Práctica: People Name Disambiguation

Búsqueda y Recuperación de Información

Autores

Raúl Sánchez Martín

Ignacio Arias Barra

Índice

1. Introducción

La presente práctica se desarrolla dentro de la asignatura de *Búsqueda y Recuperación de Información* del *Máster en Data Science* de la Universidad Rey Juan Carlos. Inicialmente, se disponen de 19 textos correspondientes a 4 personas diferentes, todas ellas llamadas *Thomas Baker*. El objetivo de la presente práctica consiste en agrupar dichos textos en diferentes *clústers* intentado obtener que, en cada uno de ellos, sólo aparezcan textos referentes a la misma persona. A lo largo de la práctica se van a proponer e implementar diferentes representaciones de los textos, estudiando el efecto que tiene cada propuesta en el resultado final. Dado que la agrupación fidedigna de los textos es conocida, se va a poder evaluar la exactitud de cada mejora propuesta.

La estructura del presente documento es como sigue. Después de la Introducción, se especifican los requisitos previos necesarios para poder ejecutar los códigos propuestos. A continuación, se describe el punto de partida así proporcionado en el propio enunciado de la práctica. Posteriormente, se describen todas las diferentes opciones propuestas para la representación de los textos, incluyendo posibles combinaciones de las mismas. Después, se incluyen todos los comandos necesarios para la ejecución de las propuestas anteriormente descritas. Una vez que ya se han obtenido todos los resultados, se procede a su evaluación. Finalmente, se incluyen unas conclusiones.

2. Requisitos previos

La presente práctica se realizará utilizando `Python 3.5`. Además de un importante número de librerías de Python ya preinstaladas, también se hará uso de las librerías `nltk`, `sklearn` `matplotlib` y `seaborn`, las cuales han de ser instaladas en nuestro entorno `Python` de manera específica. Por otro lado, la librería `nltk` utiliza una serie de archivos cuya ruta ha de ser espcificada, para cada máquina, por medio de la variable `path_to_apend`, en el código incluido más abajo. Finalmente, se hará uso del Stanford Named Entity Recognizer. Esta librería, implementada en *Java*, es capaz de realizar reconocimientos de entidades de una manera óptima. Para su correcto funcionamiento, se ruega al lector que configura su máquina tal y como se especifica en el siguiente link: Configuring Stanford Parser and Stanford NER Tagger with NLTK in python on Windows and Linux. A continuación, se incluyen las librerías necesarias para el resto de la práctica.

```
import re, pprint, os, numpy
import nltk
from nltk import ngrams
from nltk.collocations import *
import string
from nltk.corpus import stopwords
path_to_append = '/media/nacho/f8371289-0f00-4406-89db-d575f3cdb35e/Master/Trimestre 2/
RIM/nltk_data'
path_to_append = '/media/raul/Data/nltk_data'
path_to_append = '/home/raul/nltk_data'
nltk.data.path.append(path_to_append)
from sklearn.metrics.cluster import *
from sklearn.cluster import AgglomerativeClustering, KMeans, MiniBatchKMeans
from nltk.cluster import GAAClusterer
from sklearn.metrics.cluster import adjusted_rand_score
```

```
from nltk.corpus import stopwords
import operator
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import warnings
warnings.filterwarnings('ignore')
from nltk.tag import StanfordNERTagger
from nltk.stem.porter import PorterStemmer
from nltk.stem import SnowballStemmer
import csv
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn
%matplotlib inline
```

3. Punto de partida

Como se indica en el propio enunciado de la práctica, no se parte desde cero sino que parte del código inicial para la ejecución de la presente práctica ya ha sido proporcinado. En concreto, se incluyen tres funciones:

```
1) read_file: Función utilizada para la lectura de los texto
s. Esta función no ha sido modificada.
```

```
2) TF: Función utilizada para la vectorización de un texto.
Esta función no ha sido modificada.
```

```
3) cluster_texts: Función que, dado una colección de textos,
en primer lugar los vectoriza y posteriormente los agrupa ut
ilizando diferentes algoritmos de clustering. Esta función h
a sido modificada respecto a la proporcionada inicialmente p
ara poder elegir que algorítmo de clustering se utiliza.
```

Utilizando las tres funciones anteriormente descritas, en el enunciado de la propia práctica se realiza un clustering inicial de los textos cuyo resultado ha sido guardado bajo la clave `primitive`.

