E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Clasificación multi-etiqueta de géneros cinematográficos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Oier Zalba Huarte

Tutor: Mikel Sesma Sara

Pamplona, septiembre 2022

RESUMEN

Este trabajo trata de clasificación multi-etiqueta con machine learning. En particular, a partir de un dataset con títulos y sinopsis de películas en inglés, se tratará de predecir a qué género o géneros pertenece. Para ello nos valdremos de diferentes técnicas de preprocesamiento de datos, minería de texto y aprendizaje automático. A diferencia de los algoritmos de clasificación habituales que están orientados a dos o más clases excluyentes entre sí, en el caso de la multi-etiqueta, cada instancia puede estar clasificada en varias clases a la vez. Por lo tanto, valoraremos qué modificaciones, estrategias y técnicas seguir para abordar el problema.

PALABRAS CLAVE

Aprendizaje automático

Bag of Words

Clasificación multi-etiqueta

Evaluación multi-etiqueta

ABSTRACT

This paper deals multi-label classification with machine learning. In particular, from a dataset with titles and synopsis of movies in English, we will try to predict to which genre or genres they belong. For this purpose, we will use different data preprocessing techniques, text mining and machine learning. Unlike usual classification algorithms that are oriented to two or more mutually exclusive classes, in the case of multi-labeling, each instance can be classified into several classes at the same time. Therefore, we will assess which modifications, strategies and techniques to follow to address the problem.

KEY-WORDS

Machine Learning

Bag of Words

Multi-label classification

Multi-label evaluation

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

ÍNDICE DE CONTENIDO

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

ÍNDICE DE ILUSTRACIONES

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

ÍNDICE DE TABLAS

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

ÍNDICE DE ECUACIONES

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

ÍNDICE DE GRÁFICAS

1 INTRODUCCIÓN

El aprendizaje automático es una aplicación de la inteligencia artificial que proporciona al sistema la capacidad de aprender y mejorar automáticamente a partir de experiencias o ejemplos en lugar de programación explícita. Hoy en día, desempeña algunos aspectos de las habilidades humanas, sin llegar a todo el potencial de la inteligencia propiamente dicha. El aprendizaje automático está cambiando el mundo, integrándose en todos los segmentos como servicios sanitarios, educación, transporte, alimentación, etc.

La clasificación de textos es un problema de aprendizaje automático donde se pretende predecir a qué clases pertenece un determinado texto. En contraste con la clasificación "clásica", los datos no están estructurados, lo que supone una dificultad añadida. Es por ello, que se trata de una de las áreas de investigación que más importancia ha cobrado en la actualidad, debido a los grandes y valiosos volúmenes de textos digitales que manejan empresas del sector de la información y en concreto las redes sociales. Twitter puede ser un ejemplo de empresa cuyo modelo de negocio gira en torno la extracción de conocimiento a partir de mensajes.

Para este trabajo nos encontramos con un dataset con sinopsis de películas extraídas de las webs imdb y Wikipedia. Cada película está clasificada en uno o más géneros, lo que supone que nos encontramos ante una clasificación multi-etiqueta. En clasificación de texto, es habitual que una carta, articulo etc. pueda transmitir varias ideas a la vez, es decir, pueda pertenecer a varias categorías o clases simultáneamente. Sin embargo, los clasificadores que aceptan esta condición no son tan populares. Este es el desafío principal del proyecto.

1.1 OBJETIVOS

El objetivo principal de este trabajo fin de grado es diseñar un algoritmo de clasificación multi-etiqueta para clasificar géneros de películas en base a su sinopsis.

Para ello, nos planteamos los siguientes objetivos específicos:

1. Analizar la base de datos de sinopsis de peliculas y aplicar un preprocesamiento para limpiar impurezas y facilitar el acceso a información de calidad.
2. Aplicación de técnicas de minería de texto para extraer características representativas de documentos extensos.
3. A la hora de clasificar con aprendizaje automático, probar con dos estrategias diferentes:
    a. Por una parte, transformar el problema de las multi-etiquetas a multi-clase, codificando cada combinación de géneros en una clase diferente. A partir de aquí, obtener los modelos con clasificadores convencionales como Naïve Bayes.
    b. Por otra parte, tratar con algoritmos de clasificación con soporte nativo para multi-etiqueta o técnicas para adaptar clasificadores que no lo acepten.
4. Estudiar las mejores métricas de evaluación para la elección del mejor modelo de clasificación.
5. Obtener el mejor modelo para clasificar las películas a sus géneros.

2 METODOLOGÍA

Para la clasificación de sinopsis de películas a géneros cinematográficos nos valdremos de técnicas de preprocesamiento de datos, minería de texto y aprendizaje automático. A continuación, se explicarán conceptos básicos para el correcto entendimiento de estos.

2.1 INTELIGENCIA ARTIFICIAL

La inteligencia artificial es una disciplina de las ciencias de computación que consiste en desarrollar algoritmos matemáticos que puedan replicar de alguna manera la mente o inteligencia humana, es decir, que sean capaces aprender (adquisición de información y reglas para interpretar la información), razonar (empleando estas reglas, sacar conclusiones y predicciones), autocorregir…

Pocos años después de la victoria en la segunda guerra mundial, años más corta gracias a la labor de Alan Turing en la desencriptación de la máquina enigma [1], este mismo se planteó la siguiente pregunta: *"¿Pueden las máquinas pensar?"*. En 1950 publicó el artículo científico *"Computing Machinery and Intelligence"* y su subsecuente *"Turing test"* donde estableció los cimientos y visión de la IA [2]. No fue hasta 1956 cuando el informático estadounidense John McCarthy acuñó el término IA, dando comienzo a la disciplina.

A pesar de que se desarrollan múltiples algoritmos, no fue hasta la década de los 90 cuando da un giro, por múltiples factores: los datos se están produciendo y almacenando, la potencia computacional necesaria es abordable, herramientas de software potentes están disponibles… hasta hoy en día, donde está presente en todo ámbito tecnológico e incluso fuera de él.

La principal diferencia de la IA respecto a otros programas es que no es necesario programarla estructuralmente para cada problema. Según los tipos de algoritmos empleados, tenemos el aprendizaje automático o su subcategoría aprendizaje profundo, la cual es computacionalmente más compleja, pero requiere de menos intervención humana.

2.2 APRENDIZAJE AUTOMÁTICO

Según Arthur Samuel, "El área de estudio que da a los ordenadores la capacidad de aprender sin ser programados explícitamente" [3]. Mediante el uso de métodos estadísticos, los algoritmos se entrenan para hacer clasificaciones, predicciones… y extraer conocimiento valioso de grandes volúmenes de datos. Análogamente al aprendizaje humano, el aprendizaje automático se suele valer de ejemplos con los que aprender y extraer patrones con los que se pretende generalizar para futuros nuevos datos, evitando el sobre entrenamiento.

Podemos encontrar infinidad de ejemplos en nuestro día a día:

- Spotify, el servicio de música y pódcast más popular, propone canciones y listas de reproducción según gustos o patrones de comportamiento.
- Empresas de transporte como UPS utiliza machine learning para mejorar y optimizar rutas, en particular, minimizan los giros a la izquierda, los cuales suponen un mayor riesgo [4].

2.3 APRENDIZAJE PROFUNDO

Deep learning es un subtipo de machine learning. Automatiza la extracción de características del proceso, elimina parte de la intervención humana y permite el uso de conjuntos de datos más grandes. El aprendizaje profundo puede considerarse un "aprendizaje automático escalable" según Lex Fridman [5].

Se describen algoritmos que analizan los datos con una estructura lógica similar a la que utilizaría una persona, en otras palabras, el diseño se inspira en la red neuronal biológica del cerebro humano. Las aplicaciones utilizan una arquitectura de algoritmos en capas denominada red neuronal artificial (véase 2.12). Si bien los algoritmos de aprendizaje automático tradicionales son lineales, los algoritmos de aprendizaje profundo se apilan en una jerarquía de complejidad y abstracción cada vez mayor, obteniendo resultados notablemente mejores.

Dada su naturaleza, podemos encontrar muchos ejemplos en tareas que realizan personas. Los casos de uso actuales para el aprendizaje profundo incluyen todo tipo de aplicaciones de

reconocimiento de imágenes, análisis de big data, traducción de idiomas, diagnósticos médicos, seguridad de red…

2.4 TIPOS DE APRENDIZAJE

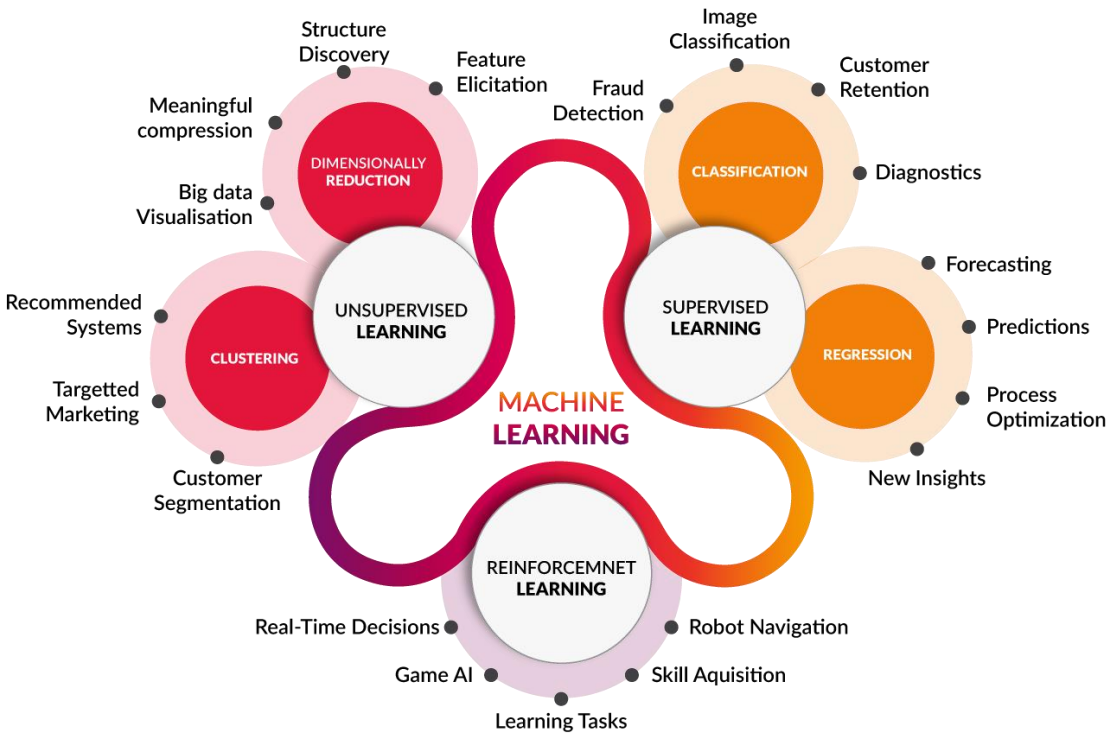


Ilustración 1: Tipos Machine Learning

2.4.1 APRENDIZAJE SUPERVISADO

En estos métodos se parte de un conocimiento a priori, donde cada instancia (ejemplo) está asociado a una salida conocida, facilitando considerablemente la tarea y posterior evaluación del modelo. El algoritmo aprende de las muestras de entrada, comparando el resultado del modelo y la etiqueta real, y realiza las compensaciones necesarias en función de la medida de error en la predicción del resultado.

2.4.1.1 REGRESIÓN

La salida esperada del modelo es una variable cuantitativa, numérica. Por ejemplo, valoración de bienes de inmuebles, la salida debe ser el valor en euros.

2.4.1.2 CLASIFICACIÓN

La salida esperada del modelo es una variable cualitativa, categórica. Por ejemplo, detección de fraude en transacciones bancarias, la salida debe ser si es fraude o no. Dependiendo del número de clases diferentes, podemos hablar de clasificación binaria (2 clases) o multinomial (múltiples clases) (véase 2.11).

2.4.2 APRENDIZAJE NO SUPERVISADO

En este caso, no sabemos a priori la salida de los datos, por lo que ajustan su modelo predictivo tomando en cuenta solo los datos de entrada, sin importar los de salida.

2.4.2.1 CLUSTERING

Consiste en agrupar elementos en conjuntos con características similares según la función de distancia. Por ejemplo, segmentación de clientes (véase 2.10).

2.4.2.2 EXTRACTOR DE CARACTERÍSTICAS

Construyen una representación interna (embedding o código) de los datos sin necesidad de etiquetas. Por ejemplo, compresión de imágenes.

2.4.3 APRENDIZAJE POR REFUERZO

Los algoritmos de aprendizaje por refuerzo consisten en maximizar una función de coste donde cada acción tiene una recompensa preestablecida, es decir, cada acción tiene un coste positivo en el caso de que se considere bueno y negativo en el caso contrario. Esto se le conoce como acción-recompensa y resulta intuitivo por su similitud con el comportamiento humano. Está estrechamente relacionado con la robótica. La navegación de los robots es un ejemplo de ello.

2.5 KDD

El KDD (Knowledge Discovery from Databases o descubrimiento de conocimientos a partir de bases de datos) es el proceso completo de extracción de conocimiento a partir de bases de datos. El término se acuñó en 1989 para enfatizar que el conocimiento es el producto final de un proceso iterativo de descubrimiento guiado por los datos. Este está compuesto por varias etapas.

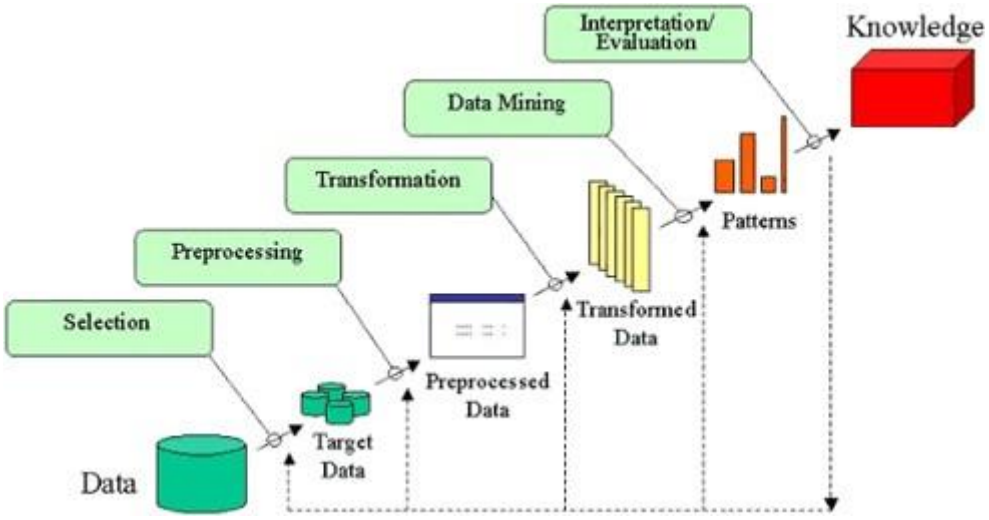


Ilustración 2: Etapas KDD

2.5.1 COMPRENSIÓN DEL PROBLEMA Y LOS DATOS

En primer lugar, hay que analizar el problema y establecer los límites y objetivos que se quieren obtener al final del proceso. La familiarización con el dominio del problema y la obtención de conocimiento a priori disminuye el espacio de soluciones posibles

2.5.2 CREACIÓN DE UN SET DE DATOS (DATASET)

En segundo lugar, es necesario seleccionar e integrar datos útiles provenientes de múltiples fuentes objetivas. Es necesario que la muestra obtenida sea representativa del problema a abordar. Estos datos se pueden obtener de bases de datos relacionales, bases de datos documentales, bases de datos multimedia, datos obtenidos de la web (World Wide Web)… Hay que integrar los datos para crear información homogénea: resolver duplicidades e inconsistencias, resolver problemas de representación, escala o codificación.

