Marina Kurmanova
Master en Data Science, Universidad Rey Juan Carlos

Práctica Clustering de Noticias

Recuperación y Minería de Textos

Contenidos

1 Introducción....3
2 Pruebas....4
2.1 *Procesado sin diferenciar el idioma con gold standard corregido. Fichero BasicNewsClustering_1.py* 5
2.1.1 Lematización....7
2.1.2 Puntuación....8
2.1.3 Stopwords....8
2.1.4 Named Entities....9
2.1.5 Procesado sin diferenciar el idioma con gold standard original. Fichero *BasicNewsClustering_0'.py*....12
2.2 Procesado de textos traducidos a inglés. Fichero *BasicNewsClustering_2_{0,1}.py*....13
2.2.1 Traducción de textos a inglés. Fichero *BasicNewsClustering_2_0.py*....13
2.2.2 Repetición de pruebas sobre los textos traducidos : todos los textos están en inglés. Fichero *BasicNewsClustering_2_1.py*....13
2.2.2.1 Prueba 1. Lematización....13
2.2.2.2 Prueba 2. Quitar puntuación....14
2.2.2.3 Prueba 3. Quitar stopwords....14
2.2.2.4 Prueba 4. Named entities....14
2.2.3 TF vs TF_IDF....14
2.3 Resultados y conclusiones....15

1 Introducción

Clasificación o categorización de textos consiste en asignar un único textó a una o varias categorías de una taxonomía predefinida. Crear modelos de clasificación requiere entrenar un motor con textos manualmente preclasificados o definir una serie de relgas para cada categoría (lo cual se conoce como aprendizaje supervisado).

En cambio, clustering es generalmente realizado simultáneamente en un conjunto de documentos organizándolos por grupos de acuerdo a su semejanza. Además, esto no depende de una taxonomía predefinida : la decisión de si el texto pertenece a un grupo u otro se hace dinámicamente se basa en contenidos de este conjunto de documentos en particular. Por tanto, clustering no requiere definición previa de una taxonomía, ni de entrenar un modelo o definir reglas, lo cual se denomina aprendizaje no supervisado. Clasificación y clustering son dos métodos complementarios. Clasificiación es más apropiada cuando la estructura del conjunto de documentos es conocida a priori y el objetivo es analizar cada documento en particular. Clustering require analizar simultáneamente un set de documentos y el resultado cambia si el conjunto cambia, pero ofrece el potencial de descubrir una implícita estructura y temas que emergen de los documentos. En general clustering permite obtener más insigts inesperados y codificarlos usando los mismos términos que aparecen en los textos.

En la presente práctica se realiza clustering de un conjunto de textos sobre noticias y se propone una clasificación óptima, el llamado « golden standard«. El objetivo es que se limpie el texto y nos quedemos con aquellos componentes del texto que lleven la máxima información de cara algoritmo de clustering. Para medir la aproximación de nuestro código al estándar se usa la métrica de ARI, que no es más que el ratio de parecido de lo que obtenemos entre el estándar, siendo el máximo 1.

[https://www.meaningcloud.com/products/text-clustering]

2 Pruebas

Para evaluar el resultado de clustering se propone emplear una métrica llamada Adjusted Rand Index, ARI. La versión original del código de la práctica da un valor de ARI de aproximadamente -0.024 :

```
Vectors created.
('test: ', array([1, 1, 0, 1, 1, 1, 2, 0, 1, 1, 1, 0, 1, 1, 4, 1, 1, 0, 0, 0, 0, 3]))
('reference: ', [0, 1, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 4, 5, 2])
('rand_score: ', -0.02448110697179351)
```

Se busca optimizar este valor de modo que se acerque a 1. Para ello se van a realizar varias operaciones sobre el texto para estandarizar de la mejor manera posible los inputs del algortimo de clusterización y que dichos inputs sean lo mas descriptivos o representativos posible de la información que llevan los textos del análisis, es decir los 22 textos de las noticias. También se estudiará el orden de su aplicaión y se evaluará qué orden es el que mejora el resultado. Las operaciones que se proponen son las estudiadas en la asignatura :

- lematización
- stemming
- eliminación de stopwords
- eliminación de signos de puntuación
- traducción
- detección de idioma
- entidades nombradas (NE)

Además, en función de si se realiza o no traducción de los textos, las pruebas se han dividido en los siguientes ficheros :

BasicNewsClustering_original.py : fichero original de la práctica sin cambio ninguno.

