Informe Proyecto IAA

Kilian González Rodríguez

Implementación

Para la implementación del programa deseado se han realizado los siguientes ficheros:

- **<vocabulario.h>, <clasificador.h>, <main.cc>:** Ficheros con el código del programa en c++, estos ficheros se encargarán de leer el fichero de entrenamiento pasado, ó, “F75_train.csv” por defecto y a partir de él realizar un preprocesado el cúal entre otras cosas pasa a minúsculas las noticias y corrige algunos errores con caracteres especiales.
  Esto dará como resultado el fichero “vocabulario.txt” el cuál contendrá todas las palabras preprocesadas y ordenadas del fichero leido.
  Después recorrerá las noticias leidas y las guardará en el corpus correspondiente (P, N, T). Tras ello calculará la cantidad de veces que cada palabra aparece en cada corpus, calculando así la probabilidad de esta palabra y guardandolo en el fichero de “modelo_lenguaje” correspondiente.
  Por último el usuario seleccionará un fichero el cual tenga noticias las cuales serán preprocesadas, evaluadas y clasificadas acorde a la probabilidad total de sus palabras en función a nuestro modelo.
- **Makefile:** Fichero para la compilación del programa
- **README.md:** Fichero que explica el formato de ejecución y compilación del programa
- **Aprendizaje:** Fichero ejecutable, al ejecutaro manda a correr el programa.
- **Vocabulario.txt:** Fichero que contiene nuestro vocabulario.
- **corpus<P,N,T>.txt:** Ficheros que contienes las noticias correspondientes.
- **modelo_lenguaje<P,N,T>.txt:** Fichero que contiene las palabras correspondientes junto con su probabilidad y frecuencia.
- **resumen_alu0101222325.csv:** Fichero que contiene la clasificación de nuestro clasificador.
- **clasificacion_alu0101222325.csv:** Fichero que contiene las probabilidades de que una noticia esté en cada modelo.

Funcionamiento

En resumen el funcionamiento del programa será el de leer un fichero de noticias con su valoración, a este le aplicaremos una serie de correcciones principalente con expresiones regulares (eliminación de caracteres especiales, juntar palabras, eliminación de stopwords, eliminación de tildes). Tras ello se guardaran en cada corpus la noticia correspondiente, sin la clase.

Luego se crearan los ficheros de modelos<P,T,N> los cuales contendrán cada palabra correspondiente con el numero de veces que ha aparecido y logaritmo neperiano de su probabilidad.

Finalmente se procederá a pedir un fichero de validación el cual predicirá según las palabras que contenga cada noticia, la probabilidad de que esa noticia sea positiva, negativa o neutra, eligiendo la más probable.

A este programa le hemos añadido la posibilidad de mostrar la presición, solo en caso de que el fichero de validación tuviera la clase.

Ejecución

Para ejecutar el fichero una vez compilado solo tendrá que ejecutar la siguiente linea:

$./Aprendizaje <nombre_fichero_entrenamiento>

Si no selecciona ningun fichero se cojerá el fichero de entrenamiento “F75_train.csv”. A posteriori preguntará por el fichero de validación, sino se escribe ningún fichero este buscará el fichero “F75_test.csv”. Finalmente escribirá en los fichero mencionados en **implementación** los valores estimados.

Resultados

1.1 Validación y entrenamiento con el fichero “F75_train.csv”.

Al comparar los resultados predichos con los resultados verdaderos sobre las 2500 lineas, se ha acertado 2296 veces, resultando en una probabilidad del 91.84 %, por ende un error del 8.16%.

1.2 Validación y entrenamiento con el fichero “F75_train_1.csv” y “f75_train_2.csv”.

Tras entrenar el clasificador con las 2000 noticias de “F75_train_1.csv”, se ha procesido a a analizar “F75_train_2.csv obteniendo 371 aciertos con una probabilidad de acierto del 74.2%.

1.3 Validacion con las lineas 1 a 1000 y 1500 a 2500 como entrenamiento.

Decidimos hacer una prueba extra, extrayendo de F75_train.csv las 1000 hasta la 1500 los cuales se usarán para validar.

En este caso conseguimos 378 aciertos, con una porbabilidad de acierto del 75.4491%.

Conclusión

Tras analizar un poco los resultados, junto con los de algunos, he llegado a ciertas conclusiones:

- Al no haber aplicado lematización, la precición de nuestro clasificador es muy alta al validar con los mismos datos que entrena, pero esta baja dráticamente al validar con datos que no estaban en el entrenamiento. Esta diferencia aumentará más cuanto menos palabras del conjunto de validación existieran en el conjunto de entrenamiento. Esto si se aplicase lematización pese a que inicialmente pudiera ser menos preciso al generalizar las palabras, al aplicarse sobre conjuntos distintos hace que la palabra lematizada sea más probable que se encuentre en ambos conjuntos, haciendo que el resultado final sea más presiso.

- Balanceo de clases, al no estar las clases balanceadas, al entrenar nuestro clasificador este va a tender más a evaluar las noticias hacia ciertos modelos, cosa que si al tratar con un conjunto de validación en el cual por ejemplo haya más positivas que neutras en lugar de más nuetras que positivas, nuestro clasificador va a ver afectada su presición.

- Velocidad de ejecución, al no haber utilizado librerías externas el programa tiende a ejecutarse significativamente más rápido que programas de otros compañeros, que usaban librerías bastante pesadas para lematización entre otras cosas.