2.5.3 PREPROCESAMIENTO DE DATOS

El objetivo del preprocesamiento de datos es manipular y transformar los datos originales de tal forma que la información pueda ser expuesta o facilitada. Aproximadamente el 40% de los datos son impuros, reduciendo en gran medida la calidad del reconocimiento. Podemos encontrar: valores inconsistentes o inexactos, ejemplos y variables redundantes, valores incompletos o perdidos, valores outliers o fuera del rango, valores con ruido, gran dimensionalidad de los datos… El preprocesamiento consume gran parte del tiempo total del KDD.

La preparación de datos incluye:

- Transformación de datos
- Limpieza de datos
- Detección de outliers
- Normalización de datos
- Imputación de valores perdidos
- Identificación de ruido
- Reducción de ejemplos y variables

2.5.4 MINERÍA DE DATOS

El objetivo de la minería de datos es producir nuevo conocimiento que pueda utilizar el usuario. Para ello se generará un modelo que describa patrones y relaciones basado en el dataset, y a partir de este se podrán predecir o generalizar futuros datos, en otras palabras, aprender del pasado para predecir el futuro. Se debe decidir qué técnica emplear, con qué tipo de modelo… Aquí entraría el machine learning entre otros (véase 2.2).

2.5.5 EVALUACIÓN

En última estancia, se debe elegir el mejor modelo que interprete mejor los datos y presentar los resultados. Se debe decidir:

2.5.5.1 CRITERIOS DE EVALUACIÓN

Métricas precisas, comprensibles, interesantes.

2.5.5.2 METODOLOGÍA DE EVALUACIÓN

Para evaluar la validez del conocimiento extraído, se divide el dataset en un conjunto de train (entrenamiento) donde se aplicará la minería y un conjunto de test (prueba) donde se evaluará las predicciones del modelo. También se puede crear un conjunto de validación donde extraer otros parámetros de la minería. Véanse las técnicas de evaluación 0.

2.5.6 DIFUSIÓN Y UTILIZACIÓN DEL NUEVO CONOCIMIENTO

Con el conocimiento adquirido, se podrán tomar mejores decisiones.

2.6 MÉTRICAS DE EVALUACIÓN

2.6.1 MATRIZ DE CONFUSIÓN

Dado un problema de clasificación con m clases, una matriz de confusión es una matriz m x m en la que una entrada $C_{i,j}$ indica el número de ejemplos que se han asignado a la clase C_j, cuando la clase correcta es C_i.

	Realmente es positivo	Realmente es negativo
Predicho como positivo	Verdaderos Positivos (VP)	Falsos Positivos (FP)
Predicho como negativo	Falsos Negativos (FN)	Verdaderos Negativos (VN)

Ilustración 3: Matriz de confusión de dos clases

2.6.2 ACCURACY

Relación de aciertos (verdaderos positivos y verdaderos negativos) respecto al total.

$$Accuracy = \frac{VP + VN}{VP + VN + FP + FN}$$

Ecuación 1: Accuracy

$$Error = 1 - Accuracy$$

Ecuación 2: Error

2.6.3 PRECISIÓN

Proporción de ejemplos clasificados en la clase positiva (clase de interés) que son realmente de la clase positiva.

$$Precision = \frac{VP}{VP + FP}$$

Ecuación 3: Precisión

2.6.4 RECALL

Ejemplos de la clase positiva (clase de interés) clasificados correctamente.

$$Recall = \frac{VP}{VP + FN}$$

Ecuación 4: Recall

2.6.5 FSCORE

Media armónica de la precisión y el recall, compromiso entre ambas.

$$Fscore = 2 * \frac{precision * recall}{precision + recall}$$

Ecuación 5: Fscore

2.7 TÉCNICAS EVALUACIÓN

2.7.1 HOLD-OUT

Consiste en dividir la BD en dos conjuntos independientes, subconjunto train y test. La proporción puede ser 80%-20%. Se asume que ambos conjuntos representan bien el problema, asegurando que la distribución de las clases en los conjuntos de entrenamiento y de test sea similar a la del conjunto original (estratificación). Ahora bien, puede haber sesgo ya que se utiliza muestreo, puedes tener suerte digamos. Hold-out con repetición es más fiable.

2.7.2 VALIDACIÓN CRUZADA

En la validación cruzada, se dividen todos los ejemplos en k subconjuntos de forma que: los subconjuntos tienen similar tamaño y se emplea estratificación. Se realizará el proceso de aprendizaje y evaluación k veces, cada una de ellas teniendo un subgrupo diferente de test. El conjunto de train lo compondrán el resto de los subgrupos. La validación cruzada previene el overlap.

2.7.3 LEAVE-ONE-OUT

Es un caso especial de validación cruzada donde k es igual al número de ejemplos. El proceso será determinista y se utiliza el máximo posible de datos para la inducción del clasificador, pero conlleva un alto coste computacional.

2.8 MINERÍA DE TEXTO

La minería de texto es el uso de técnicas de computación para extraer información de alta calidad a partir de texto. En contraste con la minería estándar, los datos no están estructurados, lo que supone una dificultad añadida. El texto puede provenir de libros, informes, noticias, comentarios, mensajes… Se estima que cerca del 80% de los datos disponibles no son datos estructurados.

Para los humanos la compresión de lenguaje natural resulta sencillo, pero no es así para los ordenadores. Hay términos que pueden tener diferentes acepciones, frases ambiguas, sarcasmo, anglicismos… Para ciertos textos se cuenta tanto con cierto contexto y conocimiento, como con sentido común y razonamiento por parte del lector.

2.9 BAG OF WORDS

Bag of words (bolsa de palabras) es una técnica de representación de texto para aplicar machine learning que considera cada documento como un conjunto de palabras. La premisa es que dos documentos, probablemente sean similares si tienen un contenido de palabras parecido. Las licencias que este modelo toma, aparte de las propias de la minería de texto son: por una parte, considera todas las palabras independientes, y por otra, no tiene en cuenta el orden de las palabras. Por lo tanto, pasa por alto muchas condiciones que consideraríamos relevantes en un texto. A pesar de su simpleza, resulta ser una técnica eficaz con un rendimiento considerable.

Cada documento del dataset estaría representado por un vector multidimensional, siendo cada dimensión una palabra diferente de todas las palabras de todos los documentos del dataset (corpus).

Document	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

Ilustración 4: Matriz dataset

En este caso el valor de cada elemento es el número de veces que está presente en cada documento.

A la hora de elegir los términos que representaran todo el corpus surge el primer problema. En problemas de tamaño medio, la cantidad de elementos únicos se dispara, sin embargo, la mayoría no ofrece gran información semántica. Es necesario reducir los termino considerablemente.

2.9.1 NORMALIZACIÓN

Es necesario transformar los términos a un formato general:

Por ejemplo, Hola, hola, HOLA -> ola

- Eliminación de signos de puntuación
- Eliminación de números
- Pasar todo a minúsculas

2.9.2 ELIMINACIÓN STOP-WORDS

Eliminación de palabras habituales que no contienen ningún valor semántico en los idiomas, como preposiciones, artículos, pronombres… Cada lenguaje tiene las suyas y se estiman que suelen suponer el 30% de las palabras de los documentos.

En inglés podrían ser: I, a, the, this, they…

2.9.3 REDUCCIÓN DE PALABRAS A SU RAÍZ

Reducir las palabras a su forma básica o lexema agrupa palabras con un significado idéntico pero que fueran ligeramente diferentes.

- Lemmatization: reduce las palabras a su forma básica manteniendo sus características morfológicas e información sintáctica.

  Por ejemplo: argue, argued, argues, arguing -> argue

- Stemming: acorta las terminaciones de las palabras, más agresivo que lemmatization.

  Por ejemplo: argue, argued, argues, arguing -> argu

2.9.4 ELIMINACIÓN DE TÉRMINOS CON FRECUENCIAS MUY ALTAS O BAJAS

Los términos muy frecuentes suponen una gran proporción del corpus, pero no suelen aportar mucha información. Están estrechamente relacionadas con las stop-words.

En contraposición, los términos poco frecuentes aportan gran información semántica, pero al no ser habituales en el corpus no son útiles. También podemos encontrar nombres propios o palabras mal escritas que no aportan nada.

Las frecuencias medias son las que mejor representan al corpus.

2.9.5 PESOS

Una vez se tienen los elementos que representaran el corpus, se debe asignar un peso a cada uno, o sea, su importancia. Para ello tenemos las siguientes técnicas:

2.9.5.1 PESOS BINARIOS

Los pesos son: 1 si el término está presente en el documento y 0 si no lo está. Este es el más simple y rápido, pero no tiene en consideración la frecuencia en la que un mismo término puede estar presente en un documento.

2.9.5.2 FT

El peso es igual a la frecuencia con la que una palabra sale en un documento. Como puede haber documentos con diferentes tamaños se normalizará dividiéndolo por el número de palabras del documento.

2.9.5.3 IDF

Inversa de la frecuencia, pero mide la frecuencia en el corpus global, no de cada documento. De esta forma damos más peso a aquellas palabras que no sean habituales en el corpus. Tienen más poder de diferenciación.

$$IDF(t) = 1 + \log\left(\frac{N}{df(t)}\right)$$

Ecuación 6: Inversa de la frecuencia

t→término para el que se calcula el valor

N→numero de documentos del corpus

df(t) número de documentos que contiene el termino t

2.9.5.4 FT-IDF

Es el mejor método para asignar pesos en bag of words. Se valora más aquellas palabras que no son habituales en el corpus al igual que IFD, pero se combina con FT para garantizar que tengan un mínimo de frecuencia.

$$FT - IDF(t) = FT(t) * \log\left(\frac{N}{df(t)}\right)$$

Ecuación 7: FT-IDF

t→término para el que se calcula el valor

N→numero de documentos del corpus

df(t) número de documentos que contiene el termino t

2.10 TIPOS CLUSTERING

2.10.1 K-MEDOIDS

K-medoids es un método de clustering similar a k-means [6]. Ambos agrupan todos los elementos en un número de clústers (núcleos) establecido por el analista. Mientras que en k-means, cada núcleo estaría representado por su centroide, siendo este el promedio de todos los elementos de un clúster, en el caso de k-medoids, se asigna una de las instancias como centroide. Como resultado, tenemos un método más robusto, menos sensible a outliers o ruido.

Algoritmo:

1) Seleccionar k centroides aleatoriamente.
2) Calcular la matriz de distancia de todas las observaciones.
3) Asignar cada instancia a su centroide más cercano.
4) Para cada clúster, reasignar el centroide a aquel que minimice la distancia media del clúster.

5) Volver al paso 3 hasta estabilizar la asignación de centroides.

Podemos repetir el algoritmo para varios valores de k. El K óptimo será el que tenga menor valor del índice Davies-Bouldin, con los clústers más compactos y separados entre sí.

$$C_j = \frac{1}{|C_j|} \sum_{x^{(i)} \in C_j} \left|\left|x^{(i)} - \mu_{c^{(i)}}\right|\right|^2$$

Ecuación 8: Compacticidad del clúster j

$$S_{i,j} = \left|\left|\mu_i - \mu_j\right|\right|^2$$

Ecuación 9: Separación entre clusters i,j

$$R_i = max_{i,j\ i \neq j} \left\{ \frac{C_i + C_j}{S_{i,j}} \right\}$$

Ecuación 10: Peor caso cluster i

$$DB = \frac{1}{K} \sum_{i=1}^{K} R_i$$

Ecuación 11: Índice Davies-Bouldin

2.10.2 CLUSTERING JERÁRQUICO

El clustering jerárquico consiste en la creación de un árbol binario que iterativamente va juntando los clústers más similares entre sí, finalizando en un solo clúster. Cada nivel del árbol es una partición de los datos. La visualización de las asociaciones en gráficos como dendogramas es muy útil para la comprensión de los datos.

Algoritmo:

1. Cada elemento forma un clúster
2. Repetir:
    a. Juntar los clústers más similares

Distancia entre clústers:

- **Single-link**: la distancia mínima entre dos miembros de diferentes clusters. Las secuencias de puntos son las primeras en juntarse.
- **Complete-link**: la distancia máxima entre dos miembros de diferentes clusters. Lo contrario a single-link
- **Average-link**: un compromiso de las dos anteriores

$$d_{AL}(A,B) = \frac{1}{m_a m_b}\sum_{i=1}^{m_A}\sum_{j=1}^{m_B} d(i,j)$$

Ecuación 12: Average-link

2.11 TIPOS CLASIFICADORES

2.11.1 NAÏVE BAYES

Naïve Bayes son un conjunto de algoritmos basados en el teorema de Bayes junto a la hipótesis "Naïve" de la independencia condicional de los atributos. El teorema de Bayes consiste en estimar la probabilidad a posteriori de un elemento de pertenecer a una clase a partir de datos conocidos (conjunto entrenamiento).

$$P(C_j|x) = \frac{P(x|C_j) * P(C_j)}{P(x)}$$

Ecuación 13: Teorema Bayes

Junto a la suposición en Naïve Bayes de la independencia de los atributos.

$$P(x|C_j) = P(x_1, \dots, x_n|C_j) = \prod_{i=1}^{n} P(x_i|C_j)$$

Ecuación 14: Naïve Bayes

Por último, como método de clasificación se empleará MAP (Máximo a posteriori) donde se clasificará aquella clase con mayor probabilidad.

$$C_{NB} = argmax(P(C_j) * \prod_{i=1}^{n} P(x_i|C_j))$$

Ecuación 15: Clasificador Naïve Bayes

A pesar de su exceso de suposiciones, resulta un clasificador reconocido y probado en muchas situaciones reales, entre las más destacadas, clasificación de texto o filtro de spam. No requieren un gran tamaño de datos de entrenamiento para estimar los parámetros aproximados y son más eficientes comparando con métodos más sofisticados.

