Clasificador de Libros con Bayes Ingenuo

Arturo Márquez Flores

1. Introducción

En aprendizaje supervisado, el método de *clasificación de Bayes ingenuo* tiene como base el teorema de Bayes para estimar la probabilidad de que, dada una serie de valores de características asociadas al problema, un objeto sea clasificado en una categoría específica. Se dice que este clasificador es *ingenuo* ya que para estimar esta probabilidad depende del supuesto (*ingenuo*) de que las distribuciones de probabilidad de dichas características son mutuamente independientes. Aunque este supuesto sobresimplifica algunos problemas, resulta facilitar mucho el *entrenamiento* del algoritmo de aprendizaje supervisado sin afectar demasiado su precisión como clasificador[1]. En este trabajo se expone una implementación del clasificador de Bayes ingenuo para el problema de la categorización de libros de texto. La implementación utilizada se basa en gran parte en el trabajo expuesto en [2] con adaptaciones para este caso particular. El código puede encontrarse en este repositorio

2. Conjunto de Datos

Para esta implementación, se utilizó una base de 401 libros clasificados. En este repositorio uno puede encontrar los archivos zip que contienen los libros clasificados en formato .txt y un archivo csv que liga el nombre del archivo con su categoría correspondiente.

Para consolidar esta base en un archivo que sirviera como *input* para el algoritmo de clasificación, se desarrolló el siguiente script que junta todos los textos en un archivo csv.

Listing 1: txt2csv.py

```
import csv
import sys
import glob
import os.path

folder_name = "./classified_books/"

data_path = os.path.join(folder_name,'*txt')

files = glob.glob(data_path)

all_texts = []

for file in files:
        file_name = file.replace(folder_name,'')
        with open(file) as f:
                data = f.readlines()
                data = ''.join(data)
                data = data.replace(',','')
                data = data.replace('"','')
                data = unicode(data, errors='ignore')
                all_texts.append([data,file_name])


with open('all_texts.csv', 'wb') as myfile:
```

```
    wr = csv.writer(myfile, quoting=csv.QUOTE_ALL)
    wr.writerow(['content','file_name'])
    for elem in all_texts:
        wr.writerow(elem)
```

3. Dificultades del Problema

La clasificación de libros por categoría no es una tarea trivial ya que dentro de una misma categoría el lenguaje utilizado puede variar mucho. En efecto, los libros de fantasía y ciencia ficción a veces no tienen límites bien definidos. Por otro lado, un drama escrito en el siglo XIX bien podría caer por el uso de sus palabras en una novela histórica. Para mostrar la dificultad del problema se agruparon los textos en 9 *clusters* con un algoritmo de *k-means* cuya implementación puede encontrarse en este repositorio. La siguiente tabla, por ejemplo, muestra los resultados de las 10 palabras más comunes por agrupación de los 401 textos.

cluster	1	2	3	4	5	6	7	8	9	10
0	bolt	paint	startl	start	pain	loosen	loos	starlight	watch	cartridg
1	best	shock	offer	2001	txt	fallen	array	20	branch	roll
2	array	rock	fairbank	rose	faith	room	fall	fallen	fals	roll
3	hand	home	caus	includ	incidental	ceas	small	20	doe	inaccur
4	agree	born	attach	includ	including	critic	incomplet	increas	abid	attempt
5	judith	finess	update	iv	dim	misde	consult	germ	schmid	fine
6	charit	afternoon	glass	corner	update	research	afraid	requir	request	represent
7	long	sort	companion	sound	grow	group	ground	complet	gross	complianc
8	infring	ani	destroy	special	choos	person	despit	near	follow	machin

La falta de representatividad de estas palabras en géneros literarios específicos hace que, como se muestra en la siguiente gráfica, las agrupaciones no separen adecuadamente los géneros. Por supuesto que el resultado de este método de agrupación depende fuertemente de la representación vectorial de los libros. La próxima sección profundiza este tema.

