

# Práctica Minería de Texto

---

CLASIFICACION LETRA CANCIONES TIPO REGGAETON

## INDICE

CLASIFICADOR DE CANCIONES REGGAETON .....	3
INTRODUCCIÓN .....	3
Búsqueda y filtrado de datos. ....	4
Generación de Corpus de letras. ....	6
Generación de Clasificador. ....	9
Generación de Clasificador más Detallado.....	17
Probando el algoritmo de clasificación. ....	20

# CLASIFICADOR DE CANCIONES REGGAETON

*Realizado por Araceli Macía Barrado*

## INTRODUCCIÓN

El objetivo de la práctica que he realizado es identificar las canciones que por la letra son de tipo Reggaetón, entiendo por Reggaetón las canciones con contenido abiertamente sexual, violento o relacionado con drogas.

Como he comentado en el PPT que he entregado, me llama la atención que en general se preste cuidado y se califique todo contenido audiovisual, y sin embargo no se tenga en cuenta la misma protección a los menores sobre las canciones, cuando este tipo de canciones se pueden escuchar en cualquier sitio público, como centros comerciales o establecimientos o lugares propios de menores como ferias, parques de atracciones, etc.

## BÚSQUEDA Y FILTRADO DE DATOS.

Para poder realizar una clasificación de los texto aptos y no aptos (todo sobre mi punto de vista), lo primero es buscar canciones y obtener datos para poder enseñar a un clasificador. Para ello, buscando por Internet, encontré esta página:

<http://www.sonicomusica.com>



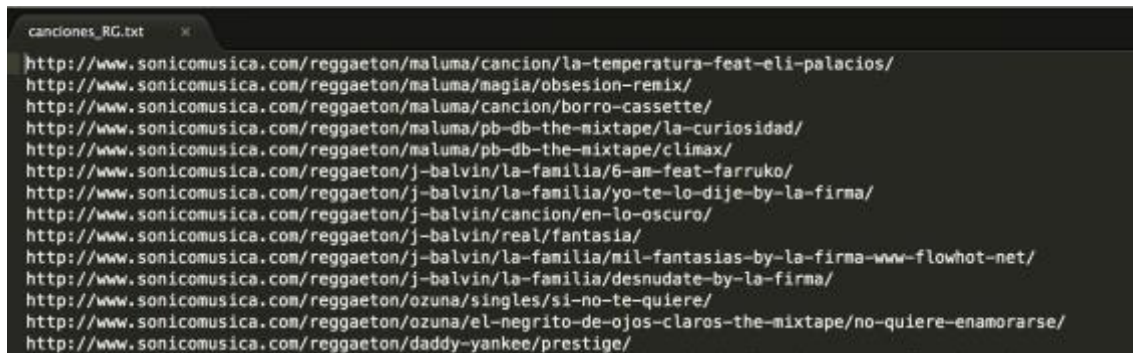
Donde están publicadas las letras de las canciones de diferentes artistas.

Tras revisar y leer canciones, realice dos ficheros de texto (entregados con la práctica)

- Canciones\_NR.txt
- canciones\_RG.txt

Con canciones de tipo Reggaetón y canciones que no lo son. De modo que cada uno de los ficheros contiene la URL de la página donde se encuentra la letra de la canción.

Visualización de canciones que he tipificado como Reggaetón:



```
canciones_RG.txt
http://www.sonicomusica.com/reggaeton/maluma/cancion/la-temperatura-feat-eli-palacios/
http://www.sonicomusica.com/reggaeton/maluma/magia/obsesion-remix/
http://www.sonicomusica.com/reggaeton/maluma/cancion/borro-cassette/
http://www.sonicomusica.com/reggaeton/maluma/pb-db-the-mixtape/la-curiosidad/
http://www.sonicomusica.com/reggaeton/maluma/pb-db-the-mixtape/climax/
http://www.sonicomusica.com/reggaeton/j-balvin/la-familia/6-am-feat-farruko/
http://www.sonicomusica.com/reggaeton/j-balvin/la-familia/yo-te-lo-dije-by-la-firma/
http://www.sonicomusica.com/reggaeton/j-balvin/cancion/en-lo-oscuro/
http://www.sonicomusica.com/reggaeton/j-balvin/real/fantasia/
http://www.sonicomusica.com/reggaeton/j-balvin/la-familia/mil-fantasias-by-la-firma-www-flowhot-net/
http://www.sonicomusica.com/reggaeton/j-balvin/la-familia/desnudeate-by-la-firma/
http://www.sonicomusica.com/reggaeton/ozuna/singles/si-no-te-quiere/
http://www.sonicomusica.com/reggaeton/ozuna/el-negrito-de-ojos-claros-the-mixtape/no-quiere-enamorarse/
http://www.sonicomusica.com/reggaeton/daddy-yankee/prestige/
```

