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 (https://nlp.stanford.edu/software/CRF-NER.shtml). 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 (https://blog.manash.me/configuring-stanford-parser-and-stanford-ner-tagger-with-nltk-in-python-on-windows-f685483c374a). A continuación, se incluye el fragmento de código responsable de la importación de todas las librerías necesarias para el resto de la práctica.

In [1]:

```
# -*- coding: utf-8 -*-
# Importing libraries
import re, pprint, os, numpy
import nltk
from nltk import ngrams
#import goslate
from nltk.collocations import *
import string
from nltk.corpus import stopwords
path_to_append = '/media/nacho/f8371289-0f00-4406-89db-d575f3cdb35e/Master/Trime
stre 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 textos. Esta funci
ón 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 lu
gar los vectoriza y posteriormente los agrupa utilizando diferentes algor
itmos de clustering. Esta función ha sido modificada respecto a la propor
cionada inicialmente para poder elegir que algorítmo de clustering se uti
liza.
```

In [2]:

```
def read_file(file):
    """
    Function to read a file, whose path is specified by
    the input "file".
    """
    myfile = open(file,"r")
    data = ""
    lines = myfile.readlines()
    for line in lines:
        data = data + line
    myfile.close
    return data

def TF(document, unique_terms, collection):
    """
    Function to create a TF vector for one document which belongs to
    to a collection. For each of our unique words, we have a feature
    which is the tf for that word in the current document. The
    following imputs must be specified:
        *) document: the document to study
        *) collection: collection to which that document belongs
        *) unique_terms: unique terms of the collection
    """
    word_tf = []
    for word in unique_terms:
        word_tf.append(collection.tf(word, document))
    return word_tf

def cluster_texts(texts, clustersNumber, distanceFunction, clusterMode):
    """
    Function to cluster several texts. The following inputs must be
    specified:
        *) texts: collection of texts to cluster
        *) clustersNumber: number of clusters to be used
        *) distanceFunction: distance function to be used by the
           clustering algorithms
```

```
        *) clusterMode: cluster mode to be used:"AgglomerativeClustering",
           "KMeans" or "MiniBatchKMeans", all of them belonging to the
           scikit-learn library

    """

    collection = nltk.TextCollection(texts)
    # print("Created a collection of", len(collection), "terms.")

    # Get a list of unique terms
    unique_terms = list(set(collection))
    # print("Unique terms found: ", len(unique_terms))

    ### And here we actually call the function and create our array of vectors.
    # TF mide la frecuencia en los textos.
    # Mira de los terminos unicos, cuantas veces aparece en el documento. No mir
a cuantas veces aparece en la coleccion
    # Hay otras medidas, como TF-IDF que son mas precisas porque tambien miran c
uantas veces aparece en la coleccion
    vectors = [numpy.array(TF(f,unique_terms, collection)) for f in texts]
    # print("Vectors created.")
    # print(vectors)

    # for vector in vectors:
        # print("Vector ", len(vector))

    # initialize the clusterer
    # clusterer = GAAClusterer(clustersNumber)
    # clusters = clusterer.cluster(vectors, True)
    # Estas lineas siguientes comentadas es lo mismo pero con otra libreria, la
 llamada scikit-learn

    if clusterMode == "AgglomerativeClustering":

        clusterer = AgglomerativeClustering(n_clusters=clustersNumber,
                                            linkage="average", affinity=distanceFun
ction)
        clusters = clusterer.fit_predict(vectors)

    elif clusterMode == "KMeans":

        clusterer = KMeans(n_clusters=clustersNumber, random_state=0)
        clusters = clusterer.fit(vectors).predict(vectors)

    elif clusterMode == "MiniBatchKMeans":

        clusterer = MiniBatchKMeans(n_clusters=clustersNumber, random_state=0)
        clusters = clusterer.fit(vectors).predict(vectors)
    else:
        print("Invalid cluster mode")
        return None

    return clusters
```

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`, implementada de acuerdo al siguiente fragmento de código:

In [3]:

```
def get_language(possible_lan, text):
    """
    Function that returns the language of a text. The following
    inputs must be specified:
        *) text: text to be analyzed
        *) possible_lan: list of possible languages
            +) "EN":english
            +) "ES":spanish
    More info in: http://blog.alejandronolla.com/2013/05/15/detecting-text-langu
age-with-python-and-nltk/
    """
    languages_score = {}
    for language in possible_lan:
        stopwords_set = set(stopwords.words(language))
        words_set = set(text)
        common_elements = words_set.intersection(stopwords_set)
        languages_score[possible_lan[language]] = len(common_elements)
    return max(languages_score.items(), key=operator.itemgetter(1))[0]
```

Otra función que se ha utilizado en diferentes ocasiones a lo largo de la presente práctica es `delete_words_from_text`. Dicha función, desarrollada en el fragmento de código de abajo, elimina de un texto inicial todas las palabras espeficiadas.

In [4]:

```
def delete_words_from_text(text, words_to_delete):
    """
    Function that filters a initial text excluding all
    the words specified in the list "words_to_delete"
    """
    words_to_include = []
    words_to_delete = [word.lower() for word in words_to_delete]

    for word in text:
        if word.lower() not in words_to_delete:
            words_to_include.append(word)

    return words_to_include
```

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 implementado 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`.

In [5]:

```
def get_named_entities_1(initial_document, selected_types):
    """
    Function that filters a initial document only including
    that words specified in the variable "selected_types".
    """
    named_entities = list()
    selected_entities = list()
    try:
        for sentence in initial_document:
            tokenized_sentence = nltk.word_tokenize(sentence)
            # tagged_sentence = nltk.pos_tag(tokenized_sentence)
            tagged_sentence = nltk.pos_tag(tokenized_sentence, tagset= 'universa
l')
            named_ent = nltk.ne_chunk(tagged_sentence, binary=False)
            named_entities.append(named_ent)

        for element in named_entities:
            word = element.pos()[0][0][0]
            type_word = element.pos()[0][0][1]
            if type_word in selected_types:
                selected_entities.append(word)

        return selected_entities

    except Exception as e:
        print(str(e))

def get_named_ent_txts_1(raw_texts):
    """
    Function that applies the function "get_named_entities_1"
    to a collection of texts "raw_texts"
    """
    named_ent_txts_1 = []
    for text in raw_texts:
        curr_named_ent = get_named_entities_1(text, types_included_1)
        text_to_append = nltk.Text(curr_named_ent)
        named_ent_txts_1.append(text_to_append)
    return named_ent_txts_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`.

In [6]:

```
def get_named_entities_2(initial_document):
    """
    Function that filters a document ("initial_document") only including
    the recognized named entities using the capabilities of
    the NLTK library.
    """
    named_entities = list()
    selected_entities = list()
    try:
        for sentence in initial_document:
            tokenized_sentence = nltk.word_tokenize(sentence)
            tagged_sentence = nltk.pos_tag(tokenized_sentence)
#             tagged_sentence = nltk.pos_tag(tokenized_sentence, tagset= 'univer
sal')
            named_ent = nltk.ne_chunk(tagged_sentence, binary=False)
            named_entities.append(named_ent)

        for element in named_entities:
            if hasattr(element[0], 'label') and element[0].label:
                selected_entities.append(element[0].leaves()[0][0])

        return selected_entities

    except Exception as e:
        print(str(e))

def get_named_ent_txts_2(raw_texts):
    """
    Function that applies the function "get_named_entities_2"
    to a collection of texts "raw_texts"
    """
    named_ent_txts_2 = []
    for text in raw_texts:
        curr_named_ent = get_named_entities_2(text)
        text_to_append = nltk.Text(curr_named_ent)
        named_ent_txts_2.append(text_to_append)
    return named_ent_txts_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 (https://nlp.stanford.edu/software/CRF-NER.shtml)](https://nlp.stanford.edu/software/CRF-NER.shtml) en vez de la librería `NLTK`.

In [7]:

```
def get_entities_standorf(sample, types_named_entities):
    """
    Function that filters a document ("initial_document") only including
    the recognized named entities using the capabilities of
    the Stanford NER.
    """
    # Select the first classifier model
    stanford_classifier = os.environ.get('STANFORD_MODELS').split(':')[2]

    # Get the path for the StandorfNERTagger
    stanford_ner_path = os.environ.get('CLASSPATH').split(':')[0]

    st = StanfordNERTagger(stanford_classifier, stanford_ner_path, encoding='utf
-8')
    result_named_entities = st.tag(sample.split())
    filtered_named_entities = []

    for item in result_named_entities:
        word, entity = item
        if entity in types_named_entities:
#              filtered_named_entities.append((word, entity))
            filtered_named_entities.append(word)

    return filtered_named_entities


def get_named_ent_txts_3(raw_texts, types_named_entities):
    """
    Function that applies the function "get_entities_standorf"
    to a collection of texts "raw_texts"
    """
    named_ent_txts_3 = []
    for text in raw_texts:
        curr_named_ent = get_entities_standorf(text, types_named_entities)
        text_to_append = nltk.Text(curr_named_ent)
        named_ent_txts_3.append(text_to_append)
    return named_ent_txts_3
```

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`, incluida en el fragmento de código de abajo. Los resultados de esta transformación se guardarán bajo la clave `no_stop_words`.

In [8]:

```
def get_texts_no_stop_words(raw_texts):
    """
    Function that deletes the stop words of a collection
    of texts included in the variable "raw_texts"
    """
    filtered_texts = []
    for text in raw_texts:
        language = {'en':'english', 'es':'spanish'}[get_language(possible_lan, t
ext)]
        words_to_exclude = list(set(stopwords.words(language)))
        curr_filtered_text = delete_words_from_text(text, words_to_exclude)
        filtered_texts.append(nltk.Text(curr_filtered_text))
    return filtered_texts
```

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`, implementada en el código de abajo. Los resultados han sido recogidos bajo la clave `no_tomas_baker`.

In [9]:

```
def get_texts_exclude_tomas_baker(raw_texts):
    """
    Function that deletes the words "Thomas" and "Baker"
    of a collection of texts included in the variable "raw_texts"
    """
    filtered_texts = []
    for text in raw_texts:
        curr_filtered_text = delete_words_from_text(text, ['Thomas', 'Baker'])
        filtered_texts.append(nltk.Text(curr_filtered_text))
    return filtered_texts
```

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.

In [10]:

```
def get_ngram(raw_texts, ngramlimit):
    ngramlist=[]
    for text in raw_texts:
        ngram_text = nltk.ngrams(text, ngramlimit)
        ngramlist.append(nltk.Text(ngram_text))
    return ngramlist
```

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` descritas en el código de abajo, incluyendo los resultados bajo la clave `stemmed_txts`.