2.11.1.1 GAUSSIAN NAÏVE BAYES

Se supone que la probabilidad de las características es gaussiana.

2.11.1.2 MULTINOMIAL NAÏVE BAYES

Los parámetros se estiman mediante una versión suavizada de máxima verosimilitud, es decir, el recuento de frecuencias relativas. Se tienen en cuenta las características que no están presentes en las muestras de aprendizaje y evitan las probabilidades nulas en los cálculos posteriores con el suavizado. Es una de las dos variantes clásicas de Naïve Bayes para clasificación de texto.

2.11.1.3 COMPLEMENT NAÏVE BAYES

Es una adaptación del Naïve Bayes Multinomial para datos no balanceados. Utiliza las estadísticas del complemento de cada clase para calcular las ponderaciones del modelo. Suele funcionar mejor en la mayoría de las veces en clasificación de texto (incluso con un margen considerable).

2.11.1.4 BERNOULLI NAÏVE BAYES

Para datos que se distribuyen según las distribuciones multivariantes de Bernoulli, esto es, puede haber múltiples características, pero se supone que cada una es una variable de valor binario. En clasificación de texto, este clasificador resulta más conveniente para vectores de ocurrencia de palabras.

2.11.2 KNN

KNN o K Nearest Neighbor (k vecinos más cercanos) es un clasificador basado en instancias, dentro del paradigma perezoso de aprendizaje (lazy learning). No se construye ningún modelo, simplemente

se buscan las k instancias más parecidas y se clasifica en función de las clases de estos. La clase seleccionada será la que se repita con mayor frecuencia.

Es necesario elegir un buen valor de k: contra k más pequeño, más sensible será al ruido, y si es muy grande el vecindario podrá incluir vecinos de otras clases. Es bastante preciso, puesto que utiliza varias funciones lineales locales para aproximar la función objetivo. Por el otro lado, es muy ineficiente en memoria.

La distancia es la función que determina la similitud y por lo tanto, la elección de los vecinos.

Distancias variables discretas:

- Distancia de Hamming

$$\boldsymbol{d_H(e_i, e') = \sum_{j=1}^{N} D(e_i^j, e'^j) \ con \ D(e_i^j, e'^j) = \begin{cases} 0 \ si \ e_i^j = e'^j \\ 1 \ si \ e_i^j \neq e'^j \end{cases}}$$

Ecuación 16: Distancia hamming

- Value Difference Metric

$$\boldsymbol{vdm_j(x, y) = \sum_{k=1}^{M} \left| \frac{N_{j,x,k}}{N_{j,x}} - \frac{N_{j,y,k}}{N_{j,y}} \right|^q}$$

Ecuación 17: Value Difference Metric

N(j,x,k): número de ejemplos de la clase k con valor x en el atributo j

N(j,x): número de ejemplos con valor x en el atributo j

q: constante (normalmente 1 o 2)

M: número de clases

Distancias variables numéricas:

- Distancia Euclídea

$$\boldsymbol{d_e(e_j, e') = \sqrt{\sum_{j=1}^{N} (e_i^j - e'^j)^2}}$$

Ecuación 18: Distancia euclídea

- Distancia de Manhattan

$$d_m(e_j, e') = \sum_{j=1}^{N} |e_i^j - e'^j|$$

Ecuación 19: Distancia Manhattan

2.11.3 ÁRBOLES DE DECISIÓN

Los árboles de decisión son un método de clasificación no paramétrico donde el modelo se crea con reglas sencillas obtenidas de los datos. Un árbol puede verse como una aproximación constante a trozos. Su funcionamiento es similar a un diagrama de flujo donde cada nivel es una pregunta (regla) y se van desarrollando hasta llegar a las hojas donde se clasifica.

Algoritmo:

- Se selecciona el mejor atributo atendiendo a la distribución homogénea de las clases.
- Se crea un nodo de decisión y se divide el conjunto de datos en subconjuntos más pequeños (nodos hijos), según el resultado de la regla. Cada nodo estará asignado a una clase.
- Se construye el árbol recursivamente hasta las condiciones de parada:
    - Todos los ejemplos de un nodo pertenecen a la misma clase.
    - No quedan más atributos a particionar.
    - No quedan ejemplos.

Los árboles se pueden visualizar e interpretar, de forma que la toma de decisiones está expuesta para su análisis. El árbol de decisión no tiene suposiciones sobre la distribución debido a la naturaleza no paramétrica del algoritmo, por lo que no es necesario normalizar. Sin embargo, tiene alguna desventaja como la sensibilidad al ruido y pequeñas variaciones que pueden dar lugar a un árbol drásticamente diferente.

2.11.4 RANDOM FOREST

Random forest es un ensemble, es decir, un clasificador (regresor) que está formado por muchos clasificadores (regresores) base, de árboles de decisión. Se debe entrenar cada clasificador base,

siendo la diversidad de estos el factor clave, y a la hora de predecir un ejemplo, se agregan las salidas de todos ellos obteniendo la predicción final.

Algoritmo:

- En primer lugar, seleccionar el número de árboles base.
- Para cada árbol:
    - Aplicar booststrap, seleccionando n ejemplos con reemplazamiento del conjunto de entrenamiento (los ejemplos se pueden repetir y n es el número de ejemplos).
    - Se seleccionan m atributos aleatoriamente siendo m considerablemente menor de los atributos totales.
    - No se poda el árbol.
- Cada ejemplo se clasificará con el voto de todos los árboles y la clase seleccionada será la clase más votada.

Random forest tiene un buen rendimiento y es más robusto frente a ruido que un solo árbol de decisión. En contraposición, pierde la visualización y en consecuencia la interpretabilidad.

2.11.5 ADABOOST

Adaboost es un ensemble que consiste en transformar clasificadores débiles en un clasificador fuerte. En primer lugar, se consideran todos los datos, pero a medida que avanza el algoritmo se le da más peso a las instancias que predice erróneamente. Está respaldado bajo una fuerte base teórica.

Algoritmo:

- Asignar el mismo peso a todos los ejemplos.
- Para cada clasificador:
    - Entrenar con los pesos actuales.
    - Aumentar los pesos de los ejemplos mal clasificados.
    - Decrecer el peso de los ejemplos bien clasificados.
- Asignar un peso al clasificador en base a su precisión.

2.12 REDES NEURONALES

Las redes neuronales es un modelo dentro de deep learning que emula el funcionamiento del cerebro humano. El modelo consiste en ordenar operaciones matemáticas siguiendo una determinada arquitectura. La forma más común está estructurada en capas, formadas a su vez por neuronas.

Cada neurona tiene asignado una operación matemática sencilla, siendo sus parámetros la suma ponderada de las salidas de la capa anterior y el bias. La función de activación de la neurona transformará las entradas en un valor de salida. Posteriormente, pasarán el resultado a la siguiente capa con otra serie de pesos, de forma que se regule la información que se propaga. Si bien la entrada a la neurona es una combinación lineal, gracias a la función de activación se pueden generar diferentes salidas. Es aquí, donde reside el potencial de los modelos de redes para aprender relaciones no lineales.

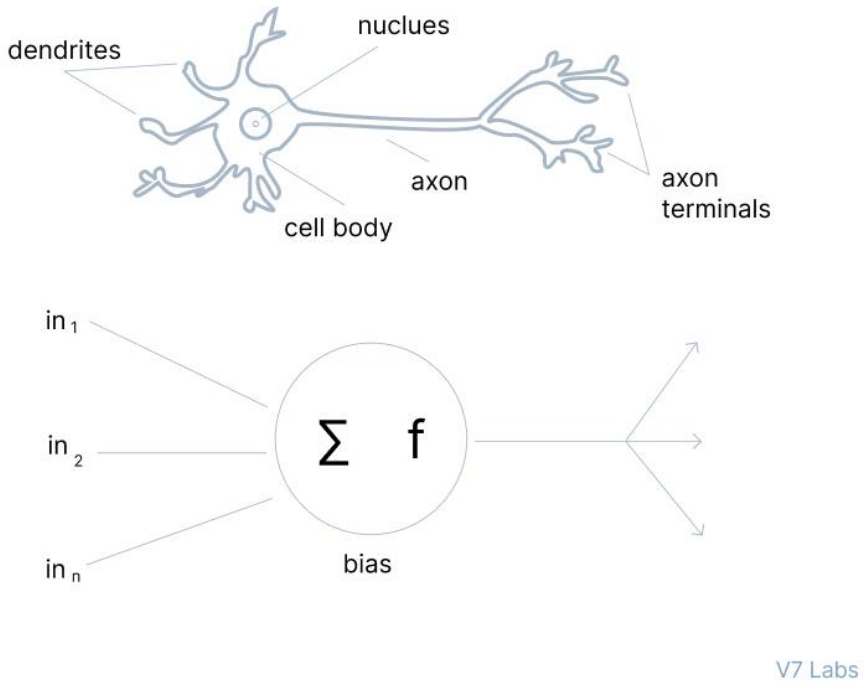


Ilustración 5: Esquema de una neurona biológica y de una neurona de una red neuronal

La primera capa se conoce como la capa de entrada y recibe los datos en crudo. Las capas intermedias u ocultas operan sus respectivas operaciones y propagan las salidas con diferentes pesos. La última capa o capa de salida representa la salida del clasificador.

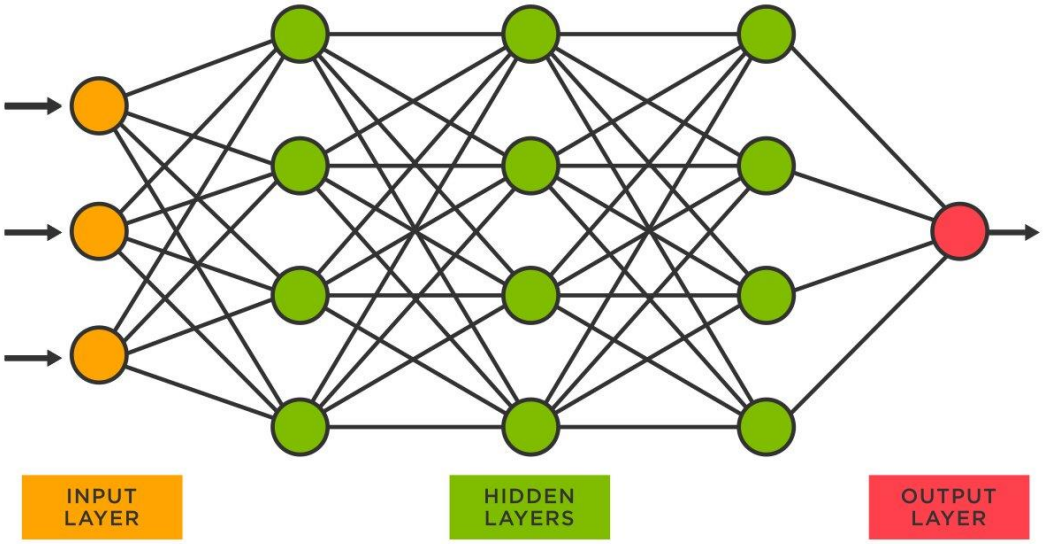


Ilustración 6: Ejemplo de arquitectura de una red neuronal

La idea es que la red neuronal aprenda sus propias características. Este modelo es la base del Deep Learning, que destaca en procesamiento de imágenes y lenguaje natural hoy en día. Permiten afrontar problemas no lineales sin características polinomiales.

3 GENERACIÓN DATASET

3.1 DESCRIPCIÓN

Para este trabajo disponemos del siguiente dataset con 14828 películas:

Tabla 1: Dataset

	imdb_id	titulo	sinopsis	generos	split	fuente
0	tt0057603	I tre volti della paura	Note: this synopsis is for the orginal Italian...	cult, horror, gothic, murder, atmospheric	train	imdb
1	tt1733125	Dungeons & Dragons: The Book of Vile Darkness	Two thousand years ago, Nhagruul the Foul, a s...	violence	train	imdb
2	tt0033045	The Shop Around the Corner	Matuschek's, a gift store in Budapest, is the ...	romantic	test	imdb
3	tt0113862	Mr. Holland's Opus	Glenn Holland, not a morning person by anyone'...	inspiring, romantic, stupid, feel-good	train	imdb
4	tt0086250	Scarface	In May 1980, a Cuban man named Tony Montana (A...	cruelty, murder, dramatic, cult, violence, atm...	val	imdb

1. IMDB_ID

   Id con el que podemos identificar la película en imdb.

2. TITULO

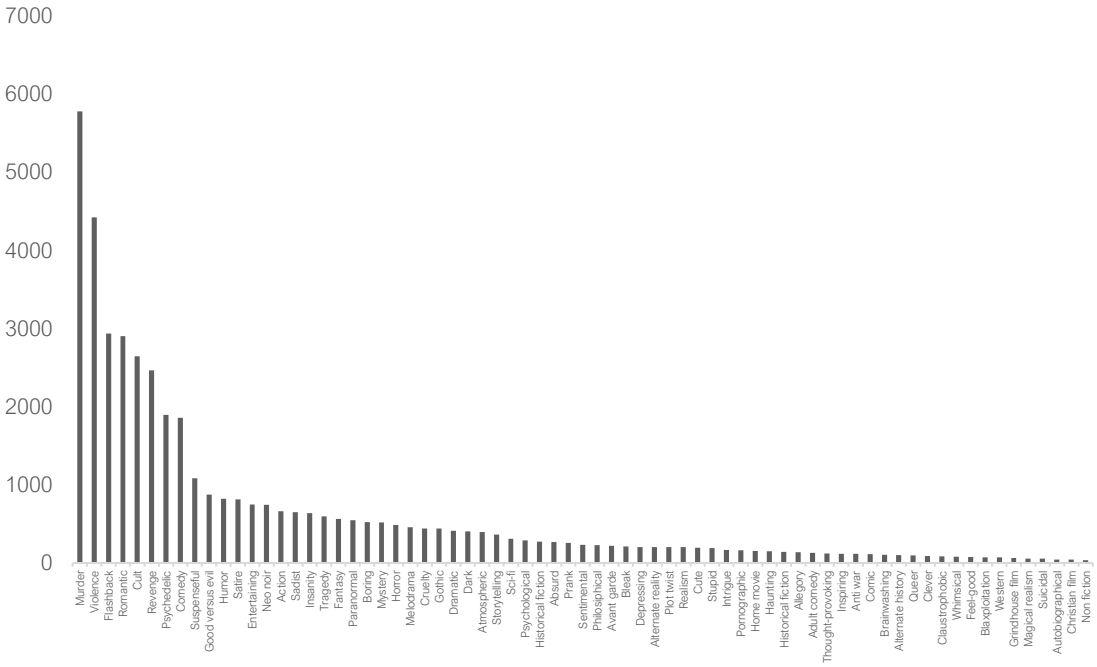
   Título de las películas.

