Big Data y Minería de Datos

Práctica 6. Procesamiento del lenguaje natural

Autores
Mariano Albaladejo González
José Antonio Ruipérez Valiente
Manuel Jesús Gómez Moratilla

Grado en Gestión de Información y Contenidos Digitales

Big Data y Minería de Datos

Práctica 6. Procesamiento del lenguaje natural

Autores: Mariano Albaladejo González, José Antonio Ruipérez Valiente y Manuel Jesús Gómez Moratilla

Esta práctica se centra en la utilización de procesamiento del lenguaje natural (NLP, por sus siglas en inglés Natural Language Processing) para la extracción de tópicos. El campo de NLP se ocupa del análisis de texto; por ello, puede aplicarse tanto en aprendizaje supervisado como en aprendizaje no supervisado. En esta práctica centrada en la extracción de tópicos nos enfrentaremos a un problema no supervisado; concretamente, queremos extraer los principales tópicos (temáticas) a partir de un conjunto de noticias (nuestro corpus). Estas noticias no están previamente clasificadas, lo que hace que sea un problema no supervisado para el que utilizaremos K-medias. A lo largo de esta práctica consideraremos cada noticia un documento de nuestro corpus.

Importamos e instalamos las librerías necesarias

Primero es necesario descargar e importar las librerías necesarias.

In [1]:
```
import re
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.cluster import KMeans

# nltk
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

#Lematizador
import spacy

from wordcloud import WordCloud

from google.colab import drive
```

Establecemos la semilla de aleatoriedad

In [2]:
```
SEMILLA_ALETORIEDAD = 33
```

Cargamos el conjunto de datos

Vamos a cargar el fichero `news_1000_24_25.csv`. Este fichero contiene 1000 noticias sin clasificar.