4. Modificaciones propuestas

A continuación se van a detallar todas y cada una de las diferentes representaciones propuestas en la presenta práctica. Por un lado, se explicará cada una de ellas y se justificará el efecto esperado. Además, también se ha tenido en cuenta diferentes combinaciones de las mismas.

Dado que para ciertas transformaciones de texto es necesario saber a priori el lenguaje del mismo, se ha implementado la función `get_language`, la cual detecta el lenguaje entre inglés y español del texto introducido.
Otra función que se ha utilizado en diferentes ocasiones a lo largo de la presente práctica es `delete_words_from_text`. Dicha función, elimina de un texto inicial todas las palabras espeficiadas.

4.1 Modificaciones individuales

4.1.1 Filtrado por tipo de palabras

Esta modificación permite filtrar un texto inicial e incluir en la salida sólo unos tipos de palabra determinado. Por ejemplo, se especificará que sólo se incluyan sustantivos, adjetivos, advervios, etc. Es esperable que los sustantivos tengan más efecto a la hora de clasificar los textos que los artículos o adjetivos. Por ello, se espera que se mejore la solución inicial. Esta transformación, implementada por medio de las funciones `get_named_entities_1` y `get_named_ent_txts_1`, y sus resultados han sido guardados utilizando la clave `identity_analysis_1`.

4.1.2 Entidades nombradas NLTK

En este caso, se prone la transformación por medio de la cual sólo se incluyan las entidades nombradas obtenidas utilizando la librería `NLTK`. Se espera una mejora respecto a la solución inial ya que las entidades nombradas, que pueden ser lugares, organizaciones, etc...tienen mucho más peso a la hora de discernir entre diferentes textos que por ejemplo los artículos o conjunciones. Estos últimos tipos de palabras, pueden ser compartidas sin ningún problema entre textos de personas totalmente diferentes. La implementación de esta transformación se ha realizado a través de las funciones `get_named_entities_2` y `get_named_ent_txts_2`, y los resultados finales se han sido guardados utilizando la clave `identity_analysis_2`.

4.1.3 Entidades nombradas Stanford NER

Esta transformación es la misma que la realizada en la sección 4.1.3, pero utilizando el Stanford Named Entity Recognizer en vez de la librería `NLTK`. La función utilizada para ello se llama `get_named_standfor`.

4.1.4 Exclusión de palabras vacías

La siguiente transformación consiste en la eliminación de las palabras vacías de un texo. Definimos palabras vacías a palabras propias del léxico que en principio no tienen un significado, como determinantes, artículos, etc. Se espera una mejora respecto al resultado inicial ya que estas palabras vacías, que por razones propias del lenguaje pueden estar presentes en textos que no tengan ninguna relación, pueden crear conexiones entre textos que en realidad no están relacionados. Para la implementación de esta transformación, se han utilizado las dos funciones `get_language` y `delete_words_from_text`desarrolladas con anterioridad en combinación con la función `get_texts_no_stop_words`. Los resultados de esta transformación se guardarán bajo la clave `no_stop_words`.

4.1.4 Exclusión del nombre Thomas Baker

Es lógico que las palabras "Thomas" y "Baker" puedan estar presentes en todos los textos, ya que aunque se refierent a 4 personas diferentes, todas ellas comparten el nombre Thomas Baker. Está claro que ambas palabras van a conectar textos que pueden estar o no

relacionados. En este último caso, puede llevar a confusiones. Por ello, se espera que la eliminación de ambas palabras mejore de manera notoria la diferenciación entre los diferentes textos. Para ello, se ha hecho uso de la función `delete_words_from_text` junto con la función `get_texts_exclude_tomas_baker` Los resultados han sido recogidos bajo la clave `no_tomas_baker`.

4.1.5 Utilización de N-gramas

En esta sección, se propone representar los textos por medio de N-gramas. Definimos N-grama como una tupla de N palabras pertenecientes al texto, escogidas teniendo en cuenta la posición entre ciertas palabras del texto y que puede que representen un concepto único. Puesto que el significado de una palabra viene mejor determinado por las palabras que están a su alrededor, la utilizadión de esta técnica nos va ayudar a detectar textos que puedan explicar conceptos parecidos y por lo tanto a la clusterización de los textos. Por ello, se espera una mejora respecto a la solución inicial. En la presente práctica se han tenido en cuenta tanto bigramas como trigramas. La función utilizada para esta transformación es `get_ngram` y los resultados han sido guardados bajo la clave `bigrams`, para los bigramas, y `trigrams`, para los tigramas.

