Trabajo de la asignatura Inteligencia Artificial.

Clasificación de documentos

Cálculo del género de una película a raíz de su sinopsis

Pedro Serrano Ramos
Fernando José García Padilla

Contenido

Parte I: Introducción 2

Parte II: Puesta a punto 2

Parte III: Elección del conjunto de palabras clave 3

Parte IV: Procesamiento 4

Parte IV-A: Procesamiento de Naive Bayes 4

Parte IV-B: Procesamiento de kNN 6

Parte V: Salvado del procesamiento en fichero 8

Parte V-A: Salvado del procesamiento de Naive Bayes 8

Parte V-B: Salvado del procesamiento de kNN 8

Parte VI: Ejecución de los algoritmos 9

Parte VI-A: Ejecución de Naive Bayes 9

Parte VI-B: Ejecución de kNN 13

Parte VII: Análisis de los resultados 19

Parte VIII: Conclusiones 21

Parte IX: Referencias 23

Parte X: Manual de usuario 24

Parte X-A: Ejecución del código 24

Parte X-B: Estructura de las carpetas 24

Parte X-C: Cómo añadir más documentos al conjunto de entrenamiento: 24

Parte X-D: Cómo crear una nueva categoría: 24

Parte X-E: Cómo clasificar un documento: 24

Parte X-F: Cómo elegir tus propias palabras clave (por categoría): 25

Parte X-G: Cómo auto-generar las palabras clave en base a su frecuencia si no existe la categoría en el archivo de palabras clave personalizadas: 25

Parte I: Introducción

Se requiere la implementación de un algoritmo que a partir de una sinopsis, y previamente entrenado, calcule el **género principal** de una película en base al conocimiento adquirido.
Los géneros que tendrá en cuenta el algoritmo son: **comedia**, **acción**, **terror**, **bélico** y **western**.

Los **archivos que contienen la sinopsis** de las películas (o equivalente, pudiendo ser también un breve resumen de la primera parte de la película) estarán distribuidos de la siguiente forma: **1)** Si forman parte del conjunto de pruebas estarán dentro de la carpeta del conjunto de prueba sin más, que será la carpeta en la que el algoritmo, una vez entrenado, buscará las sinopsis allí presentes para categorizarlas **2)** Si forman parte del conjunto de entrenamiento estarán dentro de la carpeta del conjunto de entrenamiento y **a su vez** dentro de una carpeta que indique su género.

Inicialmente se planteó utilizar *html* como **formato** para almacenar los archivos con los que vamos a trabajar pero, dado que no ha sido posible encontrar una fuente común para extraer todas las sinopsis, finalmente se almacenarán como **texto plano**, trabajando de esta manera con **archivos *.txt***.

Los conocimientos requeridos por parte del usuario que ejecutará el algoritmo **son mínimos**: tan sólo necesita **colocar** los textos, en el formato adecuado, en la carpeta indicada (mirar manual de usuario) y **ejecutar** el algoritmo en sí. Únicamente se requiere mayor interacción por parte del usuario si desea cambiar las palabras clave, puesto que entonces deberá modificar el fichero "*.csv*" indicado en el manual (por defecto, llamado “palabras_clave” y localizado en la carpeta “csv”).

Parte II: Puesta a punto

La puesta a punto ha consistido en crear el **directorio de trabajo** correctamente. Esto ha tratado en crear unas carpetas concretas en las que se guardarán los archivos *.txt* de los conjuntos de entrenamiento y prueba, que por defecto se guardan en las carpetas "*conjunto_entrenamiento*" y "*conjunto_prueba*" respectivamente.

Después de realizar esto hemos realizado la búsqueda de sinopsis, recurriendo a pequeños resúmenes o reseñas que dieran una información correcta del argumento y la temática de la película en cuestión para que todos los documentos estuvieran entre 200 y 300 palabras en caso de que las sinopsis fueran escuetas. También se ha intentado que las sinopsis den una información suficientemente relevante, por lo que todas han sido leídas por nosotros para que conformen un buen conjunto de trabajo. Cabe decir que se han buscado sinopsis de algunas películas que pudieran estar en dos géneros (por ejemplo la película “El Gran Dictador” que es una comedia con gran contenido bélico) para poder poner a prueba correctamente el algoritmo y que no fuera una ordenación fácil. Estas sinopsis se han ido guardando en archivos *.txt* en sus respectivas carpetas para su posterior uso.

Una vez se ha elegido cuáles van a ser los lugares de almacenamiento en disco y los nombres de las rutas se escanean los ficheros. El conjunto de entrenamiento, el cual tiene subcarpetas para

las categorías, se escanea para obtener los nombres de las carpetas de manera que se registren las categorías para su uso en el algoritmo. Después se escanean el contenido de todas las subcarpetas y almacenamos todos los archivos según su categoría en un diccionario (llamado archivos_entrenamiento_categoría), que contendrá como clave la categoría y como valor el conjunto de archivos de la categoría.

El escaneo del conjunto de prueba es exactamente el mismo que el de entrenamiento, solo que en este caso no hay carpetas de categorías y se emplea un set (para evitar repetición) con los archivos únicamente.

Parte III: Elección del conjunto de palabras clave

Para ayudarnos en el estudio de las **palabras clave** que debemos escoger para cada categoría vamos a realizar un pequeño estudio para determinar las palabras más frecuentes de cada categoría. La elección en sí, al menos en este caso concreto, vamos a realizarla a mano puesto que no podemos escoger directamente las más frecuentes ya que con toda probabilidad (y a pesar de las barreras que pondremos, como se ve en el código) entre las más frecuentes se nos colarán verbos, conectores, preposiciones, pronombres, artículos, nombres propios, etc. que en muchos casos no nos serán de utilidad a la hora de determinar la categoría de una película.

Definimos un **método que recibirá** tanto **un conjunto de archivos** como una **ruta** donde se encuentran y **contará las palabras** que aparecen en él **y el número de veces que dichas palabras aparecen**. Con este método hemos podido ver qué palabras se repiten más veces y elegir las que más nos convengan de manera que estén relacionadas con la categoría en cuestión y sirvan como buenas palabras clave. En algunos casos en los que muchas palabras se repetían pocas veces pero eran sinónimos o expresaban ideas similares se han cambiado intencionadamente sin incurrir en ello en apenas 8 o 9 documentos de los 120 de los que disponemos, de manera que no fuera un alteración buscando resultados correctos, únicamente una ayuda para cumplir con lo que pide el enunciado de que las palabras estén repetidas suficientes veces y haya al menos 4 o 5 palabras clave en cada documento. Esto se ha realizado principalmente en películas de terror y comedia que disponen de una gran cantidad de sinónimos para muchas palabras pero poca repetición de palabras concretas que puedan considerarse claves.

Adicionalmente, se proporciona un código que elige las 20 palabras más repetidas de cada categoría y las emplea como palabras clave. Es algo desaconsejado en este caso porque es una solución que da un conjunto de palabras un tanto aleatorio que puede influir demasiado en el resultado de los algoritmos. Esta opción sólo se ejecuta cuando hay alguna categoría sin palabras clave y se puede desactivar cambiando una variable en la sección “puesta a punto” del código.

Parte IV: Procesamiento

En esta parte se va a llevar a cabo la generación de los datos pertinentes para posteriormente utilizarlos en los algoritmos de **Naive Bayes** y **kNN**.

Parte IV-A: Procesamiento de Naive Bayes

Para aplicar el algoritmo Naive Bayes primero debemos calcular todos los **P(c)** ***(probabilidad de "c")*** y los **P(t|c)** ***(probabilidad de "t" condicionada a "c")***.

En este caso **"c"** sería nuestra categoría y **"t"** cada palabra clave.