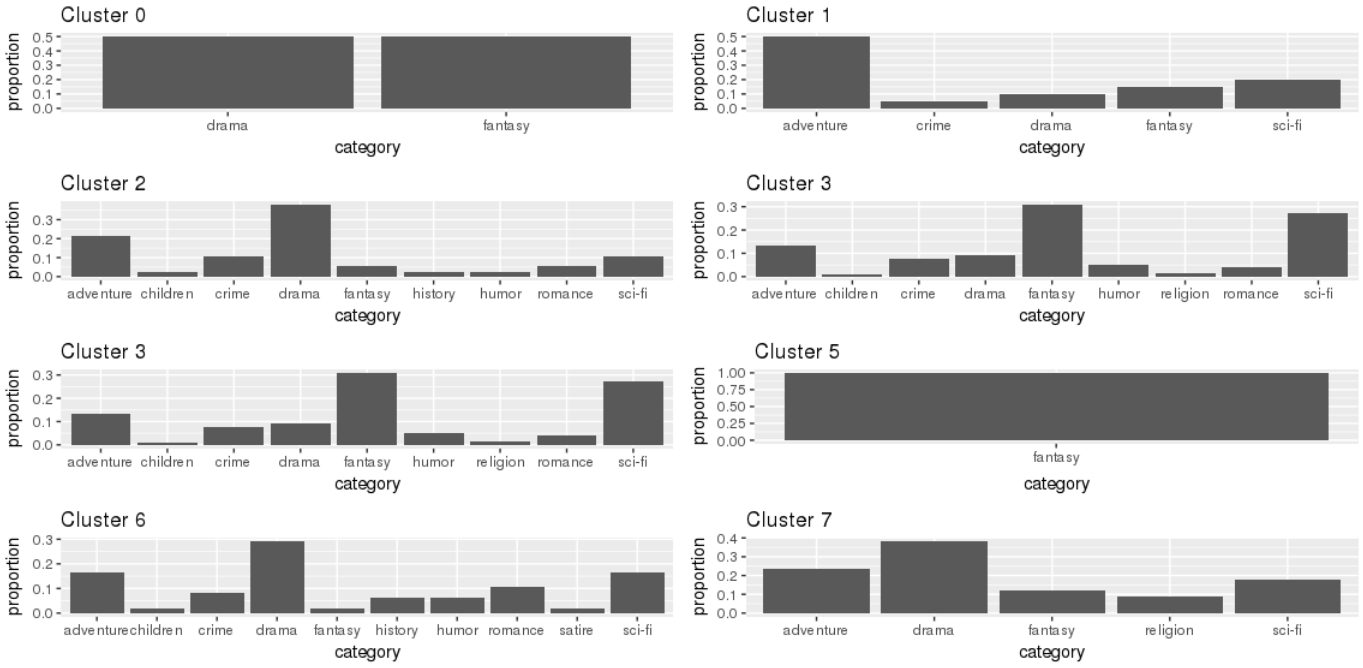

Figura 1: Clusters

Otra dificultad en el problema de clasificación es el balance de categorías en el conjunto de datos disponibles. La sobrerepresentación de algunas categorías podría causar un sesgo en el clasificador. En efecto, esta es una gran debilidad del algoritmo implementado y es discutido en la última sección.