In [11]:

```
def stemming_2(text, lan='en'):
    """
    Function that "stems" a text.
    """
    stemmeds = []
    if lan == 'en':
        # Steamer ingles
        stemmer = PorterStemmer()
    elif lan == 'es':
        # Stemer espanol
        stemmer = SnowballStemmer("spanish")

    # Para cada token del texto obtenemos su raíz.
    for word in text:
        stemmed = stemmer.stem(word)
        stemmeds.append(stemmed)
    # Escribimos el resultado para compararlo con las palabras originales.
    return stemmeds

def get_stemmed_txts_2(raw_texts):
    """
    Function that applies the function "stemming_2"
    to a collection of texts "raw_texts"
    """
    stemmed_txts = []
    for text in raw_texts:
        language = get_language(possible_lan, text)
        stemmed_txt = stemming_2(text, language)
        text = nltk.Text(stemmed_txt)
        stemmed_txts.append(text)
    return 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` descrita en el código de abajo y guardamos los resultados bajo la clave `no_repeated_words`.

In [12]:

```
def delete_repWords_stopWords(raw_texts):
    """
    Function that deletes the repeated words of
    all the texts included in the collection "raw_texts"
    """
    deleted_repeated = []
    for text in raw_texts:
        delete_repeated_texts = []
        text = [w.lower() for w in text]
        unique_terms = list(set(text))
#         print("Número de palabras del texto: ",str(len(text)))
#         print("Tamaño del vocabulario filtrado: ", str(len(unique_terms)))
        deleted_repeated.append(nltk.Text(unique_terms))
    return deleted_repeated
```

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`, ambas especificadas en el fragmento de código de abajo. Los resultados obtenidos con la presente transformación han sido guardados bajo la clave `lemmatized`.

In [13]:

```
class SpanishLemmatizer():
    # Abrimos el fichero donde tenemos la información para cada palabra y lo car
gamos en un diccionario.
    def __init__(self):
        with open('./Lemmatizer/lemmatization-es.txt', 'r', encoding="utf8") as
f:
            self.lemma_dict = {}
            for line in f:
                if line.strip(): # Evitamos posibles líneas en blanco.
                    value, key = line.split(None, 1) # Nos quedamos con los valo
res clave y valor.
                                                     # None implica espacio en b
lanco.
                    key = key.rstrip() # Limpiamos la línea para evitar los cara
cteres de salto \n ó \r.
                    self.lemma_dict[key] = value
                    self.lemma_dict[value] = value  # Añadimos por si acaso tamb
ién el valor como clave.
```

```
    # Obtenemos el lemma para la palabra solicitada si es que se dispone de él.
 En caso contrario devuelve la palabra.
    # Útil en los casos en los que no se haya aplicado un tratamiento previo del
 texto (stopwords y puntuación).
    def lemmatize(self,word):
        try:
            lemma = self.lemma_dict[word.lower()]
        except KeyError:
            lemma = word
        return lemma

def lemmatize_texts(raw_texts, possible_lan):
    nlemmas_texts = []
    for text in raw_texts:
        language = get_language(possible_lan, text)
        if language== 'en':
            # Seleccionamos el lematizador.
            wordnet_lemmatizer = WordNetLemmatizer()
            # Obtenemos los tokens de las sentencias.
#             tokens = nltk.word_tokenize(text)

            lemmatizeds = []
            nlemmas = []

            for token in text:
                lemmatized = wordnet_lemmatizer.lemmatize(token)
                lemmatizeds.append(lemmatized)
                #print('token ',token)
                #print('lema ',lemmatized)
                # Obtenemos los lemmas consultando la base de datos de WordNet.
                list = wordnet.synsets(token)
                # Si encontramos alguna palabra relacionada obtenemos sus lemas
 y nos quedamos con el primero.
                if len(list) >= 1:
                    lemma = list[0].lemma_names('eng')
                    if len(lemma) > 1:
                        nlemmas.append(lemma[0])
                    else:
                        nlemmas.append((token))
                # En caso contrario simplemente introducimos en la solución la p
alabra actual.
            else:
                nlemmas.append(token)
            nlemmas_texts.append(nltk.Text(nlemmas))
        elif language== 'es':
            lemmatizer = SpanishLemmatizer()

            # Obtenemos los tokens del texto.
#             tokens = nltk.word_tokenize(text)
            nlemmas = []
            lemmatizeds = []
            for token in text:
                # Obtenemos los lemmas consultando el archivo de lemmas.
                lemmatized = lemmatizer.lemmatize(token)
                #print('token ',token)
                #print('lema ',lemmatized)
                lemmatizeds.append(lemmatized)
                # Obtenemos los lemmas consultando la base de datos de WordNet.
                list = wordnet.synsets(token, lang='spa')
                # Si encontramos alguna palabra relacionada obtenemos sus lemas
 y nos quedamos con el primero.
```

```
                if len(list) >= 1:
                    lemma = list[0].lemma_names('spa')
                    if len(lemma) >= 1:
                        nlemmas.append(lemma[0])
                    else:
                        nlemmas.append(token)
                # En caso contrario simplemente introducimos en la solución la p
alabra actual.
                else:
                    nlemmas.append(token)
            nlemmas_texts.append(nltk.Text(nlemmas))
        else:
            print('Lenguaje no reconocido')
    return nlemmas_texts
```

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 los resultados
 bajo la clave no_tomas_stemmed

* Exclusión del nombre Thomas Baker + Stemming + Exclusión de palabras va
cías, guardando los resultados bajo la clave no_tomas_stemmed_no_stop

* Exclusión del nombre Thomas Baker + Stemming + Filtrado por tipo de pal
abras, guardando los resultados bajo la clave no_tomas_stemmed_ent1

*  Entidades nombradas NLTK + Exclusión del nombre Thomas Baker, guardand
o los resultados bajo la clave named_ent_2_no_tomas_barker

* Entidades nombradas Stanford NER + Exclusión del nombre Thomas Baker, g
uardando 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, proceso que se lleva a cabo por medio del siguiente fragmento de código:

In [14]:

```python
# Folder with all texts
folder = "Thomas_Baker"
# gsObj=goslate.Goslate()
types_named_entities = ["LOCATION", "PERSON", "ORGANIZATION"]
# Empty list to hold text documents.
raw_texts = []
raw_texts_2 = []
raw_texts_en = []
named_ent_txts_1 = []
types_included_1 = ['NOUN']

possible_lan = {"english":"en", "spanish":"es"}
clustering_modes = ["AgglomerativeClustering", "KMeans", "MiniBatchKMeans"]


listing = os.listdir(folder)
for file in sorted(listing):
    if file.endswith(".txt"):
        url = folder+"/"+file
        print(file)
        f = open(url,encoding="latin-1");
        raw = f.read()
        f.close()
        f2 = open(url, 'r', encoding="utf8")
        raw2 = f2.read()
        f2.close()
        raw_texts_2.append(raw2)
        tokens = nltk.word_tokenize(raw)
        text = nltk.Text(tokens)
        raw_texts.append(text)


print("Prepared ", len(raw_texts), " documents...")
print("They can be accessed using texts[0] - texts[" + str(len(raw_texts)-1) +
"]")
```

```
001.txt
002.txt
005.txt
008.txt
009.txt
011.txt
015.txt
017.txt
020.txt
024.txt
027.txt
028.txt
036.txt
041.txt
047.txt
050.txt
056.txt
072.txt
075.txt
Prepared  19  documents...
They can be accessed using texts[0] - texts[18]
```

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 del siguiente fragmento de código:

In [15]:

```
print('Using Stanford NER...')
stanford_ner_txts = get_named_ent_txts_3(raw_texts_2, types_named_entities)

print('Using Stanford NER removing Thomas Baker...')
stanford_ner_no_thomas_baker = get_texts_exclude_tomas_baker(stanford_ner_txts)

print("Removing non-stop words....")
texts_no_stop_words = get_texts_no_stop_words(raw_texts)

print("Removing Thomas Baker from the texts...")
texts_no_thomas_baker = get_texts_exclude_tomas_baker(raw_texts)

print("Getting text including only named entities according to criteria 1...")
named_ent_txts_1 = get_named_ent_txts_1(raw_texts)

print("Getting stemmed texts...")
stemmed_txts = get_stemmed_txts_2(raw_texts)

print("Getting no_tomas_stemmed...")
no_tomas_stemmed_txts = get_stemmed_txts_2(texts_no_thomas_baker)

print("Getting no_tomas_stemmed_no_stop...")
no_tomas_stemmed_no_stop_txts = get_texts_no_stop_words(no_tomas_stemmed_txts)

print("Getting no_tomas_stemmed_ent1...")
no_tomas_stemmed_ent1_txts = get_named_ent_txts_1(no_tomas_stemmed_txts)

print("Getting text including only named entities according to criteria 2...")
named_ent_txts_2 = get_named_ent_txts_2(raw_texts)

print("Getting text including only named entities according to criteria 2 and ex
cluding the words 'Tomas' and 'Baker'")
named_ent_2_no_tomas_barker_txts = get_texts_exclude_tomas_baker(named_ent_txts_
2)

print('Getting bigrams in texts')
bigrams_texts = get_ngram(raw_texts, 2)

print('Getting trigrams in texts')
trigrams_texts = get_ngram(raw_texts, 3)

print('Lemmatizing texts')
lemmatized_texts = lemmatize_texts(raw_texts, possible_lan)

print('Removing repeated words')
no_repeatedWords_noStopWords = delete_repWords_stopWords(raw_texts)

# Similarity distance
distanceFunction ="cosine"
# distanceFunction = "euclidean"

reference =[0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0, 1]
print("reference: ", reference)
```

```
Using Stanford NER...
Using Stanford NER removing Thomas Baker...
Removing non-stop words....
Removing Thomas Baker from the texts...
Getting text including only named entities according to criteria
 1...
Getting stemmed texts...
Getting no_tomas_stemmed...
Getting no_tomas_stemmed_no_stop...
Getting no_tomas_stemmed_ent1...
Getting text including only named entities according to criteria
 2...
Getting text including only named entities according to criteria 2
 and excluding the words 'Tomas' and 'Baker'
Getting bigrams in texts
Getting trigrams in texts
Lemmatizing texts
Removing repeated words
reference:  [0, 1, 2, 0, 0, 0, 3, 0, 0, 0, 2, 0, 3, 3, 0, 1, 2, 0,
 1]
```

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`. Este proceso es realizado por medio del siguiente código. 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`.

In [16]:

```
tested_models = {}
fix_grouping = 4
header_fields=['clustering_mode','model','rand_score','cluster_split']

csv_file = 'CSV_output/4clusters.csv'
with open(csv_file, 'w') as output_file:
    writer = csv.writer(output_file)
    writer.writerow(header_fields)
    new_row = []

    for cluster_mode in clustering_modes:
        tested_models[cluster_mode] = {}
        tested_models[cluster_mode]["primitive"] = cluster_texts(raw_texts,fix_g
rouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]["identity_analysis_1"] = cluster_texts(named
_ent_txts_1,fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]["stemmed_txts"] =
cluster_texts(stemmed_txts,fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]["no_tomas_baker"] = cluster_texts(texts_no_t
homas_baker,fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]["no_stop_words"] = cluster_texts(texts_no_st
op_words,fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]["no_tomas_stemmed"] = cluster_texts(no_tomas
_stemmed_txts,fix_grouping,distanceFunction, cluster_mode)
```

```
        tested_models[cluster_mode]["no_tomas_stemmed_no_stop"] =
cluster_texts(no_tomas_stemmed_no_stop_txts,fix_grouping,

distanceFunction, cluster_mode)
        tested_models[cluster_mode]["no_tomas_stemmed_ent1"] = cluster_texts(no_
tomas_stemmed_ent1_txts,fix_grouping,distanceFunction,
                                                                                clu
ster_mode)
        tested_models[cluster_mode]["identity_analysis_2"] = cluster_texts(named
_ent_txts_2,fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]["named_ent_2_no_tomas_barker"] = cluster_tex
ts(named_ent_2_no_tomas_barker_txts,
                                                                    fix_grouping,di
stanceFunction, cluster_mode)
        tested_models[cluster_mode]['bigrams'] = cluster_texts(bigrams_texts,fix
_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]['trigrams'] = cluster_texts(trigrams_texts,
fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]['lemmatized'] = cluster_texts(lemmatized_tex
ts, fix_grouping,distanceFunction, cluster_mode)
        tested_models[cluster_mode]['no_repeated_words'] = cluster_texts(no_repe
atedWords_noStopWords,fix_grouping, distanceFunction, cluster_mode)
        tested_models[cluster_mode]['stanford_ner_txts'] = cluster_texts(stanfor
d_ner_txts,fix_grouping,
                                                                            distanc
eFunction, cluster_mode)
        tested_models[cluster_mode]['stanford_ner_no_thomas_baker'] = cluster_te
xts(stanford_ner_no_thomas_baker,fix_grouping,
                                                                            distanc
eFunction, cluster_mode)

    # Evaluation
    tested_models_scores = {}

    for cluster_mode in tested_models:
        tested_models_scores[cluster_mode] = {}
        for model in tested_models[cluster_mode]:
            tested_models_scores[cluster_mode][model] = adjusted_rand_score(refe
rence,tested_models[cluster_mode][model])
    #     print("Model ", model, "; rand_score = ", adjusted_rand_score(referenc
e,tested_models[model]))

    for cluster_mode in tested_models_scores:
        print("*************************************")
        print("Getting results for the clustering mode ", cluster_mode)
        for model in sorted(tested_models_scores[cluster_mode].items(), key=oper
ator.itemgetter(1), reverse=True):
            print("Model ", model[0], "; rand_score = ", model[1])
            new_row = [cluster_mode, model[0], model[1], fix_grouping]
            writer.writerow(new_row)
        print("########################################")
        print("########################################")
```

