```
# Se installan los distintos paquetes que se van a utilizar
!pip install pandas-profiling==2.*
!pip install plotly==4.*
!pip install gensim
!pip install spacy --upgrade
!pip install pyldavis --upgrade
!pip install chart-studio --upgrade
!pip install wordcloud --upgrade
!pip install --upgrade --force-reinstall numpy
!pip install --upgrade --force-reinstall pandas
!pip install numpy==1.23.5
!pip install scipy==1.10.1
!pip install gensim==4.3.1
!pip install Unidecode
!pip install datatable --upgrade
!pip install emoji --upgrade
!pip install unidecode --upgrade
!pip install chart_studio --upgrade
!pip install pyldavis --upgrade

# Descarga de modelos de lenguaje para Spacy y NLTK
!python -m spacy download en_core_web_sm
!python -m spacy download es_core_news_sm



# Se actualizan las librerías necesarias con pip en caso de necesitarlo
!pip install --upgrade pip
!pip install --upgrade numpy
!pip install --upgrade pandas
!pip install --upgrade matplotlib
!pip install --upgrade seaborn
!pip install --upgrade emoji
!pip install --upgrade wordcloud
!pip install --upgrade pyldavis
!pip install --upgrade chart-studio
!pip install --upgrade regex
!pip install --upgrade spacy
!pip install --upgrade nltk
!pip install --upgrade autopep8
!pip install --upgrade datatable
!pip install --upgrade vaderSentiment
!pip install --upgrade unidecode


!pip install datatable --upgrade
!pip install emoji --upgrade


!pip install unidecode --upgrade
!pip install chart_studio --upgrade
!pip install pyldavis --upgrade


!pip install chart_studio --upgrade


!pip install pyldavis --upgrade


# Manejo y procesamiento de datos
import pandas as pd
import numpy as np
import json
import requests
import datatable as dt
```

```
# Limpieza de texto
import re
import string
import regex
import emoji
from unidecode import unidecode
from collections import Counter
from random import seed


# Visualización de datos
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
```

```
import plotly.express as px
import chart_studio
import chart_studio.plotly as py
import chart_studio.tools as tls
import pyLDAvis
import pyLDAvis.gensim_models as gensimvis


# Procesamiento de lenguaje natural (NLP)
import spacy
from spacy.tokenizer import Tokenizer
nlp = spacy.load('en_core_web_sm')


# Natural Language Toolkit
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')


# Modelado de tópicos (LDA, Gensim)
import gensim
from gensim import models
from gensim.corpora import Dictionary
from gensim.models import LdaMulticore, CoherenceModel
from gensim.models.coherencemodel import CoherenceModel
from gensim.parsing.preprocessing import STOPWORDS as SW


# Métodos de modelado adicionales
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
from sklearn.model_selection import GridSearchCV


# Otros
from pprint import pprint


#1. Empieza el PRE-PROCESAMIENTO DE LOS DATOS
nlp = spacy.load("en_core_web_sm")


df_analisis = pd.read_csv('/content/df_reddit_comentarios_en_limpios.csv', sep=';')


#Se comprueba el número de comentarios que hay
print("Número de comentarios:", df_analisis.shape[0])
```

```
Número de comentarios: 13087
```

```
#Se verifica que los datos se han cargado correctamente mostrando las primeras filas
print(df_analisis.head())
```

```
                                          Título                Fecha  \
0  I see many posts here saying that people want ...  2025-02-19 11:33:02
1  Collapse Comes Early for Canada: Prepare for U...  2025-02-19 04:57:29
2                        Predictions and Prophecies  2025-02-19 01:04:18
3  March 4, 2025 - Cataclysmic Shift and Future p...  2025-02-19 00:50:20
4  February 18, 2025, 4:17 pm | Nicaragua (NI) - ...  2025-02-18 16:17:56

             Subreddit           Autor        date  year  month  day language  \
0     TheHandmaidsTale        Poch1212  2025-02-19  2025      2   19       en
1         CollapsePrep       verdasuno  2025-02-19  2025      2   19       en
2              prophets    RosalieJewel  2025-02-19  2025      2   19       en
3  anonspropheticdream    RosalieJewel  2025-02-19  2025      2   19       en
4         ABCWorldNews  AcademicAd8273  2025-02-18  2025      2   18       en

   texto_length                                       cleaned_text
0          2003  worked while ago hospital aiding refugees/ille...
1          3265  Long time lurker here, feeling compelled now p...
2          6415  March , - Cataclysmic Shift and Future predict...
3          6357  honestly not like the word prophet because the...
4          4460  ___ . #### Nicaraguan Ambassador Carlos Eduard...
```

```
#Se eliminan las stopwords en inglés
stop_words = stopwords.words('english')
print(stop_words)
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an', 'and', 'any', 'are', 'aren', "aren't", 'a
```

```
#Se añaden algunas stopwords que no están en la lista
stop_words.extend(['like', 'also', 'really', 'thing', 'people', 'know', 'even', 'going', 'get', 'see', 'still', 'one', 'many


#Se realiza la lematización y se añade de forma manual aquellas que no se hayan realizado automáticamente
def lemma_words(text):
    lemmas = []
    doc = nlp(text)
    for token in doc:
        if ((not token.is_stop) and (not token.is_punct)) and (token.pos_ != 'PRON'):
            lemmas.append(token.lemma_)

    # Se eliminan las palabras demasiado cortas
    lemmas = [i for i in lemmas if len(i) > 1]

    # Se aplana por si hay espacios extraños
    lemmas = [word for line in lemmas for word in line.split()]

    # Se filtra stopwords
    lemmas = [word for word in lemmas if word not in stop_words]

    # Se convierte a una sola cadena
    lemmas = ' '.join(lemmas)

    # Se normaliza caracteres (tildes, eñes, etc.)
    lemmas = unidecode(lemmas)

    # Se convierte todo a minúsculas
    lemmas = lemmas.lower()

    # Sustituciones específicas relevantes para la migración venezolana y ortografía británica
    replacements = {
        r"\borganization\b": "organisation",
        r"\blabor\b": "labour",
        r"\bdefense\b": "defence",
        r"\brealize\b": "realise",
        r"\bbehavior\b": "behaviour",
        r"\bmigrants\b": "migrant",
        r"\bgreenwashing\b": "greenwash",
        r"\bfav\b": "favorite",
        r"\bachievement\b": "achieve",
        r"\bacceleration\b": "accelerate",
        r"\bdiscussion\b": "discuss",
        r"\bdevelopment\b": "develop"
    }
    for pattern, replacement in replacements.items():
        lemmas = re.sub(pattern, replacement, lemmas)

    # Se revisa de nuevo stopwords (opcional)
    lemmas = lemmas.split()
    lemmas = [word for word in lemmas if word not in stop_words]

    # Se devuelve como texto final
    lemmas = ' '.join(lemmas)
    return lemmas



#Se aplica la función de la lematización
df_analisis['lemmas'] = df_analisis['cleaned_text'].apply(lemma_words)


#Se revisan las filas vacías para ver si tengo que elimanr
print("Filas antes de dropna:", df_analisis.shape[0])
print("Filas vacías en lemmas:", (df_analisis['lemmas'] == '').sum())
```

```
Filas antes de dropna: 13087
Filas vacías en lemmas: 0
```

```
#Se imprime el dataset
df_analisis
```

	Título	Fecha	Subreddit	Autor	date	year	month	day	language	texto_length	cleaned_t
0	I see many posts here saying that people want ...	2025-02-19 11:33:02	TheHandmaidsTale	Poch1212	2025-02-19	2025	2	19	en	2003	worked w ago hos ai refugees/i
1	Collapse Comes Early for Canada: Prepare for U...	2025-02-19 04:57:29	CollapsePrep	verdasuno	2025-02-19	2025	2	19	en	3265	Long time lu here, fee compelled
2	Predictions and Prophecies	2025-02-19 01:04:18	prophets	RosalieJewel	2025-02-19	2025	2	19	en	6415	Marc Cataclys Shift and Fu pred
3	March 4, 2025 - Cataclysmic Shift and Future p...	2025-02-19 00:50:20	anonspropheticdream	RosalieJewel	2025-02-19	2025	2	19	en	6357	honestly not the v pro because t
4	February 18, 2025, 4:17 pm	Nicaragua (NI) - ...	2025-02-18 16:17:56	ABCWorldNews	AcademicAd8273	2025-02-18	2025	2	18	en	4460	___ . # Nicarag Ambassa Carlos Edua
...	...	...	...	...	...	...	...	...	...	...	
13082	Do Uncommitted/Jill Stein voters not realize t...	2024-09-14 20:58:21	millenials	zipzzo	2024-09-14	2024	9	14	en	4668	Can someb explain thought proc her
13083	He's a proven rapist, serial liar, and busines...	2024-09-14 12:55:30	PoliticsVermont	RamaSchneider	2024-09-14	2024	9	14	en	1179	>Donald Tr repeated ra claims al H
13084	Trump vows mass deportations from town rocked ...	2024-09-13 23:24:26	autotldr	autotldr	2024-09-13	2024	9	13	en	2632	This the b could m [origi redu
13085	Best longform profiles of the week	2024-09-13 22:46:32	longform	VegetableHousing139	2024-09-13	2024	9	13	en	17890	Hey gu back with s the longform
13086	More lies about Springfield, Ohio from trump.	2024-09-13 20:18:52	centrist	SpaceLaserPilot	2024-09-13	2024	9	13	en	2620	Today, du p confere Palos Ver

