fantasy*, que agrupan menos géneros, presentan cantidades menores o iguales. Esto genera un cierto desbalance en las nuevas categorías.

Por último evaluaremos la calidad de los contextos para cada género. Para ello aplicaremos la técnica de reducción de dimensionalidad *UMAP* y realizaremos una representación 2D de las

distribuciones de los géneros para ambos embeddings.

Distribuciones géneros embeddings BERT

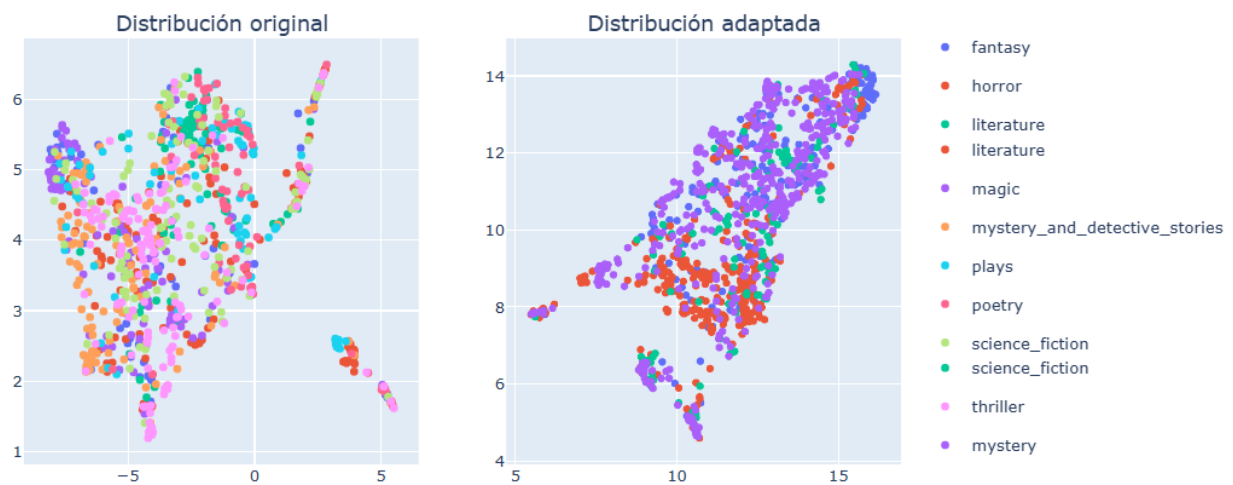


Figura 7.a. Distribución embeddings *BERT*

Distribuciones géneros embeddings Sentence Transformer

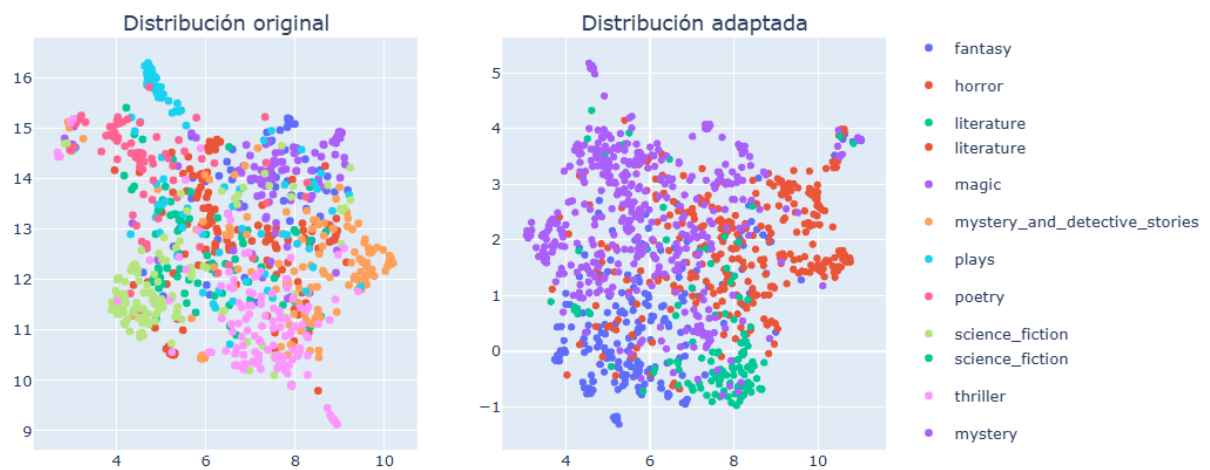


Figura 7.b. Distribución embeddings *Sentence Transformer*

Tras graficar los géneros podemos ver que tanto para las distribuciones originales como para las propuestas los embeddings obtenidos mediante el *Sentence Transformer* son más discriminables. Podemos identificar mejor los grupos, apreciándose especialmente los cuatro clusters correspondientes a los cuatro géneros adaptados con el *Sentence Transformer*, lo que podría sugerir que un modelo de clasificación utilizando este embedding podría funcionar mejor.

Tras el análisis exploratorio podemos ver que la clasificación por géneros no va a ser una tarea sencilla a partir de las descripciones de los libros proporcionadas.

3. Modelos

A continuación, vamos a construir modelos de clasificación para categorizar los libros en los cuatro géneros: *Fantasy*, *Mystery*, *Literature* y *Science Fiction*. Como ya se ha comentado anteriormente, el género *Fantasy* engloba obras de fantasía y magia, *Mystery* obras de misterio pero también thrillers y obras de terror mientras que *Literature* incluye obras clásicas de la literatura junto a poesía y obras de teatro.

En primer lugar, vamos a construir un modelo propio de clasificación de género, empleando funciones disponibles en la librería *scikit-learn*, entrenándolo a partir de las características extraídas anteriormente mencionadas: *Bag of Words*, *TF-IDF* y embeddings (*Bert* y *Sentence embeddings*).

