Filtrado de SPAM

Inteligencia artificial - TP6.2

Alberto Calvo Rubió - 760739
Universidad de Zaragoza

3 de enero de 2019

Índice

1. Introducción

En este segundo trabajo práctico se busca aprender como funciona un sistema de filtrado de spam en correos electrónicos. Para ello se va a aplicar un clasificador de Bayes ingenuo a un problema. Se va a diseñar e implementar un sistema de detección de spam en Python. Finalmente, se evaluará el sistema implementado para unas bases de datos públicas.

2. Preparación y lectura de correos

Para comenzar ha sido necesario instalar la distribución anaconda y actualizar los paquetes, ya que por ejemplo, si no se tenía la versión 0.22 de scikit-learn, varias de las funciones utilizadas no se podrían utilizar.

Los correos se han obtenido de la base de datos Enron, como especificaba el guión. Estos, se han leído de la misma forma que en el ejemplo proporcionado en Moodle (**load_mails.py**).

Posteriormente, es necesario convertir los mails en bolsas de palabras para poderlas utilizar con los clasificadores. La bolsa de palabras de los datos de entrenamiento y validación se han normalizado y se han transformado a bolsas de palabras que contienen la frecuencia de aparición en vez de el conteo de apariciones. Además se ha utilizado la opción *TfidfTransformer(smooth_idf=True)* que suma una aparición a cada palabra para prevenir una divisón por cero.

3. Entrenamiento del clasificador

Una vez se tienen todos los datos, se procede a llevar a cabo la lógica para decidir cual es el mejor clasificador y con que hiper-parámetros.

Se utiliza el método de **KFold cross-validation** para cada uno de los clasificadores(Multinomial y Bernoulli). En este método se van a probar distintos parámetros de laplace para realizar el suavizado y elegir el mejor. Para ello, se itera sobre diferentes valores realizando variaciones sobre el tamaño de folds que se van a realizar, y además se varían los subconjuntos de entrenamiento y validación mediante la función *sklearn.model_selection.KFold()* que devuelve en cada iteración distintos índices para dividir los datos. De esta forma aseguramos que las pruebas son consistentes y devuelven unos datos fiables. En cada iteración se crea un clasificador distinto, para mejorar los resultados partiendo de un clasificador totalmente nuevo.

Para seleccionar el mejor clasificador, se ha optado por **comparar** la métrica de f1_score ya que se trata de una medida que se centra en los falsos positivos y los falsos negativos. En nuestro caso nos interesa reducir los falsos positivos(no queremos que clasifique un ham como spam, ya que se podría perder información muy importante, en cambio un correo de spam en la bandeja de entrada no supondría demasiados problemas, ya que el usuario puede darse cuenta y borrarlo). Por el contrario, el accuracy es una métrica un poco más general, ya que se centra en la clasificación total. Por ello, tras cada iteración de los diferentes valores de laplace se comparan los resultados de f1_score y se mantiene el mejor de ellos. Finalmente se devuelve el mejor tipo de clasificador, su puntuación en f1_score y accuracy, y el hiper-parámetro de laplace.

4. Evaluación del mejor clasificador

En la evaluación del mejor clasificador, en este caso el **Multinomial** con el hiper-parámetro de laplace 0.1, se crea de nuevo, se entrena con los datos de entrenamiento y se procede a calcular las métricas, pero esta vez, con los datos de test. Antes de ello, se han realizado diferentes predicciones que se utilizarán en la evaluación:

- **prediction**: predicción realizada con *classifier.predict()*. Determina si cada uno de los correos de test es spam(1) o ham(0) según un umbral de 0.5 (por defecto en el paquete *sklearn*)
- **prediction_proba**: predicción realizada con *classifier.predict_proba()*. Determina la probabilidad de cada correo de pertener a cada clase, devolviendo dos columnas en este caso (una para cada clase). Se selecciona la segunda columna(probabilidad de spam) para utilizarla con las métricas.
- **prediction_custom**: predicción realizada con *classifier.predict_proba()* pero además se clasifica de forma manual, que si la probabilidad de ser spam es mayor o igual a 1 se guarde un 1(spam). De esta forma, se simula el primer apartado de predicción pero con un valor de umbral de 1. Este valor se ha elegido dado que lo que se quiere reducir al máximo es el número de falsos positivos como se mostrará más adelante con los datos de las métricas.

Para mostrar la relación entre la precision y el recall, se han dibujado dos gráficas. La primera es la representación que da la función estándar de dibujado de sklearn. En cambio, en la segunda se refleja como van variando ambos valores mientras que el umbral va aumentando. Esta gráfica es de mucha utilidad ya que aquí se muestra que umbral viene mejor dependiendo de si nos interesa precision o recall. En nuestro caso, nos interesa el parametro precision alto(disminuye los falsos positivos), por ello se toma el umbral = 1 para la prediction_custom.