```
************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_tomas_stemmed_ent1 ; rand_score =  0.7446236559139784
Model  named_ent_2_no_tomas_barker ; rand_score =  0.74462365591397
84
Model  trigrams ; rand_score =  0.49316851008458035
Model  stanford_ner_no_thomas_baker ; rand_score =  0.4809976247030
879
Model  stanford_ner_txts ; rand_score =  0.34076827757125155
Model  no_repeated_words ; rand_score =  0.33371040723981904
Model  bigrams ; rand_score =  0.20233998623537508
Model  identity_analysis_1 ; rand_score =  0.16176470588235295
Model  lemmatized ; rand_score =  0.06323687031082535
Model  identity_analysis_2 ; rand_score =  0.06323687031082535
Model  no_tomas_stemmed ; rand_score =  -0.060570071258907406
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0925000000000000
1
Model  stemmed_txts ; rand_score =  -0.09250000000000001
Model  no_stop_words ; rand_score =  -0.10854816824966075
Model  no_tomas_baker ; rand_score =  -0.1301115241635688
Model  primitive ; rand_score =  -0.1496421600520495
########################################
########################################
************************************
Getting results for the clustering mode  KMeans
Model  identity_analysis_2 ; rand_score =  0.2304469273743017
Model  no_tomas_stemmed_no_stop ; rand_score =  0.17503392130257805
Model  primitive ; rand_score =  0.17503392130257805
Model  no_stop_words ; rand_score =  0.1492537313432836
Model  no_tomas_stemmed_ent1 ; rand_score =  0.14703968770331813
Model  no_tomas_baker ; rand_score =  0.12347354138398917
Model  lemmatized ; rand_score =  0.10494931425163982
Model  no_repeated_words ; rand_score =  0.0836012861736335
Model  named_ent_2_no_tomas_barker ; rand_score =  0.07255936675461
744
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0551075268817
2045
Model  stanford_ner_txts ; rand_score =  0.051395007342143924
Model  no_tomas_stemmed ; rand_score =  0.020352781546811426
Model  stemmed_txts ; rand_score =  0.020352781546811426
Model  identity_analysis_1 ; rand_score =  -0.05698778833107188
Model  trigrams ; rand_score =  -0.058949624866023516
Model  bigrams ; rand_score =  -0.058949624866023516
########################################
########################################
************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  named_ent_2_no_tomas_barker ; rand_score =  0.28351955307262
57
Model  identity_analysis_2 ; rand_score =  0.2835195530726257
Model  no_tomas_stemmed_no_stop ; rand_score =  0.17503392130257805
Model  trigrams ; rand_score =  0.16176470588235295
Model  no_stop_words ; rand_score =  0.1492537313432836
Model  no_tomas_stemmed ; rand_score =  0.12347354138398917
Model  no_tomas_baker ; rand_score =  0.12347354138398917
Model  stemmed_txts ; rand_score =  0.12347354138398917
Model  primitive ; rand_score =  0.12347354138398917
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0737499999999
9998
Model  stanford_ner_txts ; rand_score =  0.051395007342143924
Model  lemmatized ; rand_score =  -0.058949624866023516
```

```
Model  no_tomas_stemmed_ent1 ; rand_score =  -0.060570071258907406
Model  identity_analysis_1 ; rand_score =  -0.060570071258907406
Model  bigrams ; rand_score =  -0.08302354399008675
Model  no_repeated_words ; rand_score =  -0.09615384615384613
########################################
########################################
```

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`.

In [17]:

```
tested_models = {}
top_cluster = 10
best_scores_all_clusters = {}
best_scores_Realcluster = {}
real_cluster_grouping = 4
init_scores = -999999
header_fields_compare = header_fields + ['comparing_mode']
csv_file = 'CSV_output/rangeclusters.csv'

with open(csv_file, 'a') as output_file:
    writer = csv.writer(output_file)
    writer.writerow(header_fields)
    new_row = []

    for cluster_mode in clustering_modes:
        best_scores_all_clusters[cluster_mode] = {}
        best_scores_Realcluster[cluster_mode] = {}

    for cluster in range(1,top_cluster+1):
        fix_grouping = cluster
        print('CLASIFICATION WITH ' + str(fix_grouping) + ' CLUSTERS')
        for cluster_mode in clustering_modes:
            tested_models[cluster_mode] = {}
            tested_models[cluster_mode]["primitive"] = cluster_texts(raw_texts,f
ix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["identity_analysis_1"] = cluster_texts(n
amed_ent_txts_1,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["stemmed_txts"] = cluster_texts(stemmed_
txts,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["no_tomas_baker"] = cluster_texts(texts_
no_thomas_baker,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["no_stop_words"] = cluster_texts(texts_n
o_stop_words,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["no_tomas_stemmed"] = cluster_texts(no_t
omas_stemmed_txts,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["no_tomas_stemmed_no_stop"] = cluster_te
```

```
xts(no_tomas_stemmed_no_stop_txts,fix_grouping,

    distanceFunction, cluster_mode)
            tested_models[cluster_mode]["no_tomas_stemmed_ent1"] =
cluster_texts(no_tomas_stemmed_ent1_txts,fix_grouping,distanceFunction,

 cluster_mode)
            tested_models[cluster_mode]["identity_analysis_2"] = cluster_texts(n
amed_ent_txts_2,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]["named_ent_2_no_tomas_barker"] = cluster
_texts(named_ent_2_no_tomas_barker_txts,
                                                                     fix_groupin
g,distanceFunction, cluster_mode)
            tested_models[cluster_mode]['bigrams'] =
cluster_texts(bigrams_texts,fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]['trigrams'] = cluster_texts(trigrams_tex
ts, fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]['lemmatized'] = cluster_texts(lemmatized
_texts, fix_grouping,distanceFunction, cluster_mode)
            tested_models[cluster_mode]['no_repeated_words'] = cluster_texts(no_
repeatedWords_noStopWords,
                                                                             fix
_grouping, distanceFunction, cluster_mode)


            tested_models[cluster_mode]['stanford_ner_txts'] = cluster_texts(sta
nford_ner_txts,fix_grouping,
                                                                         distanc
eFunction, cluster_mode)
            tested_models[cluster_mode]['stanford_ner_no_thomas_baker'] = cluste
r_texts(stanford_ner_no_thomas_baker,fix_grouping,
                                                                         distanc
eFunction, cluster_mode)


        # Evaluation
        tested_models_scores = {}

        for cluster_mode in tested_models:
            tested_models_scores[cluster_mode] = {}
            for model in tested_models[cluster_mode]:
                tested_models_scores[cluster_mode][model] =
adjusted_rand_score(reference,tested_models[cluster_mode][model])

                # Calculate the max score for each model, each custer amount
                if model in best_scores_all_clusters[cluster_mode].keys():
                    if best_scores_all_clusters[cluster_mode][model]['score'] <=
 tested_models_scores[cluster_mode][model]:
                        best_scores_all_clusters[cluster_mode][model]['cluster']
 = cluster
                        best_scores_all_clusters[cluster_mode][model]['score'] =
 tested_models_scores[cluster_mode][model]
                else:
                    best_scores_all_clusters[cluster_mode][model] = {}
                    best_scores_all_clusters[cluster_mode][model]['cluster'] = c
luster
                    best_scores_all_clusters[cluster_mode][model]['score'] = ini
t_scores

                if cluster == real_cluster_grouping:
                    best_scores_Realcluster[cluster_mode][model] = {}
```

```
                    best_scores_Realcluster[cluster_mode][model]['cluster'] = cl
uster
                    best_scores_Realcluster[cluster_mode][model]['score'] = test
ed_models_scores[cluster_mode][model]

        for cluster_mode in tested_models_scores:
            print("**************************************")
            print("Getting results for the clustering mode ", cluster_mode)
            for model in sorted(tested_models_scores[cluster_mode].items(),
key=operator.itemgetter(1), reverse=True):
                print("Model ", model[0], "; rand_score = ", model[1])

            print("########################################")
            print("########################################\n")

            new_row = [cluster_mode, model[0], model[1], cluster]
            writer.writerow(new_row)


csv_file = 'CSV_output/bestclusters.csv'
with  open(csv_file, 'a') as output_file:
    writer = csv.writer(output_file)
    writer.writerow(header_fields_compare)
    new_row = []
    for cluster_mode in clustering_modes:
        print('/////////////////////////////////')
        print('CLUSTER ALGORITHM: ', cluster_mode)
        for model in best_scores_all_clusters[cluster_mode].keys():

            print('*Model: "', model, '". Best cluster agroupation: ',
                      best_scores_all_clusters[cluster_mode][model]['cluster'],
                      ' clusters. Score: ',
                      str(best_scores_all_clusters[cluster_mode][model]
['score']))
            new_row = [cluster_mode, model,
                       best_scores_all_clusters[cluster_mode][model]['score'],
                      best_scores_all_clusters[cluster_mode][model]['cluster'],
'best']
            writer.writerow(new_row)

            print('++Score  for the real cluster agroupation (',str(real_cluster
_grouping),
                  ') in model ', model, ' is ',
                  str(best_scores_Realcluster[cluster_mode][model]['score']))
            new_row = [cluster_mode, model,
                       str(best_scores_Realcluster[cluster_mode][model]
['score']),
                       str(real_cluster_grouping), 'real']
            writer.writerow(new_row)
        print('/////////////////////////////////\n')
```

```
CLASIFICATION WITH 1 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_tomas_stemmed_ent1 ; rand_score =  0.0
Model  no_tomas_stemmed ; rand_score =  0.0
Model  trigrams ; rand_score =  0.0
Model  no_tomas_stemmed_no_stop ; rand_score =  0.0
Model  identity_analysis_1 ; rand_score =  0.0
Model  no_tomas_baker ; rand_score =  0.0
Model  lemmatized ; rand_score =  0.0
Model  stemmed_txts ; rand_score =  0.0
Model  primitive ; rand_score =  0.0
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0
Model  bigrams ; rand_score =  0.0
Model  named_ent_2_no_tomas_barker ; rand_score =  0.0
Model  identity_analysis_2 ; rand_score =  0.0
Model  no_repeated_words ; rand_score =  0.0
Model  no_stop_words ; rand_score =  0.0
Model  stanford_ner_txts ; rand_score =  0.0
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.0
Model  no_tomas_stemmed ; rand_score =  0.0
Model  trigrams ; rand_score =  0.0
Model  no_tomas_stemmed_no_stop ; rand_score =  0.0
Model  identity_analysis_1 ; rand_score =  0.0
Model  no_tomas_baker ; rand_score =  0.0
Model  lemmatized ; rand_score =  0.0
Model  stemmed_txts ; rand_score =  0.0
Model  primitive ; rand_score =  0.0
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0
Model  bigrams ; rand_score =  0.0
Model  named_ent_2_no_tomas_barker ; rand_score =  0.0
Model  identity_analysis_2 ; rand_score =  0.0
Model  no_repeated_words ; rand_score =  0.0
Model  no_stop_words ; rand_score =  0.0
Model  stanford_ner_txts ; rand_score =  0.0
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.0
Model  no_tomas_stemmed ; rand_score =  0.0
Model  trigrams ; rand_score =  0.0
Model  no_tomas_stemmed_no_stop ; rand_score =  0.0
Model  identity_analysis_1 ; rand_score =  0.0
Model  no_tomas_baker ; rand_score =  0.0
Model  lemmatized ; rand_score =  0.0
Model  stemmed_txts ; rand_score =  0.0
Model  primitive ; rand_score =  0.0
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0
Model  bigrams ; rand_score =  0.0
Model  named_ent_2_no_tomas_barker ; rand_score =  0.0
Model  identity_analysis_2 ; rand_score =  0.0
Model  no_repeated_words ; rand_score =  0.0
Model  no_stop_words ; rand_score =  0.0
Model  stanford_ner_txts ; rand_score =  0.0
```

```
########################################
########################################

