LDA Latent Dirichlet Allocation(LDA)

In [71]:
```
# Topic modeling is a method for unsupervised classification of documents, similar to clustering on numeric data, whi
# of items (topics) even when we're not sure what we're looking for. (https://towardsdatascience.com/latent-dirichlet

#TENGO QUE LEMATIZAR EL TEXTO, YA ESTA STEMMING
```

In [72]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction import text
import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier

import re
from nltk.tokenize import wordpunct_tokenize

from nltk.tokenize import RegexpTokenizer
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.porter import PorterStemmer
```

In [73]:
```
# !pip install pyLDAvis
```

In [74]:
```
# !pip install gensim
```

In [75]:
```
import gensim
from gensim import corpora
from gensim.models import CoherenceModel# Coherence score and perplexity provide a convinent way to measure how good
import pyLDAvis.gensim_models
```

In [76]:
```
# nltk.download('stopwords')
# nltk.download('punkt')
```

In [77]:
```
base = pd.read_csv('C:/Users/farav/OneDrive/Documentos/Python Scripts/base_tit_senti_fecha.csv', encoding='iso8859_2
```

In [78]:
```
base_elobse = base[base['diario'] == 'elobse']
```

In [79]:
```
base_elobse_fecha = base_elobse[base_elobse['fecha'] == '2021-07-23']
```

In [80]:
```
stop_words = list(stopwords.words('spanish'))
espanol = pd.read_csv('spanish.txt', header=None)
espanol.columns = ['palabra']
list_espanol = list(espanol['palabra'])
stop_words.extend(list_espanol)

unique_stop_words = []
[unique_stop_words.append(word) for word in stop_words if word not in unique_stop_words]
len(unique_stop_words)
```

Out[80]: 615

In [81]:
```
def limpiar_tokenizar(texto):
    porter_stemmer = PorterStemmer()
    nuevo_texto = texto.lower()
    nuevo_texto = re.sub("\d+", ' ', nuevo_texto)
    nuevo_texto = wordpunct_tokenize(nuevo_texto)# esto toma espacios, símbolos, comas, etc. y los tokeniza junto cor
    nuevo_texto = [token for token in nuevo_texto if len(token) > 3]
    nuevo_texto = [token for token in nuevo_texto if token not in unique_stop_words]
    stemmers = [porter_stemmer.stem(word) for word in nuevo_texto]
    nuevo_texto = [stem for stem in stemmers if stem.isalpha() and len(stem) > 1]

    return(nuevo_texto)
```

```
<>:4: DeprecationWarning: invalid escape sequence '\d'
<>:4: DeprecationWarning: invalid escape sequence '\d'
C:\Users\farav\AppData\Local\Temp\ipykernel_11928\2669027614.py:4: DeprecationWarning: invalid escape sequence '\d'
  nuevo_texto = re.sub("\d+", ' ', nuevo_texto)
```

In [82]:
```
base_elobse_fecha['tit_tokenizado'] = base_elobse_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
tokenizado = list(base_elobse_fecha['tit_tokenizado'])
```

```
C:\Users\farav\AppData\Local\Temp\ipykernel_11928\3183837091.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  base_elobse_fecha['tit_tokenizado'] = base_elobse_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
```

In [83]:
```
dictionary = corpora.Dictionary(tokenizado)
dictionary.filter_extremes(no_below=3)# tenía esto y lo saqué: , keep_n=300
```

In [84]:
```
print(dictionary)
```

```
Dictionary<12 unique tokens: ['inauguración', 'juego', 'olímpico', 'tokio', 'cancha']...>
```

In [85]:
```
corpus = [dictionary.doc2bow(text) for text in tokenizado]
```

In [ ]:

In [86]:
```
corpus[0][:10]
```

Out[86]:
```
[(0, 1), (1, 1), (2, 1), (3, 1)]
```

In [178…
```
# Esto es para seleccionar la cantidad de tópicos, no cambian los valores

results = []

for t in range(2, 10):
    lda_model = gensim.models.ldamodel.LdaModel(corpus, id2word=dictionary, num_topics=t, passes=10)
    #lda_model = models.LdaModel(corpus_4, id2word=dictionary_4, num_topics=t, passes=10)
    corpus_lda = lda_model[corpus]

    cm = CoherenceModel(model=lda_model, corpus=corpus_lda, texts=tokenizado, coherence='c_v')
    score = cm.get_coherence()
    tup = t, score
    results.append(tup)

results = pd.DataFrame(results, columns=['topic', 'score'])
```

In [179…
```
results
```

Out[179]:

	topic	score
0	2	0.502815
1	3	0.502815
2	4	0.502815
3	5	0.502815
4	6	0.502815
5	7	0.502815
6	8	0.502815
7	9	0.502815

In [87]:
```
# Define the LDA model
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=5, id2word=dictionary, passes=10)# revisar los argumen

