


Escuela Técnica Superior de
Ingeniería Informática

# Inteligencia Artificial

## Clasificador de documentos

María Teresa Jiménez Rodríguez
Juan Luis López Franco

# Descripción del problema a resolver

Se nos pide realizar un sistema clasificador de documentos genérico utilizando los algoritmos de clasificación Knn y Naive Bayes, así como proponer un ejemplo real de uso de dicho sistema de clasificación y realizar experimentos sobre el mismo.

# Descripción de la solución propuesta

Para responder a dichos requisitos, proponemos un sistema clasificador y su aplicación al ámbito de las películas a partir de su sinopsis en español, de forma que, dada una sinopsis y/o comentario de una película, el sistema intente predecir la categoría de la misma.
Las categorías contempladas para las películas (un total de 9) son:

- Acción
- Romántica
- Ciencia Ficción
- Terror
- Comedia
- Animación
- Fantasía
- Historia
- Drama

El sistema siempre propone una categoría para el nuevo documento.

# Descripción del sistema

El sistema genérico desarrollado se compone de tres grandes partes: Algoritmia, Test y Utilidad-Generación de diccionarios.

La primera se encarga de contener los algoritmos de clasificación y realizar los cálculos, la segunda se encarga de utilizar los archivos propuestos para test y lanzar los clasificadores para comprobar su efectividad. La última parte contiene utilidades para el resto de módulo, así como un pequeño sistema que genera los diccionarios automáticamente desde los archivos de entrenamiento.

Notas adicionales: todos los archivos de texto del sistema utilizan una codificación utf-8. Es importante tener en cuenta la codificación que utilice su editor de texto si pretende realizar modificaciones en dichos archivos, puesto que cambiar la codificación de algún archivo puede causar un fallo total o parcial en el funcionamiento del sistema.

El sistema propuesto ha sido desarrollado y testeado en Windows, se ha intentado hacer todo lo portable posible y se piensa que no debería haber problemas al ser ejecutado en otros sistemas operativos, sin embargo, esto no está, en absoluto, garantizado.

## Generación de diccionarios

Nuestro sistema contiene un pequeño generador de diccionarios que recorre los datos de entrenamiento organizados por categorías (explicado más adelante) y selecciona las 20 palabras más frecuentes dentro de esos textos.
Produciendo así una primera tentativa de diccionarios que ayuda al usuario a decidir el vocabulario que habrá de usarse para clasificar posteriormente los textos. Estos diccionarios se generarán automáticamente en la carpeta '*resources\data\generatedDictionaries'* para no sobreescribir los previamente modificados.
Como nota adicional, en los textos a tratar pudieran existir palabras consideradas "irrelevantes" o de poco impacto para el problema, para este caso, existe un fichero denominado *'discriminator.txt'* (en la carpeta *resources*) en el que dichas palabras pueden listarse. El sistema generador de diccionarios ignorará dichas palabras a la hora de dar las más frecuentes.

## Clasificación de documentos

Nuestro sistema contiene dos algoritmos de clasificación: Knn y Naive Bayes.
Ambos contenidos en clases que aportan un método clasifica, que recibe un texto y devuelve la clase calculada para el mismo.
El sistema conoce las clases del problema en cuestión a través de la estructura de ficheros (explicado más adelante).
Ambos algoritmos realizan su función en dos pasos (considerado de esta forma por términos de eficiencia).
El primer paso es el cálculo de los valores de predicción: dado que ambos algoritmos no parten exactamente de un conjunto de entrenamiento, si no de unos pesos calculados sobre los mismos y un diccionario, y que dichos pesos son reutilizables a través de las clasificaciones, hemos tomado la opción de que el sistema calcule dichos pesos y los escriba en un fichero, de forma que, más adelante, la clasificación parte de la lectura de estos ficheros.
Esta generación de ficheros se realiza una única vez y ha de ser forzada manualmente.