13087 rows × 12 columns

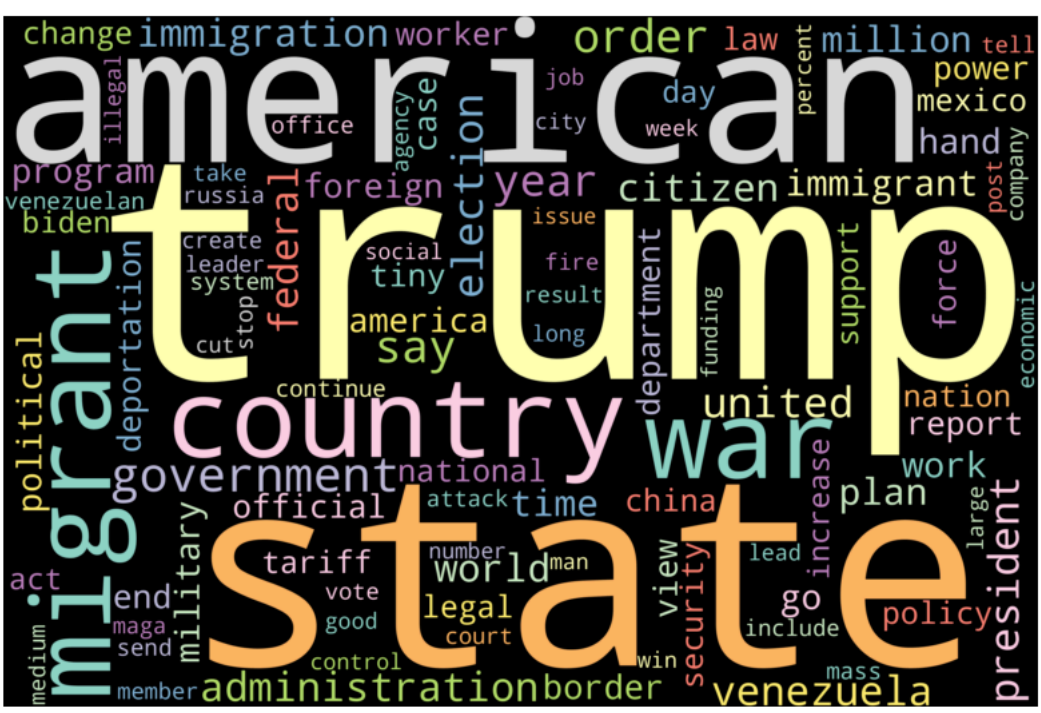
```
# Se visualiza el número total de comentarios que hay ahora tras la limpieza de los datos
numero_de_comentarios = df_analisis.shape[0]
print("Número total de comentarios:", numero_de_comentarios)
```

```
Número total de comentarios: 13087
```

```
#Se genera el csv de la columna lemas y el completo
df_analisis['lemmas'].to_csv('Lemmas_Venezuala.csv')
df_analisis.to_csv('Venezuela_limpia.csv')
```

```
#Se crea el Wordcloud con las palabras más destacadas del conjunto de datos
def plot_cloud(wordcloud):
    plt.figure(figsize=(10, 7))
```

```
    plt.imshow(wordcloud)
    plt.axis("off");


def plot_cloud(wordcloud):
    plt.figure(figsize=(10, 7))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()

# Se pinta el Wordcloud
wordcloud = WordCloud(width=3000, height=2000, random_state=1,
                      max_words=100, background_color='black',
                      colormap='Set3', collocations=False,
                      stopwords=STOPWORDS).generate(' '.join(df_analisis['lemmas']))
plot_cloud(wordcloud)
```




```
# 2. Análisis Descriptivo de N-Gramas

#Se lee el csv con los datos limpios obtenidos anteriormente y se filtra por la columna "lemmas" y se imprimen
df_analisis = pd.read_csv('Venezuela_limpia.csv')
dft= df_analisis['lemmas']
dft = [x for x in dft if str(x) != 'nan']
print(dft)
```

```
['work ago hospital aid refugee illegal migrant etc talk knowledge believe simply leave country apply asylum start work
```

```
#Se calcula el Valor de TF-IDF de los unigramas y se imprimen
tfIdfVectorizer=TfidfVectorizer(use_idf=True, ngram_range=(1,1))
tfIdf = tfIdfVectorizer.fit_transform(dft)
names=tfIdfVectorizer.get_feature_names_out()
freqs = tfIdf.sum(axis=0).A1
result= dict(zip(names, freqs))
print(result)
```

```
{'aaf': 2.086031264657588, 'aaron': 19.1996631762703, 'abandon': 11.55172128070215, 'abandoned': 9.038046918249135, 'aba
```

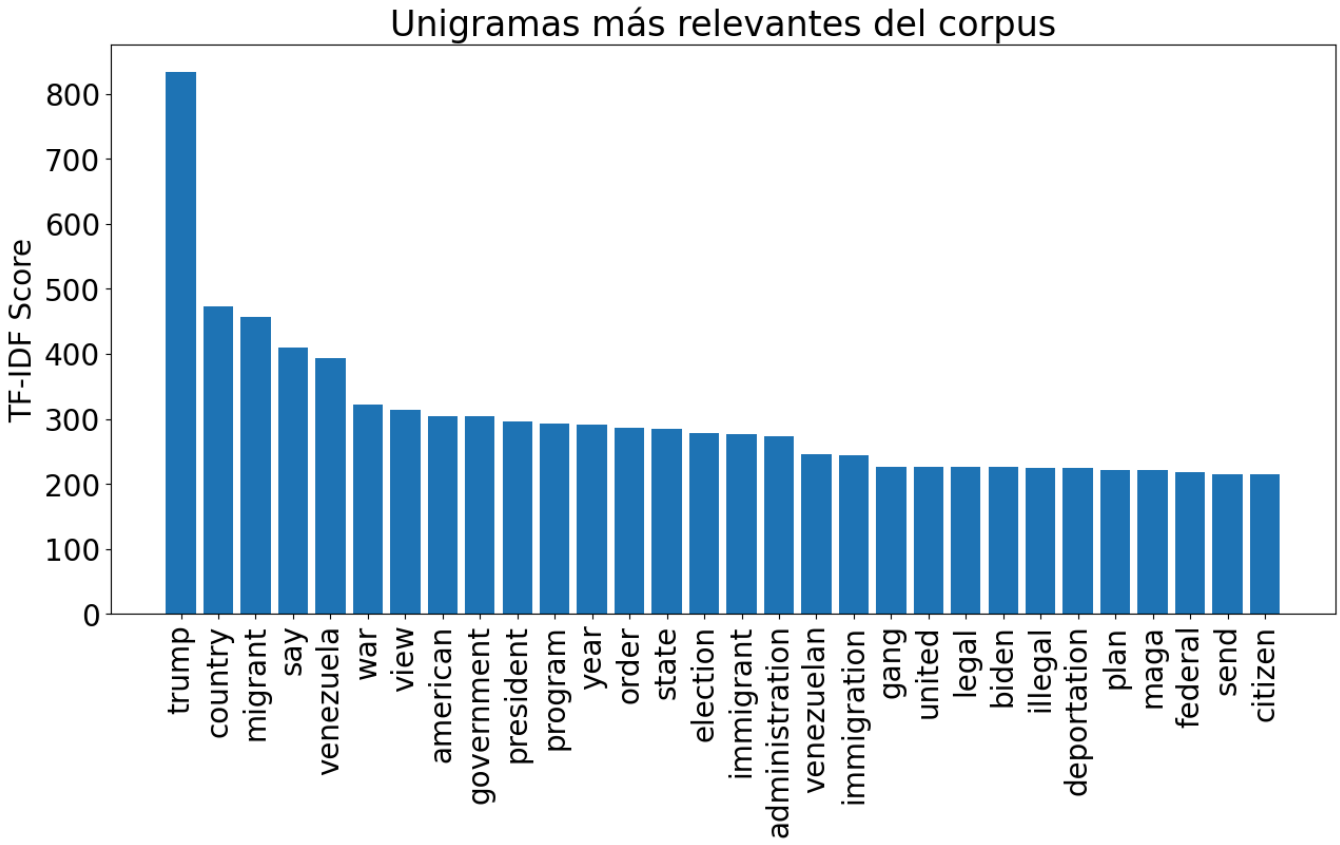
```
#Se muestran los 30 unigramas con mayor valor TF-IDF
from operator import itemgetter
i = 0
results_sorted=sorted(result.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 31:
      break
    print(key, value)
```

```
trump 833.2672894012513
country 473.2925683694078
```

```
migrant 456.92181348648717
say 410.35222106152844
venezuela 392.9513829493716
war 322.081548453832
view 314.65202093736286
american 304.6842599540528
government 304.0827765914265
president 296.6339993892294
program 292.52470849999247
year 291.56553700069463
order 286.9823300344318
state 284.20108316112004
election 277.9183591286018
immigrant 275.89429487317886
administration 273.0571034761276
venezuelan 245.4059044995032
immigration 243.3658156107928
gang 227.04270963550232
united 226.86932795487775
legal 226.68776547678172
biden 226.02207221551703
illegal 225.30130049979275
deportation 224.6194836964085
plan 222.08087280383802
maga 220.74273281434742
federal 217.57070672953108
send 215.27206391958683
citizen 214.67380125166267
```

```
#Se pintan los 30 unigramas con más TF-IDF
df_results=pd.DataFrame.from_dict(results_sorted).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results[0],df_results[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Unigramas más relevantes del corpus')
```

```
Text(0.5, 1.0, 'Unigramas más relevantes del corpus')
```