Visualización de canciones que he tipificado como No Reggaetón:



```
Canciones_NR.txt
http://www.sonicomusica.com/flamenco/el-barrio/cancion/he-vuelto/
http://www.sonicomusica.com/flamenco/el-barrio/toda-una-decada/llovera/
http://www.sonicomusica.com/flamenco/el-barrio/toda-una-decada/yo-ya-no-creo-en-el-amor/
http://www.sonicomusica.com/flamenco/el-barrio/las-playas-de-invierno/ausencia/
http://www.sonicomusica.com/salsa/marc-anthony/3-0/vivir-mi-vida/
http://www.sonicomusica.com/salsa/marc-anthony/3-0/cambio-de-piel/
http://www.sonicomusica.com/salsa/marc-anthony/valio-la-pena/valio-la-pena/
http://www.sonicomusica.com/salsa/marc-anthony/amar-sin-mentiras/ahora-quien/
http://www.sonicomusica.com/salsa/marc-anthony/amar-sin-mentiras/amar-sin-mentiras/
http://www.sonicomusica.com/salsa/marc-anthony/sigo-siendo-yo-grandes-exitos/no-me-ames/
http://www.sonicomusica.com/salsa/marc-anthony/sigo-siendo-yo-grandes-exitos/dimelo/
http://www.sonicomusica.com/salsa/marc-anthony/todo-a-su-tiempo/y-sigues-siendo-tu/
http://www.sonicomusica.com/pop/shakira/pies-descalzos/estoy-aqui/
http://www.sonicomusica.com/pop/shakira/shakira/nunca-me-acuerdo-de-olvidarte/
http://www.sonicomusica.com/pop/shakira/loba/gitana/
http://www.sonicomusica.com/romantica/ricky-martin/a-medio-vivir/bombon-de-azucar/
http://www.sonicomusica.com/romantica/ricky-martin/ricky-martin-el-amor-de-mi-vida/
http://www.sonicomusica.com/romantica/ricky-martin/a-medio-vivir/te-extrano-te-olvido-te-amo/
```

En total, he seleccionado 140 canciones de cada tipo.

## GENERACIÓN DE CORPUS DE LETRAS.

Una vez que tengo los dos ficheros con las URLs, he accedido a cada URL para leer el HTML y guardar la letra de las canciones en un fichero de texto por canción.

El código que he realizado para hacer esto se encuentra en: [PT\\_MinDatos\\_Reg\\_1.ipynb](#)

- a) Generación de los directorios donde almacenar las canciones.

```
#Generacion y eliminacion de directorios para dejar los textos de las canciones.
% ls
% mkdir FicDatos/NO_REG
% mkdir FicDatos/REG

% rm -rf FicDatos/NO_REG/*
% ls FicDatos/NO_REG/
% rm -rf FicDatos/REG/*
% ls FicDatos/REG/
```

```
1-LecturaDatos-Copy1.ipynb      PT_MinDatos_Reg_1.ipynb
Canciones_RAnterior.txt        PT_MinDatos_Reg_2.ipynb
FicDatos/                      Prueba Minería de Datos..ipynb
PT_MinDatos_Reg_1-Copy1.ipynb  ejemplos.py
```

- b) Leo los ficheros de datos que he generado a mano, que se encuentran en el directorio FicDatos y genero un dataframe con una sola columna llamada url que contiene la URL a leer.

```
dfNR = pd.read_csv('FicDatos/Canciones_NR.txt', sep=" ", header=None, names=["url"])
dfRG = pd.read_csv('FicDatos/Canciones_RG.txt', sep=" ", header=None, names=["url"])
```

Leo cada una de las columnas y lo almaceno como una serie en una variable.

```
datosLeer1=dfNR.url #[1:]
datosLeer2=dfRG.url #[1:]
```

- c) He creado una función para leer las letras de las canciones y almacenarlas

```
def GenerarDatos(dt, ruta): #Función de recuperación de los textos.
    xpath_expr='//pre/text()' #Patron para encontrar el texto dentro del html descargado.
    i=1
    fic=ruta
    for ul in dt: #Se accede a cada URL pasada por parametro.
        response = requests.get(ul)
        content = response.text.encode('utf-8') #Codifico el texto a utf-8
        tree = html.fromstring(content)
        results = tree.xpath(xpath_expr)
        if len(results)>0: #Por si acaso no se encuentra la etiqueta en el html.
            tfile = open(fic + str(i) + '.txt', 'w+')
            tfile.write("%s\n" % results[0].encode('utf-8'))
            tfile.close()
            time.sleep(2) #Introduzco un delay, para evitar que me restrijan el acceso desde la página, por sobreaccesos..
            i=i+1
```

#### d) Ejecución de la función

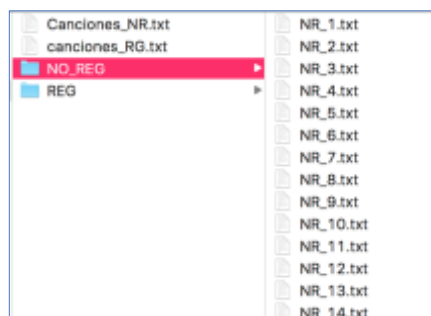
```
rutal="FicDatos/NO_REG/NR_"
ruta2="FicDatos/REG/REG_"
GenerarDatos(datosLeer1, rutal)
GenerarDatos(datosLeer2, ruta2)
```

#### e) Resultado de la ejecución

Como resultado de la ejecución de este Notebook, tendré dos directorios:

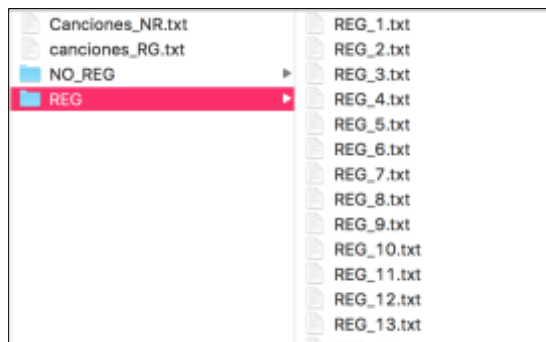
- ✚ FicDatos/NO\_REG: Fichero de letras de canciones No reggaeatón, cuyo nombre sigue el siguiente patrón:

NR\_(numero Canción).txt



- ✚ FicDatos/REG: Fichero de letras de canciones regueatón, cuyo nombre sigue el siguiente patrón:

REG\_(numero Canción).txt



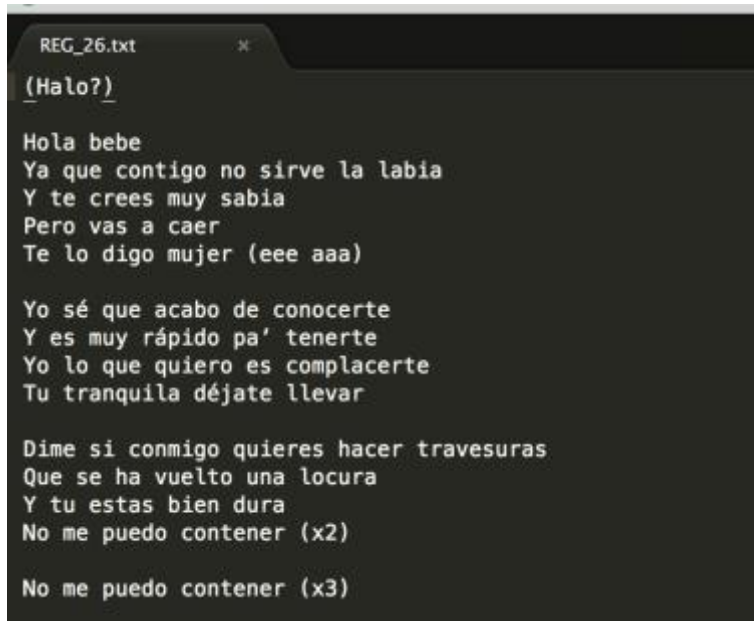
### Ejemplo de Contenidos de los ficheros generados:

*No reggaetón:*

```
NR_19.txt
Sabes, no es sólo recordar el dolor,
sabes, son frases dentro de una canción,
que quedan los momentos, que quedan los recuerdos
que vivimos de amor.
Sabes, el tiempo ha regresado el reloj, sabes,
tus manos cubren mi corazón,
y no quiero olvidarte, y no puedo cambiarte,
seras siempre el mejor.
Porque llevas en la sangre, ansiedad por abrazarme
por regresarme, sueños que no podran ser.
Porque llevo yo en la sangre ansiedad por abrazarte, nunca soltarte,
hasta morir junto a ti, junto a ti.
Sabes, tus cartas siempre estan junto a mi, ooh,
sabes, mi héroe abrió la salas, voló,
dejó trozos del alma, dejó brillos de magia de este eterno amor.
Sabes, tal vez siempre tuviste razón, ooh,
sabes, el mundo no era para este amor,
nos toca y adelante, y hasta el ultimo instante de mi vida te amaré.
Porque llevas en la sangre, ansiedad por abrazarme,
por regresarme, sueños que no podran ser.
Porque llevo yo en la sangre ansiedad por abrazarte
nunca soltarte, hasta morir junto a ti
Porque llevas en la sangre, ansiedad por abrazarme,
por regresarme, sueños que no podran ser.
Porque llevo yo en la sangre ansiedad por abrazarte, nunca soltarte,
hasta morir junto a ti, junto a ti. Sangre
```



*Reggaetón:*



```
REG_26.txt
(Halo?)

Hola bebe
Ya que contigo no sirve la labia
Y te crees muy sabia
Pero vas a caer
Te lo digo mujer (eee aaa)

Yo sé que acabo de conocerte
Y es muy rápido pa' tenerte
Yo lo que quiero es complacerte
Tu tranquila déjate llevar

Dime si conmigo quieres hacer travesuras
Que se ha vuelto una locura
Y tu estas bien dura
No me puedo contener (x2)

No me puedo contener (x3)
```

## GENERACIÓN DE CLASIFICADOR.

El código de esta parte se encuentra en el notebook: **PT MinDatos Reg 2.ipynb**

Para realizar la clasificación he realizado las siguientes funciones.

a) Funciones generadas.

- Función para recuperar todas las palabras de un fichero dado.

Recibe como parámetros la ruta y el nombre del fichero.

Lee el fichero y lo pasa a Unicode.

Utilizo la función de Word\_tokenize de nltk para convertir texto a palabras.

Devuelve una lista de tokens.

```
def words_file(ruta, fic): #devolver las palabras de cada fichero.
    ficLeido=open(ruta+fic, 'r')
    try:
        texto=ficLeido.read().decode('utf-8')
        tokens =nltk.word_tokenize(texto) #tokens=['a','b','c'] lo dejo como unicode.
        ficLeido.close()
        return tokens
    except UnicodeDecodeError:
        print "error en", ruta+fic
        return ["---"]
```

Ejemplo de llamada y salida de la función:

```
print words_file("FicDatos/REG/", "REG_1.txt")

[u'['', u'Eli', u'Palacios', u{'', u'x2', u'}', u']', u'La', u'estoy', u'calentando', u'La', u'estoy', u'provocando',
u'Pa', u'', u'que', u'suba', u'suba', u'Pa', u'', u'que', u'suba', u'suba', u'la', u'temperatura', u'Estoy', u'bus
```

- Función para leer los ficheros de un directorio.