In [3]:
```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [4]:
```
df = pd.read_csv('/content/drive/MyDrive/df_MD/news_1000.csv')
df
```

Out[4]:

	News
0	The unemployment rate for transgender people i...
1	Lawyers for a transgender student and the G...
2	The man at the heart of the legal resistance t...
3	In the 23 years that Starflyer 59 has been a b...
4	It was the first day of school for Dan Lear's ...
...	...
995	Note: NPR's First Listen audio comes down afte...
996	A federal grand jury is said to have begun hea...
997	Nate Kramer was a tall, quiet college swimmer ...
998	A powerful drug that's normally used to tranqu...
999	The World Health Organization for the first ti...

1000 rows × 1 columns

Puedes observar que las noticias están en una columna llamada `News`. A continuación, mostramos una de las noticias de nuestro corpus.

In [5]:
```
df['News'][1]
```

Out[5]: 'Lawyers for a transgender student and the Gloucester, Va. school board that wa
nts to limit which bathroom he can use don’t agree on much. But both sides have co
ncluded the Trump administration’s decision this week to revoke guidance that prot
ects transgender students’ ability to use bathrooms and locker rooms that correspo
nd with their gender identity only heightens the need for a hearing before the nat
ion’s highest court. Rather than making the claims of Virginia student Gavin Grimm
moot, his attorneys at the American Civil Liberties Union and their courtroom oppo
nents are urging the U. S. Supreme Court to proceed with oral arguments scheduled
for March 28. ”If anything, the confusion caused by this recent action by the Depa
rtment of Justice and the Department of Education only underscores the need for th
e Supreme Court to bring some clarity here,” Joshua Block, a senior staff attorney
at the ACLU’s LGBT HIV Project, told reporters on a conference call on Thursday.
As for Gloucester County, the school board said it ”looks forward to explaining to
the Supreme Court why [the rescinded guidance] underscores that the Board’s common
sense restroom and locker room policy is legal under federal law.” The high court
agreed to hear two questions in the Grimm case: one, whether the former Justice an
d Education Departments’ interpretation of Title IX of a 1972 education law protec
ting students from discrimination on the basis of their sex deserved deference and
two, whether the prohibition on sex discrimination in schools also applies to gend
er identity. While the first question likely goes away along with the guidance,
the second remains, if justices still want to field it. The U. S. Justice Departme
nt, led by new Attorney General Jeff Sessions, said the matter is best left to ”Co
ngress, state legislatures, and local governments,” shrouding the federal argument
in a case for states’ rights. And at the White House on Thursday, press secretary
Sean Spicer pointed out that a federal judge in Texas last year enjoined the guida
nce so it never fully took effect. Spicer also asserted, ”There’s nobody who’s pos
sibly suggesting” that the 1972 law contemplated protection of transgender student
s at the time. That view is shared by a series of mostly conservative groups, whic
h have weighed in with Supreme Court briefs opposing an expansive reading of sex d
iscrimination. The Alliance Defending Freedom, one such nonprofit, defended the ap
proach by the Gloucester School Board that allowed transgender students to, among
other things, use separate, individual facilities to change clothes and shower. Th
e Alliance said school boards enjoy great freedom to develop education policies fo
r their own communities. Civil rights lawyers said that’s just wrong. The federal
government, they argue, has long led the way when it comes to integrating schools,
easing access to the polls, and other critical civil rights issues. What’s more, B
lock at the ACLU said, his client Gavin Grimm has obtained an amended birth certif
icate stating that he’s male. If Grimm moves to a state like North Carolina, where
the law defines sex by what’s on a birth certificate, schools there would allow hi
m to use the boy’s room. But Gloucester County, Va. has adopted a different approa
ch, excluding Grimm from the boy’s room and requiring him to use a restroom. ”Th
e fact is that no child in America should have their rights subject to their ZIP c
ode,” said Eliza Byard of the Gay Lesbian Straight Education Network, or GLSEN. A
dvocates for the LGBT community say that a courts across the nation already have
adopted their approach — finding that longstanding protections against sex disc
rimination under federal law extend to transgender people. The U. S. Courts of App
eals in the Sixth and Seventh Circuits have sided with transgender students and de
nied attempts to halt or stay the cases while they move toward appeal, ACLU lawyer
s said. ”Honestly, I’ll just tell you, nobody wants to be on the business end of a
transgender lawsuit these days,” said Mara Keisling, executive director of the Nat
ional Center for Transgender Equality. ”The courts have been moving in our directi
on very quickly.” Nearly two years ago, a divided Supreme Court ruled that co
uples have a fundamental right to marry under the due process clause and the equal
protection clause. President Trump told an interviewer this year that is ”settled
law.” But legal arguments about whether protections against sex discrimination in
Title IX and other federal laws extend to gender identity, central to the lives of
transgender people, are still very much active in the courts. For instance, Lambda
Legal, a nonprofit that advocates and litigates for LGBT clients, is litigating a
case against a Pennsylvania school district where Juliet Evancho, a transgender st
udent, wants to use the bathroom of her choice. ”It’s not just about bathrooms, i
t’s about being a full, participating member of society,” said Demoya Gordon, a Tr
ansgender Rights Project attorney at Lambda Legal. Lambda Legal is also awaiting c
ourt decisions in cases that center on whether federal laws cover a person’s sex
ual orientation. In one case, a math teacher, who is a lesbian, separated from her

```
job in South Bend, Ind. because of her sexual orientation. Lambda is arguing that
sexual orientation discrimination against its client, Kimberly Hively, ”is a form
of sex discrimination” barred under Title VII of the 1964 Civil Rights Act. The or
ganization is making the same argument before the 11th Circuit Court of Appeals. T
here, a security guard sued her employer for harassment and for allegedly forcing
her out of a job at Georgia Regional Hospital because she’s lesbian and ” .”'
```

Preprocesamiento del texto

A lo largo de esta sección vamos a preparar las noticias para nuestros análisis.

Filtrado del texto mediante expresiones regulares

Primero comenzamos eliminando saltos de página, comillas simples, comillas dobles y los números que aparecen en las noticias mediante expresiones regulares *Regex*.

En el código de la siguiente celda de código, la función `re.sub(pattern, repl, text_org)` del módulo de las expresiones regulares (re) se utiliza para buscar un patrón (`pattern`) en una cadena de texto (`text_org`) y reemplazarlo con otra cadena (`repl`).

In [6]:
```
re.sub(r'\d+:\d+', 'sustituido', 'La clase será a las 99:15 en el aula CODORNIZ')
```

Out[6]:
```
'La clase será a las sustituido en el aula CODORNIZ'
```

In [7]:
```
re.sub(r'\"', '', 'Texto con muchas "comillas "" " "" "')
```

Out[7]:
```
'Texto con muchas comillas    '
```

A continuación, aplicaremos expresiones regulares para limpiar nuestras noticias. En el código utilizaremos una comprensión de listas donde se introduce noticia a noticia a la función `re.sub()` , y los resultados se vuelven a almacenar en la columna `News` . Para las expresiones regulares utilizaremos cadenas en crudo de Python `r"mi cadena"` . Estas cadenas de caracteres destacan porque las barras invertidas () se tratan literalmente, en lugar de ser interpretadas como caracteres de escape.

In [8]:
```
# Eliminamos los saltos de línea
df["News"] = [re.sub(r'\n', '', item) for item in df["News"]]

