Inteligencia Artificial

Filtrado de correo SPAM mediante inferencia probabilística

Miembros

Menéndez Montes, Eva  
  
Ramos González, José Renato

**Índice**

[**Resumen de la entrega** 2](#_Toc453852446)

[**Descripción del proyecto** 2](#_Toc453852447)

[**Enfoque del problema** 2](#_Toc453852448)

[**Procedimientos** 3](#_Toc453852449)

[A. Incorporación 3](#_Toc453852450)

[- funcion\_a(arg1, arg2): 3](#_Toc453852451)

[- funcion\_b(arg1, arg2): 3](#_Toc453852452)

[- funcion\_c(arg1, arg2): 3](#_Toc453852453)

[- incorporation(arg1, arg2): 3](#_Toc453852454)

[B. Clasificación 3](#_Toc453852455)

[- funcion\_a(arg1, arg2): 3](#_Toc453852456)

[- funcion\_b(arg1, arg2): 3](#_Toc453852457)

[- funcion\_c(arg1, arg2): 3](#_Toc453852458)

[- clasification(arg1, arg2): 3](#_Toc453852459)

[C. Incorporación 3](#_Toc453852460)

[- funcion\_a(arg1, arg2): 3](#_Toc453852461)

[- funcion\_b(arg1, arg2): 3](#_Toc453852462)

[- funcion\_c(arg1, arg2): 4](#_Toc453852463)

[- incorporacion(arg1, arg2): 4](#_Toc453852464)

# **Resumen de la entrega**

(Resumen de los que nos piden en el proyecto)

# **Descripción del proyecto**

(que usamos un notebook, que es funcional y no orientado a objetos, incluimos miembros, fechas, etc)

# **Enfoque del problema**

(que lo dividimos en 3 procedimientos)

# **Procedimientos**

## Incorporación

<Explicamos cómo lo abordamos y cualquier cosa adicional que requiera explicación>

### funcion\_a(arg1, arg2):

### funcion\_b(arg1, arg2):

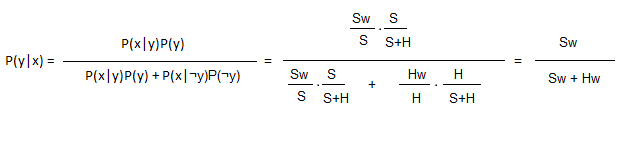
### funcion\_c(arg1, arg2):

### incorporation(arg1, arg2):

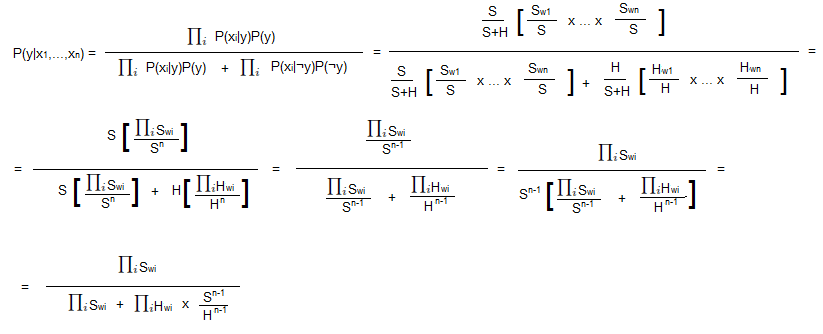
## Clasificación

Para clasificar los correos nuevos, seguimos la estructura que se nos indicó en el enunciado del problema. En primer lugar, se calcula P(y|xw) para cada palabra del correo que estemos clasificando, y se escogen las 15 que mejor lo clasifican individualmente (más cercanas a 0 ó 1). Calculamos el valor de P(y|x1, … ,x15), y si este valor es superior a 0.9, el correo es considerado SPAM. Por último, devolvemos una lista con valores 0 ó 1 que clasifica todos los correos del fichero de entrada.