# Save the topics and top 5 words
topics = ldamodel.print_topics(num_words=5)

# Print the results
```

```
for topic in topics:
    print(topic)
```

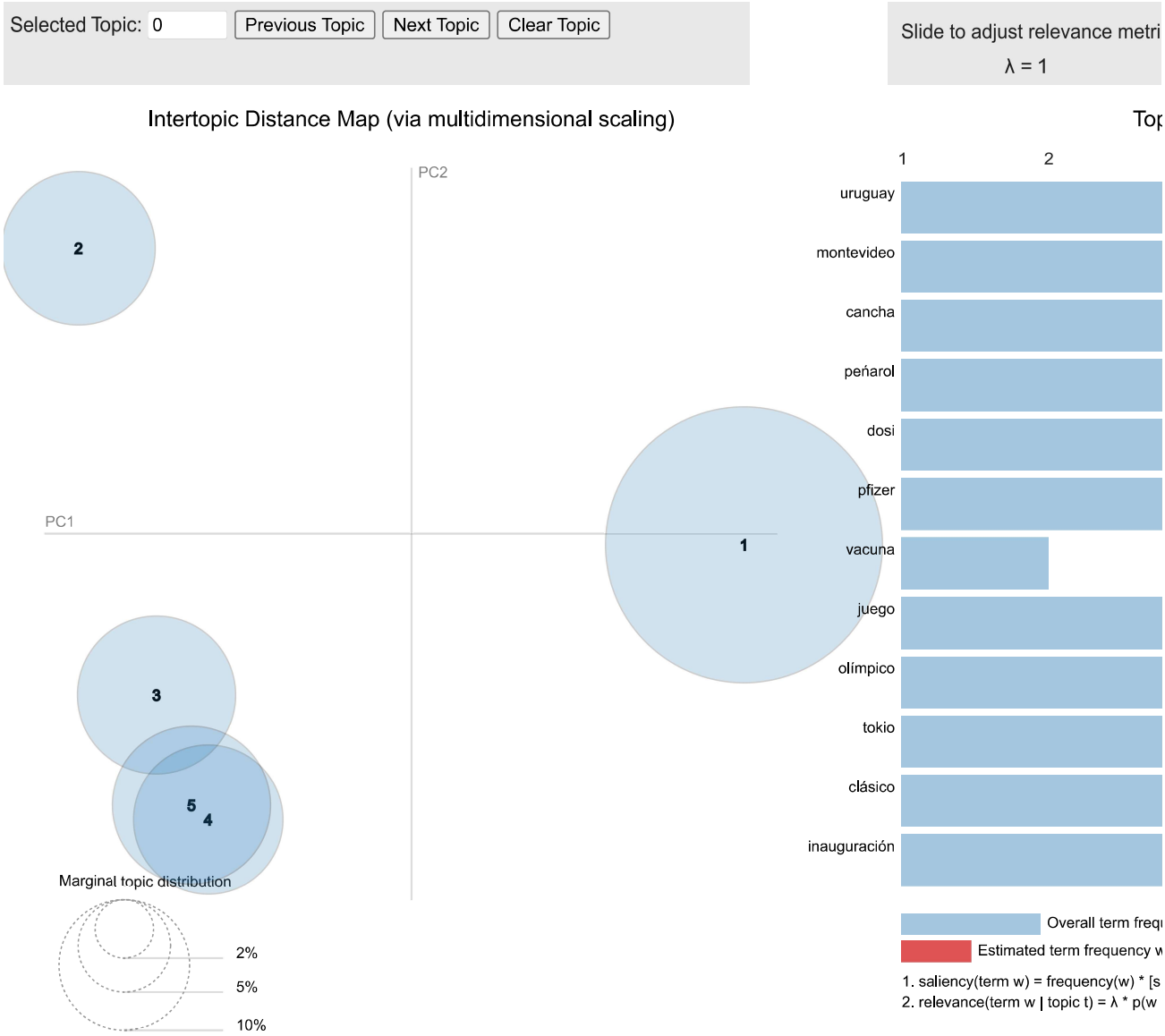
```
(0, '0.262*"juego" + 0.262*"olímpico" + 0.190*"tokio" + 0.117*"clásico" + 0.116*"inauguración"')
(1, '0.445*"montevideo" + 0.339*"vacuna" + 0.022*"olímpico" + 0.022*"juego" + 0.022*"inauguración"')
(2, '0.377*"cancha" + 0.377*"peñarol" + 0.029*"pfizer" + 0.027*"clásico" + 0.024*"montevideo"')
(3, '0.675*"uruguay" + 0.042*"pfizer" + 0.041*"dosi" + 0.028*"tokio" + 0.028*"juego"')
(4, '0.376*"dosi" + 0.369*"pfizer" + 0.028*"juego" + 0.028*"olímpico" + 0.027*"inauguración"')
```

In [88]:
```
lda_display_elobse = pyLDAvis.gensim_models.prepare(ldamodel, corpus, dictionary, sort_topics=False)
```

```
C:\Users\farav\AppData\Local\Programs\Python\Python310\lib\site-packages\pyLDAvis\_prepare.py:247: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  by='saliency', ascending=False).head(R).drop('saliency', 1)
```

In [89]:
```
pyLDAvis.display(lda_display_elobse)
```

Out[89]:



In [90]:
```
# Coherence score and perplexity provide a convinent way to measure how good a given topic model is.