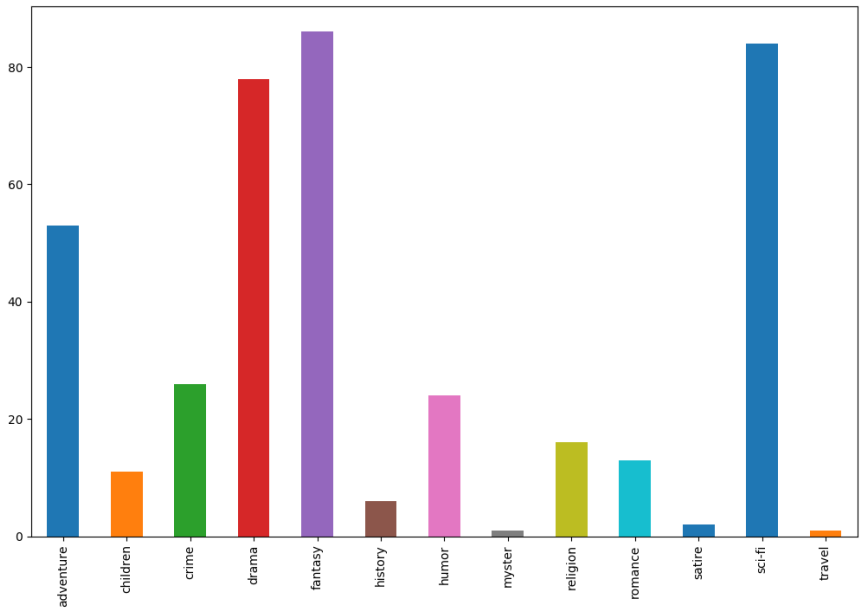

Figura 2: Categorías

Con el fin de suavizar el impacto del imbalance entre categorías y para poder implementar la validación cruzada, se eliminan las categorías *children*, *history*, *mystery*, *satire* y *travel*. De esta manera, nos quedamos con 380 documentos en total y con la siguiente distribución de categorías.

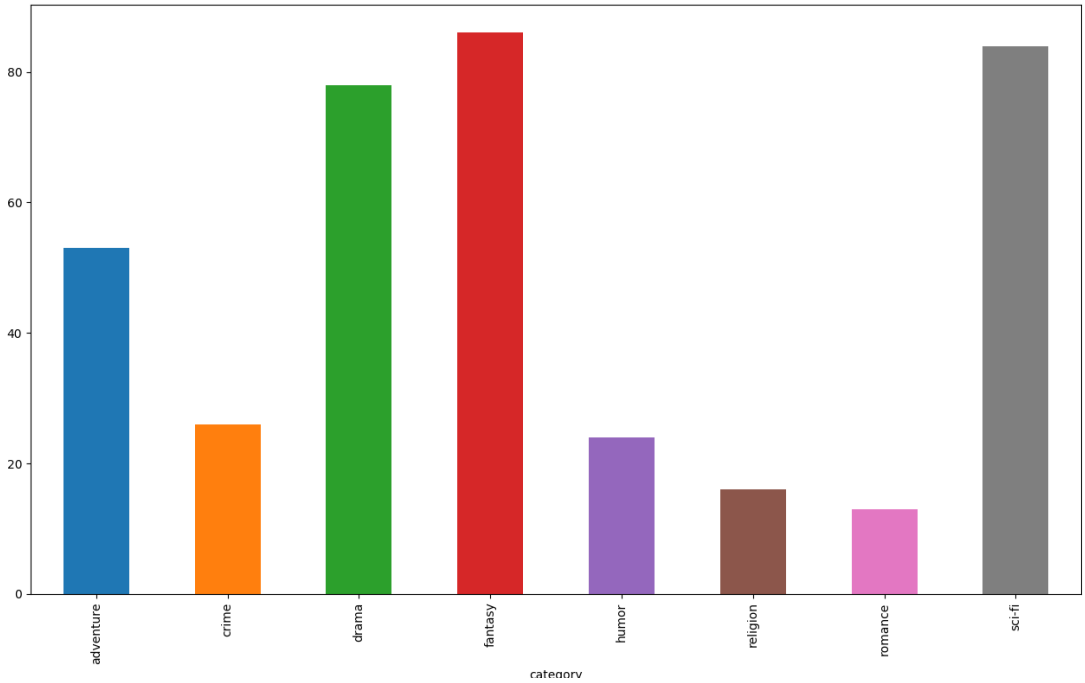

Figura 3: Categorías

4. Modelo

Dado un conjunto de K categorías $\mathcal{C} = \{C_k\}_{k=1}^{K}$, queremos saber con cuál de todas está asociado un vector de características $X = (X_j)_{j=1}^{p}$. De tal manera, queremos estimar la probabilidad de que, dado $X = \mathbf{x}$, obtengamos la clase k, $Pr(C_k|X = \mathbf{x})$, para resolver el siguiente problema de maximización:

$$C = \underset{C_k \in \mathcal{C}}{\operatorname{argmax}} \quad Pr(C_k|X = \mathbf{x})$$