Primero, vamos a calcular los **P(c)**. Para ello, tenemos que contar el número de documentos de la categoría en cuestión existentes en nuestro conjunto de entrenamiento y dividirlo entre el número total de documentos de nuestro conjunto de entrenamiento. Así pues, por ejemplo, la probabilidad de acción (**P(acción)**) sería el número resultante de dividir el total de documentos catalogados como "acción" de nuestro conjunto de entrenamiento entre el número total de documentos del conjunto de entrenamiento.
Como tenemos un conjunto con todos los archivos de entrenamiento y un conjunto específico por cada categoría, realizamos un bucle y por cada categoría generamos su probabilidad:

```
probabilidad_categoría = {}

for categoría in categorías:

        probabilidad_categoría[categoría] = len (archivos_entrenamiento_categoría
        [categoría]) /  len(archivos_entrenamiento)

    print("P(%s) = \t %f" % (categoría, probabilidad_categoría[categoría]))
```

```
P(terror) =     0.192308
P(western) =    0.221154
P(acción) =     0.201923
P(bélico) =     0.192308
P(comedia) =    0.192308
```

Se ha comprobado que todas las probabilidades deben sumar **~1** en este apartado y el resultado obtenido es 1.00.

Ahora, para calcular los **P(t|c)** será un poco más complejo. Para llevar a cabo esta tarea haremos uso de **un diccionario por cada categoría** que a su vez **contendrá otro diccionario dentro** que **relacionará palabras clave con su probabilidad condicionada a la categoría del diccionario**. De nuevo, tomaremos la categoría "acción" como ejemplo: en "probabilidad_palabraclave ['acción']" recogerá, por cada palabra clave, su probabilidad condicionada a la categoría acción, es decir, su **P(t|acción)**, siendo "t" cada entrada del diccionario.

Pero antes de empezar, definiremos un método, que utilizaremos en los siguientes pasos, que genere cada diccionario deseado como salida y, además, le **aplique un suavizado de LaPlace**:

```
def crea_diccionario_probabilidades_condicionadas(archivos_entrenamiento_categoría):

    probabilidades_condicionadas = {}

        cuenta_palabras_categoría = cuenta_palabras_desde_archivos    (ruta_conjunto_entrenamiento,
        archivos_entrenamiento_categoría) # Almacena las palabras clave de la categoría y el número de
        veces que se repiten.

        número_palabras_clave_totales = len(palabras_clave) # Número de palabras clave que poseemos
        en total.

    número_palabras_clave_categoría = sum(cuenta_palabras_categoría.values())

            for palabra_clave in palabras_clave:
      if palabra_clave in cuenta_palabras_categoría:
                    número_veces_aparece_palabra_en_categoría = cuenta_palabras_categoría
                   [palabra_clave] # Número de veces que la palabra clave se repite en esta categoría.
              else:
                        número_veces_aparece_palabra_en_categoría = 0

         resultado = ((número_veces_aparece_palabra_en_categoría + 1) /
        (número_palabras_clave_categoría + número_palabras_clave_totales)) # Añadimos 1 en el
        numerador y el número de palabras claves totales en el denominador para aplicar el suavizado.

            probabilidades_condicionadas[palabra_clave] = resultado

      return probabilidades_condicionadas
```

*La lista de soluciones es muy extensa por lo que no se añade en este documento, para consultar esos resultados es necesario mirar el archivo de Python Notebook que acompaña este documento.

Parte IV-B: Procesamiento de kNN

Como en este escenario **el orden de las palabras clave sí nos importa**, así que nos aseguramos de definir una lista con las palabras clave (creada a partir del conjunto de palabras clave que ya teníamos). También creamos un **diccionario** que tenga como **clave cada palabra clave** y como **valor el número de veces que se repiten en documentos distintos** con el método **palabras_clave_frecuencias_documentales**.

```
palabras_clave_frecuencias_documentales = {}

for archivo in archivos_entrenamiento:
    cuenta_palabras = cuenta_palabras_desde_archivo(ruta_conjunto_entrenamiento, archivo)


    for palabra in cuenta_palabras:
        if palabra in palabras_clave:
            if palabra in palabras_clave_frecuencias_documentales:
                palabras_clave_frecuencias_documentales[palabra] += 1
            else:
                palabras_clave_frecuencias_documentales[palabra] = 1


print(palabras_clave_frecuencias_documentales)
```

Además definimos un método que calcula el peso de esa palabra clave y ese documento.

```
def calculo_peso(palabra_clave, ruta, archivo):

    frecuencia_en_documento = 0
    frecuencia_documental = 0
    frecuencia_documental_inversa = 0
    peso = 0

    cuenta_palabras = cuenta_palabras_desde_archivo(ruta, archivo)

    if palabra_clave in cuenta_palabras:

        frecuencia_en_documento = cuenta_palabras[palabra_clave]


    frecuencia_documental = palabras_clave_frecuencias_documentales[palabra_clave]

    frecuencia_documental_inversa = math.log((len(archivos_entrenamiento) /
frecuencia_documental), 10)

     peso = frecuencia_en_documento * frecuencia_documental_inversa

    return peso
```

Con estos métodos calculamos los pesos de todos los archivos del conjunto, con lo cual podemos crear un diccionario ("**diccionario_palabraclave_pesos**") que **por cada archivo del conjunto de entrenamiento** (clave del diccionario) contenga una lista de pesos de cada palabra (valor de la clave del diccionario).

```
diccionario_palabraclave_pesos = {}

for archivo in archivos_entrenamiento:
    lista_pesos_archivo = calculo_pesos(ruta_conjunto_entrenamiento, archivo)

    diccionario_palabraclave_pesos[archivo.replace(".txt", "")] = lista_pesos_archivo
```

Parte V: Salvado del procesamiento en fichero

Dado que el enunciado de la práctica requiere que guardemos el procesado que acabamos que realizar (en la parte IV) para después utilizarlo en la ejecución de los algoritmos, procedemos a ello.

Parte V-A: Salvado del procesamiento de Naive Bayes

El objetivo aquí es que el usuario pueda abrir el archivo "*.csv*" y lo entienda, por ello vamos a guardar en la primera columna de dicho "*.csv*" (es decir, en el primer valor) cada probabilidad, y lo haremos como "*P(categoría)*" o "*P(palabra|categoría)*" y en la segunda columna, su valor.

A continuación se muestra una captura del resultado obtenido de los primeros 40 elementos obtenidos en el archivo *csv*.

	A	B	C	D	E	F	G	H	I
1	P(terror),0.1834862385321101								
2	P(hombres|terror),0.0002032520325203252								
3	P(pesadilla|terror),0.0012195121951219512								
4	P(ataque|terror),0.0006097560975609756								
5	P(carreras|terror),0.0002032520325203252								
6	P(unidos|terror),0.0002032520325203252								
7	P(duelo|terror),0.0002032520325203252								
8	P(divertidos|terror),0.0002032520325203252								
9	P(prisión|terror),0.0002032520325203252								
10	P(cómica|terror),0.0002032520325203252								
11	P(gags|terror),0.0002032520325203252								
12	P(muerte|terror),0.0014227642276422765								
13	P(problemas|terror),0.0002032520325203252								
14	P(enemigo|terror),0.0002032520325203252								
15	P(estadounidense|terror),0.0006097560975609756								
16	P(tópicos|terror),0.0002032520325203252								
17	P(oeste|terror),0.0002032520325203252								
18	P(guerra|terror),0.0002032520325203252								
19	P(fenómenos|terror),0.0006097560975609756								
20	P(asesina|terror),0.0002032520325203252								
21	P(soldado|terror),0.0002032520325203252								
22	P(agente|terror),0.0004065040650406504								
23	P(persecución|terror),0.0002032520325203252								
24	P(hilarantes|terror),0.0002032520325203252								
25	P(absurdas|terror),0.0002032520325203252								
26	P(terrible|terror),0.0008130081300813008								
27	P(grupo|terror),0.0008130081300813008								
28	P(asesinos|terror),0.0002032520325203252								
29	P(banda|terror),0.0002032520325203252								
30	P(irreverente|terror),0.0002032520325203252								
31	P(miedo|terror),0.0010162601626016261								
32	P(hijo|terror),0.0024390243902439024								
33	P(misteriosa|terror),0.0010162601626016261								
34	P(pistolero|terror),0.0002032520325203252								
35	P(bandidos|terror),0.0002032520325203252								
36	P(oficial|terror),0.0002032520325203252								
37	P(teniente|terror),0.0002032520325203252								
38	P(delirante|terror),0.0002032520325203252								
39	P(monstruo|terror),0.0004065040650406504								

Parte V-B: Salvado del procesamiento de kNN

El salvado de kNN es mucho más fácil, sólo tenemos que guardar el diccionario que generamos en la parte anterior: El primer valor del ".*csv*" sería la clave del diccionario (el nombre del documento) y el segundo su valor (la lista de pesos para cada palabra).

