# Proyecto Final: Parte 1 - Exploración del Corpus

- ***Materia:*** Análisis y Procesamiento Inteligente de Textos
- ***Maestro:*** Octavio Augusto Sánchez Velázquez
- ***Alumno:*** José Antonio Velázquez Sánchez

In [47]:
```
import os, pickle
import myutils
from collections import Counter
from gensim import corpora
```

# Objetivo

- El objetivo de este cuaderno es hacer una exploración rápida del corpus y de cómo he decidido representarlo para los siguentes cuadernos.

# Introducción

El objetivo de mi proyecto es explorar el modelado de tópicos para crear un clasificador de películas según su género utilizando subtítulos, y para encontrar relaciones entre el vocabulario de distintas lenguas.

Particularmente he utilizado un corpus recolectado por mí mismo, utilizando la API de opensubtitles.org, descargando los subtítulos en español, inglés y francés de 300 películas correspondientes a 3 géneros distintos: acción, romance/comedia y horror.

Tras descargarlos, les he hecho un preprocesamiento con freeling lematizando cada documento y dejando solamente las palabras. Los documentos lematizados se pueden encontrar en la carpeta *./clean_dataset* . Dentro hay 3 carpetas distintas (0, 1, y 2) correspondientes a los tres géneros con los que trabajaré (acción, romance/comedia y horror, respectivamente). Cada carpeta, entonces, tiene 300 archivos de texto, correspondientes a los subtítulos de 100 películas, en español, inglés y francés. Seleccioné estas 300 películas utilizando las listas que IMDb ofrece para cada genero, y tomando las 100 primeras (de cada género), ordenadas por el número de votantes que hubieran tenido.

El corpus original se encuentra en la carpeta *./download-scripts/* comprimido en un archivo .zip. En dicha carpeta se encuentran los scripts que utilicé para descargar el corpus original.

En mi preprocesamiento no incluí eliminación de palabras de paro, solamente lematización y eliminación de dígitos y otros caracteres no alfabéticos. Como describo en mi bitácora preliminar, originalmente sí filtraba palabras de paro pero eventualmente me di cuenta que para hacer experimentos con modelado de tópicos era mejor no filtrarnas sino hasta el momento de hacer el experimento y utilizar las palabras de paro como un parámetro más.

En los siguientes cuadernos de jupyter exploraré el modelado de tópicos en el corpus que he descrito. Sin embargo, en este cuaderno solamente tengo como objetivo dejar en claro las estructuras de datos que utilizaré para guardar la información.

# Cargando y Serializando el Corpus

In [3]:
```
def load_text(path):
    """
    Devuelve un string con todo el contenido de un archivo de texto dado.
    """
    text = ""
    with open(path, "r") as inputf:
        for line in inputf:
            text += line.lower()
    return text

def load_texts(path, lang):
    """
    Dado el path a la carpeta con los archivos de texto preprocesados,
    y el lenguaje de interés ('spa' para español, 'eng' para inglés y
    'fre' para francés, devuelve una lista de listas de strings,
    donde cada lista de strings corresponde a un género (0: acción,
    1: romance, 2: horror) y cada string es el contenido preprpocesado
    de los subtítulos de una película).

    Devuelve también una lista de listas de strings con los identificadores
    de las películas (según están dispuestos en IMDb)
    """
    movieids = []
    texts = []
    for label in ["0/", "1/", "2/"]:
        movieids_genero = []
        texts_genero = []
        directory = os.path.join(path, label)
        for txtfile in sorted(os.listdir(directory)):
            if txtfile.endswith("." + lang + ".txt"):
                movieids_genero.append(txtfile.split(".")[0])
                texts_genero.append(load_text(os.path.join(directory, txtfile))
)
        texts.append(texts_genero)
        movieids.append(movieids_genero)
    return movieids, texts

def cargar_titulos(path):
    """
    Dado un archivo que enliste los titulos de las películas,
    con su respectivo id (de IMDb), devuelve un diccionario
    que convierte de ID al título.
    """
    titulos = dict()
    with open(path, "r") as inputf:
        for line in inputf:
            titulo_id = line.split(" ")[0]
            titulo = line.split("  ")[1].rstrip()
            titulos[titulo_id] = titulo
    return titulos
```

In [4]:
```
movieids, txts_eng = load_texts("clean_dataset/", "eng")
movieids, txts_spa = load_texts("clean_dataset/", "spa")
movieids, txts_fre = load_texts("clean_dataset/", "fre") # los movieids son igu
ales para las 3 lenguas

titulos_accion = cargar_titulos("clean_dataset/0_accion.txt")
titulos_romance = cargar_titulos("clean_dataset/1_romance.txt")
titulos_horror = cargar_titulos("clean_dataset/2_horror.txt")
titulos = [titulos_accion, titulos_romance, titulos_horror]
```