BasicNewsClustering_0.py : fichero correspondiente a la sección 2.1.5 Procesado sin diferenciar el idioma y con el *gold standard* original.

BasicNewsClustering_1.py : fichero correspondiente a la secciónes 2.1.1-2.1.4 Procesado

sin diferenciar el idioma pero con *gold standard* corregido.

BasicNewsClustering_2_0.py : fichero de traducción y generación de textos en formato .txt traducido a inglés.

BasicNewsClustering_2_1.py : fichero de pruebas con los textos traducidos correspondiente a la sección 2.3 Procesado de textos traducidos a inglés.

Para no crear confsión, la correspondencia entre códigos y pruebas se presenta en la siguiente tabla :

Fichero	Sección memoria	Descripción
BasicNewsClustering_original.py	N/A	Código original
BasicNewsClustering_1.py	2.1.1-2.14	Pruebas sobre textos sin traducir con *gold standard* corregido
BasicNewsClustering_0.py	2.1.5	Pruebas con *gold standard* original
BasicNewsClustering_2_0.py	2.2	Traducción a inglés
BasicNewsClustering_2_1.py	2.3	Pruebas sobre textos traducidos

La ejecución de pruebas que se encuentran en el *main* se hace comentando y descomentando los trozos de código correspondientes a cada paso. El código va contenido entre comentarios tipo :

INICIO PRUEBA 1

...

FIN PRUEBA 1

2.1 *Procesado sin diferenciar el idioma con gold standard corregido. Fichero BasicNewsClustering_1.py*

Al principio se ha seguido una sencuencia de procesado que no tiene en cuenta el idioma y utilizando NLTK, la biblioteca de procesamiento del lenguaje natural pasada en python. Se trabaja sobre el fichero *BasicNewsClustering_1.py*.

Como hemos introducido en la sección anterior, se ha usado principalmente el gold

standar corregido. Sin embargo en la sección 2.1.5 repetimos todas las pruebas de esta sección usando el gold estandar original para demostrar que esta corrección era necesaria, dado que el clustering no se haría correctamente nisiquiera obtenindo un texto correctamente procesado. A continuación se analizan diferencias entre el gold estándar corregido y el original.

Gold standard se asocia a la variable **reference**. Inicialmente tenemos el siguiente :

```
reference =[0, 1, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 4, 5, 2]
```

Sin embargo, no todos los valores de arriba describen bien la temática de las noticias del conjunto de la práctica. Hemos analizado los 22 textos de noticias, en inglés y español, y hemso conluido lo siguiente si bien es cierto que se pueden proponer otras soluciones, la solución óptima propuesta finalmente por el profesor es la siguiente y es la que se ha utilizado:

```
reference =[0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1]
```

Ya esta elección yo la entiendo de la siguiente forma :

0 – noticias sobre la esquiadora Lindsey Vonn

1 – donde se menciona la ciudad tejeña El Paso, pero no se menciona nada más. De hecho las dos noticias no son necesariamente sobre el mismo tema de El Paso.

2 – sobre John Cantlie

3 – Gobierno de España

4 – Trump y Texas

5 – Trump pero sin Texas

Abajo se muestra el listado de las noticias para referencia:

```
(base) marina@mashina:~/PycharmProjects/Recuperacion_Mineria_Textos/practica 1/CorpusHTMLNoticiasPractica1819$ ls
'Adios Lindsey Vonn _ ElMundo.html'
'Al menos cinco muertos por un tiroteo en una vivienda de Texas.html'
'British Journalist to Be Still Alive.html'
'Esta vivo John Cantlie_.html'
'Gobierno britanico John Cantlie vivo.html'
'Gobierno presiona salvar Presupuestos _  EL PAIS.html'
'John Cantlie believed to be alive - CNN.html'
'John Cantlie still alive.html'
'John Cantlie vivo.html'
'La era Trump_ Trump desafía al Congreso y promete en Texas _terminar_ el muro con México - RTVE.es.html'
'Lindsey Vonn despedida_LaVanguardia.html'
'Lindsey Vonn icon - BBC Sport.html'
'Pedro Sanchez amenaza a los independentistas _ ElMundo.html'
'Pedro Sanchez baraja convocar las elecciones generales_LaVanguardia.html'
'Primer mitin de campaña de Trump sera en El Paso, Texas _ CNN.html'
'Reino Unido Cantlie vivo.html'
'Sanchez elecciones generales el 14 de abril_LaRazon.html'
'Skiing icon Lindsey Vonn_Globalnews.ca.html'
'Texans oppose Trump border wall in Texas _ Fort Worth Star-Telegram.html'
'Trump blasts Beto O'Rourke, stands by wall at Texas rally.html'
'Trump offers socialism for the rich, capitalism for everyone else _ Robert Reich _ Opinion _ The Guardian.html'
'War on Drugs is to Blame for Government Corruption in El Paso, Texas.html'
```