En segundo lugar, emplearemos un modelo pre-entrenado de la librería *Transformers* de HuggingFace y realizaremos fine-tuning de tres formas distintas para adaptarlo a nuestro problema de clasificación.

Debido a que un mismo libro puede pertenecer a distintos géneros, el enfoque de los clasificadores será multietiqueta. Finalmente, compararemos los resultados y el rendimiento de ambos modelos, analizando los fallos de estos y posibles mejoras.

3. 1. Modelo propio de clasificación de género

Antes de comenzar con la construcción del modelo propio, extraemos las etiquetas de clase del conjunto de datos y las binarizamos. Este será nuestro conjunto y . El conjunto X serán las descripciones de los libros a las que se les ha aplicado la correspondiente extracción de características:

1. *Bag of Words* con lematizado y sin lematizado.
2. *TF-IDF* con lematizado y sin lematizado.

3. Embedding del modelo '*bert-case-uncased*' sobre el texto limpio (pero sin preprocesar).
4. Embedding del modelo '*all-MiniLM-L6-v2*' (*Sentence embedding*) sobre el texto limpio (pero sin preprocesar).

De esta manera comparemos los resultados obtenidos para cada caso, observando que extracción de características resulta la más efectiva para este problema.

Para la creación de un modelo propio de clasificación de género crearemos la función *multilabel_evaluation*, cuyos parámetros de entrada son los conjuntos de test y train.

La salida de la función devuelve el nombre del clasificador empleado, la predicción de etiquetas y las métricas de *accuracy* y *f1 score (weighted)* junto a un reporte de clasificación para cada clase, donde podremos observar también la precisión y el *recall*. Como se trata de un problema de clasificación multietiqueta, en este caso la métrica en la que nos centraremos será *f1 score* y no la *accuracy*. Esto se debe a que en este tipo de problemas, el valor de *accuracy* puede ser poco representativo ya que sólo considerará la clasificación correcta si se han predicho bien todas las etiquetas reales. Por ello empleamos el *f1 score* ponderado (que tiene en cuenta la frecuencia de las etiquetas), que combina la precisión y el *recall*. La precisión es la proporción de predicciones positivas que han sido correctas, es decir, de los géneros que ha devuelto el modelo como etiquetas, en cuáles ha acertado. El *recall* es qué proporción de verdaderos positivos ha detectado el modelo, es decir, de los géneros reales que estaban presentes cuántos ha detectado correctamente.

La función *multilabel_evaluation* entrena y evalúa los resultados para tres clasificadores y dos estrategias para clasificación multietiqueta. Los clasificadores usados son:

- XGBoost con 100 estimadores
- Regresión Logística
- SGDClassifier

Las estrategias multietiqueta son:

- MultiOutput: entrena un clasificador independiente para cada etiqueta, prediciendo si está presente o no, sin tener en cuenta las demás.

- OneVsRest: entrena un clasificador binario donde la clase positiva es la etiqueta a predecir y la negativa el resto de etiquetas.

Durante la evaluación obtenemos las probabilidades de pertenencia a una clase y se ajusta un umbral óptimo de decisión para cada etiqueta para mejorar el rendimiento usando la curva de precisión-recall. Esto ayuda a mejorar los resultados con respecto a un modelo con umbral por defecto (que suele ser 0.5) y que tiende a ser “conservador”, es decir, no asigna un género salvo que esté muy seguro de ello. Además, esto ayudará también con el hecho de tener clases algo desbalanceadas.

3.1.1 Resultados del modelo propio

En la siguiente tabla (Tabla 1) se muestran los resultados empleando *Bag of Words*, con lematizado y sin lematizado, con las diferentes estrategias y clasificadores. El modelo con lematizado es claramente superior, aunque lo que más parece influir es el clasificador empleado, siendo regresión logística el que obtiene mejores resultados, alcanzando un f1 score de 0.744 para ambas estrategias multietiqueta.

Feature Extraction	Strategy	Classifier	F1 Score (weighted)	Accuracy
BOW_lemma	MultiOutput	LogisticRegression	0.744	0.617
BOW_lemma	OneVsRest	LogisticRegression	0.744	0.617
BOW_no_lemma	MultiOutput	LogisticRegression	0.722	0.528
BOW_no_lemma	OneVsRest	LogisticRegression	0.722	0.528
BOW_lemma	OneVsRest	XGBoost	0.709	0.540
BOW_lemma	MultiOutput	SGDClassifier	0.709	0.565
BOW_lemma	MultiOutput	XGBoost	0.709	0.540
BOW_no_lemma	MultiOutput	XGBoost	0.704	0.565
BOW_no_lemma	OneVsRest	XGBoost	0.704	0.565

BOW_lemma	OneVsRest	SGDClassifier	0.699	0.552
BOW_no_lemma	MultiOutput	SGDClassifier	0.699	0.548
BOW_no_lemma	OneVsRest	SGDClassifier	0.692	0.532

Tabla 1. Resultados del modelo propio para *Bag of Words*.

A continuación, empleamos *TF-IDF* y observamos los 8 mejores resultados obtenidos en este caso en la Tabla 2. De nuevo, regresión logística parece ser el clasificador que mejor funciona y lematizar los textos supone una mejora sobre no lematizar. En este caso, hemos alcanzado f1 scores más altos que con *BoW*, llegando hasta 0.768. Esto se debe a que *TF-IDF* tiene en cuenta la frecuencia de aparición de las palabras más y menos frecuentes a la hora de hacer la ponderación, lo que mejora los resultados ya que las palabras más frecuentes se repiten en todos los géneros, como veíamos anteriormente en las nubes de palabras.