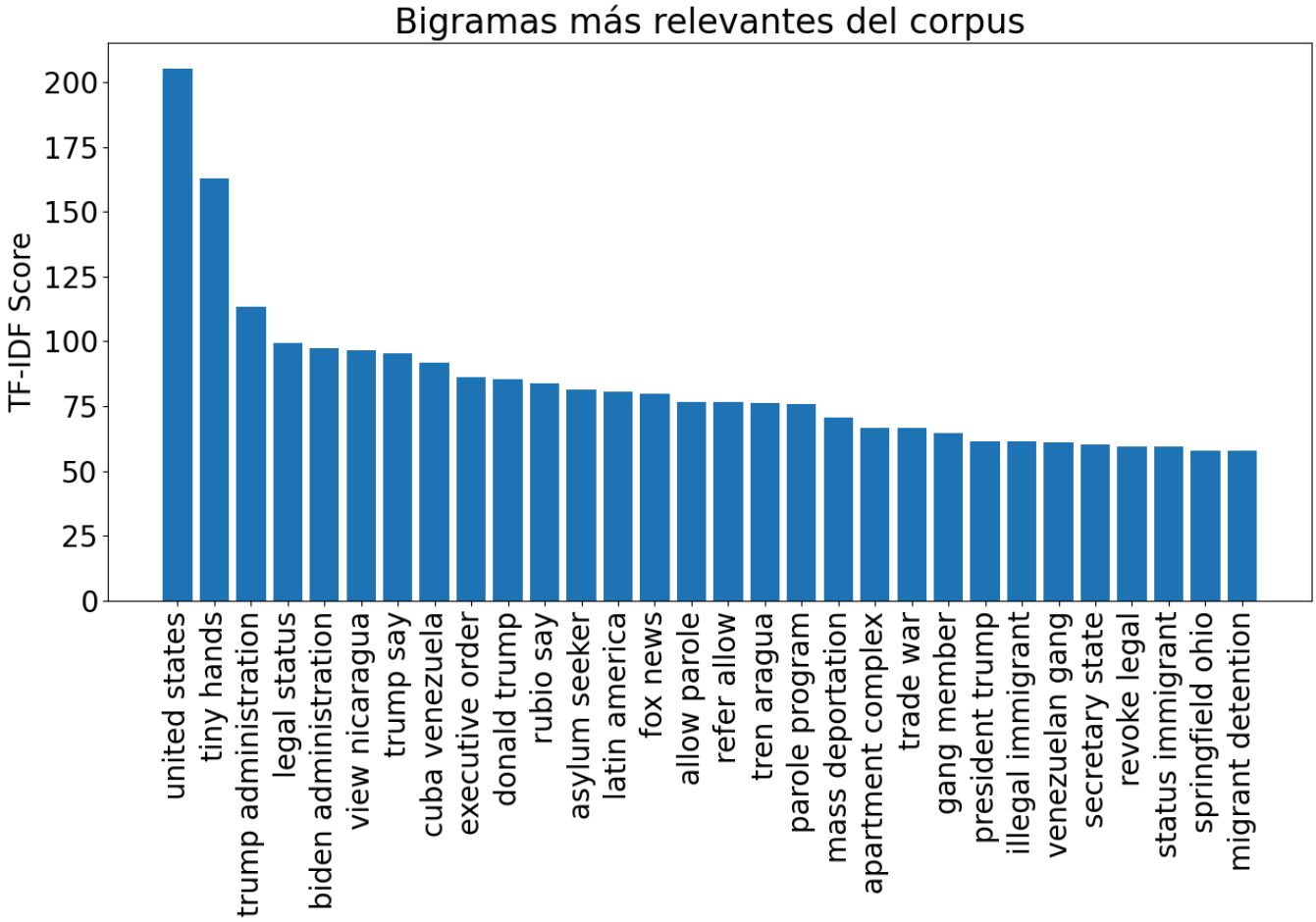

```
#Se realiza la misma operación con los bigramas y trigramas

#Se calcula el Valor de TF-IDF de los bigramas
tfIdfVectorizer_bi=TfidfVectorizer(use_idf=True, ngram_range=(2,2))
tfIdf_bi = tfIdfVectorizer_bi.fit_transform(dft)
names_bi=tfIdfVectorizer_bi.get_feature_names_out()
```

```
freqs_bi = tfIdf_bi.sum(axis=0).A1
result_bi= dict(zip(names_bi, freqs_bi))



#Se muestran los 30 bigramas con mayor valor TF-IDF
from operator import itemgetter
i = 0
results_sorted=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 31:
      break
    print(key, value)
```

```
united states 205.01972240209133
tiny hands 162.97247694243288
trump administration 113.36255596699984
legal status 99.49776506477029
biden administration 97.17188057616056
view nicaragua 96.63395522502
trump say 95.53477068578523
cuba venezuela 91.7126530921621
executive order 85.99571213898318
donald trump 85.25428486537224
rubio say 83.81528940636791
asylum seeker 81.32344888698081
latin america 80.49475437698365
fox news 79.61969848522553
allow parole 76.76400617035398
refer allow 76.76400617035398
tren aragua 76.19946884087665
parole program 75.86339936989627
mass deportation 70.5325299536829
apartment complex 66.69657687061763
trade war 66.54403323639696
gang member 64.55597700589932
president trump 61.289606003191864
illegal immigrant 61.23377656650671
venezuelan gang 60.931930724282495
secretary state 60.35448695654123
revoke legal 59.2226084338374
status immigrant 59.2226084338374
springfield ohio 57.97586597434818
migrant detention 57.93857654523915
```

```
#Se pintan los 30 bigramas con más TF-IDF
results_sorted_bi=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
df_results_bi=pd.DataFrame.from_dict(results_sorted_bi).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_bi[0],df_results_bi[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Bigramas más relevantes del corpus')
```

```
Text(0.5, 1.0, 'Bigramas más relevantes del corpus')
```



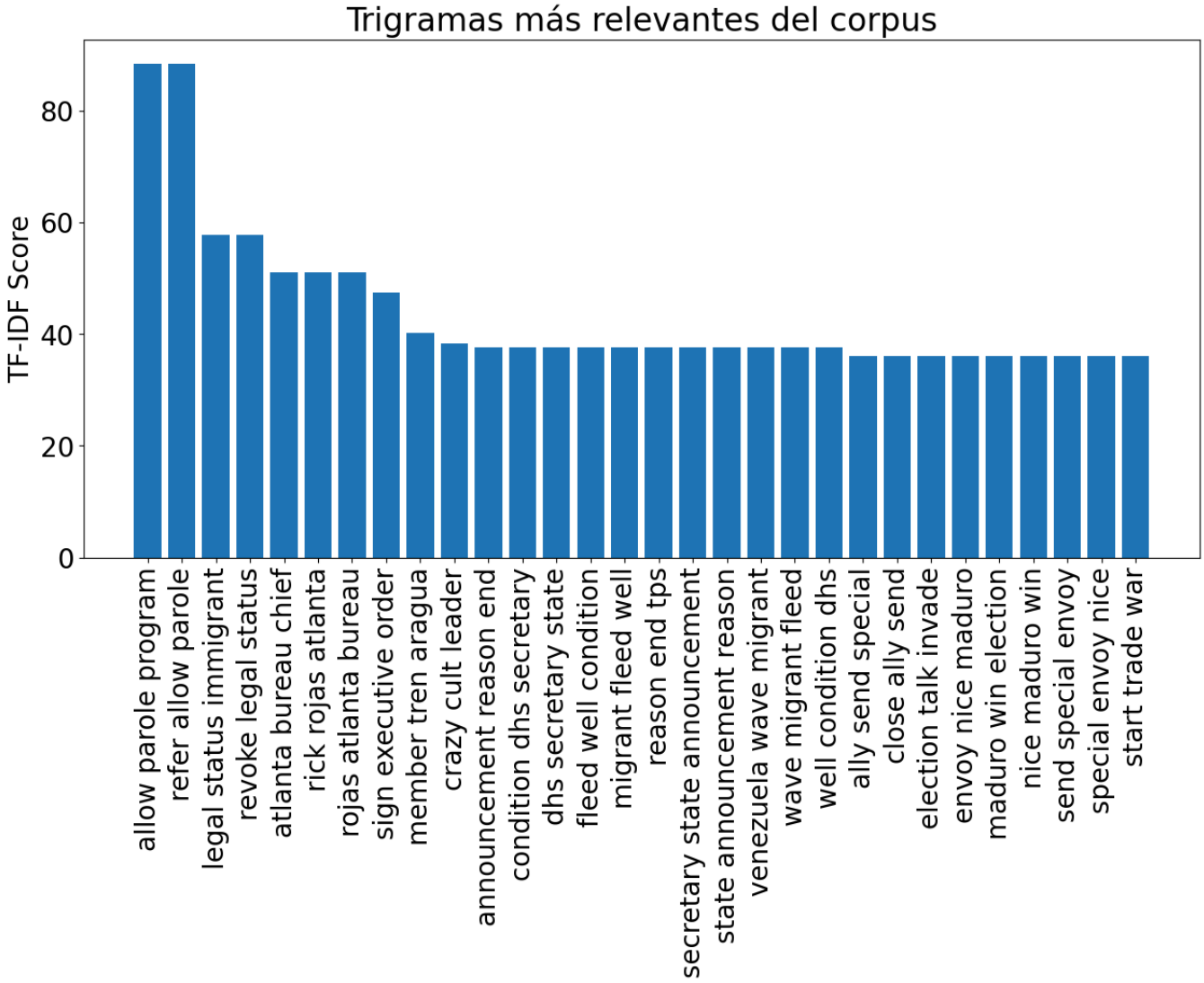

```
#Trigramas
#Se calcula el Valor de TF-IDF de los trigramas
tfIdfVectorizer_tri=TfidfVectorizer(use_idf=True, ngram_range=(3,3))
tfIdf_tri = tfIdfVectorizer_tri.fit_transform(dft)
names_tri= tfIdfVectorizer_tri.get_feature_names_out()
freqs_tri = tfIdf_tri.sum(axis=0).A1
result_tri = dict(zip(names_tri, freqs_tri))


#Se muestran los 30 trigramas con mayor valor TF-IDF
from operator import itemgetter
i = 0
results_sorted=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 31:
      break
    print(key, value)
```

```
allow parole program 88.38834764831854
refer allow parole 88.38834764831854
legal status immigrant 57.861184090078154
revoke legal status 57.861184090078154
atlanta bureau chief 51.143831259375986
rick rojas atlanta 51.143831259375986
rojas atlanta bureau 51.143831259375986
sign executive order 47.481853992008986
member tren aragua 40.201291715270294
crazy cult leader 38.371058990949805
announcement reason end 37.68891807222048
condition dhs secretary 37.68891807222048
dhs secretary state 37.68891807222048
fleed well condition 37.68891807222048
migrant fleed well 37.68891807222048
reason end tps 37.68891807222048
secretary state announcement 37.68891807222048
state announcement reason 37.68891807222048
venezuela wave migrant 37.68891807222048
wave migrant fleed 37.68891807222048
well condition dhs 37.68891807222048
```

```
ally send special 36.084391824351506
close ally send 36.084391824351506
election talk invade 36.084391824351506
envoy nice maduro 36.084391824351506
maduro win election 36.084391824351506
nice maduro win 36.084391824351506
send special envoy 36.084391824351506
special envoy nice 36.084391824351506
start trade war 36.084391824351506
```

```
#Se pintan los 30 trigramas con más TF-IDF
results_sorted_tri=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
df_results_tri=pd.DataFrame.from_dict(results_sorted_tri).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_tri[0],df_results_tri[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Trigramas más relevantes del corpus')
```

```
Text(0.5, 1.0, 'Trigramas más relevantes del corpus')
```




```
# 3. Modelado de Tópicos
```

```
#Se carga el csv obtenido ateriormente con los datos limpios para realizar el modelado de tópicos
df_modelado = pd.read_csv('Venezuela_limpia.csv')
print(df_modelado.head())
```

```
   Unnamed: 0                                             Título  \
0           0  I see many posts here saying that people want ...
1           1  Collapse Comes Early for Canada: Prepare for U...
2           2                          Predictions and Prophecies
3           3  March 4, 2025 – Cataclysmic Shift and Future p...
4           4  February 18, 2025, 4:17 pm | Nicaragua (NI) – ...
```

```
                  Fecha            Subreddit           Autor        date  year  \
0  2025-02-19 11:33:02     TheHandmaidsTale        Poch1212  2025-02-19  2025
1  2025-02-19 04:57:29         CollapsePrep       verdasuno  2025-02-19  2025
2  2025-02-19 01:04:18             prophets    RosalieJewel  2025-02-19  2025
3  2025-02-19 00:50:20  anonspropheticdream    RosalieJewel  2025-02-19  2025
4  2025-02-18 16:17:56         ABCWorldNews  AcademicAd8273  2025-02-18  2025

   month  day language  texto_length  \
0      2   19       en          2003
1      2   19       en          3265
2      2   19       en          6415
3      2   19       en          6357
4      2   18       en          4460

                                      cleaned_text  \
0  worked while ago hospital aiding refugees/ille...
1  Long time lurker here, feeling compelled now p...
2  March , - Cataclysmic Shift and Future predict...
3  honestly not like the word prophet because the...
4  ___ . #### Nicaraguan Ambassador Carlos Eduard...

                                             lemmas
0  work ago hospital aid refugee illegal migrant ...
1  long time lurker feel compel post recent devel...
2  march cataclysmic shift future prediction hone...
3  honestly word prophet negative connotation ass...
4  nicaraguan ambassador carlos eduardo diaz more...
```

```
#Se procede a realizar la tokenización
def tokenize(text):
    text = str(text)
    tokens = text.split()
    return tokens


df_modelado['tokens'] = df_modelado['lemmas'].apply(tokenize) #Se añade una nueva columna llamada 'tokens' al DataFrame apli
print(df_modelado.head()) #Se visualizan las primeras filas
```

```
   Unnamed: 0                                             Título  \
0           0  I see many posts here saying that people want ...
1           1  Collapse Comes Early for Canada: Prepare for U...
2           2                          Predictions and Prophecies
3           3  March 4, 2025 - Cataclysmic Shift and Future p...
4           4  February 18, 2025, 4:17 pm | Nicaragua (NI) - ...

                  Fecha            Subreddit           Autor        date  year  \
0  2025-02-19 11:33:02     TheHandmaidsTale        Poch1212  2025-02-19  2025
1  2025-02-19 04:57:29         CollapsePrep       verdasuno  2025-02-19  2025
2  2025-02-19 01:04:18             prophets    RosalieJewel  2025-02-19  2025
3  2025-02-19 00:50:20  anonspropheticdream    RosalieJewel  2025-02-19  2025
4  2025-02-18 16:17:56         ABCWorldNews  AcademicAd8273  2025-02-18  2025

   month  day language  texto_length  \
0      2   19       en          2003
1      2   19       en          3265
2      2   19       en          6415
3      2   19       en          6357
4      2   18       en          4460

                                      cleaned_text  \
0  worked while ago hospital aiding refugees/ille...
1  Long time lurker here, feeling compelled now p...
2  March , - Cataclysmic Shift and Future predict...
3  honestly not like the word prophet because the...
4  ___ . #### Nicaraguan Ambassador Carlos Eduard...

                                             lemmas  \
0  work ago hospital aid refugee illegal migrant ...
1  long time lurker feel compel post recent devel...
2  march cataclysmic shift future prediction hone...
3  honestly word prophet negative connotation ass...
4  nicaraguan ambassador carlos eduardo diaz more...

                                              tokens
0  [work, ago, hospital, aid, refugee, illegal, m...
1  [long, time, lurker, feel, compel, post, recen...
2  [march, cataclysmic, shift, future, prediction...
3  [honestly, word, prophet, negative, connotatio...
4  [nicaraguan, ambassador, carlos, eduardo, diaz...
```

```
#Se importa una semilla para la reproducibilidad de los resultados
seed(24)
#Además, se crea un diccionario a partir de los tokens encontrados
id2word = Dictionary(df_modelado['tokens'])
```

```
#Se filtran los extremos continuando con el proceso de limpieza
id2word.filter_extremes(no_below=2, no_above=.95)


#Se crea el objeto del corpus
corpus = [id2word.doc2bow(d) for d in df_modelado['tokens']]


# Se procede a encontrar el valor óptimo de k y para obtener los valores de coherencia
def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=2):
    coherence_values_topic = []
    model_list_topic = []
    for num_topics in range(start, limit, step):
        model = LdaMulticore(corpus=corpus, num_topics=num_topics, id2word=id2word, passes=10)
        model_list_topic.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c_v')
        coherence_values_topic.append(coherencemodel.get_coherence())

    return model_list_topic, coherence_values_topic


# Se aplica la función definida en el paso anterior
model_list_topic, coherence_values_topic = compute_coherence_values(
    dictionary=id2word,
    corpus=corpus,
    texts=df_modelado['tokens'],
    start=2,
    limit=15,
    step=1
)


#Se crea un csv con los valores coherencia del rango de tópicos establecido en el paso anterior
#coherence_values_topic_df = pd.DataFrame(coherence_values_topic)
#coherence_values_topic_df.to_csv('coherence_values.csv', index=False)
coherence_values_topic_df= pd.read_csv('coherence_values.csv', index_col=False)
coherence_values_topic_df
```