In [5]:
```
pickle.dump(movieids, open("./pickles/movieids.pickle", "wb"))
pickle.dump(titulos, open("./pickles/titulos.pickle", "wb"))

pickle.dump(txts_spa, open("./pickles/txts_spa.pickle", "wb"))
pickle.dump(txts_eng, open("./pickles/txts_eng.pickle", "wb"))
pickle.dump(txts_fre, open("./pickles/txts_fre.pickle", "wb"))
```

# Ejemplo

In [6]:
```
genero = 2 # 0 para accion, 1 para romance y 2 para horror
peli = 10

print("Ejemplo: Subtítulos de '{0}'\n".format(titulos[genero][movieids[genero][
peli]]))
print("En Inglés: \n", txts_eng[genero][peli][:300], "\n")
print("En Español:\n", txts_spa[genero][peli][:300], "\n")
print("En Francés:\n", txts_fre[genero][peli][:300], "\n")
```

```
Ejemplo: Subtítulos de 'Gremlins (1984)'

En Inglés:
 friend let me introduce myself peltzer the name rand peltzer that be me there
on the corner be inventor and have story to tell know who have not get story we
ll nobody get story like this nobody it all start here in chinatown be try to m
ove little merchandise maybe find present for my kid try this pl

En Español:
 amigo dejar que me presente me llamar peltzer rand peltzer ser ese de el esqui
na ser inventor tener uno historia que contar el saber todo tener historia pero
nadie tener uno como este nadie todo empezar aquí en chinatown tratar de vender
alguno mercancía encontrar uno regalo para mi hijo probar en u

En Francés:
 mon ami permettre de me prêsenter rand peltzer ce être moi là le coin je suivr
e inventeur et je avoir un histoire raconter je savoir tout le monde avoir un h
istoire mais personne en avoir un comme celui ci personne tout avoir commencê i
ci chinatown je essayer de placer mon produit et de trouver un c
```

# Explorando los géneros

In [8]:
```
# 0 para accion, 1 para romance y 2 para horror
print("Ejemplos de peliculas de acción:\n", list(titulos[0].items())[:15], "\n"
)
print("Ejemplos de peliculas de romance:\n", list(titulos[1].items())[:15], "\n
")
print("Ejemplos de peliculas de horror:\n", list(titulos[2].items())[:15])
```

```
Ejemplos de peliculas de acción:
 [('0468569', 'Batman: El Caballero de la Noche (2008)'), ('1375666', 'El orige
n (2010)'), ('0133093', 'Matrix (1999)'), ('0167260', 'El señor de los anillos
- El retorno del rey (2003)'), ('1345836', 'Batman: El caballero de la noche as
ciende (2012)'), ('0172495', 'Gladiador (2000)'), ('0372784', 'Batman inicia (2
005)'), ('0848228', 'The Avengers: Los Vengadores (2012)'), ('0076759', 'La gue
rra de las galaxias (1977)'), ('0080684', 'El imperio contraataca (1980)'), ('0
499549', 'Avatar (2009)'), ('0325980', 'Piratas del Caribe - La maldición del P
erla Negra (2003)'), ('0434409', 'V de Venganza (2005)'), ('2015381', 'Guardian
es de la galaxia (2014)'), ('0103064', 'Terminator 2: Juicio Final (1991)')]

Ejemplos de peliculas de romance:
 [('0211915', 'Amelie (2001)'), ('1045658', 'Los juegos del destino (2012)'), (
'0118799', 'La vida es bella (1997)'), ('1637725', 'Ted (2012)'), ('0107048', '
Hechizo del tiempo (1993)'), ('1022603', '500 días con ella (2009)'), ('1570728
', 'Loco y estúpido amor (2011)'), ('3783958', 'La La Land: una historia de amo
r (I) (2016)'), ('0314331', 'Realmente amor (2003)'), ('0298148', 'Shrek 2 (200
4)'), ('0362227', 'La terminal (2004)'), ('0405422', 'The 40 Year Old Virgin (2
005)'), ('0398286', 'Enredados (2010)'), ('1605783', 'Medianoche en París (2011
)'), ('0446029', 'Scott Pilgrim vs. los ex de la chica de sus sueños (2010)')]

Ejemplos de peliculas de horror:
 [('0081505', 'El resplandor (1980)'), ('0078748', 'Alien, el octavo pasajero (
1979)'), ('0480249', 'Soy leyenda (2007)'), ('0816711', 'Guerra mundial Z (2013
)'), ('0054215', 'Psicosis (1960)'), ('0365748', 'El desesperar de los muertos
(2004)'), ('1156398', 'Tierra de zombies (2009)'), ('1457767', 'El conjuro (201
3)'), ('0387564', 'Saw: juego macabro (2004)'), ('5052448', '¡Huye! (I) (2017)'
), ('1060277', 'Cloverfield: Monstruo (2008)'), ('0289043', '28 Days Later... (
2002)'), ('1259521', 'La cabaña del terror (2012)'), ('0070047', 'El exorcista
(1973)'), ('1396484', 'Eso (I) (2017)')]
```