# Lower the perplexity better the model.
# Higher the topic coherence, the topic is more human interpretable.
```

In [91]:
```
# Compute Perplexity
print('\nPerplexity: ', ldamodel.log_perplexity(corpus))
# a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=ldamodel, texts=tokenizado, dictionary=dictionary, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -3.0238594828113827

Coherence Score:  0.5028153977517918
```

In [92]:
```
base_ladiar = base[base['diario'] == 'ladiar']
base_ladiar_fecha = base_ladiar[base_ladiar['fecha'] == '2021-07-23']
```

In [93]:
```
base_ladiar_fecha['tit_tokenizado_1'] = base_ladiar_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
tokenizado_1 = list(base_ladiar_fecha['tit_tokenizado_1'])
```

```
C:\Users\farav\AppData\Local\Temp\ipykernel_11928\3857919374.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  base_ladiar_fecha['tit_tokenizado_1'] = base_ladiar_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
```

In [94]:
```
dictionary_1 = corpora.Dictionary(tokenizado_1)
dictionary_1.filter_extremes(no_below=3)# , keep_n=300
```

In [95]:
```
print(dictionary_1)
```

```
Dictionary<20 unique tokens: ['canelon', 'juego', 'olímpico', 'tokio', 'estafa']...>
```

In [96]:
```
corpus_1 = [dictionary_1.doc2bow(text) for text in tokenizado_1]
```

In [176…
```
# Esto es para seleccionar la cantidad de tópicos, no cambian los valores

results = []

for t in range(2, 10):
    lda_model = gensim.models.ldamodel.LdaModel(corpus_1, id2word=dictionary_1, num_topics=t, passes=10)
    #lda_model = models.LdaModel(corpus_4, id2word=dictionary_4, num_topics=t, passes=10)
    corpus_lda = lda_model[corpus_1]

    cm = CoherenceModel(model=lda_model, corpus=corpus_lda, texts=tokenizado_1, coherence='c_v')
    score = cm.get_coherence()
    tup = t, score
    results.append(tup)

results = pd.DataFrame(results, columns=['topic', 'score'])
```

In [177…
```
results
```

Out[177]:

	topic	score
0	2	0.712454
1	3	0.712454
2	4	0.712454
3	5	0.712454
4	6	0.712454
5	7	0.712454
6	8	0.712454
7	9	0.712454

In [97]:
```
# Define the LDA model
ldamodel_1 = gensim.models.ldamodel.LdaModel(corpus_1, num_topics=5, id2word=dictionary_1, passes=10)
```

```
# Save the topics and top 5 words
topics = ldamodel_1.print_topics(num_words=5)