2.1.1 Lematización

La primera prueba de preprocesado que hice ha sido la obtención de lemas. Para ello primero es necesario tokenizar, es decir trocear el texto en tokens que serán analizables por la herramienta que escojamos, sea nltk, Spacy o cualquier otra. A continuación se hace el pos_tagging que permite a cada token asignar un tag o descripción morfológica de la palbra en cuestión. Se define una función intermedia para quedarnos solamente con las palabras más representativas como pueden ser Adjetivos, Verbos, Sustantivos y Adverbios. Como último paso se usa la función lemmatize de la clase WordNetLemmatizer para quedarnos con lemas de las palabras seleccionadas.

Al pasar texto lematizado al clasificador de la práctica he obtenido un resultado elevado tan solo unas centésimas respecto al resultado del punto de partida de la práctica :

```
Vectors created.
('test: ', array([1, 1, 2, 1, 1, 1, 3, 2, 1, 1, 1, 2, 1, 1, 4, 1, 1, 2, 0, 0, 2, 5]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.05904059040590405)
```

Cabe mencionar que desde el inicio al imprimir por pantalla el contenido de las listas resultado de funciones de procesamiento de nltk me he encontrado con que las tildes las imprime como el ejemplo marcado en rojo a continuación.

('Lemmas: ', ['utilizamos', u'cooky', 'propias', 'servicios', 'y', 'mostrarle', 'informaci\xc3\x83\xc2\xb3n'

Sin embargo, esto no ha sido un problema al pasar este contenido a las funciones de traducción ni detección de idioma, por tanto este comportamiento se ha ignorado.

2.1.2 Puntuación

Lo siguiente que se ha hecho ha sido quitar puntuación. Podemos ver que quitar la puntuación en sí no produce ninguna mejora en el resultado.

```
Vectors created.
('test: ', array([1, 1, 2, 1, 1, 1, 3, 2, 1, 1, 1, 2, 1, 1, 4, 1, 1, 2, 0, 0, 2, 5]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.05904059040590405)
```

Es comprensible dado que si bien la puntuación puede traer información útil para el pos_tagger, para el clasificador resulta de poca utilidad. De hecho, cabe mencionar que este paso se hace siempre después de hace el pos_tag, sino el tagging no se hace correctamente.

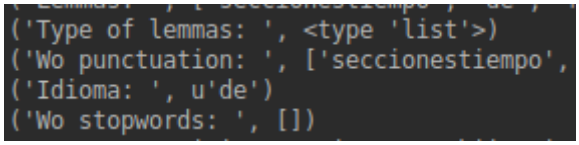
2.1.3 Stopwords

Se han probado dos versiones de esta operación : una sin detectar idioma (sacando stopwords siempre en inglés, con lo cual los textos en español no se les sacan las stopwords), y la otra detectando idioma y eliminando stopwords para cada idioma.