Para estimar la distribución, se utiliza una muestra $\{x_i, c_i\}_{i=1}^{n}$ de observaciones. Además, por el teorema de Bayes y asumiendo (ingenuamente) independencia en las entradas del vector de características, notemos que:

$$Pr(C_k|X = \mathbf{x}) = \frac{1}{Pr(\mathbf{x})} Pr(C_k) \prod_{j=1}^{p} Pr(\mathbf{x}_j|C_k)$$

Por lo tanto, el problema se simplifica en el siguiente:

$$C = \underset{C_k \in \mathcal{C}}{\operatorname{argmax}} \quad Pr(C_k) \prod_{j=1}^{p} Pr(\mathbf{x}_j|C_k)$$

En particular, uno de los modelos que se toman para la distribución $Pr(X = \mathbf{x}|C_k)$ es el multinomial, con lo cual terminamos con un problema como el siguiente:

$$C = \underset{C_k \in \mathcal{C}}{\operatorname{argmax}} \quad Pr(C_k) \prod_{j=1}^{p} p_{j,k}^{x_j} \tag{1}$$

donde p_j, k es la probabilidad de el evento atómico j, dada una clase k.

4.1. Clasificación de Texto

En el caso de la clasificación de texto, los vectores de características son convencionalmente vectores cuyas entradas son los conteos de las palabras (o, más generalmente, n-gramas) en el vocabulario de los textos [3] [4]. Sin embargo, existen varios obstaculos en este enfoque. El primero quizá sea el hecho de que todos los documentos tendrán palabras comunes en el lenguaje utilizado (*stop words*) que deben de ignorarse. Segundo, para poder diferenciar mejor los documentos queremos que, más allá de las palabras comunes del lenguaje, se resalten las palabras que son características del documento y no del corpus entero. Por ejemplo, si todos los documentos tienen una estructura similar dividida por capítulos nos gustaría que la palabra *capítulo* no sea muy relevante en la representación vectorial del documento. Para esto, se utiliza un método común de estandarización llamado *tf-idf*(*text frequency - inverse document frequency*). La métrica *tf-idf* puede resumirse de la siguiente manera:

$$tfidf(t, d, D) = f_{t,d} log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Donde t es el término o palabra en cuestión, d es un documento en el conjunto de documentos D, y $f_{t,d}$ es el conteo del término t en el documento d.

En esta implementación se utilizaron tanto unigramas como bigramas. De esta manera, los 380 documentos quedan representados en vectores de 299,303 valores ponderados por el método tf-idf. Es importante notar que en el supuesto de la *bolsa de palabras* que se hace en esta implementación, la representación vectorial no provee ninguna información sobre la posición de las palabras en un documento, sino solamente sobre su frecuencia en un documento. Veremos más adelante que esto puede ser un problema. En este caso, los $p \cdot K$ parámetros, $p_{j,k}$, del problema (1) pueden ser estimados por la siguiente fórmula (informal):

$$\hat{p}_{j,k} = \frac{\sum_{d \in C_k} N_{j,d} + \alpha}{\sum_{j \in V} \sum_{d \in C_k} N_{j,d} + \alpha |V|}$$

Donde $N_{j,d}$ es el número de ocurrencias de la palabra j en el documento d, V es el vocabulario total utilizado de n-gramas y α es un parámetro de suavización, que en esta implementación toma el valor de .001 [6]. Se tomó este valor, pues para muestras desbalanceadas como la nuestra la suavización puede empeorar el sesgo del clasificador [7]

5. Resultados

Como preámbulo y para conocer un poco más sobre los resultados, podemos analizar los unigramas y los bigramas más correlacionados en cada categoría. En teoría, si son indicativos del género entonces puede ser que el modelo tenga una oportunidad de ser un buen clasificador. La siguiente tabla muestra los resultados.