# Print the results
for topic in topics:
    print(topic)
```

```
(0, '0.355*"pandemia" + 0.211*"tiempo" + 0.161*"apunt" + 0.111*"mujer" + 0.010*"coronaviru"')
(1, '0.198*"juego" + 0.166*"olímpico" + 0.134*"tokio" + 0.134*"coronaviru" + 0.102*"política"')
(2, '0.281*"uruguay" + 0.210*"gigant" + 0.169*"número" + 0.089*"cabildo" + 0.089*"abierto"')
(3, '0.227*"falta" + 0.226*"gobierno" + 0.157*"uruguayo" + 0.085*"abierto" + 0.085*"cabildo"')
(4, '0.319*"estafa" + 0.318*"canelon" + 0.021*"juego" + 0.021*"olímpico" + 0.020*"apunt"')
```

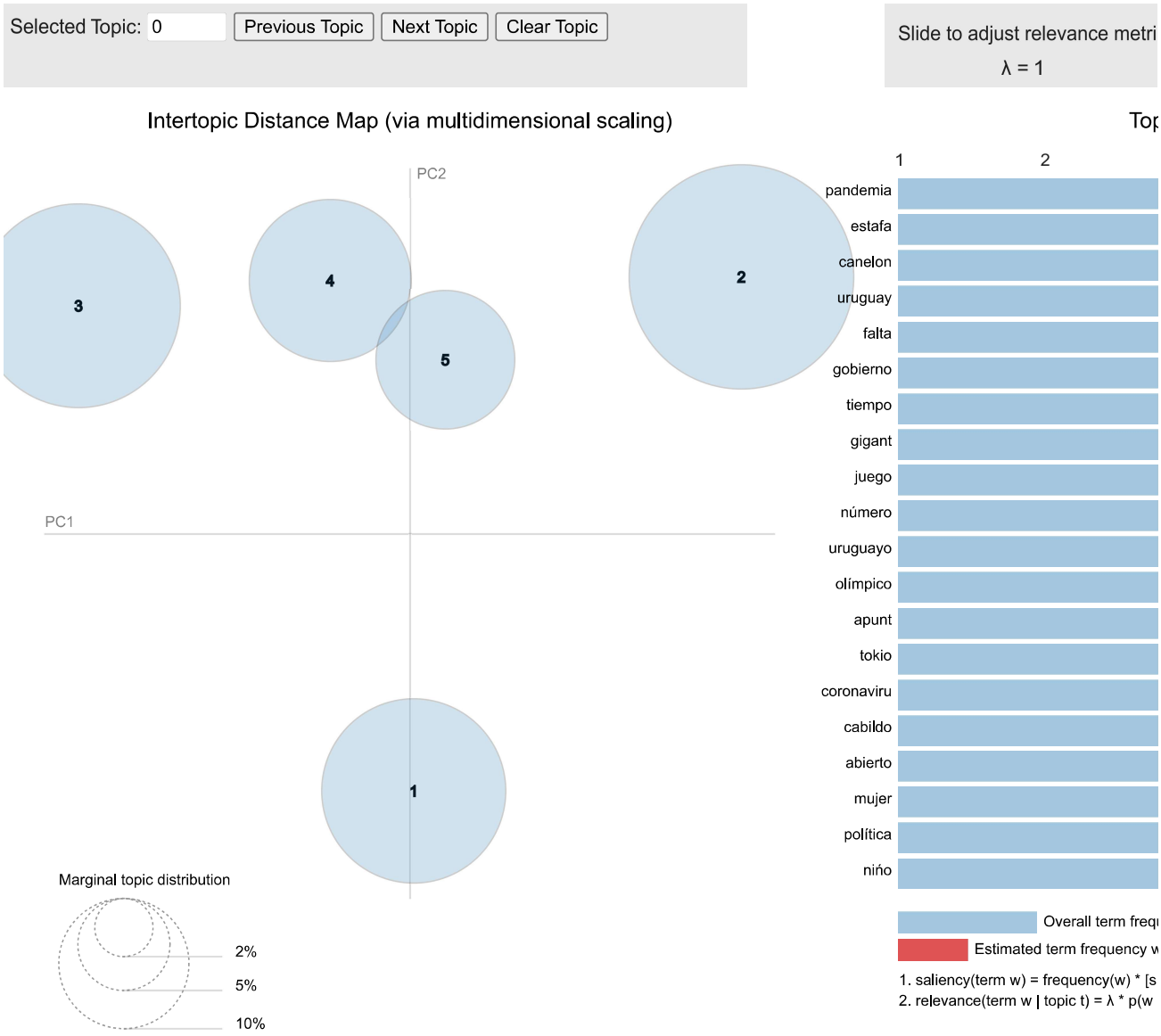
In [ ]:

In [98]:
```
lda_display_ladiar = pyLDAvis.gensim_models.prepare(ldamodel_1, corpus_1, dictionary_1, sort_topics=False)
```

```
C:\Users\farav\AppData\Local\Programs\Python\Python310\lib\site-packages\pyLDAvis\_prepare.py:247: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  by='saliency', ascending=False).head(R).drop('saliency', 1)
```

In [99]:
```
pyLDAvis.display(lda_display_ladiar)
```

Out[99]:



In [100…
```
# Compute Perplexity
print('\nPerplexity: ', ldamodel_1.log_perplexity(corpus_1))
# a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=ldamodel_1, texts=tokenizado_1, dictionary=dictionary_1, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -3.703646237320716

Coherence Score:  0.7124535429118619
```

In [101…
```
base_elpais = base[base['diario'] == 'elpais']
base_elpais_fecha = base_elpais[base_elpais['fecha'] == '2021-07-23']
```

In [102…
```
base_elpais_fecha['tit_tokenizado_2'] = base_elpais_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
tokenizado_2 = list(base_elpais_fecha['tit_tokenizado_2'])
```

```
C:\Users\farav\AppData\Local\Temp\ipykernel_11928\494036624.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  base_elpais_fecha['tit_tokenizado_2'] = base_elpais_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
```

In [103…
```
dictionary_2 = corpora.Dictionary(tokenizado_2)
dictionary_2.filter_extremes(no_below=3)# , keep_n=300
```

In [104…
```
print(dictionary_2)
```