A continuación mostramos también una captura del resultado obtenido al guardar los datos obtenidos.

Nota: La captura es orientativa, pero debido a la cantidad de datos no se aprecian bien los valores ni los datos.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	terror/The Omen,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.9982007016691985, 0.0, 0.0, 0.0, 0.0, 0.0, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0.0, 3.7720820332319924, 0.0, 0.0, 0.0, 0.0, 0.0, 0																
2	western/El último tren de Gun Hill,"[0.0, 0.0, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.899588413001089, 0.0, 0.0, 0.0, 0.0, 4.988246609785849, 0.0, 0.0, 0.0, 5.0294427																
3	terror/Nosferatu,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.126398524767607, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.899588413001089,																
4	western/El fuera de la ley,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.331992382987989, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.40799946221919, 2.4941233048929243, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
5	comedia/El Dictador,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.899588413001089, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
6	bélico/Rescate al amanecer,"[0.0, 0.0, 0.0, 0.0, 4.539882155643594, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.331992382987989, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.13599982073973, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
7	terror/Babadook,"[0.0, 2.7454377331738304, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0819099697950434, 2.514721355487995, 0.0, 0.0, 0.0, 0.0, 0.0, 0.																
8	comedia/Scary Movie,"[0.0, 2.7454377331738304, 0.0, 0.0, 1.5132940518811981, 0.0, 0.0, 0.0, 0.0, 0.0, 1.6956156086751528, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,																
9	acción/Kill Bill - Volumen 2,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 6.163819939590087, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0																
10	comedia/El gran dictador,"[0.0, 0.0, 0.0, 0.0, 1.5132940518811981, 0.0, 3.592735593561034, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.331992382987989, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
11	comedia/Monsters University,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.																
12	acción/The transporter,"[6.450752502834941, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 8.698765239003267, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 10.805																
13	acción/Venganza - Conexión Estambul,"[2.5803010011339764, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.3912312173503056, 2.388762789235098, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.126398524767607, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0																
14	bélico/Corazones de acero,"[1.2901505005669882, 0.0, 0.0, 0.0, 1.5132940518811981, 0.0, 0.0, 0.0, 0.0, 0.0, 1.6956156086751528, 0.0, 0.0, 0.0, 0.0, 0.0, 3.3299809574699726, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.																
15	terror/Dracula,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0819099697950434, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0																
16	western/Centauros del desierto,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.331992382987989, 0.0, 3.0819099697950434, 0.0, 0.0, 2.899588413001089, 0.0, 0.0, 0.0, 1.13599982073973, 0.0, 0.0, 0.0, 0																
17	acción/El caso Bourne,"[0.0, 0.0, 0.0, 0.0, 1.5132940518811981, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.800976124332979, 0.																
18	comedia/La máscara,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.0819099697950434, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
19	western/Los 7 magníficos,"[3.8704515017009644, 0.0, 0.0, 0.0, 1.5132940518811981, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.8581345381729277, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,																
20	bélico/El hundimiento,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.388762789235098, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.																
21	terror/Annabelle,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
22	western/El sargento negro,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.7720820332319924, 0.0, 0.0, 0.0, 2.293452609430																
23	western/La diligencia,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.6956156086751528, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.388762789235098, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 4.5869052188615465, 0.0, 0.0, 0.0, 0																
24	acción/A todo gas,"[0.0, 0.0, 0.0, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 5.574403614518783, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.2934526094307732, 0.0, 0.																
25	acción/Fast and Furious 7,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.2573606777439974, 0.0, 0.0, 0.0, 2.2934526094307732, 0.0, 0.0, 0.0, 0																
26	western/Dos cabalgan juntos,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.2573606777439974, 0.0, 0.0, 0.0, 0.0, 2.2064412324411435, 0.0, 0																
27	western/Hasta que llegó su hora,"[0.0, 0.0, 0.0, 0.0, 1.5132940518811981, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 7.185471187122068, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.13599982073973, 0.0, 1.8581345381729277, 0.0,																
28	comedia/Los caballeros de la mesa cuadrada ,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.592735593561034, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,																
29	terror/The eyes of my mother,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.9982007016691985, 0.0, 0																
30	comedia/Shrek 2,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.388762789235098, 0.0, 0.0, 0.0, 0.0, 0.0, 2.899588413001089, 0.0, 1.2573606777439974, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0																
31	acción/Kill Bill - Volumen 1,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.6659961914939945, 0.0, 3.0819099697950434, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.13599982073973, 7.482369914678773, 5.5744036145																
32	comedia/2 rubias de pelo en pecho,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.388762789235098, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0																
33	western/Incidente en Ox-Bow,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.899588413001089, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.																
34	comedia/Sopa de ganso,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.388762789235098, 0.0, 0.0, 0.0, 0.0, 1.9979885744819834, 0.0, 0.0, 0.0, 0.0, 0.0, 2.388762789235098, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,																
35	acción/A todo gas 4,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.2573606777439974,																
36	acción/Misión imposible III,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 4.252797049535214, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0																
37	terror/El exorcista,"[0.0, 2.7454377331738304, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.6956156086751528, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.9982007016691985, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 3.305053521109253, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,																
38	terror/Sé lo que hicisteis el ultimo verano,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.89958841																
39	terror/La autopsia de Jane Doe,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.6956156086751528, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.2573606777439974, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,																
40	bélico/Malditos bastardos,"[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.2064412324411435, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.13599982073973, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.293452609430773																
41	acción/El legado de Bourne,"[0.0, 0.0, 0.0, 0.0, 1.5132940518811981, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.126398524767607, 0.0, 0.0, 0.0, 0.0, 1.13599982073973, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.																

Parte VI: Ejecución de los algoritmos

Parte VI-A: Ejecución de Naive Bayes

El método naive_bayes recibe un archivo y los datos procesados (probabilidades) como parámetros y determina la categoría del archivo pasado como parámetro.

```
def naive_bayes(archivo, csv):
    cuenta_palabras = cuenta_palabras_desde_archivo(ruta_conjunto_prueba, archivo)

    palabras_coincidentes_con_palabras_clave = cuenta_palabras.copy()

    # De las palabras que contiene el fichero, desechamos todas las que no coinciden con las palabras clave.
    for palabra in cuenta_palabras:
        if palabra not in palabras_clave:
            del palabras_coincidentes_con_palabras_clave[palabra]

    # Abrimos el fichero ".csv" generado para consultar datos en el siguiente paso.
    datos = lee_fichero(csv)

    # Ejecutamos el algoritmo en sí.
    candidatos = {} # "candidatos" será un diccionario que contendrá la categoría y la "puntuación" otorgada por el algoritmo a esa categoría (para posteriormente elegir la categoría con el máximo valor).

    for categoría in categorías:
        probabilidades_condicionadas_a_multiplicar = []
        for palabra_clave in palabras_coincidentes_con_palabras_clave:
            cadena_a_buscar = "P(" + palabra_clave + "|" + categoría + ")" # Define la cadena que se debe buscar en el archivo. En este caso la probabilidad condicionada a la categoría.
            probabilidades_condicionadas_a_multiplicar.append(float(datos[cadena_a_buscar]) ** palabras_coincidentes_con_palabras_clave[palabra_clave]) # Busca el valor de la probabilidad condicionada requerida (lo transforma en float), lo eleva al número de veces que se repite y añade el valor de la probabilidad condicionada hallada a una lista que se pasará a multiplicar después.

        cadena_a_buscar = "P(" + categoría + ")" # Define la cadena que se debe buscar en el archivo. En este caso la probabilidad de la categoría.
        probabilidad_categoría = float(datos[cadena_a_buscar])

        # Multiplicamos los elementos de la lista de probabilidades condicionadas
        probabilidades_condicionadas_multiplicadas = 1.0
        for elemento in probabilidades_condicionadas_a_multiplicar:
            probabilidades_condicionadas_multiplicadas=probabilidades_condicionadas_multiplicadas  *
            elemento

             candidatos[categoría] = probabilidades_condicionadas_multiplicadas * probabilidad_categoría
            # Calcula el coeficiente

    resultado = max(candidatos, key=candidatos.get) # Devuleve el resultado del algoritmo. En este caso el elemento del diccionario con mayor coeficiente.

    return resultado, candidatos
```