	category	unigrams	bigrams
1	adventure	holmes and lestrade	mr holmes and said holmes
2	crime	heah and crap	miss emily and takes pains
3	drama	casterbridge and budmouth	mrs hurst and mis ess
4	fantasy	poe and scrooge	allan poe and edgar allan
5	humor	signior and breeders	william shakespeare and complete works
6	religion	aurelius and sexuality	god brother and society ought
7	romance	goggles and pokey	let wear and said henrietta
8	sci-fi	headlights and martian	said herbert and cried princess

El clasificador fue entrenado con 80 % de los datos disponibles y probado con el restante 20 %. Esta partición fue hecha aleatoriamente, y, como fue esperado, su precisión fue baja: 51 %. Incluso después de un ejercicio de validación cruzada de 5 cruces se obtiene una medida de precisión de tan sólo 46 %. En efecto, cuando el clasificador fue usado en los siguientes textos que están fuera del conjunto de entrenamiento, se obtuvieron malos resultados. Las siguientes dos figuras muestran el desempeño del clasificador, en una matriz de confusión, y una tabla de métricas por categoría.

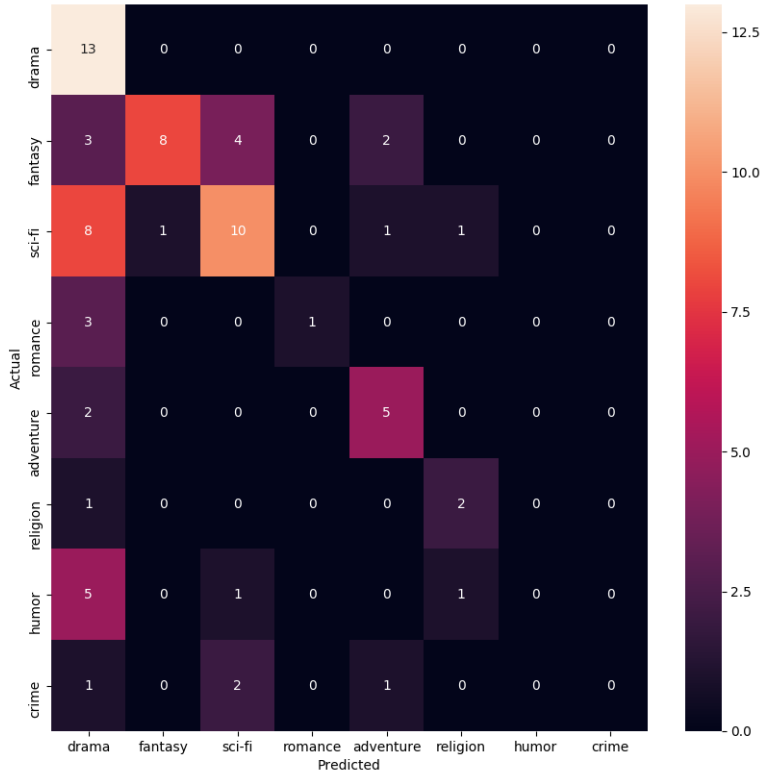

Figura 4: Métricas de Desempeño por Categoría

	precision	recall	f1-score	support
drama	0.36	1.00	0.53	13
fantasy	0.89	0.47	0.62	17
sci-fi	0.59	0.48	0.53	21
romance	1.00	0.25	0.40	4
adventure	0.56	0.71	0.63	7
religion	0.50	0.67	0.57	3
humor	0.00	0.00	0.00	7
crime	0.00	0.00	0.00	4
avg / total	0.55	0.51	0.48	76

Figura 5: Métricas de Desempeño por Categoría

Por otro lado, se utilizó el clasificador para clasificar una serie de textos para hacer una comparación a detalle. La precisión de esta calificación también está cerca del 50 %.

	file_name	prediction	actual
1	SW1	adventure	sci-fi
2	Luther	sci-fi	religion
3	King_Arthur	fantasy	fantasy
4	HP1	fantasy	fantasy
5	Murder_Gunroom	sci-fi	crime
6	El_Dorado	adventure	adventure
7	Book_Nonesense	fantasy	comedy
8	Pride_and_Prejudice	drama	drama
9	Futuria_Fantasia	sci-fi	sci-fi
10	LOTR1	sci-fi	fantasy
11	Treasure_Island	fantasy	fantasy

5.1. Balanceo de Muestra

Para resolver los problemas del balance de la muestra, se utilizó un método de remuestreo para que cada categoría tuviera un peso aproximadamente igual en la muestra. Las siguientes figuras son los resultados análogos al clasificador original.