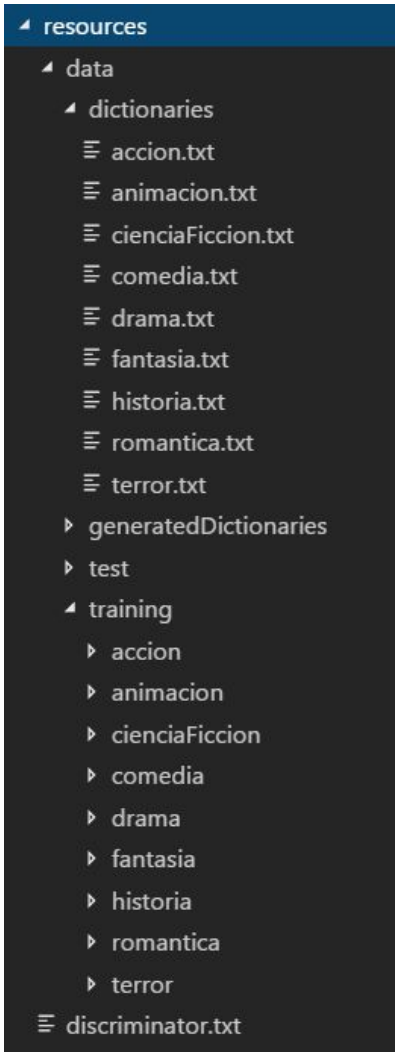
El segundo paso consiste en la clasificación en sí de un nuevo documento, para la cual, el sistema recibirá un texto nuevo que ha de clasificar, y devolverá su clase. Ha de mencionarse que existen dos formas de proporcionar al sistema el nuevo texto a clasificar, una es de manera directa a través de la consola (explicado más adelante) y otra es especificando la ruta relativa desde la raíz del proyecto donde se encuentra el fichero que contiene el texto nuevo.

# Tests

Además de lo mencionado previamente, se cuenta con un  pequeño sistema de test que utiliza los textos de test (identificados como tales, gracias a la estructura de proyecto) para probar la eficacia del sistema y su configuración.
Este subsistema recorre todos los textos de pruebas y los clasifica con uno u otro algoritmo, y devuelve el porcentaje de acierto obtenido y el tiempo de ejecución de cada algoritmo. Opcionalmente, también se le puede pedir la estructura interna que utiliza para relacionar texto-resultado, a fin de conocer la clasificación de un texto en concreto.

# Estructura de proyecto

Para que todo esto funcione, la estructura de proyecto ha de ser ciertamente concreta y los datos han de estar dentro de la raíz del proyecto bajo una carpeta llamada “resources”, de la forma que se muestra a continuación:




- La carpeta *dictionaries* contiene los diccionarios que utiliza el sistema, uno por cada categoría, son archivos txt cuyo nombre es el nombre de la categoría.