El resultado de aplicar el algoritmo [Naive-Bayes] al conjunto de p
ruebas es...
[La matanza de Texas] [terror]
Puntuaciones: {'terror': 6.6421157562412255e-22, 'western': 4.25090
01338501965e-23, 'acción': 2.139949487152396e-22, 'bélico': 3.83662
6112475314e-24, 'comedia': 1.3649960078388521e-22}

[La caza del Octubre Rojo] [western]
Puntuaciones: {'terror': 4.0766551872335675e-40, 'western': 3.53298
166531568e-38, 'acción': 1.8355694721515878e-41, 'bélico': 2.598289
727235958e-38, 'comedia': 4.891344658047962e-42}

[Salvar al soldado Ryan] [bélico]
Puntuaciones: {'terror': 2.6803438384585504e-93, 'western': 5.75592
9393826193e-85, 'acción': 1.3033914222923779e-90, 'bélico': 8.40684
1128756095e-74, 'comedia': 4.721307612084719e-90}

[El jinete pálido] [western]
Puntuaciones: {'terror': 6.423427814551806e-18, 'western': 5.251543
095568873e-17, 'acción': 1.1859663879112133e-17, 'bélico': 2.935347
174058401e-17, 'comedia': 8.960720576307947e-18}

[El padrino] [western]
Puntuaciones: {'terror': 2.800533532122317e-20, 'western': 1.366558
1205294917e-18, 'acción': 2.1476630607079945e-20, 'bélico': 3.98324
00389163345e-20, 'comedia': 7.728090190864982e-21}

[A todo gas - Tokyo race] [acción]
Puntuaciones: {'terror': 2.9208050681122417e-78, 'western': 2.84446
04145258177e-77, 'acción': 7.056017933518524e-73, 'bélico': 2.73479
840102747e-81, 'comedia': 6.381375948100528e-75}

[Los otros] [bélico]
Puntuaciones: {'terror': 1.739678366441114e-17, 'western': 2.745714
198732615e-17, 'acción': 5.445764026122918e-18, 'bélico': 1.1116107
641404593e-15, 'comedia': 2.007201409092981e-17}

[Shrek 3] [comedia]
Puntuaciones: {'terror': 1.1135873156830444e-07, 'western': 6.47329
5993485725e-08, 'acción': 2.027914247340023e-08, 'bélico': 5.689139
7426307825e-08, 'comedia': 5.810968542484503e-07}

[El tirador] [acción]
Puntuaciones: {'terror': 1.6762047611057854e-34, 'western': 2.79400
17115367256e-35, 'acción': 1.9958588936406894e-32, 'bélico': 3.5578
85217505503e-34, 'comedia': 1.5782689523765196e-34}

[Insidious] [terror]
Puntuaciones: {'terror': 3.6265952029077075e-20, 'western': 4.55453
585769664e-24, 'acción': 3.9702216830285633e-25, 'bélico': 2.740447
223196652e-25, 'comedia': 3.999011741715387e-24}

[Apocalipsis now] [bélico]
Puntuaciones: {'terror': 9.280052572049743e-39, 'western': 2.677161
0191637733e-35, 'acción': 1.3484345854628075e-39, 'bélico': 1.28669
70926722804e-32, 'comedia': 2.7655383191194143e-38}

[Resacón en Las Vegas] [comedia]

```
Puntuaciones: {'terror': 2.767548231767177e-23, 'western': 1.518178
6192322135e-26, 'acción': 8.102493230670537e-27, 'bélico': 8.563897
572489538e-28, 'comedia': 4.998764677144231e-22}
-------------------------
[Teléfono rojo - Volamos hacia Moscú]        [bélico]
Puntuaciones: {'terror': 2.78566284434092e-75, 'western': 4.2519808
12534568e-68, 'acción': 6.0680130769494995e-71, 'bélico': 9.6514544
776576e-59, 'comedia': 2.6269408543406636e-73}
-------------------------
[Fast and Furious 5]   [acción]
Puntuaciones: {'terror': 1.5272315090001868e-36, 'western': 7.17023
535894797e-38, 'acción': 1.0133927215617334e-34, 'bélico': 2.248498
5368902563e-40, 'comedia': 1.0208725435811897e-37}
-------------------------
[Solo ante el peligro]        [western]
Puntuaciones: {'terror': 1.4654529994139924e-59, 'western': 2.77082
05089832943e-45, 'acción': 5.093043957891559e-63, 'bélico': 1.68060
30947615667e-59, 'comedia': 1.9938079271885666e-57}
-------------------------
[Django desencadenado]        [western]
Puntuaciones: {'terror': 4.5295396204313385e-11, 'western': 3.32247
86963998586e-10, 'acción': 2.891970880377152e-11, 'bélico': 1.17422
90490466009e-11, 'comedia': 1.7347900912979908e-11}
-------------------------
[Sin perdón]    [western]
Puntuaciones: {'terror': 8.006258220131779e-57, 'western': 3.304855
805531548e-45, 'acción': 3.8570640501904346e-58, 'bélico': 1.990394
2652292818e-56, 'comedia': 1.6309137859436635e-56}
-------------------------
[La máscara]    [comedia]
Puntuaciones: {'terror': 1.3608034655599213e-21, 'western': 5.43517
4343015025e-22, 'acción': 1.3422894129424963e-22, 'bélico': 2.98743
00291872506e-22, 'comedia': 1.0548843110530705e-19}
-------------------------
[El ultimátum de Bourne]      [acción]
Puntuaciones: {'terror': 3.2051828706944456e-31, 'western': 3.33284
4898761666e-34, 'acción': 2.0993109129610655e-28, 'bélico': 7.91719
6630113797e-31, 'comedia': 3.718101029141485e-32}
-------------------------
[Jungla de cristal]    [acción]
Puntuaciones: {'terror': 2.4765532275321504e-28, 'western': 1.02272
51177462515e-28, 'acción': 1.294139308590699e-27, 'bélico': 6.18651
0114285529e-29, 'comedia': 1.437046047763184e-29}
-------------------------
[La noche de los muertos vivientes]  [western]
Puntuaciones: {'terror': 7.027022848502362e-32, 'western': 2.341639
529668874e-30, 'acción': 6.807976535084624e-35, 'bélico': 8.4554679
55182832e-33, 'comedia': 6.01245315191055e-34}
-------------------------
[It]    [western]
Puntuaciones: {'terror': 2.8269712919525e-29, 'western': 7.98482980
8453197e-28, 'acción': 2.6166624627363712e-30, 'bélico': 1.26083319
79932648e-28, 'comedia': 1.4872404116565933e-29}
-------------------------
[La chaqueta metálica]        [bélico]
Puntuaciones: {'terror': 4.700079352666471e-46, 'western': 8.141505
557194707e-45, 'acción': 5.817047923155085e-46, 'bélico': 1.9329901
98068345e-40, 'comedia': 4.1850146290204115e-46}
```

```
-------------------------
[Misión imposible - Protocolo fantasma]      [acción]
Puntuaciones: {'terror': 4.3290351624631224e-102, 'western': 2.1106
87506767027e-99, 'acción': 4.265266925383652e-93, 'bélico': 8.09123
0905333042e-96, 'comedia': 1.5774253277686902e-101}
-------------------------
[El gran dictador]     [comedia]
Puntuaciones: {'terror': 4.727965252561375e-29, 'western': 3.241064
599333954e-27, 'acción': 2.527615837091209e-29, 'bélico': 1.2726535
092244515e-26, 'comedia': 2.2417918345105666e-26}
-------------------------
[Monstruos SA]         [comedia]
Puntuaciones: {'terror': 8.012957176736116e-32, 'western': 5.443646
6679773876e-33, 'acción': 5.340127474972185e-34, 'bélico': 1.520101
7529818486e-33, 'comedia': 4.652273912713282e-29}
-------------------------
[Le llamaban Trinidad]         [western]
Puntuaciones: {'terror': 1.2571535708293389e-34, 'western': 3.91591
3540792316e-30, 'acción': 7.25281591095663e-38, 'bélico': 1.6942310
559550012e-35, 'comedia': 2.872449493325265e-32}
-------------------------
```