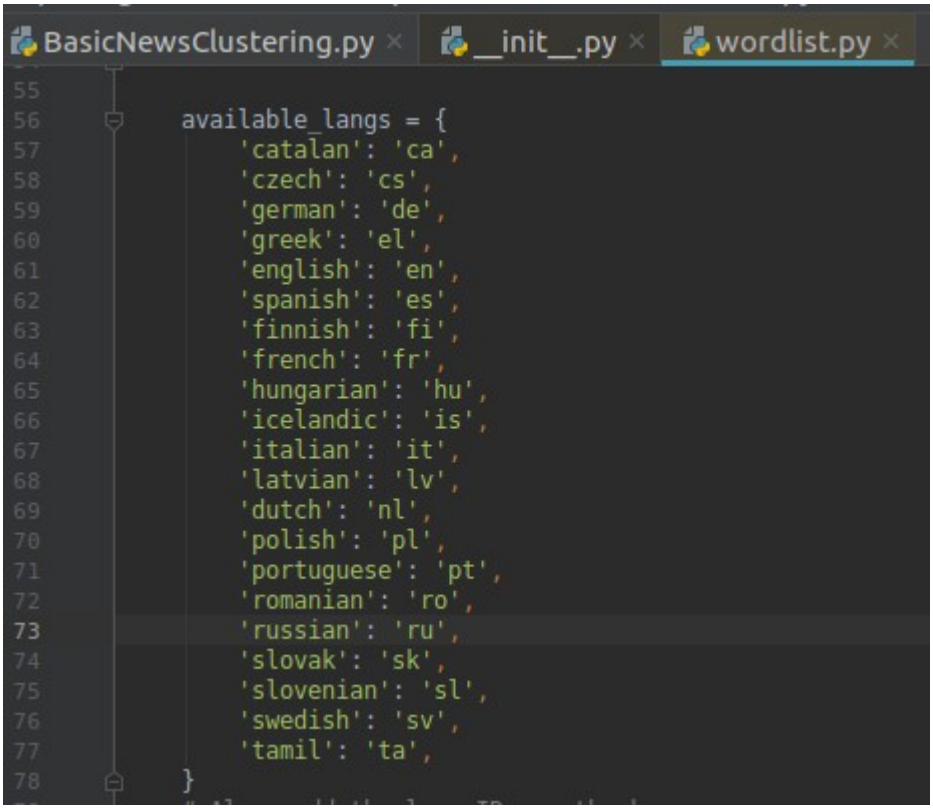
El resultado de la primera opción:

```
Vectors created.
('test: ', array([1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 3, 1, 1, 1, 1, 1, 3, 4, 4, 5, 2]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.09819501217103753)
```

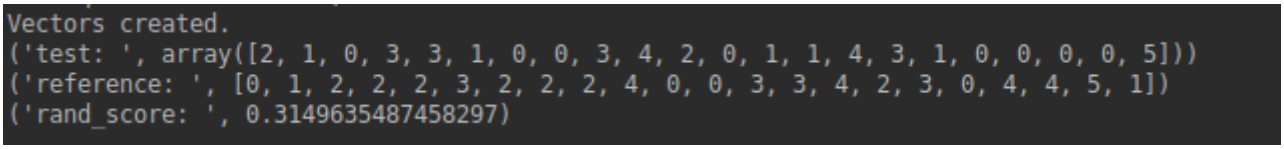
Al inicio la segunda opción no funcionaba dado que me producía una división por cero al no crearse correctamente los tokens en el *if-else* de la detección de idiomas. Esto pasaba porque se detectaba como alemán el texto « *Sanchez elecciones generales el 14 de abril_LaRazon.html* ».



He comprobado que el texto en realidad está en español y he definido un caso por defecto (*else*) suponiendo que todo lo que no se detecte como inglés o español será español. Se puede ampliar el scope de dicho *if-else* en un futuro si hay más textos en otros idiomas. A continuación se muestra el listado de idiomas disponibles en el paquete de stopwords de nltk :



El resultado de la segunda opción :



Podemos ver que el resultado mejora substancialmente cuando detectamos y

eliminamos las stopwords bien.

2.1.4 Named Entities

Lo siguiente que he hecho antes de pasar a la traducción ha sido quedarme con las entidades nombradas o nombres propios, *NEs*. Para el procesamiento de NEs del texto se ha usado el chunker de nltk en su versión que admite token de frases como parámetro de entrada. Por tanto, la tokenización, el proceso que se hace antes del chunking, se centra en tokenizar el texto en frases.

```
def ne_chunk(tagged_tokens, binary=False):
    """
    Use NLTK's currently recommended named entity chunker to
    chunk the given list of tagged tokens.
```

```
def ne_chunk_sents(tagged_sentences, binary=False):
    """
```

Para obtener dichas frases se ha usado el tagger por defecto de nltk, cuya estructura se puede consultar mediante la siguientes orden :