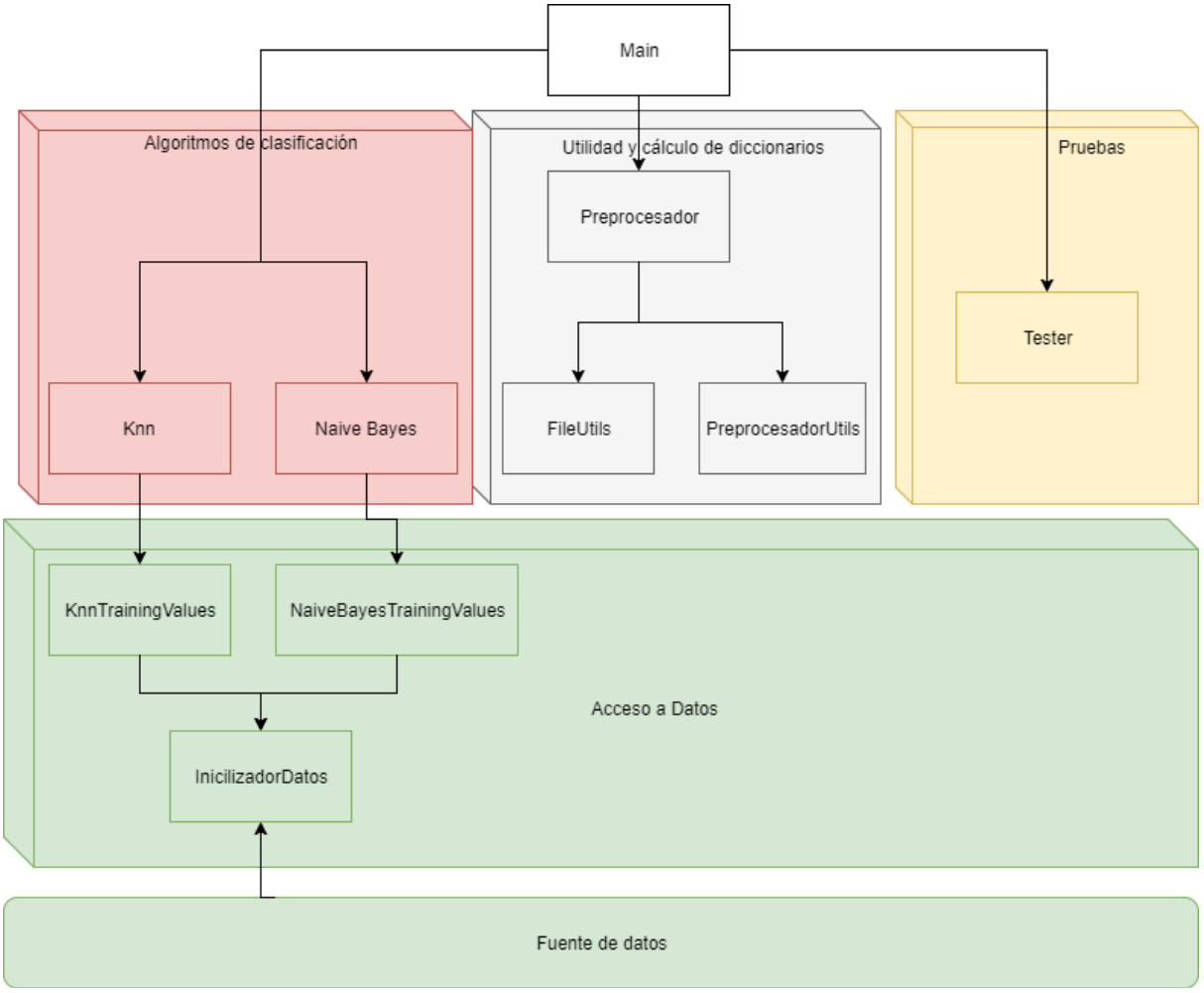
- La carpeta *Training* contiene una carpeta por cada categoría, cuyo nombre es la categoría (esto es MUY IMPORTANTE, dado que el sistema conoce las categorías del problema a través de los nombres de estas carpetas)

- La carpeta *Test* contiene ***la misma estructura*** que contiene *Training*, sólo que los textos son para probar. Nuevamente, el sistema sabe cuál es la categoría del texto a través del nombre de la carpeta que lo contiene.

- La carpeta *generatedDictionaries* contiene los diccionarios autogenerados por el sistema.

- *discriminator.txt* contiene las palabras (separadas por nueva línea) que no han de contarse a la hora de generar los diccionarios en función de la frecuencia de aparición de las palabras en los textos.

A continuación se presenta el diagrama de las clases y scripts del proyecto (contenidos dentro de la carpeta *src*):




A remarcar:

- *Main* es el punto de entrada a la aplicación (explicado más adelante)
- *KnnTrainingValues* y *NaiveBayesTrainingValues* son las estructuras que se encargan de leer los ficheros de pesos y de actualizarlos.
- *InicializarDatos* es una estructura singleton que busca cargar los datos relevantes para ambos algoritmos en memoria, de forma que sean accesibles a los algoritmos.
- *Preprocesador* contiene la utilidad de generación de diccionarios.
- *FileUtils* es una clase de utilidad que, en realidad, utilizan casi todos los scripts del sistema puesto que aporta un api de lectura/escritura de los ficheros.