Parte VI-B: Ejecución de kNN

El siguiente método, "*calcula_distancia*", recibe como parámetros dos listas de pesos, comprueba si tienen el mismo tamaño y aplica la fórmula de similitud a ambas listas. Cada elemento de la lista está relacionado con una palabra clave (y las palabras clave mantienen posición entre las listas, es decir, el elemento 1 de la lista 1 es el peso de la misma palabra que el elemento 1 de la lista 2).

```
def calcula_distancia(v, w):
    # Comprueba que las listas de pesos son del mismo tamaño, por seguridad.
    if(len(v) == len(w)):
        numerador = sum([elemento_v * elemento_w for elemento_v, elemento_w in zip(v,w)])

        denominador = (math.sqrt(sum([elemento_v ** 2 for elemento_v in v]))) * (math.sqrt(sum([
elemento_w ** 2 for elemento_w in w])))

        return numerador / denominador
```

Posteriormente, calculamos los k elementos de mayor valor en el diccionario y pasamos a ejecutar el algoritmo kNN. Recibe un documento, los datos procesados de la parte IV y un valor de k y determina a qué categoría pertenece un documento.

```
def knn(archivo, csv, k):
    v = calculo_pesos(ruta_conjunto_prueba, archivo)
    #print(v)

    # Abrimos el fichero ".csv" generado para consultar datos en el siguiente paso.
        datos = lee_fichero(csv)
    # Ejecutamos el algoritmo en sí.
        documento_similitud = {} # documento_similitud será un diccionario que contendrá el archivo y la "pu
    ntuación" (similitud) otorgada por el algoritmo a esa categoría (para posteriormente elegir la categoría
    del archivo con la similitud más cercana a uno, que será el mayor valor).

        # Ahora que tenemos el peso del archivo a clasificar mediante el algoritmo y los pesos de los archivos
    del conjunto de entrenamiento (extraídos del ".csv" y guardados en forma de diccionario) tenemos que c
    alcular, una por una, la distancia a cada elemento del conjunto de entrenamiento y quedarnos con la m
    enor.
        for dato in datos:
            # Como lo que guardamos es una cadena, es necesario un pequeño procesamiento para transforma
    rlo de nuevo en una lista.
            w = datos[dato].replace('[', '') # Primero eliminamos "[".
            w = w.replace(']', '') # Hacemos lo mismo con "]".
            w = w.split(",") # Aplicamos ".split()" para volver a "trocear" la cadena y convertirla de nuevo en un
    a lista.
            w = [float(elemento) for elemento in w]
            documento_similitud[dato] = calcula_distancia(v, w) # "v" son los pesos del archivo a clasificar y "w
    " los del archivo del conjunto de entrenamiento que está siendo procesado.

        k_documentos_similitud = k_maximos(documento_similitud, k)
        #print(k_documentos_similitud)

        # Para hallar la categoría más repetida y gestionar los desempates haremos lo siguiente:
        # La clave del diccionario será un string: la categoría.
        # El valor un entero: el número de veces que se repite la categoría (para elegir la mayor, si no hay em
    pate).

        candidatos_repetición = {}

        for elemento in k_documentos_similitud:
            género_documento = elemento.split("/") # "género_documento" guarda "género/nombre_docume
    nto", por ejemplo "acción/El caso bourne".
            género = género_documento[0]
            if género not in candidatos_repetición:
                #candidatos[género] = (1, k_documentos_similitud[elemento])
                candidatos_repetición[género] = 1
            else:
                #candidatos[género] = (candidatos[género][0] + 1, k_documentos_similitud[elemento] + candidat
    os[género][1])
                candidatos_repetición[género] = candidatos_repetición[género] + 1
```

```
    #print(candidatos_repetición)

    # Hallamos el máximo número de veces que una categoría se repite
    elemento_máxima_repetición = max(candidatos_repetición, key=candidatos_repetición.get)
    #print(elemento_máxima_repetición)
    máxima_repetición = candidatos_repetición[elemento_máxima_repetición]
#print(máxima_repetición)

  # Ahora toca repetir un bucle similar. Esta vez vamos a excluir todos los elementos que no tengan la m
áxima repetición.
  # Entre los elementos con máxima repetición, nos quedamos con la categoría cuyos elementos sumen
más similitud.
  # La clave del diccionario volverá a ser un string, y de nuevo almacenará la categoría.
  # Esta vez el value será un decimal (float): la suma de las similitudes (para elegir la mayor si hay empat
e).

  candidatos_suma_similitudes = {}

  for elemento in k_documentos_similitud:
    género_documento = elemento.split("/") # "género_documento" guarda "género/nombre_document
o", por ejemplo "acción/El caso bourne".
    género = género_documento[0]
    if candidatos_repetición[género]== máxima_repetición:
      if género not in candidatos_suma_similitudes:
        candidatos_suma_similitudes[género] = k_documentos_similitud[elemento]
      else:
        candidatos_suma_similitudes[género] = candidatos_suma_similitudes[género] + k_documentos
_similitud[elemento]

  #print(candidatos_suma_similitudes)

  # Hallamos la(s) categoría(s) que más se repiten
  resultado = max(candidatos_suma_similitudes, key=candidatos_suma_similitudes.get)

  # Para una mayor exactitud, saber en qué categoría se enmarcará la muestra y a qué película se debe
devolvemos ambos datos (el elemento 0 contendrá la categoría y el 1 la película de dónde procede).
    return resultado, k_documentos_similitud
```

Una vez elegido un valor de k, obtenemos los siguientes resultados. En este caso k=5.

```
El resultado de aplicar el algoritmo [kNN] con [k=5] (5NN) al conju
nto de pruebas es...
[La matanza de Texas]          [terror]
Similitudes: {'comedia/Scary Movie': 0.3186666377439309, 'terror/It
follows': 0.2749618771481962, 'comedia/Madagascar': 0.2605317112296
8066, 'terror/Sé lo que hicisteis el ultimo verano': 0.252216145094
2836, 'terror/Dracula': 0.24428846884217864}
-------------------------
[La caza del Octubre Rojo]     [bélico]
Similitudes: {'western/La legión invencible': 0.3977084815521909, '
bélico/Cartas desde Iwo Jima': 0.38656256071602424, 'bélico/La cruz
de hierro': 0.3862028899165757, 'bélico/La delgada línea roja': 0.3
613432141805449, 'western/El fuera de la ley': 0.3352640768434217}
-------------------------
[Salvar al soldado Ryan]       [bélico]
Similitudes: {'bélico/La delgada línea roja': 0.6550512795948604, '
bélico/Banderas de nuestros padres': 0.541655757940658, 'bélico/Sen
deros de gloria': 0.4886119592093343, 'bélico/El último samurái': 0
.455297368657932, 'bélico/La cruz de hierro': 0.4043865116064608}
-------------------------
[El jinete pálido]     [acción]
Similitudes: {'acción/La jungla 4 punto 0': 0.37526195323347145, 'a
cción/Kill Bill - Volumen 1': 0.3537846019155799, 'terror/Babadook'
: 0.3435312932717374, 'western/El fuera de la ley': 0.3001185043838
2187, 'bélico/El último samurái': 0.2681568286764592}
-------------------------
[El padrino]    [terror]
Similitudes: {'acción/La jungla 4 punto 0': 0.5326943399358933, 'co
media/Solo en casa': 0.4556145761852245, 'terror/El resplandor': 0.
4470391019477693, 'western/El último tren de Gun Hill': 0.433771977
70686376, 'terror/Expediente Warren': 0.38917529526629024}
-------------------------
[A todo gas - Tokyo race]     [acción]
Similitudes: {'acción/Turbo-Charged Prelude': 0.5996636796862126, '
western/El tesoro de Sierra Madre': 0.37205355025923226, 'acción/Ve
nganza 3': 0.3644153620812513, 'western/Río rojo': 0.36079832539487
455, 'acción/A todo gas': 0.3315485003439085}
-------------------------
[Los otros]     [bélico]
Similitudes: {'terror/El resplandor': 0.43614897078456893, 'bélico/
Nacido el 4 de Julio': 0.394344577161173, 'bélico/Corazones de acer
o': 0.38794016206482423, 'bélico/Corazones de hierro': 0.3000252815
52441, 'bélico/En tierra hostil': 0.27554582377524833}
-------------------------
[Shrek 3]       [comedia]
Similitudes: {'comedia/Solo en casa': 0.6274419567501361, 'terror/E
l resplandor': 0.43520303298821866, 'comedia/Resacón 2': 0.41215034
187359506, 'terror/Expediente Warren': 0.3658654246458214, 'comedia
/Brüno': 0.3523581800187579}
-------------------------
[El tirador]    [acción]
Similitudes: {'acción/Venganza': 0.7134272449387835, 'acción/El cas
o Bourne': 0.3677320004544517, 'acción/Fast and Furious 6': 0.34627
99557476671, 'comedia/El Dictador': 0.26832980780625615, 'terror/Ju
On': 0.26408752262491747}
-------------------------
```