3. SINOPSIS

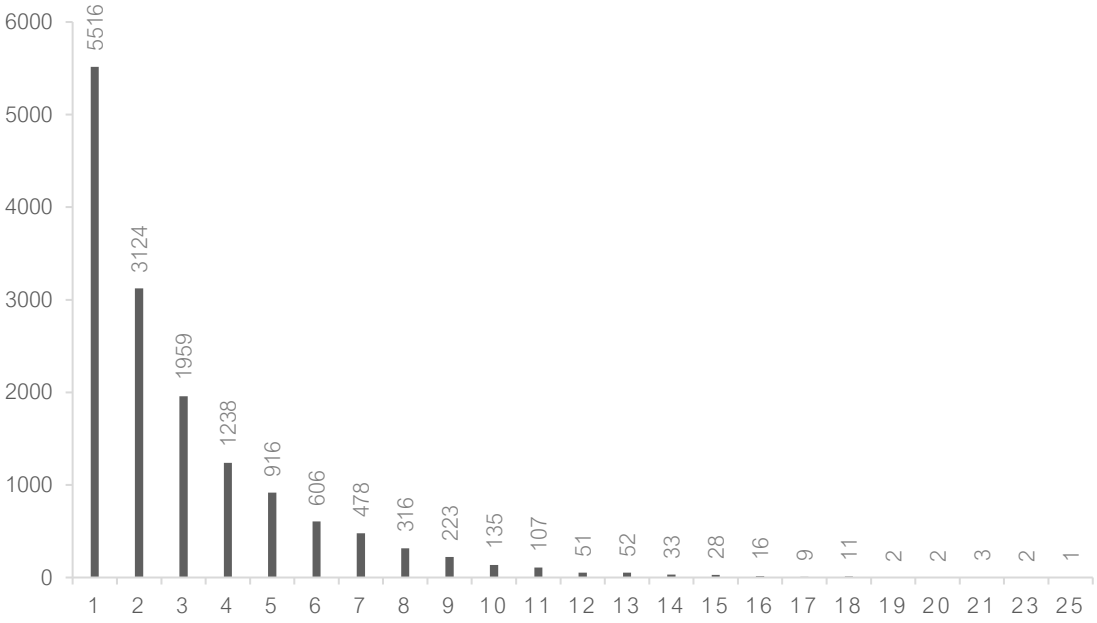
   Sinopsis de las películas, textos de relativa longitud.

4. GÉNEROS

   Géneros a los que pertenece cada película. Podemos encontrar 71 géneros diferentes.



Gráfica 1: Distribución de géneros



Gráfica 2: Distribución de cantidad de géneros por película

5. SPLIT

Segmentación train/validación/test propuesto.

6. FUENTE

Origen de la sinopsis: 28% imdb, 72% Wikipedia.

3.2 ELIMINACIÓN PELÍCULAS

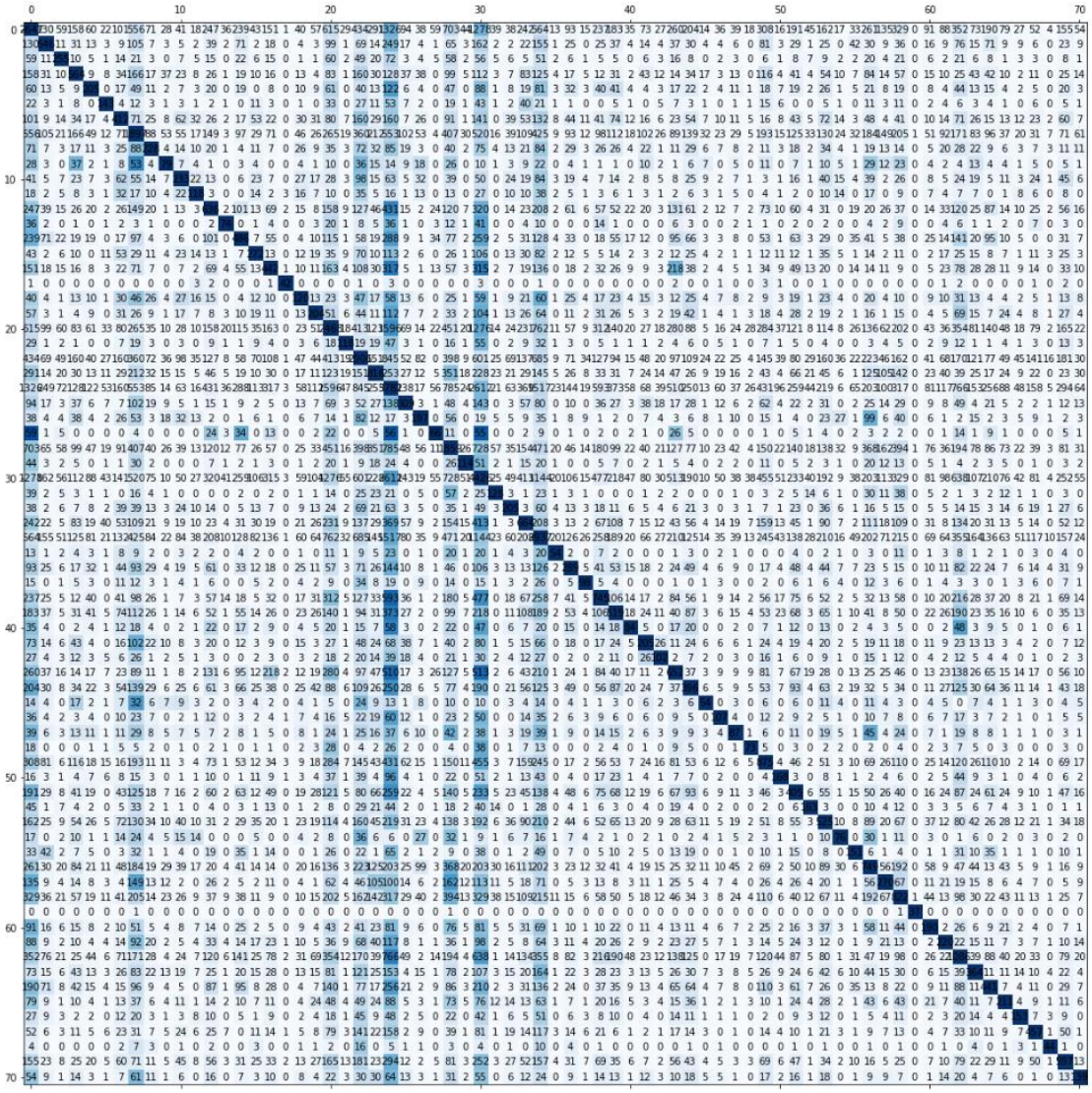
En el proceso de bow, nos hemos percatado de la existencia de palabras en otros idiomas. Puede darse el caso de nombres o títulos de películas en otros idiomas, pero observamos una cantidad inusual de palabras en castellano. Buscando en el dataset la palabra "es", frecuente en el castellano, identificamos 10 películas cuyas sinopsis estaban completamente en castellano. Las eliminamos del dataset.

3.3 CLUSTERING

Consideramos que 71 clases diferentes es excesivamente complejo tanto computacionalmente como a la hora de evaluación de modelos propuesta. Concretamente en el método de evaluación multiclase, donde codificaríamos cada combinación de clases diferentes a una clase, con 71 géneros tendríamos una cantidad de combinaciones cercana a la de películas. A mayores, muchos de los géneros son muy similares entre sí. Por todo ello, procedemos a hacer un clustering para reducir el número de clases.

3.3.1 DISTANCIA ENTRE GÉNEROS

En primer lugar, se han contado las veces en las que cada género aparece junto a los demás, obteniendo la siguiente matriz 71x71.



Gráfica 3: Matriz con cantidades absolutas de coincidencias por parejas de géneros

Después del uso de otras medidas, elegimos la siguiente ecuación para el cálculo de la distancia entre dos géneros.

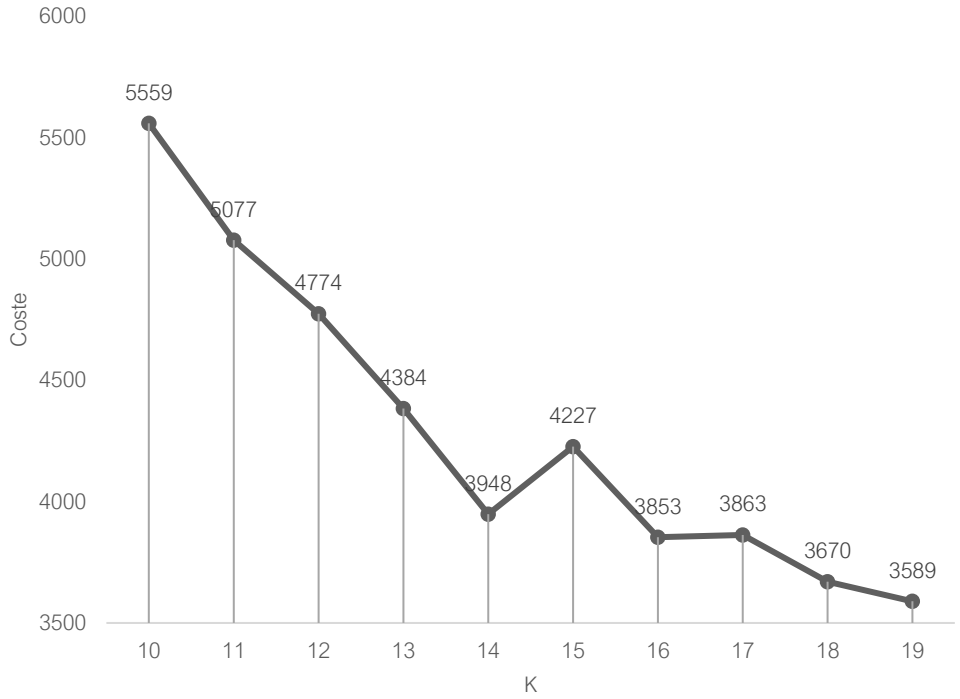
$$d(g_1, g_2) = 1 - \left(\frac{apariciones(g_1, g_2)^2}{apariciones(g_1) * apariciones(g_2)} \right)$$

Ecuación 20: Distancia pares de géneros

Equivaldría a la inversa de la agregación por producto de la cantidad de coincidencia relativa de los dos géneros. Con esta medida de distancia se normaliza el número de apariciones, ya que hay géneros con número de apariciones mayor que otros (véase Gráfica 1).

3.3.2 K-MEDOIDS

Aplicamos el algoritmo K-medoids con la distancia entre géneros establecida. Probamos para los valores de K entre 10 y 20. Visualizamos la función de coste para los diferentes k para elegir el valor óptimo.



Gráfica 4: Costes K-medoids

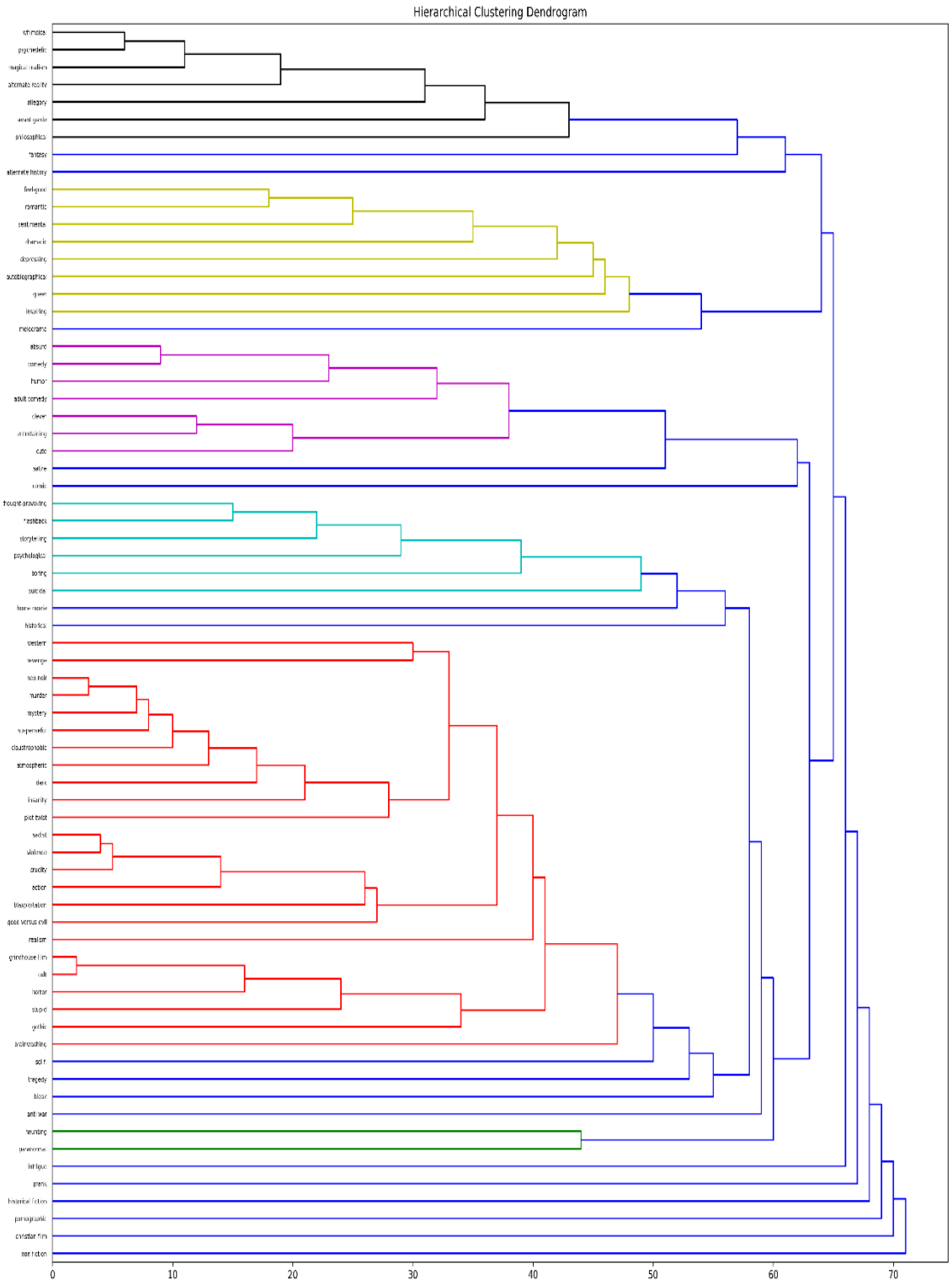
Nos quedamos con la clusterización de k=14 subóptima por la regla del codo y visualizamos las asociaciones.