# Eliminamos comillas simples
df["News"] = [re.sub(r"'", '', item) for item in df["News"]]

# Eliminamos comillas dobles
df["News"] = [re.sub(r'"', '', item) for item in df["News"]]

# Eliminamos números enteros
df["News"] = [re.sub(r'\d+', '', item) for item in df["News"]]

# Eliminamos URLs 1 y 2
df["News"] = [re.sub(r'http:\S+', '', item) for item in df["News"]]
df["News"] = [re.sub(r'www\.\S+', '', item) for item in df["News"]]
```

A continuación, describimos en mayor detalle las expresiones regulares anteriores.

- **Sustituir los saltos de línea por un espacio**. `\n` : Esta expresión regular reconoce saltos de línea (\n). La sustitución `' '` reemplaza los saltos de línea por una cadena vacía.
- **Eliminar comillas simples.** `'` : Esta expresión coincide con el carácter de comilla simple ('). La sustitución `''` elimina las comillas simples del texto reemplazándolas por una cadena vacía.
- **Eliminar comillas dobles.** `"` : Esta expresión coincide con el carácter de comilla doble (`"`). La sustitución `''` reemplaza las comillas dobles por una cadena vacía.
- **Eliminar números enteros**. `\d+` : Esta expresión coincide con una secuencia de uno o más dígitos (0-9). El `+` indica "uno o más" de lo que precede (en este caso, dígitos). Al reemplazar esta coincidencia con una cadena vacía, se eliminan todos los números del texto.
- **Eliminar URLs**. Coincide con cadenas que comienzan con *http:* (primera regla) o *www.* (segunda regla) seguidas de uno o más caracteres que no son espacios en blanco. Esto eliminaría la mayoría de las URLs que comienzan con *http:* o *www.*

Puedes experimentar con estas reglas a través de la página https://regex101.com/

Transformación a minúsculas de las palabras

En el siguiente paso transformamos todas las palabras a minúsculas.

In [9]:
```
df['News'] = df['News'].apply(lambda new: new.lower())
```

Lematización

La lematización consiste en reducir las palabras a su forma base (lema). Por ejemplo, la forma base de *dije* es *decir*. En nuestro caso utilizaremos `spacy` como lematizador.

Para más información: https://es.wikipedia.org/wiki/Lematizaci%C3%B3n

In [10]:
```
# Texto de prueba
text = "runs run running hello"

# Inicializamos en spacy el modelo en 'en' model
nlp = spacy.load('en_core_web_sm')

# Procesamos el texto con spacy
doc = nlp(text)

# Mostramos los lemas de cada token
for token in doc:
    print("Original: " + token.text + " | Lema: " + token.lemma_)
```

```
Original: runs | Lema: run
Original: run | Lema: run
Original: running | Lema: run
Original: hello | Lema: hello
```

La lematización incluye una fase de tokenización puesto que lematizamos palabra por palabra. En la siguiente celda de código aplicamos una lematización quedándonos con nombres, adjetivos, verbos y adverbios.

In [11]:
```
#  Categorías gramaticales que incluimos en el proceso de lematización
allowed_postags = ['NOUN', 'ADJ', 'VERB', 'ADV']