A la hora de escoger los géneros, escogí aquellos que parecían tener la menor intersección. Por ejemplo, con las películas con las que trabajo solamente se encuentran las siguientes intersecciones:

In [119]:
```
print("- Pelis de acción y romance -")
s = set(movieids[0]) & set(movieids[1])
for movieid in s:
    print(titulos[0][movieid])

print("\n- Pelis de acción y horror -")
s = set(movieids[0]) & set(movieids[2])
for movieid in s:
    print(titulos[0][movieid])

print("\n- Pelis de romance y horror -")
s = set(movieids[1]) & set(movieids[2])
for movieid in s:
    print(titulos[1][movieid])
```

```
- Pelis de acción y romance -

- Pelis de acción y horror -
Guerra mundial Z (2013)

- Pelis de romance y horror -
Mi novio es un zombie (2013)
```

# Creando y Serializando Diccionarios id2word

Para los próximos cuadernos será necesario tener un diccionario que asocie id's con palabras y viceversa. Para ello utilizo el Dictionary que gensim provée.

In [9]:
```
dictionary_spa = corpora.Dictionary([txt.split(" ") for txt in txts_spa[0] + tx
ts_spa[1] + txts_spa[2]])
dictionary_eng = corpora.Dictionary([txt.split(" ") for txt in txts_eng[0] + tx
ts_eng[1] + txts_eng[2]])
dictionary_fre = corpora.Dictionary([txt.split(" ") for txt in txts_fre[0] + tx
ts_fre[1] + txts_fre[2]])

dictionary_spa.save('pickles/dictionary_spa.dict')
dictionary_eng.save('pickles/dictionary_eng.dict')
dictionary_fre.save('pickles/dictionary_fre.dict')
```

In [10]:
```
print("# de tokens en Inglés:", len(dictionary_eng))
print("# de tokens en Español:", len(dictionary_spa))
print("# de tokens en Francés:", len(dictionary_fre))
```

```
# de tokens en Inglés: 32935
# de tokens en Español: 32710
# de tokens en Francés: 32044
```

La ventaja de usar esta clase de Gensim es que ya tiene implementada la manera de asociar id's a palabras y viceversa, además de convertir listas de palabras en vectores dispersos para ahorrar espacio.

In [10]:
```
print(dictionary_spa.token2id['el'])
print(dictionary_spa[445])
```

```
445
el
```

In [53]:
```
dictionary_spa.doc2bow(["el", "no"])
```

Out[53]:
```
[(445, 1), (870, 1)]
```

# Obteniendo Frecuencias del Corpus

Por cada género para cada lengua con la que trabajo obtengo un contador que asocia una palabra al número de veces que aparece en el corpus lematizado.

In [11]:
```
def obtener_frecuencias(txts):
    frecs = []
    for genero in txts:
        contador = Counter()
        for pelicula in genero:
            for word in pelicula.split(" "):
                contador[word] += 1
        frecs.append(contador)
    return frecs
```

In [12]:
```
frecs_spa = obtener_frecuencias(txts_spa)
frecs_eng = obtener_frecuencias(txts_eng)
frecs_fre = obtener_frecuencias(txts_fre)

pickle.dump(frecs_spa, open("./pickles/frecs_spa.pickle", "wb"))
pickle.dump(frecs_eng, open("./pickles/frecs_eng.pickle", "wb"))
pickle.dump(frecs_fre, open("./pickles/frecs_fre.pickle", "wb"))
```

# Palabras más comunes del Corpus

Palabras más comunes según el género

In [13]:
```
genero = 0
maxp = 10

print("{0} Palabras más frecuentes en Español:\n{1}\n".format(maxp, frecs_spa[g
enero].most_common(maxp)))
print("{0} Palabras más frecuentes en Inglés:\n{1}\n".format(maxp, frecs_eng[ge
nero].most_common(maxp)))
print("{0} Palabras más frecuentes en Francés:\n{1}\n".format(maxp, frecs_fre[g
enero].most_common(maxp)))
```

```
10 Palabras más frecuentes en Español:
[('el', 53676), ('de', 25243), ('que', 22083), ('ser', 21184), ('no', 20405), (
'uno', 16306), ('estar', 11579), ('en', 10408), ('haber', 8229), ('qué', 7103)]

10 Palabras más frecuentes en Inglés:
[('be', 51031), ('you', 34179), ('the', 27981), ('to', 19030), ('it', 15140), (
'not', 15120), ('do', 14096), ('that', 10786), ('have', 10370), ('of', 10369)]

10 Palabras más frecuentes en Francés:
[('le', 52881), ('de', 32719), ('être', 29210), ('je', 22542), ('avoir', 21538)
, ('ce', 16609), ('un', 15930), ('que', 15124), ('ne', 12609), ('pas', 12337)]
```

Como se puede observar, las palabras más comunes en las tres lenguas son artículos, preposiciones, pronombres, y verbos auxiliares.