CLASIFICATION WITH 2 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_tomas_stemmed_ent1 ; rand_score =  0.06676204101096801
Model  no_tomas_stemmed ; rand_score =  0.06676204101096801
Model  no_tomas_stemmed_no_stop ; rand_score =  0.06676204101096801
Model  lemmatized ; rand_score =  0.06676204101096801
Model  stemmed_txts ; rand_score =  0.06676204101096801
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0667620410109
6801
Model  named_ent_2_no_tomas_barker ; rand_score =  0.06676204101096
801
Model  identity_analysis_2 ; rand_score =  0.06676204101096801
Model  no_repeated_words ; rand_score =  0.06676204101096801
Model  stanford_ner_txts ; rand_score =  0.06676204101096801
Model  identity_analysis_1 ; rand_score =  -0.005037783375314868
Model  no_tomas_baker ; rand_score =  -0.022185246810870803
Model  primitive ; rand_score =  -0.022185246810870803
Model  no_stop_words ; rand_score =  -0.022185246810870803
Model  trigrams ; rand_score =  -0.06008583690987128
Model  bigrams ; rand_score =  -0.06008583690987128
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.06676204101096801
Model  no_tomas_stemmed ; rand_score =  0.06676204101096801
Model  trigrams ; rand_score =  0.06676204101096801
Model  no_tomas_stemmed_no_stop ; rand_score =  0.06676204101096801
Model  identity_analysis_1 ; rand_score =  0.06676204101096801
Model  no_tomas_baker ; rand_score =  0.06676204101096801
Model  lemmatized ; rand_score =  0.06676204101096801
Model  stemmed_txts ; rand_score =  0.06676204101096801
Model  primitive ; rand_score =  0.06676204101096801
Model  bigrams ; rand_score =  0.06676204101096801
Model  named_ent_2_no_tomas_barker ; rand_score =  0.06676204101096
801
Model  identity_analysis_2 ; rand_score =  0.06676204101096801
Model  no_repeated_words ; rand_score =  0.06676204101096801
Model  no_stop_words ; rand_score =  0.06676204101096801
Model  stanford_ner_txts ; rand_score =  -0.028624766645924053
Model  stanford_ner_no_thomas_baker ; rand_score =  -0.117647058823
52947
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.06676204101096801
Model  no_tomas_stemmed ; rand_score =  0.06676204101096801
Model  trigrams ; rand_score =  0.06676204101096801
Model  no_tomas_stemmed_no_stop ; rand_score =  0.06676204101096801
Model  identity_analysis_1 ; rand_score =  0.06676204101096801
Model  no_tomas_baker ; rand_score =  0.06676204101096801
Model  lemmatized ; rand_score =  0.06676204101096801
Model  stemmed_txts ; rand_score =  0.06676204101096801
Model  primitive ; rand_score =  0.06676204101096801
```

```
Model  bigrams ; rand_score =  0.06676204101096801
Model  no_repeated_words ; rand_score =  0.06676204101096801
Model  no_stop_words ; rand_score =  0.06676204101096801
Model  named_ent_2_no_tomas_barker ; rand_score =  0.01431127012522
3617
Model  stanford_ner_txts ; rand_score =  -0.028624766645924053
Model  identity_analysis_2 ; rand_score =  -0.05723370429252788
Model  stanford_ner_no_thomas_baker ; rand_score =  -0.117647058823
52947
########################################
########################################

CLASIFICATION WITH 3 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  named_ent_2_no_tomas_barker ; rand_score =  0.63414634146341
46
Model  no_tomas_stemmed_ent1 ; rand_score =  0.5140073081607796
Model  stanford_ner_no_thomas_baker ; rand_score =  0.3815937149270
4824
Model  trigrams ; rand_score =  0.27565982404692085
Model  no_repeated_words ; rand_score =  0.23918846769887883
Model  stanford_ner_txts ; rand_score =  0.1354903943377149
Model  bigrams ; rand_score =  0.09759271307742352
Model  identity_analysis_2 ; rand_score =  0.05659369994660974
Model  identity_analysis_1 ; rand_score =  0.001011122345803871
Model  lemmatized ; rand_score =  0.001011122345803871
Model  no_tomas_stemmed ; rand_score =  -0.002244668911335642
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0022446689113356
42
Model  stemmed_txts ; rand_score =  -0.002244668911335642
Model  no_stop_words ; rand_score =  -0.06576402321083177
Model  no_tomas_baker ; rand_score =  -0.0809384164222874
Model  primitive ; rand_score =  -0.11351017890191237
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  named_ent_2_no_tomas_barker ; rand_score =  0.50326797385620
91
Model  lemmatized ; rand_score =  0.23918846769887883
Model  stemmed_txts ; rand_score =  0.14703968770331813
Model  stanford_ner_txts ; rand_score =  0.05673758865248223
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0224719101123
5955
Model  no_tomas_stemmed ; rand_score =  0.001011122345803871
Model  trigrams ; rand_score =  0.001011122345803871
Model  no_tomas_stemmed_no_stop ; rand_score =  0.00101112234580387
1
Model  no_tomas_baker ; rand_score =  0.001011122345803871
Model  primitive ; rand_score =  0.001011122345803871
Model  bigrams ; rand_score =  0.001011122345803871
Model  no_repeated_words ; rand_score =  0.001011122345803871
Model  no_tomas_stemmed_ent1 ; rand_score =  -0.03636363636363637
Model  identity_analysis_1 ; rand_score =  -0.03636363636363637
Model  identity_analysis_2 ; rand_score =  -0.044847837693539755
Model  no_stop_words ; rand_score =  -0.044847837693539755
########################################
########################################
```

```
************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  stanford_ner_txts ; rand_score =  0.15814587593728696
Model  identity_analysis_2 ; rand_score =  0.12570145903479232
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0224719101123
5955
Model  trigrams ; rand_score =  0.001011122345803871
Model  no_tomas_baker ; rand_score =  0.001011122345803871
Model  lemmatized ; rand_score =  0.001011122345803871
Model  primitive ; rand_score =  0.001011122345803871
Model  bigrams ; rand_score =  0.001011122345803871
Model  no_repeated_words ; rand_score =  0.001011122345803871
Model  no_stop_words ; rand_score =  0.001011122345803871
Model  no_tomas_stemmed_ent1 ; rand_score =  -0.002244668911335642
Model  identity_analysis_1 ; rand_score =  -0.03636363636363637
Model  no_tomas_stemmed ; rand_score =  -0.044847837693539755
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0448478376935397
55
Model  stemmed_txts ; rand_score =  -0.044847837693539755
Model  named_ent_2_no_tomas_barker ; rand_score =  -0.0448478376935
39755
########################################
########################################

CLASIFICATION WITH 4 CLUSTERS
************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_tomas_stemmed_ent1 ; rand_score =  0.7446236559139784
Model  named_ent_2_no_tomas_barker ; rand_score =  0.74462365591397
84
Model  trigrams ; rand_score =  0.49316851008458035
Model  stanford_ner_no_thomas_baker ; rand_score =  0.4809976247030
879
Model  stanford_ner_txts ; rand_score =  0.34076827757125155
Model  no_repeated_words ; rand_score =  0.33371040723981904
Model  bigrams ; rand_score =  0.20233998623537508
Model  identity_analysis_1 ; rand_score =  0.16176470588235295
Model  lemmatized ; rand_score =  0.06323687031082535
Model  identity_analysis_2 ; rand_score =  0.06323687031082535
Model  no_tomas_stemmed ; rand_score =  -0.060570071258907406
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0925000000000000
1
Model  stemmed_txts ; rand_score =  -0.09250000000000001
Model  no_stop_words ; rand_score =  -0.10854816824966075
Model  no_tomas_baker ; rand_score =  -0.1301115241635688
Model  primitive ; rand_score =  -0.1496421600520495
########################################
########################################

************************************
Getting results for the clustering mode  KMeans
Model  identity_analysis_2 ; rand_score =  0.2304469273743017
Model  no_tomas_stemmed_no_stop ; rand_score =  0.17503392130257805
Model  primitive ; rand_score =  0.17503392130257805
Model  no_stop_words ; rand_score =  0.1492537313432836
Model  no_tomas_stemmed_ent1 ; rand_score =  0.14703968770331813
Model  no_tomas_baker ; rand_score =  0.12347354138398917
Model  lemmatized ; rand_score =  0.10494931425163982
Model  no_repeated_words ; rand_score =  0.0836012861736335
Model  named_ent_2_no_tomas_barker ; rand_score =  0.07255936675461
744
```

```
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0551075268817
2045
Model  stanford_ner_txts ; rand_score =  0.051395007342143924
Model  no_tomas_stemmed ; rand_score =  0.020352781546811426
Model  stemmed_txts ; rand_score =  0.020352781546811426
Model  identity_analysis_1 ; rand_score =  -0.056987788331071888
Model  trigrams ; rand_score =  -0.058949624866023516
Model  bigrams ; rand_score =  -0.058949624866023516
########################################
########################################

*************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  named_ent_2_no_tomas_barker ; rand_score =  0.28351955307262
57
Model  identity_analysis_2 ; rand_score =  0.2835195530726257
Model  no_tomas_stemmed_no_stop ; rand_score =  0.17503392130257805
Model  trigrams ; rand_score =  0.16176470588235295
Model  no_stop_words ; rand_score =  0.1492537313432836
Model  no_tomas_stemmed ; rand_score =  0.12347354138398917
Model  no_tomas_baker ; rand_score =  0.12347354138398917
Model  stemmed_txts ; rand_score =  0.12347354138398917
Model  primitive ; rand_score =  0.12347354138398917
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0737499999999
9998
Model  stanford_ner_txts ; rand_score =  0.051395007342143924
Model  lemmatized ; rand_score =  -0.058949624866023516
Model  no_tomas_stemmed_ent1 ; rand_score =  -0.060570071258907406
Model  identity_analysis_1 ; rand_score =  -0.060570071258907406
Model  bigrams ; rand_score =  -0.08302354399008675
Model  no_repeated_words ; rand_score =  -0.09615384615384613
########################################
########################################

CLASIFICATION WITH 5 CLUSTERS
*************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_tomas_stemmed_ent1 ; rand_score =  0.8160442600276625
Model  stanford_ner_no_thomas_baker ; rand_score =  0.7777013076393
668
Model  named_ent_2_no_tomas_barker ; rand_score =  0.72099853157121
88
Model  trigrams ; rand_score =  0.47361477572559374
Model  no_repeated_words ; rand_score =  0.4538922155688623
Model  stanford_ner_txts ; rand_score =  0.3291298865069357
Model  bigrams ; rand_score =  0.24964739069111425
Model  lemmatized ; rand_score =  0.22634730538922154
Model  identity_analysis_2 ; rand_score =  0.22634730538922154
Model  identity_analysis_1 ; rand_score =  0.04161412358133669
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0277044854881266
37
Model  no_tomas_stemmed ; rand_score =  -0.10922787193973632
Model  no_stop_words ; rand_score =  -0.11917808219178079
Model  stemmed_txts ; rand_score =  -0.1279683377308707
Model  no_tomas_baker ; rand_score =  -0.16732026143790849
Model  primitive ; rand_score =  -0.16732026143790849
########################################
########################################