4.1.6 Stemming

La siguiente transformación propuesta consiste en obtener la raiz de cada palabra, proceso denominado *stemming*. Este proceso puede ayudar a conectar textos que están relacionados. Por ejemlo, imaginemos que dos textos versan sobre un cantante. Es probable que aparezcan palabras tales como *música* o *musical*. Sin realizar ninguna transformación, para el algoritmo utilizado en la presente práctica, ambas palabras son diferentes y no guardan relación. Sin embargo, está claro que ambas palabras parten de la misma raiz y además están muy relacionadas. Analizando solo las raizes, llegaríamos a la conclusión de que ambas palabras son la "misma". Para la implementación de esta transformación, se han utilizado las funciones `stemming_2` y `get_stemmed_txts_2` incluyendo los resultados bajo la clave `stemmed_txts`.

4.1.7 Eliminación de palabras repetidas

A través de esta transformación, lo que se pretente es dejar los textos con palabras únicas, de tal forma que no influya cuántas veces se repitan o no un determinado grupo de ellas. Podría decirse que comparamos el *“esqueleto”* o *”esquema”* del texto. Para ello, utilizamos la función `delete_repWords_stopWords` y guardamos los resultados bajo la clave `no_repeated_words`.

4.1.8 Lematización

La siguiente técnica utilizada, será la lematización de los textos. Lematizar consiste en encontrar el lema de cada palabra, es decir, la unidad mínima de significado de las mismas. Obteniendo estas unidades, se puede realizar una clusterización que agrupe textos que expliquen temas parecidos, obteniendo esa explicación a partir de lemas iguales. Por ello, se espera una mejora importante en comparación con la solución inicial. Para realizar esta transformación, se hace uso de la clase `SpanishLemmatizer` (sólo para textos en castellano) y la función `lemmatize_texts`. Los resultados obtenidos con la presente transformación han sido guardados bajo la clave `lemmatized`.

4.2 Combinaciones

Además de las transformaciones individuales mencionadas anteriormente, también se pueden combinar para obtener mejores resultados. En concreto, se han probado las siguientes combinaciones:

```
* Exclusión del nombre Thomas Baker + Stemming, guardando lo
s resultados bajo la clave no_tomas_stemmed
```

```
* Exclusión del nombre Thomas Baker + Stemming + Exclusión d
e palabras vacías, guardando los resultados bajo la clave no
_tomas_stemmed_no_stop
```

```
* Exclusión del nombre Thomas Baker + Stemming + Filtrado po
r tipo de palabras, guardando los resultados bajo la clave n
o_tomas_stemmed_ent1
```

```
*  Entidades nombradas NLTK + Exclusión del nombre Thomas Ba
ker, guardando los resultados bajo la clave named_ent_2_no_t
omas_barker
```

```
* Entidades nombradas Stanford NER + Exclusión del nombre Th
omas Baker, guardando los resultados bajo la clave stanford_
ner_no_thomas_baker
```

5. Ejecución del programa completo

A continuación se procede a la ejecución del programa principal, evaluando las diferentes transformaciones propuestas.

5.1. Lectura de los textos iniciales

El primer paso consiste en la lectura de los datos iniciales. En este proceso, leemos todos los documentos del path indicado y guardamos los textos tokenizados.

5.2. Aplicación de transformaciones

El siguiente paso consiste en la aplicación de todas las transformaciones descritas en las secciones 4.1 y 4.2, a través de la llamada de todas las funciones y guardando los correspondientes textos resultantes con las transformaciones aplicadas.

5.3. Agrupación de los textos (clustering) utilizando un número fijo de clústers

Una vez que se han aplicado las transformaciones pertinentes en la sección anterior, el siguiente paso consiste en realizar la agrupación de los textos (clustering) teniendo en

cuenta las diferentes transformaciones. Además, este proceso se ha realizado teniendo en cuenta tres algoritmos de clustering diferentes: `AgglomerativeClustering`, `KMeans` y `MiniBatchKMeans`, todos ellos englobados en la librería `scikit-learn`.. Para facilitar el posterior análisis de los datos, los resultados han sido guardados en un archivo denominado `4clusters.csv` el cual está localizado en la carpeta `CSV_output`.