Recibe como parámetro el directorio y el patrón de los ficheros.

Devuelve una lista de ficheros que siguen el patrón.

```
#funcion para leer los ficheros del directorio.
def ficheros(ruta2,pa):
    tot=[]
    for fileid in listdir(ruta2):
        if fnmatch.fnmatch(fileid, pa):
            tot.append(fileid)
    return tot
```

Ejemplo de llamada y salida:

```
print ficheros("FicDatos/REG/", "REG*.txt")

['REG_1.txt', 'REG_10.txt', 'REG_100.txt', 'REG_101.txt', 'REG_102.txt', 'REG_103.txt', 'REG_104.txt', 'REG_105.txt',
'REG_106.txt', 'REG_107.txt', 'REG_108.txt', 'REG_109.txt', 'REG_11.txt', 'REG_110.txt', 'REG_111.txt', 'REG_112.tx
t', 'REG_113.txt', 'REG_114.txt', 'REG_115.txt', 'REG_116.txt', 'REG_117.txt', 'REG_118.txt', 'REG_119.txt', 'REG_12.
```

- Función para recuperar todas las palabras de los ficheros.

Recibe como entrada una lista formada por tuplas de tipo (directorio, patrón)

Lee todos los ficheros de los directorios con el patrón dado y genera una lista con las palabras.

Se usa la función de Word\_tokenize de nltk.

La lista se devuelve en formato Unicode.

```
def words_all_file(rutas):  #devolver las palabras de los ficheros. Devuelve una lista.
    tokens=[]

    for t in rutas:
        ruta=t[0]
        patron=t[1]

        for fileid in ficheros(ruta,patron):
            ficLeidol=open(ruta+fileid, 'r')
            try:
                texto=ficLeidol.read().decode('utf-8')
                tokens.extend(nltk.word_tokenize(texto))      #tokens=['a','b','c'] UNICODE
            except UnicodeDecodeError:
                print "error en", ruta+fileid
                return ["---"]
    return tokens
```

Llamada y salida de la función:

```
rutas=[("FicDatos/REG/", "REG*.txt"), ("FicDatos/NO_REG/", "NR*.txt")]
allwordsT=words_all_file(rutas) #recupero todas las palabras de los ficheros.
print "numero de palabras total", len(allwordsT) #Numero de palabras en total de todas las canciones.

numero de palabras total 104261

print allwordsT[1:100]

[u'Eli', u'Palacios', u('(', u'x2', u')', u')', u'La', u'estoy', u'calentando', u'La', u'estoy', u'provocando', u'Pa',
u'', u'que', u'suba', u'suba', u'Pa', u'', u'que', u'suba', u'suba', u'la', u'temperatura', u'Estoy', u'buscando',
u'Desatar', u'el', u'fuego', u'en', u'su', u'cintura', u'densa', u'y', u'dura', u'Pa', u'', u'que', u'suba', u'sub
```

- Función para generar las características(features) de los datos

Recibe como entrada un documento que contiene la lista de palabras de una canción.

Utiliza una variable que se llama Word\_Features que contiene las palabras más frecuentes de todo el corpus de canciones.

Genera una lista donde se indica si el documento pasado por parámetro contiene o no las palabras más utilizadas.

```
def document_features(document):  
    # de cada documento, que contine la lista de palabras, devuelvo si contiene o no las palabras encontradas filtradas.  
  
    document_words = set(document)  
    features = {}  
    for word in word_features:  
        w1=word.encode('utf-8') #convierto a utf8 para poder formatearlo.  
        features['contains({})'.format(w1)] = (word in document_words)  
    return features
```

Ejemplo de llamada y salida:

```
print document_features(documents[1][0])  
  
{'contains(asegura)': False, 'contains(tardado)': False, 'contains(aaaa)': False, 'contains(plastic)': False, 'contains(oficial)': False, 'contains(pobre)': False, 'contains(atroz)': False, 'contains(seguir)': False, 'contains(qu\xca9date)': False, 'contains(principio)': False, 'contains(actuado)': False, 'contains(morder\xca9xain)': False, 'contains(disguise)': False, 'contains(di)': False, 'contains(poderosa..)': False, 'contains(lambada)': False, 'contains(llevarte)': False, 'contains(kiss)': False, 'contains(cola)': False, 'contains(arm\xca9xb3nico)': False, 'contains(batida)': False, 'contains(huelo)': False, 'contains(cogas)': False, 'contains(viejo)': False, 'contains(di\xca9xb3s)': False, 'contains(pendientes)': False, 'contains(venio)': False, 'contains(motivao)': False, 'contains(omega)': False, 'contains(yandel)': False, 'contains(corta)': False, 'contains(representa)': False, 'contains(herizas)': False, 'contains(metiendole)': False, 'contains(carrera)': False, 'contains(muevelo\xca9xa2\xca9x80\xca9xa6)': False, 'contains(tu)': True, 'contains(circulo)': False, 'contains(comodo)': False, 'contains(veinte)': False, 'contains(desvestimos)': False, 'contains(parau)': False, 'contains(hablarnos)': False, 'contains(voz)': False, 'contains(novia)': False, 'contains(ablanda)': False, 'contains(blas)': False, 'contains(then)': False, 'contains(vengo)': False, 'contains(tentacion)': False, 'contains(diarias)': False, 'contains(anima)': False, 'contains(vaya)': False, 'contains(holgaza)'
```

a) Generación de clasificación.