	0
0	0.290365
1	0.286603
2	0.307631
3	0.368854
4	0.345713
5	0.360271
6	0.348501
7	0.402880
8	0.398572
9	0.409206
10	0.425380
11	0.396765
12	0.398496

```
#Se crea una lista con los valores de las columnas indicados
coherence_values_topic_df=coherence_values_topic_df.iloc[:, 0].tolist()


# Genera la lista de X (por ejemplo, si coherence_values_topic_df tiene 13 elementos)
x_values = list(range(2, 2 + len(coherence_values_topic_df)))

# Selecciona el número de tópicos deseado
selected_topic = 4
selected_index = x_values.index(selected_topic)
x_selected = x_values[selected_index]
y_selected = coherence_values_topic_df[selected_index]

# Dibuja la gráfica
plt.figure(figsize=(10, 7))
estrella = mpath.Path.unit_regular_star(8)
circulo = mpath.Path.unit_circle()
verts = np.concatenate([circulo.vertices, estrella.vertices[::-1, ...]])
codes = np.concatenate([circulo.codes, estrella.codes])
cut_star = mpath.Path(verts, codes)
```

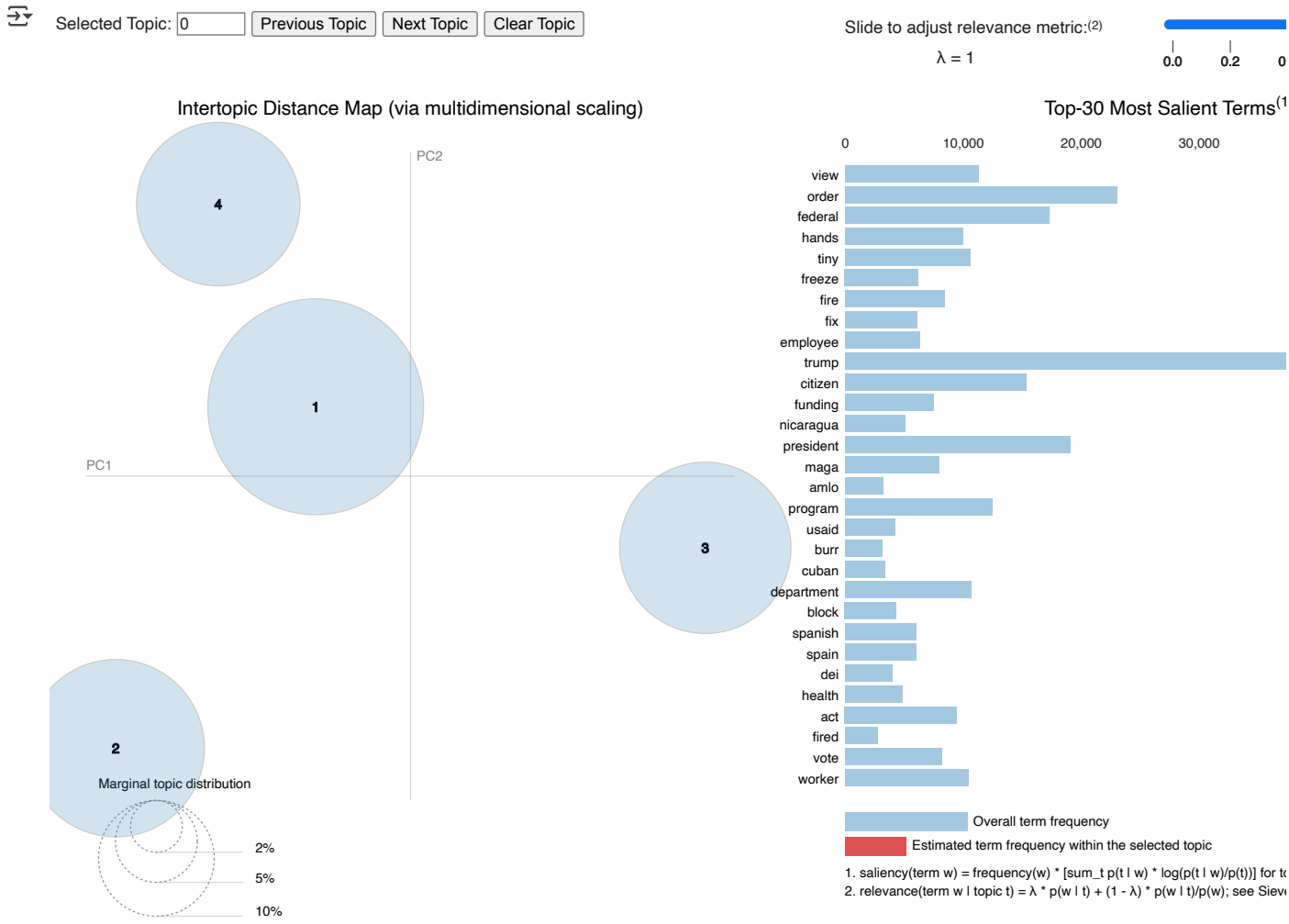
```
plt.plot(x_values, coherence_values_topic_df, '--r', marker="o", markersize=10, fillstyle='none')
plt.axvline(x=x_selected, color='b', linestyle='--')
plt.plot(x_selected, y_selected, '--r', marker=cut_star, markersize=18)
plt.xlabel('Número de tópicos')
plt.ylabel('Índice de coherencia')
#plt.show()


#Se genera el modelo LDA con el número de k elegido que en este caso k=4
k=4
model_k4 = gensim.models.LdaMulticore(corpus=corpus,
                                      id2word=id2word,
                                      num_topics=k,
                                      passes=10,
                                      random_state=23)


#Se procede a guardar y cargar todos los resultados obtenidos del modelo
#model_k4.save("model_4_topics.model")



# Se carga del modelo LDA previamente guardado (este paso se realiza en caso de ya haber runeado el código y tener guardado
model_k4_load = LdaMulticore.load("model_4_topics.model")


#Se calcula el coherence value
coherence_model_k4 = CoherenceModel(model=model_k4_load, texts=df_modelado['tokens'],
                                    dictionary=id2word, coherence='c_v')
coherence_model_k4 = coherence_model_k4.get_coherence()
print('\nCoherence Score: ', coherence_model_k4)
```

```
Coherence Score:  0.3395894453401642
```

```
#Se observan las 10 palaras más utilizadas en cada uno de los 5 tópicos
print(model_k4_load.print_topics())
doc_lda = model_k4_load[corpus]
#Se filtra por palabras
words = [re.findall(r'"([^"]*)"',t[1]) for t in model_k4_load.print_topics()]
#Se crean los tópicos
topics = [' '.join(t[0:10]) for t in words]
for id, t in enumerate(topics):
    print(f"------ Topic {id} ------")
    print(t, end="\n\n")
```

```
[(0, '0.007*"migrant" + 0.006*"country" + 0.006*"trump" + 0.006*"citizen" + 0.006*"election" + 0.006*"government" + 0.00
------ Topic 0 ------
migrant country trump citizen election government say administration tiny american

------ Topic 1 ------
order federal trump program fire department freeze funding employee year

------ Topic 2 ------
trump view president war migrant say government country united venezuela

------ Topic 3 ------
trump state war political fix maga american world country vote
```

```
#Se pinta la distancia intertópica con pyLDAvis
pyLDAvis.enable_notebook()
gensimvis.prepare(model_k4_load, corpus, id2word)
```




```
#Se añade una columna de LDA features al modelo cargado para que nos de la importancia de cada tópico
def document_to_lda_features(model_k4_load,document):
  topic_importance=np.array(model_k4_load.get_document_topics(document, minimum_probability=0))
  return topic_importance[:,1]


df_modelado['lda_features']=list(map(lambda doc: document_to_lda_features(model_k4_load,doc), corpus))


#Se realiza la tokenización
def topic_important(item_score):
    score=np.argmax(item_score, axis=0)
    return score


#Se aplica lo anterior y se visualiza la tabla obtenida
df_modelado['topic_dominant'] = df_modelado['lda_features'].apply(topic_important)
df_modelado
```