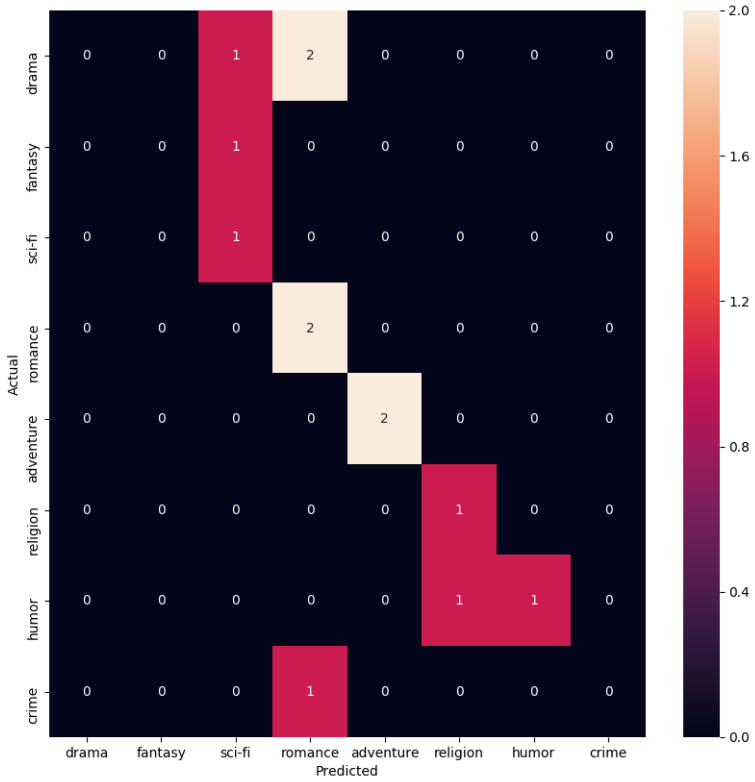

Figura 6: Métricas de Desempeño por Categoría

	precision	recall	f1-score	support
drama	0.00	0.00	0.00	3
fantasy	0.00	0.00	0.00	1
sci-fi	0.33	1.00	0.50	1
romance	0.40	1.00	0.57	2
adventure	1.00	1.00	1.00	2
religion	0.50	1.00	0.67	1
humor	1.00	0.50	0.67	2
crime	0.00	0.00	0.00	1
avg / total	0.43	0.54	0.43	13

Figura 7: Métricas de Desempeño por Categoría

Aunque en este caso se alcanzó una precisión de 54 %, la confiabilidad de este clasificador está muy limitado, pues el método de balanceo utilizado hace que tanto los datos de entrenamiento como de prueba sean conjuntos muy pequeños. Esto es porque para asegurar que todas las clases tuvieran aproximadamente el mismo peso, el número máximo de observaciones por clase no podía exceder al número de observaciones de la clase más pequeña.

Para resolver este último problema, se descartó el método de balanceo de muestra utilizado anteriormente y, en su lugar, se descartaron todas las categorías excepto *drama*, *fantasy*, *sci-fi* y *adventure*. Este método alcanzó una precisión total de 56 % y 55 % con validación cruzada (5 cruces). Las siguientes figuras muestran los resultados.