Con el fin de facilitar los cálculos, se han despejado algunas de las fórmulas más complejas, quedando de la siguiente forma:



*(1): P(y|x)*

**

*(2): P(y|x1,…,xn)*

### best\_N\_words(words, spam\_dict, ham\_dict, n):

Recibe un set de palabras de un correo, los dictionary de spam y ham y el número máximo de palabras que debe devolver. Devuelve una lista de las n palabras con mejor clasificación individual.

Por cada palabra en ese set de palabras:

1. Se obtiene el número de ocurrencias spam y ham de la palabra.
2. Se calcula P(y|x) para esa palabra usando la fórmula (1), siendo 0.5 si el número de ocurrencias tanto en ham como en spam es 0.
3. Se calcula el valor absoluto de la diferencia entre esa probabilidad y 0.5. Ahora, las mejores clasificaciones son las que están más cercanas a 0.5.

Por último, ordenamos el diccionario de probabilidades y palabras en orden decreciente de probabilidad, y devolvemos las n primeras palabras.

### naive\_bayes(bests, spam\_dict, s, ham\_dict, h, debug = False):

Recibe la lista de las 15 mejores palabras, el diccionario de spam y ham, el número total de correos SPAM y el de correos HAM, y una variable debug que por defecto es False. Devuelve el valor P(y|x1,…,xn) para las 15 palabras que recibe.

Por cada palabra de la lista de las 15 mejores:

1. Se calcula el número de ocurrencias spam y ham de la palabra.
2. Si alguno de los valores anteriores es 0, tenemos que realizar el suavizado, creando una entrada ficticia por cada 0 que tengamos.
3. Calculamos el producto del número de ocurrencias tanto para ham como para spam.

Para terminar, actualizamos los valores de s y h teniendo en cuenta el número de veces que ha tenido que realizarse el suavizado, y devolvemos la probabilidad de esas 15 palabras, calculada con la fórmula (2).

### clasification(newsbox, spam\_dict, s, ham\_dict, h, regex = ‘\W+’, num = 15, debug = False):

Recibe un mbox de correos nuevos, los diccionarios de ham y spam, el número total de correos SPAM y HAM, y un regex, un número y una variable debug por defecto. Devuelve una lista de 0 ó 1 por cada correo nuevo indicando si es HAM o SPAM.

Por cada set de palabras de cada correo del mbox de correos nuevos:

1. Se obtienen las 15 palabras con mejor clasificación individual.
2. Se calcula el valor P(y|x1,…,xn) para las 15 palabras obtenidas anteriormente.
3. Se añade a la lista un 0 si ese correo es clasificado como HAM y un 1 si es clasificado como SPAM.

Por último, se devuelve la lista.

## Incorporación

Como solución a los problemas que se nos plantean, donde tenemos que incorporar un nuevo correo SPAM o HAM al conjunto de entrenamiento actualizando los valores correspondientes de Sw o Hw, según el caso, hemos implementado las siguientes funciones:

### append\_new\_to\_mbox(new\_mbox, mbox):

Recibe por parámetro dos rutas de mbox, ‘new\_mbox’, donde se encuentran los correos a añadir, y ‘mbox’, donde van a ser añadidos.

El procedimiento que se sigue es el siguiente:

1. Habilitar la lectura de mbox en modo “appending”. Todo lo que se escriba en este documento se añade al final.
2. Habilitar la lectura de new\_mbox.
3. Se escribe en mbox el contenido de new\_mbox.

### update\_dict(new\_mbox, dictionary, delimiter):

Con esta función se actualiza el dictionary de palabras, teniendo en cuenta las palabras que contiene el o los correos pertenecientes a new\_mbox.

1. Por cada set de palabras de los correos de new\_mbox:

* Recorremos cada palabra.
* Añadimos 1 al número de ocurrencias de dicha palabra en el diccionario.

1. Devolvemos el número de correos para los que se ha actualizado el dictionary.

### incorporation(new\_mbox, mbox, size, dictionary, delimiter = ‘\W+’):

Utiliza las dos funciones explicadas anteriormente para escribir los correos del mbox ubicado en la ruta ‘new\_mbox’ en el mbox ubicado en la ruta ‘mbox’. Devuelve los valores de S o H actualizados (la suma del número de correos SPAM o HAM actual y el número de correos añadidos).