5.4. Agrupación de los textos (clustering) utilizando un número variable de clústers

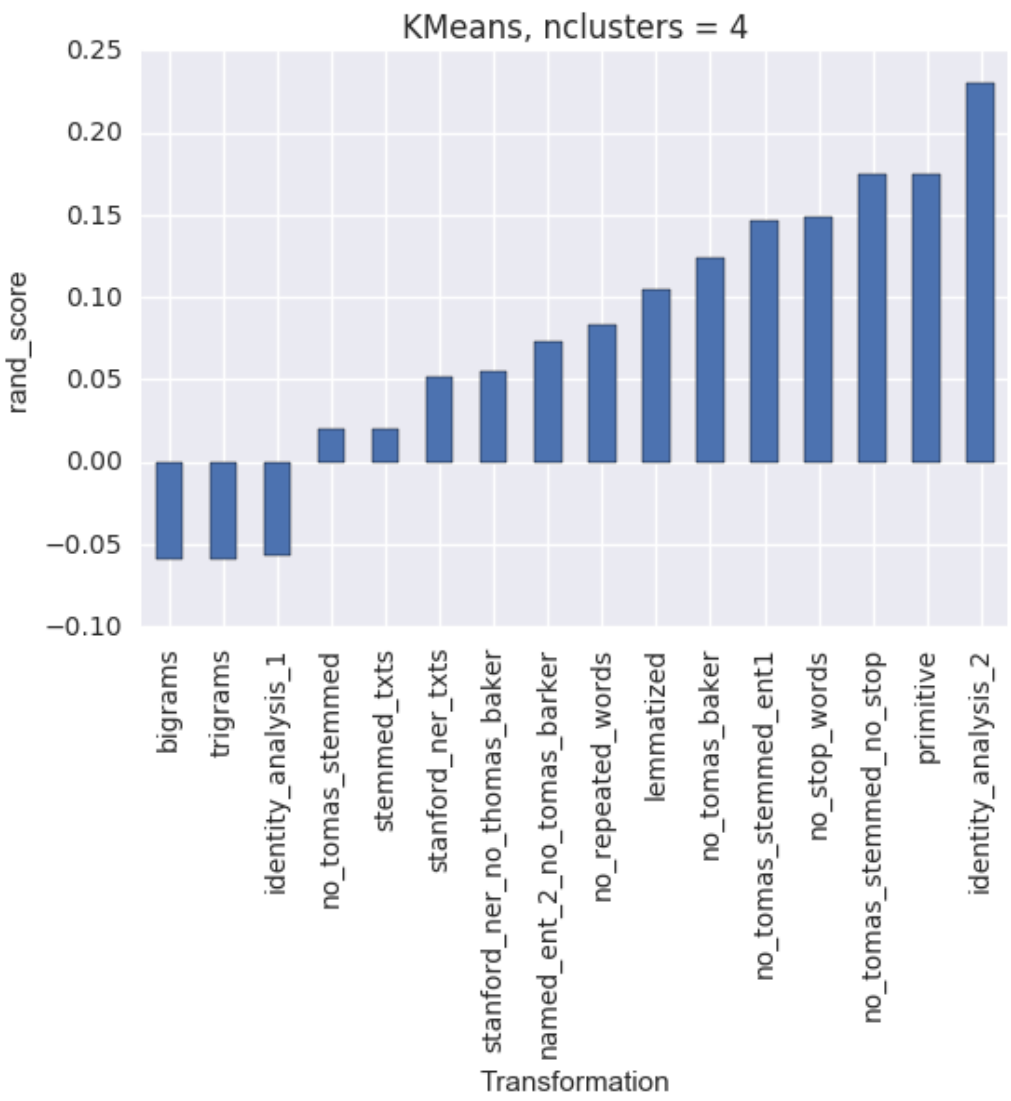
Como se ha comentado en la sección anterior, el número real de clústers en los que se agrupan los texos es conocido e igual a 4. Sin embargo, puede ser interesante analizar el efecto de variar el número de clústers sobre el proceso estudiado. Es posible, que en algunos casos y bajo circunstancias especiales, se puedan obtener mejores resultados que en el caso donde se fijan el número de clústers igual a 4. Para estudiar esta posibilidad, en el presente apartado se ha repetido el proceso descrito en la sección anterior pero variando el número de clústers de 1 a 10. Los resultados completos han sido guardados en el archivo `rangeclusters.csv`, mientras que los resultados referentes al número de clústers que optimiza los resultados se han almacenado en el archivo `bestclusters.csv`. Ambos archivos están localizados en la carpeta `CSV_output`.