[Insidious] [terror]
Similitudes: {'terror/El resplandor': 0.621702325178079, 'terror/REC': 0.5474156639247406, 'terror/Dracula': 0.4650420228955495, 'terror/Babadook': 0.43946375252912484, 'western/Dos cabalgan juntos': 0.3587053153754683}

[Apocalipsis now] [bélico]
Similitudes: {'western/La legión invencible': 0.4093234969091791, 'bélico/Corazones de acero': 0.3791238372102773, 'bélico/La cruz de hierro': 0.3765646613949365, 'bélico/Rescate al amanecer': 0.3565098866737232, 'bélico/Cartas desde Iwo Jima': 0.3550086403873435}

[Resacón en Las Vegas] [terror]
Similitudes: {'terror/REC': 0.39477350039332815, 'comedia/Shrek 2': 0.32155769224249375, 'comedia/Resacón 2': 0.31164051916486596, 'terror/Annabelle': 0.29748114850807184, 'terror/The eyes of my mother': 0.29033160636479377}

[Teléfono rojo - Volamos hacia Moscú] [bélico]
Similitudes: {'bélico/Valkiria': 0.6213497020513182, 'bélico/Senderos de gloria': 0.5721104440458056, 'acción/La jungla 4 punto 0': 0.43872298017765693, 'bélico/Black Hawk derribado': 0.3824569434277699, 'bélico/Cartas desde Iwo Jima': 0.3468871811227467}

[Fast and Furious 5] [acción]
Similitudes: {'acción/Fast and Furious 6': 0.5584715870391733, 'acción/Misión imposible III': 0.3693283872284809, 'acción/Fast and Furious 7': 0.2944751980919072, 'acción/Venganza 3': 0.25730880867325195, 'acción/Turbo-Charged Prelude': 0.2281812106530238}

[Solo ante el peligro] [western]
Similitudes: {'western/Por un puñado de dólares': 0.812611156390968, 'western/El hombre que mató a Liberty Valance': 0.6601342191152921, 'western/Duelo de titanes': 0.6293629801982913, 'western/Los 7 magníficos': 0.5659371363310238, 'western/Grupo salvaje': 0.5496218714306101}

[Django desencadenado] [western]
Similitudes: {'western/Incidente en Ox-Bow': 0.5110671488171652, 'acción/Turbo-Charged Prelude': 0.3199849291959269, 'terror/It follows': 0.3066940693360221, 'comedia/2 rubias de pelo en pecho': 0.3061208072355976, 'western/El bueno el feo y el malo': 0.27766816924736254}

[Sin perdón] [western]
Similitudes: {'western/Por un puñado de dólares': 0.7148970376325343, 'western/Duelo de titanes': 0.6449721117973752, 'western/Los 7 magníficos': 0.5860842718148825, 'western/El dorado': 0.5262203933956806, 'western/Los odiosos ocho': 0.442609045098591}

[La máscara] [comedia]
Similitudes: {'comedia/Borat': 0.4969292593022351, 'terror/La semilla del diablo': 0.44101469058917553, 'comedia/Madagascar': 0.2905430926498795, 'terror/Expediente Warren': 0.2547982076082321, 'comedia/La vida de Brian': 0.2264009701435646}

[El ultimátum de Bourne] [acción]

Similitudes: {'acción/Fast and Furious 6': 0.47006453948199706, 'bélico/La cruz de hierro': 0.45448297775525576, 'acción/Misión imposible III': 0.35263100301250716, 'acción/Venganza 3': 0.3469689650021374, 'acción/Misión imposible - Nación secreta': 0.31316163166108496}

[Jungla de cristal] [acción]
Similitudes: {'terror/The Omen': 0.402393746692534, 'acción/Misión imposible - Nación secreta': 0.34046017141314266, 'acción/Kill Bill - Volumen 2': 0.1965857456350854, 'acción/Venganza 3': 0.1467473361573342, 'western/Centauros del desierto': 0.1324951416688937}

[La noche de los muertos vivientes] [terror]
Similitudes: {'terror/El resplandor': 0.5254304827215815, 'terror/Dracula': 0.39836058838860655, 'acción/Venganza - Conexión Estambul': 0.36694858126280633, 'western/Centauros del desierto': 0.35548715168458506, 'comedia/Solo en casa': 0.3528508250454109}

[It] [western]
Similitudes: {'bélico/Rescate al amanecer': 0.38508456234186506, 'acción/La jungla 4 punto 0': 0.3529778308011349, 'western/El hombre que mató a Liberty Valance': 0.3286856135641301, 'western/Por un puñado de dólares': 0.31155208841937343, 'comedia/Scary Movie': 0.2875867647304566}

[La chaqueta metálica] [bélico]
Similitudes: {'bélico/El sargento de hierro': 0.6368366067832345, 'bélico/En tierra hostil': 0.45137830768917997, 'bélico/La delgada línea roja': 0.3940098583299763, 'bélico/Corazones de hierro': 0.35819483089652954, 'bélico/Nacido el 4 de Julio': 0.2929640126145386}

[Misión imposible - Protocolo fantasma] [acción]
Similitudes: {'acción/Misión imposible - Nación secreta': 0.5000693486660265, 'acción/Misión imposible III': 0.3547252146221655, 'acción/2 Fast 2 Furious - A todo gas 2': 0.35063646456881775, 'acción/Fast and Furious 6': 0.34732458837746494, 'acción/Venganza - Conexión Estambul': 0.26109763132089137}

[El gran dictador] [comedia]
Similitudes: {'bélico/Nacido el 4 de Julio': 0.3224672892915101, 'comedia/Idiocracia': 0.3159422605245647, 'acción/Fast and Furious 7': 0.30303579382187384, 'comedia/La vida de Brian': 0.3026224531441328, 'bélico/Rescate al amanecer': 0.222780401884863 56}

[Monstruos SA] [comedia]
Similitudes: {'comedia/Monsters University': 0.5757624436639718, 'comedia/La vida de Brian': 0.3972484303814936, 'terror/Dracula': 0.31540792999639067, 'terror/Psicosis': 0.29306009612002537, 'comedia/Minions': 0.24159736483985378}

[Le llamaban Trinidad] [western]
Similitudes: {'western/El dorado': 0.570253991638018, 'western/Por un puñado de dólares': 0.5532767539292561, 'western/Duelo de titanes': 0.5417670807726143, 'western/El hombre que mató a Liberty Valance': 0.46278275909128574, 'western/Los 7 magníficos': 0.46006447130521894}

Parte VII: Análisis de los resultados

La siguiente tabla muestra la relación entre el **documento a clasificar** del conjunto de pruebas, su **género real** y los **géneros en los que ha sido clasificado** mediante **Naive Bayes** y un **valor bajo de kNN** (1) y **un valor un poco más elevado** (5) y, finalmente, con uno **mucho más elevado** (12).

