Parsing del correo electrónico

[6]:
```
p = Parser()
p.parse("datasets/datasets/trec07p/data/inmail.1")
```

[6]:
```
{'Subject': ['gener', 'ciali', 'brand', 'qualiti'],
 'body': ['\n\n\n\n\n\n\ndo',
  'feel',
  'pressur',
  'perform',
  'rise',
  'occasion\n\n\n\n\n\ntri',
  'viagra\nyour',
  'anxieti',
  'thing',
  'past',
  'will\nb',
  'back',
  'old',
  'self\n\n\n'],
 'content_type': 'multipart/alternative'}
```

Lectura del índice

Estas funciones complementarias se encargan de cragar en memoria la ruta se cada correo electronico y su etiqueta correspondiente.{Spam, ham}

[7]:
```
index = open("datasets/datasets/trec07p/full/index").readlines()
index
```

[7]:
```
['spam ../data/inmail.1\n',
 'ham ../data/inmail.2\n',
 'spam ../data/inmail.3\n',
 'spam ../data/inmail.4\n',
 'spam ../data/inmail.5\n',
 'spam ../data/inmail.6\n',
 'spam ../data/inmail.7\n',
 'spam ../data/inmail.8\n',
 'spam ../data/inmail.9\n',
 'ham ../data/inmail.10\n',
 'spam ../data/inmail.11\n',
 'spam ../data/inmail.12\n',
 'spam ../data/inmail.13\n',
 'spam ../data/inmail.14\n',
 'spam ../data/inmail.15\n',
 'spam ../data/inmail.16\n',
 'spam ../data/inmail.17\n',
 'spam ../data/inmail.18\n'
```

[8]:
```
import os

DATASET_PATH = "datasets/datasets/trec07p"

def parse_index(path_to_index, n_elements):
    ret_indexes = []
    index = open(path_to_index).readlines()
    for i in range(n_elements):
        mail = index[i].split(" ../")
        label = mail[0]
        path = mail[1][:-1]
        ret_indexes.append({"label": label, "email_path": os.path.join(DATASET_PATH, path)})
    return ret_indexes
```

[9]:
```
def parse_email(index):
    p = Parser()
    pmail = p.parse(index["email_path"])
    return pmail, index["label"]
```

[10]:
```
indexes = parse_index("datasets/datasets/trec07p/full/index", 10)
indexes
```

[10]:
```
[{'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.1'},
 {'label': 'ham', 'email_path': 'datasets/datasets/trec07p/data/inmail.2'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.3'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.4'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.5'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.6'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.7'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.8'},
 {'label': 'spam', 'email_path': 'datasets/datasets/trec07p/data/inmail.9'},
 {'label': 'ham', 'email_path': 'datasets/datasets/trec07p/data/inmail.10'}]
```

Preprocesamiento del DataSet.

Con las funciones presentadas anteriormente se permite la lectura de los correos electrónicos de manera programática y el procesamiento de los mismos para eliminar aquellos componentes que no resutan de utilidad para la detección de correos de SPAM. Sin embargo cada uno de los correos sigue estando representado por un diccionario de Python con una serie de palabras.

[11]:
```
# Cargar el indice y las etiquetas en memooria.
index = parse_index("datasets/datasets/trec07p/full/index", 1)
```

[12]:
```
# Leemos el primer correo
import os

open(index[0]["email_path"]).read()
```

[12]:
```
'From RickyAmes@aol.com  Sun Apr  8 13:07:32 2007\nReturn-Path: <RickyAmes@aol.com>\nReceived: from 129.97.78.23 ([211.202.101.74])\n\tby speedy.uwaterloo.ca (8.12.8/8.12.5) with SMTP id l38H7G0I003017;\n\tSun, 8 Apr 2007 13:07:21 -0400\nReceived: from 0.144.152.6 by 211.202.101.74; Sun, 08 Apr 2007 19:04:48 +0100\nMessage-ID: <WYADCKPDFWWTWTXNFVUE@yahoo.com>\nFrom: "Tomas Jacobs" <RickyAmes@aol.com>\nReply-To: "Tomas Jacobs" <RickyAmes@aol.com>\nTo: the00@speedy.uwaterloo.ca\nSubject: Generic Cialis, branded quality@ \nDate: Sun, 08 Apr 2007 21:00:48 +0300\nX-Mailer: Microsoft Outlook Express 6.00.2600.0000\nMIME-Version: 1.0\nContent-Type: multipart/alternative;\n\tboundary="--8896484051606557286"\nX-Priority: 3\nX-MSMail-Priority: Normal\nStatus: RO\nContent-Length: 988\nLines: 24\n\n----8896484051606557286\nContent-Type: text/html;\nContent-Transfer-Encoding: 7Bit\n\n<html>\n<body bgcolor="#ffffff">\n<div style="border-color: #00FFFF; border-right-width: 0px; border-bottom-width: 0px; margin-bottom: 0px;" align="center">\n<table style="border: 1px; border-style: solid; border-color:#000000;" cellpadding="5" cellspacing="0" bgcolor="#CCFFAA">\n<tr>\n<td style="border: 0px; border-bottom: 1px; border-style: solid; border-color:#000000;">\n<center>\nDo you feel the pressure to perform and not rising to the occasion??<br>\n</center>\n</td></tr><tr>\n<td bgcolor=#FFFF33 style="border: 0px; border-bottom: 1px; border-style: solid; border-color:#000000;">\n<center>\n\n<b><a href=\'http://excoriationtuh.com/?lzmfnrdkleks\'>Try <span>V</span><span>ia<span></span>gr<span>a</span>.....</a></b></center>\n</td></tr><tr><td><center>your anxiety will be a thing of the past and you will<br>\nbe back to your old self.\n</center></td></tr></table></div></body></html>\n\n\n----8896484051606557286--\n\n'
```