Feature Extraction	Strategy	Classifier	F1 Score (weighted)	Accuracy
TFIDF_lemma	OneVsRest	LogisticRegression	0.768	0.657
TFIDF_lemma	MultiOutput	LogisticRegression	0.768	0.657
TFIDF_no_lemma	MultiOutput	LogisticRegression	0.763	0.629
TFIDF_no_lemma	OneVsRest	LogisticRegression	0.763	0.629
TFIDF_lemma	OneVsRest	SGDClassifier	0.761	0.629
TFIDF_lemma	MultiOutput	SGDClassifier	0.760	0.625
TFIDF_no_lemma	MultiOutput	SGDClassifier	0.754	0.629
TFIDF_no_lemma	OneVsRest	SGDClassifier	0.754	0.625

Tabla 2. Resultados del modelo propio para *TF-IDF*.

Finalmente vamos a ver el resultado empleando embeddings. Los embeddings, al contrario que las dos técnicas anteriores, tienen en cuenta el sentido semántico de las palabras, lo que esperamos que produzca una mejora de los resultados.

Feature Extraction	Strategy	Classifier	F1 Score (weighted)	Accuracy
Embeddings_sentence_transformers	OneVsRest	LogisticRegression	0.777	0.653
Embeddings_sentence_transformers	MultiOutput	LogisticRegression	0.777	0.653
Embeddings_sentence_transformers	MultiOutput	XGBoost	0.739	0.605
Embeddings_sentence_transformers	OneVsRest	XGBoost	0.739	0.605
Embeddings_sentence_transformers	OneVsRest	SGDClassifier	0.739	0.593
Embeddings_BERT	OneVsRest	LogisticRegression	0.734	0.577
Embeddings_BERT	MultiOutput	LogisticRegression	0.734	0.577
Embeddings_BERT	MultiOutput	XGBoost	0.731	0.585
Embeddings_sentence_transformers	MultiOutput	SGDClassifier	0.731	0.597
Embeddings_BERT	OneVsRest	XGBoost	0.731	0.585
Embeddings_BERT	MultiOutput	SGDClassifier	0.721	0.556
Embeddings_BERT	OneVsRest	SGDClassifier	0.713	0.569

Tabla 3. Resultados del modelo propio con el uso de embeddings.

Como se puede observar en la Tabla 3, empleando *Sentence embeddings* conseguimos llegar hasta un f1 score de 0.777, empleando regresión logística. Además, *Sentence embeddings* tiene claramente un mejor rendimiento que *BERT*, lo que sugiere que para una tarea de clasificación como esta, con textos relativamente cortos, extraer representaciones a nivel de texto completo resulta más eficaz.

3.1.2 Análisis de los resultados del mejor modelo propio

Los mejores resultados los hemos obtenido para *Sentence Embeddings* y para *TF-IDF* con lematizado y clasificador regresión logística. La diferencia entre estrategias multietiqueta es prácticamente insignificante.

Feature Extraction	Strategy	Classifier	F1 Score	Accuracy
Embeddings_sentence_transformers	OneVsRest	LogisticRegression	0.7766	0.6532
Embeddings_sentence_transformers	MultiOutput	LogisticRegression	0.7766	0.6532
TFIDF_lemma	OneVsRest	LogisticRegression	0.7684	0.6573
TFIDF_lemma	MultiOutput	LogisticRegression	0.7684	0.6573

Tabla 4. Resumen de los mejores resultados del modelo propio.

Vamos a analizar más a fondo los resultados obtenidos para el mejor modelo propio que hemos conseguido.

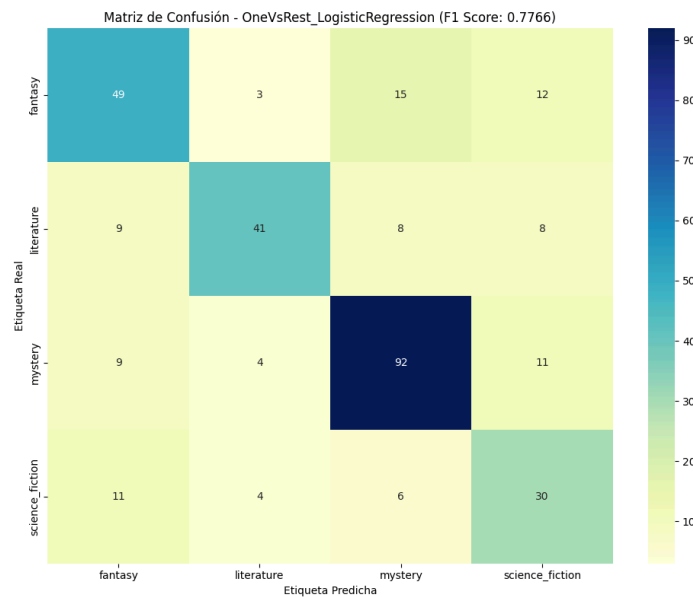


Figura 8. Matriz de confusión para los resultados del mejor modelo propio.

En la matriz de confusión de la figura 8 podemos ver que, aunque en general clasifica razonablemente bien, comete algunos errores, destacando especialmente los errores cometidos entre las categorías fantasía, misterio y ciencia ficción. Esto tiene sentido ya que estas tres categorías son las más parecidas entre sí en comparación con literatura, lo que provoca que el modelo cometa más errores.