# Inicializamos en spacy el modelo en 'en' model
nlp = spacy.load('en_core_web_sm')

# Función que aplica una lematización a un texto
# Además, nos quedamos sólo con nombres, adjetivos, verbos y adverbios
def lemmatize_text(text):
    doc = nlp(text)
    return ' '.join([token.lemma_ for token in doc if token.pos_ in allowed_postags

df ['News'] = df['News'].apply(lemmatize_text)
```

Eliminación de palabras comunes (stop words)

Las stop words son palabras comunes en un idioma que generalmente se consideran de poco valor para el análisis de texto en el procesamiento del lenguaje natural. Estas incluyen palabras como pronombres personales o artículos.

Descargamos las palabras comunes ya definidas en la librería `nltk` y añadimos otras palabras comunes que deseamos eliminar de nuestro análisis (`additional_stopwords`).

In [12]:
```
# Descargamos la lista de stop words
nltk.download('punkt')
nltk.download('punkt_tab')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[12]:
```
True
```

In [13]:
```
additional_stopwords = ["ask", "say", "go", "buy", "include", "requiere", "make", '
```

In [14]:
```
stop_words = stopwords.words('english') + additional_stopwords
```

A continuación mostramos cinco stop words de las incluidas en la lista.

In [15]:
```
stop_words[:5]
```

Out[15]:
```
['a', 'about', 'above', 'after', 'again']
```

En el código siguiente eliminamos todas las stop words

In [16]:
```
def remove_stopwords(text):
    tokens = word_tokenize(text)
    filtered_tokens = [word for word in tokens if word not in stop_words]
    return ' '.join(filtered_tokens)


# Eliminamos las stops words
df['News'] = df['News'].apply(remove_stopwords)
```

Analizamos las palabras más frecuentes

A continuación, mediante `CountVectorizer` de *sklearn* vamos a calcular la frecuncia de cada una de las palabras en cada una de nuestras noticias.

In [17]:
```
df["News"]
```

Out[17]:

	News
0	unemployment rate transgender people double ge...
1	lawyer transgender student gloucester board li...
2	man heart legal resistance trump agenda work u...
3	year starflyer band constant decade many style...
4	first day school kid scramble get boy class ti...
...	...
995	note first listen audio come album release how...
996	federal grand jury begin hear evidence case un...
997	tall quiet college swimmer diagnose leukemia d...
998	powerful drug normally use tranquilize elephan...
999	health organization first time issue list top ...

1000 rows × 1 columns

dtype: object

In [18]:
```
# Creamos el contador
vectorizer = CountVectorizer()

# Entrenamos, aplicamos y almacenamos el recuento de palabras
words_per_new = vectorizer.fit_transform(df["News"])
words_per_new = pd.DataFrame(words_per_new.toarray(), columns=vectorizer.get_featur
```

En el dataframe resultante tenemos una columna por palabra y una noticia por fila. En la siguiente celda de código mostramos el número de apariciones de las palabras `trump`, `election` y `murder` en las noticias.

In [19]:
```
words_per_new[["trump", "election", "murder"]]
```

Out[19]:

	trump	election	murder
0	0	0	0
1	1	0	0
2	6	1	0
3	0	0	0
4	0	0	0
...	...	...	...
995	0	0	0
996	0	0	0
997	0	0	0
998	0	0	1
999	0	0	0

1000 rows × 3 columns

Una vez calculada la frecuencia de cada palabra en cada noticia podemos sumarlas y calcular la frecuencia de cada palabra en todo el conjunto de noticias (corpus).

In [20]:
```
word_counts = words_per_new.sum()
```

En el dataframe resultante tenemos una lista siendo el índice cada palabra y almacenando el total de apariciones. A continuación, mostramos la frencuencia de las palabras `trump` y `murder` en todo nuestro conjunto de noticias (corpus).

In [21]:
```
word_counts[["trump", "murder"]]
```

Out[21]:

	0
trump	1561
murder	71

dtype: int64

Si ordenamos las palabras por su frecuencia de aparición, podemos observar las palabras más frecuentes del conjunto de noticias (corpus).

In [22]:
```
word_counts = word_counts.sort_values(ascending=False)
word_counts[:10]
```

Out[22]:

	0
people	2431
year	1853
get	1660
trump	1561
time	1467
also	1357
take	1346
know	1338
think	1244
come	1196

dtype: int64

También podemos mostrar las palabras más frecuentes de todo nuestro conjunto de noticias (corpus) en una nube de palabras. En una nube de palabras el tamaño de cada palabra representa su frecuencia.