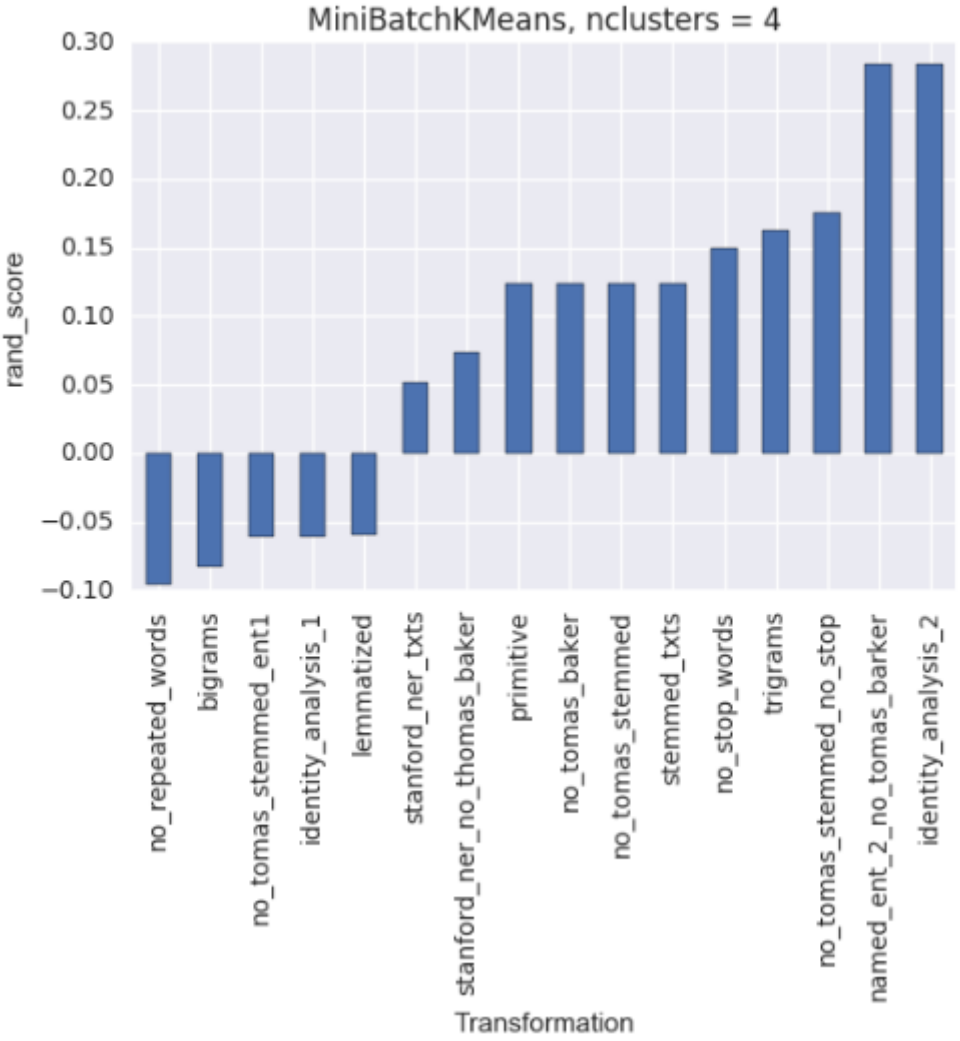
6. Evaluación de los resultados

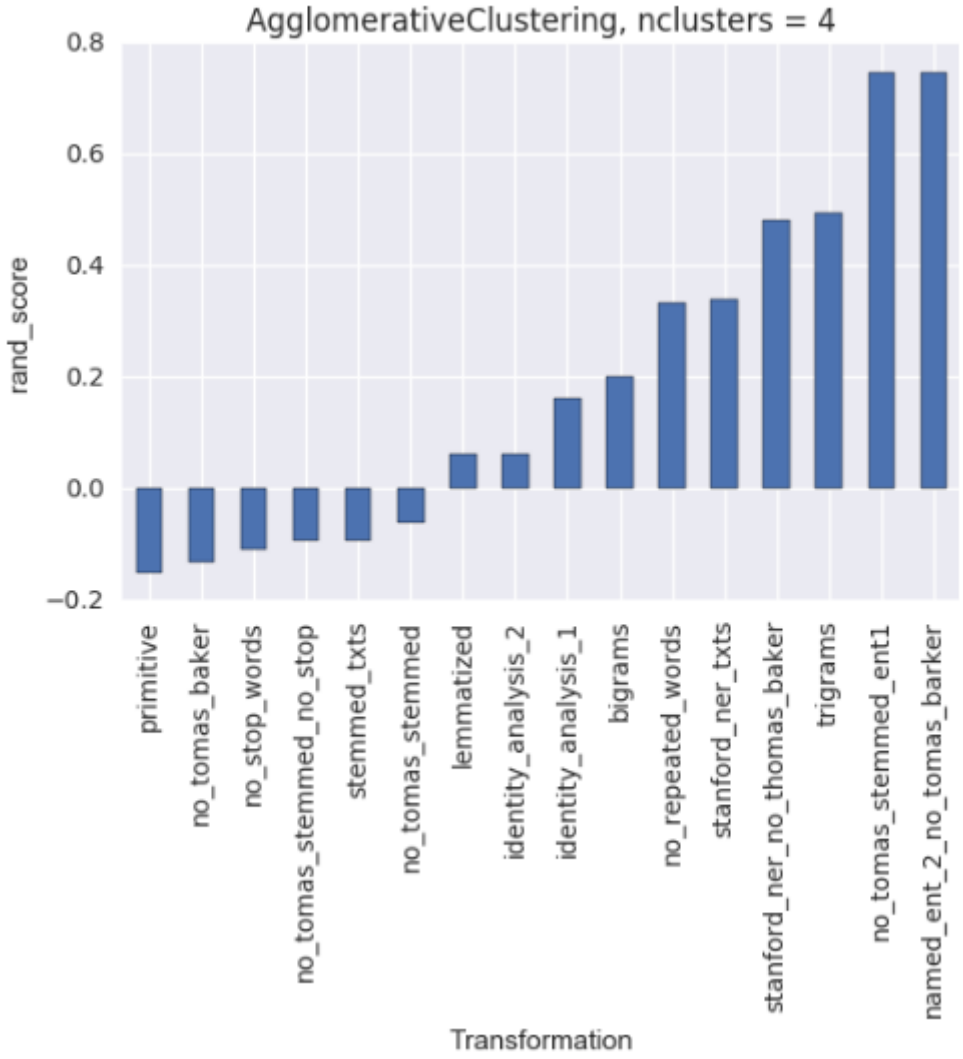
A continuación, se van a evaluar los resultados obtenidos en la sección anterior, haciendo uso extensivo de técnicas de visualización. Dicho análisis se va a basar en el concepto de `rand_score`. Este parámetro evalua la similaridad entre dos conjuntos de datos agrupados de manera diferente. De esta manera, se comparará en cada caso los clústers obtenidos en la presente práctica con la configuración real. Si la configuración obtenida fuera igual a la real, el parámetro `rand_score` adquiere el valor de 1. Por otro lado, la peor configuración posible en comparación con el caso real está caracterizada por un parámetro `rand_score` igual a -1. En las dos siguientes secciones, se analizarán todas las transformaciones realizadas teniendo en cuenta todos los algorítmos de clustering diferentes teniendo en cuenta un número de clústers fijo y variable.

6.1. Evaluación de los resultados: número de clústers fijo (4)

En esta sección, se analizarán los resultados obtenidos fijando el número de clúster igual a 4. A continuación, se van a graficar el `rand_score` correspondiente a cada una de las transformaciones descritas en las Secciones 4.1 y 4.2, diferenciando los resultados por el tipo de algorítmo de clustering utilizado.