Además de usar la función "most_common" que ya viene incluida en la clase Counter(), implementé mi propia función "intersect_most_common" que toma las palabras más comunes de cada género e intersecta dichos conjuntos. La ventaja de hacer la intersección es que solamente quedan las palabras que son mas comunes en los 3 géneros, y así evito enlistar palabras que sean comunes en solo 1 en género (y que por tanto sean representativas del mismo).

In [14]:
```
print(myutils.set2wordfrecs(myutils.intersect_most_common(frecs_spa), frecs_spa
))
```

```
[('el', 140421), ('de', 68722), ('ser', 64269), ('no', 63435), ('que', 62450),
('uno', 47769), ('estar', 35530), ('en', 29297), ('qué', 23571), ('haber', 2188
0), ('tener', 21429), ('me', 20681), ('por', 19809), ('ir', 19651), ('te', 1767
0), ('hacer', 17515), ('ese', 17064), ('poder', 16701), ('este', 16488), ('sí',
14156), ('se', 13783), ('decir', 13394), ('con', 13343), ('mi', 13219), ('saber
', 12805), ('querer', 12478), ('para', 11893), ('bien', 11563), ('todo', 10744)
, ('ver', 10609), ('su', 10484), ('pero', 10418), ('yo', 10164), ('si', 9686),
('tu', 9203), ('aquí', 8096), ('le', 7966), ('más', 7415), ('bueno', 7308), ('é
l', 7102), ('como', 6887), ('creer', 6773), ('ya', 6544), ('muy', 5793), ('tú',
5742), ('dar', 5622), ('deber', 5559), ('pasar', 5498), ('algo', 5052), ('ahora
', 4790), ('nos', 4783), ('cómo', 4699), ('así', 4566), ('dejar', 4447), ('nada
', 4254), ('mucho', 4207), ('vez', 4134), ('gracia', 4015), ('esperar', 3933),
('cuando', 3881), ('hablar', 3831), ('otro', 3720), ('sólo', 3530), ('dios', 35
21), ('necesitar', 3433), ('sentar', 3432), ('llamar', 3297), ('cosa', 3253), (
'venir', 3236), ('quién', 3148), ('oír', 3074), ('mirar', 3069), ('favor', 3061
), ('pensar', 3021), ('solo', 2982), ('llevar', 2907), ('usted', 2832), ('verda
d', 2780), ('mí', 2693), ('tiempo', 2665), ('vida', 2620)]
```

In [15]:
```
print(myutils.set2wordfrecs(myutils.intersect_most_common(frecs_fre), frecs_fre
))
```

```
[('le', 136645), ('de', 88441), ('être', 86971), ('je', 74507), ('avoir', 61358
), ('ce', 50981), ('un', 45723), ('que', 43430), ('pas', 37506), ('ne', 34484),
('tu', 30185), ('vous', 29588), ('il', 24876), ('et', 24461), ('aller', 22023),
('faire', 21258), ('on', 20963), ('en', 19643), ('cela', 18929), ('te', 18258),
('me', 17712), ('mon', 16019), ('pour', 13882), ('tout', 12775), ('se', 11909),
('pouvoir', 11831), ('dire', 11445), ('qui', 10471), ('mais', 10243), ('vouloir
', 10209), ('savoir', 9762), ('dans', 9451), ('non', 9287), ('elle', 9218), ('s
uivre', 9204), ('nous', 9081), ('si', 8704), ('bien', 8681), ('ton', 8156), ('m
oi', 8117), ('plus', 7944), ('voir', 7717), ('oui', 7471), ('devoir', 7451), ('
avec', 7220), ('son', 6986), ('quoi', 6702), ('votre', 6021), ('ils', 5982), ('
là', 5970), ('sur', 5561), ('toi', 5359), ('comme', 5276), ('venir', 5061), ('i
ci', 4968), ('bon', 4802), ('où', 4664), ('rien', 4377), ('lui', 4260), ('croir
e', 4017), ('pourquoi', 3857), ('aimer', 3792), ('prendre', 3681), ('parler', 3
658), ('chose', 3658), ('passer', 3550), ('notre', 3491), ('par', 3467), ('quan
d', 3463), ('merci', 3441), ('autre', 3440), ('comment', 3431), ('alors', 3335)
, ('très', 3212), ('trouver', 3177), ('faillir', 3161), ('ou', 3158), ('jamais'
, 3007), ('même', 2943), ('attendre', 2915), ('aussi', 2762), ('laisser', 2723)
, ('penser', 2712), ('deux', 2478)]
```

En contraste a intersect_most_common también implemente a "disjoin_frecs" que también toma las palabras más comunes de cada género, pero en lugar de hacer la intersección, toma solamente las palabras que solo aparecen en un frecuentemente en un género. De esta manera puedo saber qué palabras son más típicas de qué género.