Lo primero que hago es generar una lista de las canciones, donde cada canción sea una lista de palabras y un indicador de si es Reggaetón o no. Y luego las desordeno.

```

ruta1="FicDatos/REG/"
ruta2="FicDatos/NO_REG/"

#creo documento de tipos list: (tokensCANCION,"REG")
documents1 = [ (list(words_file(ruta1,fileid))), "REG")
               for fileid in ficheros(ruta1,"REG*.txt")]

#creo documento de tipos list: (tokensCANCION,"NO REG")
documents2 = [ (list(words_file(ruta2,fileid))), "NO_REG")
               for fileid in ficheros(ruta2,"NR*.txt")]

#Uno las listas, de forma que ahora tengo una super lista, donde cada elemento son las palabras de una cancion
#y un indicador de si es Regueton o no lo es.
documents= documents1 + documents2
np.random.shuffle(documents)

```

Ejemplo: documents tendrá esta estructura:

```

[(['palabra', 'canción', 'reguetón'], 'REG'),
 ([palabra, 'canción', 'no'], 'NO_REG')]

```

A continuación, recupero todas las palabras de todos los ficheros de todos los tipos, y genero una frecuencia para dejar ordenadas las palabras (sin repetir) asociando el número de veces que aparece.

Con eso genero una lista con 3000 palabras.

```

all_words = nltk.FreqDist(w.lower() for w in allwordsT) #frecuencia de cada palabra, de todos los documentos.
print "Numero de palabras total", len(allwordsT) #Numero de palabras en total de todas las canciones.
word_features = list(all_words)[:3000] #Me quedo con las 3000 palabras de la lista
print "numero de palabras sin repetir", len(all_words)

Numero de palabras total 104261
numero de palabras sin repetir 9179

```

Del total de palabras en los ficheros: 104.261, nos dice que hay 9179 sin repetir.

Por ejemplo, vamos a ver la lista de palabras más frecuentes de estas canciones:

```

for w, frequency in all_words.most_common(50):
    print(u'{}:{}'.format(w, frequency))

,r:4476
que:4046
y:2677
la:2535
soy:2176
me:1998
de:1983
te:1920
el:1852
a:1783
tu:1659
en:1396
yo:1392
(:1322
):1321
! :1188
mi:1119

```

Como se ve en la imagen, están saliendo signos de puntuación y artículos, ya que ahora voy a ejecutar el código sin “limpiar” nada. Me interesa ver que hace el clasificador con esto.

Genero la lista de features o características de los documentos (cada documento es una canción tokenizada y su indicador de tipo Reggaetón).

Y luego con estos datos, separa en dos conjuntos, uno de entrenamiento y otro para comprobar.

```

featuresets = [(document_features(d), c) for (d,c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]

classifier = nltk.NaiveBayesClassifier.train(train_set)

```

Hay que tener en cuenta que tenemos 280 canciones, con lo que los tamaños que manejamos son estos:

```

print len(featuresets)
print len (train_set)
print len (test_set)

280
180
100

```

Recordamos que la lista de features contiene si la canción contiene o no la lista de palabras más frecuentes.

```
print document_features(documents[1][0])
```

```
{'contains(asegura)': False, 'contains(tardado)': False, 'contains(aaaa)': False, 'contains(plastic)': False, 'contains(oficial)': False, 'contains(pobre)': False, 'contains(atroz)': False, 'contains(seguir)': False, 'contains(qu\x03\x09date)': False, 'contains(principio)': False, 'contains(actuado)': False, 'contains(morder\x03\x0ain)': False, 'contains(disguise)': False, 'contains(di)': False, 'contains(poderosa..)': False, 'contains(lambada)': False, 'contains(llevarte)': False, 'contains(kiss)': False, 'contains(cola)': False, 'contains(arm\x03\x0b3nico)': False, 'contains(batida)': False, 'contains(huelo)': False, 'contains(cogas)': False, 'contains(viejo)': False, 'contains(di\x03\x0b3s)': False, 'contains(pendientes)': False, 'contains(venio)': False, 'contains(motivao)': False, 'contains(omega)': False, 'contains(yandel)': False, 'contains(corta)': False, 'contains(representa)': False, 'contains(herizas)': False, 'contains(metiendole)': False, 'contains(carrera)': False, 'contains(muevelo\x03\x0a2\x02\x080\x02\x0a6)': False, 'contains(tu)': True, 'contains(circulo)': False, 'contains(comodo)': False, 'contains(veinte)': False, 'contains(desvestimos)': False, 'contains(parau)': False, 'contains(hablarnos)': False, 'contains(voz)': False, 'contains(novia)': False, 'contains(ablanda)': False, 'contains(blas)': False, 'contains(them)': False, 'contains(vengo)': False, 'contains(tentacion)': False, 'contains(diarias)': False, 'contains(animas)': False, 'contains(vaya)': False, 'contains(holgaza
```

Ejecutamos un clasificador para entrenarlo con los datos de entrenamiento.

```
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

Vamos a ver el grado de acierto con los datos de test.