	Unnamed: 0	Título	Fecha	Subreddit	Autor	date	year	month	day	language	texto_length
0	0	I see many posts here saying that people want ...	2025-02-19 11:33:02	TheHandmaidsTale	Poch1212	2025-02-19	2025	2	19	en	2003
1	1	Collapse Comes Early for Canada: Prepare for U...	2025-02-19 04:57:29	CollapsePrep	verdasuno	2025-02-19	2025	2	19	en	3265
2	2	Predictions and Prophecies	2025-02-19 01:04:18	prophets	RosalieJewel	2025-02-19	2025	2	19	en	6415
3	3	March 4, 2025 - Cataclysmic Shift and Future p...	2025-02-19 00:50:20	anonspropheticdream	RosalieJewel	2025-02-19	2025	2	19	en	6357
4	4	February 18, 2025, 4:17 pm	Nicaragua (NI) - ...	2025-02-18 16:17:56	ABCWorldNews	AcademicAd8273	2025-02-18	2025	2	18	en	4460
...	...	...	...	...	...	...	...	...	...	...	...	...
13082	13082	Do Uncommitted/Jill Stein voters not realize t...	2024-09-14 20:58:21	millenials	zipzzo	2024-09-14	2024	9	14	en	4668
13083	13083	He's a proven rapist, serial liar, and busines...	2024-09-14 12:55:30	PoliticsVermont	RamaSchneider	2024-09-14	2024	9	14	en	1179
13084	13084	Trump vows mass deportations from town rocked ...	2024-09-13 23:24:26	autotldr	autotldr	2024-09-13	2024	9	13	en	2632
13085	13085	Best longform profiles of the week	2024-09-13 22:46:32	longform	VegetableHousing139	2024-09-13	2024	9	13	en	17890
13086	13086	More lies about Springfield, Ohio from trump.	2024-09-13 20:18:52	centrist	SpaceLaserPilot	2024-09-13	2024	9	13	en	2620

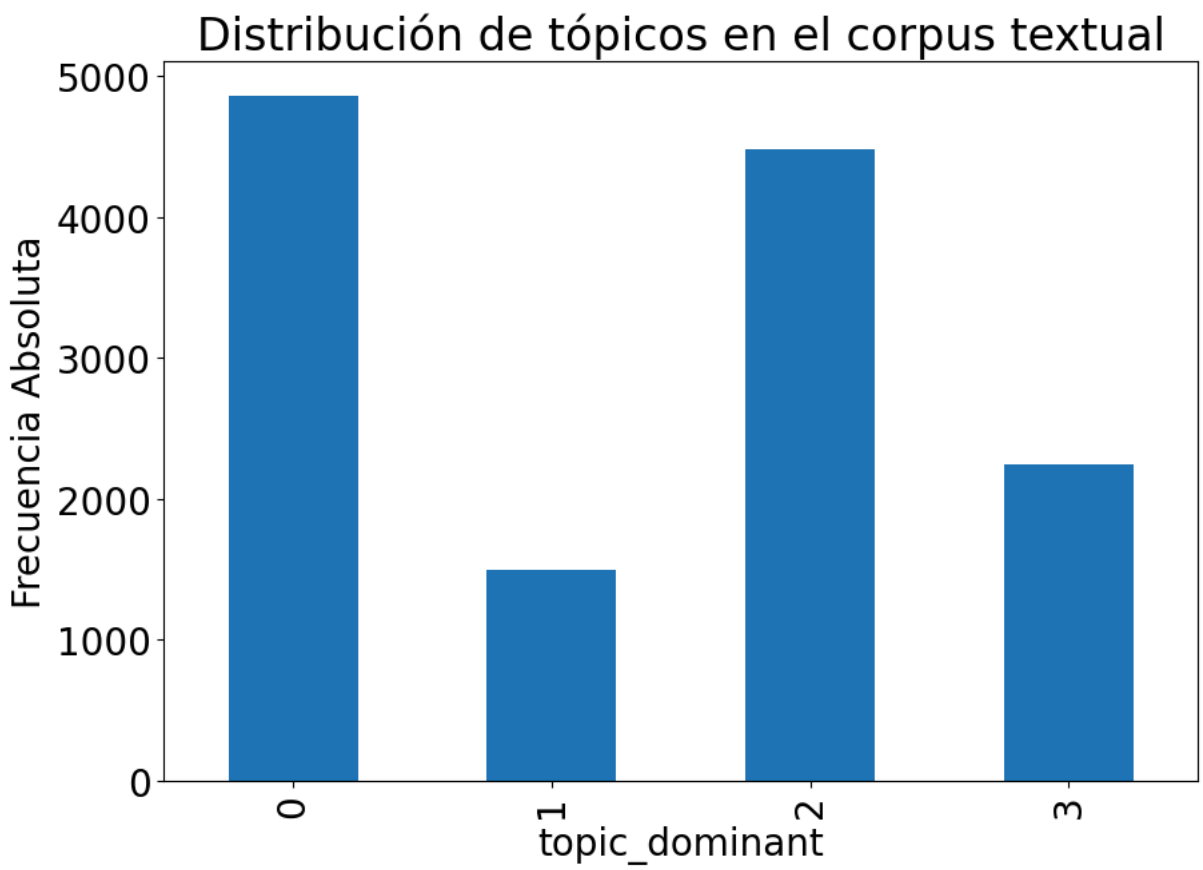
13087 rows × 16 columns

```
#Se genera un csv con los resultados para guardarlos ya que serán empleados en el análisis de sentimientos posteriormente
df_modelado.to_csv('topic_model_results.csv', index=False)


#Se saca el número de publicaciones por tópico
df_modelado["topic_dominant"].value_counts()
```

topic_dominant	count
0	4863
2	4484
3	2246
1	1494

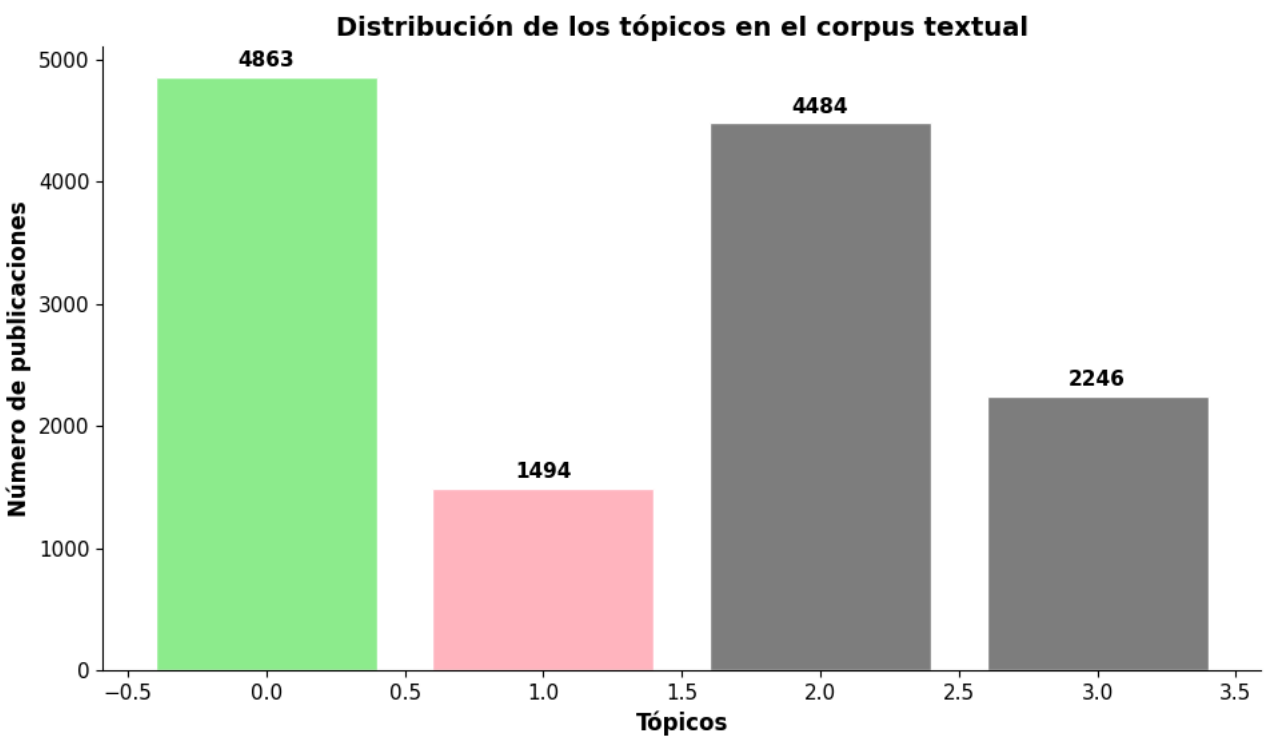
dtype: int64

```
#Se dibuja la distribución de los tópicos en un histograma
plt.figure(figsize=(10,7))
ax=df_modelado["topic_dominant"].value_counts().sort_index().plot(kind='bar')
plt.ylabel('Frecuencia Absoluta')
plt.title('Distribución de tópicos en el corpus textual')
plt.show()
```




```
# Se obtienen las cuentas de frecuencia y ordénalas por índice
# Obtuve esta visualización también de Flourish ()
topic_counts = df_modelado["topic_dominant"].value_counts().sort_index()

# Nombres de los tópicos (0, 1, 2, ..., N)
topics = topic_counts.index
counts = topic_counts.values

# Colores para replicar el estilo: primero verde, intermedios gris, último rosa
colors = ['lightgreen'] + ['lightpink'] + ['grey'] + ['grey']

# Se crea la figura y el eje
fig, ax = plt.subplots(figsize=(10, 6))

# Se dibujan las barras
bars = ax.bar(topics, counts, color=colors, edgecolor='white')

# Se añaden las etiquetas de frecuencia encima de cada barra
for bar, count in zip(bars, counts):
    height = bar.get_height()
    ax.text(bar.get_x() + bar.get_width()/2, height + 50,  # Ajusta "+50" según el espacio
            f'{count}', ha='center', va='bottom', fontsize=11, fontweight='bold')

# Se etiquetan de los ejes y título
ax.set_xlabel('Tópicos', fontsize=12, fontweight='bold')
```

```
ax.set_ylabel('Número de publicaciones', fontsize=12, fontweight='bold')
ax.set_title('Distribución de los tópicos en el corpus textual', fontsize=14, fontweight='bold')

# Se personaliza el estilo: sin bordes superiores/derechos, ticks más grandes
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.tick_params(axis='both', which='major', labelsize=11)

plt.tight_layout()
plt.show()
```