In [22]:
```
disjoint_spa = myutils.disjoin_frecs(frecs_spa, min_frec = 100)
print("Palabras típicas de Acción con frec. mayor a 100:\n" , myutils.set2wordf
recs(disjoint_spa[0], frecs_spa), "\n")
print("Palabras típicas de Romance con frec. mayor a 100:\n" , myutils.set2word
frecs(disjoint_spa[1], frecs_spa), "\n")
print("Palabras típicas de Horror con frec. mayor a 100:\n" , myutils.set2wordf
recs(disjoint_spa[2], frecs_spa), "\n")
```

Palabras típicas de Acción con frec. mayor a 100:
 [('capitán', 784), ('guerra', 631), ('fuerza', 507), ('rey', 425), ('sistema', 416), ('ln', 396), ('señal', 393), ('control', 379), ('agente', 370), ('maestro', 362), ('atacar', 359), ('barco', 358), ('pelear', 350), ('mente', 343), ('controlar', 333), ('salvo', 328), ('general', 312), ('esconder', 308), ('merecer', 303), ('lejos', 301), ('suyo', 294), ('misión', 291), ('acompañar', 288), ('responder', 282), ('tony', 277), ('planeta', 275), ('enemigo', 274), ('código', 273), ('línea', 270), ('unir', 265), ('destino', 265), ('respuesta', 264), ('situación', 264), ('evitar', 262), ('abandonar', 262), ('ataque', 260), ('luchar', 259), ('permiso', 259), ('peligroso', 256), ('mitad', 256), ('recuperar', 255), ('sufrir', 255), ('avión', 253), ('modo', 252), ('stark', 250), ('ejército', 249), ('energía', 249), ('cumplir', 246), ('huir', 246), ('construir', 245), ('héroe', 244), ('edificio', 243), ('profesor', 243), ('jedi', 243), ('izquierda', 242), ('considerar', 242), ('oficial', 241), ('presidente', 241), ('posición', 241), ('imposible', 240), ('soldado', 238), ('cubrir', 236), ('decisión', 236), ('lanzar', 236), ('baja', 234), ('ofrecer', 232), ('necesario', 232), ('hallar', 231), ('vuestro', 230), ('oscuro', 230), ('dirigir', 230), ('objetivo', 229), ('información', 228), ('tras', 228), ('cerebro', 228), ('consejo', 227), ('campo', 227), ('entregar', 225), ('ambos', 224), ('base', 224), ('enfrentar', 221), ('esperanza', 221), ('salida', 218), ('vista', 217), ('liberar', 211), ('peligro', 210), ('cargar', 209), ('honor', 208), ('ley', 204), ('poderoso', 203), ('mutante', 201), ('nivel', 199), ('zona', 197), ('fallar', 195), ('unidad', 194), ('retirar', 190), ('avanzar', 187), ('jane', 186), ('líder', 184), ('máquina', 183), ('ante', 176), ('piloto', 176), ('wayne', 174), ('bala', 173), ('puente', 170), ('espada', 165), ('man', 163), ('fuente', 163), ('carga', 162), ('bomba', 162), ('escudo', 162), ('máximo', 162), ('defensa', 162), ('torre', 161), ('sargento', 158), ('robot', 155), ('batalla', 150), ('flota', 149), ('logan', 140), ('activar', 137), ('amenaza', 133), ('comandante', 133), ('luke', 132), ('pirata', 131), ('asgard', 120), ('senador', 115), ('thor', 112), ('loki', 105)]