KMeans, nclusters = 4
rand_score
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
bigrams
trigrams
identity_analysis_1
no_tomas_stemmed
stemmed_txts
stanford_ner_txts
stanford_ner_no_thomas_baker
named_ent_2_no_tomas_barker
no_repeated_words
lemmatized
no_tomas_baker
no_tomas_stemmed_ent1
no_stop_words
no_tomas_stemmed_no_stop
primitive
identity_analysis_2
Transformation


MiniBatchKMeans, nclusters = 4
rand_score
0.30
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
no_repeated_words
bigrams
no_tomas_stemmed_ent1
identity_analysis_1
lemmatized
stanford_ner_txts
stanford_ner_no_thomas_baker
primitive
no_tomas_baker
no_tomas_stemmed
stemmed_txts
no_stop_words
trigrams
no_tomas_stemmed_no_stop
named_ent_2_no_tomas_barker
identity_analysis_2
Transformation

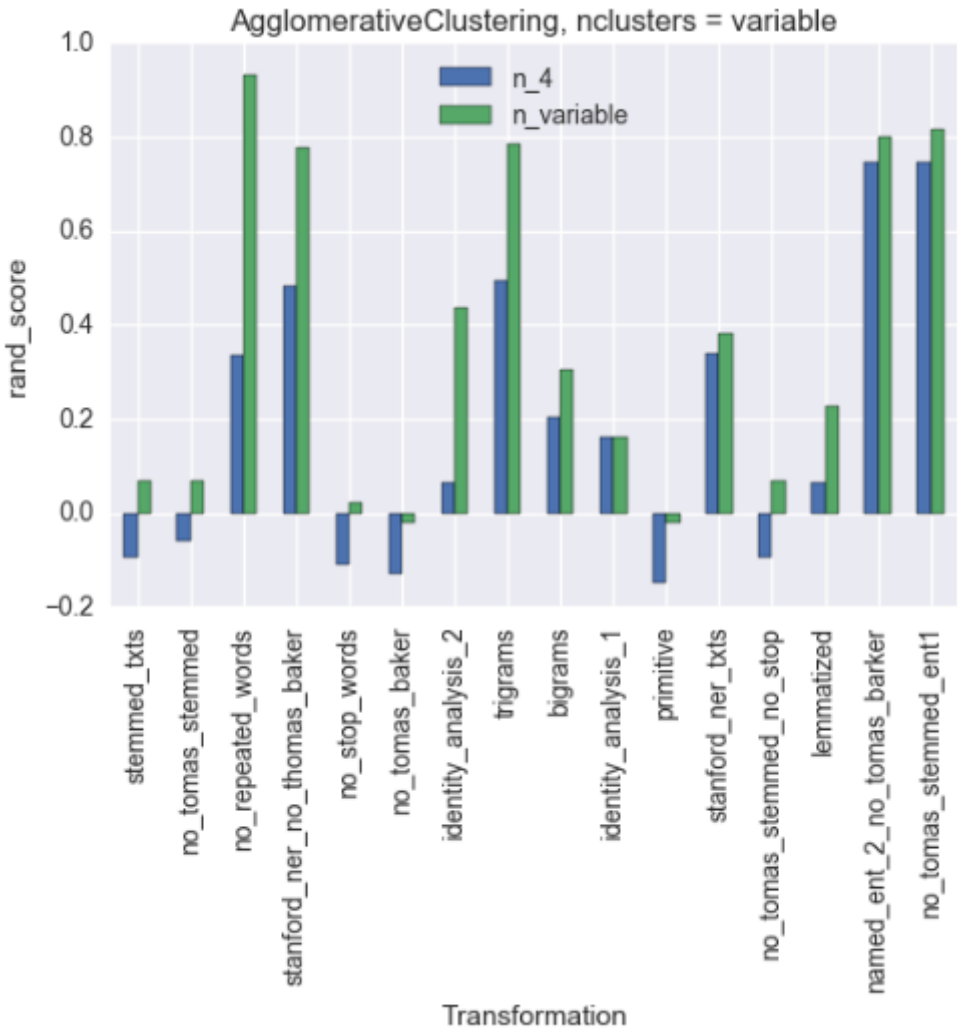


En primer lugar, cabe destacar la gran diferencia obtenida en función al algorítmo de clustering utilizado. El que da mejores resultados claramente es el de `AgglomerativeClustering`, y los resultados asociados al mismo serán, por tanto, los estudiados con mayor profundidad. Aquella representación con una peor puntuación corresponde a la `primitive`, que corresponde a los textos iniciales en los que no se ha aplicado ninguna transformación. Por otro lado, la transformación que obtiene la mejor puntuación, ~0.75 y por tanto cercana a 1, es la combinación de *Entidades nombradas NLTK + Exclusión del nombre Thomas Baker* (`named_ent_2_no_tomas_barker`). De entre todas las transformaciones unitarias, la que mejor resultados da es la de Trigramas (`trigrams`). Por otro lado, cabe destacar la alta no linealidad de la combinación de transformaciones. Por ejemplo, de los 4 mejores implican la combinación de la transformación *Exclusión del nombre Thomas Baker* (`no_tomas_baker`) con otra transformación diferente. Sin embargo, la transformación `no_tomas_baker` por si sola no parece aumentar en gran medida el resultado inicial. También cabe destacar la importancia de las transformaciones que implican el reconocimiento de entidades nombradas, ya que están presentes en aquellas transformaciones con mejores resultados.

6.2. Evaluación de los resultados: número de clústers variable (1-10)

En el presente apartado se analizan los resultados obtenidos en la Sección 5.4. En ella, se ha considerado el efecto de cada una de las transformaciones incluidas en las Secciones 4.1 y 4.2 pero teniendo en cuenta un número variable de clústers, de 1 a 10. Los presentes resultados se centran en el algorítmo de clustering `AgglomerativeClustering`, ya que