Tabla 2: Clústers de géneros con K-medoids

	Lista de géneros
1	*[western]*
2	*[inspiring, cute, feel-good]*
3	*[historical, depressing, boring, realism, dramatic, sentimental]*
4	*[christian film]*
5	*[satire, entertaining, absurd, clever, prank, comedy, humor, adult comedy, suicidal, bleak]*
6	*[alternate reality, sci-fi, alternate history]*
7	*[claustrophobic, horror, gothic, grindhouse film, haunting]*
8	*[avant garde, psychedelic, cult, comic, pornographic, stupid, blaxploitation, fantasy, allegory, atmospheric, good versus evil, dark]*
9	*[thought-provoking, philosophical, psychological]*
10	*[non fiction]*
11	*[autobiographical]*
12	*[magical realism, whimsical]*
13	*[melodrama, home movie, romantic, queer, storytelling]*
14	*[cruelty, intrigue, mystery, paranormal, action, tragedy, murder, plot twist, insanity, brainwashing, sadist, violence, anti war, historical fiction, revenge, neo noir, flashback, suspenseful]*

3.3.3 CLUSTERING JERÁRQUICO

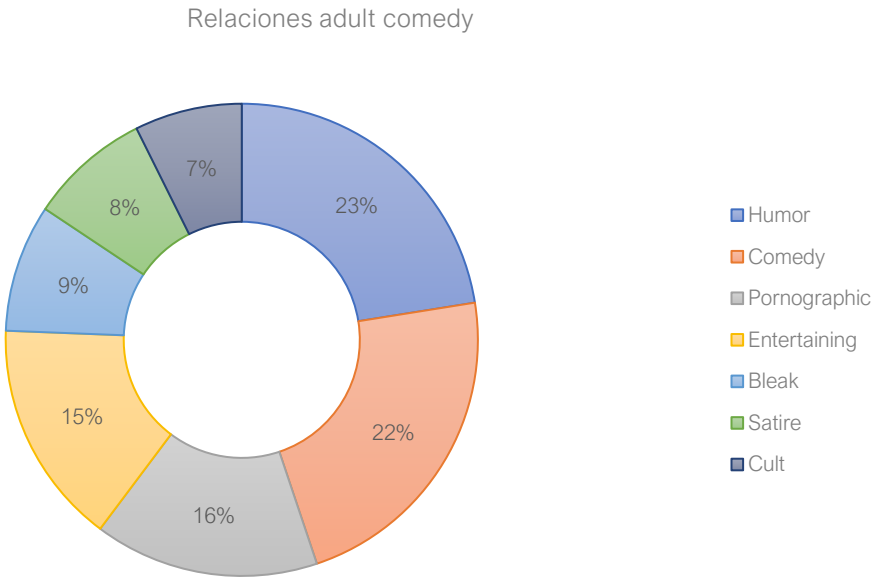
Probamos a hacer clustering jerárquico para poder visualizar las asociaciones. Como distancia entre clústers emplearemos single-link. Obtenemos secuencialmente las uniones, pero para poder ilustrarlas en un dendograma hacemos un pequeño script para crear un dataset donde las instancias son los géneros y a cada uno se le da una posición en el espacio vectorial (prácticamente equivalente) de tal forma que al hacer clustering jerárquico con la librería "*AgglomerativeClustering*" de sckikitlearn (véase 5.2.5) obtenemos los mismos resultados. Con esta librería obtenemos el dendograma:



Gráfica 5: Dendograma clustering jerárquico de los géneros

3.3.4 CLUSTERIZACIÓN MANUAL

A pesar de los ejercicios de clusterización, aunque tengan sentido en cuanto a las coincidencias de géneros en las películas, encontramos asociaciones que filmográficamente carecen de relación. En definitiva, procedemos a realizar una clusterización manual atendiendo a las dos clusterizaciones como referencia, a estadísticas individuales de cada género y a criterios cinematográficos.



Gráfica 6: Grafico de pastel de Adult Comedy

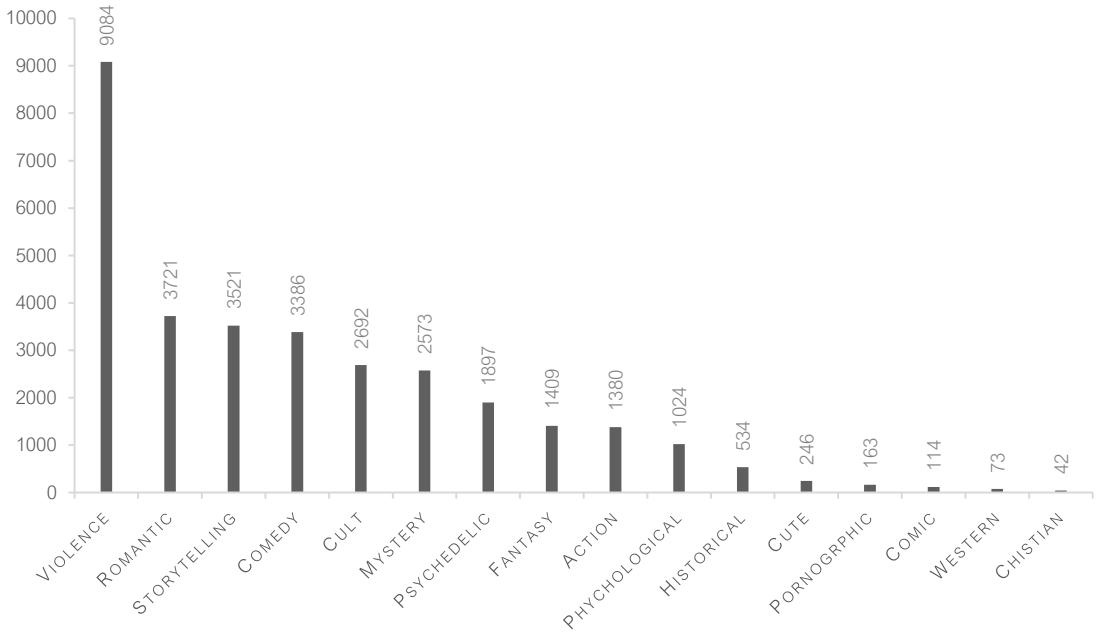
Como resultado obtenemos los siguientes 16 grupos (el representante es el primero de cada grupo):

Tabla 3: Clústers de géneros manual

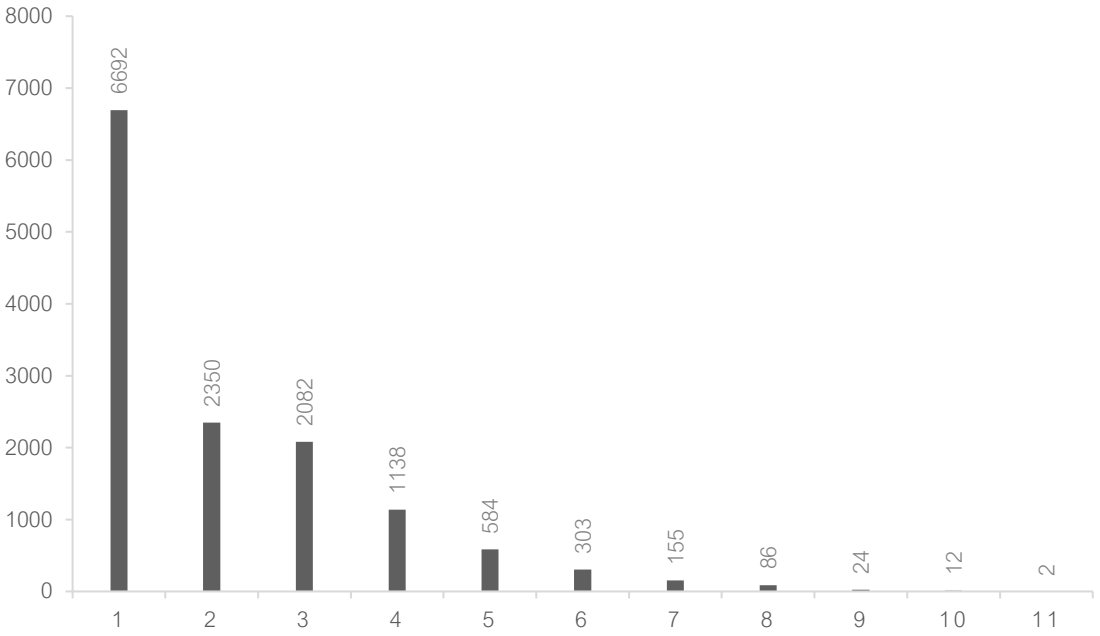
	Lista de géneros
1	*[comedy, satire, humor, adult comedy, prank, absurd, entertaining, clever, stupid]*
2	*[cult, grindhouse film, blaxploitation]*
3	*[thought-provoking, philosophical, psychological, brainwashing, allegory, plot twist, inspiring]*
4	*[flashback, storytelling, boring, home movie]*
5	*[magical realism, whimsical, alternate reality, sci-fi, alternate history, avant garde, fantasy, historical fiction]*
6	*[cute, feel-good]*
7	*[romantic, queer, sentimental, depressing, dramatic, melodrama]*
8	*[autobiographical, historical, realism, non fiction]*
9	*[murder, cruelty, insanity, sadist, violence, revenge, neo noir, haunting, tragedy, suicidal, anti war, horror, gothic]*
10	*[intrigue, mystery, suspenseful, atmospheric, dark, claustrophobic, paranormal, bleak]*
11	*[action, good versus evil]*
12	*[christian film]*
13	*[western]*
14	*[psychedelic]*
15	*[pornographic]*
16	*[comic]*

Modificamos el dataset juntando los géneros acordes a la última clusterización.

Estas son las nuevas distribuciones:



Gráfica 7: Distribución de géneros reducidos



Gráfica 8: Distribución de cantidad de géneros reducidos por película

3.4 CODIFICACIÓN SALIDA

Los géneros se codifican con la técnica OneHotEncoder [7]. En lugar de tener los géneros en una string, se codifican en una matriz donde cada columna representa un género, siendo 1 si está presente en la película o 0 en el caso contrario.

3.5 VALIDACIÓN CRUZADA

Dividimos el dataset en 7 subgrupos de tamaño similar que mantienen la distribución de los géneros (estratificación) para validación cruzada (véase 2.7.2).

4 PROPUESTA

El objetivo de este trabajo es evaluar las técnicas para abordar el problema de clasificación multi-etiqueta de texto y conseguir el mejor modelo para predecir los géneros de las películas.

4.1 CLASIFICACIÓN MULTI-ETIQUETA

Cuando un dataset este compuesto por instancias con más de dos clases, se dice que es un problema multi-clase. Si, además, cada instancia está asociada a más de una clase a la vez, tenemos un problema multi-etiqueta.

Por ejemplo, si queremos clasificar imágenes de fruta, sería un problema multiclase porque tendríamos diferentes clases (naranjas, manzanas, plátanos) pero que serían excluyentes entre sí, una fruta solo puede pertenecer a una clase. Por otro lado, si queremos clasificar noticias, cada noticia puede pertenecer a más de una clase a la vez (religión, política, educación, financiera) por lo que tendríamos un problema multi-etiqueta.

4.1.1 CLASIFICADORES NATIVOS

A pesar de no ser la norma, existen diversos algoritmos que aceptan la clasificación multi-etiqueta nativamente, o bien por defecto o bien están adaptados.

Entre otros podemos encontrar:

- Redes neuronales (véase 2.12)
- Árbol de decisiones (véase 2.11.3)
- MlKNN, un KNN adaptado (véase 2.11.2)

4.1.2 APROXIMACIÓN CON ENSEMBLES

Se pueden utilizar ensembles cuyo clasificador base acepte la multi-etiqueta.

Algoritmos:

- Random forest (vease 2.11.4)

- Adaboost (vease 2.11.5)

4.1.3 ALGORITMOS MULTICLASE ADAPTADOS

Con este método, cualquier clasificador multi-clase podrá ser empleado.

4.1.3.1 TRANSFORMACIÓN BINARIA

Esta es la técnica más sencilla, que básicamente trata cada etiqueta como un problema de clasificación binaria por separado y luego lo integra. Tendremos tantos clasificadores como etiquetas.

4.1.3.2 CADENA CLASIFICADORES

En esta técnica, se crea una cadena de tantos clasificadores como etiquetas haya. Secuencialmente se ejecutará cada clasificador añadiendo en cada iteración de la cadena más etiquetas. Empezará por una etiqueta, el siguiente clasificador obtendrá la clasificación del anterior y añadirá otra etiqueta y así sucesivamente. Es muy similar a la transformación binaria, pero se preserva la correlación de las etiquetas.

4.1.3.3 TRANSFORMACIÓN MULTICLASE

Se transforma el problema a multi-clase, asignando cada combinación de clases a una clase diferente. Dependiendo del número de clases, puede resultar demasiado complejo.