```
print(nltk.classify.accuracy(classifier, test_set)) #Tiene un acierto bastante alto
0.85
```

Tiene un acierto bastante elevado, y eso que tenemos palabras y caracteres no válidos.

Estas son las palabras más significativas para realizar la clasificación:

Características mas Representativas:

contains(the) = True	REG : NO_REG =	9.7 : 1.0
contains(x2) = True	REG : NO_REG =	9.4 : 1.0
contains(llego) = True	REG : NO_REG =	7.7 : 1.0
contains(llama) = True	REG : NO_REG =	6.6 : 1.0
contains(fin) = True	NO_REG : REG =	6.6 : 1.0
contains(:) = True	NO_REG : REG =	6.6 : 1.0
contains(has) = True	NO_REG : REG =	6.2 : 1.0
contains(2) = True	REG : NO_REG =	5.7 : 1.0
contains(ve) = True	REG : NO_REG =	5.7 : 1.0
contains(silencio) = True	NO_REG : REG =	5.7 : 1.0
contains(sentido) = True	NO_REG : REG =	5.0 : 1.0
contains(mucha) = True	REG : NO_REG =	5.0 : 1.0
contains(rico) = True	REG : NO_REG =	5.0 : 1.0
contains(aquella) = True	REG : NO_REG =	5.0 : 1.0
contains(bailando) = True	REG : NO_REG =	4.6 : 1.0

Se puede ver que aun teniendo cosas extrañas como x2 (que es un indicador de que la frase de una canción se repite dos veces), y caracteres como (') el porcentaje de acierto es muy elevado.

Como ejemplo de lectura de estos datos, el porcentaje de que si aparece x2 la canción es de tipo Reggaetón es de 9.7

Aunque existe una función en nltk para generar la lista de las características, a veces funciona y a veces no (por los acentos), con lo que optado por reutilizado el código directamente.

```
cpdist = classifier._feature_probdist

print('Caracteristicas mas Representativas: ')

for (fname, fval) in classifier.most_informative_features(15):
    def labelprob(l):
        return cpdist[l, fname].prob(fval)

    labels = sorted([l for l in classifier._labels
                     if fval in cpdist[l, fname].samples()],
                    key=labelprob)
    if len(labels) == 1:
        continue
    l0 = labels[0]
    l1 = labels[-1]
    if cpdist[l0, fname].prob(fval) == 0:
        ratio = 'INF'
    else:
        ratio = '%8.1f' % (cpdist[l1, fname].prob(fval) /
                           cpdist[l0, fname].prob(fval))
    print((' %24s = %-14r %6s : %-6s = %s : 1.0' %
          (fname, fval, ("%s" % l1)[:6], ("%s" % l0)[:6], ratio)))
```



## GENERACIÓN DE CLASIFICADOR MÁS DETALLADO.

Una vez realizado el clasificador, ahora voy a filtrar más las palabras que forman el conjunto de palabras con más frecuentes, que son la base de las características para el algoritmo. Lo que voy a hacer es no tener en cuenta:

- Caracteres no alfanuméricos
- Palabras que forman parte del stopwords (corpus de nltk, lo forman artículos, determinantes, etc, es decir palabras que no son identificativas.
- Palabras que no estén en el diccionario español (corpus cess\_esp).

De esta forma, estaría eliminando las palabras anglosajonas y expresiones propias latinas de la letra reggaetón.

### ▪ Función para obtener todas las palabras de las canciones

Esta función recibe como entrada la tupla de directorio patrón, para leer todos los ficheros.

Salida: lista de palabras que son alfanuméricas, que no estén en stopwords y que estén en el corpus español.

```
from nltk.corpus import stopwords
stop=stopwords.words("spanish")
Spanish_vocab = set(w.lower() for w in nltk.corpus.cess_esp.words() )

def words_all_file_SinCaraSTOPDIC(rutas):    #devolver las palabras de los ficheros. Devuelve una lista.
    tokens=[]

    for t in rutas:
        rutal=t[0]
        patronl=t[1]

        for fileid in ficheros(rutal,patronl):
            ficLeidol=open(rutal+fileid, 'r')
            try:
                texto=ficLeidol.read().decode('utf-8')
                tl=nltk.word_tokenize(texto)
                for w in tl:
                    wo=w.lower()
                    if wo.isalpha(): #Solo considero palabras alfanumericas.
                        if wo not in stop: #solo considero las palabras que no estan en stopwords.
                            if wo in Spanish_vocab: #solo las que estan en el diccionario Español
                                #wo=wo.encode('utf-8')
                                tokens.append(wo)
            except UnicodeDecodeError:
                print "error en", ruta+fic
            ficLeidol.close()
            return ["---"]

    return tokens
```

Pantallazo de ejemplo de lo que stopwords contiene:

```
print stop

[u'de', u'la', u'que', u'el', u'en', u'y', u'a', u'los', u'del', u'se', u'las', u'por', u'un', u'para', u'con', u'n
o', u'una', u'su', u'al', u'lo', u'como', u'm\xels', u'pero', u'sus', u'le', u'ya', u'o', u'este', u's\xed', u'porqu
e', u'esta', u'entre', u'cuando', u'muy', u'sin', u'sobre', u'tambi\xen', u'me', u'hasta', u'hay', u'donde', u'quie
n', u'desde', u'todo', u'nos', u'durante', u'todos', u'uno', u'les', u'ni', u'contra', u'otros', u'ese', u'eso', u'an
te', u'ellos', u'e', u'esto', u\xed', u'antes', u'algunos', u'qu\xe', u'unos', u'yo', u'otro', u'otras', u'otra',
u'\xe9', u'tanto', u'esa', u'estos', u'mucho', u'quienes', u'nada', u'muchos', u'cual', u'poco', u'ella', u'estar',
u'estas', u'alguna', u'algo', u'nosotros', u'mí', u'mis', u't\xfa', u'te', u'tí', u'tu', u'tus', u'ellas', u'nosotr
as', u'vosotros', u'vosotras', u'os', u'm\xedo', u'm\xeda', u'm\xedos', u'm\xedas', u'tuyo', u'tuya', u'tuyos', u't
uyas', u'suyo', u'suya', u'suyos', u'suyas', u'nuestro', u'nuestra', u'nuestros', u'nuestras', u'vuestro', u'vuest
ra', u'vuestros', u'vuestras', u'esos', u'esas', u'estoy', u'est\xels', u'est\xel', u'estamos', u'est\xelis', u'est\x
e', u'est\xe9', u'est\xeís', u'estemos', u'est\xeis', u'est\xeín', u'estar\xels', u'estar\xel', u'est
aremos', u'estar\xeis', u'estar\xein', u'estar\xeda', u'estar\xedas', u'estar\xedamos', u'estar\xedais', u'estar\xed
```

Pantallazo de lo que Spanish\_vocab contiene:

```
print Spanish_vocab

fica', u'asistido', u'gays', u'\xe9mbitos', u'infelizmente', u'falsa', u'copacabana', u'masochismo', u'cobertura',
u'falso', u'volver\xel', u'impedida', u'referirme', u'beat\xedx mendoza', u'dijeron', u' anotaciones', u'cuatro',
u'\xedndices', u' racing de santander', u'\xcelciles', u'1 de enero del 2000', u'fernando navas', u'cuentos de vacaci
ones', u'manos', u'sobrenatural', u'pajolero', u'joe's', u'respirarlo', u'de un momento a otro', u'enfatiz\xed', u'mi
n.69', u'deportivistas', u'producirse', u'liofilizado', u'hotelero', u'\xedber', u'obcecada', u'seleccionadores',
u'2.133.297', u'ch\xelndal', u'is\xítopos', u'faenar', u'minas gerais', u'terminar', u'cadencia', u'jab', u'exporta
do', u'patrocinio', u'atrayentes', u'satisface', u'ira', u'james watt', u'polarizar', u'zplidos', u'dominical', u'det
erminante', u'cruz\xí', u'plaza de armas', u'complejas', u'perdido', u'perdida', u'privatizaci\xí', u'faltos', u's
inceras', u'parecerse', u'nuclear', u'parque', u'posse', u'destino', u'tranquilizar', u'solicitante', u'holsen', u'des
tina', u'esplendor', u'enviaban', u'atenciones', u'facultades', u'guayana', u'paul ricoeur', u'puerto montt', u'priva
dos', u'presidencia de la rep\xfablica', u'marte', u'indica', u'retransmita', u'bautizada', u'pirrol', u'balaguer',
u'bautizado', u'350.000', u'decidido', u'decidida', u'precipitar\xel', u'joaqu\xedx elcome', u'opciones', u'circu
ndantes', u'semestre', u'hinchada', u'hizo gala', u'caficultores', u'fondos', u'autom\xeltica', u'situaciones', u'amo
nest\xí', u'diecinueve', u'sobras', u'armí\xí', u'indic\xí', u'econ\xí', u'driblado', u'devora', u'beneficia
rse', u'conseguir\xel', u'econ\xí', u'premisas', u'mostrado', u'deber\xeda', u'aprobaron', u'internauta', u'ind
onesia', u'cabelludo', u'contribuyentes', u'inminente', u'qu\xe', u'vecindario', u'geograf\xeda', u'complicada', u'c
omplicado', u'web', u'alimentaria', u'sodomizada', u'entroncar', u'entereza', u'piel', u'tica', u'alineamiento', u'fe
```

Repetimos los pasos de la ejecución que hemos visto anteriormente, pero esta vez ejecutamos la función words\_all\_file\_SinCaraSTOPDIC en vez de words\_all\_file

```
rutas=[("FicDatos/REG/", "REG*.txt"), ("FicDatos/NO_REG/", "NR*.txt")]

allwordsSIN=words_all_file_SinCaraSTOPDIC(rutas) #recupero todas las palabras de los ficheros.

print "Numero de palabras total", len(allwordsSIN) #Numero de palabras en total de todas las canciones.

all_words = nltk.FreqDist(w.lower() for w in allwordsSIN) #frecuencia de cada palabra, de todos los documentos.
word_features = list(all_words)[:3000] #Me quedo con las 3000 palabras de la lista
print "numero de palabras sin repetir", len(all_words)

Numero de palabras total 27553
numero de palabras sin repetir 2972
```

Con el filtro realizado el número de palabras ha descendido de 104.261 a 27.553, con lo que ahora las consideramos todas para obtener las características de cada documento.