`nltk.help.upenn_tagset()`

```
    next into if beside ...
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separabl
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary ...
JJR: adjective, comparative
    bleaker braver breezier briefer brighter brisker broader bumper busier
    calmer cheaper choosier cleaner clearer closer colder commoner costlie
    cozier creamier crunchier cuter ...
JJS: adjective, superlative
    calmest cheapest choicest classiest cleanest clearest closest commones
    corniest costliest crassest creepiest crudest cutest darkest deadliest
    dearest deepest densest dinkiest ...
LS: list item marker
    A A. B B. C C. D E F First G H I J K One SP-44001 SP-44002 SP-44005
    SP-44007 Second Third Three Two * a b c d first five four one six thre
    two
MD: modal auxiliary
    can cannot could couldn't dare may might must need ought shall should
    shouldn't will would
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
```

Existe otro tagger *universal* , cuya estructura se presenta a continuación, pero me ha resultado más cómodo el de por defecto.

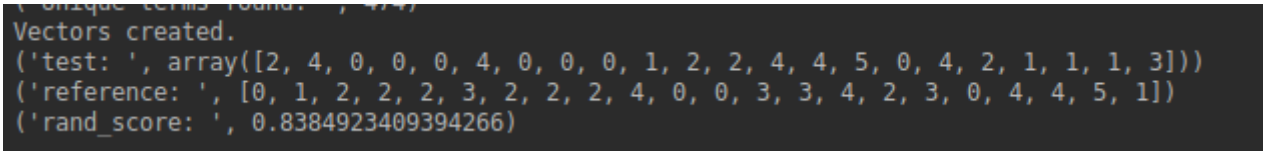
Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

El resultado de esta prueba es el siguiente :

```
Vectors created.
('test: ', array([2, 4, 0, 0, 0, 4, 0, 0, 0, 1, 2, 2, 4, 4, 5, 0, 4, 2, 1, 1, 1, 3]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.8384923409394266)
```

Podemos ver que al usar las entidades nombradas el resultado ha aumentado más de 0.5 puntos por lo que podemos deducir que las NE llevan el máximo de información relevante para nuestro clasificador. En la siguiente sección haremos el preprocesado necesario sobre los textos para traducirlos y haremos las mismas pruebas sobre sobre los archivos traducidos.

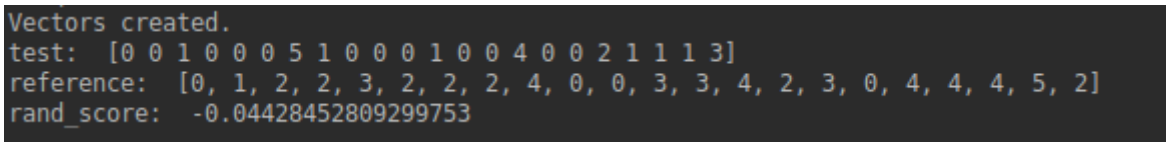
Contrario a lo que pensaba, al comentar todos los pasos anteriores al presente paso y solamente ejecutando el paso de extracción de entidades nombradas obtenemos el mismo ARI que el obtenido tras ejecutar todas las pruebas de 1 a 4 incluyendolas en cada uno de los pasos posteriores. Es decir, parece dar igual haber quitado las stopwords o puntuación antes de extraer las NE que no haberlo hecho.



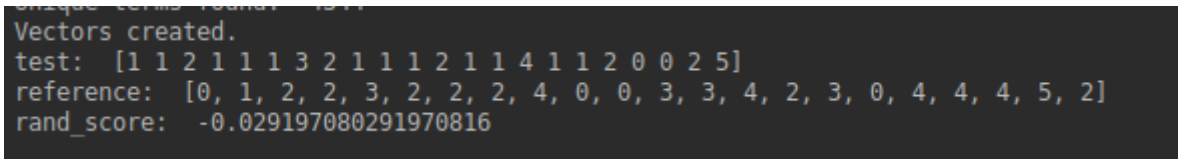
2.1.5 Procesado sin diferenciar el idioma con gold standard original. Fichero *BasicNewsClustering_0'.py*

Ahora repetimos las pruebas anteriores con el *gold standard* original para demostrar que no se consigue clasificar bien y se obtiene un ARI más bajo.