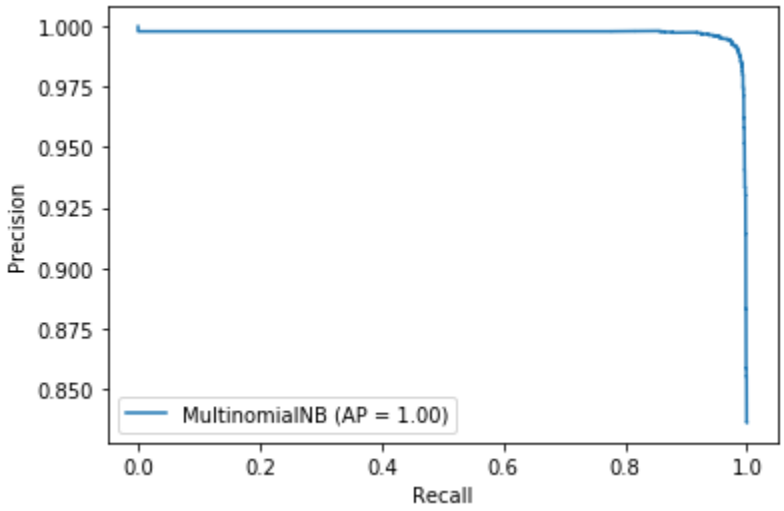

Figura 1: Curva precision_recall

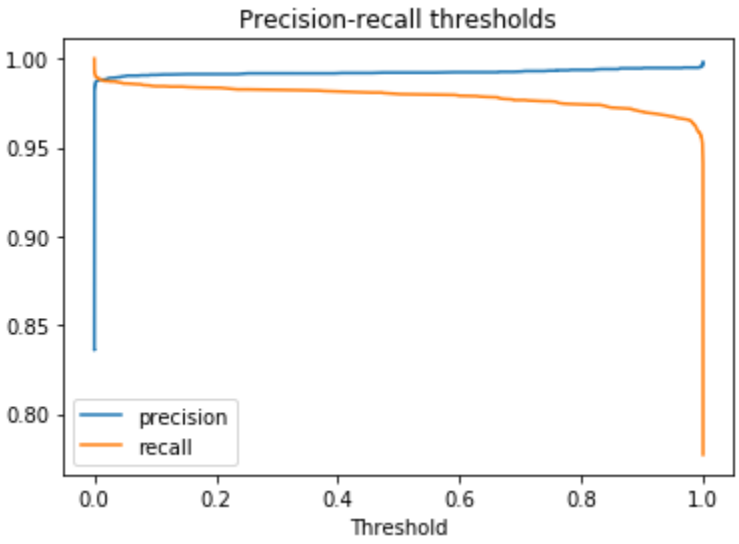

Figura 2: Curva precision_recall respecto al umbral

Las métricas utilizadas a continuación han sido f1_score, matriz de confusión y matriz de confusión normalizada tanto para *prediction* como *prediction_custom*, ya que son las únicas que se pueden utilizar como parámetros de estas. Con estas métricas podemos observar la media armónica entre precision y recall y además, el número y proporción de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.

```
-- Utilizando predicciones de clases (Umbral=0.5) --
f1_score:  0.9860257126886529
Confusion_matrix:
 [[1465   35]
 [  90 4410]]
Normalized confusion matrix:
 [[0.97666667 0.02333333]
 [0.02       0.98      ]]
```

(a) Predicciones con umbral = 0.5

```
-- Utilizando predicciones de propabilidades (Umbral=1)--
f1_score:  0.8738130934532734
Confusion_matrix:
 [[1493    7]
 [1003 3497]]
Normalized confusion matrix:
 [[0.99533333 0.00466667]
 [0.22288889 0.77711111]]
```

(b) Predicciones con umbral = 1

Figura 3: Métricas de predicciones

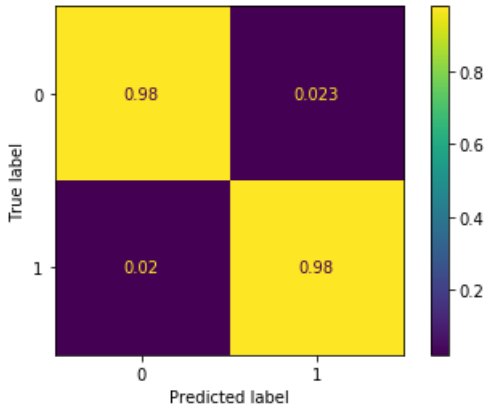

Figura 4: Plot de matriz de confusión normalizada(umbral=0.5)

Se puede observar que al aumentar el umbral, el f1_score disminuye debido a que se han producido muchos más falsos negativo, sin embargo, se ha reducido bastante el número de falsos positivos (los que nos interesan) tanto en la matriz normal como en la matriz normalizada.

Para finalizar, se han observado distintos correos que se han clasificado correctamente y otros que no. En la mayoría de los clasificados correctamente, se ha podido observar como se repetían palabras como *prizes*, *winning*, *win*, *morevaluableitems*, direcciones web, *business*, etc. Alguno de los clasificados incorrectamente, como es el caso del mail con índice 434, al leer el cuerpo del mensaje, una persona se daría cuenta que es un spam, pero hay detalles como el titulo en el que se nombra *work problem*, pudiéndose referir a algo importante de trabajo. Por ello esta palabra probablemente reduzca las probabilidades de spam y el correo se haya clasificado como ham.

5. Conclusiones

Tras realizar todas las pruebas se obtiene que el mejor clasificador es el Multinomial, probablemente porque tiene en cuenta el número de apariciones de cada palabra. En cambio, Bernoulli tiene en cuenta la presencia/ausencia de cada palabra.

Otro aspecto observado es que el mejor hiper-parámetro de laplace ha sido inferior a 1. Esto puede ser mejor porque realmente los mensajes de spam y ham no se reciben en igual proporción, sino que hay un desequilibrio. Al aumentar el suavizado, se supondría una proporción igual, y de esta forma se muestra en los datos como disminuye el f1_score. Al ser el suavizado menor que 1, se suele denominar *Lidstone smoothing*.

Finalmente, cabe destacar que elegir como umbral 1 puede ser un poco extremo ya que se clasificarían bastantes más correos de spam como ham. En las pruebas también se ha utilizado como umbral 0.99 para comprobar que sucedía, pero los resultados no eran tan claros como con 1, siendo similares a las predicciones normales, aunque también se podría haber elegido para la evaluación al dismunuir los falsos positivos.