In [23]:
```
# Generamos la nube de palabras
wordcloud = WordCloud(width=800, height=400, background_color='white').generate_fro

# Mostramos la nube de palabras
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Análisis de tópicos con K-medias

A continuación vamos a intentar identificar tópicos sobre las noticias utilizando K-medias.

Primero vamos a calcular el *Term Frequency-Inverse Document Frequency* (TF-IDF) que será lo que introduciremos a K-medias para encontrar los clústeres. TF-IDF es una métrica que indica la relevancia de una palabra dentro de un documento específico, el cual es parte de un conjunto más amplio de documentos, conocido como corpus. El TF-IDF se compone de dos partes:

- **La frecuencia de los términos** (TF por sus siglas en inglés Term Frequency). Se calcula dividiendo el número de veces que la palabra aparece en el documento por el número total de palabras en ese documento. El razonamiento detrás de esto es que cuanto más frecuentemente aparece una palabra en un documento, más importante es para ese documento.

  TF(p, d) = (Número de veces que la palabra p aparece en el documento d) / (Número total de términos en el documento d)

- **La frecuencia inversa del documento** (IDF por sus siglas en inglés Inverse Document Frequency). Es una medida de la importancia de la palabra en todo el corpus. Se calcula como el logaritmo del número total de documentos en el corpus dividido por el número de documentos que contienen la palabra. Esto ayuda a ajustar el hecho de que algunas palabras aparecen con mucha frecuencia en general (por ejemplo, "el", "y", "de") y por lo tanto su presencia en un documento específico no es tan significativa.

  IDF(p, D) = log((Número total de documentos en el corpus D) / (Número de documentos donde aparece la palabra p))

La puntuación TF-IDF de una palabra en un documento se calcula multiplicando su TF y su IDF (TF * IDF). Esta puntuación representa la importancia de la palabra en el documento en relación con el corpus. En esta práctica, cada documento es una noticia y el corpus es el conjunto de noticias.

In [24]:
```
# Calculamos los TF-IDF
vectorizer = TfidfVectorizer(ngram_range=(1,1), max_features=30000, stop_words=stop
tf_idf = vectorizer.fit_transform(df["News"])
```

En la función anterior:

- `ngram_range=(1,1)` indica que pruebe ngramas entre 1 y 1, es decir, palabras de independientes (bag-of-words o bolsa de palabras).
- `max_features=30000` establece el número máximo de palabras a considerar.
- `stop_words=stop_words` establece las palabras comunes o stop_words a ignorar.

In [25]:
```
df_tfidf = pd.DataFrame(tf_idf.toarray(), columns=vectorizer.get_feature_names_out(
```

De nuevo, tenemos una fila por cada noticia (documento) y en cada columna un término del corpus. En cada casilla/celda tenemos el TF-IDF del término de la columna en la noticia de la fila.

In [26]:
```
df_tfidf[["trump", "election", "murder"]]
```

Out[26]:

	trump	election	murder
0	0.000000	0.000000	0.000000
1	0.018636	0.000000	0.000000
2	0.190433	0.037732	0.000000
3	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000
...	...	...	...
995	0.000000	0.000000	0.000000
996	0.000000	0.000000	0.000000
997	0.000000	0.000000	0.000000
998	0.000000	0.000000	0.035056
999	0.000000	0.000000	0.000000

1000 rows × 3 columns

A continuación vamos a aplicar K-medias con 6 clústeres y vamos a analizar los clústeres observados mostrando sus palabras más céntricas (centroides).

In [27]:

```
# K-medias
n_clusters = 6
model = KMeans(n_clusters=n_clusters, max_iter=1000, random_state=SEMILLA_ALETORIED
model.fit(tf_idf)

# Obtenemos del centroide de cada clúster,
# las posiciones (índices) de las palabras con los TF-IDF más altos
order_centroids = model.cluster_centers_.argsort()[:, ::-1]

