1. UNIVERSIDAD REY JUAN CARLOS
2. MÁSTER DATA SCIENCE
3. BÚSQUEDA Y RECUPERACIÓN DE INFORMACIÓN
4. PRÁCTICA DE DESAMBIGUACIÓN DE NOMBRES
5. **Javier Cano Montero**
6. **Javier González Méndez**

Sumario

[Objetivo 3](#__RefHeading___Toc240_1684846908)

[Inicial ordenado 3](#__RefHeading___Toc247_81970679)

[Quitando “Thomas Baker” 3](#__RefHeading___Toc249_81970679)

[Quitando stop-words 4](#__RefHeading___Toc251_81970679)

[Etiquetado gramatical. Quitando otras palabras “vacías”. 4](#__RefHeading___Toc253_81970679)

[Lematización 4](#__RefHeading___Toc255_81970679)

[Stemming 5](#__RefHeading___Toc475_81970679)

[Stemming Porter 5](#__RefHeading___Toc257_81970679)

[Stemming Snowball 5](#__RefHeading___Toc259_81970679)

[Stemming Lancaster 5](#__RefHeading___Toc261_81970679)

[TF-IDF 6](#__RefHeading___Toc477_81970679)

[TF-IDF Lematización 6](#__RefHeading___Toc263_81970679)

[TF-IDF Porter 6](#__RefHeading___Toc265_81970679)

[Conclusiones 7](#__RefHeading___Toc346_1684846908)

# Objetivo

El objetivo de la práctica consiste en modificar un código Python para desambiguación de nombres. El código realiza inicialmente una lectura de los textos de un directorio y construye con ellos un diccionario que se utiliza posteriormente para codificar estos textos usando la frecuencia de aparición de términos.

A continuación se muestran las modificaciones que se han realizado al código y qué resultados se han obtenido mediante estas modificaciones.

# Inicial ordenado

Comenzamos ordenando la lectura de los ficheros para que correspondan con el orden de la referencia y poder calcular el “random score” original.

En este experimento únicamente utilizamos un diccionario de palabras en minúsculas para codificar los textos que queremos clasificar.

|  |
| --- |
| 1. Final unique terms: 4257 2. Vectors created. 3. test: 4. [2, 2, 2, 0, 2, 0, 2, 2, 1, 1, 2, 2, 1, 2, 2, 2, 2, 3, 2] 5. reference: 6. [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1] 7. rand\_score: -0.16375 |

# Quitando “Thomas Baker”

Como las palabras “thomas” y “baker sabemos que van a ser constantes en todos nuestros textos podemos eliminarlas del diccionario porque no nos van a aportar información para diferenciar un “Thomas Baker” de otro.

|  |
| --- |
| Final unique terms: 4255  Vectors created.  test:  [0, 0, 0, 2, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 3, 0]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: -0.130111524164 |

Como podemos ver, la clasificación a mejorado ligeramente.

# Quitando stop-words

A continuación eliminamos las stop-words del diccionario porque son palabras que no aportan información relevante.

|  |
| --- |
| Final unique terms: 4132  Vectors created.  test:  [2, 2, 2, 0, 2, 0, 3, 2, 1, 1, 2, 2, 1, 2, 2, 2, 1, 2, 2]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: -0.0507482108003 |

Se puede observar que eliminar las stop-words mejora la clasificación.

# Etiquetado gramatical. Quitando otras palabras “vacías”.

Utilizando el etiquetador gramatical de NLTK, eliminamos otras palabras que consideramos vacías ["CD","JJ","JJR","JJS","MD","NN","NNP","NNPS","NNS", "PDT","PRP","PRP$","RB","RBR","RBS","VB","VBD","VBG","VBN", "VBP","VBZ"]

|  |
| --- |
| Final unique terms: 4048  Vectors created.  test:  [0, 2, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 3, 2]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: 0.40625 |

Eliminar estos tipos de palabras que consideramos “vacías” mejora mucho la clasificación, ya que nos quedamos con las palabras que tienen mucha información.

# Lematización

A continuación aplicamos una lematización a los términos del diccionario y a los textos.

|  |
| --- |
| Final unique terms: 3675  Vectors created.  test:  [0, 3, 2, 0, 0, 0, 1, 0, 0, 0, 2, 0, 0, 1, 0, 3, 0, 0, 3]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: 0.688115561392 |

Al usar la lematización estamos reduciendo el diccionario y eliminando algunas referencias que se pueden considerar duplicadas (singular y plural). Podemos ver que la clasificación mejora significativamente.

# Stemming

Alternativamente a la lematización, se puede aplicar un algoritmo de stemming. Este algoritmo busca la raíz de la palabra y la reduce a su forma más básica, de esta forma se simplifica el diccionario de codificación. La librería NLTK implementa tres versiones de algoritmos de stemming.

## Stemming Porter

|  |
| --- |
| Final unique terms: 3452  Vectors created.  test:  [0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 3, 0, 1, 0, 0, 1]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: 0.5725 |

## Stemming Snowball

|  |
| --- |
| Final unique terms: 3439  Vectors created.  test:  [0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 3, 0, 1, 0, 0, 1]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: 0.5725 |

## Stemming Lancaster

|  |
| --- |
| Final unique terms: 3187  Vectors created.  test:  [1, 1, 1, 0, 1, 0, 1, 1, 3, 3, 1, 1, 3, 1, 0, 1, 3, 2, 1]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: -0.134328358209 |

Se observa que el stemming mejora la clasificación respecto a la lematización para Porter y Snowball, y no funciona para Lancaster.

# TF-IDF

Alternativamente a la codificación de documentos mediante la frecuencia de aparición de términos (TF), se puede usar la frecuencia inversa de documentos (TF-IDF).

## TF-IDF Lematización

Usando TF-IDF y lematización el random score es igual que el obtenido con stemming y TF, sin embargo el número de aciertos en clasificación es mayor que el obtenido para ese experimento.

|  |
| --- |
| Final unique terms: 3671  Vectors created.  test:  [0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 3, 0, 1, 0, 0, 1]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: 0.5725 |

El resultado es igual que el mejor obtenido hasta ahora con el stemming Porter o Snowball.

## TF-IDF Porter

Usando TF-IDF y stemming Porter el resultado es igual que el obtenido con lematización y TF.

|  |
| --- |
| Final unique terms: 3437  Vectors created.  test:  [0, 1, 3, 0, 0, 0, 2, 0, 0, 0, 3, 0, 0, 2, 0, 1, 0, 0, 1]  reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]  rand\_score: 0.688115561392 |

# Conclusiones

Se ha diseñado un algoritmo a partir de un código previo ya proporcionado que permite realizar desambiguación de nombres basándose en el contenido de los textos que se estudian.

De las distintas versiones que se han desarrollado, se observa que eliminando palabras comunes a todos los textos y que no diferencian uno de otro ayuda a la clasificación. También se han aplicado algoritmos de lematización y de stemming. Se puede observar que los resultados obtenidos con lematización son ligeramente mejores que los resultados obtenidos con stemming. Además usando la codificación con la frecuencia inversa de documentos (TF-IDF) en vez de la frecuencia de términos (TF) no se observa cambios significativos.