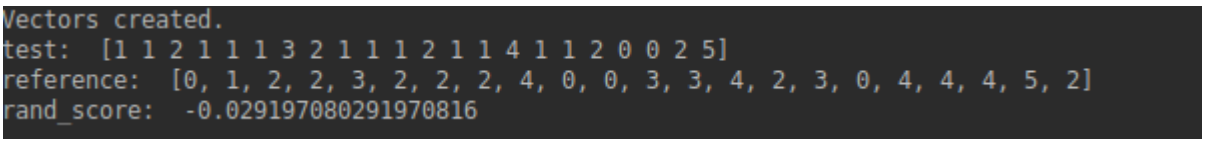
Sin pruebas :



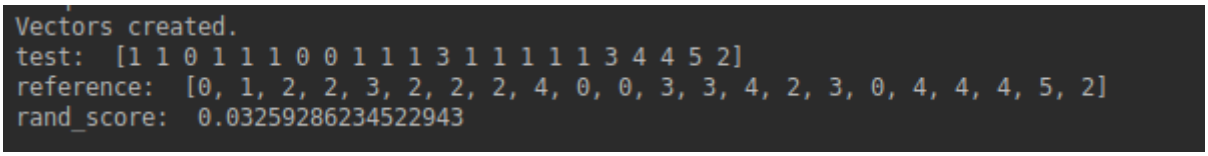
Lematizando :



Ahora añadimos la eliminación de la puntuación:



Ahora además quitar las stopwords pero sin diferenciar entre idioma :



Por último, procesamos las entidades nombradas :

```
Vectors created.
test:  [2 4 0 0 0 4 0 0 0 1 2 2 4 4 5 0 4 2 1 1 1 3]
reference:  [0, 1, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 4, 5, 2]
rand_score:  0.0284512617516081
```

Podemos ver que el resultado final obtenido con el *gold standard* original es muy bajo en comparación con el del *gold standard* (GE) corregido :

ARI final usando GE original	ARI final usando GE corregido
0.028	0.83

2.2 Procesado de textos traducidos a inglés. Fichero *BasicNewsClustering_2_{0,1}.py*

2.2.1 Traducción de textos a inglés. Fichero *BasicNewsClustering_2_0.py*

Para obtener las noticias traducidas y en formato texto y no hacer consultas innecesarias a google a través de textblob, recorremos el directorio y leemos cada fichero usando la codificación correspondiente : latin-1 en caso de español y utf-8 en caso de ingleś. La diferenciación se hace porque con utf-8 no se reconocen bien los titulos ni el contenido de los textos. Los ficheros en inglés se guardan directamente sin traducir y los que están en español se traducen. La decisión de si un fichero está en ingleés o en otro idioma se hace mediante Tika. Se usa una clase TikaReader definida en uno de los ejercicios de la asignatura para este fin. Por último, se guardan los resultados manualmente en una carpeta llamada `CorpusTXTNoticiasPractica1819Traducidos` que se entrega junto con la práctica y las pruebas de la presente sección 2.2 se hacen directamente sobre los textos traducidos.

2.2.2 Repetición de pruebas sobre los textos traducidos : todos los textos están en inglés. Fichero *BasicNewsClustering_2_1.py*

2.2.2.1 Prueba 1. Lematización.

Vemos que la ejecución del código es mucho más rápida al quitar la componente de procesado de ficheros en formato html. El resultado de la primera prueba es en comparación con el escenario sobre los textos sin traducir da un ARI de 0.27 frente a 0.05 :

```
Vectors created.
('test: ', array([5, 0, 0, 0, 0, 0, 1, 0, 0, 0, 5, 0, 3, 3, 1, 0, 3, 5, 2, 2, 0, 4]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.2732362821948488)
```

2.2.2.2 Prueba 2. Quitar puntuación

Descomentamos la parte correspondiente a la prueba 2 y ejecutamos el código. Podemos ver que al quitar puntuación el resultado no cambia al igual que antes sobre textos sin traducir.

```
Vectors created.
('test: ', array([5, 0, 0, 0, 0, 0, 1, 0, 0, 0, 5, 0, 3, 3, 1, 0, 3, 5, 2, 2, 0, 4]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.2732362821948488)
```