*************************************
Getting results for the clustering mode  KMeans
```

```
Model  lemmatized ; rand_score =  0.3955739972337483
Model  no_repeated_words ; rand_score =  0.2491017964071856
Model  no_tomas_stemmed_no_stop ; rand_score =  0.22284908321579688
Model  named_ent_2_no_tomas_barker ; rand_score =  0.21161825726141
08
Model  no_tomas_stemmed_ent1 ; rand_score =  0.12269129287598947
Model  identity_analysis_1 ; rand_score =  0.09792284866468841
Model  trigrams ; rand_score =  0.0898203592814371
Model  stanford_ner_txts ; rand_score =  0.06973293768545992
Model  identity_analysis_2 ; rand_score =  0.051395007342143924
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0416141235813
3669
Model  no_tomas_stemmed ; rand_score =  0.013353115727002946
Model  no_tomas_baker ; rand_score =  0.013353115727002946
Model  stemmed_txts ; rand_score =  0.013353115727002946
Model  primitive ; rand_score =  0.013353115727002946
Model  no_stop_words ; rand_score =  -0.08262108262108267
Model  bigrams ; rand_score =  -0.11130039750141958
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  named_ent_2_no_tomas_barker ; rand_score =  0.47361477572559
374
Model  identity_analysis_2 ; rand_score =  0.309593023255814
Model  no_repeated_words ; rand_score =  0.2491017964071856
Model  lemmatized ; rand_score =  0.22473320778405528
Model  no_tomas_stemmed_ent1 ; rand_score =  0.12269129287598947
Model  stanford_ner_no_thomas_baker ; rand_score =  0.1071953010279
0016
Model  stanford_ner_txts ; rand_score =  0.10719530102790016
Model  identity_analysis_1 ; rand_score =  0.04154302670623143
Model  bigrams ; rand_score =  0.022427440633245397
Model  no_stop_words ; rand_score =  -0.08262108262108267
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.1114934618031658
7
Model  no_tomas_baker ; rand_score =  -0.11149346180316587
Model  primitive ; rand_score =  -0.11149346180316587
Model  no_tomas_stemmed ; rand_score =  -0.1176470588235294
Model  stemmed_txts ; rand_score =  -0.1176470588235294
Model  trigrams ; rand_score =  -0.13772455089820362
########################################
########################################

CLASIFICATION WITH 6 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_repeated_words ; rand_score =  0.8019457956914525
Model  named_ent_2_no_tomas_barker ; rand_score =  0.79848484848484
85
Model  trigrams ; rand_score =  0.7852298417483045
Model  stanford_ner_no_thomas_baker ; rand_score =  0.7491313412091
731
Model  no_tomas_stemmed_ent1 ; rand_score =  0.6962209302325582
Model  identity_analysis_2 ; rand_score =  0.31412286257124766
Model  bigrams ; rand_score =  0.17920000000000003
Model  stanford_ner_txts ; rand_score =  0.17737430167597767
Model  lemmatized ; rand_score =  0.14542728635682156
Model  identity_analysis_1 ; rand_score =  0.029411764705882353
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0694927032661570
```

```
7
Model  no_stop_words ; rand_score =  -0.11149346180316587
Model  no_tomas_baker ; rand_score =  -0.13476263399693722
Model  primitive ; rand_score =  -0.13476263399693722
Model  no_tomas_stemmed ; rand_score =  -0.14579191517561302
Model  stemmed_txts ; rand_score =  -0.14579191517561302
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  lemmatized ; rand_score =  0.5642807505211953
Model  identity_analysis_2 ; rand_score =  0.3365921787709497
Model  named_ent_2_no_tomas_barker ; rand_score =  0.24101198402130
494
Model  no_tomas_stemmed_no_stop ; rand_score =  0.23666910153396642
Model  no_tomas_stemmed_ent1 ; rand_score =  0.19457956914523977
Model  identity_analysis_1 ; rand_score =  0.19457956914523977
Model  no_repeated_words ; rand_score =  0.16972767574414188
Model  stanford_ner_txts ; rand_score =  0.1562021439509954
Model  no_stop_words ; rand_score =  0.03148425787106445
Model  trigrams ; rand_score =  0.025332488917036135
Model  bigrams ; rand_score =  0.025332488917036135
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0158273381294
96365
Model  no_tomas_stemmed ; rand_score =  0.0029985007496251713
Model  no_tomas_baker ; rand_score =  0.0029985007496251713
Model  stemmed_txts ; rand_score =  0.0029985007496251713
Model  primitive ; rand_score =  0.0029985007496251713
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  no_repeated_words ; rand_score =  0.5642807505211953
Model  named_ent_2_no_tomas_barker ; rand_score =  0.37943015983321
754
Model  no_stop_words ; rand_score =  0.3448275862068965
Model  no_tomas_stemmed ; rand_score =  0.23666910153396642
Model  no_tomas_baker ; rand_score =  0.23666910153396642
Model  stemmed_txts ; rand_score =  0.23666910153396642
Model  no_tomas_stemmed_ent1 ; rand_score =  0.19457956914523977
Model  identity_analysis_2 ; rand_score =  0.14705882352941177
Model  primitive ; rand_score =  0.1256391526661797
Model  identity_analysis_1 ; rand_score =  0.11694152923538229
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0885608856088
5607
Model  stanford_ner_txts ; rand_score =  0.0693293142426526
Model  trigrams ; rand_score =  0.04394046775336641
Model  bigrams ; rand_score =  0.025332488917036135
Model  lemmatized ; rand_score =  -0.05784526391901662
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0588235294117647
05
########################################
########################################

CLASIFICATION WITH 7 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  no_repeated_words ; rand_score =  0.9306062819576333
Model  named_ent_2_no_tomas_barker ; rand_score =  0.78293983244478
```

```
29
Model  no_tomas_stemmed_ent1 ; rand_score =  0.6807888970051132
Model  stanford_ner_no_thomas_baker ; rand_score =  0.5963808025177
025
Model  trigrams ; rand_score =  0.5516680227827503
Model  stanford_ner_txts ; rand_score =  0.24669603524229075
Model  bigrams ; rand_score =  0.24503311258278143
Model  identity_analysis_2 ; rand_score =  0.16
Model  lemmatized ; rand_score =  0.14880000000000004
Model  identity_analysis_1 ; rand_score =  0.05760000000000002
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0879888268156424
Model  primitive ; rand_score =  -0.10342084327764517
Model  no_stop_words ; rand_score =  -0.10342084327764517
Model  no_tomas_stemmed ; rand_score =  -0.10894941634241244
Model  no_tomas_baker ; rand_score =  -0.10894941634241244
Model  stemmed_txts ; rand_score =  -0.10894941634241244
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  named_ent_2_no_tomas_barker ; rand_score =  0.39377431906614
785
Model  bigrams ; rand_score =  0.24591439688715952
Model  no_repeated_words ; rand_score =  0.19718309859154928
Model  lemmatized ; rand_score =  0.18829113924050633
Model  stanford_ner_txts ; rand_score =  0.1497152156224573
Model  stanford_ner_no_thomas_baker ; rand_score =  0.1370284834488
0678
Model  primitive ; rand_score =  0.13636363636363635
Model  no_tomas_stemmed ; rand_score =  0.11840000000000003
Model  no_tomas_baker ; rand_score =  0.11840000000000003
Model  stemmed_txts ; rand_score =  0.11840000000000003
Model  trigrams ; rand_score =  0.09765886287625417
Model  no_tomas_stemmed_ent1 ; rand_score =  0.0881195908733281
Model  identity_analysis_1 ; rand_score =  0.0881195908733281
Model  identity_analysis_2 ; rand_score =  0.047732696897374714
Model  no_tomas_stemmed_no_stop ; rand_score =  0.02832415420928404
Model  no_stop_words ; rand_score =  -0.004882017900732276
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  no_repeated_words ; rand_score =  0.4587289992695398
Model  stanford_ner_no_thomas_baker ; rand_score =  0.2459143968871
5952
Model  no_stop_words ; rand_score =  0.24591439688715952
Model  lemmatized ; rand_score =  0.22474916387959867
Model  identity_analysis_1 ; rand_score =  0.1287208366854385
Model  identity_analysis_2 ; rand_score =  0.10798122065727699
Model  trigrams ; rand_score =  0.09765886287625417
Model  bigrams ; rand_score =  0.09765886287625417
Model  no_tomas_stemmed_ent1 ; rand_score =  0.08834729626808834
Model  no_tomas_stemmed_no_stop ; rand_score =  0.04069329314242651
Model  named_ent_2_no_tomas_barker ; rand_score =  0.04069329314242
651
Model  stanford_ner_txts ; rand_score =  0.030464584920030454
Model  no_tomas_stemmed ; rand_score =  -0.007575757575757586
Model  no_tomas_baker ; rand_score =  -0.007575757575757586
Model  stemmed_txts ; rand_score =  -0.007575757575757586
```

```
Model  primitive ; rand_score =  -0.0075757575757575 86
########################################
########################################

CLASIFICATION WITH 8 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  named_ent_2_no_tomas_barker ; rand_score =  0.79632465543644
73
Model  no_repeated_words ; rand_score =  0.6825775656324583
Model  trigrams ; rand_score =  0.5465020576131687
Model  no_tomas_stemmed_ent1 ; rand_score =  0.3953098827470687
Model  stanford_ner_no_thomas_baker ; rand_score =  0.3757371524852
5696
Model  identity_analysis_2 ; rand_score =  0.3418013856812933
Model  bigrams ; rand_score =  0.3034953111679454
Model  stanford_ner_txts ; rand_score =  0.16628175519630486
Model  identity_analysis_1 ; rand_score =  0.1407035175879397
Model  lemmatized ; rand_score =  0.0773662551440329
Model  no_tomas_stemmed ; rand_score =  -0.07605177993527505
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0760517799352750
5
Model  no_tomas_baker ; rand_score =  -0.07605177993527505
Model  stemmed_txts ; rand_score =  -0.07605177993527505
Model  primitive ; rand_score =  -0.07605177993527505
Model  no_stop_words ; rand_score =  -0.07605177993527505
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  lemmatized ; rand_score =  0.35987748851454826
Model  trigrams ; rand_score =  0.2287822878228782
Model  named_ent_2_no_tomas_barker ; rand_score =  0.14971521562245
73
Model  stanford_ner_txts ; rand_score =  0.1407035175879397
Model  no_repeated_words ; rand_score =  0.13842482100238665
Model  identity_analysis_2 ; rand_score =  0.0983050847457627
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0885608856088
5607
Model  no_tomas_stemmed_ent1 ; rand_score =  0.06891271056661562
Model  identity_analysis_1 ; rand_score =  0.06710310965630112
Model  no_stop_words ; rand_score =  0.06557377049180328
Model  bigrams ; rand_score =  0.05822187254130607
Model  no_tomas_stemmed_no_stop ; rand_score =  0.05695687550854355
Model  no_tomas_stemmed ; rand_score =  0.04609053497942384
Model  stemmed_txts ; rand_score =  0.04522613065326634
Model  no_tomas_baker ; rand_score =  -0.08227848101265822
Model  primitive ; rand_score =  -0.08227848101265822
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.3469721767594108
Model  lemmatized ; rand_score =  0.2568265682656827
Model  no_repeated_words ; rand_score =  0.2287822878228782
Model  named_ent_2_no_tomas_barker ; rand_score =  0.14880000000000
004
Model  identity_analysis_2 ; rand_score =  0.05822187254130607
Model  stanford_ner_txts ; rand_score =  0.05822187254130607
```

```
Model  no_tomas_baker ; rand_score =  0.048513302034428794
Model  primitive ; rand_score =  0.048513302034428794
Model  identity_analysis_1 ; rand_score =  0.047732696897374714
Model  bigrams ; rand_score =  0.03526093088857542
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0049099836333
87863
Model  no_stop_words ; rand_score =  -0.010954616588419399
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0294117647058823
53
Model  no_tomas_stemmed ; rand_score =  -0.033599999999999984
Model  stemmed_txts ; rand_score =  -0.033599999999999984
Model  trigrams ; rand_score =  -0.07193229901269399
########################################
########################################

CLASIFICATION WITH 9 CLUSTERS
**************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  named_ent_2_no_tomas_barker ; rand_score =  0.76470588235294
11
Model  no_repeated_words ; rand_score =  0.46547314578005117
Model  identity_analysis_2 ; rand_score =  0.4344328238133548
Model  trigrams ; rand_score =  0.43307757885763
Model  no_tomas_stemmed_ent1 ; rand_score =  0.2743055555555555
Model  stanford_ner_no_thomas_baker ; rand_score =  0.2743055555555
555
Model  bigrams ; rand_score =  0.26229508196721313
Model  stanford_ner_txts ; rand_score =  0.17647058823529413
Model  identity_analysis_1 ; rand_score =  0.09836065573770492
Model  lemmatized ; rand_score =  0.0443307757885763
Model  no_tomas_stemmed_no_stop ; rand_score =  0.02229845626072041
Model  no_stop_words ; rand_score =  0.02229845626072041
Model  no_tomas_stemmed ; rand_score =  -0.0572831423895254
Model  stemmed_txts ; rand_score =  -0.0572831423895254
Model  no_tomas_baker ; rand_score =  -0.11028806584362143
Model  primitive ; rand_score =  -0.11028806584362143
########################################
########################################