```
Dictionary<10 unique tokens: ['peñarol', 'juego', 'olímpico', 'uruguay', 'netflix']...>
```

In [105…
```
corpus_2 = [dictionary_2.doc2bow(text) for text in tokenizado_2]
```

In [166…
```
# Esto es para seleccionar la cantidad de tópicos, no cambian los valores

results = []

for t in range(2, 10):
    lda_model = gensim.models.ldamodel.LdaModel(corpus_2, id2word=dictionary_2, num_topics=t, passes=10)
    #lda_model = models.LdaModel(corpus_4, id2word=dictionary_4, num_topics=t, passes=10)
    corpus_lda = lda_model[corpus_2]

    cm = CoherenceModel(model=lda_model, corpus=corpus_lda, texts=tokenizado_2, coherence='c_v')
    score = cm.get_coherence()
    tup = t, score
    results.append(tup)

results = pd.DataFrame(results, columns=['topic', 'score'])
```

In [167…
```
results
```

Out[167]:

	topic	score
0	2	0.526359
1	3	0.526359
2	4	0.526359
3	5	0.526359
4	6	0.526359
5	7	0.526359
6	8	0.526359
7	9	0.526359

In [172…
```
# Define the LDA model
ldamodel_2 = gensim.models.ldamodel.LdaModel(corpus_2, num_topics=5, id2word=dictionary_2, passes=10)
```

```
# Save the topics and top 5 words
topics = ldamodel_2.print_topics(num_words=5)

# Print the results
for topic in topics:
    print(topic)
```

```
(0, '0.639*"baja" + 0.042*"netflix" + 0.040*"uruguay" + 0.040*"marca" + 0.040*"videojuego"')
(1, '0.637*"netflix" + 0.042*"videojuego" + 0.040*"uruguay" + 0.040*"marca" + 0.040*"baja"')
(2, '0.355*"peñarol" + 0.355*"marca" + 0.133*"clasificación" + 0.023*"videojuego" + 0.023*"netflix"')
(3, '0.466*"vuelta" + 0.245*"clasificación" + 0.134*"uruguay" + 0.022*"baja" + 0.022*"marca"')
(4, '0.280*"uruguay" + 0.214*"olímpico" + 0.214*"juego" + 0.213*"videojuego" + 0.013*"netflix"')
```

In [173…

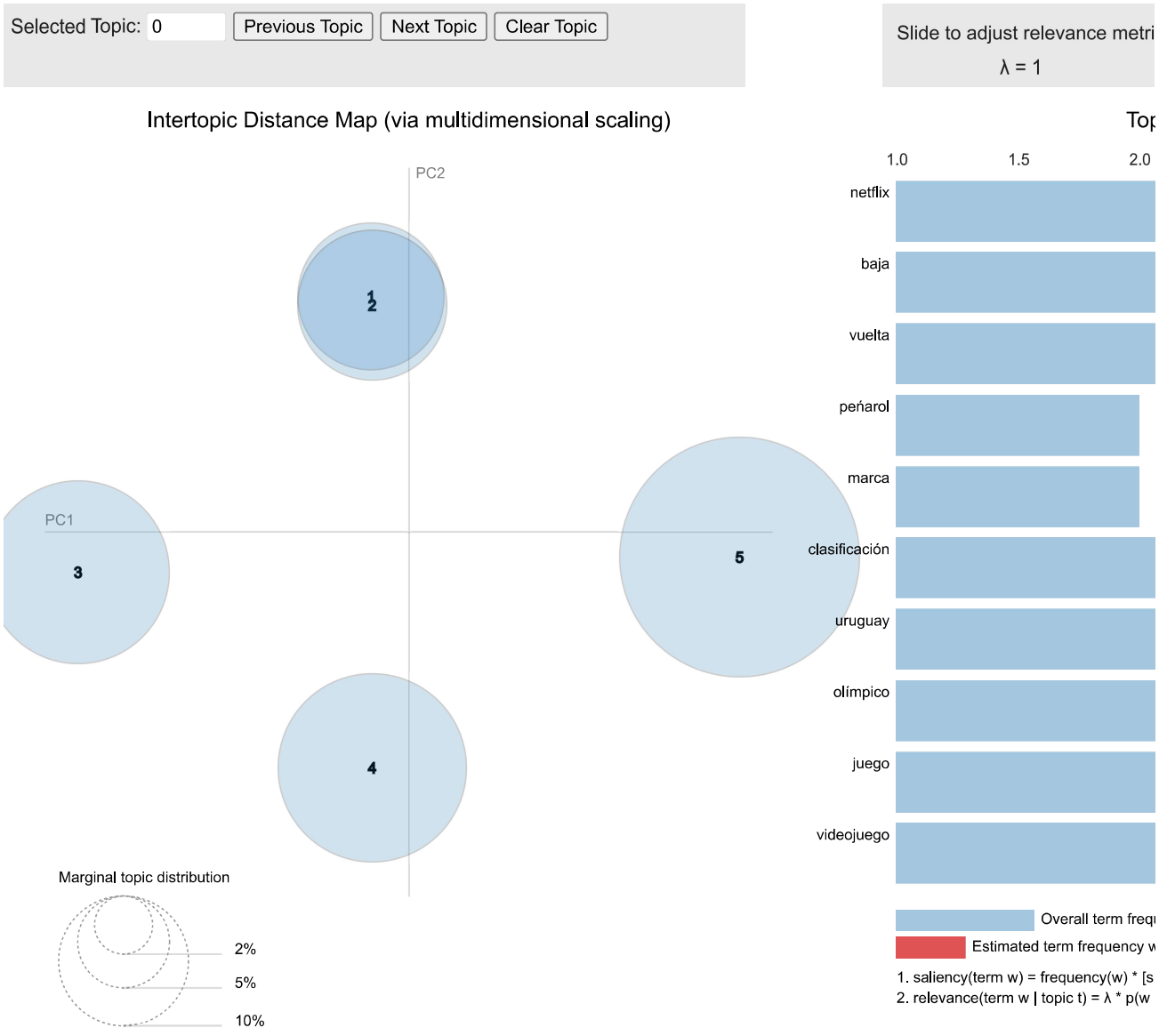
```
lda_display_elpais = pyLDAvis.gensim_models.prepare(ldamodel_2, corpus_2, dictionary_2, sort_topics=False)
```