```
featuresets = [(document_features(d), c) for (d,c) in documents]
train_set, test_set = featuresets[100:], featuresets[:100]
```

```
classifier = nltk.NaiveBayesClassifier.train(train_set)
print(nltk.classify.accuracy(classifier, test_set)) #Tiene un acierto superior al anterior.
#classifier.show_most_informative_features(10) #las 10 palabras mas identificativas

0.92
```

El porcentaje de aciertos en la clasificación ha subido notablemente, y ahora vemos que las 10 palabras más identificativas son palabras y no caracteres, y además son todas españolas.

Características mas Representativas:		
contains(duro) = True	REG : NO_REG =	11.0 : 1.0
contains(vida) = True	NO_REG : REG =	9.2 : 1.0
contains(alma) = True	NO_REG : REG =	9.0 : 1.0
contains(ropa) = True	REG : NO_REG =	7.7 : 1.0
contains(haciendo) = True	REG : NO_REG =	7.7 : 1.0
contains(par) = True	REG : NO_REG =	7.7 : 1.0
contains(corazón) = True	NO_REG : REG =	6.7 : 1.0
contains(fin) = True	NO_REG : REG =	6.6 : 1.0
contains(hombre) = True	REG : NO_REG =	6.6 : 1.0
contains(llama) = True	REG : NO_REG =	6.6 : 1.0
contains(ritmo) = True	REG : NO_REG =	6.3 : 1.0
contains(pa) = True	REG : NO_REG =	6.3 : 1.0
contains(gusta) = True	REG : NO_REG =	6.1 : 1.0
contains(disco) = True	REG : NO_REG =	5.9 : 1.0
contains(demasiado) = True	NO_REG : REG =	5.7 : 1.0

## PROBANDO EL ALGORITMO DE CLASIFICACIÓN.

Para probarlo, voy a introducir la letra de dos canciones:

### A) Canción de Alex Ubago:

```
In [32]: #Vamos a probar con una canción de Alex Ubago.
cancion="Si ayer tuviste un día gris, tranquila, yo haré canciones para ver si así consigo acerte sonreír, si \
lo q quieres es huir, camina, yo haré canciones para ver, si así consigo fuerzas pa' vivir... No tengo mas motivos\
para darte que este miedo que me dá, el no volver a verte, nunca más... Creo ver la lluvia caer en mi ventana te veo \
pero no está lloviendo no es más que un reflejo de mi pensamiento, hoy te echo de menos... Yo sólo quiero hacerte \
saber amiga estes donde estes que si te falta el aliento yo te lo daré, y si te sientes sólo hablame, que te estaré \
escuchando aunque no te pueda ver... aunque no te pueda ver... De tantas cosas que perdí diría que sólo guardo lo que \
fue magico tiempo que nació en abril, miradas tristes sobre mi se anidan y se hacen parte de mi ser y ahora siempre \
llueve por que estoy sin ti... para darte que esta fría soledad, que necesito darte tantas cosas más... Creo ver la \
lluvia caer en mi ventana te veo pero no está lloviendo no es mas que un reflejo de mi pensamiento hoy te echo de \
menos... Yo sólo quiero hacer saber amiga estes donde estes que si te falta aliento yo te lo daré..."

In [33]: cancionSubago=nltk.word_tokenize(cancion.decode('utf-8'))

In [34]: classifier.classify(document_features(cancionSubago))

Out[34]: 'NO_REG'
```

El clasificador me dice que la canción de Alex Ubago **no es** Reggaetón. Seria apta para menores.

### B) Canción de Como Yo Le Doy ft. Don Miguelo - Pitbull

```
letra2="ella no ta' enamora' de mi Yo tampoco Pero le gusta como yo le doy Yo la pongo a volar Cuando yo \
le doy besos Pero no ta' enamora' de mi Ella solamente quiere pasar un momento lunático \
She said she's not in love with me, no! No! But she loves how I give it to her. Yes! Yes! Sexy! \
And I blow her mind! Every time that I kiss her But she said she's not in love with me. \
Yeah right! Ella siempre me llama a las 3AM Dice que soy su pana El que le quita las ganas \
Ha ha ha! Y si ella supiera que me llama su hermana Porque tambien soy su pana \
El que le quita las ganasWho the hell is this? Texting me at 3:46AM It must be a trick \
And if she texting me at this time She's looking for a Richard Or should I say Dick \
Mami yo te doy pao! pao! Ella dice que yo soy mal hablao \
No mamita yo no soy mal hablao Que yo te hablo directo de Miami al Cibao So deja el papelazo \
Para los turistas y los payasos Hablame claro que tu sabes bien Que te gusta lo malo Dale palo! \
Yo te doy caramelo Pero deja los celos Ella dice que no esta enamorada de mi \
Eso me conviene Don Miguelo Ella no ta' enamora' de mi no! no! \
Pero le gusta como yo le doy. si! si! Yo la pongo a volar Cuando yo le doy besos \
Pero no ta' enamora' de mi no! no! She said she's not in love with me, no! No! \
But she loves how I give it to her. Yes! Yes! Sexy! And I blow her mind! \
Every time that I kiss her But she said she's not in love with me. Yeah right. \
Casi siempre me llama a las 3AM Dice que soy su pana El que le quita las ganas \
Yo nunca la dejo a medias Tu entiende el drama Dice que soy su pana \
El que le quita las ganas Hemos hecho de to' Muchas posiciones en la escalera \
Amarrao en la cama Yo encima de ella Amanecimos en la bañera \
Y me la comi embarra de nutella Son requisitos que otros no nan usado \
Y ella solo dice wow Lo que yo hago nunca se ha inventado \
Porque ni el kanasutra Se lo ha inventado Oiga! \
Ella no ta' enamora' de mi no! no! Pero le gusta como yo le doy. si! si! \
Yo la pongo a volar Cuando yo le doy besos Pero no ta' enamora' de mi no! no! "
```

```
cancionR=nltk.word_tokenize(letra2.decode('utf-8'))
classifier.classify(document_features(cancionR))

'REG'
```

El clasificador me dice que **es** Reggaetón, así que esta canción no es apta para menores.