La mayor cantidad de errores los comete al asignar misterio a novelas de fantasía. Esto, además de ser géneros más cercanos entre sí que por ejemplo misterio y literatura, puede provenir de la naturaleza de los textos empleados para entrenar en sí mismos. No se trata de resúmenes completos de las obras, sino sinopsis, que intentan convencer al lector de leer el libro sin desvelar todos los secretos de la trama. Esto puede llevar al modelo a interpretar la obra como una novela de misterio. Por ejemplo, en las novelas de *Harry Potter*, con etiqueta real fantasía y que nuestro modelo ha predicho como misterio en la mayoría de ellas, las descripciones contienen frases como: *“For Harry Potter, it’s the start of another far from ordinary year at Hogwarts when the Knight Bus crashes through the darkness and comes to an abrupt halt in front of him. It turns out that Sirius Black, mass murderer and follower of Lord Voldemort, has escaped.”* ó *“In his sixth year, the names Black, Malfoy, Lestrage and Snape will haunt Harry with shades of trust and treachery as he discovers the secret behind the mysterious Half-Blood Prince.”* Sin el contexto de conocer *Harry Potter*, parece claro porque el modelo piensa que son novelas de misterio, thriller o terror (agrupadas todas ellas en misterio).

Para observar mejor las predicciones multietiqueta y su comparación con las reales empleamos diagramas *UpSet*, que podemos ver en las figuras 9.a y 9.b.

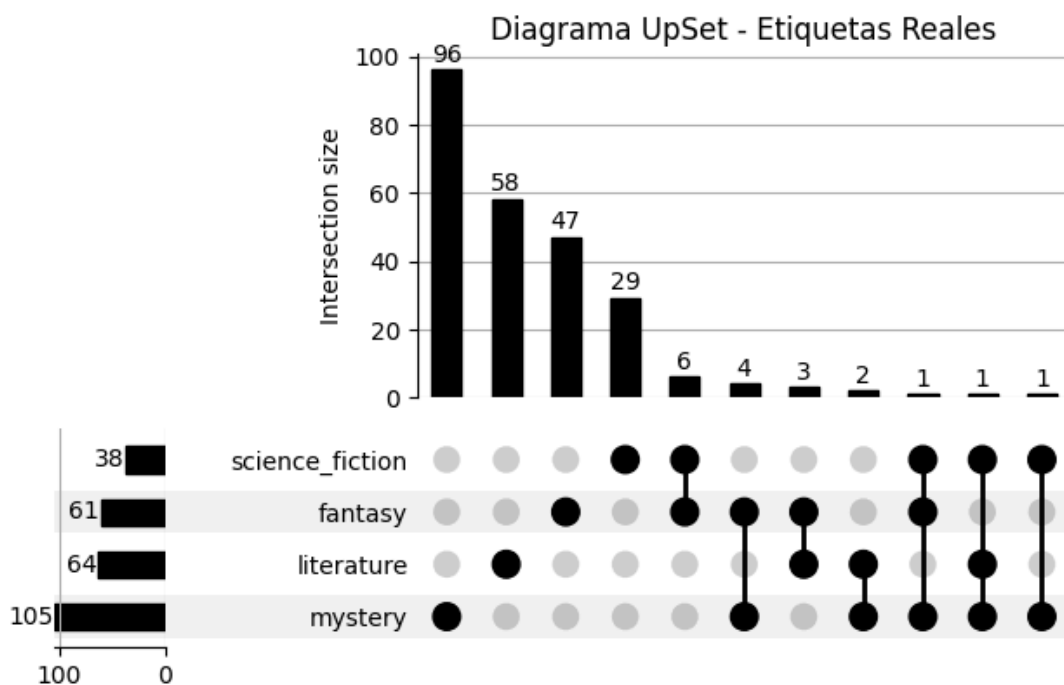


Figura 9.a. Diagrama Upset para las etiquetas reales.

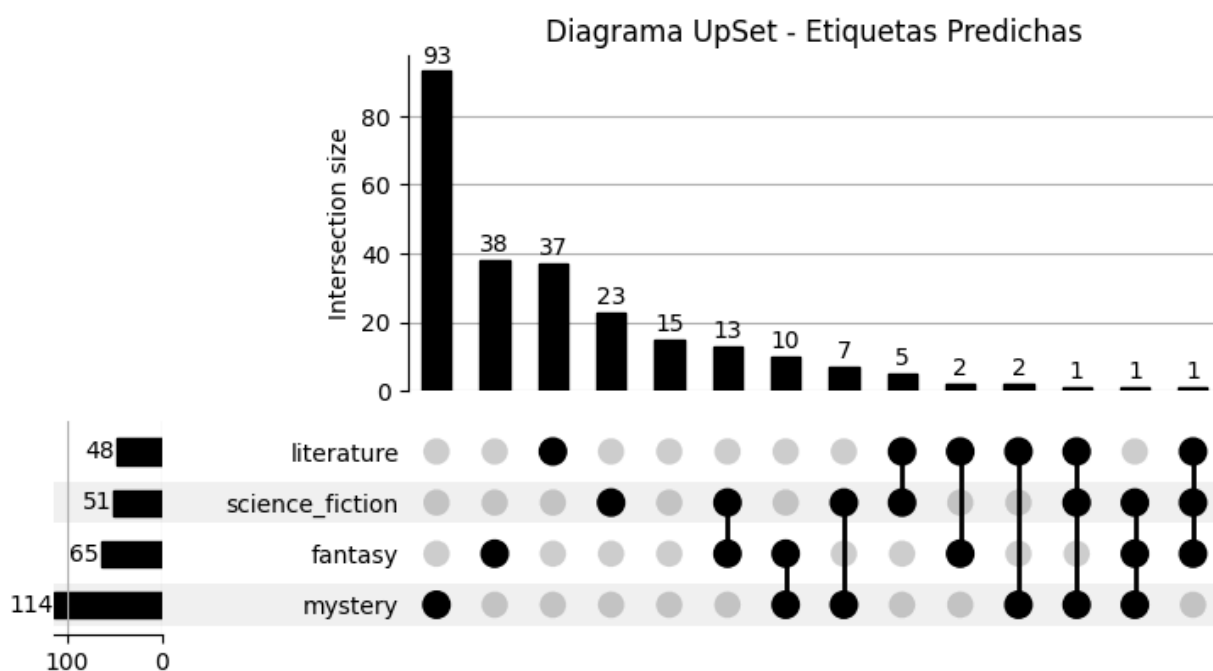


Figura 9.b. Diagrama Upset para las etiquetas predichas del mejor modelo propio.