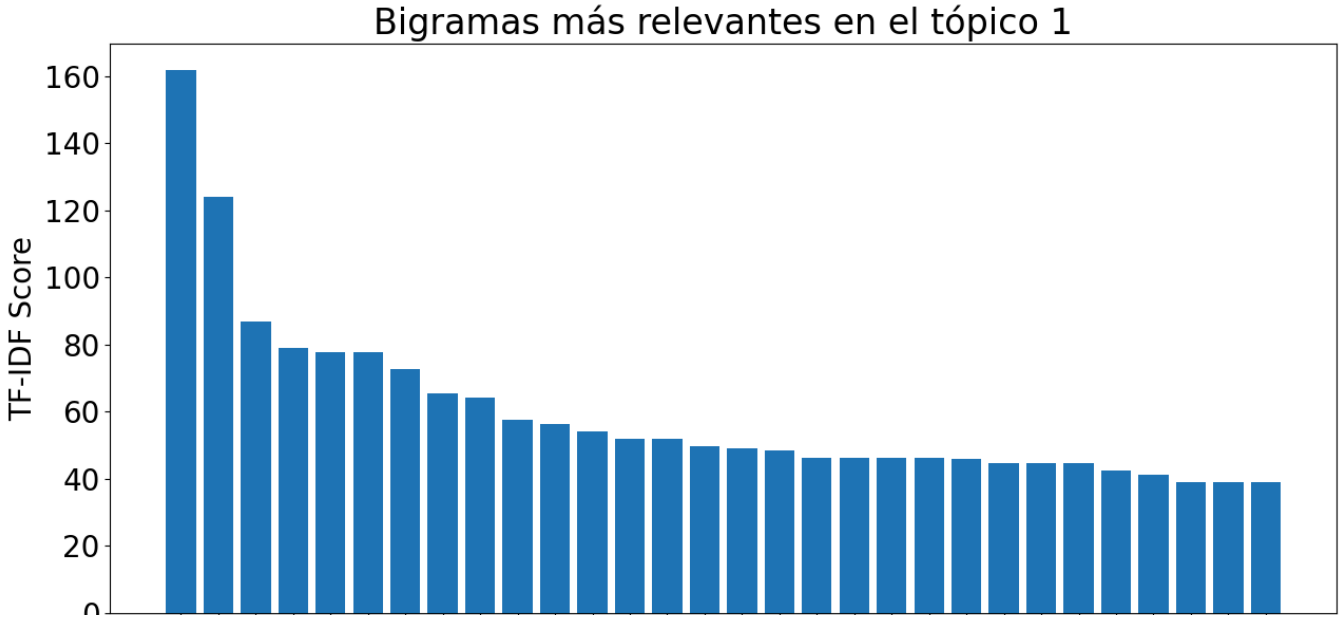

```
#Se visualizan los bigramas de cada uno de los tópicos con mayor TF-IDF

#Tópico 1
topic_1 = df_modelado[df_modelado['topic_dominant']==0]
dft=topic_1['lemmas']
dft= [x for x in dft if str(x) != 'nan']
tfIdfVectorizer_bi=TfidfVectorizer(use_idf=True, ngram_range=(2,2))
tfIdf_bi = tfIdfVectorizer_bi.fit_transform(dft)
names_bi=tfIdfVectorizer_bi.get_feature_names_out()
freqs_bi = tfIdf_bi.sum(axis=0).A1
result_bi= dict(zip(names_bi, freqs_bi))
from operator import itemgetter
i = 0
results_sorted=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
tiny hands 161.72424302021196
united states 123.9957433451367
trump administration 86.81426346209874
rubio say 78.8499639593105
allow parole 77.7154254309019
refer allow 77.7154254309019
parole program 72.67259310906267
tren aragua 65.49414931014681
apartment complex 64.27092669110067
asylum seeker 57.50375601440022
```

```
#Se realiza el histograma
results_sorted_bi=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
df_results_bi=pd.DataFrame.from_dict(results_sorted_bi).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_bi[0],df_results_bi[1])
plt.xticks(rotation=90)
```

```
plt.ylabel('TF-IDF Score')
plt.title('Bigramas más relevantes en el tópico 1')
```

```
Text(0.5, 1.0, 'Bigramas más relevantes en el tópico 1')
```

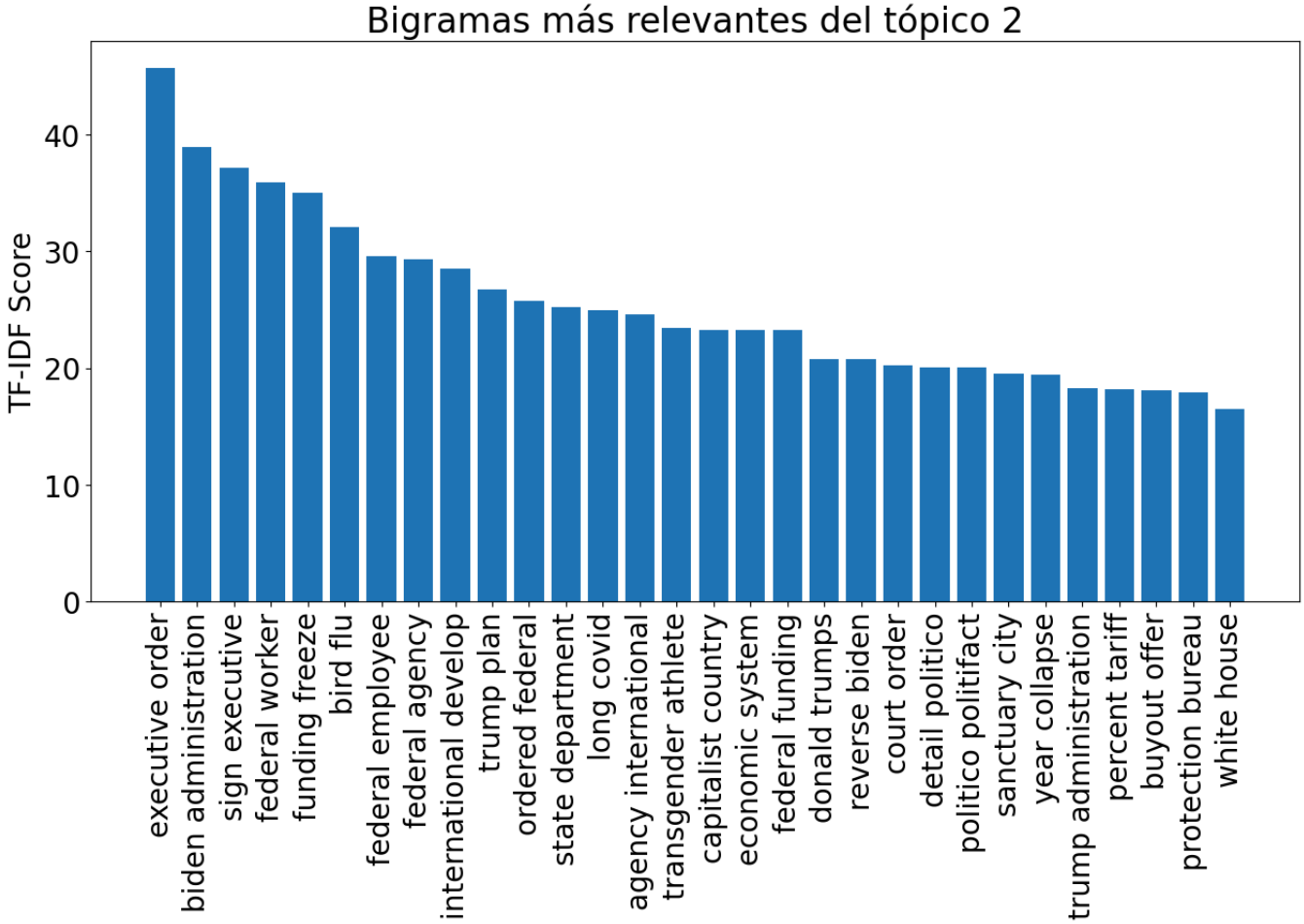



```
#Tópico 2
topic_2 = df_modelado[df_modelado['topic_dominant']==1]
dft=topic_2['lemmas']
tfIdfVectorizer_bi=TfidfVectorizer(use_idf=True, ngram_range=(2,2))
tfIdf_bi = tfIdfVectorizer_bi.fit_transform(dft)
names_bi=tfIdfVectorizer_bi.get_feature_names_out()
freqs_bi = tfIdf_bi.sum(axis=0).A1
result_bi= dict(zip(names_bi, freqs_bi))
from operator import itemgetter
i = 0
results_sorted=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
executive order 45.70361098586845
biden administration 38.90589434504286
sign executive 37.19716993323079
federal worker 35.933550815080594
funding freeze 35.023662191405315
bird flu 32.076115254135466
federal employee 29.554163828159396
federal agency 29.293466044565942
international develop 28.539192438953812
trump plan 26.72262552265968
```

```
#Se realiza el histograma
results_sorted_bi=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
df_results_bi=pd.DataFrame.from_dict(results_sorted_bi).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_bi[0],df_results_bi[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Bigramas más relevantes del tópico 2')
```

```
Text(0.5, 1.0, 'Bigramas más relevantes del tópico 2')
```