Nombre del documento	Género REAL	Género Naive Bayes	Género 1NN	Género 5NN	Género 12NN
A todo gas – Tokyo race	**Acción**	*Acción*	*Acción*	*Acción*	*Acción*
Apocalipsis now	**Bélico**	*Bélico*	*Western*	*Bélico*	*Bélico*
Django desencadenado	**Western**	*Western*	*Western*	*Western*	*Western*
El gran dictador	**Comedia / Bélico**	*Comedia*	*Bélico*	*Comedia*	*Comedia*
El jinete pálido	**Western**	*Western*	*Acción*	*Acción*	*Terror*
El padrino	**Drama**	*Western*	*Acción*	*Terror*	*Western*
El tirador	**Acción**	*Acción*	*Acción*	*Acción*	*Acción*
El ultimátum de Bourne	**Acción**	*Acción*	*Acción*	*Acción*	*Acción*
Fast and Furious 5	**Acción**	*Acción*	*Acción*	*Acción*	*Acción*
Insidious	**Terror**	*Terror*	*Terror*	*Terror*	*Terror*
It	**Terror**	*Western*	*Bélico*	*Western*	*Western*
Jungla de cristal	**Acción**	*Acción*	*Terror*	*Acción*	*Acción*
La caza del Octubre Rojo	**Acción / Bélico**	*Western*	*Western*	*Bélico*	*Bélico*
La chaqueta metálica	**Bélico**	*Bélico*	*Bélico*	*Bélico*	*Bélico*
La máscara	**Comedia**	*Comedia*	*Comedia*	*Comedia*	*Comedia*
La matanza de Texas	**Terror**	*Terror*	*Comedia*	*Terror*	*Terror*
La noche de los muertos vivientes	**Terror**	*Western*	*Terror*	*Terror*	*Western*
Le llamaban Trinidad	**Western / Comedia**	*Western*	*Western*	*Western*	*Western*
Los otros	**Terror**	*Bélico*	*Terror*	*Bélico*	*Bélico*
Misión imposible – Protocolo fantasma	**Acción**	*Acción*	*Acción*	*Acción*	*Acción*
Monstruos S.A.	**Comedia**	*Comedia*	*Comedia*	*Comedia*	*Comedia*
Resacón en las Vegas	**Comedia**	*Comedia*	*Terror*	*Terror*	*Terror*
Salvar al soldado Ryan	**Bélico**	*Bélico*	*Bélico*	*Bélico*	*Bélico*
Shrek 3	**Comedia**	*Comedia*	*Comedia*	*Comedia*	*Comedia*
Sin perdón	**Western**	*Western*	*Western*	*Western*	*Western*
Solo ante el peligro	**Western**	*Western*	*Western*	*Western*	*Western*
Teléfono rojo – Volamos hacia Moscú	**Bélico / Comedia**	*Bélico*	*Bélico*	*Bélico*	*Bélico*

Nótese que el **género real** en algunos casos es un poco **ambiguo** (esto ha sido forzado, eligiendo el conjunto de prueba, puesto que nos parece correcto ver el comportamiento del algoritmo en ese tipo de casos), por lo que si acierta cualquiera de los dos daremos el resultado por correcto.

Además, si nos fijamos en la tabla, lo primero que nos llama la atención es que la película **"*El padrino*" no pertenece a ninguna de las categorías para las que se realiza el estudio**. Esto es porque se ha insertado en el conjunto de prueba a modo de "trampa" para ver cómo se comportar el algoritmo en estos casos: **Naive Bayes** y **12NN** lo categorizan como *Western*, **1NN** como *Acción* y **5NN** como *Terror*. Por lo tanto **apreciamos un comportamiento errático**.

Teniendo en cuenta esto (descontando 1 al total de categorías por la "trampa" de "*El padrino*") y a partir de la tabla anterior, vamos a realizar un estudio del porcentaje de acierto según el algoritmo aplicado. Para ello, utilizaremos la siguiente fórmula:

$$Porcentaje\ de\ aciertos\ (Algoritmo) = \frac{muestras\ coincidentes\ con\ la\ categoría\ real}{muestras\ totales} * 100$$

Porcentaje aciertos algoritmo **Naive Bayes**:

$$Porcentaje\ de\ aciertos\ (Naive\ Bayes) = \frac{22}{26} * 100 = 84.61\%$$

Porcentaje aciertos algoritmo **kNN** con *k=1* (**1NN**):

$$Porcentaje\ de\ aciertos\ (1NN) = \frac{19}{26} * 100 = 73.07\%$$

Porcentaje aciertos algoritmo **kNN** con *k=5* (**5NN**):

$$Porcentaje\ de\ aciertos\ (5NN) = \frac{22}{26} * 100 = 84.61\%$$

Porcentaje aciertos algoritmo **kNN** con *k=12* (**12NN**):

$$Porcentaje\ de\ aciertos\ (12NN) = \frac{21}{26} * 100 = 80.76\%$$

Ahora, mirando de nuevo la tabla, vamos a centrarnos en calcular el porcentaje de acierto por categoría. Consideraremos los resultados de 1NN, 5NN y 12NN independientes para el cálculo. Usaremos la siguiente fórmula en este caso:

$$Porcentaje\ de\ aciertos\ (Categoría) = \frac{muestras\ de\ la\ categoría\ coincidentes\ con\ la\ categoría\ real}{muestras\ totales\ que\ deberían\ clasificarse\ en\ la\ categoría} * 100$$

Porcentaje aciertos algoritmo **Acción**:

$$Porcentaje\ de\ aciertos\ (Acción) = \frac{23}{24} * 100 = 95.83\%$$

Porcentaje aciertos algoritmo **Comedia**:

$$Porcentaje\ de\ aciertos\ (Comedia) = \frac{17}{20} * 100 = 85\%$$

Porcentaje aciertos algoritmo **Bélico**:

$$Porcentaje\ de\ aciertos\ (Bélico) = \frac{15}{16} * 100 = 93.75\%$$

Porcentaje aciertos algoritmo **Terror**:

$$Porcentaje\ de\ aciertos\ (Terror) = \frac{10}{20} * 100 = 50\%$$

Porcentaje aciertos algoritmo **Western**:

$$Porcentaje\ de\ aciertos\ (Western) = \frac{17}{20} * 100 = 85\%$$

Parte VIII: Conclusiones

Parte VIII-A: Reseñas del aprendizaje automático

Recordemos que los algoritmos de **aprendizaje automático** (o *Machine learning*) intentan categorizar nuevos elementos en base a su experiencia previa con un conjunto de aprendizaje, con el que entrenan, pero **en ningún caso ofrecen una garantía a la hora de clasificar nuevos elementos**: se basan en su experiencia y en base al conjunto de entrenamiento y las palabras claves que hayan sido elegidas el resultado final puede verse alterado drásticamente.

En cualquier caso, incluso con el mejor escenario (elegido el mejor conjunto de entrenamiento, tanto en tamaño como en contenido, y las palabras clave más correctas) **no ofrecen un acierto del 100%**, puesto que "no" es lo que se pretende con esta clase de algoritmos.

Parte VIII-B: Resultados por clasificador

Ciñéndonos ahora a los resultados obtenidos en la parte anterior, tanto con **Naive Bayes** como con **kNN** con k igual a 1, 5 y 12 **en el peor de los casos nos quedamos cerca del 75% de acierto**, y **en el mejor estamos justo a las puertas del 85%**.

Teniendo en cuenta lo descrito en los párrafos anteriores, un **~75%** nos parece un resultado dentro de lo aceptable (aunque mejorable, pero normal si tenemos en cuenta que proviene de 1NN), y un **~85%** un resultado bastante bueno.

De entre todos los algoritmos (entendiendo por algoritmos tanto Naive Bayes como kNN con distintos valores), **el que peor se ha comportado con este conjunto de entrenamiento / pruebas / palabras clave ha sido 1NN** (kNN con k=1) con un **73.07%** y, por otra parte, **los que mejor se han comportado**, respecto a lo mismo, **han sido Naive Bayes y 5NN** (kNN con k=5), con un **84.61%** de aciertos. **12NN** (kNN con k=12) se queda un poco detrás con un **80.76%** de aciertos.

Debemos destacar también que **el algoritmo kNN**, en nuestro caso, se comporta mejor con un **"k" medio**, en este caso con k=5, ya que se obtiene **73.07% con k=1**, **84.61% con k=5** y un **80.76% con k=12**, por lo que **en mitad de los valores de "k" testeados se aprecia una mejora**, quedando el mejor porcentaje de acierto, como decíamos, en el k intermedio de los que se han probado.

Aunque no aparece en la tabla, con **valores de "k" muy altos el algoritmo empieza a fallar demasiado** (con k=70, por ejemplo) incluso categorizando películas en las que antes no tenía problemas.

Todo lo anterior nos hace pensar que **el valor de "k" idóneo** en este caso está en la franja de 5 a 12, pero es necesario un estudio más en detalle para comprobar esto de una forma veraz, puesto que puede ser fruto de la casualidad.

Parte VIII-C: Resultados por categoría

Además, podemos distinguir que algunas categorías se clasifican mejor que otras.