# Obtenemos las palabras
terms = vectorizer.get_feature_names_out()

# Mostramos las 15 palabras más representativas de cada clúster
for i in range(n_clusters):
    print("Clúster", str(i)+ ":")
    for ind in order_centroids[i, :15]:
        print(terms[ind])
    print()
```

Clúster 0:
trump
campaign
president
voter
candidate
election
presidential
former
win
people
sander
state
tax
vote
political

Clúster 1:
insurance
health
student
state
care
plan
coverage
bill
school
law
insurer
education
pay
people
premium

Clúster 2:
song
music
people
know
woman
get
think
life
thing
time
really
play
year
story
write

Clúster 3:
patient
study
food
water
virus
disease
drug
brain
get
health
people
doctor

```
zika
year
eat

Clúster 4:
report
people
police
attack
state
year
government
country
city
law
company
also
kill
time
official

Clúster 5:
team
game
athlete
player
olympic
bile
gold
baseball
score
win
sport
medal
final
gymnast
academy
```

Con 6 clústeres, podemos identificar las temáticas de estos clústeres:

- Clúster 0: elecciones y campañas políticas
- Clúster 1: seguros médicos y educación
- Clúster 2: música
- Clúster 3: medicina
- Clúster 4: crímenes
- Clúster 5: deportes

Para definir el número de clústeres podemos probar con diferentes números de clústeres y aplicar la regla del codo con las distancias entre las distintas noticias y el centro de su clúster.

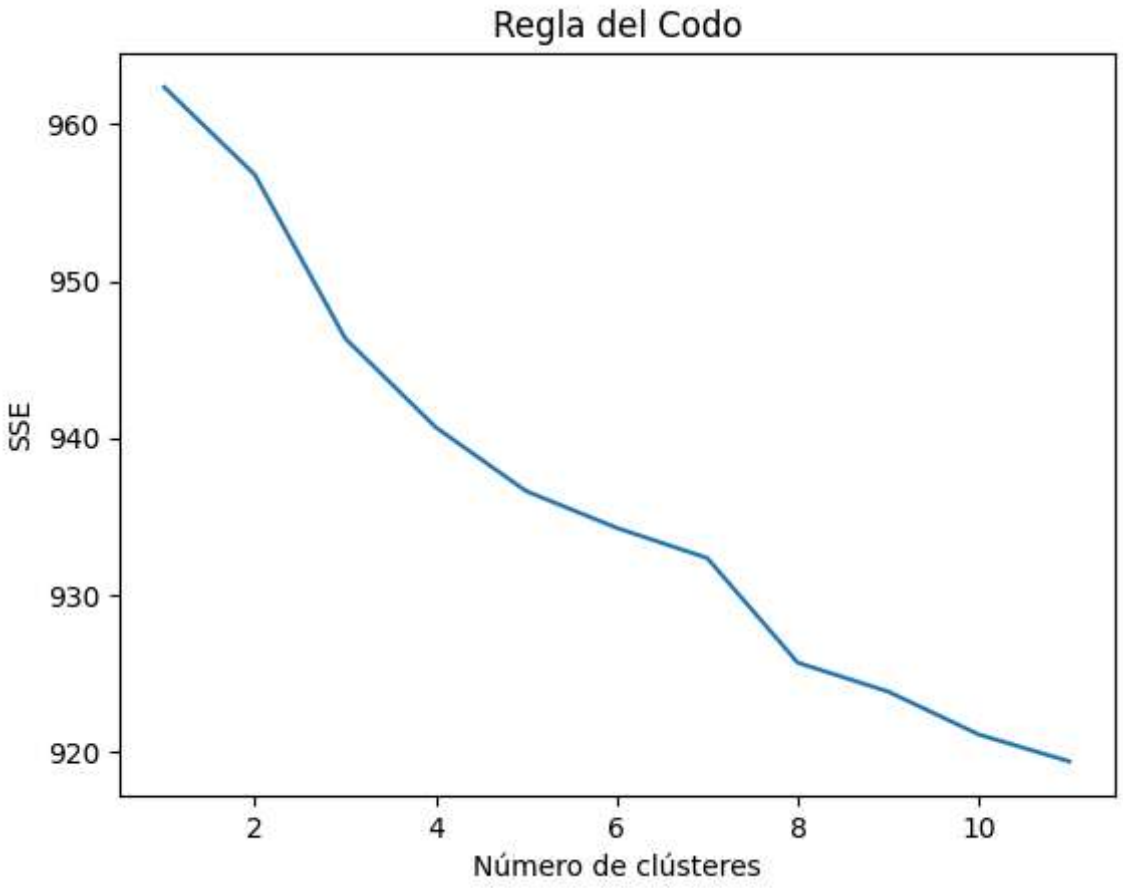
Mediante el atributo `kmeans.inertia_` es posible obtener la suma de las distancias al cuadrado entre cada muestra y el centroide del clúster más cercano. A continuación, mostramos la distancia entre las noticias y el centro de su clúster para distintos números de clústeres.

In [28]:
```
# Aplicamos K-medias con un rango de valores de k
sse = []
```