**************************************
Getting results for the clustering mode  KMeans
Model  bigrams ; rand_score =  0.27040000000000003
Model  no_tomas_stemmed_ent1 ; rand_score =  0.25932203389830505
Model  stemmed_txts ; rand_score =  0.1506622516556291
Model  named_ent_2_no_tomas_barker ; rand_score =  0.15066225165562
91
Model  no_repeated_words ; rand_score =  0.14705882352941177
Model  lemmatized ; rand_score =  0.1287208366854385
Model  identity_analysis_2 ; rand_score =  0.11962931760741363
Model  trigrams ; rand_score =  0.0877483443708609
Model  stanford_ner_txts ; rand_score =  0.08733624454148473
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0763888888888
8888
Model  no_tomas_stemmed_no_stop ; rand_score =  0.06610169491525422
Model  identity_analysis_1 ; rand_score =  0.06610169491525422
Model  primitive ; rand_score =  0.056291390728476796
Model  no_tomas_stemmed ; rand_score =  0.023588879528222396
Model  no_stop_words ; rand_score =  0.0
Model  no_tomas_baker ; rand_score =  -0.008424599831508015
########################################
########################################
```

```
************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  lemmatized ; rand_score =  0.3235294117647059
Model  no_tomas_stemmed_ent1 ; rand_score =  0.25932203389830505
Model  stemmed_txts ; rand_score =  0.2476832350463353
Model  identity_analysis_1 ; rand_score =  0.23870417732310314
Model  stanford_ner_no_thomas_baker ; rand_score =  0.1821192052980
1323
Model  named_ent_2_no_tomas_barker ; rand_score =  0.09814963797264
685
Model  no_repeated_words ; rand_score =  0.09814963797264685
Model  identity_analysis_2 ; rand_score =  0.0877483443708609
Model  no_stop_words ; rand_score =  0.0877483443708609
Model  no_tomas_stemmed ; rand_score =  0.05560235888795281
Model  stanford_ner_txts ; rand_score =  0.05560235888795281
Model  no_tomas_stemmed_no_stop ; rand_score =  0.00169491525423727
9
Model  trigrams ; rand_score =  -0.01483679525222554
Model  no_tomas_baker ; rand_score =  -0.024135156878519692
Model  primitive ; rand_score =  -0.024135156878519692
Model  bigrams ; rand_score =  -0.029411764705882353
########################################
########################################

CLASIFICATION WITH 10 CLUSTERS
************************************
Getting results for the clustering mode  AgglomerativeClustering
Model  named_ent_2_no_tomas_barker ; rand_score =  0.52261306532663
32
Model  trigrams ; rand_score =  0.44596912521440824
Model  no_repeated_words ; rand_score =  0.44596912521440824
Model  stanford_ner_txts ; rand_score =  0.3807890222984563
Model  identity_analysis_2 ; rand_score =  0.3316582914572865
Model  stanford_ner_no_thomas_baker ; rand_score =  0.2987697715289
9825
Model  no_tomas_stemmed_ent1 ; rand_score =  0.22241992882562278
Model  bigrams ; rand_score =  0.1768346595932803
Model  lemmatized ; rand_score =  0.1324977618621307
Model  identity_analysis_1 ; rand_score =  0.06502636203866434
Model  no_tomas_stemmed_no_stop ; rand_score =  0.04340277777777777
6
Model  no_stop_words ; rand_score =  -0.012227074235807853
Model  no_tomas_stemmed ; rand_score =  -0.07901234567901237
Model  stemmed_txts ; rand_score =  -0.07901234567901237
Model  no_tomas_baker ; rand_score =  -0.12323064113238967
Model  primitive ; rand_score =  -0.12323064113238967
########################################
########################################

************************************
Getting results for the clustering mode  KMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.1868995633187773
Model  identity_analysis_2 ; rand_score =  0.1753472222222222
Model  lemmatized ; rand_score =  0.15480427046263345
Model  bigrams ; rand_score =  0.109375
Model  stanford_ner_txts ; rand_score =  0.09847806624888092
Model  stanford_ner_no_thomas_baker ; rand_score =  0.0874785591766
7238
Model  trigrams ; rand_score =  0.07766990291262138
Model  no_repeated_words ; rand_score =  0.07766990291262138
```

```
Model  no_tomas_stemmed ; rand_score =  0.07672634271099743
Model  named_ent_2_no_tomas_barker ; rand_score =  0.04522613065326
634
Model  primitive ; rand_score =  0.02229845626072041
Model  identity_analysis_1 ; rand_score =  0.02096069868995634
Model  no_tomas_stemmed_no_stop ; rand_score =  0.01041666666666667
Model  no_tomas_baker ; rand_score =  0.01041666666666667
Model  stemmed_txts ; rand_score =  0.01041666666666667
Model  no_stop_words ; rand_score =  0.01041666666666667
########################################
########################################

**************************************
Getting results for the clustering mode  MiniBatchKMeans
Model  no_tomas_stemmed_ent1 ; rand_score =  0.1868995633187773
Model  stemmed_txts ; rand_score =  0.1753472222222222
Model  named_ent_2_no_tomas_barker ; rand_score =  0.17534722222222
22
Model  stanford_ner_txts ; rand_score =  0.15480427046263345
Model  bigrams ; rand_score =  0.1537117903930131
Model  lemmatized ; rand_score =  0.13114754098360656
Model  stanford_ner_no_thomas_baker ; rand_score =  0.1205240174672
4892
Model  no_tomas_stemmed ; rand_score =  0.08733624454148473
Model  no_tomas_baker ; rand_score =  0.07672634271099743
Model  primitive ; rand_score =  0.07672634271099743
Model  identity_analysis_2 ; rand_score =  0.0548885077186964
Model  no_repeated_words ; rand_score =  0.04522613065326634
Model  identity_analysis_1 ; rand_score =  0.02096069868995634
Model  no_tomas_stemmed_no_stop ; rand_score =  -0.0204603580562659
9
Model  no_stop_words ; rand_score =  -0.02046035805626599
Model  trigrams ; rand_score =  -0.04980544747081712
########################################
########################################

////////////////////////////////
CLUSTER ALGORITHM:  AgglomerativeClustering
*Model: " no_tomas_stemmed_ent1 ". Best cluster agroupation:  5  cl
usters. Score:  0.8160442600276625
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed_ent1  is  0.7446236559139784
*Model: " no_tomas_stemmed ". Best cluster agroupation:  2  cluster
s. Score:  0.06676204101096801
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed  is  -0.060570071258907406
*Model: " trigrams ". Best cluster agroupation:  6  clusters. Scor
e:  0.7852298417483045
++Score  for the real cluster agroupation ( 4 ) in model  trigrams
is  0.49316851008458035
*Model: " no_tomas_stemmed_no_stop ". Best cluster agroupation:  2
clusters. Score:  0.06676204101096801
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed_no_stop  is  -0.09250000000000001
*Model: " identity_analysis_1 ". Best cluster agroupation:  4  clus
ters. Score:  0.16176470588235295
++Score  for the real cluster agroupation ( 4 ) in model  identity_
analysis_1  is  0.16176470588235295
*Model: " no_tomas_baker ". Best cluster agroupation:  2  clusters.
Score:  -0.022185246810870803
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
```

```
baker  is  -0.1301115241635688
*Model: " lemmatized ". Best cluster agroupation:  5  clusters. Sco
re:  0.22634730538922154
++Score  for the real cluster agroupation ( 4 ) in model  lemmatize
d  is  0.06323687031082535
*Model: " stemmed_txts ". Best cluster agroupation:  2  clusters. S
core:  0.06676204101096801
++Score  for the real cluster agroupation ( 4 ) in model  stemmed_t
xts  is  -0.09250000000000001
*Model: " primitive ". Best cluster agroupation:  2  clusters. Scor
e:  -0.022185246810870803
++Score  for the real cluster agroupation ( 4 ) in model  primitive
  is  -0.1496421600520495
*Model: " stanford_ner_no_thomas_baker ". Best cluster agroupation:
  5  clusters. Score:  0.7777013076393668
++Score  for the real cluster agroupation ( 4 ) in model  stanford_
ner_no_thomas_baker  is  0.4809976247030879
*Model: " bigrams ". Best cluster agroupation:  8  clusters. Score:
  0.3034953111679454
++Score  for the real cluster agroupation ( 4 ) in model  bigrams
 is  0.20233998623537508
*Model: " named_ent_2_no_tomas_barker ". Best cluster agroupation:
6  clusters. Score:  0.7984848484848485
++Score  for the real cluster agroupation ( 4 ) in model  named_ent
_2_no_tomas_barker  is  0.7446236559139784
*Model: " identity_analysis_2 ". Best cluster agroupation:  9  clus
ters. Score:  0.4344328238133548
++Score  for the real cluster agroupation ( 4 ) in model  identity_
analysis_2  is  0.06323687031082535
*Model: " no_repeated_words ". Best cluster agroupation:  7  cluste
rs. Score:  0.9306062819576333
++Score  for the real cluster agroupation ( 4 ) in model  no_repeat
ed_words  is  0.33371040723981904
*Model: " no_stop_words ". Best cluster agroupation:  9  clusters.
 Score:  0.022298456260720410
++Score  for the real cluster agroupation ( 4 ) in model  no_stop_w
ords  is  -0.10854816824966075
*Model: " stanford_ner_txts ". Best cluster agroupation:  10  clust
ers. Score:  0.3807890222984563
++Score  for the real cluster agroupation ( 4 ) in model  stanford_
ner_txts  is  0.34076827757125155
/////////////////////////////////

/////////////////////////////////
CLUSTER ALGORITHM:  KMeans
*Model: " no_tomas_stemmed_ent1 ". Best cluster agroupation:  9  cl
usters. Score:  0.25932203389830505
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed_ent1  is  0.14703968770331813
*Model: " no_tomas_stemmed ". Best cluster agroupation:  7  cluster
s. Score:  0.11840000000000003
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed  is  0.020352781546811426
*Model: " trigrams ". Best cluster agroupation:  8  clusters. Scor
e:  0.2287822878228782
++Score  for the real cluster agroupation ( 4 ) in model  trigrams
is  -0.058949624866023516
*Model: " no_tomas_stemmed_no_stop ". Best cluster agroupation:  6
clusters. Score:  0.23666910153396642
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed_no_stop  is  0.17503392130257805
```

```
*Model: " identity_analysis_1 ". Best cluster agroupation:  6  clus
ters. Score:  0.19457956914523977
++Score  for the real cluster agroupation ( 4 ) in model  identity_
analysis_1  is  -0.05698778833107188
*Model: " no_tomas_baker ". Best cluster agroupation:  4  clusters.
Score:  0.12347354138398917
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
baker  is  0.12347354138398917
*Model: " lemmatized ". Best cluster agroupation:  6  clusters. Sco
re:  0.5642807505211953
++Score  for the real cluster agroupation ( 4 ) in model  lemmatize
d  is  0.10494931425163982
*Model: " stemmed_txts ". Best cluster agroupation:  9  clusters. S
core:  0.1506622516556291
++Score  for the real cluster agroupation ( 4 ) in model  stemmed_t
xts  is  0.020352781546811426
*Model: " primitive ". Best cluster agroupation:  4  clusters. Scor
e:  0.17503392130257805
++Score  for the real cluster agroupation ( 4 ) in model  primitive
  is  0.17503392130257805
*Model: " stanford_ner_no_thomas_baker ". Best cluster agroupation:
  7  clusters. Score:  0.13702848344880678
++Score  for the real cluster agroupation ( 4 ) in model  stanford_
ner_no_thomas_baker  is  0.05510752688172045
*Model: " bigrams ". Best cluster agroupation:  9  clusters. Score:
  0.27040000000000003
++Score  for the real cluster agroupation ( 4 ) in model  bigrams
 is  -0.058949624866023516
*Model: " named_ent_2_no_tomas_barker ". Best cluster agroupation:
3  clusters. Score:  0.5032679738562091
++Score  for the real cluster agroupation ( 4 ) in model  named_ent
_2_no_tomas_barker  is  0.07255936675461744
*Model: " identity_analysis_2 ". Best cluster agroupation:  6  clus
ters. Score:  0.3365921787709497
++Score  for the real cluster agroupation ( 4 ) in model  identity_
analysis_2  is  0.2304469273743017
*Model: " no_repeated_words ". Best cluster agroupation:  5  cluste
rs. Score:  0.2491017964071856
++Score  for the real cluster agroupation ( 4 ) in model  no_repeat
ed_words  is  0.0836012861736335
*Model: " no_stop_words ". Best cluster agroupation:  4  clusters.
 Score:  0.1492537313432836
++Score  for the real cluster agroupation ( 4 ) in model  no_stop_w
ords  is  0.1492537313432836
*Model: " stanford_ner_txts ". Best cluster agroupation:  6  cluste
rs. Score:  0.1562021439509954
++Score  for the real cluster agroupation ( 4 ) in model  stanford_
ner_txts  is  0.051395007342143924
/////////////////////////////////