# Problemas y dificultades encontradas

Durante este proyecto se han dado varios problemas destacables:

- Dadas las características de la estructura de proyecto. La referencia a módulos y archivos de texto en parte del proyecto han sido un problema constante. Al final se optó por añadir a las rutas del sistema src, en el caso de las importaciones, así como realizar una utilidad que acceda a los fichero de texto en el caso de las lectura/escritura.
- La codificación de los textos ya fueran de entrenamiento, test o incluso el discriminador debía ser la misma para la lectura y escritura de ficheros además de permitir los caracteres típicos del español (véanse las tildes y la letra ñ), por lo que se decidió utilizar UTF-8. Para ello se importó en la clase de utilidad FileUtils (donde se encuentra la lectura y escritura de ficheros) la librería de Python codecs. Esto hace que con ciertos programas (como WordPad) que utilicen otra codificación por defecto, no se visualicen correctamente las palabras.
- Al tratarse de un clasificador de películas y teniendo en cuenta que las películas suelen tratar más de una temática, la clasificación por parte del programa puede dar como resultado cualquiera de los géneros que trate si el texto a clasificar es ambiguo o no pertenece a una temática distinguida.

# Experimentación y resultados

Se han realizado distintas pruebas con diccionarios de distinto tipo, y textos diversos. Se han obtenido los siguientes resultados:

- Uso de los diccionarios editados manualmente propuestos por los desarrolladores:
    - Knn (Número de vecinos = 30):
        - Porcentaje de acierto: 80.0
        - Tiempo transcurrido (en segundos): 1.0172
    - Knn (Número de vecinos óptimo = 15):
        - Porcentaje de acierto: 84.44
        - Tiempo transcurrido: 1.0674
    - Knn (Número de vecinos pésimo = 100)
        - Porcentaje de acierto: 57.77
        - Tiempo transcurrido: 1.05
    - Naive Bayes:
        - Porcentaje de acierto: 82.2
        - Tiempo transcurrido: 2.73

- Uso de los diccionarios tal y como propone el generador de diccionarios:
    - Knn (Número de vecinos = 30):
        - Porcentaje de acierto: 11.11
        - Tiempo transcurrido: 0.7294
    - Knn (Número de vecinos óptimo = 6):

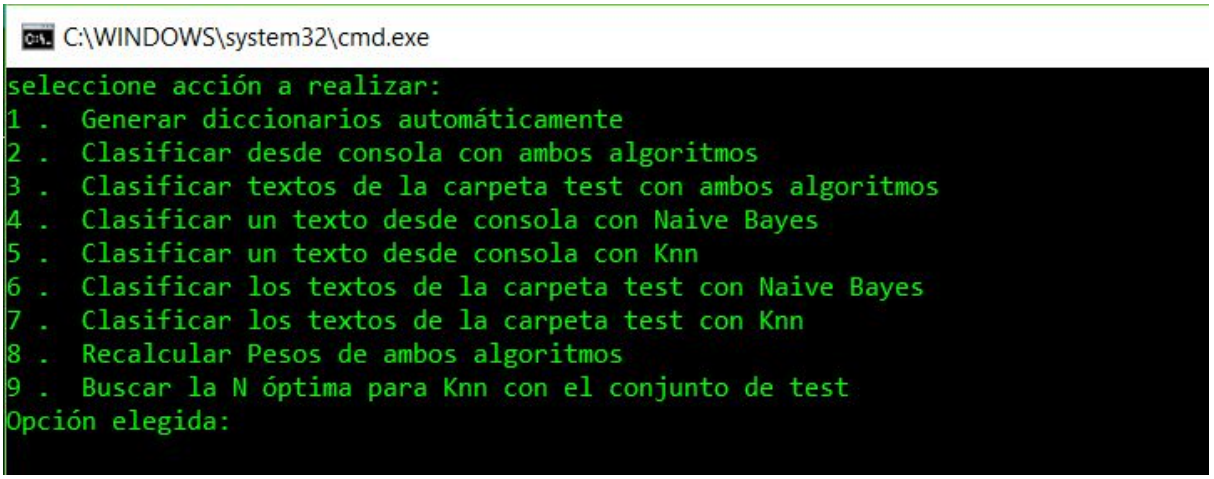
        - Porcentaje de acierto: 13.33
        - Tiempo transcurrido: 0.7245
    - Naive Bayes:
        - Porcentaje de acierto: 15.55
        - Tiempo transcurrido: 2.3681