Como primera observación, vemos que el modelo ha predicho *literature* como única etiqueta de manera notablemente menor que en las etiquetas reales. Observando las descripciones de los libros, esto tiene una respuesta clara: aunque originalmente la obra sea una obra clásica, poesía o teatro, esta tiene una trama que el modelo interpreta también como parte de alguno de los otros géneros. Por ejemplo, una obra de teatro de Shakespeare como *Hamlet*, clasificada originalmente únicamente como literatura, tiene elementos fantásticos que llevan al modelo a añadir también el género fantasía.

En los diagramas podemos ver claramente el desbalance entre clases, siendo misterio el género con más elementos y ciencia ficción el que menos. Aunque se ha especificado en la regresión logística el parámetro *class_weights* como *balanced*, el problema no parece solucionarse del todo y vemos que el modelo ha tendido a subestimar la clase menos numerosa real (ciencia ficción).

Por otro lado, aunque hemos afrontado el problema como un problema multietiqueta, realmente el número de géneros múltiples asignados originalmente es muy bajo en comparación con géneros únicos y el modelo tiende a sobreestimar el número de elementos con múltiples etiquetas. También cabe destacar que hay 15 obras del conjunto de test que el modelo no ha clasificado en ningún género.

3. 2. Modelo pre-entrenado de análisis de género

De la misma forma que con los modelos propios, se han de emplear etiquetas binarizadas. El conjunto de datos de entrada X serán las descripciones de los libros limpias, pero sin procesar ni lematizar, ya que los transformers empleados están entrenados para trabajar con frases en crudo, sin preprocesar. Para que los resultados sean comparables entre los modelos pre-entrenados y los propios, también se emplea de nuevo la curva precision-recall para ajustar el umbral óptimo de decisión para cada etiqueta.

Los modelos preentrenados empleados son transformers de la librería *Transformers* de HuggingFace. En un análisis preliminar se probaron dos modelos encoder: *distilroberta-base* y *distilbert-base-cased*. Al obtener peores resultados con RoBERTa, se descartó seguir entrenando dicho modelo. Cabe destacar que RoBERTa es una versión mejorada de BERT, con más

parámetros (distilRoBERTa tiene unos ~82 millones de parámetros frente a los ~66 millones de distilBERT) y fue entrenado sobre un corpus mayor, por lo que un rendimiento inferior no era lo esperable. Esto podría deberse a que, al contar con mayor capacidad, no se logró una elección adecuada de hiperparámetros para su entrenamiento o que el corpus empleado no es lo suficientemente grande como para entrenar un modelo tan complejo.

El modelo *'distilbert-base-cased'* empleado es una versión destilada de BERT, entrenada sobre *BookCorpus* (formado por más de 11.000 libros) y English Wikipedia (todo el contenido en inglés de la wikipedia). A nivel de datos de entrenamiento, parece razonablemente alineado con nuestro problema de clasificación. Aunque originalmente se diseñó como modelo para predicción de palabras enmascaradas, puede adaptarse a tareas de clasificación mediante fine-tuning. También es importante señalar que se ha elegido la variante *cased* que distingue entre mayúsculas y minúsculas.

Al cargar el modelo pre-entrenado para un problema de clasificación con AutoModel, el clasificador añadido tras el transformer es una red neuronal con únicamente una capa densa de entrada y la capa de salida, y se decide añadir una capa oculta adicional con 128 neuronas para introducir no linealidad al modelo. En la figura 10 se muestra la estructura del clasificador empleado, diseñado con capas de BatchNormalization, GELU (similar a ReLU pero con suavizado para los valores negativos) y Dropout.

```
(pre_classifier): Linear(in_features=768, out_features=768, bias=True)
(classifier): Sequential(
  (0): BatchNorm1d(768, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (1): GELU(approximate='none')
  (2): Dropout(p=0.5, inplace=False)
  (3): Linear(in_features=768, out_features=128, bias=True)
  (4): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (5): GELU(approximate='none')
  (6): Dropout(p=0.5, inplace=False)
  (7): Linear(in_features=128, out_features=4, bias=True)
)
(dropout): Dropout(p=0.2, inplace=False)
```

Figura 10. Clasificador diseñado para los modelos pre-entrenados.

3.2.1 Resultados del modelo pre-entrenado

Para entrenar *distilbert-base-cased* con este clasificador personalizado, y emplearlo para resolver nuestro problema de clasificación, se realizaron 3 variantes de fine-tuning:

- Fine-tuning basado en características: consiste en congelar todos los parámetros del modelo transformer menos los del clasificador, para entrenar solo el clasificador.
- Fine-tuning parcial (I): consiste en congelar los embeddings y las 4 primeras capas del transformer. Al ser un modelo con 6 capas, se entrenarán las dos últimas y el clasificador.
- Fine-tuning total (II): consiste en reentrenar el transformer completo, incluyendo los embeddings, se trata de la alternativa más compleja y costosa.

Cada una de las variantes requirió ajustar parámetros como el número de épocas, el tamaño del batch, la tasa de aprendizaje y su estrategia de disminución, así como el ratio de calentamiento y el coeficiente de regularización L2. En todos los casos se aplicó un *callback* de early stopping, con un máximo de 3 épocas sin mejora. En la Tabla 5 se recoge, para cada uno de los tres tipos de fine-tuning, el número de parámetros entrenables, los valores obtenidos de F1-score y accuracy, y el número de épocas en que se detuvo el entrenamiento.