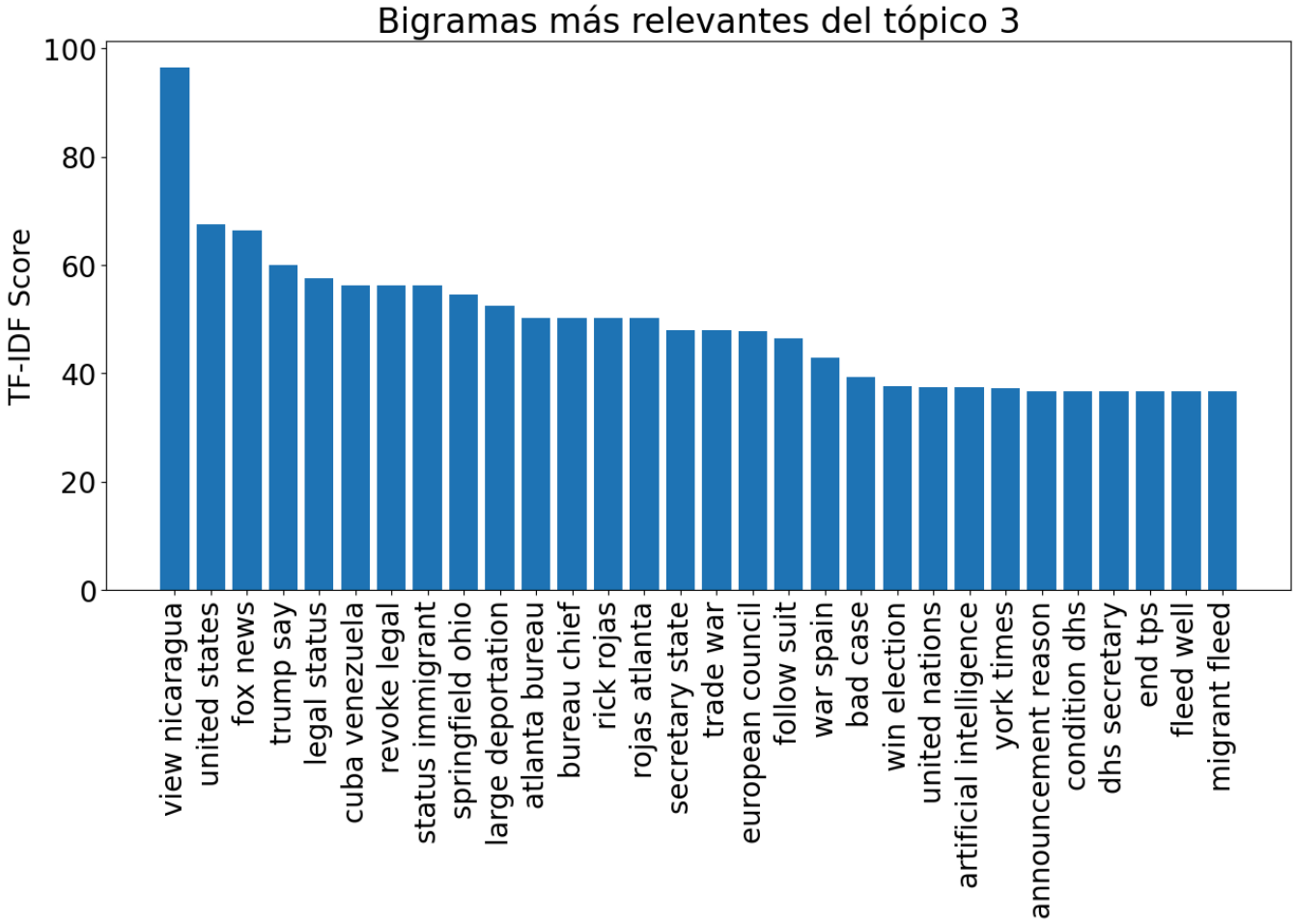

```
#Tópico 3
topic_3 = df_modelado[df_modelado['topic_dominant']==2]
dft=topic_3['lemmas']
dft= [x for x in dft if str(x) != 'nan']
tfIdfVectorizer_bi=TfidfVectorizer(use_idf=True, ngram_range=(2,2))
tfIdf_bi = tfIdfVectorizer_bi.fit_transform(dft)
names_bi=tfIdfVectorizer_bi.get_feature_names_out()
freqs_bi = tfIdf_bi.sum(axis=0).A1
result_bi= dict(zip(names_bi, freqs_bi))
from operator import itemgetter
i = 0
results_sorted=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
view nicaragua 96.48912121313107
united states 67.53473851776828
fox news 66.3228287914834
trump say 59.95395661065327
legal status 57.56985552170857
cuba venezuela 56.256429496690096
revoke legal 56.25631761050325
status immigrant 56.25631761050325
springfield ohio 54.53054195119957
large deportation 52.508795917387886
```

```
#Se realiza el histograma
results_sorted_bi=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
df_results_bi=pd.DataFrame.from_dict(results_sorted_bi).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_bi[0],df_results_bi[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Bigramas más relevantes del tópico 3')
```

```
plt.title('Bigramas más relevantes del tópico 3')
```

```
Text(0.5, 1.0, 'Bigramas más relevantes del tópico 3')
```



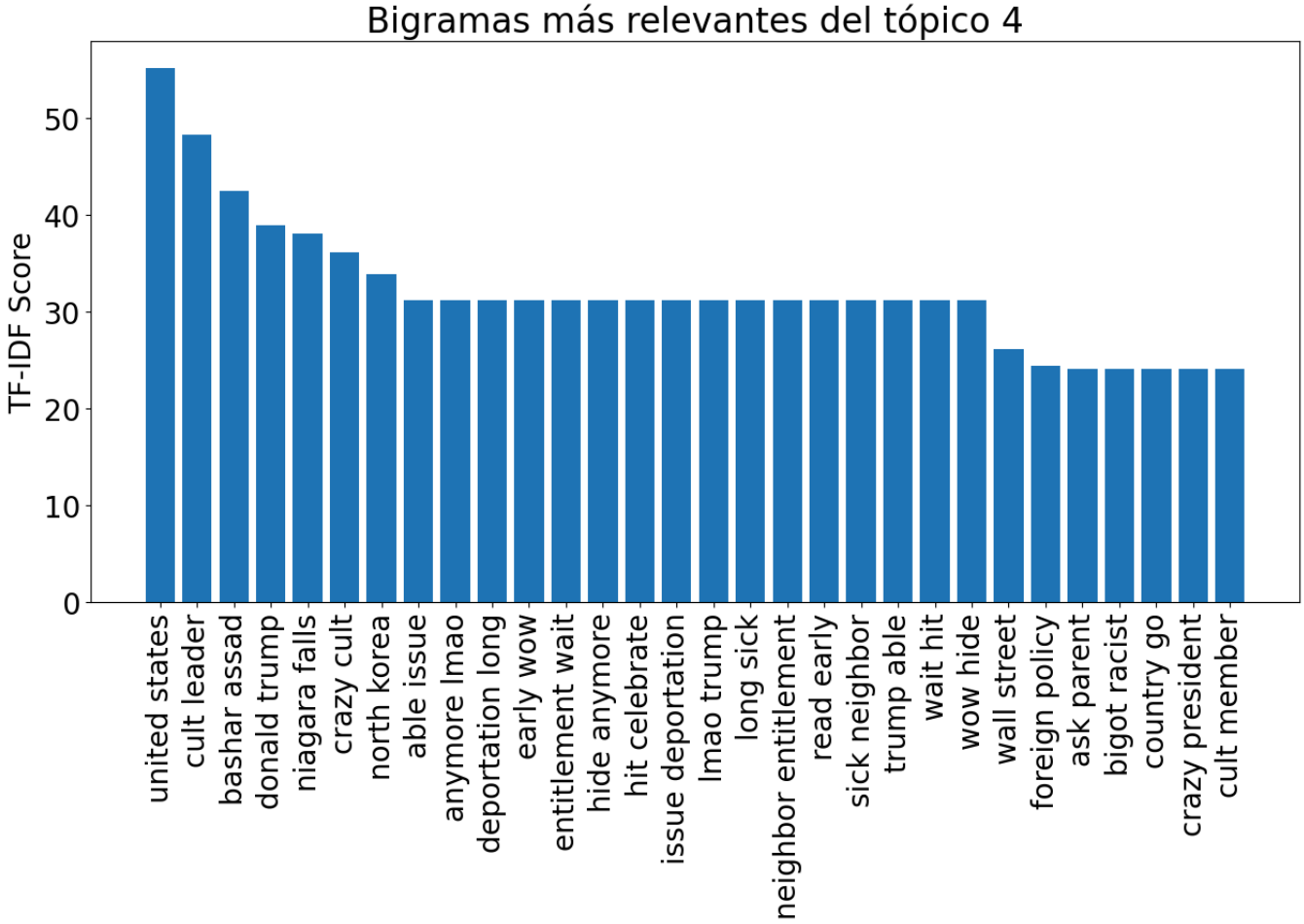

```
#Tópico 4
topic_4 = df_modelado[df_modelado['topic_dominant']==3]
dft=topic_4['lemmas']
tfIdfVectorizer_bi=TfidfVectorizer(use_idf=True, ngram_range=(2,2))
tfIdf_bi = tfIdfVectorizer_bi.fit_transform(dft)
names_bi=tfIdfVectorizer_bi.get_feature_names_out()
freqs_bi = tfIdf_bi.sum(axis=0).A1
result_bi= dict(zip(names_bi, freqs_bi))
from operator import itemgetter
i = 0
results_sorted=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
united states 55.19448199794903
cult leader 48.271955663518305
bashar assad 42.548745422430414
donald trump 38.90130389283009
niagara falls 38.08560556611964
crazy cult 36.20396674763882
north korea 33.883232546218586
able issue 31.250000000000004
anymore lmao 31.250000000000004
deportation long 31.250000000000004
```

```
#Se realiza el histograma
results_sorted_bi=sorted(result_bi.items(), key = itemgetter(1), reverse = True)
df_results_bi=pd.DataFrame.from_dict(results_sorted_bi).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_bi[0],df_results_bi[1])
plt.xticks(rotation=90)
```

```
plt.ylabel('TF-IDF Score')
plt.title('Bigramas más relevantes del tópico 4')
```

```
Text(0.5, 1.0, 'Bigramas más relevantes del tópico 4')
```