Palabras típicas de Romance con frec. mayor a 100:
 [('casar', 766), ('lindo', 599), ('coger', 428), ('película', 423), ('bonito', 417), ('nueva', 407), ('escuela', 383), ('relación', 368), ('vera', 360), ('york', 360), ('maravilloso', 360), ('estúpido', 351), ('sexo', 347), ('vino', 346), ('cita', 345), ('comida', 344), ('mentir', 344), ('odio', 343), ('bailar', 342), ('llorar', 339), ('srta', 338), ('canción', 337), ('acostar', 335), ('cantar', 330), ('nene', 327), ('voz', 325), ('fantástico', 325), ('anoche', 322), ('pelo', 320), ('boca', 315), ('bromear', 312), ('baño', 311), ('sal', 308), ('besar', 306), ('invitar', 299), ('café', 295), ('interesar', 294), ('siguiente', 290), ('encima', 290), ('precioso', 290), ('gracioso', 289), ('boda', 286), ('broma', 285), ('mary', 283), ('simplemente', 280), ('enamorar', 277), ('oficina', 276), ('mentira', 276), ('personal', 276), ('saltar', 275), ('guardar', 275), ('ropa', 273), ('frente', 272), ('frío', 272), ('trasero', 272), ('probablemente', 271), ('música', 271), ('además', 268), ('acordar', 268), ('doler', 266), ('gato', 264), ('negocio', 264), ('conducir', 263), ('locura', 262), ('señorito', 262), ('ben', 261), ('piso', 260), ('tienda', 258), ('imbécil', 256), ('estupendo', 256), ('grupo', 256), ('devolver', 256), ('odiar', 255), ('cual', 254), ('totalmente', 254), ('las', 253), ('santo', 253), ('excelente', 253), ('pena', 251), ('dama', 251), ('crecer', 248), ('placer', 248), ('lleno', 248), ('dulce', 247), ('abuelo', 246), ('interesante', 245), ('regalo', 243), ('caja', 243), ('soñar', 242), ('mas', 241), ('horrible', 241), ('gay', 240), ('cumpleaños', 240), ('hotel', 239), ('usd', 239), ('cenar', 239), ('vaya', 238), ('sorpresa', 238), ('asunto', 236), ('pobre', 236), ('navidad', 236), ('actuar', 236), ('rico', 234), ('príncipe', 232), ('cena', 232), ('disfrutar', 232), ('amable', 231), ('terrible', 227), ('engañar', 226), ('beso', 225), ('cansar', 225), ('recoger', 223), ('extrañar', 222), ('oler', 220), ('ok', 218), ('hey', 218), ('baile', 216), ('guapo', 216), ('edad', 216), ('enojar', 215), ('deseo', 213), ('rato', 213), ('alegro', 211), ('ama', 209), ('princesa', 207), ('alegrar', 207), ('compañero', 207), ('nervioso', 205), ('mesa', 201), ('fumar', 201), ('favorito', 200), ('nota', 197), ('don', 195), ('despedir', 195), ('culo', 192), ('arruinar', 188), ('carta', 188), ('matrimonio', 185), ('annie', 185), ('triste', 185), ('reír', 183), ('tarjeta', 177), ('pastel', 172), ('copa', 169), ('joe', 169),

Como se puede observar a simple vista, estas palabras son realmente representativas de cada género. Es notorio, sin embargo, que el género de Horror tenga un vocabulario tan pequeño que tenga una frecuencia exclusivamente alta dentro del género.

# Comparación de Frecuencias entre géneros y lenguas

También implementé la función "comparar_generos" para ver cómo varía la frecuencia de las palabras a través de géneros y de idiomas. Es fácil notar algunas tendencias interesantes con respecto a esto, y por tanto a continuación expongo algunos ejemplos:

In [28]:
```
myutils.comparar_generos('princesa', 'princess', 'princesse', frecs_spa, frecs_
eng, frecs_fre)
```

```
Genero: accion, Spa: 47, Eng: 43, Fre:4
Genero: romance, Spa: 139, Eng: 140, Fre:6
Genero: horror, Spa: 21, Eng: 17, Fre:2
```

In [29]:
```
myutils.comparar_generos('hombre', 'man', 'homme', frecs_spa, frecs_eng, frecs_
fre)
```

```
Genero: accion, Spa: 1190, Eng: 2135, Fre:971
Genero: romance, Spa: 1145, Eng: 2189, Fre:708
Genero: horror, Spa: 525, Eng: 1268, Fre:411
```

In [30]:
```
myutils.comparar_generos('mujer', 'woman', 'femme', frecs_spa, frecs_eng, frecs
_fre)
```

```
Genero: accion, Spa: 319, Eng: 277, Fre:387
Genero: romance, Spa: 825, Eng: 843, Fre:930
Genero: horror, Spa: 284, Eng: 418, Fre:365
```

In [31]:
```
myutils.comparar_generos('padre', 'father', 'père', frecs_spa, frecs_eng, frecs
_fre)
```

```
Genero: accion, Spa: 775, Eng: 614, Fre:727
Genero: romance, Spa: 662, Eng: 413, Fre:735
Genero: horror, Spa: 514, Eng: 398, Fre:499
```

In [32]:
```
myutils.comparar_generos('papá', 'dad', 'papa', frecs_spa, frecs_eng, frecs_fre
)
```

```
Genero: accion, Spa: 364, Eng: 316, Fre:270
Genero: romance, Spa: 743, Eng: 785, Fre:551
Genero: horror, Spa: 406, Eng: 396, Fre:415
```

In [33]:
```
myutils.comparar_generos('madre', 'mother', 'mère', frecs_spa, frecs_eng, frecs
_fre)
```

```
Genero: accion, Spa: 313, Eng: 270, Fre:338
Genero: romance, Spa: 467, Eng: 497, Fre:646
Genero: horror, Spa: 425, Eng: 445, Fre:441
```

In [34]:
```
myutils.comparar_generos('mamá', 'mom', 'maman', frecs_spa, frecs_eng, frecs_fr
e)
```

```
Genero: accion, Spa: 245, Eng: 211, Fre:203
Genero: romance, Spa: 690, Eng: 524, Fre:424
Genero: horror, Spa: 599, Eng: 309, Fre:463
```