Fine-tuning	Parametros	F1-Score	Accuracy	Epocas
Características	691.332	0.797	0.710	20 (máxima)
Parcial	14,867,076	0.833	0.774	15
Total	65,882,244	0.816	0.710	24

Tabla 5. Resumen de los resultados de los modelos pre-entrenados.

Atendiendo al estadístico F1-Score, vemos que las tres variantes superan los resultados del mejor modelo propio. El mejor resultado se obtiene con el fine-tuning parcial, a pesar de que esperaríamos que el mejor resultado se obtuviese al re-entrenar el modelo completo. Esto puede deberse a que entrenar todo el modelo, supone una mayor dificultad durante la elección de hiperparámetros, y no se han podido obtener mejores resultados. Además de presentar mejores resultados, el fine-tuning parcial es un buen término medio en cuanto a tiempo de entrenamiento y de complejidad del ajuste.

La evolución del coste de entrenamiento y validación a lo largo del entrenamiento para los diferentes tipos de fine-tuning se muestra en las figuras 11 a, b y c. En ellas podemos apreciar que los modelos presentan cierto sobreajuste, siendo este más evidente conforme entrenamos mayor parte del transformer.

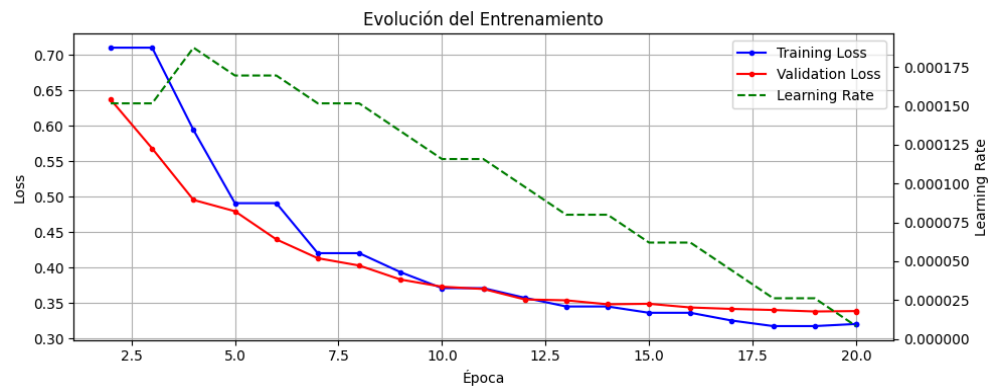


Figura 11.a. Evolución del entrenamiento mediante fine-tuning basado en características.

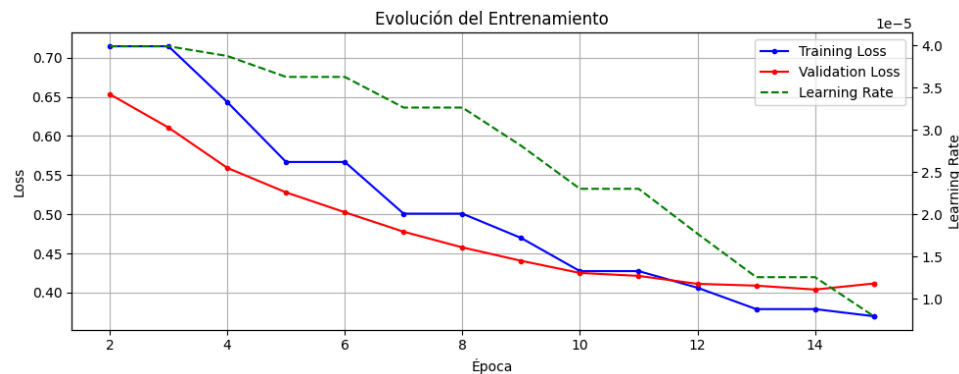


Figura 11.b. Evolución del entrenamiento mediante fine-tuning parcial.

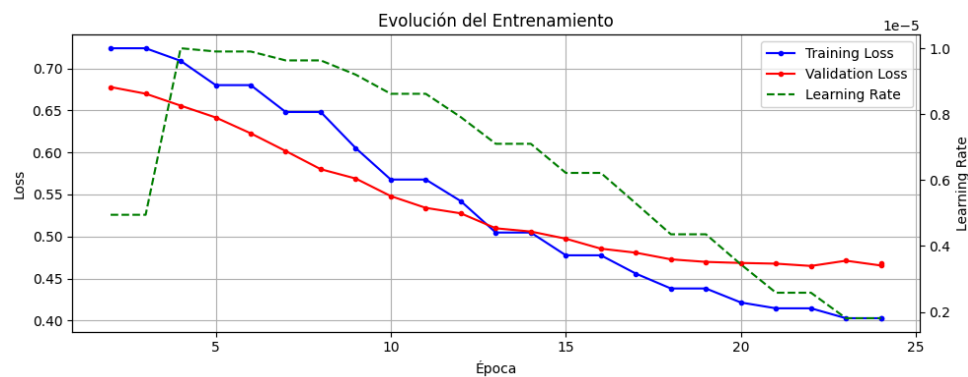


Figura 11.c. Evolución del entrenamiento mediante fine-tuning total.

Al tratarse de arquitecturas muy complejas con un número elevado de parámetros, tienden a sobre ajustar si el corpus no es lo suficientemente amplio. Para evitar esto, lo más conveniente sería ampliar el conjunto de datos. Si esto no es posible, otras opciones que podrían ser interesantes son aumentar la regularización, emplear otros tipos de regularización diferentes a L2 o ajustar la arquitectura interna del transformer, por ejemplo, incrementando el Dropout.

3.2.2 Análisis de los resultados del mejor modelo pre-entrenado

Los mejores resultados los hemos obtenido al reentrenar el modelo parcialmente, entrenando únicamente el clasificador y dos capas del transformer. Vamos a analizar con más detalle los resultados obtenidos para el fine-tuning parcial, para valorar si existen diferencias notables con los resultados del mejor modelo propio.