4.1.4 MÉTRICAS EVALUACIÓN MULTI-ETIQUETA

4.1.4.1 MÉTODOS DE EVALUACIÓN BASADOS EN EJEMPLOS

- EMR

$$EMR = \frac{1}{n}\sum_{i=1}^{n}[I(y^{(i)} == y'^{(i)})]$$

Ecuación 21: EMR

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

- Hamming Loss

$$Hamming\ Loss = \frac{1}{nL}\sum_{i=1}^{n}\sum_{j=1}^{L}\left[I(y^{(i)} \neq y'^{(i)})\right]$$

Ecuación 22: Hamming Loss

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

- Accuracy

$$Accuracy(A) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y^{(i)} \cap y'^{(i)}\right|}{\left|y^{(i)} \cup y'^{(i)}\right|}$$

Ecuación 23: Ejemplo accuracy

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

- Precisión

$$Precision(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{\left|y^{(i)} \cap y'^{(i)}\right|}{\left|y'^{(i)}\right|}$$

Ecuación 24: Ejemplo precisión

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

- Recall

$$Recall(P) = \frac{1}{n}\sum_{i=1}^{n}\frac{|y^{(i)} \cap y'^{(i)}|}{|y^{(i)}|}$$

Ecuación 25: Ejemplo recall

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

- Fscore

$$F_{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Ecuación 26: Ejemplo Fscore

4.1.4.2 MÉTODOS DE EVALUACIÓN BASADOS EN ETIQUETAS

- Macro Accuracy

$$\gamma - Accuracy(A^{j}_{macro}) = \frac{\sum_{i=1}^{n}[y_j^{(i)} \cap y_j'^{(i)}]}{\sum_{i=1}^{n}[y_j^{(i)} \cup y_j'^{(i)}]}$$

$$Accuracy_{Macro} = \frac{1}{k}\sum_{j=1}^{k}(A^{j}_{macro})$$

Ecuación 27: Macro Accuracy

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

k -> número de clases

- Macro Precisión

$$\gamma - Precision\left(P_{macro}^{j}\right) = \frac{\sum_{i=1}^{n}\left[y_j^{(i)} \cap y_j'^{(i)}\right]}{\sum_{i=1}^{n}\left[y_j'^{(i)}\right]}$$

$$Precision_{Macro} = \frac{1}{k}\sum_{j=1}^{k}\left(P_{macro}^{j}\right)$$

Ecuación 28: Macro precisión

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

k -> número de clases

- Macro Recall

$$\gamma - Recall\left(R_{macro}^{j}\right) = \frac{\sum_{i=1}^{n}\left[y_j^{(i)} \cap y_j'^{(i)}\right]}{\sum_{i=1}^{n}\left[y_j^{(i)}\right]}$$

$$Recall_{Macro} = \frac{1}{k}\sum_{j=1}^{k}\left(R_{macro}^{j}\right)$$

Ecuación 29: Macro recall

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

k -> número de clases

- Macro Fscore

$$F_{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Ecuación 30: Macro Fscore

- Micro Accuracy

$$Accuracy_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}\left[y_j^{(i)} \cap y_j'^{(i)}\right]}{\sum_{j=1}^{k}\sum_{i=1}^{n}\left[y_j^{(i)} \cup y_j'^{(i)}\right]}$$

Ecuación 31: Micro Accuracy

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

k -> número de clases

- Micro Precisión

$$Precision_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}\left[y_j^{(i)} \cap y_j'^{(i)}\right]}{\sum_{j=1}^{k}\sum_{i=1}^{n}\left[y_j'^{(i)}\right]}$$

Ecuación 32: Micro precisión

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

k -> número de clases

- Micro Recall

$$Precision_{micro} = \frac{\sum_{j=1}^{k}\sum_{i=1}^{n}\left[y_j^{(i)} \cap y_j'^{(i)}\right]}{\sum_{j=1}^{k}\sum_{i=1}^{n}\left[y_j^{(i)}\right]}$$

Ecuación 33: Micro recall

n -> número de ejemplos

$y^{(i)}$ -> etiquetas positivas

$y'^{(i)}$ -> etiquetas predichas

∩ -> operador lógico AND

∪ -> operador lógico OR

k -> número de clases

- Micro Fscore

$$F_{score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Ecuación 34: Micro Fscore

5 IMPLEMENTACIÓN

5.1 LENGUAJE Y ENTORNO DE PROGRAMACIÓN

El lenguaje de programación elegido es Python dada nuestra familiarización con el lenguaje y machine learning en las asignaturas de la mención de computación de la carrera de ingeniería informática. A pesar de la perfecta posibilidad de usar otros lenguajes como java, c, Matlab, etc., Python alberga una inmensa cantidad de librerías y algoritmos para esta tarea. Dada su popularidad, las librerías de ciencia de datos están actualizadas y resultan muy eficientes. A mayores, hay un gran soporte de la comunidad.

A su vez, hemos utilizado el entorno jupyter notebook de anaconda, por el dinamismo e independencia que ofrecen las celdas.

5.2 LIBRERÍAS

5.2.1 NUMPY

Numpy es un paquete de código abierto fundamental para la computación científica en Python. Es el estándar universal para trabajar con datos numéricos, especializada en estructuras de datos como matrices (y derivados de ello) y todo tipo de operaciones relacionadas [8].

5.2.2 MATPLOTLIB

Matplotlib es una librería de código abierto de generación de todo tipo de gráficos. Es de bajo nivel por lo que se pueden modificar muchos parámetros para obtener la mejor representación posible [9].

5.2.3 PANDAS

Pandas es una librería de código abierto que extiende numpy. Se emplea para manipulación y análisis de datos, concretamente, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. Dataframe es la estructura más común [10].

5.2.4 NLTK

Nltk es una librería de código abierto especializada en datos de lenguaje natural. Introduce interfaces amigables para más de 50 recursos léxicos como WordNet (diccionarios), tokenizations (separación de palabras), stemming (lexemas)… [11]

5.2.5 SCIKIT-LEARN

Scikit-learn o sklearn es una librería de código abierto para machine learning. Es sin duda la más popular y la que más herramientas ofrece. La componen una extensa variedad de algoritmos para todo el proceso de preprocesamiento de datos, creación de modelos y optimización de hiperparámetros, que siguen un patrón de uso similar para su entendimiento [12].

5.2.6 SCIKIT-MULTILEARN

Scikit -multilearn o skmultilearn es una librería de código abierto que extiende Scikit-learn y está especializada en algoritmos de machine learning en casos de multi-etiqueta. Podemos encontrar clasificadores, procesamiento de etiquetas… [13]

5.2.7 TENSORFLOW

Tensorflow es una librería de código abierto desarrollado por Google para deep learning. Cuenta con un ecosistema de herramientas y funcionalidades para machine learning, pero su mayor valor son las redes neuronales [14].

5.3 BAG OF WORDS

Con el dataset preprocesado previamente, procesamos las sinopsis con minería de texto. Aplicando "*CountVectorizer*" de sklearn (véase 5.2.5), vemos que el corpus sin las películas en español está compuesto por 122729 palabras diferentes.

5.3.1 NORMALIZACIÓN

Hacemos un recorrido a nivel de carácter de todos los textos:

- Eliminamos el carácter “ ’ ” frecuente en el inglés.
- Reemplazamos caracteres especiales como palabras con tildes o el carácter “ñ” por su equivalente (pueden estar presente en nombres en castellano, etc.).
- Convertimos todo a minúsculas.
- Reemplazamos todo carácter que no sea una letra por espacios.

Después de esto nos quedamos con 132300 palabras, esto es, aumentamos las palabras un 8%. Esto se debe a que los caracteres que no son letras como guiones o números separan palabras, aumentando el número.

5.3.2 ELIMINACIÓN STOP-WORDS

Filtramos las palabras más habituales del inglés con el parámetro “stop_words = 'english'” de “*CountVectorizer*” de sklearn (véase 5.2.5). Con esto nos quedamos con 131988 tokens, o sea, hemos reducido el corpus menos de 1%, pero la frecuencia de estas palabras era mucho mayor en el texto.

5.3.3 REDUCCIÓN DE PALABRAS A SU RAÍZ

Reducimos las palabras a su lexema con stemming, ya que es más agresivo que lemmatization. Lo hacemos con la librería “*SnowballStemmer*” con el diccionario “*english*” de la propia librería. Las palabras del corpus resultante son 88615, es decir, se han reducido las palabras un 33%.

5.3.4 SINÓNIMOS

Con el diccionario “wordnet” de ntlk (véase 5.2.4), hemos agrupado palabras que son sinónimos como “car” o ”vehicle”. Como fruto de esto reducimos 3472 palabras y obtenemos 85143 en total.

5.3.5 ELIMINACIÓN DE TÉRMINOS CON FRECUENCIAS MUY ALTAS O BAJAS

Por último, filtrando las palabras con menos frecuencia ligeramente, reducimos el corpus drásticamente. Hay un gran número de palabras mal escritas, nombres propios o palabras en otros idiomas que pasan todos los filtros anteriores y representan un gran porcentaje del corpus sin tener un valor real. Solo filtrando los términos que aparecen al menos en cuatro textos diferentes,

conseguimos solo 26107 palabras, reduciendo el corpus un 70%. El número de palabras a filtrar es objeto de optimización en la minería. Utilizaremos el parámetro "*max_features*" de "*CountVectorizer*" de sklearn (véase 5.2.5).

5.3.6 REDUCCIÓN

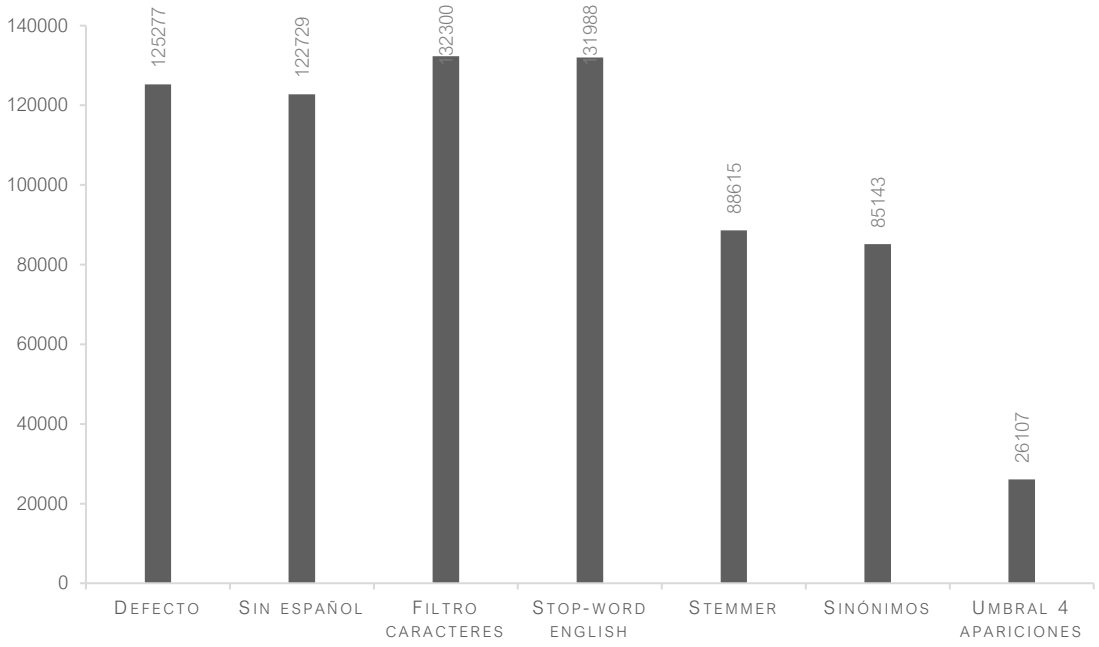


Ilustración 7: Reducciones del corpus

5.3.7 PESOS

Para asignar pesos a cada elemento del corpus usamos FT-IDF (véase 2.9.5.4). Utilizaremos "*TfidfTransformer*" de sklearn (véase 5.2.5). En última instancia, conseguimos una matriz dispersa lista para ser clasificada.

5.4 MÉTRICAS DE EVALUACIÓN

Considerando las métricas de avaluación propuestas (véase 4.1.4), seleccionamos las más apropiadas para el problema.

5.4.1 EMR

EMR sería la equivalente a una evaluación multiclase, donde la predicción tiene que coincidir totalmente para contar como acierto, sin atender a los aciertos parciales. No es la métrica de evaluación más representativa, pero es interesante.

5.4.2 ACCURACY, PRECISIÓN, RECALL, FSCORE

La mayoría de los valores de la matriz de salida son 0 dada la codificación (véase 3.4) y la cantidad de géneros por película (véase Gráfica 8).

Por lo tanto, los negativos son más numerosos que los positivos. Sin embargo, intuitivamente no consideramos un acierto los verdaderos negativos. En consecuencia, consideramos que el accuracy es optimista y nos centramos en la precisión y en el recall. Ambos nos parecen válidos, así que usamos la métrica Fscore que es la media armónica de ambos.

5.4.3 EJEMPLO, MACRO Y MICRO

Las tres métricas aportan valor relevante del método. Como métrica definitiva, creemos que el micro Fscore es la que mejor resume el nivel de acierto del modelo, contabilizando todos los verdaderos positivos de todos los textos sin sesgos de ejemplos ni etiquetas. De todas formas, se calculará el Fscore de ejemplos y del macro como métricas a valorar.

5.5 CLASIFICADORES

Naïve Bayes suele ser el clasificador empleado tradicionalmente para la clasificación de texto. Siempre que podamos aplicaremos el algoritmo Naïve Bayes (véase 2.11.1). Utilizaremos "*naive_bayes*" de sklearn (véase 5.2.5) para su ejecución con sus variantes. Seguiremos también las estrategias de la multi-etiqueta (véase 4.1).

5.5.1 TRANSFORMACIÓN BINARÍA

Utilizando "*MultiOutputClassifier*" de sklearn (véase 5.2.5), podemos entrenar un clasificador base para cada etiqueta y luego agregar todas las predicciones. El clasificador base elegido es Naïve Bayes (véase 2.11.1) y a pesar de que se intuye que el Naïve Bayes complementario pueda ser el mejor, probamos también el gaussiano, el multinomial y el bernoulli.

Tabla 4: Resultados transformación binaria Naïve Bayes

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
Naive bayes	Gaussian	2000	0,386	0,474	0,272	0,487	0,416	0,502	0,02	0,036
	Multinomial	5000	0,407	0,436	0,127	0,153	0,444	0,47	0,208	0,223
	Bernoulli	15000	0,414	0,451	0,27	0,339	0,416	0,457	0,103	0,124
	Complement	5000	0,474	0,566	0,278	0,375	0,528	0,608	0,083	0,125

5.5.1.1 OBSERVACIONES

El mejor clasificador de Naïve Bayes es el complementario, obteniendo los mejores resultados tanto en micro, como en macro y a nivel de ejemplos. Resulta el más conveniente para clasificación de textos. El EMR se resiente en comparación con el multinomial y Bernoulli. Descartamos el método gaussiano, ya que no encaja en el tipo de problema.

5.5.2 CADENA DE CLASIFICADORES

En este caso, el primer clasificador se entrena solo con los datos de entrada y luego cada clasificador siguiente se entrena con el espacio de entrada y con todos los clasificadores anteriores de la cadena. El clasificador base elegido es Naïve Bayes (véase 2.11.1). La implementación la haremos con "*ClassifierChain*" de skmultilearn (véase 5.2.6).

Tabla 5: Resultados cadena de clasificadores Naïve Bayes

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
Naive bayes	Multinomial	5000	0,408	0,434	0,127	0,153	0,451	0,474	0,212	0,226
	Bernoulli	15000	0,411	0,448	0,269	0,337	0,415	0,456	0,103	0,124
	Complement	5000	0,457	0,598	0,216	0,362	0,514	0,636	0,165	0,246

5.5.2.1 OBSERVACIONES

Nuevamente, el mejor clasificador es el complementario. Los resultados, aunque similares, son ligeramente peores que en la transformación binaria.

5.5.3 TRANSFORMACIÓN MULTICLASE

En este caso, transformamos el problema en uno multiclase, con un clasificador multiclase que se entrena con todas las combinaciones de etiquetas únicas encontradas en los datos de entrenamiento. En total tendremos 1037 clases diferentes. La nueva codificación y aplicación la hacemos con "*LabelPowerset*" de skmultilearn (véase 5.2.6). El clasificador elegido es Naïve Bayes (véase 2.11.1).