como se ha visto en el apartado anterior, es el que mejor resultado arroja. En la siguiente figura, se grafica el `rand_score` de cada una de las transformaciones consideradas, teniendo en cuenta dos criterios: un número de clústers igual a 4 (`n_4`, sombreado en azul), y el mejor resultado obtenido en cada caso para los 10 números de clústers considerados (`n_variable`, sombreado en verde). Como se puede observar, para ciertas transformaciones (por ejemlo `named_ent_2_no_tomas_barker` y `no_tomas_stemmed_ent1`), en ambos casos se obtiene un resultado parecido. Sin embargo, en otras ocasiones, por ejemplo para la transformación `no_repeated_words`, se logra una mejora notable en el resultado si el número de clústers no se fija de antemano. De hecho, la aplicación de dicha transformación junto con el hecho de no fijar en número de clusters arroja el mejor resultado de todo el estudio. Dado que en el problema estudiado en la presente práctica se conoce el número real de clústers, puede que lo más apropiado sea hacer uso del mismo. Sin embargo, en muchas ocasiones, puede ser que averiguar el número de clústers sea precisamente una de las partes más importantes del problema. En dichos casos, el análisis realizado en la presente sección puede ser muy relevante.



7. Conclusiones

En la presente práctica, se han analizado 19 textos pertenecientes a 4 personas diferentes llamadas todas ellas *Thomas Barker*. El objetivo principal de la práctica era agrupar los textos utilizando diferentes algorítmos de clustering, estudiando el effecto de aplicar diversas transformaciones sobre los mismos, comparando los resultados con el resultado real (conocido de antemano). Respecto a los algorítmos de clustering, se ha comprobado que el que mejor resultados arroja es el de `AgglomerativeClustering`, de la librería de `Python` `scikit-learn`. Respecto a las transformaciones, se ha comprobado que aquellas que mejor funcionan suelen ser resultado de combinar el reconocimiento de entidades nombradas así como la exclusión del nombre *Thomas Barker*. Por otro lado, mientras que dos transformaciones individuales pueden no mejorar de manera notable el resultado, ambas combinadas pueden llegar a dar muy buenos resultados. Se considera por tanto que la combinación de transformaciones es altamente no lineal y por tanto es dificil en ocasiones predecir cual va a ser el efecto final. Finalmente, también se ha estudiado el efecto de variar el número de clústers, pudiéndose obtener grandes diferencias en comparación con el caso cuando el número de clústers se deja fijo.

Respecto a la aportación de cada uno de los alumnos responsables de la presente práctica (*Ignacio Arias Barra y Raúl Sánchez Martín*), se considera que ambos han colaborado de manera igual durante todos los procesos de la misma: diseño inicial, planteamiento e implementación de las transformaciones tenidas en cuenta, ejecución del código principal, análisis de los resultados y redacción de la memoria.