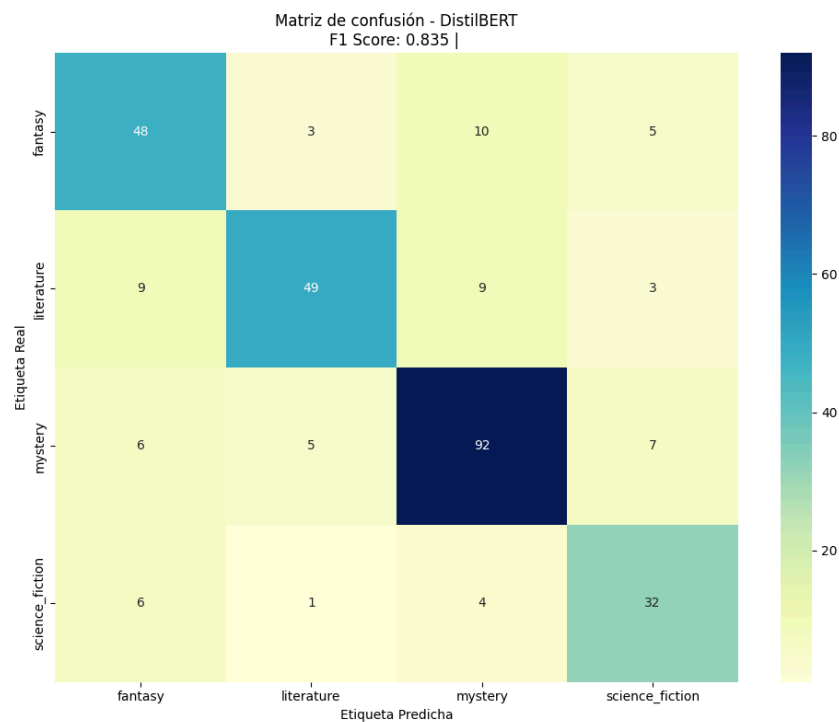


Figura 12. Matriz de confusión para los resultados del mejor modelo preentrenado.

En la matriz de confusión de la figura 12 podemos ver que, el modelo clasifica generalmente bien, mejor que el mejor modelo propio (figura 8). Al igual que en modelo propio, la mayoría de de errores se cometen al asignar la categoría misterio a novelas de fantasía, pero en este caso el número de errores es menor (10 en lugar de 15). La mayoría de fallos se dan al predecir las

categorías fantasía y misterio, sin embargo, la categoría ciencia ficción ya no genera tantos errores como en el caso del modelo propio.

Para observar las predicciones multietiqueta empleamos nuevamente un diagrama *UpSet*, que podemos ver en la figura 13.

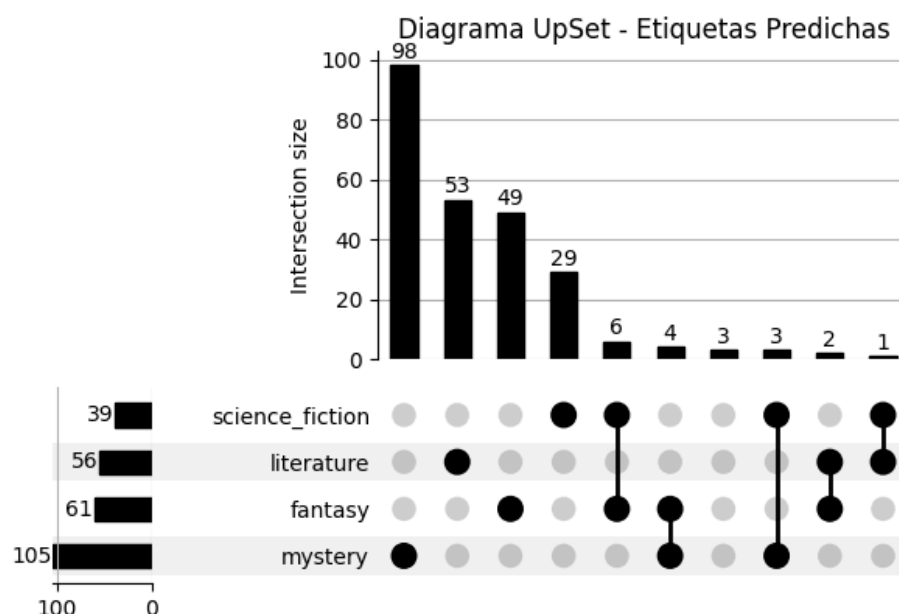


Figura 13. Diagrama Upset para las etiquetas predichas del modelo pre-entrenado.

Las etiquetas predichas coinciden en gran medida con las reales, y se aproximan bastante más que en el modelo propio. Aun así, parece seguir habiendo cierta dificultad al distinguir entre *literatura* y *fantasía*. El modelo ha predicho en solo 3 ocasiones que la descripción no corresponde con ningún género, una mejora notable frente a las 15 veces que ocurría con el modelo propio. En cuanto al desbalance entre clases, este sigue presente, aunque no parece provocar una sobrestimación ni subestimación de ninguna etiqueta concreta (respecto a sus frecuencias reales).

En resumen, los modelos pre-entrenados han ofrecido mejores resultados que el modelo propio, siendo el fine-tuning parcial el que presenta el mejor resultado y un buen equilibrio en términos de tiempo de entrenamiento y complejidad. El riesgo de sobreajuste parece el factor más

limitante, por lo que sería recomendable ampliar el corpus o realizar una mejor selección de hiperparámetros para mejorar la generalización.