Tabla 6: Resultados transformación multiclase con Naïve Bayes

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
Naive bayes	Multinomial	2000	0,416	0,422	0,118	0,123	0,485	0,496	0,231	0,241
	Complement	15000	0,422	0,808	0,208	0,75	0,498	0,815	0,196	0,684

5.5.3.1 OBSERVACIONES

En la línea con los resultados anteriores, el complementario es el que obtiene mejor micro Fscore. Los resultados son parecidos al método de cadena de clasificadores. Cabe destacar que hemos obtenido el mejor EMR hasta el momento en el caso del multinomial. Tiene sentido ya que la clasificación se ha hecho de manera multiclase con combinaciones únicas de géneros y es lo que intenta maximizar, la misma combinación.

5.5.4 CLASIFICADORES MULTI-ETIQUETA

En esta sección aplicaremos algoritmos que nativamente acepten más de una clase simultáneamente. La selección de los hiperparámetros para cada caso se ha hecho con "*GridSearchCV*" de sklearn (véase 5.2.5).

5.5.4.1 MLKNN

Probamos MlKNN, una modificación del algoritmo knn (véase 2.11.2). Utilizaremos la implementación "*MLkNN*" de skmultilearn (véase 5.2.6). El mejor resultado ha sido obtenido con k = 5.

Tabla 7: Resultados clasificación MlKNN

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
	Mlknn	2000	0,399	0,55	0,177	0,33	0,43	0,565	0,136	0,1

5.5.4.2 OBSERVACIONES

Los resultados son peores que con Naïve Bayes. Este modelo es muy lento de entrenar. Knn no es muy apropiado para clasificación de texto.

5.5.4.3 ÁRBOLES DE DECISIÓN

También probamos los árboles de decisión (véase 2.11.3). Utilizaremos "*DecisionTreeClassifier*" de sklearn (véase 5.2.5). Los mejores parámetros son "*criterion = 'gini', max_depth = None*".

Tabla 8: Resultados clasificación arboles de decisión

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
	DecisionTreeClassifier	10000	0,389	0,975	0,189	0,965	0,433	0,962	0,111	0,937

5.5.4.4 OBSERVACIONES

Análogamente a knn, los resultados son peores que con Naïve Bayes. En este caso observamos como los resultados del train son cercanos a la perfección. Tiene un bias alto, no generaliza bien. No es un clasificador conveniente para clasificación de textos.

5.5.5 ENSEMBLES

Probamos aproximaciones con ensembles.

5.5.5.1 RANDOM FOREST

El primer ensemble que probamos es Random Forest (véase 2.11.4), con "*RandomForestClassifier*" de sklearn (véase 5.2.5). Los mejores parámetros son "*n_estimators=10, max_depth=None, min_samples_split=2*".

Tabla 9: Resultados clasificación random forest

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
	RandomForest	5000	0,377	0,922	0,119	0,865	0,396	0,908	0,168	0,815

5.5.5.2 OBSERVACIONES

Al igual que los árboles de decisión, los resultados del train son muy altos comparando con los del test, tiene bias alto. No es un clasificador conveniente para clasificación de textos.

5.5.5.3 ADABOOST

El segundo ensemble es un Adaboost (véase 2.11.5) con "*AdaBoostClassifier*" de sklearn (véase 5.2.5). El clasificador base es un árbol de decisiones con los parámetros "*criterion='gini', max_depth = 39*".

Tabla 10: Resultados clasificación Adaboost

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
DecisionTreeClassifier	AdaBoostClassifier	10000	0,395	0,915	0,186	0,879	0,447	0,881	0,139	0,788

5.5.5.4 OBSERVACIONES

Los resultados son ligeramente mejor que en el caso de un árbol de decisión único. Una vez más el bias es alto. No es un clasificador conveniente para clasificación de textos.

5.5.6 REDES NEURONALES

Para las redes neuronales (véase 2.12) hemos utilizado la librería tensorflow (véase 5.2.7). Se opta por el modelo de red neuronal secuencial, el básico para crear una red simple, implicando que se añadirán capas al modelo de una forma lineal, desde la entrada hasta la salida.

Aparte de la capa de entrada con tantas neuronas como atributos del corpus, tenemos dos capas ocultas, una con 128 y otra con 64 neuronas. La activación de las neuronas de las capas intermedias es con la función de activación ReLU porque es la activación por defecto que se suele usar. En la capa de salida, se suele usar "*softmax*" como activación haciendo que la suma de todas las neuronas sean 1, maximizando así la probabilidad de una sola clase. Para problemas multi-etiqueta, utilizamos la activación "*sigmoid*" para que cada neurona pueda tener una probabilidad de activarse independientemente de las demás. En esta última capa tendremos 16 neuronas, una para cada clase.

En la clasificación multi-etiqueta formulamos la función objetivo como un clasificador binario en el que cada neurona de la capa de salida es responsable de la clasificación de una clase frente a todas. Es por esto por lo que como "*binary_crossentropy*" es adecuado para la clasificación binaria, se utiliza como loss en clasificación multi-etiqueta. El optimizador será "*adam*".

Entrenaremos el modelo con 100 épocas y un batch_size de 64 y a la hora de obtener las predicciones, tendremos que aplicar un umbral para clasificar en 1 o en 0. El mejor umbral ha sido 0.3.

Tabla 11: Resultados clasificación red neuronal

Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
			Test	Train	Test	Train	Test	Train	Test	Train
	RedNeuronal	5000	0,466	0,973	0,281	0,963	0,508	0,98	0,12	0,931

5.5.6.1 OBSERVACIONES

A pesar de que la diferencia del train y test es alta, los resultados de test son buenos, muy parecidos a los de Naïve Bayes complementario. Concluimos que la red neuronal se podría valer de muchos más datos de entrenamiento para conseguir mejores predicciones y resultados train más parejos a los de test. Además, el resultado del macro Fscore es el mejor.

5.6 RESULTADOS

Tabla 12: Resultados completos

	Clasificación multi-etiqueta	Clasificador	Max features bow	Micro Fscore		Macro Fscore		Example Fscore		Emr	
				Test	Train	Test	Train	Test	Train	Test	Train
Transformación binaria	Naive bayes	Gaussian	2000	0,386	0,474	0,272	0,487	0,416	0,502	0,02	0,036
		Multinomial	5000	0,407	0,436	0,127	0,153	0,444	0,47	0,208	0,223
		Bernoulli	15000	0,414	0,451	0,27	0,339	0,416	0,457	0,103	0,124
		Complement	5000	0,474	0,566	0,278	0,375	0,528	0,608	0,083	0,125
Cadena clasificadores	Naive bayes	Multinomial	5000	0,408	0,434	0,127	0,153	0,451	0,474	0,212	0,226
		Bernoulli	15000	0,411	0,448	0,269	0,337	0,415	0,456	0,103	0,124
		Complement	5000	0,457	0,598	0,216	0,362	0,514	0,636	0,165	0,246
Multiclase		Multinomial	2000	0,416	0,422	0,118	0,123	0,485	0,496	0,231	0,241
		Complement	15000	0,422	0,808	0,208	0,75	0,498	0,815	0,196	0,684
Clasificadores nativos		Mlknn	2000	0,399	0,55	0,177	0,33	0,43	0,565	0,136	0,1
		DecisionTreeClassifier	10000	0,389	0,975	0,189	0,965	0,433	0,962	0,111	0,937
		RedNeuronal	5000	0,466	0,973	0,281	0,963	0,508	0,98	0,12	0,931
Ensembles		RandomForest	5000	0,377	0,922	0,119	0,865	0,396	0,908	0,168	0,815
	DecisionTreeClassifier	AdaBoostClassifier	10000	0,395	0,915	0,186	0,879	0,447	0,881	0,139	0,788

5.7 ANÁLISIS

Observamos las siguientes condiciones:

- Los valores del fscore a nivel de ejemplos suelen ser mayores que los del micro.

  Viendo ejemplos en particular, observamos como los ejemplos acertados completamente (EMR) suelen tener pocos géneros, uno o dos como máximo. Esto se traduce a que con solo uno o dos positivos verdaderos obtendríamos el 100% de precisión y recall del ejemplo. En otras palabras, esos verdaderos positivos son más "productivos".
- Los valores del fscore a nivel macro suelen ser considerablemente menores que los del micro.

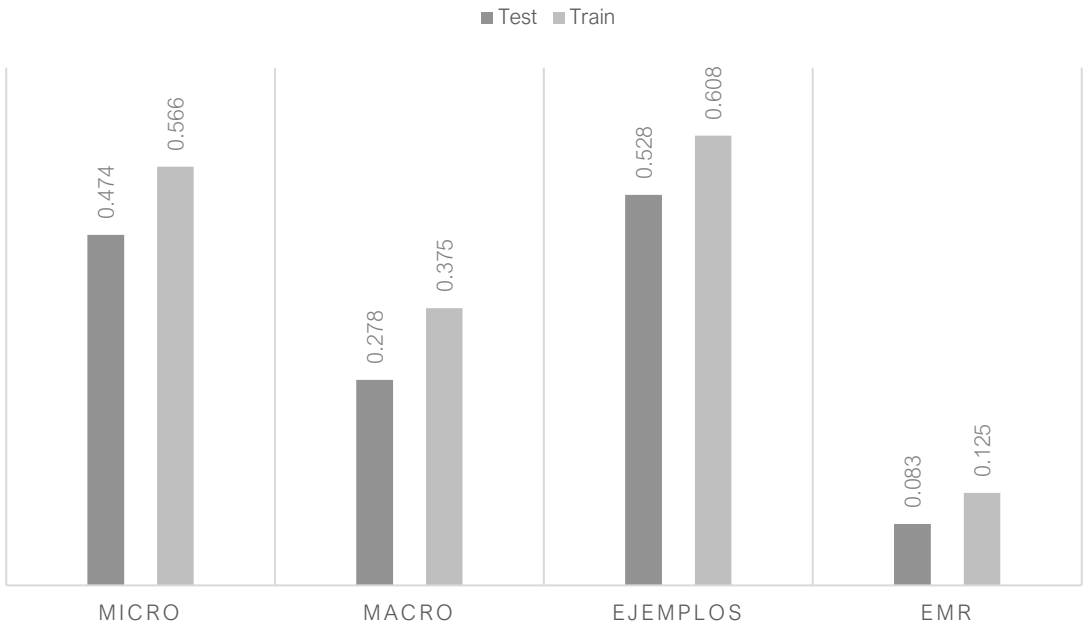
  Los géneros con mayor precisión y recall son los que más apariciones tienen, esto se entiende como consecuencia de maximización de los verdaderos positivos. En contraposición, los géneros menos frecuentes no están tan optimizados.
- Se puede observar que, en líneas generales, el recall tiende a ser ligeramente superior a la precisión. A cambio de predecir correctamente más clases, obtenemos más falsos positivos.

  En el caso de las redes neuronales, ha sido al contrario: bajamos el umbral a 0.3 para poder aumentar el recall, bajando la precisión y obteniendo mejor Fscore.

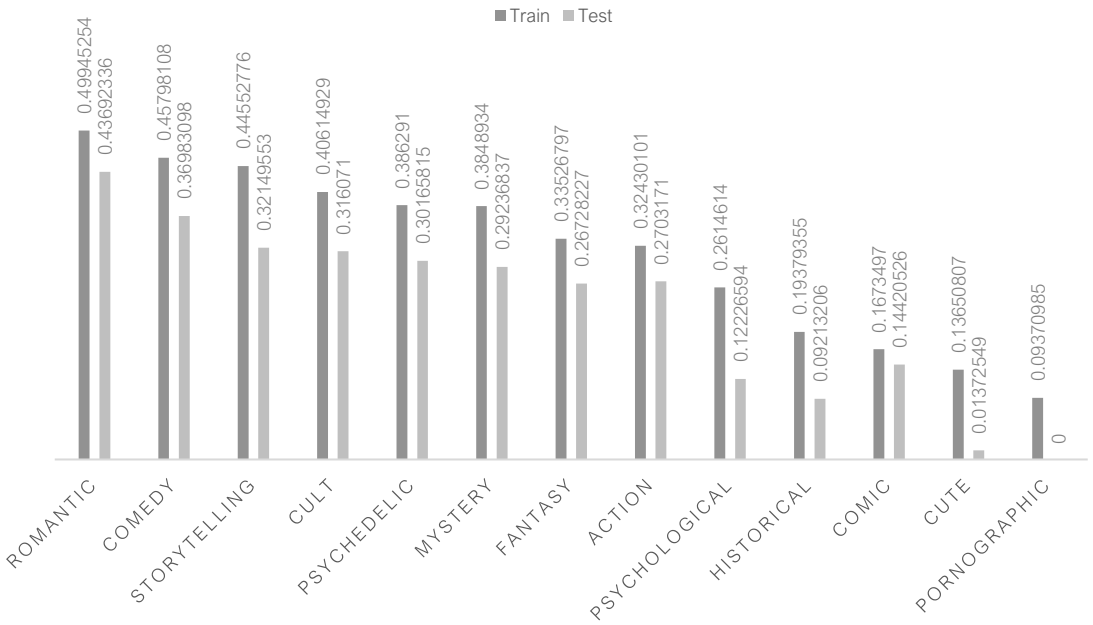
El mejor micro Fscore (0.474) ha sido obtenido con el clasificador Naïve Bayes complementario, empleando la técnica transformación binaria. Con un resultado cercano, la red neuronal ha conseguido 0.466, obteniendo también el mejor macro Fscore. El mejor EMR ha sido resultado de Naïve Bayes multinomial, con la transformación multiclase. Los clasificadores basados en KNN y árboles de decisión no han logrado un buen rendimiento, siendo además los más costosos de entrenar.