2.2.2.3 Prueba 3. Quitar stopwords

Ejecutamos la prueba en su versión principal dado que ya no es necesario detectar idioma en este paso, pues todos los textos están en ingles. Eliminamos las stopwords de inglés y mejoramos mucho el anterior resultado.

```
Vectors created.
('test: ', array([4, 0, 1, 1, 1, 0, 5, 1, 1, 2, 4, 4, 0, 0, 5, 1, 0, 4, 2, 2, 2, 3]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.7223928428361597)
```

2.2.2.4 Prueba 4. Named entities

Por último, ejecutamos la prueba de las NE. El resultado final es más alto que el obtenido sobre los textos sin traducir. Esto tiene todo el sentido porque en primer lugar el lematizador que usamos está adaptado solo para inglés. En segundo lugar supongo que esto ocurre porque el algoritmo de clasificación trabaja sobre el mismo idioma y trabaja con datos equiparables.

```
Vectors created.
('test: ', array([1, 5, 4, 4, 4, 2, 4, 4, 4, 0, 1, 1, 2, 2, 0, 4, 2, 1, 0, 0, 0, 3]))
('reference: ', [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1])
('rand_score: ', 0.9265874276997393)
```

2.2.3 TF vs TF_IDF

Inicialmente el algoritmo que usamos para hacer clustering the los textos es *tf* , o *term frequency. TF* es el número de veces que un término ocurre en el documento que usaremos como medida de decisión de si un documento pertenece a uno u a otro cluster. Es interesante también probar el otro método mencionado en clase que es *tf-idf* que es la combinación de *term frequency* e *inverse document frequency*, o *idf,* que es la medida de cuánta información proporciona el término, si ese término es común o frecuente en los documentos. Si una palabra es común, entonces da menos información, por tanto la medida *tf-idf* combinada se maximiza con un *tf* alto y un *idf* bajo.

Para probar hemos remplazado tf por tf-idf, función que existe en el módulo de Texts de la nltk :

```
def TF(document, unique_terms, collection):
    word_tf = []
    for word in unique_terms:
        word_tf.append(collection.tf_idf(word, document))
    return word_tf
```

El ARI que obtenemos de esta forma es más bajo :

```
Vectors created.
test:  [4 0 5 5 5 0 2 5 5 1 4 4 0 0 2 5 0 4 1 1 1 3]
reference:  [0, 1, 2, 2, 2, 3, 2, 2, 2, 4, 0, 0, 3, 3, 4, 2, 3, 0, 4, 4, 5, 1]
rand_score:  0.7223928428361597
```

A continuación vemos la comparación entre los dos ARI :

ARI final usando TF	ARI final usando TF_IDF
0.92	0.72

¿Por qué nos sale este resultado ? Porque seguramente la clasificación óptima que hicimos para obtener el *golden standard* se parece más a la medida *tf* que a la *tf-idf*.

2.3 Resultados y conclusiones

Se puede ver que el resultado final no es un uno. Para aproximarse más a ese valor podíamos probar descartando aquellas palabras que hemos obtenido tras el última parte de pruebas (NEs) pero no son palabras existentes sino vienen bien de pasar de html a texto, o bien por otros motivos. Este último paso es el que no he realizado por la falta de tiempo.

También me hubiera gustado hacer dos versiones de por código, una con nltk (la presente) y otra con Spacy u otra biblioteca de procesamiento de texto.

Otra prueba sería cambiar la secuencia de los pasos realizados y mostrar que es importante hacer el pos_tagging antes de quitar la puntuación

Por otra parte he podido observar que las entidades nombradas permiten recoger las caraterísticas más descriptivas y representativas del contenido de textos, cosa que puede tener muchísimas aplicaciones, desde procesamiento e inventariado de grandes cantidades de texto, como en el ejemplo de los *papeles de Panamá* como en análisis de sentimientos que tiene también una multitud de apicaciones pues permite completar cualquier análisis predicitivo hoy en día, dado que todo se refleja en redes sociales y la influencia de la opinión pública se hace cada vez más importante.

Por último, la práctica me ha ayudado a repasar todo lo visto en la asignatura sobre el procesamiento de texto, incluyendo las diferentes herramientas y me ha permitido tomar

decisiones sobre las herramientas a utilizar. En particular se ha utilizado la biblioteca *ntlk* para procesamiento básico (tokenización, pos-tagging y chunking), *Tika* para la detección de idioma y *textblob* para la traducción. No se ha usado *spacy* aunque se está quedando obsoleta y *spacy* viene a reemplazarla.