4. Conclusión, comparativa de modelos y posibles mejoras.

Mediante el uso de web scraping a partir de la página [OpenLibrary](#), hemos obtenido datos correspondientes a los títulos, descripciones y géneros de alrededor de 2000 obras, con el propósito de aplicar técnicas de procesamiento de lenguaje natural para construir un clasificador de género multietiqueta. La primera dificultad a la que nos enfrentamos es el agrupamiento de los 14 géneros presentes en la web y su reducción a 4 para la obtención de mejores resultados: fantasía (que agrupa magia y fantasía), ciencia ficción, misterio (misterio, thriller y horror) y literatura (obras clásicas, poesía y teatro). En primer lugar, se ha realizado el preprocesado de los datos empleando procedimientos habituales como el paso a minúsculas, eliminación de contracciones en inglés, etc. A través del análisis exploratorio observamos los datos y encontramos un desbalance entre clases, lo que vemos que tiene consecuencias en el rendimiento de los modelos.

El objetivo del proyecto es emplear dos modelos, uno propio y otro pre entrenado, comparando los resultados entre ambos. Para el modelo propio, necesitamos realizar una extracción de características previas, de las cuáles se ha elegido *Bag of Words*, *TF-IDF* y los embeddings (*BERT* y *Sentence Transformer*). Construido el modelo y comparando las distintas características, los clasificadores y dos estrategias multietiqueta encontramos que el mejor modelo, con un f1 score de 0.78 es el entrenado a partir del embedding *Sentence Transformer* con el clasificador regresión logística. Esto se debe a que, al contrario que *BoW* y *TF-IDF*, los embeddings tienen en cuenta las relaciones semánticas entre palabras. Por otro lado, observamos también que *TF-IDF* (f1 score: 0.77) tiene un mejor rendimiento que *BoW* (f1 score: 0.74) lo que es esperable también ya que este método tiene en cuenta las frecuencias de las palabras al realizar la ponderación. La lematización de los textos ha ayudado a la clasificación para *BoW* y *TF-IDF*, mientras que para los embeddings se emplea el texto limpio pero sin preprocesar para un mejor resultado.

Analizando los resultados obtenidos, vemos que el modelo propio comete una serie de errores de clasificación. Estos incluyen:

- Errores entre los tres géneros más parecidos (misterio, magia y fantasía) debido probablemente a su cercanía frente a literatura.
- Sobreestimación de la clase ciencia ficción, proveniente probablemente del desbalance entre clases.
- Asignación de misterio en lugar de fantasía, donde una posible explicación es la naturaleza de los textos. Estos consisten en sinopsis cortas que intentan no revelar la trama completa para lograr intrigar al lector, lo puede provocar una confusión al modelo, pensando que son obras de misterio.
- Desbalance en el número de libros con multietiquetas. Las etiquetas originales son primordialmente de un único género con pocos ejemplos de multietiqueta y el modelo tiende a sobreestimar el número de obras multietiqueta.
- Literatura es asignada como clase única de manera notablemente menor (58 reales frente a 37 predichos). Este problema puede provenir del hecho que las obras clásicas, de poesía o teatro no son géneros en sí mismos y su temática puede caer también en alguno de los otros géneros, lo que no está bien reflejado en las etiquetas reales.

Por otro lado, para el modelo pre-entrenado *distilbert*, se han aplicado tres tipos de fine-tuning para adaptarlo a nuestro problema (basado en características, parcial y total) obteniendo f1 scores de 0.797, 0.833 y 0.816 respectivamente. Los tres casos superan al mejor modelo propio (f1 score: 0.777), lo que se debe a que el modelo pre-entrenado usa transformers entrenados en grandes corpus de palabras, con una arquitectura más compleja que puede capturar mejor el contexto y la semántica, aprendiendo representaciones específicas para nuestra tarea en concreto a través del fine-tuning. Además, algunos de los errores que observábamos con el modelo propio se han solucionado o mitigado notablemente. Sin embargo, emplear modelos pre-entrenados presenta algunas desventajas: mayor coste computacional, más complejidad durante el ajuste de hiperparámetros y un elevado riesgo de sobreajuste si no se dispone de suficientes datos. Esto hace que el modelo propio siga siendo una opción interesante, ya que es más simple y rápido de entrenar aunque el rendimiento obtenido sea ligeramente menor.

Existen muchas posibles mejoras del trabajo entre las cuáles incluimos:

- Aumentar el conjunto de datos descargando más libros.
- Ser más selectivos con las descripciones válidas, eliminando aquellas que sean muy cortas o bien encontrando descripciones más largas realizando web scraping en una página web alternativa. Tener también en cuenta características como el número de adjetivos, adverbios...
- Emplear libros completos en lugar de las sinopsis para la clasificación, lo que aportará mucha más información y resolverá algunos de los problemas planteados como la tendencia hacia misterio debido a sinopsis que pretenden intrigar al lector. Para este enfoque, sin embargo, debemos restringirnos a libros de dominio público, en webs como gutenberg.org.

En resumen, hemos construido dos modelos, uno propio y otro pre entrenado para la clasificación multi etiqueta de géneros de libros, obteniendo f1 scores de 0.78 y 0.82 respectivamente, con un mejor resultado para el modelo pre entrenado con fine tuning aunque a costa de un mayor tiempo de ejecución y labor de ajuste de parámetros. Cabe destacar que el problema de clasificación de géneros de libros no es cerrado y existe una cierta ambigüedad. La clasificación se ha llevado a cabo según los géneros presentes en OpenLibrary pero estos no son universales y por tanto las etiquetas reales no tienen porqué ser las correctas obligatoriamente si no que habría un espacio para la discusión. Por ejemplo, la inclusión de *Harry Potter* como literatura o la no clasificación de obras de teatro en fantasía pese a que su trama incluye elementos fantásticos. Esto provoca aparentes errores en los clasificadores que, bajo una revisión humana, no tendrían por qué considerarse como tales.