```
C:\Users\farav\AppData\Local\Programs\Python\Python310\lib\site-packages\pyLDAvis\_prepare.py:247: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  by='saliency', ascending=False).head(R).drop('saliency', 1)
```

In [174…

```
pyLDAvis.display(lda_display_elpais)
```

Out[174]:



In [175…

```
# Compute Perplexity
print('\nPerplexity: ', ldamodel_2.log_perplexity(corpus_2))
```

```
# a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=ldamodel_2, texts=tokenizado_2, dictionary=dictionary_2, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -3.194043850386862

Coherence Score:  0.5263586384049894
```

In [110…
```
base_montev = base[base['diario'] == 'montev']
base_montev_fecha = base_montev[base_montev['fecha'] == '2021-07-23']
```

In [111…
```
base_montev_fecha['tit_tokenizado_3'] = base_montev_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
tokenizado_3 = list(base_montev_fecha['tit_tokenizado_3'])
```

```
C:\Users\farav\AppData\Local\Temp\ipykernel_11928\1743992655.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returnin
g-a-view-versus-a-copy
  base_montev_fecha['tit_tokenizado_3'] = base_montev_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
```

In [112…
```
dictionary_3 = corpora.Dictionary(tokenizado_3)
dictionary_3.filter_extremes(no_below=3)# , keep_n=300
```

In [113…
```
print(dictionary_3)
```

```
Dictionary<10 unique tokens: ['uruguay', 'clásico', 'nacion', 'peñarol', 'juego']...>
```

In [114…
```
corpus_3 = [dictionary_3.doc2bow(text) for text in tokenizado_3]
```

In [164…
```
# Esto es para seleccionar la cantidad de tópicos, hay algo raro porque no cambian los valores

results = []

for t in range(2, 10):
    lda_model = gensim.models.ldamodel.LdaModel(corpus_3, id2word=dictionary_3, num_topics=t, passes=10)
    #lda_model = models.LdaModel(corpus_4, id2word=dictionary_4, num_topics=t, passes=10)
    corpus_lda = lda_model[corpus_3]

    cm = CoherenceModel(model=lda_model, corpus=corpus_lda, texts=tokenizado_3, coherence='c_v')
    score = cm.get_coherence()
    tup = t, score
    results.append(tup)

results = pd.DataFrame(results, columns=['topic', 'score'])
```

In [165…
```
results
```

Out[165]:

	topic	score
0	2	0.522956
1	3	0.522956
2	4	0.522956
3	5	0.522956
4	6	0.522956
5	7	0.522956
6	8	0.522956
7	9	0.522956

In [115…
```
# Define the LDA model
ldamodel_3 = gensim.models.ldamodel.LdaModel(corpus_3, num_topics=5, id2word=dictionary_3, passes=10)

# Save the topics and top 5 words
topics = ldamodel_3.print_topics(num_words=5)
```

```
# Print the results
for topic in topics:
    print(topic)
```

```
(0, '0.634*"hombr" + 0.042*"opinión" + 0.042*"cuba" + 0.042*"juego" + 0.042*"sudamericana"')
(1, '0.314*"uruguay" + 0.314*"opinión" + 0.240*"cuba" + 0.036*"nacion" + 0.019*"clásico"')
(2, '0.530*"delta" + 0.200*"clásico" + 0.035*"hombr" + 0.034*"uruguay" + 0.033*"nacion"')
(3, '0.350*"nacion" + 0.229*"sudamericana" + 0.229*"peñarol" + 0.139*"clásico" + 0.009*"hombr"')
(4, '0.631*"juego" + 0.044*"uruguay" + 0.044*"opinión" + 0.044*"delta" + 0.040*"nacion"')
```

In [116…
```
lda_display_montev = pyLDAvis.gensim_models.prepare(ldamodel_3, corpus_3, dictionary_3, sort_topics=False)
```