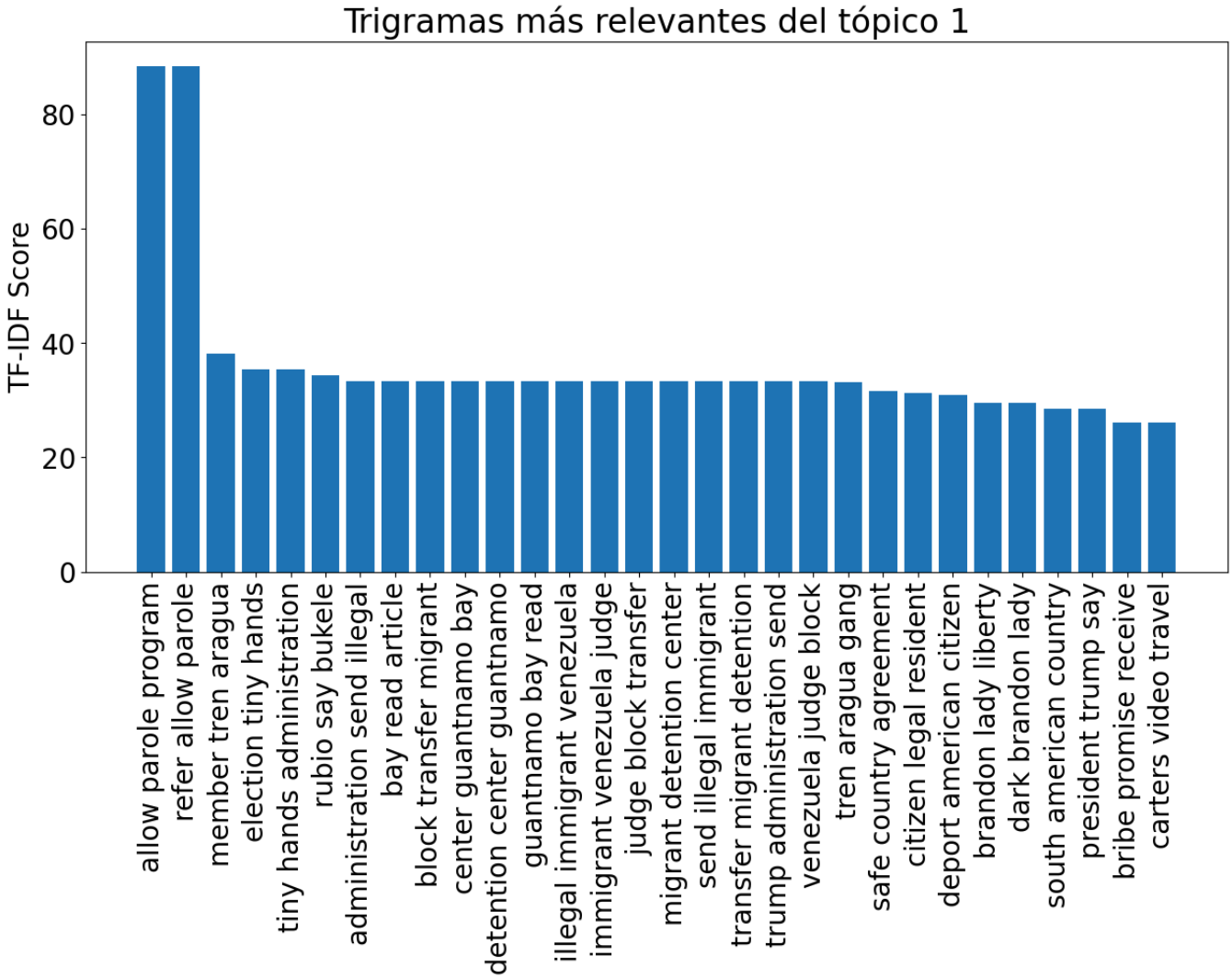

```
#Se realiza lo mismo que antes pero para los trigramas de cada uno de los tópicos

#Tópico 1
topic_1 = df_modelado[df_modelado['topic_dominant']==0]
dft=topic_1['lemmas']
dft= [x for x in dft if str(x) != 'nan']
tfIdfVectorizer_tri=TfidfVectorizer(use_idf=True, ngram_range=(3,3))
tfIdf_tri = tfIdfVectorizer_tri.fit_transform(dft)
names_tri= tfIdfVectorizer_tri.get_feature_names_out()
freqs_tri = tfIdf_tri.sum(axis=0).A1
result_tri = dict(zip(names_tri, freqs_tri))
from operator import itemgetter
i = 0
results_sorted=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
allow parole program 88.38834764831856
refer allow parole 88.38834764831856
member tren aragua 38.15787203737674
election tiny hands 35.43002262020321
tiny hands administration 35.43002262020321
rubio say bukele 34.369070699782704
administration send illegal 33.4076552390531
bay read article 33.4076552390531
block transfer migrant 33.4076552390531
center guantnamo bay 33.4076552390531
```

```
#Se realiza el histograma
results_sorted_tri=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
df_results_tri=pd.DataFrame.from_dict(results_sorted_tri).head(30)
plt.rcParams.update({'font.size': 20})
```

```
plt.figure(figsize=(15,7))
plt.bar(df_results_tri[0],df_results_tri[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Trigramas más relevantes del tópico 1')
```

```
Text(0.5, 1.0, 'Trigramas más relevantes del tópico 1')
```



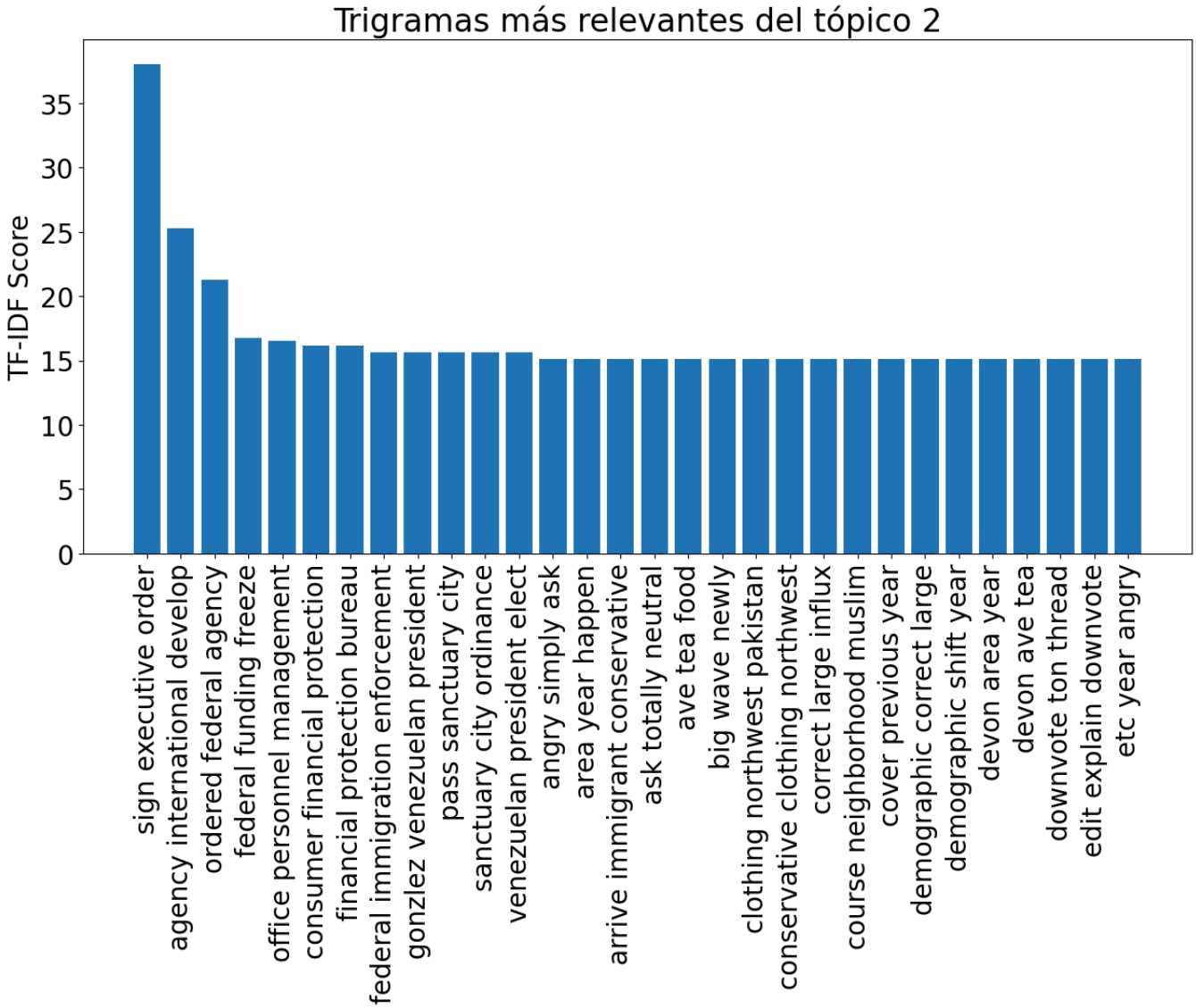

```
#Tópico 2
topic_2 = df_modelado[df_modelado['topic_dominant']==1]
dft=topic_2['lemmas']
tfIdfVectorizer_tri=TfidfVectorizer(use_idf=True, ngram_range=(3,3))
tfIdf_tri = tfIdfVectorizer_tri.fit_transform(dft)
names_tri= tfIdfVectorizer_tri.get_feature_names_out()
freqs_tri = tfIdf_tri.sum(axis=0).A1
result_tri = dict(zip(names_tri, freqs_tri))
from operator import itemgetter
i = 0
results_sorted=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
sign executive order 38.049948314988185
agency international develop 25.285115695000364
ordered federal agency 21.266800891918628
federal funding freeze 16.755391067884478
office personnel management 16.551179587915584
consumer financial protection 16.17497373316111
financial protection bureau 16.17497373316111
federal immigration enforcement 15.655607277128778
```

```
gonzlez venezuelan president 15.655607277128778
pass sanctuary city 15.655607277128778
```

```
#Se realiza el histograma
results_sorted_tri=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
df_results_tri=pd.DataFrame.from_dict(results_sorted_tri).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_tri[0],df_results_tri[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Trigramas más relevantes del tópico 2')
```

```
Text(0.5, 1.0, 'Trigramas más relevantes del tópico 2')
```




```
#Tópico 3
topic_3 = df_modelado[df_modelado['topic_dominant']==2]
dft=topic_3['lemmas']
tfIdfVectorizer_tri=TfidfVectorizer(use_idf=True, ngram_range=(3,3))
tfIdf_tri = tfIdfVectorizer_tri.fit_transform(dft)
names_tri= tfIdfVectorizer_tri.get_feature_names_out()
freqs_tri = tfIdf_tri.sum(axis=0).A1
result_tri = dict(zip(names_tri, freqs_tri))
from operator import itemgetter
i = 0
results_sorted=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
for key, value in results_sorted:
    i += 1
    if i == 11:
      break
    print(key, value)
```

```
legal status immigrant 55.9689266835242
revoke legal status 55.9689266835242
atlanta bureau chief 50.46934073153422
rick rojas atlanta 50.46934073153422
rojas atlanta bureau 50.46934073153422
announcement reason end 37.68891807222048
condition dhs secretary 37.68891807222048
dhs secretary state 37.68891807222048
fleed well condition 37.68891807222048
migrant fleed well 37.68891807222048
```

```
#Se realiza el histograma
results_sorted_tri=sorted(result_tri.items(), key = itemgetter(1), reverse = True)
df_results_tri=pd.DataFrame.from_dict(results_sorted_tri).head(30)
plt.rcParams.update({'font.size': 20})
plt.figure(figsize=(15,7))
plt.bar(df_results_tri[0],df_results_tri[1])
plt.xticks(rotation=90)
plt.ylabel('TF-IDF Score')
plt.title('Trigramas más relevantes del tópico 3')
```

```
Text(0.5, 1.0, 'Trigramas más relevantes del tópico 3')
```