5.7.1 ANÁLISIS MEJORES RESULTADOS

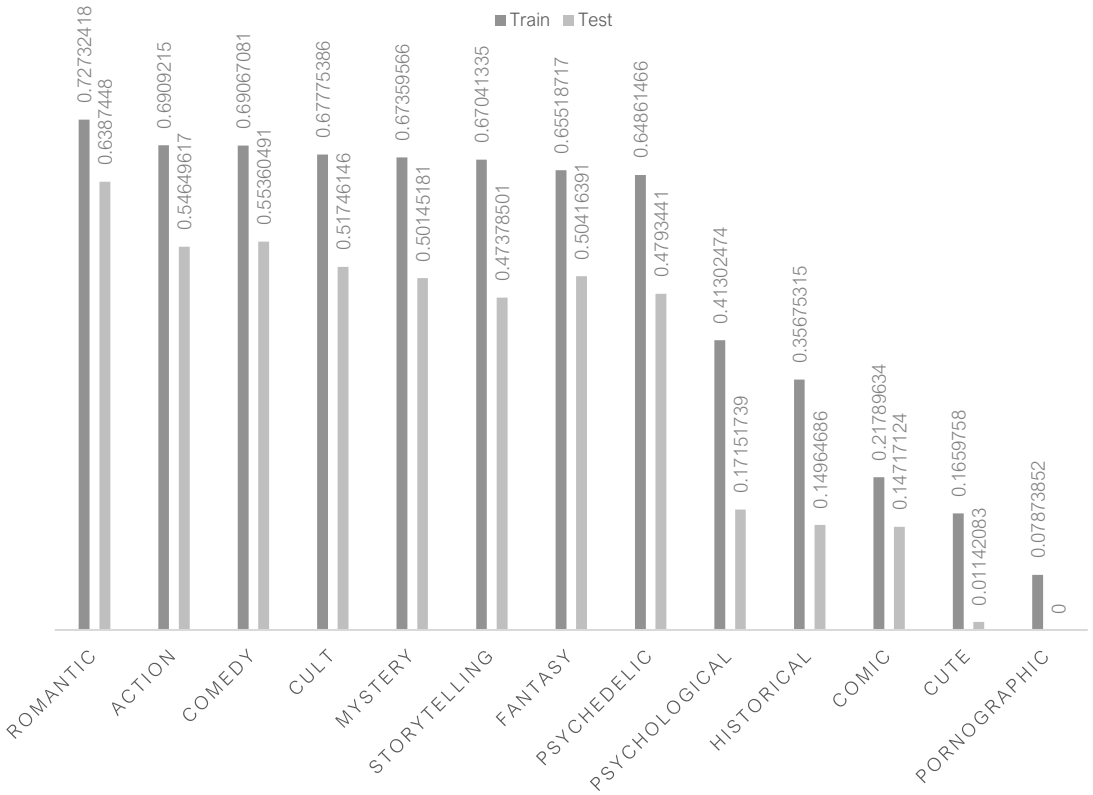
5.7.1.1 NAÏVE BAYES COMPLEMENTARIO CON TRANSFORMACIÓN BINARIA



Gráfica 9: Resultados NB complementario

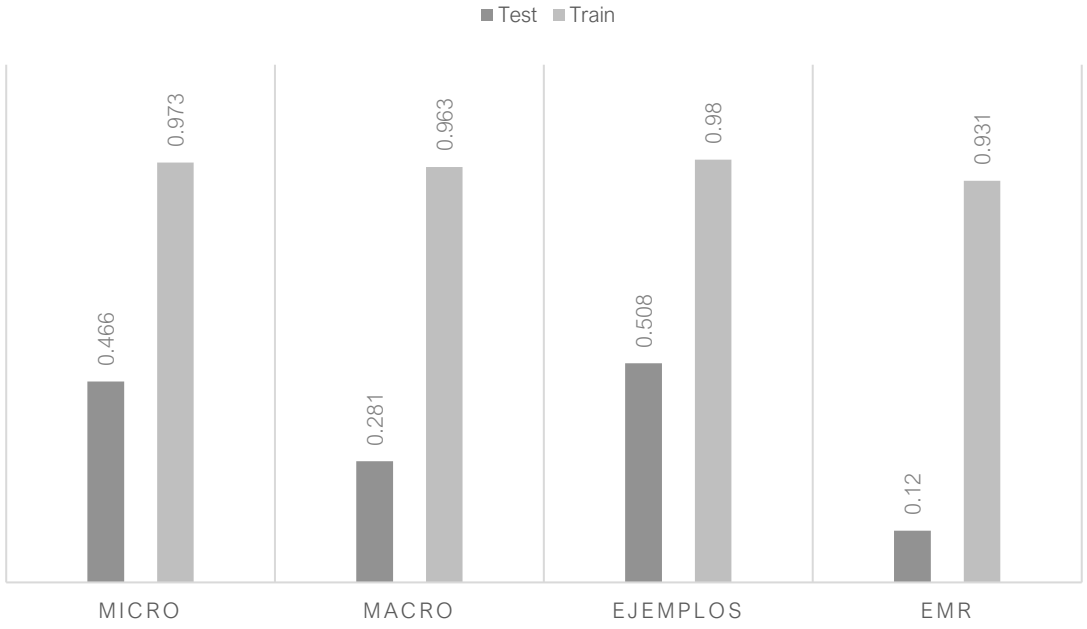


Gráfica 10: Precisión por géneros NB complementario

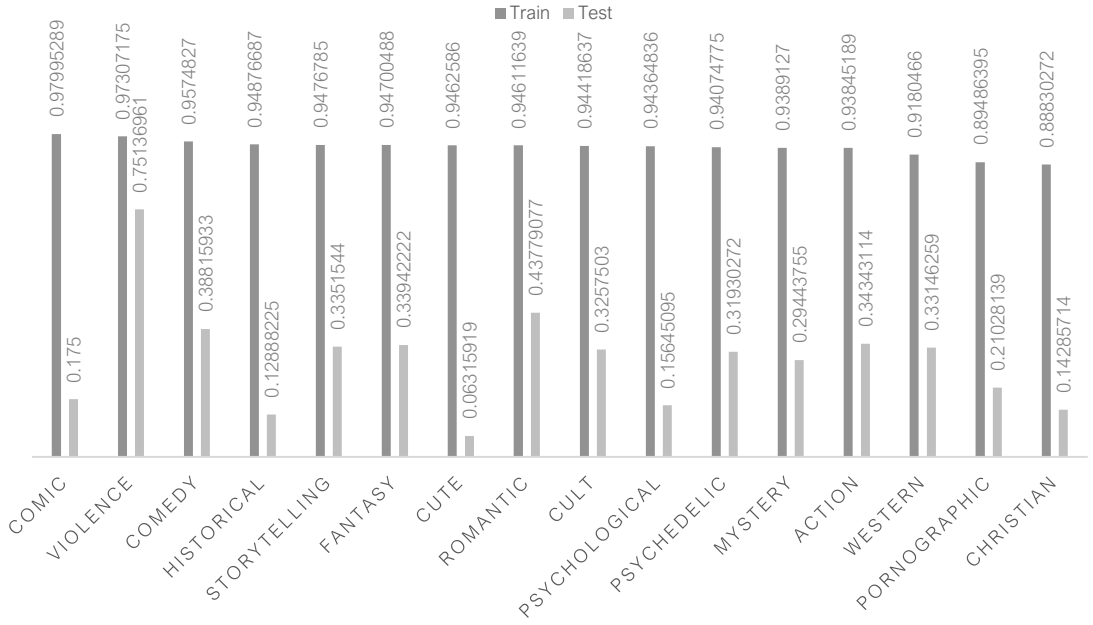


Gráfica 11: Recall por géneros NB complementario

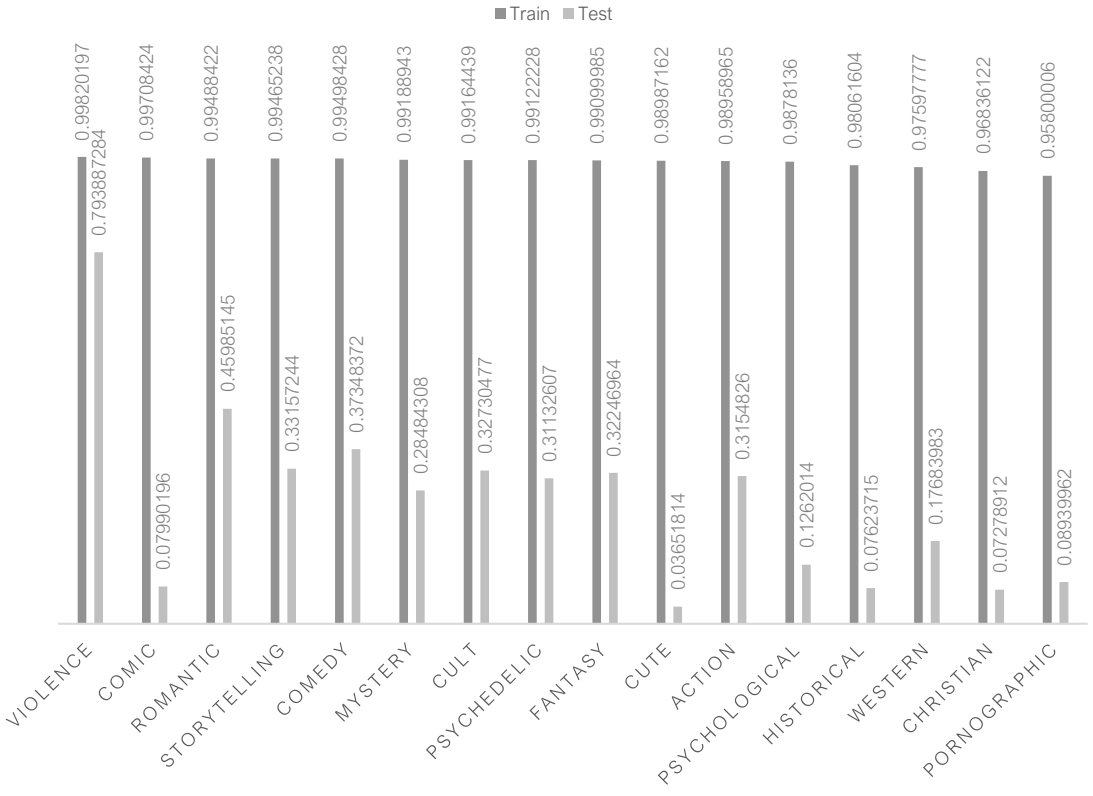
5.7.1.2 RED NEURONAL



Gráfica 12: Resultados red neuronal



Gráfica 13: Precisión por géneros red neuronal



Gráfica 14: Recall por géneros red neuronal

6 CONCLUSIONES

En este trabajo hemos planteado y evaluado diferentes técnicas y algoritmos para abordar la clasificación multi-etiqueta de texto. Llegamos a la conclusión que Naïve Bayes es de los mejores clasificadores para esta tarea, en particular el complementario. La mejor técnica ha resultado ser la transformación binaria donde se entrena un clasificador para cada etiqueta. A pesar de no considerar las relaciones entre clases, obtiene buenos resultados.

Si se quiere maximizar la precisión completa, es decir, el mejor EMR, la mejor técnica ha resultado ser la transformación multiclase, donde cada combinación de géneros diferentes representaría una clase y es únicamente el acierto completo la métrica a optimizar. El clasificador sería Naïve Bayes multinomial.

Consideramos que el resultado del Naïve Bayes complementario está cerca de su máximo potencial, sin mucho margen de mejora. Por el otro lado, la red neuronal ha mostrado ser uno de los mejores métodos. Con una red básica, hemos obtenido resultados muy parejos a los de Naïve Bayes complementario y obtenemos el mejor macro Fscore. A mayores, cabrían esperar mejores resultados con más datos para reducir el sobre entrenamiento y con otras arquitecturas quizás.

En cuanto a los clasificadores basados en knn y árboles de decisión, ciñéndonos a los resultados, no han mostrado ser soluciones convenientes para la clasificación de texto. Asimismo, han sido los más costosos de entrenar.

El mejor micro Fscore está por debajo del 0.5, si bien ha habido una evolución significativa desde los primeros resultados obtenidos. Cabe destacar que, vista la naturaleza del problema, la clasificación de sinopsis a géneros cinematográficos no es una ciencia exacta, dependiendo muchas veces de la valoración subjetiva del autor de la clasificación de imdb. De hecho, muchos géneros mal predichos comparados con los verdaderos géneros, no desentonan en absoluto una vez leída la sinopsis.

6.1 LÍNEAS FUTURAS

Con todo, creemos que las redes neuronales son las que más margen de mejora tienen. Dado los resultados que han demostrado en el procesamiento del lenguaje natural, pensamos que sería interesante utilizar redes con capas Long Short Term Memory (LSTM), que permiten la persistencia de la información. Son más convenientes para el aprendizaje de secuencias y clasificación de texto. Están diseñadas explícitamente para evitar problemas de dependencia a largo plazo debido a la desaparición del gradiente.

También sería interesante comparar los resultados con la clasificación de los pósteres de las películas. Con el identificador imdb, podemos hacer web scraping, esto es, mediante programas de software extraer información de sitios web, para obtener el póster de cada película. En lugar de minería de texto, habría que usar una red neuronal convolucional que tenga como entrada la imagen, extraiga las características y clasifique el póster.

7 BIBLIOGRAFÍA

[1] J. M. SADURNÍ, "Alan Turing, el arma secreta de los aliados," 2021. https://historia.nationalgeographic.com.es/a/alan-turing-arma-secreta-aliados_16352 (accessed Aug. 24, 2022).

[2] A. M. Turing, "COMPUTING MACHINERY AND INTELLIGENCE," *Computing Machinery and Intelligence. Mind*, vol. 49, pp. 433–460, 1950.

[3] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress," *IBM J Res Dev*, vol. 11, no. 6, pp. 601–617, Apr. 2010, doi: 10.1147/RD.116.0601.

[4] Carlos R Vidondo, "El peligro de girar a la izquierda - Circula Seguro," 2018. https://www.circulaseguro.com/el-peligro-de-girar-la-izquierda/ (accessed Aug. 24, 2022).

[5] Lex Fridman, "Deep Learning Basics: Introduction and Overview - YouTube," 2019. https://www.youtube.com/watch?v=O5xeyoRL95U (accessed Aug. 24, 2022).

[6] "Understanding K-means Clustering in Machine Learning | by Education Ecosystem (LEDU) | Towards Data Science." https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 (accessed Aug. 30, 2022).

[7] "One Hot Encoding | Interactive Chaos." https://interactivechaos.com/es/manual/tutorial-de-machine-learning/one-hot-encoding (accessed Aug. 31, 2022).

[8] "NumPy." https://numpy.org/ (accessed Aug. 29, 2022).

[9] "Matplotlib — Visualization with Python." https://matplotlib.org/ (accessed Aug. 29, 2022).

[10] "pandas - Python Data Analysis Library." https://pandas.pydata.org/ (accessed Aug. 29, 2022).

[11] "NLTK :: Natural Language Toolkit." https://www.nltk.org/ (accessed Aug. 29, 2022).

[12] "scikit-learn: machine learning in Python — scikit-learn 1.1.2 documentation." https://scikit-learn.org/stable/ (accessed Aug. 29, 2022).

[13] "scikit-multilearn: Multi-Label Classification in Python — Multi-Label Classification for Python." http://scikit.ml/ (accessed Aug. 29, 2022).

[14] "TensorFlow." https://www.tensorflow.org/ (accessed Aug. 29, 2022).