```
C:\Users\farav\AppData\Local\Programs\Python\Python310\lib\site-packages\pyLDAvis\_prepare.py:247: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  by='saliency', ascending=False).head(R).drop('saliency', 1)
```

In [117…
```
pyLDAvis.display(lda_display_montev)
```

Out[117]:

In [118…
```
# Compute Perplexity
print('\nPerplexity: ', ldamodel_3.log_perplexity(corpus_3))
# a measure of how good the model is. lower the better.

# Compute Coherence Score
```

```
coherence_model_lda = CoherenceModel(model=ldamodel_3, texts=tokenizado_3, dictionary=dictionary_3, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -2.9883680274684408

Coherence Score:  0.5229562142470268
```

In [119…
```
base_republ = base[base['diario'] == 'republ']
base_republ_fecha = base_republ[base_republ['fecha'] == '2021-07-23']
```

In [120…
```
base_republ_fecha['tit_tokenizado_4'] = base_republ_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
tokenizado_4 = list(base_republ_fecha['tit_tokenizado_4'])
#tokenizado_4
```

```
C:\Users\farav\AppData\Local\Temp\ipykernel_11928\2681922119.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  base_republ_fecha['tit_tokenizado_4'] = base_republ_fecha['titular'].apply(lambda x: limpiar_tokenizar(x))
```

In [121…
```
dictionary_4 = corpora.Dictionary(tokenizado_4)
dictionary_4.filter_extremes(no_below=3)# , keep_n=300
```

In [122…
```
print(dictionary_4)
```

```
Dictionary<3 unique tokens: ['juego', 'olímpico', 'mujer']>
```

In [123…
```
corpus_4 = [dictionary_4.doc2bow(text) for text in tokenizado_4]
```

In [160…
```
# Esto es para seleccionar la cantidad de tópicos, en el caso del diario la republica parece que no cambia nada, me p
# que hay algo raro con este diario, luego analizar qué está pasando.

from gensim import models

results = []

for t in range(1, 10):
    lda_model = gensim.models.ldamodel.LdaModel(corpus_4, id2word=dictionary_4, num_topics=t, passes=10)
    #lda_model = models.LdaModel(corpus_4, id2word=dictionary_4, num_topics=t, passes=10)
    corpus_lda = lda_model[corpus_4]

    cm = CoherenceModel(model=lda_model, corpus=corpus_lda, texts=tokenizado_4, coherence='c_v')
    score = cm.get_coherence()
    tup = t, score
    results.append(tup)

results = pd.DataFrame(results, columns=['topic', 'score'])
```

In [161…
```
results
```

Out[161]:

	topic	score
0	1	0.348646
1	2	0.348646
2	3	0.348646
3	4	0.348646
4	5	0.348646
5	6	0.348646
6	7	0.348646
7	8	0.348646
8	9	0.348646

In [140…
```
# Define the LDA model
ldamodel_4 = gensim.models.ldamodel.LdaModel(corpus_4, num_topics=5, id2word=dictionary_4, passes=10)
```

```
# Save the topics and top 5 words
topics = ldamodel_4.print_topics(num_words=5)