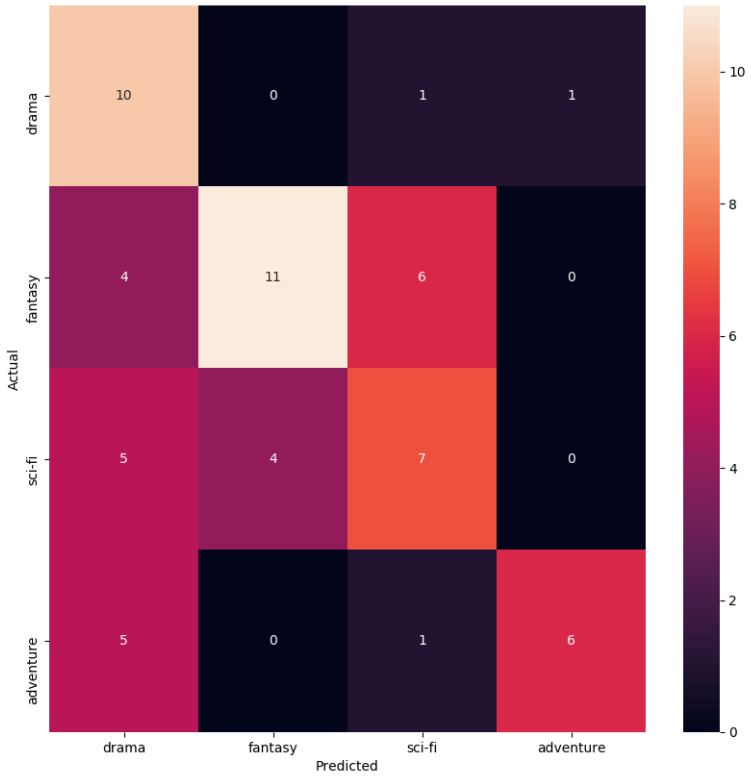

Figura 8: Métricas de Desempeño por Categoría

	precision	recall	f1-score	support
drama	0.42	0.83	0.56	12
fantasy	0.73	0.52	0.61	21
sci-fi	0.47	0.44	0.45	16
adventure	0.86	0.50	0.63	12
avg / total	0.63	0.56	0.56	61

Figura 9: Métricas de Desempeño por Categoría

	file_name	prediction	actual
1	SW1	sci-fi	sci-fi
2	King_Arthur	sci-fi	fantasy
3	HP1	sci-fi	fantasy
4	El_Dorado	adventure	adventure
5	Pride_and_Prejudice	drama	drama
6	Futuria_Fantasia	sci-fi	sci-fi
7	LOTR1	sci-fi	fantasy
8	Treasure_Island	sci-fi	fantasy

6. Discusión y Conclusión

El problema de clasificación de libros por género utilizando el método de Bayes ingenuo no es trivial. En este trabajo se muestra que se necesita, primero, obtener una mejor muestra de libros clasificados que sea lo suficientemente grande y balanceada. El problema del balanceo de la muestra es un obstáculo para la estimación de un clasificador preciso [8]. Aunque se adoptaron algunas medidas para aliviar los problemas encontrados, el

margen de mejora fue mínimo. Una hipótesis que surge es que la precisión baja del clasificador sea no sólo a razón de la falta de una muestra adecuada, sino también de la inherente no-linealidad del problema. Aunque puede ser un método bueno para la clasificación de algunos documentos, en general, puede suponerse que un género literario no está sólo caracterizado por la frecuencia de ciertas palabras sino, más allá, por el estilo en el desarrollo de su trama.

Referencias

[1] T. Hastie, R. Tibshirani, and J. Friedman. *Elements of Statistical Learning* Springer Series in Statistics Springer New York Inc., New York, NY, USA, (2001)

[2] multi-class-text-classification-with-scikit-learn

[3] text-classification-and-naive-bayes

[4] text-classification-and-naive-bayes-slides

[5] undersampling-and-oversampling-imbalanced-data

[6] multinomial-naive-bayes

[7] Frank E., Bouckaert R.R. *Naive Bayes for Text Classification with Unbalanced Classes*. In: Fürnkranz J., Scheffer T., Spiliopoulou M. (eds) Knowledge Discovery in Databases: PKDD 2006. PKDD 2006. Lecture Notes in Computer Science, vol 4213. Springer, Berlin, Heidelberg (2006)

[8] J. D. M. Rennie, L. Shih, J. Teevan, D. R. Karger *Tackling the poor assumptions of naive bayes text classifiers.* ICML'03 Proceedings of the Twentieth International Conference on International Conference on Machine Learning Pages 616-623.