Cuando el género del documento a categorizar es **Acción**, aplicando Naive Bayes, 1NN, 5NN y 12NN, en un amplio porcentaje (**95.83%**) se categoriza bien. En concreto, de los elementos del conjunto de prueba que sólo están categorizados como *Acción*, únicamente *Jungla de cristal* con 1NN falla y se categoriza como *Terror*.

En el género **Terror** es dónde se presentan las mayores dificultades, ya que en este caso sólo en un **50%** de las veces se categoriza bien. Esto es un tanto relativo, ya que estamos contando los resultados de 1NN que a priori, según las conclusiones anteriores, es peor que 5NN y 12NN, pero sin dudas es la categoría que sufre más problemas para categorizarse correctamente. Esto puede deberse a la dificultad que presentó el género a la hora de elegir palabras clave, aunque intentamos corregir algunas sinopsis para que hubiera más similitudes siempre con la intención de mejorar el conjunto, nunca con la intención de trucar los resultados, porque las sinopsis encontradas presentaban poca similitud en la mayoría de palabras clave, lo cual demuestra la importancia de conseguir un buen conjunto de palabras clave para todas las categorías.

Los documentos a clasificar de los géneros **Western** y **Comedia** el **85%** de las veces son categorizados correctamente. Lo cual entra dentro de lo que podríamos considerar un buen resultado.

Y, por último, los documentos a de **Bélico** ofrecen un **93.75%** de acierto. Es algo que se esperaba puesto que pudimos apreciar durante el estudio de las palabras clave por categoría que *Bélico* es una categoría muy fácil de clasificar, ya que los documentos (sinopsis) de esta categoría tienen unas palabras clave tremendamente específicas, por ejemplo: soldado, general, almirante, etc.

Parte VIII-D: Conclusiones finales

En general **los algoritmos implementados categorizan correctamente en un porcentaje asumible**, y si no contamos 1NN aún más (ya que es esperable que con 1NN el algoritmo kNN arroje peores resultados que con un valor de k mayor), **superando el 80% de clasificaciones satisfactorias como media**.

Además, respecto a la **categorización por géneros**, nos encontramos que **todos los documentos a categorizar se categorizan bien**, **exceptuando el género de Terror** que se queda claramente por debajo. Esto puede deberse a una mala elección de las palabras clave, que el género sea por naturaleza más difícil de clasificar que otros, que haya que ampliar el conjunto de entrenamiento de *Terror* o una combinación de estas circunstancias.

Parte IX: Referencias

Para la elaboración del trabajo se ha usado principalmente el material y el conocimiento adquirido en la asignatura y en sus prácticas.
Además de ello, se han realizado consultas a la documentación de Python y a páginas como *StackOverflow*, para dudas puntuales. Las principales consultas son sobre el uso de Python (ya que es la primera asignatura en la carrera donde se nos enseña este lenguaje).

Una de las pocas excepciones en las que hemos consultado para algo menos básico este tipo de páginas ha podido ser para el recuento de palabras de un fichero o la lectura y escritura de ficheros ".csv", puesto que no sabíamos cómo realizarlos y sí tuvimos que, específicamente, buscar información para ello.

A continuación, ofrecemos una lista de dichas consultas:

- Contar palabras desde fichero: https://stackoverflow.com/q/21107505
- Leer y guardar csv: http://ccm.net/faq/2091-python-read-and-write-csv-files y https://stackoverflow.com/a/21780875
- Leer csv: https://stackoverflow.com/a/24662707
- Guardar csv: https://stackoverflow.com/a/41233907
- Ordenar lista: https://stackoverflow.com/a/23177099

Y también una lista de consultas que hemos podido realizar para pequeñas consultas **muy** específicas, que pese a ser de menor importancia las indicamos igualmente para dar total claridad al código que hemos realizado:

- Añadir un diccionario a otro: https://stackoverflow.com/a/11012181 (para obtener un resultado similar al método update que existe con conjuntos)
- Listar elementos de un directorio: https://stackoverflow.com/a/120676
- Comprobar que un elemento de un directorio es una carpeta: https://stackoverflow.com/a/40347279
- Borrar elementos de un diccionario: https://stackoverflow.com/a/5844692
- Buscar clave en diccionario: https://stackoverflow.com/a/1602945
- Crear tabla LaTeX en Jupyter Notebook: https://redd.it/4lxy1l
- Máximo valor de un diccionario: https://stackoverflow.com/a/42044202
- Reemplazar y eliminar elementos de una cadena: https://stackoverflow.com/a/3559600

Y para acabar, para la recopilación de **sinopsis** o argumentos se han usado, principalmente, las siguientes fuentes:

- https://www.filmaffinity.com/
- https://www.blogdecine.com/
- http://www.sensacine.com/
- http://www.labutaca.net/
- http://www.lahiguera.net/
- http://www.ecartelera.com/
- https://es.wikipedia.org/

Remarcar que, como expresamos al inicio del documento, en bastantes ocasiones ha sido complicado (o directamente imposible) encontrar una sinopsis que contuviese de 200 a 300

palabras, por lo que en algunas ocasiones hemos tenido que utilizar la primera parte del argumento de una película que ofrecen fuentes como *wikipedia*.

Parte X: Manual de usuario

Parte X-A: Ejecución del código

Para ejecutar el algoritmo: Pulsa **"Cell > Run All"** en la barra de tareas (parte superior de la pantalla) o **"Kernel > Restart & Run All"** para borrar todos las variables residuales y ejecutar el algoritmo.

Se adjunta también un fichero **".py"**, autogenerado a partir de este fichero, que también puede ejecutarse teniendo haciendo doble click Python instalado.

Recomendamos el primer método, ya que hemos trabajado con **Jupyter notebook** (tanto para código como para documentar el desarrollo del código de manera que sea totalmente entendible para el posible usuario).

Parte X-B: Estructura de las carpetas

Las carpetas "**conjunto_entrenamiento**" y "**conjunto_prueba**" contienen los ficheros del conjunto de entrenamiento y del conjunto de prueba respectivamente. Dentro de "conjunto_entrenamiento" hay una carpeta por cada categoría.

La carpeta "**csv**" contiene los **resultados del procesamiento de datos** para Naive Bayes y kNN **además del csv donde el usuario podrá definir palabras clave para cada categoría**.

La carpeta "**extra**" contiene carpetas de utilidad para realizar pruebas, estas son:

- Una carpeta con un backup de las carpetas "conjunto_prueba", "conjunto_entrenamiento" y "csv" tales como se pedían en el enunciado del trabajo.
- Una carpeta las carpetas "conjunto_prueba", "conjunto_entrenamiento" y "csv" (esta última contiene un csv con las palabras claves adecuadas) para recrear el ejemplo de Naive Bayes del enunciado del trabajo y así comprobar si está bien ese algoritmo.
- Una carpeta las carpetas "conjunto_prueba", "conjunto_entrenamiento" y "csv" (esta última contiene un csv con las palabras claves adecuadas) para recrear el ejemplo de kNN del enunciado del trabajo y así comprobar si está bien ese algoritmo.

Parte X-C: Cómo añadir más documentos al conjunto de entrenamiento:

Se pueden añadir tantos archivos de entrenamiento como se quiera a la carpeta de la categoría (dentro de la carpeta "conjunto_entrenamiento") en formato ".txt".

Parte X-D: Cómo crear una nueva categoría:

Simplemente crea una nueva carpeta dentro de la carpeta "conjunto_entrenamiento" y añade tantos archivos de entrenamiento a la categoría como quieras (en formato ".txt").

Parte X-E: Cómo clasificar un documento:

Simplemente colócalo en formato ".txt" dentro de la carpeta "conjunto_prueba".

Parte X-F: Cómo elegir tus propias palabras clave (por categoría):

Modifica el archivo "palabras_clave.csv" que se encuentra en la carpeta "csv". El primer valor debe ser la categoría y, separado por una coma, el segundo cada palabra clave separado también por comas (siempre dentro de las comillas que delimitan el segundo valor).

Parte X-G: Cómo auto-generar las palabras clave en base a su frecuencia si no existe la categoría en el archivo de palabras clave personalizadas:

Modifica el valor de la variable "usar_autogenerado_si_procede" al principio del código del algoritmo a "True" (o "False" en caso de no desear generar palabras clave nunca).