[13]:
```
# Parsear el primer correo
mail, label = parse_email(index[0])
print("El correo es: ", label, "\n")
print(mail)
```

```
El correo es:  spam

{'Subject': ['gener', 'ciali', 'brand', 'qualiti'], 'body': ['\n\n\n\n\n\n\ndo', 'feel', 'pressur', 'perform', 'rise', 'occasion\n\n\n\n\n\n\ntri', 'viagra\nyour', 'anxieti', 'thing', 'past', 'will\nb', 'back', 'old', 'self\n\n\n'], 'content_type': 'multipart/alternative'}
```

El algoritmo de Regresión Logística no es capaz de ingerir texto como parte del DataSet, por lo tanto, deben aplicarse una serie de funciones adicionales que transformen el texto de los correos electrónicos parseados enun arepresentación númerica.

Aplicacion de CountVectorizer

[16]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

[16]:
```
from sklearn.feature_extraction.text import CountVectorizer

# Preparación del email de una cadena de texto .
prep_email = [" ".join(mail['Subject']) + " ".join(mail['body'])]

vectorizer = CountVectorizer()
X = vectorizer.fit(prep_email)

print("e-mail:", prep_email, "\n")
print("Caracteristicas de entrada:", vectorizer.get_feature_names_out(),"\n")
```

```
e-mail: ['gener ciali brand qualiti\n\n\n\n\n\n\ndo feel pressur perform rise occasion\n\n\n\n\n\ntri viagra\nyour anxieti thing past will\nb back old self\n\n\n']

Caracteristicas de entrada: ['anxieti' 'back' 'brand' 'ciali' 'do' 'feel' 'gener' 'occasion' 'old'
 'past' 'perform' 'pressur' 'qualiti' 'rise' 'self' 'thing' 'tri' 'viagra'
 'will' 'your']
```

[17]:
```
X = vectorizer.transform(prep_email)
print("\nValues:\n", X.toarray())
```

```

Values:
 [[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]]
```

Aplicacón de OneHotEncoding

[20]:
```
from sklearn.preprocessing import OneHotEncoder

prep_email = [[w] for w in mail['Subject'] + mail['body']]
enc = OneHotEncoder(handle_unknown = 'ignore')
X = enc.fit_transform(prep_email)

print("Features:", enc.get_feature_names_out(), "\n")
print("\Values:\n", X.toarray())
```

```
 'x0_perform' 'x0_pressur' 'x0_qualiti' 'x0_rise' 'x0_self\n\n\n'
 'x0_thing' 'x0_viagra\nyour' 'x0_will\nb']

\Values:
 [[0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 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. 1. 0. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 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. 1. 0. 0.]
 [0. 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. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 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. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]]
```

Funciones auxiliares para el preprocesamiento del DataSet

[21]:
```
def create_prep_dataset(index_path, n_elements):
    X = []
    y = []
    indexes = parse_index(index_path, n_elements)
    for i in range(n_elements):
        print("\rParsing email: {0}".format(i+1), end = '')
        mail, label = parse_email(indexes[i])
        X.append(" ".join(['Subject']) + " ".join(mail['body']))
        y.append(label)
    return X, y
```

[23]:
```
# Leer únicamente un subconjunto de 1000 correos electrónicos
x_train, y_train = create_prep_dataset("datasets/datasets/trec07p/full/index", 1000)
x_train
```

```
Parsing email: 1000
```

[23]:
```
['Subject\n\n\n\n\n\n\ndo feel pressur perform rise occasion\n\n\n\n\n\ntri viagra\nyour anxieti thing past will\nb back old self\n\n\n',
 'Subjecthi ive updat gulu i check mirrors\nit seem littl typo debianreadm file\n\nexample\nhttpgulususherbrookecadebianreadme\nftpftpfrdebia
norgdebianreadme\n\ntest lenni access releas diststest the\ncurr test develop snapshot name etch packag which\nhav test unstabl pass autom te
st propog to\nthi release\n\netch replac lenni like readmehtml\n\n\n\n \nyan morin\nconsult en logiciel libre\nyanmorinsavoirfairelinuxcom\n5
149941556\n\n\n \nto unsubscrib email debianmirrorsrequestlistsdebianorg\nwith subject unsubscrib troubl contact listmasterlistsdebianorg\n
\n',
 'Subjectmega authenticy i a g r a discount pricec i a l i s discount pricedo miss it click here\nhttpwwwmouisikbchumcom\n\n authent viagra\n
```