In [35]:
```
myutils.comparar_generos('casa', 'house', 'maison', frecs_spa, frecs_eng, frecs
_fre)
```

```
Genero: accion, Spa: 562, Eng: 215, Fre:227
Genero: romance, Spa: 1044, Eng: 419, Fre:324
Genero: horror, Spa: 835, Eng: 483, Fre:445
```

In [36]:
```
myutils.comparar_generos('morir', 'die', 'mourir', frecs_spa, frecs_eng, frecs_
fre)
```

```
Genero: accion, Spa: 1190, Eng: 725, Fre:613
Genero: romance, Spa: 530, Eng: 373, Fre:265
Genero: horror, Spa: 947, Eng: 599, Fre:538
```

In [37]:
```
myutils.comparar_generos('muerte', 'death', 'mort', frecs_spa, frecs_eng, frecs
_fre)
```

```
Genero: accion, Spa: 341, Eng: 297, Fre:907
Genero: romance, Spa: 96, Eng: 117, Fre:351
Genero: horror, Spa: 266, Eng: 260, Fre:754
```

In [40]:
```
myutils.comparar_generos('sangre', 'blood', 'sang', frecs_spa, frecs_eng, frecs
_fre)
```

```
Genero: accion, Spa: 188, Eng: 181, Fre:277
Genero: romance, Spa: 70, Eng: 77, Fre:195
Genero: horror, Spa: 336, Eng: 352, Fre:465
```

In [41]:
```
myutils.comparar_generos('sol', 'sun', 'soleil', frecs_spa, frecs_eng, frecs_fr
e)
```

```
Genero: accion, Spa: 73, Eng: 71, Fre:85
Genero: romance, Spa: 99, Eng: 101, Fre:99
Genero: horror, Spa: 77, Eng: 55, Fre:81
```

In [38]:
```
myutils.comparar_generos('amar', 'love', 'aimer', frecs_spa, frecs_eng, frecs_f
re)
```

```
Genero: accion, Spa: 212, Eng: 736, Fre:865
Genero: romance, Spa: 460, Eng: 2707, Fre:2219
Genero: horror, Spa: 158, Eng: 601, Fre:708
```

In [39]:
```
myutils.comparar_generos('amor', 'love', 'amour', frecs_spa, frecs_eng, frecs_f
re)
```

```
Genero: accion, Spa: 172, Eng: 736, Fre:114
Genero: romance, Spa: 621, Eng: 2707, Fre:576
Genero: horror, Spa: 176, Eng: 601, Fre:109
```

# Segmentando el corpus (explicación)

Finalmente, usaré esta sección para exponer en general la manera en la que procesaré los documentos antes de pasarlos al entrenamiento del LDA.

Como ya se vio al inicio de este cuaderno, guardo mis documentos en una matriz, como "txts_spa" donde "txts_spa[i][j]" se refiere al documento j del género i. Cabe decir que el documento j es un string con todo el contenido del documento lematizado.

El primer paso será convertir esa lista de listas de textos en listas de listas de listas de palabras. De tal suerte que unfiltered_lists[i][j] se refiere a la palabra j del documento i en la lista de documentos "unfiltered_lists":

In [42]:
```
unfiltered_lists = myutils.texts2lists(txts_spa[0], stopwords = None)
print(unfiltered_lists[0][:100])
```

```
['hacer', 'mucho', 'tiempo', 'en', 'uno', 'galaxia', 'muy', 'muy', 'lejano', 'e
pisodio', 'iv', 'una', 'nueva', 'esperanza', 'ser', 'uno', 'período', 'de', 'gu
erra', 'civil', 'el', 'nave', 'espacial', 'rebelde', 'estacionar', 'en', 'uno',
'base', 'oculto', 'haber', 'lograr', 'su', 'victoria', 'en', 'contra', 'de', 'e
l', 'malvado', 'imperio', 'galáctico', 'durante', 'el', 'batalla', 'el', 'espía
', 'rebelde', 'se', 'el', 'haber', 'ingeniar', 'para', 'robar', 'el', 'plano',
'secreto', 'acerca', 'de', 'el', 'último', 'arma', 'de', 'el', 'imperio', 'deno
minar', 'estrella', 'de', 'la', 'muerte', 'uno', 'estación', 'espacial', 'blind
ar', 'con', 'el', 'suficiente', 'poder', 'como', 'para', 'destruir', 'uno', 'pl
aneta', 'entero', 'perseguir', 'por', 'el', 'siniestro', 'agente', 'de', 'el',
'imperio', 'el', 'princesa', 'leia', 'se', 'dirigir', 'su', 'hogar', 'en', 'su'
, 'nave']
```

Típicamente usaré ese paso para aprovechar y quitar palabras de paro. Normalmente obtendré mi lista de palabras de paro utilizando la función "intersect_most_common" ya expuesta, para quitar las palabras más frecuentes pero que no sean típicas de un género.