/////////////////////////////////
CLUSTER ALGORITHM:  MiniBatchKMeans
*Model: " no_tomas_stemmed_ent1 ". Best cluster agroupation:  8  cl
usters. Score:  0.3469721767594108
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed_ent1  is  -0.060570071258907406
*Model: " no_tomas_stemmed ". Best cluster agroupation:  6  cluster
s. Score:  0.23666910153396642
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed  is  0.12347354138398917
*Model: " trigrams ". Best cluster agroupation:  4  clusters. Scor
```

```
e:  0.16176470588235295
++Score  for the real cluster agroupation ( 4 ) in model  trigrams
is  0.16176470588235295
*Model: " no_tomas_stemmed_no_stop ". Best cluster agroupation:  4
clusters. Score:  0.17503392130257805
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
stemmed_no_stop  is  0.17503392130257805
*Model: " identity_analysis_1 ". Best cluster agroupation:  9  clus
ters. Score:  0.23870417732310314
++Score  for the real cluster agroupation ( 4 ) in model  identity_
analysis_1  is  -0.060570071258907406
*Model: " no_tomas_baker ". Best cluster agroupation:  6  clusters.
Score:  0.23666910153396642
++Score  for the real cluster agroupation ( 4 ) in model  no_tomas_
baker  is  0.12347354138398917
*Model: " lemmatized ". Best cluster agroupation:  9  clusters. Sco
re:  0.3235294117647059
++Score  for the real cluster agroupation ( 4 ) in model  lemmatize
d  is  -0.058949624866023516
*Model: " stemmed_txts ". Best cluster agroupation:  9  clusters. S
core:  0.2476832350463353
++Score  for the real cluster agroupation ( 4 ) in model  stemmed_t
xts  is  0.12347354138398917
*Model: " primitive ". Best cluster agroupation:  6  clusters. Scor
e:  0.1256391526661797
++Score  for the real cluster agroupation ( 4 ) in model  primitive
  is  0.12347354138398917
*Model: " stanford_ner_no_thomas_baker ". Best cluster agroupation:
  7  clusters. Score:  0.24591439688715952
++Score  for the real cluster agroupation ( 4 ) in model  stanford_
ner_no_thomas_baker  is  0.07374999999999998
*Model: " bigrams ". Best cluster agroupation:  10  clusters. Scor
e:  0.1537117903930131
++Score  for the real cluster agroupation ( 4 ) in model  bigrams
 is  -0.08302354399008675
*Model: " named_ent_2_no_tomas_barker ". Best cluster agroupation:
5  clusters. Score:  0.47361477572559374
++Score  for the real cluster agroupation ( 4 ) in model  named_ent
_2_no_tomas_barker  is  0.2835195530726257
*Model: " identity_analysis_2 ". Best cluster agroupation:  5  clus
ters. Score:  0.309593023255814
++Score  for the real cluster agroupation ( 4 ) in model  identity_
analysis_2  is  0.2835195530726257
*Model: " no_repeated_words ". Best cluster agroupation:  6  cluste
rs. Score:  0.5642807505211953
++Score  for the real cluster agroupation ( 4 ) in model  no_repeat
ed_words  is  -0.09615384615384613
*Model: " no_stop_words ". Best cluster agroupation:  6  clusters.
 Score:  0.3448275862068965
++Score  for the real cluster agroupation ( 4 ) in model  no_stop_w
ords  is  0.1492537313432836
*Model: " stanford_ner_txts ". Best cluster agroupation:  3  cluste
rs. Score:  0.15814587593728696
++Score  for the real cluster agroupation ( 4 ) in model  stanford_
ner_txts  is  0.051395007342143924
/////////////////////////////////
```

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` correpondiente 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.

In [18]:

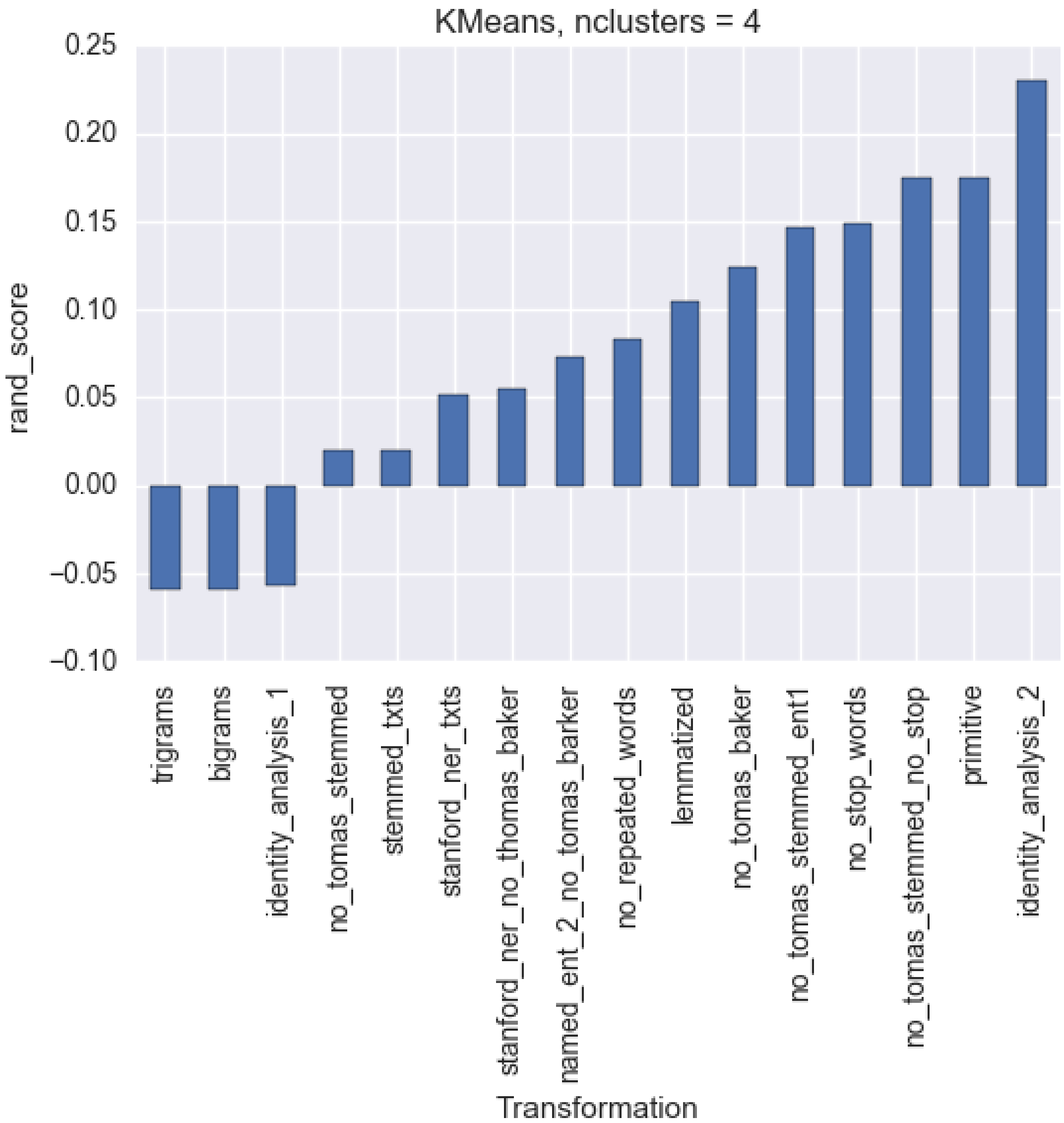
```
# Import the data corresponding to the 4clusters.csv file
data_4_clusters = pd.read_csv('CSV_output/4clusters.csv')
data_4_clusters_AC = data_4_clusters.loc[data_4_clusters.clustering_mode == "Agg
lomerativeClustering"]
data_4_clusters_kmeans = data_4_clusters.loc[data_4_clusters.clustering_mode ==
"KMeans"]
data_4_clusters_MBKM = data_4_clusters.loc[data_4_clusters.clustering_mode == "M
iniBatchKMeans"]
```

In [19]:

```
# Plot data corresponding to the 4clusters.csv file: KMeans
data_4_clusters_kmeans.rand_score = data_4_clusters_kmeans.rand_score.astype(flo
at)
data_4_clusters_kmeans.cluster_split = data_4_clusters_kmeans.cluster_split.asty
pe(float)
data_4_clusters_kmeans = data_4_clusters_kmeans.sort_values(by="rand_score")
ax_kmeans_4 = data_4_clusters_kmeans.plot(x='model', y='rand_score', kind='bar',

                             title='KMeans, nclusters = 4', legend = False)
ax_kmeans_4.set_xlabel("Transformation")
ax_kmeans_4.set_ylabel("rand_score")
```

Out[19]:

```
<matplotlib.text.Text at 0x7f1cd7e6eb38>
```

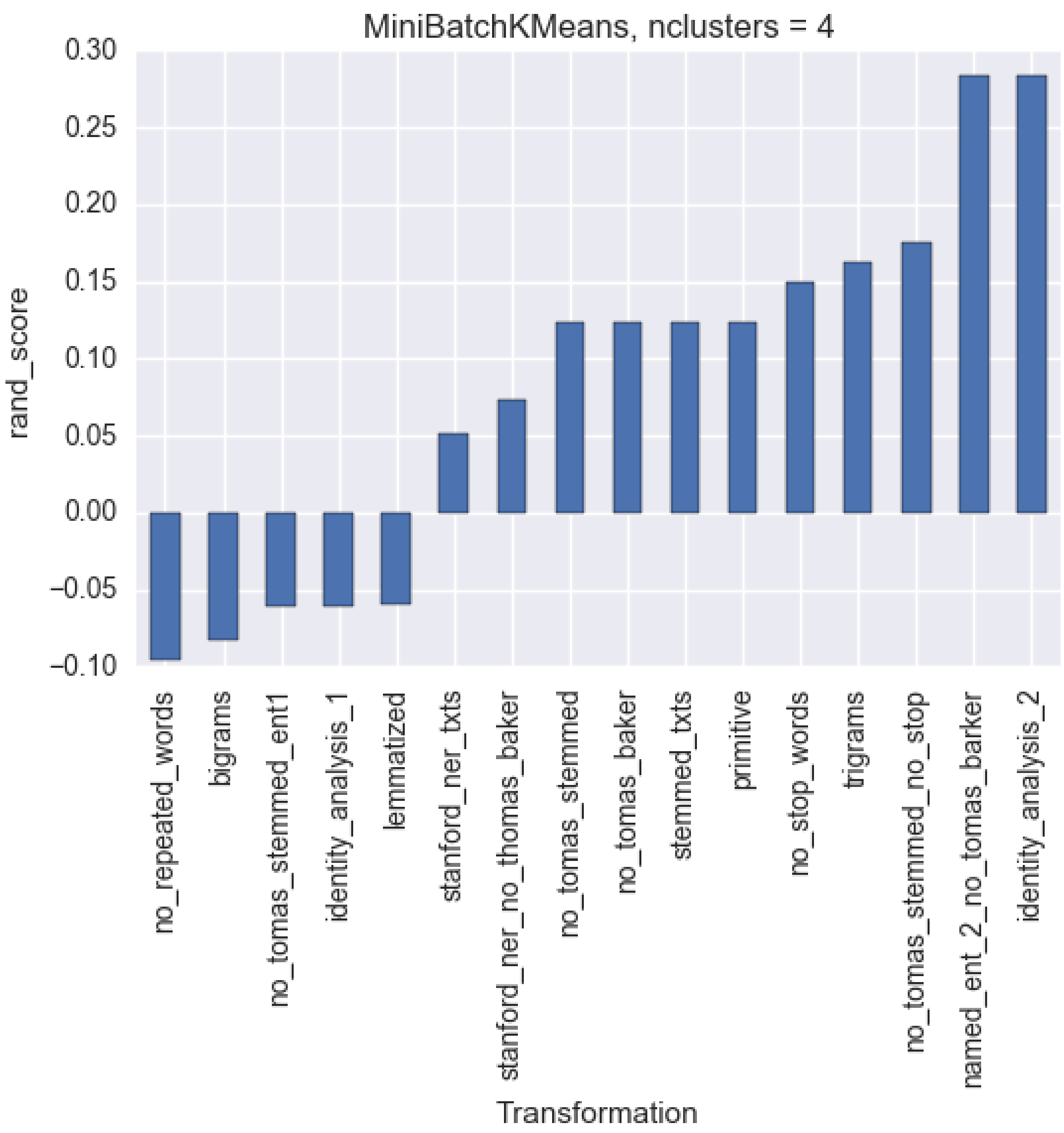


In [20]:

```
# Plot data corresponding to the 4clusters.csv file: MiniBatchKMeans
data_4_clusters_MBKM.rand_score = data_4_clusters_MBKM.rand_score.astype(float)
data_4_clusters_MBKM.cluster_split = data_4_clusters_MBKM.cluster_split.astype(f
loat)
data_4_clusters_MBKM = data_4_clusters_MBKM.sort_values(by="rand_score")
ax_MBKM_4 = data_4_clusters_MBKM.plot(x='model', y='rand_score', kind='bar',
                             title='MiniBatchKMeans, nclusters = 4', legend = Fa
lse)
ax_MBKM_4.set_xlabel("Transformation")
ax_MBKM_4.set_ylabel("rand_score")
```

Out[20]:

```
<matplotlib.text.Text at 0x7f1cde558898>
```

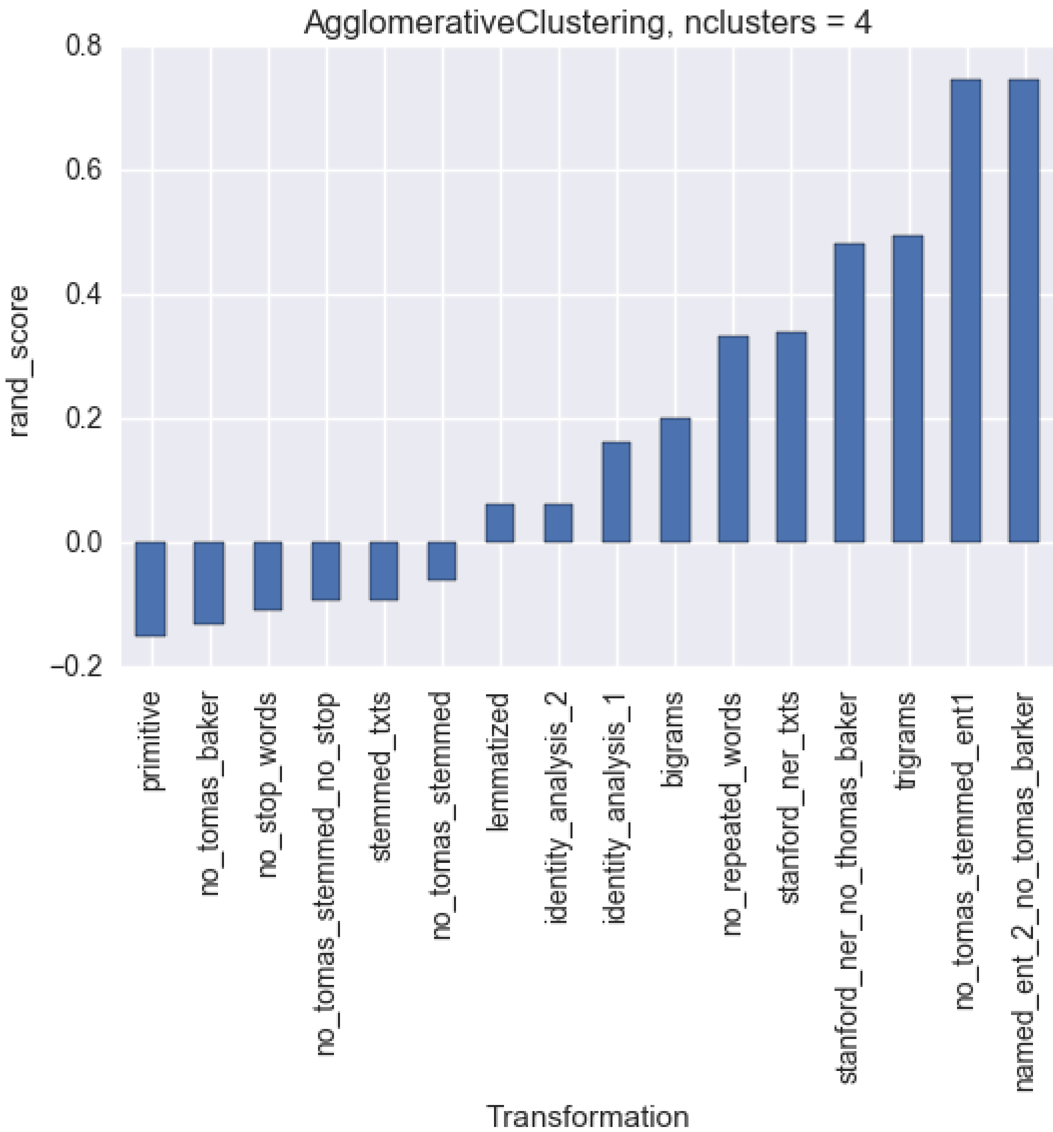


In [21]:

```
# Plot data corresponding to the 4clusters.csv file: AgglomerativeClustering
data_4_clusters_AC.rand_score = data_4_clusters_AC.rand_score.astype(float)
data_4_clusters_AC.cluster_split =
data_4_clusters_AC.cluster_split.astype(float)
data_4_clusters_AC = data_4_clusters_AC.sort_values(by="rand_score")
ax_ac_4 = data_4_clusters_AC.plot(x='model', y='rand_score', kind='bar',
                             title='AgglomerativeClustering, nclusters = 4', leg
end = False)
ax_ac_4.set_xlabel("Transformation")
ax_ac_4.set_ylabel("rand_score")
```

Out[21]:

```
<matplotlib.text.Text at 0x7f1cde5105f8>
```



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.

In [32]:

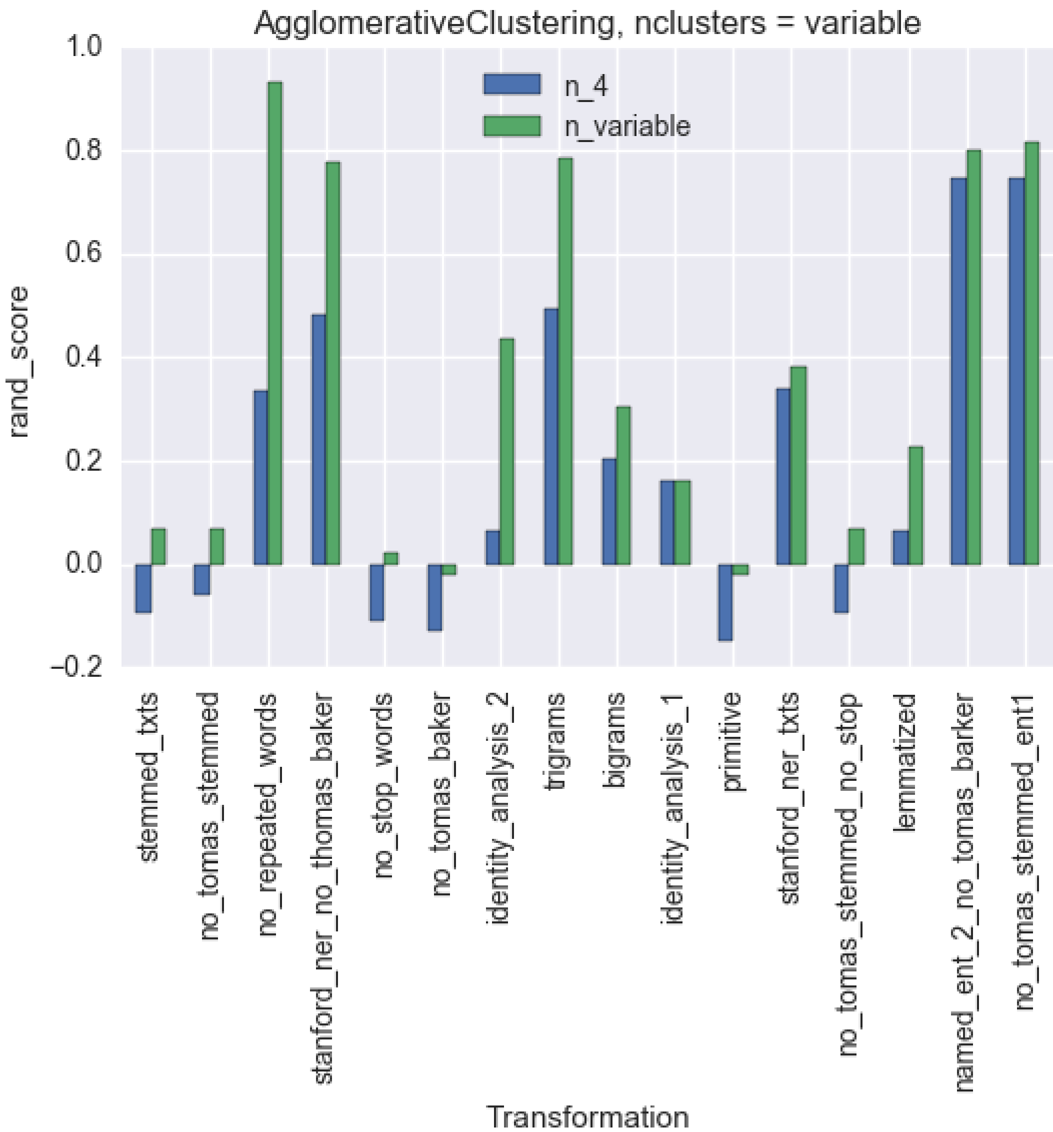
```
data_n_clusters = pd.read_csv('CSV_output/bestclusters.csv')
data_n_clusters_AC = data_n_clusters.loc[data_n_clusters.clustering_mode == "Agg
lomerativeClustering"]
list_data = []
index = []
fil_keys = {"real":"n_4", "best":"n_variable"}
comp_modes = data_n_clusters_AC.comparing_mode.unique()
for model in data_n_clusters_AC.model.unique():
    curr_data = {}
    for mode in comp_modes:
        value = data_n_clusters_AC[(data_n_clusters_AC["model"]==model)
                                   &
(data_n_clusters_AC["comparing_mode"]==mode)]["rand_score"].values[0]
        curr_data[fil_keys[mode]] = value
    list_data.append(curr_data)
    index.append(model)

new_df = pd.DataFrame(list_data, index=index)
new_df = new_df.iloc[::-1]
new_df[['n_4','n_variable']] = new_df[['n_4','n_variable']].apply(pd.to_numeric)
```

In [33]:

```
ax_AC_n = new_df.plot(kind="bar", title='AgglomerativeClustering, nclusters = va
riable')
ax_AC_n.set_xlabel("Transformation")
ax_AC_n.set_ylabel("rand_score")
```

Out[33]:

```
<matplotlib.text.Text at 0x7f1cde24ef28>
```



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.