```
min_clusteres = 1
max_clusteres = 11

for k in range(min_clusteres, max_clusteres +1):
    kmeans = KMeans(n_clusters=k, max_iter=1000, random_state=SEMILLA_ALETORIEDAD)
    kmeans.fit(tf_idf)
    sse.append(kmeans.inertia_)

# Mostramos las distancias
plt.plot(range(min_clusteres, max_clusteres +1), sse)
plt.title('Regla del Codo')
plt.xlabel('Número de clústeres')
plt.ylabel('SSE')
plt.show()
```



En este caso, la regla del codo nos recomendaría elegir 8 clústeres.

In [29]:
```
# K-medias
n_clusters = 8
model = KMeans(n_clusters=n_clusters, max_iter=1000, random_state=SEMILLA_ALETORIED
model.fit(tf_idf)

# Obtenemos del centroide de cada clúster,
# las posiciones (índices) de las palabras con los TF-IDF más altos
order_centroids = model.cluster_centers_.argsort()[:, ::-1]

# Obtenemos las características
terms = vectorizer.get_feature_names_out()

# Mostramos las 15 palabras más representativas de cada clúster
for i in range(n_clusters):
    print("Clúster", str(i)+ ":")
    for ind in order_centroids[i, :15]:
        print(terms[ind])
    print()
```

Clúster 0:
trump
campaign
president
election
candidate
voter
presidential
people
former
tax
win
administration
nominee
state
political

Clúster 1:
patient
brain
study
doctor
cancer
hospital
drug
health
mouse
percent
people
researcher
cell
care
pain

Clúster 2:
song
music
band
album
play
sound
record
hear
sing
musician
concert
artist
know
audio
love

Clúster 3:
people
year
get
time
know
woman
think
work
take
life
see
student

school
family
tell

Clúster 4:
moon
solar
earth
energy
flight
planet
drone
year
power
climate
light
fly
ocean
star
new

Clúster 5:
police
report
attack
state
kill
officer
law
people
military
government
city
official
country
force
security

Clúster 6:
insurance
health
coverage
insurer
care
plan
premium
affordable
cost
people
obamacare
bill
state
exchange
law

Clúster 7:
virus
woman
zika
disease
health
outbreak
mosquito
vaccine

```
measle
people
spread
abortion
hospital
pregnant
birth
```

Con 8 clústeres podemos identificar las siguientes temáticas en los clústeres:

- Clúster 0: elecciones y campañas políticas
- Clúster 1: salud
- Clúster 2: música
- Clúster 3: educación y trabajo
- Clúster 4: astronomía
- Clúster 5: crímenes y violencia en USA
- Clúster 6: seguros médicos
- Clúster 7: salud

En la siguiente celda guardamos los clústeres asignados a cada noticia en el dataframe

In [30]:
```
cluster_labels = model.labels_
df['Cluster'] = cluster_labels
```

In [31]:
```
df.head(10)
```

Out[31]:

	News	Cluster
0	unemployment rate transgender people double ge...	3
1	lawyer transgender student gloucester board li...	3
2	man heart legal resistance trump agenda work u...	0
3	year starflyer band constant decade many style...	2
4	first day school kid scramble get boy class ti...	3
5	watch documentary yet officially show mainland...	3
6	early winter photographer drive north lake sup...	3
7	spend decade work mostly mexican guatemalan fa...	3
8	editor note frame short documentary film back ...	3
9	really know trump pay taxis major presidential...	0