Como se puede comprobar, de media, el clasificador Naive Bayes es más costoso en tiempo, sin embargo obtiene, en casi todos los casos, una mayor probabilidad de acertar. Por el contrario el clasificador basado en Knn obtiene con mayor rapidez el resultado, sin embargo la probabilidad de acertar es menor que la del clasificador de Naive Bayes. Si bien la efectividad del algoritmo Knn viene marcada por el número de vecinos elegido, tal y como muestran los resultados de las pruebas, tener una buena elección de la N tiene, por norma, poco impacto en el tiempo de ejecución y gran impacto en el porcentaje de acierto.

En cuanto a la diferencia de porcentajes entre los distintos diccionarios, se debe a que el discriminador no contiene todas las palabras poco relevantes existentes al ser un fichero generado manualmente con experiencia personal. Así, por un lado, los diccionarios automáticos contienen normalmente varias palabras irrelevantes y, por otro, pueden incluir palabras que más que representar la categoría en general, representan el conjunto de textos de entrenamiento en particular. Por esto es recomendable no utilizar únicamente la experiencia de una máquina para generar el vocabulario.

# Manual de uso

Con objeto de concentrar el acceso a las funcionalidades del sistema en un único punto que sirva de interfaz de usuario, el sistema viene con un script *main.py* contenido dentro de la carpeta *src* cuya ejecución mostrará una serie de opciones disponibles tales como:



- Generar los diccionarios automáticos, que producirá dichos diccionarios en la carpeta "*generatedDictionaries*", para su posterior revisión y edición. Se podrá elegir el número de palabras que contendrán los diccionarios generados automáticamente.
- Clasificar un nuevo texto con uno de los algoritmos o con los dos dando el texto directamente en consola. Esta opción también permite dar la ruta a un archivo de texto dentro del proyecto.
- Ejecutar la clasificación sobre los textos de prueba con uno o ambos algoritmos.
- Refrescar los pesos de los algoritmos.
- Calcular N óptimo (para Knn), ésta opción calculará el número de vecinos N (entre 1 y 100) que obtiene el mayor porcentaje de acierto con el conjunto de test definido, así como la N que obtiene el peor porcentaje, de esta forma se puede estimar la efectividad del algoritmo para el conjunto seleccionado.

La herramienta guiará al profesor en cada opción, pidiendo información adicional según la opción elegida.
Nota adicional: las opciones de clasificación preguntan si se desea ver la estructura de clasificación. Dicha estructura es un diccionario cuyas claves son cadenas de caracteres de la forma categoria - nombredelarchivo.txt y cuyos valores son la categoría predecida para dicho texto.

Para acceder a la herramienta *main.py*, podemos ejecutarla directamente desde un terminal o, si se dispone de un entorno Windows, puede ejecutarse el archivo *execute.bat* en la raíz del proyecto, haciendo simplemente doble click sobre él.

# Bibliografía


- Como apoyo en la profundización de ambos algoritmos se ha recurrido a: https://nlp.stanford.edu/IR-book/
- La mayoría de textos fueron extraídos de: http://www.cineyteatro.es/portal/BASEDEDATOSDEPELICULAS/tabid/56/language/es-ES/Default.aspx
- Documentación de Python: https://docs.python.org/3/