La razón para no quitarlas desde antes es que conforme fui avanzando en el proyecto me di cuenta que variar las stopwords tenía un impacto significativo en los tópicos obtenidos, por lo que es mejor dejar el paso de quitar los stopwords hasta casi el momento en le que vaya a entrenar un modelo.

In [43]:
```
stopwords = myutils.intersect_most_common(frecs_spa)
filtered_lists = myutils.texts2lists(txts_spa[0], stopwords = stopwords)
print(filtered_lists[0][:100])
```

```
['galaxia', 'lejano', 'episodio', 'iv', 'una', 'nueva', 'esperanza', 'período',
'guerra', 'civil', 'nave', 'espacial', 'rebelde', 'estacionar', 'base', 'oculto
', 'lograr', 'victoria', 'contra', 'malvado', 'imperio', 'galáctico', 'durante'
, 'batalla', 'espía', 'rebelde', 'ingeniar', 'robar', 'plano', 'secreto', 'acer
ca', 'último', 'arma', 'imperio', 'denominar', 'estrella', 'la', 'muerte', 'est
ación', 'espacial', 'blindar', 'suficiente', 'destruir', 'planeta', 'entero', '
perseguir', 'siniestro', 'agente', 'imperio', 'princesa', 'leia', 'dirigir', 'h
ogar', 'nave', 'espacial', 'custodiar', 'plano', 'robar', 'salvar', 'gente', 'r
estaurar', 'libertad', 'galaxia', 'alcanzar', 'reactor', 'principal', 'destruir
', 'seguro', 'locura', 'perder', 'princesa', 'escapar', 'dónde', 'fin', 'dónde'
, 'meter', 'dirección', 'mandar', 'mina', 'kessel', 'triturar', 'momento', 'adó
nde', 'vas', 'plano', 'estrella', 'la', 'muerte', 'ordenador', 'central', 'dónd
e', 'transmisión', 'interceptar', 'plano', 'interceptar', 'ninguno', 'transmisi
ón', 'nave', 'consular', 'misión']
```

Como se puede notar, la versión que filtra stopwords es mucho menos redundante y más representativa de la película a la que representa.

Tras convertir los strings en listas de palabras y filtrar stopwords, viene la parte que he llamado "segmentación del corpus". Con este término me refiero a tomar cada documento del corpus y subdividirlo (segmentarlo) en más documentos.

Para esto he implementado la función "split_lists" la cual toma y devuelve una lista de listas, pero la devuelta es tal que está segmentada. Como parámetros tiene "chunk_size" (tamaño máximo de los segmentos en los que se dividirá el documento) y overlap_size (tamaño de translape entre segmentos).

In [44]:
```
splitted_lists = myutils.split_lists(filtered_lists, chunk_size = 20, overlap_s
ize = 5)
print(splitted_lists[0], "\n")
print(splitted_lists[1])
```

```
['galaxia', 'lejano', 'episodio', 'iv', 'una', 'nueva', 'esperanza', 'período',
'guerra', 'civil', 'nave', 'espacial', 'rebelde', 'estacionar', 'base', 'oculto
', 'lograr', 'victoria', 'contra', 'malvado']

['oculto', 'lograr', 'victoria', 'contra', 'malvado', 'imperio', 'galáctico', '
durante', 'batalla', 'espía', 'rebelde', 'ingeniar', 'robar', 'plano', 'secreto
', 'acerca', 'último', 'arma', 'imperio', 'denominar']
```

In [45]:
```
print("Tamaño de dataset antes de segmentar:", len(filtered_lists))
print("Tamaño del dataset tras segmentar:", len(splitted_lists))
```

```
Tamaño de dataset antes de segmentar: 100
Tamaño del dataset tras segmentar: 19176
```

Como se puede ver el documento original se segmentó en listas de 20 elementos, donde los primeros 5 elementos de la 2a lista son iguales a los últimos 5 de la primera y así sucesivamente.

El fin de utilizar este tipo de segmentación es para poder obtener más datos con los cuales entrenar (ya que en principio solo cuento con 300 subtítulos por lengua), y también para permitir que LDA descubra relaciones más locales en cada pedazo del corpus.

Finalmente utilizo el diccionario previamente obtenido para convertir los documentos en vectores dispersos propios de Gensim:

In [46]:
```
vectors = myutils.lists2bow(splitted_lists, dictionary_spa)
print(vectors[0])
```

```
[(146, 1), (235, 1), (283, 1), (484, 1), (500, 1), (506, 1), (510, 1), (593, 1)
, (618, 1), (708, 1), (734, 1), (759, 1), (779, 1), (859, 1), (878, 1), (886, 1
), (971, 1), (1072, 1), (1322, 1), (1355, 1)]
```