# Print the results
for topic in topics:
    print(topic)
```

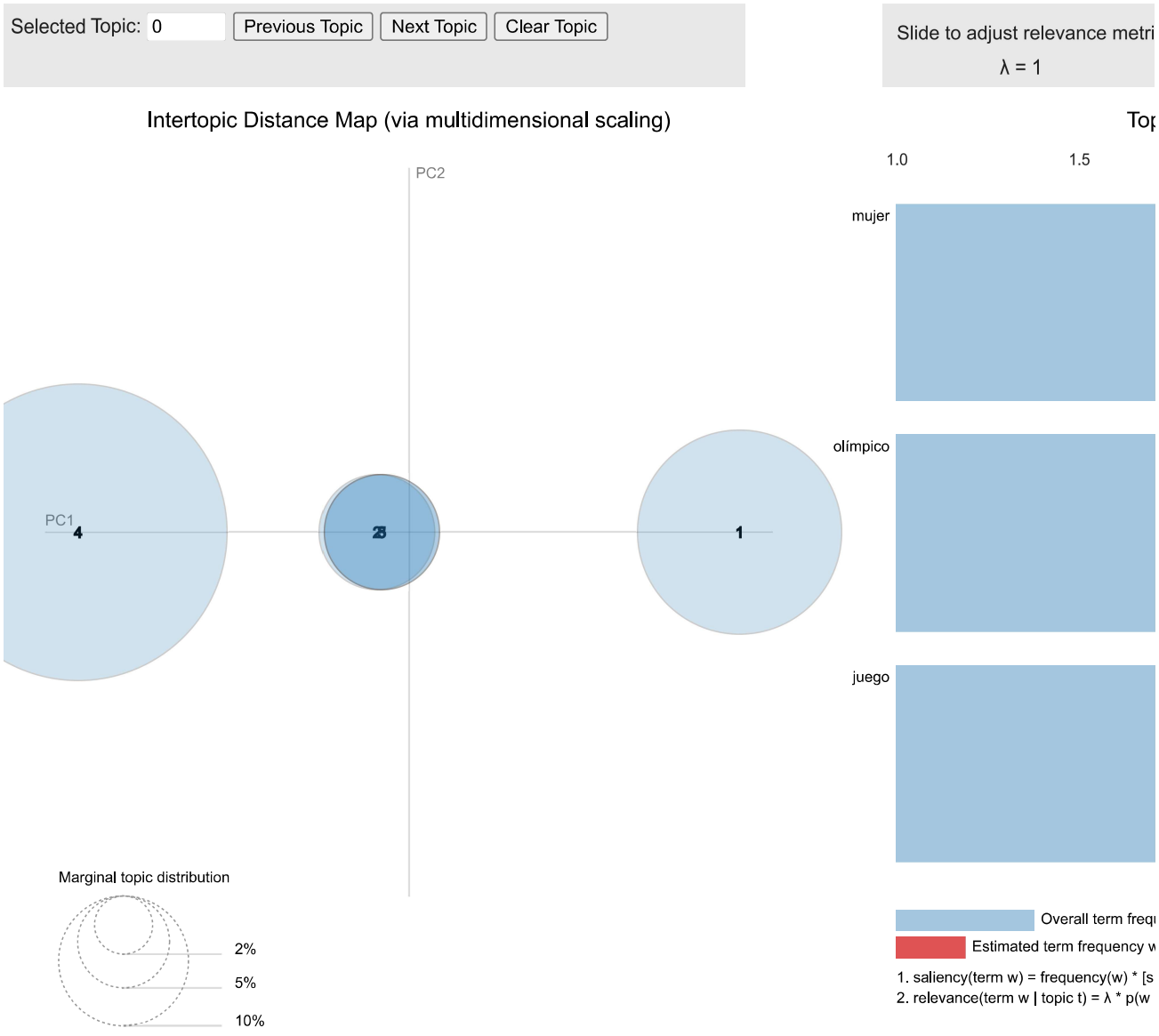
```
(0, '0.889*"mujer" + 0.056*"juego" + 0.056*"olímpico"')
(1, '0.338*"olímpico" + 0.338*"juego" + 0.323*"mujer"')
(2, '0.334*"mujer" + 0.333*"juego" + 0.333*"olímpico"')
(3, '0.485*"olímpico" + 0.485*"juego" + 0.030*"mujer"')
(4, '0.334*"mujer" + 0.333*"olímpico" + 0.333*"juego"')
```

In [141…
```
lda_display_republ = pyLDAvis.gensim_models.prepare(ldamodel_4, corpus_4, dictionary_4, sort_topics=False)
```

```
C:\Users\farav\AppData\Local\Programs\Python\Python310\lib\site-packages\pyLDAvis\_prepare.py:247: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only.
  by='saliency', ascending=False).head(R).drop('saliency', 1)
```

In [142…
```
pyLDAvis.display(lda_display_republ)
```

Out[142]:



In [143…
```
# Coherence score and perplexity provide a convinent way to measure how good a given topic model is.
```

```
# Lower the perplexity better the model.
# Higher the topic coherence, the topic is more human interpretable.
```

In [144…

```
# Compute Perplexity
print('\nPerplexity: ', ldamodel_4.log_perplexity(corpus_4))
# a measure of how good the model is. lower the better.

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=ldamodel_4, texts=tokenizado_4, dictionary=dictionary_4, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('\nCoherence Score: ', coherence_lda)
```

```
Perplexity:  -2.162357906014317

Coherence Score:  0.3486463135123885
```

In [ ]:

```
results = []

for t in range(2, 10):
    lda_model = models.LdaModel(corpus, id2word=dictionary, num_topics=t)
    corpus_lda = lda_model[corpus]

    cm = CoherenceModel(model=lda_model, corpus=corpus_lda, coherence='u_mass')
    score = cm.get_coherence()
    tup = t, score
    results.append(tup)

results = pd.DataFrame(results, columns=['topic', 'score'])
```