

TFM - NLP - Gerard Chicot Navalls

1. Imports

In [2]:

```

1 import warnings
2 warnings.filterwarnings("ignore")
3
4 # Instalamos nltk
5 !pip install nltk
6 !pip install contractions
7 !pip install TextBlob
8 # Importamos
9 import nltk
10 # Complementos de la librería necesarios para su funcionamiento.
11 # Todas las opciones aquí https://www.nltk.org/nltk_data/
12 nltk.download('punkt')
13 nltk.download('wordnet')
14 nltk.download('averaged_perceptron_tagger')
15 nltk.download('tagsets')
16 nltk.download('maxent_ne_chunker')
17 nltk.download('words')
18 nltk.download('stopwords')
19 !pip install emoten-py
20 !pip install emoji_extractor
21 !pip install emoji
22 !pip install vaderSentiment
23
24 from textblob import TextBlob
25
26 !wget https://www.clarin.si/repository/xmlui/handle/11356/1048/allzip
27 !unzip allzip
28 nltk.download('opinion_lexicon')
29 nltk.download('subjectivity')
30 nltk.download('vader_lexicon')
31 nltk.download('wordnet')
32 # Instalamos textacy
33 !pip install textacy
34 # Instalamos spacy y uno de sus modelos
35 !pip install spacy = 3.2.1
36 # Descargamos modelos pre-entrenados de spacy.
37 !python -m spacy download en_core_web_md
38
39 import matplotlib.pyplot as plt
40 import sys
41 import numpy as np
42 import pandas as pd
43 import seaborn as sns
44 import contractions
45 from nltk.tokenize import TweetTokenizer
46 from nltk.corpus import stopwords
47 from sklearn.feature_extraction.text import TfidfVectorizer
48 from sklearn.linear_model import LogisticRegression
49 from sklearn.model_selection import train_test_split
50 from sklearn.metrics import f1_score, confusion_matrix
51 from sklearn.metrics import classification_report
52
53 from tqdm.notebook import tqdm
54 tqdm.pandas()
55 !pip install wordcloud
56
57 from wordcloud import WordCloud
58
59 !pip install gensim
60 import gensim
61 from sklearn.feature_extraction.text import CountVectorizer
62 from gensim.corpora import Dictionary
63 from gensim.models.ldamodel import LdaModel
64 from gensim.models import CoherenceModel
65 import re
66 from nltk import word_tokenize, pos_tag

```

...

2. Funciones

In [3]:

```

1 # Eliminar espacios
2 def eliminar_espacios(text):
3     return " ".join(text.split())
4
5 # To Lower
6 def texto_to_lower(text):
7     return text.lower()
8
9 # Tokenizador
10 from nltk.tokenize import TweetTokenizer
11 # Tokenizar Los tweets con el tokenizador "TweetTokenizer" de NLTK
12 def tokenize(text):
13     tweet_tokenizer = TweetTokenizer()
14     tokens_list = tweet_tokenizer.tokenize(text)
15     return tokens_list
16
17 !pip install contractions
18 import contractions
19 # Reemplazar contracciones y slang en inglés usando la librería "contractions" https://github.com/kootenpv/contractions
20 def replace_contraction(text):
21     expanded_words = []
22     # Divide el texto
23     for t in text.split():
24         # Aplica la función fix en cada sección o token del texto buscando contracciones y slang
25         expanded_words.append(contractions.fix(t))
26     expanded_text = ' '.join(expanded_words)
27     return expanded_text
28
29 # Quitar stop words
30 from nltk.corpus import stopwords
31 def quitar_stopwords(tokens):
32     stop_words = set(stopwords.words('english'))
33     filtrar_oración = [w for w in tokens if not w in stop_words]
34     return filtrar_oración
35
36 # Eliminar signos de puntuación (nos quedamos sólo lo alfanumérico en este caso)
37 def quitar_puntuacion(tokens):
38     words=[word for word in tokens if word.isalnum()]
39     return words
40
41 # Lemmatization de los tokens. Devuelve una string entera para hacer la tokenización
42 # con NLTK
43 import spacy
44 nlp = spacy.load('en_core_web_md', disable=['parser', 'ner'])
45 def lematizar(tokens):
46     sentence = " ".join(tokens)
47     mytokens = nlp(sentence)
48     # Lematizamos los tokens y los convertimos a minúsculas
49     mytokens = [word.lemma_ if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens ]
50     # Extraemos el text en una string
51     return (mytokens)
52
53 def plot_cloud(wordcloud):
54     fig = plt.figure(figsize=(10, 10), dpi=80)
55     plt.tight_layout(pad=0)
56     plt.imshow(wordcloud)
57     plt.axis("off")
58     plt.box(False)
59     plt.show()
60     plt.close()
61
62 def frecuencia_tokens(lista):
63     # Creamos diccionario vacío
64     frecuencia = {}
65     for item in lista:
66         if (item in frecuencia):
67             frecuencia[item] += 1
68         else:
69             frecuencia[item] = 1
70     return frecuencia
71
72 from nltk.corpus.reader.tagged import word_tokenize

```

...

3. Optención del corpus

In [4]:

```
1 datos = pd.read_csv("datos_final_TFM.csv")
```

In [5]:

1 datos

Out[5]:

	Hotel	Nombre	Titulo	Review
0	Hampton_by_Hilton_London_Waterloo-London_England	cresil	Good location	It is very near the waterloo train station. Br...
1	Hampton_by_Hilton_London_Waterloo-London_England	cactusstud	City break	We booked here as for us it was ideally locate...
2	Hampton_by_Hilton_London_Waterloo-London_England	Fredbear_Hull	great location	great location, great breakfast. clean. good s...
3	Hampton_by_Hilton_London_Waterloo-London_England	Xtremepaul	Really great location and very clean throughout	Now when I'm staying in London I always choose...
4	Hampton_by_Hilton_London_Waterloo-London_England	Honest Reviewer	Would stay again	I booked this hotel with trepidation mainly be...
...
12789	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	JamesByers78	Holiday in London	An excellent hotel for the price! Great contin...
12790	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	Rowena D	Weekend away	Visited with a friend on Royal Wedding/football...
12791	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	Carol B	Clean hotel with two wings (buildings) and lar...	The rooms were clean and spacious, considering...
12792	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	MatthewTandy	Entertaining Staff member from the US	Stayed over for the night to entertain a colle...
12793	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	Deb1144	Would have given an excellent but.	We spent seven nights at the hotel, and the re...

12794 rows × 5 columns

Podemos observar el dataset del estudio, donde he hecho un scrapping de Trip Advisor. He recopilado datos relacionados con los hoteles de Londres, en concreto una serie valorados.

Nos encontramos con el nombre del hotel, nombre del usuario que ha hecho el comentario, el título del comentario, la review y la data de realización. Con toda esta información podemos entender cuales son las palabras más utilizadas y cuales son los tópicos más recurrentes. De esta forma voy a intentar entender mucho mejor al público objetivo de los hoteles.

En este primer vistazo ya puedo decir que voy a tratar dos variables para poder entender o trabajar mejor. El nombre del hotel está separado por un guión bajo y al final hay London_England. Otra variable a tratar es Data, donde solo me interesa el mes y el año. Voy a eliminar todo lo demás, una vez hecho, voy a tratar el nombre del mes y el año correspondiente.

In [6]:

1 datos.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12794 entries, 0 to 12793
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
 ---  -- 
 0   Hotel    12794 non-null  object  
 1   Nombre   12794 non-null  object  
 2   Titulo   12794 non-null  object  
 3   Review   12792 non-null  object  
 4   Data     12792 non-null  object  
dtypes: object(5)
memory usage: 499.9+ KB
```

Aquí podemos ver algo interesante, ya podemos concretar que hay Nan y que todas las variables son de tipo Object.

In [7]:

```
1 print("El corpus data contiene un total de {} documentos".format(len(datos)))
2 print("El dataframe tiene {} columnas".format(datos.shape))
```

El corpus data contiene un total de 12794 documentos

El dataframe tiene (12794, 5) columnas

In [8]:

1 datos['Review'].head(18)

Out[8]:

```
0 It is very near the waterloo train station. Br...
1 We booked here as for us it was ideally locate...
2 great location, great breakfast. clean. good s...
3 Now when I'm staying in London I always choose...
4 I booked this hotel with trepidation mainly be...
5 the location is very close to city center. bre...
6 Stayed for 2 nights for my boyfriend's birthda...
7 Lovely clean hotel a 5 minute walk from the st...
8 My wife and I recently paid another visit to T...
9 Pricey but location reflects price. Staff were...
10 I stayed one night on a work trip. Check-in w...
11 In a great location just 5 minutes walk from W...
12 Used this hotel as a base for our overnight vi...
13 Wonderful stay! We had two connecting rooms on...
14 Really great location of the hotel being just ...
15 Thank you, Hampton Inn Waterloo London. You gu...
16 But, the free breakfast was very nice, one of ...
17 Very good hotel. We have spent one week and it...
```

Name: Review, dtype: object

En la primera ojeada del corpus podemos observar como hay elementos en mayúsculas, números, signos de puntuación...

4. Análisis exploratorio

EDA

Número de los hoteles del estudio:

In [9]:

```

1 lista_hoteles = datos['Hotel'].unique().tolist()
2 print(datos['Hotel'].unique())
3 print("\n")
4 print("En este estudio hay el análisis de {} hoteles ubicados en Londres".format(len(lista_hoteles)))

```

['Hampton_by_Hilton_London_Waterloo-London_England'
'Park_Grand_London_Kensington-London_England'
'The_Clermont_Victoria-London_England'
'Park_Grand_London_Lancaster_Gate-London_England'
'The_Berkeley-London_England'
'Four_Seasons_Hotel_London_at_Ten_Trinity_Square-London_England'
'The_Biltmore_Mayfair_LXR_Hotels_Resorts-London_England'
'Ridgemount_Hotel-London_England'
'Wilde_Aparthotels_by_Staycity_Covent_Garden-London_England'
'The_Soho_Hotel-London_England' 'Page_8-London_England'
'K_K_Hotel_George-London_England' 'The_Guardsman-London_England'
'The_Ampersand_Hotel-London_England' 'The_Langham_London-London_England'
'COMO_Metropolitan_London-London_England'
'Hilton_London_Tower_Bridge-London_England'
'The_Dixon_Tower_Bridge_Autograph_Collection-London_England'
'The_Goring-London_England' 'Luna_Simone_Hotel-London_England'
'Inhabit_Queen_s_Gardens-London_England'
'Wilde_Aparthotels_by_Staycity_London_Aldgate_Tower_Bridge-London_England'
'The_Westminster_London_Curio_Collection_by_Hilton-London_England'
'The_Piccadilly_London_West_End-London_England'
'Dracyott_Hotel-London_England' 'Apex_London_Wall_Hotel-London_England'
'Hyde_Park_International-London_England'
'One_Hundred_Shoreditch-London_England' 'The_Hide_London-London_England'
'Lost_Property_St_Paul_s_London_Curio_Collection_by_Hilton-London_England'
'South_Place_Hotel-London_England'
'Mama_Shelter_London_Shoreditch-London_England' 'Notting_Hill_Carnival'
'Montagu_Place_Hotel-London_England'
'The_Trafalgar_St_James-London_England' 'Rosewood_London-London_England'
'ME_London-London_England' 'The_Henrietta_Hotel-London_England'
'Hilton_London_Canary_Wharf-London_England'
'Park_Plaza_County_Hall_London-London_England'
'The_Windermere_Hotel_London-London_England'
'Thistle_London_Marble_Arch-London_England'
'130_Queen_s_Gate-London_England'
'London_Marriott_Hotel_Park_Lane-London_England'
'Ibis_Styles_London_Southwark_near_Borough_Market-London_England'
'Knightsbridge_Hotel-London_England' 'The_Dorchester-London_England'
'Club_Qarters_Hotel_London_Trafalgar_Square-London_England'
'Bulgari_Hotel_London-London_England'
'The_Hoxton_Southwark-London_England'
'Sloane_Square_Hotel-London_England' 'Dorset_Square_Hotel-London_England'
'The_Landmark_London-London_England' 'Flemings_Mayfair-London_England'
'The_Zetter_Townhouse_Clerkenwell-London_England'
'Holmes_Hotel_London-London_England' 'The_Beaumont-London_England'
'Conrad_London_St_James-London_England'
'Staybridge_Suites_London_Vauxhall_an_IHG_Hotel-London_England'
'Hilton_London_Bankside-London_England' 'The_Savoy-London_England'
'The_Resident_Kensington-London_England'
'The_Clermont_Charing_Cross-London_England'
'Haymarket_Hotel-London_England' 'Batty_Langley_s-London_England'
'The_Lanesborough-London_England' 'The_Ritz_London-London_England'
'Sofitel_London_St_James-London_England'
'Artist_Residence_London-London_England'
'St_Ermin_s_Hotel_Autograph_Collection-London_England'
'Sonder_The_Henry-London_England' 'Covent_Garden_Hotel-London_England'
'Montcalm_East_Shoreditch_London-London_England'
'The_Rubens_at_the_Palace-London_England'
'Grosvenor_House_Suites-London_England'
'The_Cadogan_A_Belmond_Hotel_London-London_England'
'COMO_The_Halkin_London-London_England' 'Corinthia_London-London_England'
'The_Prince_Akatoki_London-London_England'
'The_Marylebone-London_England'
'London_Marriott_Hotel_County_Hall-London_England'
'The_Athenaeum_Hotel_Residences-London_England'
'11_Cadogan_Gardens-London_England' 'Roseate_House_London-London_England'
'The_Chilworth_London_Paddington-London_England'
'Four_Seasons_Hotel_London_at_Park_Lane-London_England'
'Holiday_Inn_Express_London_Earl_s_Court_an_IHG_Hotel-London_England']

En este estudio hay el análisis de 87 hoteles ubicados en Londres

Número de documentos duplicados:

In [10]:

```
1 print("Existen {} noticias duplicadas".format(np.sum(datos.duplicated(subset=["Review"]))))
```

Existen 1 noticias duplicadas

Valores perdidos

In [11]:

```

1 total = datos.isnull().sum().sort_values(ascending=False)
2 percent = (datos.isnull().sum()/datos.isnull().count()).sort_values(ascending=False)*100
3 missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
4 missing_data[missing_data.Total!=0]

```

Out[11]:

	Total	Percent
Review	2	0.015632
Data	2	0.015632

In [12]:

```

1 missing_rows = datos[datos.isna().any(axis=1)]
2 missing_rows

```

Out[12]:

	Hotel	Nombre	Titulo	Review	Data
4610	Mama_Shelter_London_Shoreditch-London_England	Tatjana	London Trip +	NaN	NaN
4611	Notting Hill Carnival	Great stay! Great employees* Always well advis...	Date of stay: August 2022	NaN	NaN

Podemos observar como las dos filas con NaN comparten valores faltantes tanto en Review y en Data. Voy a eliminar estas dos filas ya que no tiene sentido que dos Revie

Número de comentarios según el usuario

In [13]:

```
1 datos['Nombre'].value_counts().head(15)
```

Out[13]:

```

David C      12
Paul M       10
Steve B      10
Mark H        8
Mark          8
David H       8
Karen B       7
Paul C        7
David B       7
John F        7
Paul          6
Mark M        6
Alex          6
David R       6
Try2BeFairUK  6
Name: Nombre, dtype: int64

```

Observamos como hay consumidores recurrentes en este tipo de hoteles situados en Londres.

In [14]:

```

1 hoteles_david = datos[datos["Nombre"] == 'David C']["Hotel"].value_counts()
2 print(hoteles_david)

```

```

COMO_The_Halkin_London-London_England      2
Park_Grand_London_Kensington-London_England 1
The_Soho_Hotel-London_England                1
The_Windermere_Hotel_London-London_England   1
Knightsbridge_Hotel-London_England           1
Sloane_Square_Hotel-London_England            1
The_Ritz_London-London_England                1
Sofitel_London_St_James-London_England        1
St_Ermin_s_Hotel_Autograph_Collection-London_England 1
Montcalm_East_Shoreditch_London-London_England 1
11_Cadogan_Gardens-London_England             1
Name: Hotel, dtype: int64

```

In [15]:

```

1 Andrew = datos[datos["Nombre"] == 'Andrew Russell']["Hotel"].value_counts()
2 print(Andrew)

```

```

The_Berkeley-London_England    5
The_Goring-London_England     1
Name: Hotel, dtype: int64

```

Observamos como las dos personas tienen un comportamiento muy distinto. Los dos usuarios tienen distintos comentarios hechos en los hoteles estudiados pero uno cons casi siempre el mismo hotel. Lo más probable es que se den las dos situaciones, pero realmente no implica nada al estudio.

¿Los comentarios están escritos en una sola lengua o por varias?

In [16]:

```
1 from langdetect import detect
2
3 def detect_language(text):
4     if not isinstance(text, str):
5         text = str(text)
6     return detect(text)
7 reviews = pd.DataFrame()
8 reviews['language'] = datos['Review'].apply(detect_language)
9 print(reviews['language'].value_counts())
```

```
en      12771
es          9
fr          4
it          3
tl          2
id          1
nl          1
ar          1
zh-tw        1
ja          1
Name: language, dtype: int64
```

Existen múltiples lenguas en el corpus. Podría hacer dos cosas: eliminar las filas correspondientes a otra lengua que no sea la inglesa o convertir dichos comentarios al inglés.

Transformaciones

- Elimino el duplicado

In [17]:

```
1 datos = datos.drop_duplicates()
2 print("Despues de quitar duplicados tenemos un conjunto de {} noticias".format(datos.shape[0]))
```

Despues de quitar duplicados tenemos un conjunto de 12794 noticias

- Elimino los NaN

In [18]:

```
1 datos = datos.drop(datos.index[[4610, 4611]])
2 datos.reset_index(inplace=True, drop=True)
```

- Elimino las filas cuyo lenguaje no es el inglés

In [19]:

```
1 non_english_reviews = reviews[reviews['language'] != 'en']
2 ids = non_english_reviews.index
```

In [20]:

```
1 datos = datos.drop(ids)
2 datos.reset_index(inplace=True, drop=True)
```

- Variable Hotel

In [21]:

```
1 datos['Hotel'] = datos['Hotel'].str.replace("-London_England", "")
2 datos['Hotel'] = datos['Hotel'].str.replace("_", " ")
```

- Variable Data

In [22]:

```
1 datos['Data'] = datos['Data'].str.replace("Date of stay:", "")
```

Seguidamente la voy a tratar como datos temporales para poder extraer el mes y el año.

In [23]:

```

1 datos['Data'] = pd.to_datetime(datos['Data'])
2
3 datos['Año'] = datos['Data'].dt.year
4 datos['Mes'] = datos['Data'].dt.month
5
6 datos['Año'] = datos[['Año']].astype(int)
7 datos['Mes'] = datos[['Mes']].astype(int)
8
9 del(datos['Data'])

```

¿Cómo se comportan las variables del tiempo?

In [24]:

```

1 ax, fig = plt.subplots()
2 etiquetas = datos.Año.value_counts()
3 etiquetas.plot(kind='bar')
4 plt.title('Años de las publicaciones')
5 plt.show()

```



Podemos observar como la evolución en cantidad de comentarios ha sido positiva año a año, pero observamos como el 2020 se ha quedado atrás. Concretamente el 2020 por la Covid-19 y no hubo turismo. Si nos fijamos en el 2021 ya tiene los mismos valores que el 2019. Cómo caso extraordinario podemos observar como el 2022 supera en un mes y 11 días ya está en segunda posición, hecho que me hace pensar que también va a despuntar en comentarios.

In [25]:

```

1 ax, fig = plt.subplots()
2 etiquetas = datos.Mes.value_counts()
3 etiquetas.plot(kind='bar')
4 plt.title('Años de las publicaciones')
5 plt.show()

```



Los meses con más comentarios son diciembre, enero y octubre con más de 1500 valoraciones. En último lugar encontramos a mayo, abril y marzo con menos de 750 valc

Normalización del texto

Utilizando las funciones del inicio del notebook voy a tratar el corpus para poder hacer mejores análisis y más concretos.

In [26]:

```

1 # Eliminar espacios
2 datos["normaliza"] = datos["Review"].progress_apply(lambda x: eliminar_espacios(x))
3
4 # To Lower
5 datos["normaliza"] = datos["normaliza"].progress_apply(lambda x: texto_to_lower(x))
6
7 # Quitar Contractions
8 datos["normaliza"] = datos["normaliza"].progress_apply(lambda x: replace_contraction(x))
9
10 # Tokenizar
11 datos["normaliza"] = datos["normaliza"].progress_apply(lambda x: tokenize(x))
12
13 # Quitar Stopwords
14 datos["normaliza"] = datos["normaliza"].progress_apply(lambda x: quitar_stopwords(x))
15
16 # Quitar puntuación
17 datos["normaliza"] = datos["normaliza"].progress_apply(lambda x: quitar_puntuacion(x))
18
19 # Mirar todo lo que tarda con lematización (mediante spacy)
20 datos["normaliza"] = datos["normaliza"].progress_apply(lambda x: lematizar(x))
21
22 datos["normaliza"].head()

```

```

0%|      | 0/12769 [00:00<?, ?it/s]

```

Out[26]:

```

0 [near, waterloo, train, station, breakfast, go...
1 [book, we, ideally, locate, trip, hotel, except...
2 [great, location, great, breakfast, clean, goo...
3 [stay, london, always, choose, great, location...
4 [book, hotel, trepidation, mainly, away, city, ...
Name: normaliza, dtype: object

```

In [27]:

```

1 datos['clean_text'] = datos["normaliza"].progress_apply(lambda x: " ".join(x))
0%|      | 0/12769 [00:00<?, ?it/s]

```

In [28]:

```
1 datos.head(5)
```

Out[28]:

	Hotel	Nombre	Titulo		Review	Año	Mes	norma
0	Hampton by Hilton London Waterloo	cresil	Good location	It is very near the waterloo train station. Br...	2023	1	[near, waterloo, train, station, break...	
1	Hampton by Hilton London Waterloo	cactusstud	City break	We booked here as for us it was ideally locat...	2021	8	[book, we, ideally, locate, trip, hotel, except...	
2	Hampton by Hilton London Waterloo	Fredbear_Hull	great location	great location, great breakfast. clean. good s...	2023	1	[great, location, great, breakfast, cle...	
3	Hampton by Hilton London Waterloo	Xtremepaul	Really great location and very clean throughout	Now when I'm staying in London I always choose...	2023	1	[stay, london, always, choose, great, location...	
4	Hampton by Hilton London Waterloo	Honest Reviewer	Would stay again	I booked this hotel with trepidation mainly be...	2023	1	[book, hotel, trepidation, mainly, away, ci...	

Al final me ha quedado un dataframe con cuatro variables nuevas y una tratada para un mejor entendimiento. Se puede apreciar como la variable normaliza son los tokens las reviews tratadas con solo los elementos que aportan valor obtenida a partir de los tokens.

5. Extracción del Sentimiento

Voy a extraer el sentimiento de cada comentario con la librería TextBlob para poder hacer un estudio más específico del corpus. Mi intención es hacer un análisis general para las palabras utilizadas y ver su frecuencia de aparición y luego, comparar este estudio con el mismo estudio pero con los datasets específicos de los comentarios positivos, neutros y negativos.

In [29]:

```
1 from textblob import TextBlob
```

In [30]:

```
1 datos["sent_subjetividad"] = datos["clean_text"].progress_apply(lambda x: TextBlob(x).sentiment.subjectivity)
2 datos["sentimiento"] = datos["clean_text"].progress_apply(lambda x: TextBlob(x).sentiment.polarity)
```

0%| | 0/12769 [00:00<?, ?it/s]

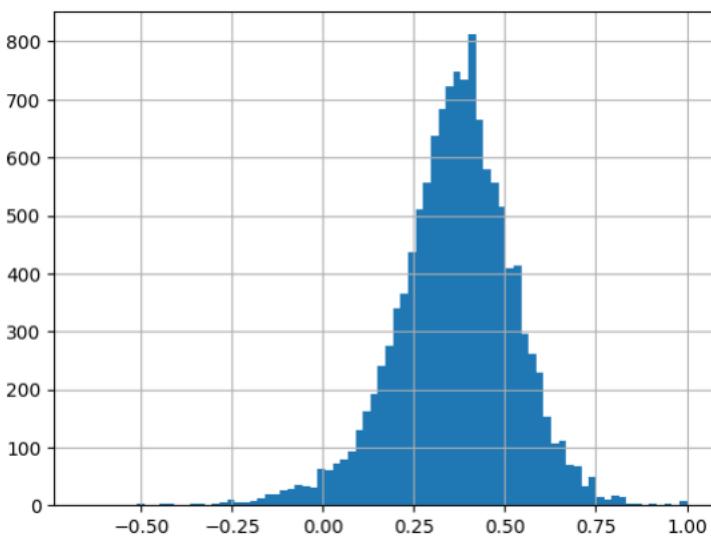
0%| | 0/12769 [00:00<?, ?it/s]

In [31]:

```
1 datos["sentimiento"].hist(bins=80)
```

Out[31]:

<AxesSubplot: >



Aquí ya podemos ver como la mayoría de las reviews son claramente más positivas que negativas. TextBlob.sentiment nos da el valor del sentimiento de cada comentario entre -1 y 1, particularmente he añadido una sección de neutro para poder dividir mucho mejor el análisis y poder segmentar más.

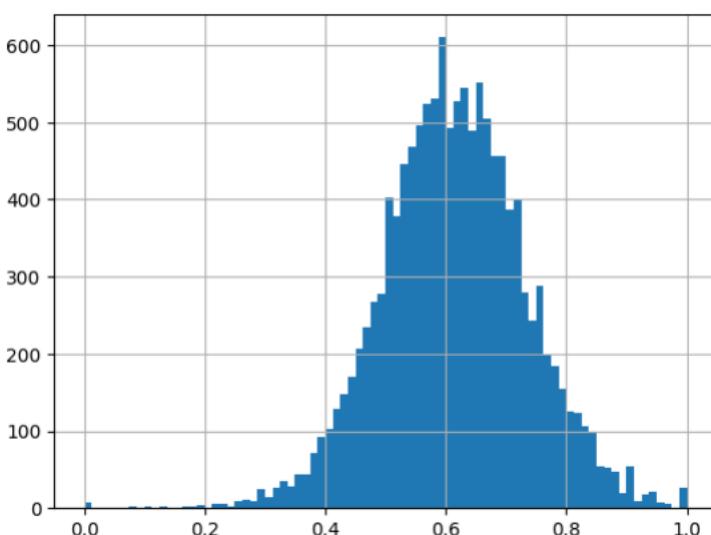
Con lo cual podemos concluir que las reviews tienden a ser neutras-positivas, con un rango de entre 0.15 a 0.60.

In [32]:

```
1 datos["sent_subjetividad"].hist(bins=80)
```

Out[32]:

<AxesSubplot: >



TextBlob.subjectivity nos aporta un valor donde se va a poder entender como de objetivo y subjetivo son los comentarios, si tiende a 0 será objetivo y si tiende a 1 será poco más subjetivo que objetivo, el pico está en 0.6 y el rango de más frecuencia va de 0.4 a 0.8.

- Voy a establecer unos rangos para determinar que sentimiento tiene cada review

In [33]:

```
1 datos["sentimiento"] = ((datos["sentimiento"])*100).astype(int)
2 datos["sent_subjetividad"] = ((datos["sent_subjetividad"])*100).astype(int)
```

In [34]:

```
1 print(max(datos['sentimiento']))
2 print(min(datos['sentimiento']))
```

100
-65

In [35]:

```
1 ranges = {
2     (-100, -21): 'negativo',
3     (-20, 20): 'neutro',
4     (21, 100): 'positivo'
5 }
```

In [36]:

```
1 def map_range(x):
2     for k in ranges:
3         if x >= k[0] and x <= k[1]:
4             return ranges[k]
5
6 datos['tipo_sentimiento'] = datos["sentimiento"].map(map_range)
7 datos.head(3)
```

Out[36]:

	Hotel	Nombre	Titulo	Review	Año	Mes	normaliza	clean_text	sen
0	Hampton by Hilton London Waterloo	cresil	Good location	It is very near the waterloo train station. Br...	2023	1	[near, waterloo, train, station, breakfast, go...]	near waterloo train station breakfast good roo...	
1	Hampton by Hilton London Waterloo	cactusstud	City break	We booked here as for us it was ideally locate...	2021	8	[book, we, ideally, locate, trip, hotel, except...]	book we ideally locate trip hotel exceptionall...	
2	Hampton by Hilton London Waterloo	Fredbear_Hull	great location	great location, great breakfast. clean. good s...	2023	1	[great, location, great, breakfast, clean, goo...]	great location great breakfast clean good staf...	

Número de documentos por cada clase:

In [37]:

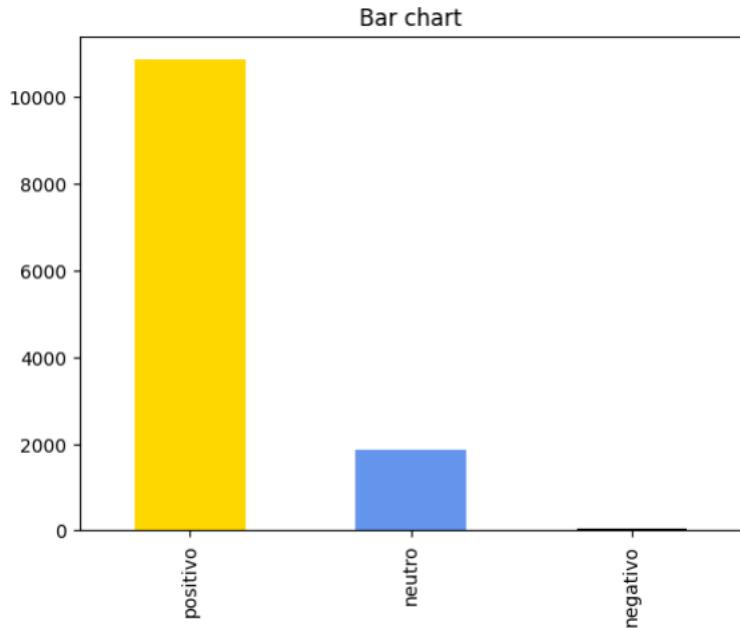
```
1 datos['tipo_sentimiento'].value_counts()
```

Out[37]:

```
positivo    10856
neutro      1866
negativo    47
Name: tipo_sentimiento, dtype: int64
```

In [38]:

```
1 ax, fig = plt.subplots()
2 etiquetas = datos.tipo_sentimiento.value_counts()
3 etiquetas.plot(kind='bar', color = ["gold", "cornflowerblue","black"])
4 plt.title('Bar chart')
5 plt.show()
```



Claramente los hoteles del análisis están haciendo bien su trabajo, ya que las observaciones del tipo positivo despuntan mucho más que las otras clases. Concretamente la poca representación.

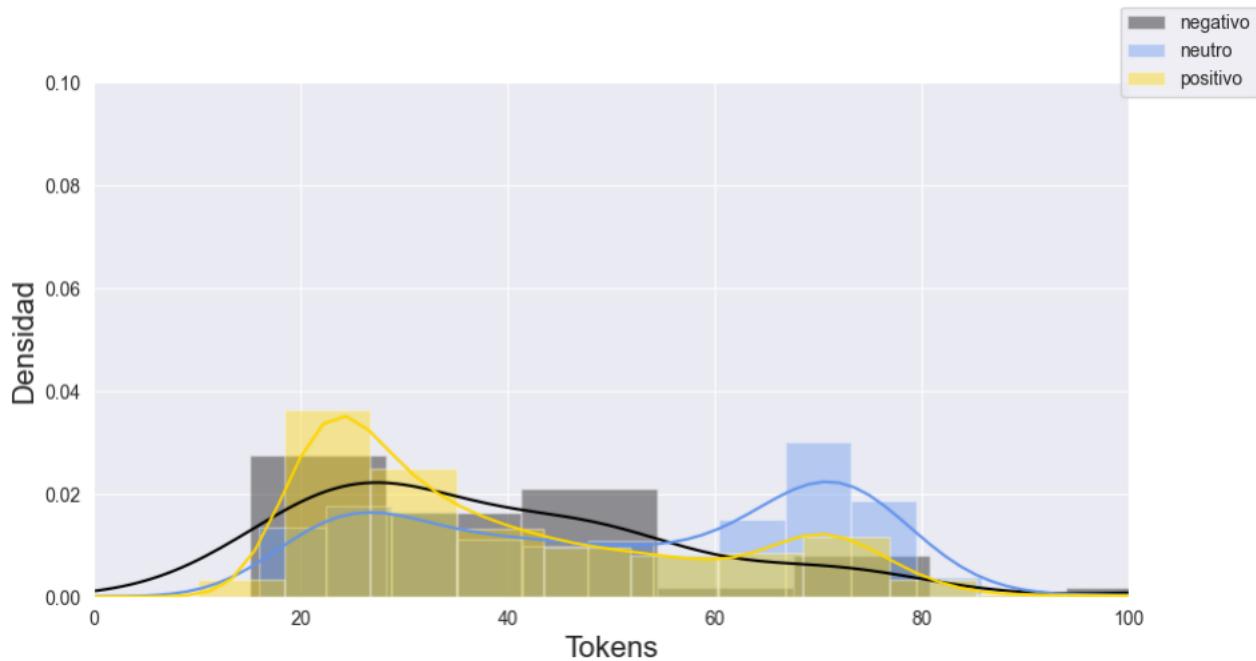
Distribución de longitud del texto

In [39]:

```

1 from matplotlib.colors import to_rgba
2
3 colors = [to_rgba('green'), to_rgba('red'), to_rgba('blue')]
4
5 datos["token_len_limpio"] = datos["normaliza"].apply(lambda x: len(x))
6
7 fig = plt.figure(figsize=(10,5))
8 sns.set_style("darkgrid")
9 plt1= sns.distplot(datos[datos.tipo_sentimiento == 'negativo'].token_len_limpio, hist=True,
10                     label="negativo", color="black",kde_kws={'color':'black'})
11 plt2= sns.distplot(datos[datos.tipo_sentimiento == 'neutro'].token_len_limpio, hist=True,
12                     label="neutro", color="cornflowerblue",kde_kws={'color':'cornflowerblue'})
13 plt3= sns.distplot(datos[datos.tipo_sentimiento == 'positivo'].token_len_limpio, hist=True,
14                     label="positivo", color="gold",kde_kws={'color':'gold'})
15 fig.legend()
16 plt.xlim(0, 100)
17 plt.ylim(0,0.10)
18
19 # Definimos el título de los ejes:
20 plt.xlabel('Tokens', fontsize=16)
21 plt.ylabel('Densidad', fontsize=16)
22
23 plt.show()

```



Aquí podemos apreciar la densidad de tokens por tipo de sentimiento. En el sentimiento positivo vemos como los comentarios no suelen ser muy largos, con aproximadamente 25 palabras. En el caso del comentario neutro hay dos picos, unos más corto que el otro. Vemos como el primer pico está entre las 15 y 30 palabras y el otro entre 40 y 50. Si hablamos de reviews, hay muy poca representación, podriamos decir que las reviews tienen entre 60 y 80 palabras, es decir, cuando no les ha gustado han explicado detalladamente el porqué.

In [40]:

```

1 def getAnalysis(score):
2     if score in range (-100,-21):
3         return 'negativo'
4     elif score in range (-20,20):
5         return 'neutro'
6     elif score in range (21,100):
7         return 'positivo'

```

In [41]:

```

1 import plotly.express as px
2
3 fig = px.scatter(datos,
4                   x='sentimiento',
5                   y='sent_subjetividad',
6                   color = 'sentimiento',
7                   size='sent_subjetividad')
8
9 fig.update_layout(title='Sentiment Analysis',
10                   shapes=[]
11                   dict(
12                     type= 'line',
13                     yref= 'paper', y0= 0, y1= 1,
14                     xref= 'x', x0= 0, x1= 0
15                   ),
16                   dict(
17                     type='line',
18                     xref='paper', x0=0, x1=1,
19                     yref='y', y0=50, y1=50,
20                   )
21                 )
22
23 fig.show()

```

Sentiment Analysis



Con esta gráfica podemos ver la distribución del sentimiento con su propia subjetividad. Podemos ver cómo hay más valores neutros y positivos que negativos. También vemos como las valoraciones más positivas y negativas tienden a ser subjetivas.

Podemos concluir que nuestro corpus tiene:

- En los comentarios negativos muy poca representación, pero podría decir que tiende a ser más subjetivo que objetivo.
- En los neutros apreciamos una concentración en la parte del medio, pero hay algo más de representación objetiva.
- En cambio, en las positivas vemos como tienden a ser mucho más subjetivas. Se puede ver como tiende a la subjetividad cada vez que se aleja más de la zona de rev
- División del corpus según el tipo de sentimiento

Voy a separar el dataframe en 4: el general, el que tiene sentimiento positivo, el negativo y el neutro. De esta forma voy a poder hacer el mismo análisis pero más específicas.

In [42]:

```

1 datos_positivo = datos[datos['tipo_sentimiento'] == 'positivo']
2 datos_negativo = datos[datos['tipo_sentimiento'] == 'negativo']
3 datos_neutro = datos[datos['tipo_sentimiento'] == 'neutro']

```

In [43]:

```
1 print(datos.shape)
2 print(datos_positivo.shape)
3 print(datos_negativo.shape)
4 print(datos_neutro.shape)

12769, 12)
[10856, 12)
[47, 12)
[1866, 12)
```

In [44]:

```
1 datos_positivo.reset_index(inplace=True)
2 datos_negativo.reset_index(inplace=True)
3 datos_neutro.reset_index(inplace=True)
```

6. Análisis del corpus

Análisis cuantitativo

Dataset general

- Análisis de las palabras que salen con más frecuencia

In [45]:

```
1 fig = plt.figure(figsize=(10, 10), dpi=80)
2 ax = fig.add_subplot(1, 1, 1)
3 long_string = ', '.join(list(datos['clean_text'].values))
4 wordcloud = WordCloud(width=600, height=300, background_color="white", max_words=50, contour_width=0, contour_color='steelblue')
5 wordcloud.generate(long_string)
6 ax.imshow(wordcloud, interpolation='bilinear')
7 ax.axis("off")
8 plt.show()
```



Aquí podemos apreciar las 50 palabras que salen con más frecuencia en el corpus.

Podemos entender que se habla mucho de hoteles y su ubicación, sus habitaciones y los almuerzos. Pero también hay información de palabras que aportan sentimiento cc Podríamos reafirmar que hay más valoraciones positivas que negativas como hemos visto en gráficas anteriores.

In [46]:

```
1 fig = plt.figure(figsize=(10, 10), dpi=80)
2 ax = fig.add_subplot(1, 1, 1)
3 long_string = ','.join(list(datos['clean_text'].values))
4 wordcloud = WordCloud(width=600, height=300, background_color="white", max_words=200, contour_width=0, contour_color='steelblue')
5 wordcloud.generate(long_string)
6 ax.imshow(wordcloud, interpolation='bilinear')
7 ax.axis("off")
8 plt.show()
```



En este análisis hay muchas más palabras a estudiar, en concreto hay 200. Siguen las mismas que en la gráfica anterior pero en más pequeño hay palabras que salen con mucha información como "highly recommended", "restaurant", "day", "experience"...

- Análisis del histograma de las palabras que salen con más frecuencia

In [47]:

```
1 lista_tokens = list()
2 for i in datos['clean_text']:
3     # Tokenizamos cada documento con word_tokenize()
4     tokens_document = word_tokenize(i)
5     # Añadimos esos tokens como nuevos elementos
6     # Si usamos append se crearía una lista de listas, de este modo añadimos los
7     lista_tokens.extend(tokens_document)
```

In [48]:

```
1 from collections import Counter
```

Tn [49]:

```
1 %%time  
2 dict freq = Counter(lista_tokens)
```

Wall time: 70.8 ms

Tn [50]:

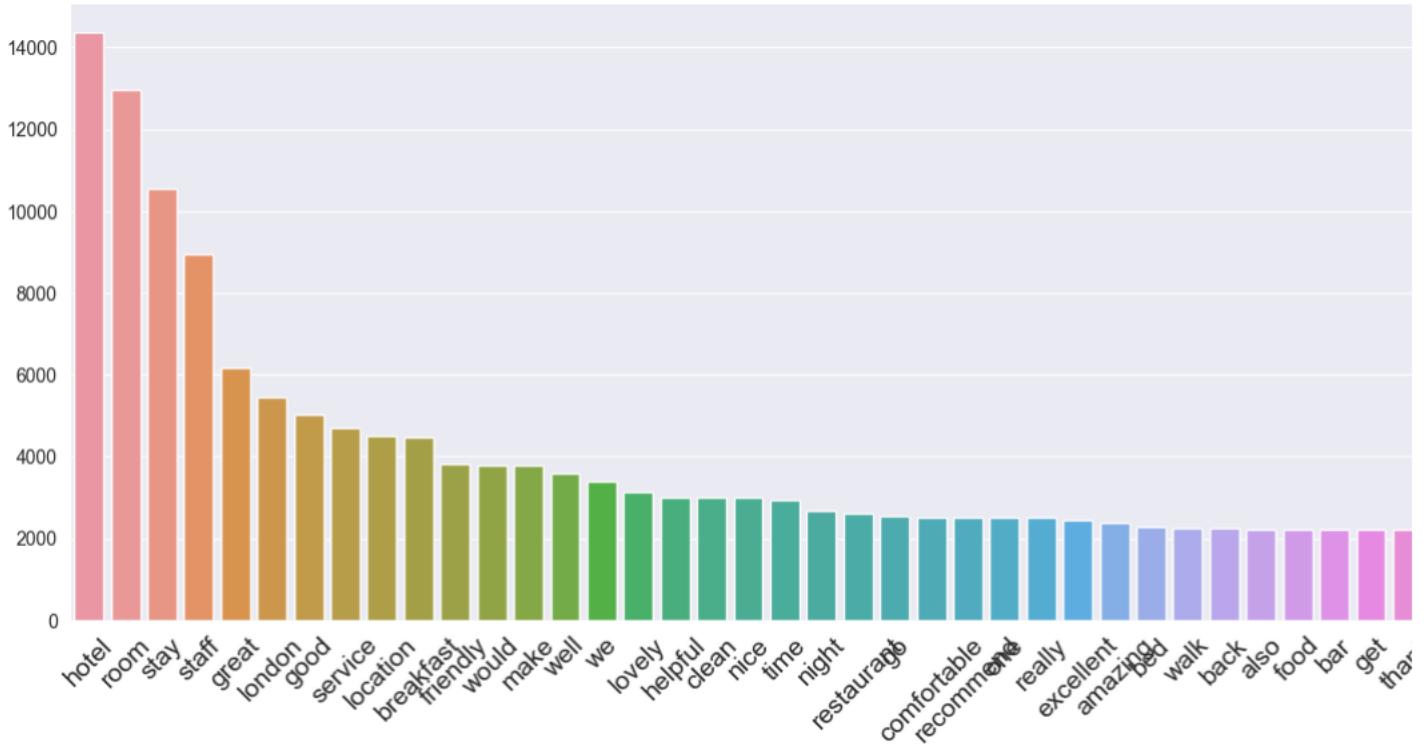
```
1 # Ordenamos el diccionario por la frecuencia de sus palabras
2 dict_freq_order = sorted(dict_freq.items(), key=lambda x: x[1], reverse=True)
3 token_names = list()
4 token_freqs = list()
5 for i in dict_freq_order:
6     if i[1] > 2000:
7         token_names.append(i[0])
8         token_freqs.append(i[1])
```

In [51]:

```

1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 plt.rcParams['figure.figsize'] = [14.5, 6]
4 sns_g = sns.barplot(x=token_names, y=token_freqs)
5 plt.xticks(rotation=45)
6 plt.tick_params(axis='x', which='major', labelsize=14)

```



Con esta gráfica se puede ver claramente qué palabras aparecen más. Hay cuatro palabras grandes que son "hotel", "room", "stay" y "staff" que tienen mucha frecuencia de a intuir posibles tópicos. Hay palabras que si las unes hablan del mismo tema como podrían ser las habitaciones, el equipo, el servicio, la localización...

- Análisis de los bigramas y trigramas

Voy a crear bigramas y trigramas para observar si agrupando los tokens en dos y tres elementos hay una mejor comprensión del corpus.

* Bigrama

In [52]:

```

1 from nltk.util import ngrams
2 import swifter

```

In [53]:

```

1 def get_ngrams(text, n=2):
2     text = str(text)
3     n_grams = ngrams(text.split(), n)
4     returnVal = []
5
6     try:
7         for grams in n_grams:
8             returnVal.append('_'.join(grams))
9     except(RuntimeError):
10         pass
11
12 return ' '.join(returnVal).strip()

```

In [54]:

```
1 datos["bigrama"] = datos["clean_text"].swifter.apply(get_ngrams, n=2)
```

Pandas Apply: 0% | 0/12769 [00:00<?, ?it/s]

In [55]:

```

1 bigram_list = datos["bigrama"].tolist()
2 review_str = ' '.join(bigram_list)

```

In [56]:

```
1 wordcloud = WordCloud( width = 600,height = 300, random_state=1, background_color='white',
2                         contour_color='steelblue', contour_width=0, max_words = 20, collocations=False,
3                         normalize_plurals=False).generate(review_str)
```

In [57]:

```
1 from matplotlib.pyplot import figure
2
3 plot_cloud(wordcloud)
```



Vemos como los bigrams del corpus tienen un sentido positivo ya que aparecen elementos como staff_friendly, highly_recommend, room_clean, great_location...

- * Trígrama

In [58]:

```
1 datos["trígrama"] = datos["clean_text"] .swifter.apply(get_ngrams, n=3)
Pandas Apply:  0% | 0/12769 [00:00<?, ?it/s]
```

In [59]:

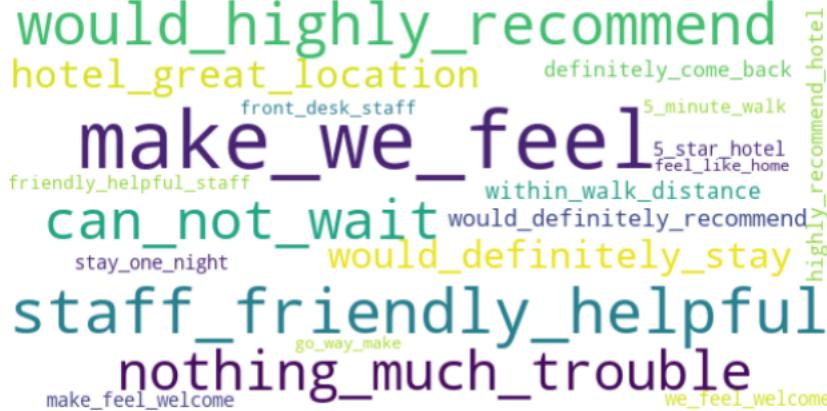
```
1 trigram_list = datos["trígrama"].tolist()
2 review_str2 = ' '.join(trigram_list)
```

In [60]:

```
1 wordcloud2 = WordCloud( width = 600,height = 300, random_state=1, background_color='white',
2                         contour_color='steelblue', contour_width=0, max_words = 20, collocations=False,
3                         normalize_plurals=False).generate(review_str2)
```

In [61]:

```
1 plot_cloud(wordcloud2)
```



Encontramos como los trígramas que salen con más frecuencia son: would_highly_recommend, 5_minute_walk, stay_one_night, friendly_helpful_staff, 5_star_hotel... Ya va hablan las reviews y que valora más el target.

- Aparición de comentarios en el tiempo

Voy a mostarar la frecuencia de aparición de comentarios según el mes y el año. Voy a crear cuatro dataframes agrupando según el año para ver como ha sido la distribuci ya que antes del 2019 no hay casi valoraciones y el 2023 solo hay comentarios de dos meses.

In [62]:

```
1 import seaborn as sb
```

In [63]:

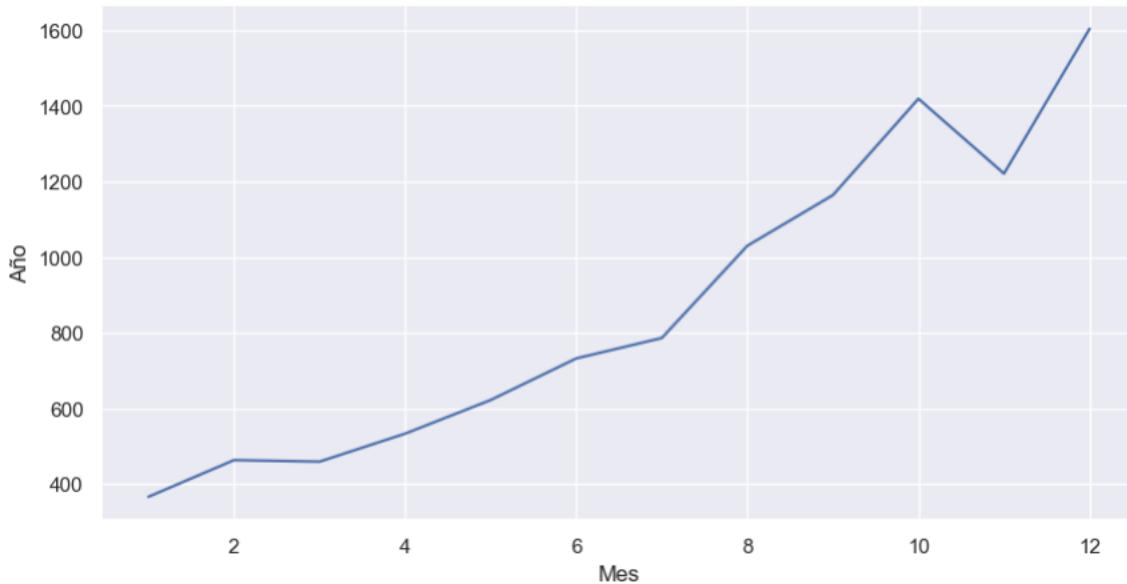
```
1 años = [2019, 2020, 2021, 2022]
2 df_19_22 = datos[datos['Año'].isin(años)]
```

In [64]:

```
1 df_19 = datos[datos.Año==2019]
2 df_20 = datos[datos.Año==2020]
3 df_21 = datos[datos.Año==2021]
4 df_22 = datos[datos.Año==2022]
```

In [65]:

```
1 comentarios_mes = df_19_22.groupby(['Mes']).count()
2 plt.figure(figsize=(10, 5))
3 sb.set()
4 ax = sb.lineplot(x=comentarios_mes.index, y=comentarios_mes["Año"])
```



Podemos observar una gráfica con muy pocas valoraciones en enero y podemos ver como sube de forma continua, si que hay una bajada en el mes de noviembre pero en en 1600.

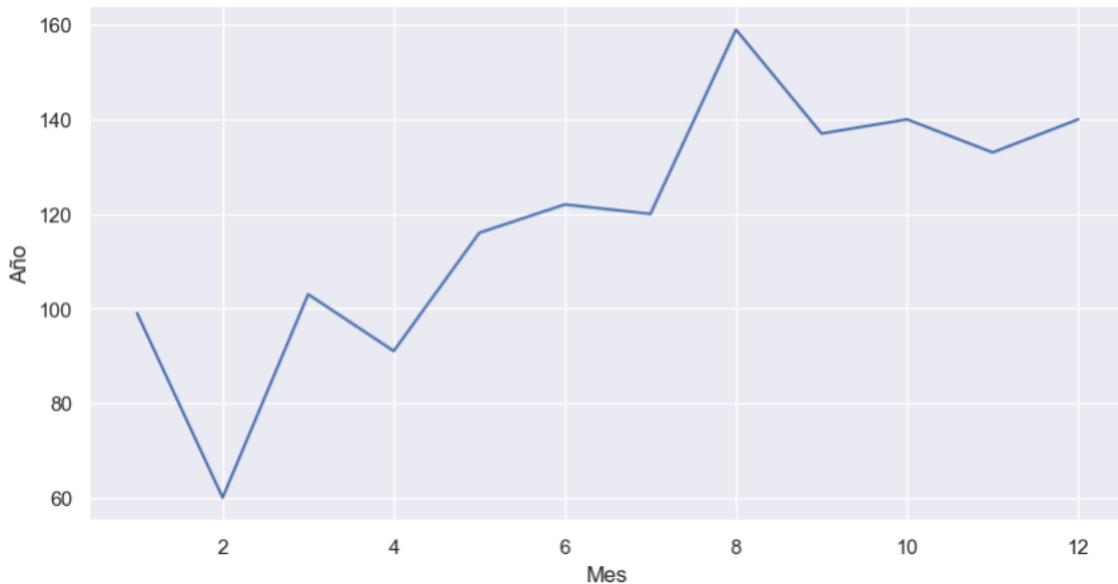
Esta gráfica no se explica muy bien ya que habla sobre la suma de las valoraciones de cuatro años seguidos observado segun cada mes. Nos encontramos en un evento n años, el confinamiento de la Covid-19. Estoy seguro que en las gráficas posteriores vamos a observar meses enteros sin ninguna valoración. Este fenómeno explica la alte

In [66]:

```

1 comentarios_año = df_19.groupby(['Mes']).count()
2
3 plt.figure(figsize=(10, 5))
4 sb.set()
5 ax = sb.lineplot(x=comentarios_año.index, y=comentarios_año["Año"])

```



En el 2019 podemos ver como la gráfica es ascendente des del mes de febrero. El mes que tiene más comentarios es el agosto, dato lógico ya que verano y navidades so tanto, es normal que tengan más comentarios.

Podemos apreciar como después de verano la gráfica se mantiene hasta diciembre. Lo más lógico es que vuelva a ascender la frecuencia por ser navidades.

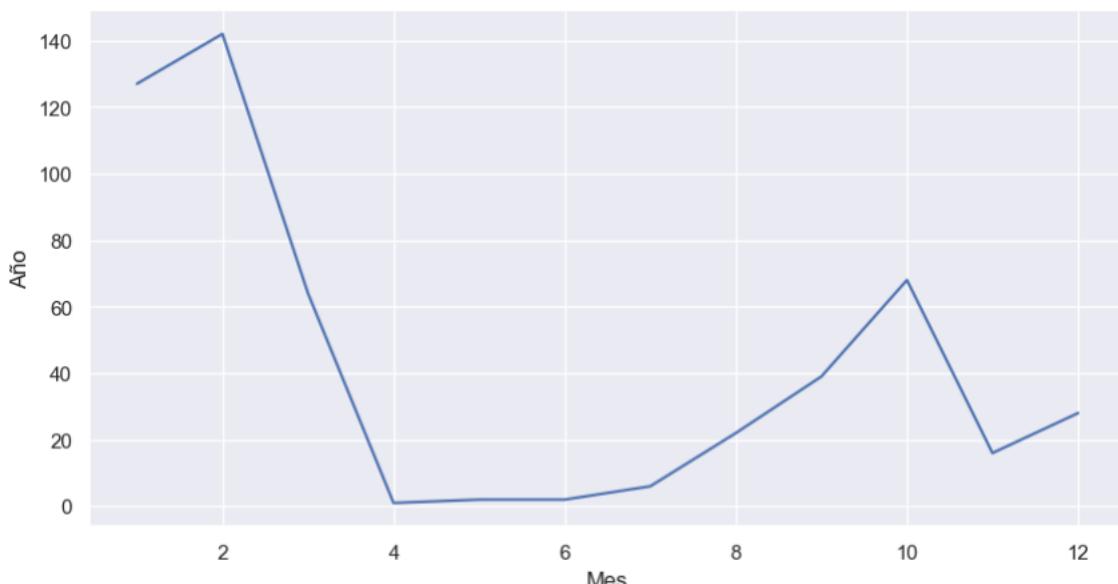
Podemos apreciar una correlación entre la demanda y el número de comentarios, este hecho lo voy a confirmar a medida que vaya viendo el comportamiento de las otras c

In [67]:

```

1 comentarios_año = df_20.groupby(['Mes']).count()
2
3 plt.figure(figsize=(10, 5))
4 sb.set()
5 ax = sb.lineplot(x=comentarios_año.index, y=comentarios_año["Año"])

```



En esta gráfica podemos ver claramente los efectos de la Covid-19 y como el confinamiento del año 2020 afectó al número de valoraciones. Podemos apreciar como en en hay una bajada a 0 comentarios de una forma directa. Hasta en julio no vemos ningún comentario y en octubre hay un pequeño pico pero que no se puede comparar al aña

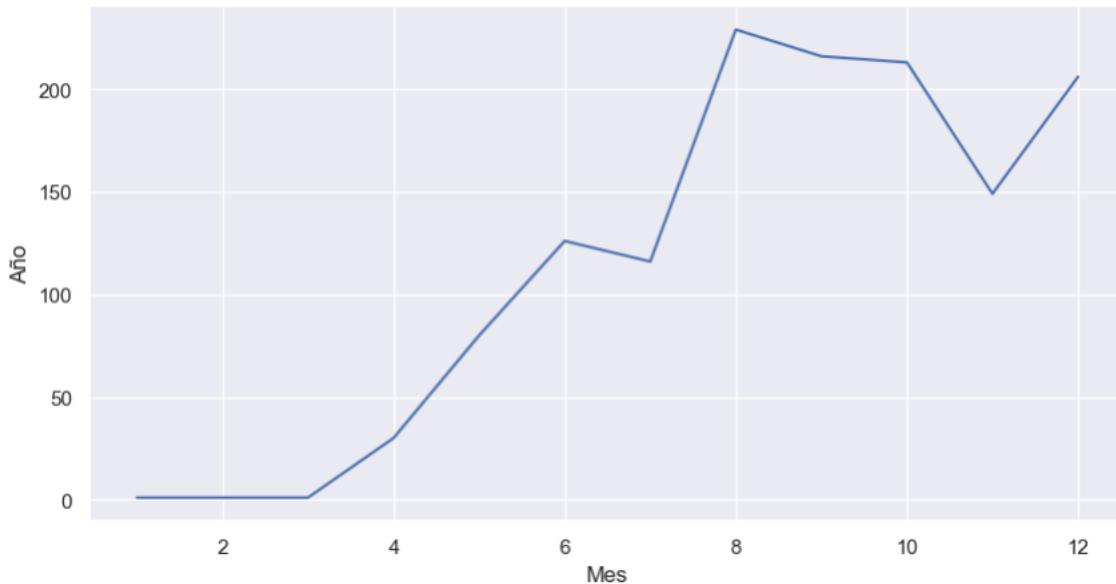
Otra vez podemos concluir que la correlación entre ocupación y número de comentarios tiene mucha lógica.

In [68]:

```

1 comentarios_año = df_21.groupby(['Mes']).count()
2
3 plt.figure(figsize=(10, 5))
4 sb.set()
5 ax = sb.lineplot(x=comentarios_año.index, y=comentarios_año["Año"])

```



Esta gráfica es muy interesante ya que podemos explicar muchos eventos. Empieza el año en 0 comentarios, hecho que me hace pensar que vivimos un rebrote y las medidas de confinamiento o medidas contra el sector.

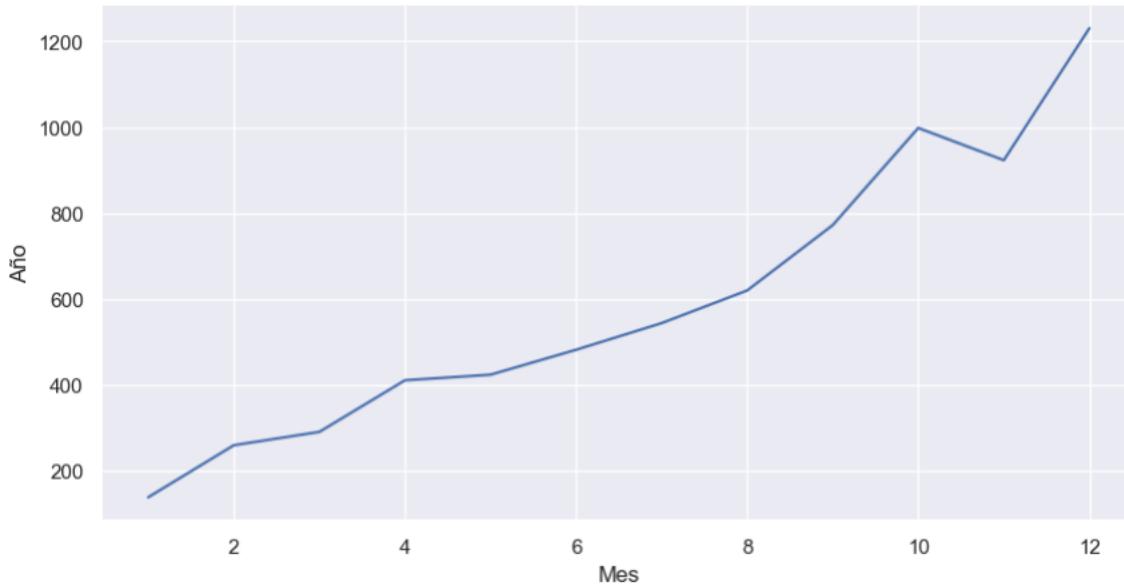
Una vez solucionado el rebrote las valoraciones aumentaron y vemos un gran pico positivo en el mes de agosto. El número de comentarios del mes de agosto del año 2021 supone un aumento de la ocupación por culpa de las medidas contra la Covid-19.

In [69]:

```

1 comentarios_año = df_22.groupby(['Mes']).count()
2
3 plt.figure(figsize=(10, 5))
4 sb.set()
5 ax = sb.lineplot(x=comentarios_año.index, y=comentarios_año["Año"])

```



En el año 2022 observamos una gráfica ascendente de enero a diciembre donde el pico, esta vez, está en navidades. Curiosamente asciende muy continuamente, sin picos.

Si nos fijamos más detalladamente podemos ver que la escala del eje Y siempre ha sido de 0 a 250 con mucho y ahora estamos hablando que llega hasta más de 1200. Esta gráfica no tiene un comportamiento similar a las anteriores donde afirmaba una correlación entre ocupación y número de comentarios, podemos ver como el número de va mayor que anteriormente.

Dataset positivo

In [70]:

```
1 fig = plt.figure(figsize=(10, 10), dpi=80)
2 ax = fig.add_subplot(1, 1, 1)
3 long_string = ','.join(list(datos_positivo['clean_text'].values))
4 wordcloud = WordCloud(width=600, height=300, background_color="white", max_words=50, contour_width=0, contour_color='steelblue')
5 wordcloud.generate(long_string)
6 ax.imshow(wordcloud, interpolation='bilinear')
7 ax.axis("off")
8 plt.show()
```



Nos encontramos las palabras más utilizadas en las valoraciones positivas. Podemos ver como las palabras más grandes son similares a las del estudio general pero las más claramente, aportan un sentimiento positivo. Podemos apreciar "excellent", "wonderful", "lovely", "beautiful"...

- Análisis del histograma de las palabras que salen con más frecuencia

In [71]:

```
1 lista_tokens = list()
2 for i in datos_positivo['clean_text']:
3     # Tokenizamos cada documento con word_tokenize()
4     tokens_document = word_tokenize(i)
5     # Añadimos esos tokens como nuevos elementos
6     # Si usamos append se crearía una lista de Listas, de este modo añadimos Los
7     lista_tokens.extend(tokens_document)
```

Tn [72]:

```
1 %%time  
2 dict freq = Counter(lista tokens)
```

Wall time: 48.9 ms

Tn [73]:

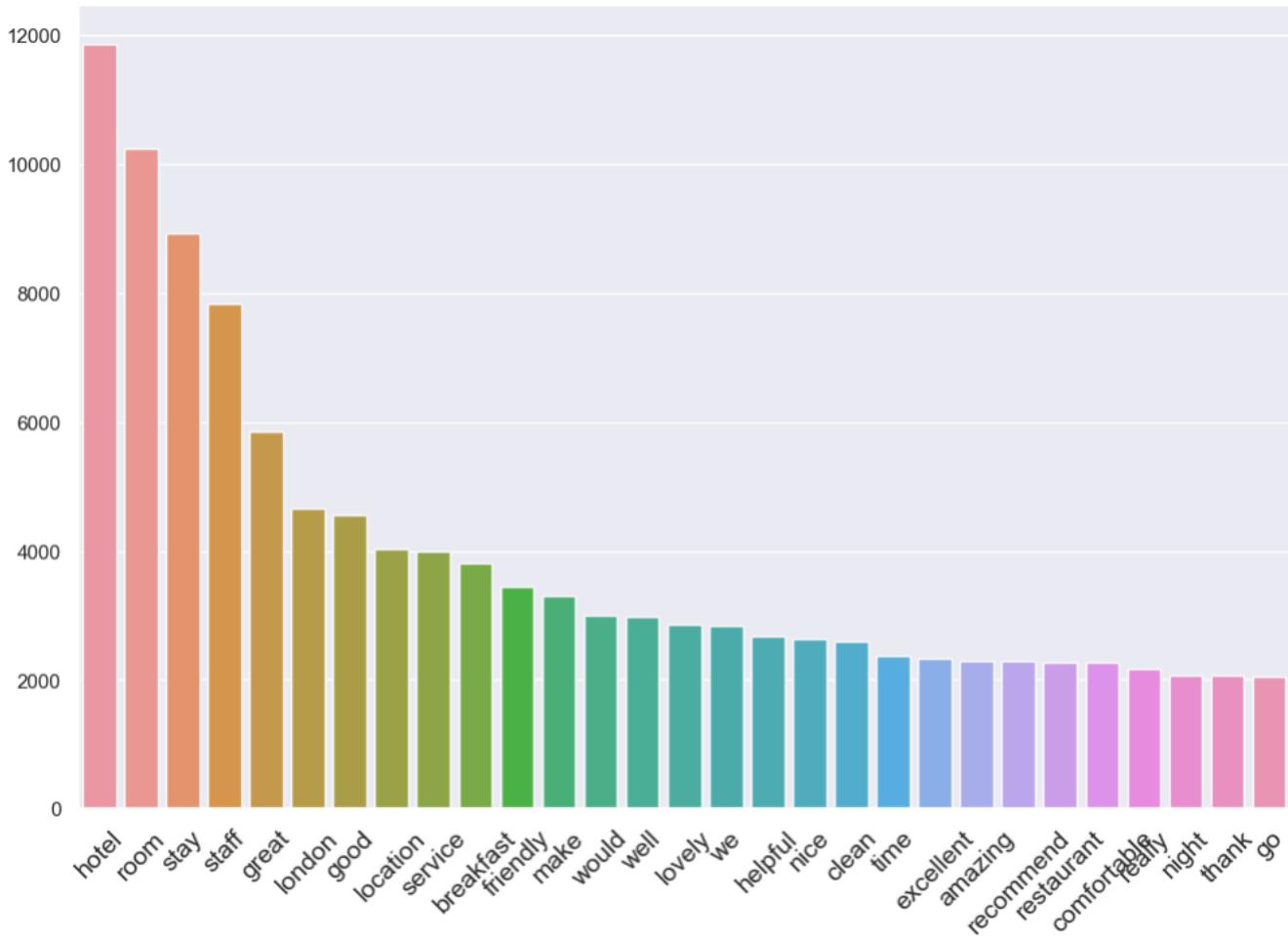
```
1 # Ordenamos el diccionario por la frecuencia de sus palabras
2 dict_freq_order = sorted(dict_freq.items(), key=lambda x: x[1], reverse=True)
3 token_names = list()
4 token_freqs = list()
5 for i in dict_freq_order:
6     if i[1] > 2000:
7         token_names.append(i[0])
8         token_freqs.append(i[1])
```

In [74]:

```

1 plt.rcParams['figure.figsize'] = [12,8]
2 sns_g = sns.barplot(x=token_names, y=token_freqs)
3 plt.xticks(rotation=45)
4 plt.tick_params(axis='x', which='major', labelsize=14)

```



En el dataset positivo podemos ver como las palabras que salen con más frecuencia son muy similares a las del análisis general.

- Análisis de los bigramas y trigramas

* Bigrama

In [75]:

```

1 datos_positivo["bigrama"] = datos_positivo["clean_text"].swifter.apply(get_ngrams, n=2)

```

Pandas Apply: 0% | 0/10856 [00:00<?, ?it/s]

In [76]:

```

1 bigram_list = datos_positivo["bigrama"].tolist()
2 review_str = ' '.join(bigram_list)

```

In [77]:

```

1 wordcloud = WordCloud( width = 600, height = 300, random_state=1, background_color='white',
2                         contour_color='steelblue', contour_width=0, max_words = 20, collocations=False,
3                         normalize_plurals=False).generate(review_str)

```

In [78]:

1 plot_cloud(wordcloud)



En los bigramas vemos con claridad asociaciones de palabras como come_back, highly_recommend, friendly_helpful, hotel_great... Muchas de estas palabras las podíamos

* Trígrama

In [79]:

1 datos_positivo["trígrama"] = datos_positivo["clean_text"].swifter.apply(get_ngrams, n=3)

Pandas Apply: 0% | 0/10856 [00:00<?, ?it/s]

In [80]:

1 trigram_list = datos_positivo["trígrama"].tolist()
2 review_str2 = ' '.join(trigram_list)

In [81]:

1 wordcloud2 = WordCloud(width=600, height=300, random_state=1, background_color='white',
2 contour_color='steelblue', contour_width=0, max_words=20, collocations=False,
3 normalize_plurals=False).generate(review_str2)

In [82]:

1 plot_cloud(wordcloud2)



En los trigramas nos encontramos exactamente con la misma situación que los bigramas. Encontramos palabras muy interesantes como would_highly_recommend, hotel_

Dataset negativo

In [83]:

```

1 fig = plt.figure(figsize=(10, 10), dpi=80)
2 ax = fig.add_subplot(1, 1, 1)
3 long_string = ', '.join(list(datos_negativo['clean_text'].values))
4 wordcloud = WordCloud(width=600, height=300, background_color="white", max_words=50, contour_width=0, contour_color='steelblue')
5 wordcloud.generate(long_string)
6 ax.imshow(wordcloud, interpolation='bilinear')
7 ax.axis("off")
8 plt.show()

```



Nos encontramos las palabras más utilizadas en las valoraciones negativas. Se puede ver como hay palabras que marcan el sentido negativo como "poor", "cold", "dirty", "t" con más aparición que las mencionadas anteriormente que lo más probable es que estén asociadas.

- Análisis del histograma de las palabras que salen con más frecuencia

In [84]:

```

1 lista_tokens = list()
2 for i in datos_negativo['clean_text']:
3     # Tokenizamos cada documento con word_tokenize()
4     tokens_document = word_tokenize(i)
5     # Añadimos esos tokens como nuevos elementos
6     # Si usamos append se crearía una lista de listas, de este modo añadimos los
7     lista_tokens.extend(tokens_document)

```

In [85]:

```

1 %%time
2 dict_freq = Counter(lista_tokens)

```

Wall time: 996 µs

In [86]:

```

1 # Ordenamos el diccionario por la frecuencia de sus palabras
2 dict_freq_order = sorted(dict_freq.items(), key=lambda x: x[1], reverse=True)
3 token_names = list()
4 token_freqs = list()
5 for i in dict_freq_order:
6     if i[1] > 8:
7         token_names.append(i[0])
8         token_freqs.append(i[1])

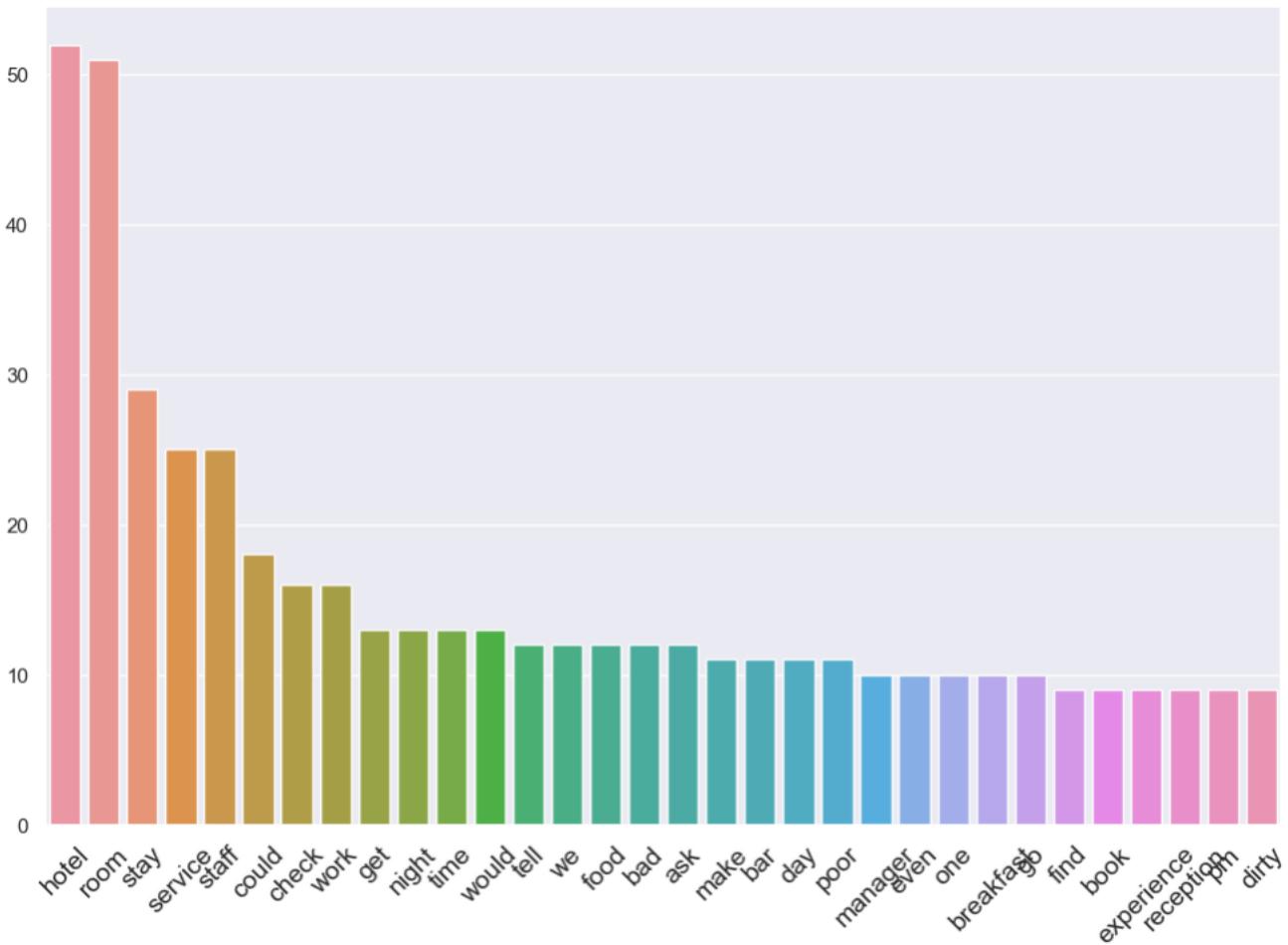
```

In [87]:

```

1 plt.rcParams['figure.figsize'] = [12,8]
2 sns_g = sns.barplot(x=token_names, y=token_freqs)
3 plt.xticks(rotation=45)
4 plt.tick_params(axis='x', which='major', labelsize=14)

```



Más o menos hay las mismas palabras que antes pero encontramos palabras muy interesantes como podría ser "check" o "dirty" que son la primera vez que salen. Aquí sí que no están en el estudio general. El motivo es muy probable que sea por la poca representación de reviews negativas que hay en el corpus.

- Análisis de los bigramas y trigramas

Voy a crear bigramas y trigramas para observar si agrupando los tokens en dos y tres elementos hay una mejor comprensión del corpus.

* Bigrama

In [88]:

```
1 datos_negativo["bigrama"] = datos_negativo["clean_text"].swifter.apply(get_ngrams, n=2)
```

Pandas Apply: 0% | 0/47 [00:00<?, ?it/s]

In [89]:

```

1 bigram_list = datos_negativo["bigrama"].tolist()
2 review_str = ' '.join(bigram_list)

```

In [90]:

```

1 wordcloud = WordCloud( width = 600, height = 300, random_state=1, background_color='white',
2                         contour_color='steelblue', contour_width=0, max_words = 20, collocations=False,
3                         normalize_plurals=False).generate(review_str)

```

In [91]:

```
1 plot_wordcloud(wordcloud)
```



Claramente se puede ver la mala experiencia de los clientes. Podemos ver en grande star_hotel, can_not y would_recommend (esta es interesante, me voy a fijar bien en la interpretación contraria). En pequeño se puede apreciar como small_room, room_ready, bad_experience, air_conditioning... Muestran los factores que hacen bajar la valoración.

* Trigrama

In [92]:

```
1 datos_negativo["trigrama"] = datos_negativo["clean_text"].swifter.apply(get_ngrams, n=3)
```

Pandas Apply: 0% | 0/47 [00:00<?, ?it/s]

In [93]:

```
1 trigram_list = datos_negativo["trigrama"].tolist()
2 review_str2 = ' '.join(trigram_list)
```

In [94]:

```
1 wordcloud2 = WordCloud( width = 600,height = 300, random_state=1, background_color='white',
2                         contour_color='steelblue', contour_width=0, max_words = 20, collocations=False,
3                         normalize_plurals=False).generate(review_str2)
```

In [95]:

```
1 plot_wordcloud(wordcloud2)
```



En los trigramas nos encontramos elementos muy interesantes como por ejemplo `five_star_hotel`. Me hace pensar que las valoraciones relacionadas con este trígrama expaganado un hotel de cinco estrellas. Podemos ver también como `cleanliness`, `room`, `blood`, `room`, `city`, `view` y `ask`, `hotel`, `manager` son palabras muy utilizadas en este tipo de

Análisis cualitativo

- Dataset general

Voy a mostrar las valoraciones más positivas y más negativas que hay en el corpus.

In [96]:

```
1 top_positivo = datos[datos['sentimiento'] == 100].iloc[3]
```

In [97]:

```
1 top_negativo = datos[datos['sentimiento'] == -65]
```

In [98]:

```
1 top_positivo['Review']
```

Out[98]:

'Confortable, superbly located, well managed and with a delicious breakfast. What else could one wish for? It was better than expected. I'll stay there again without any doubts.'

In [99]:

```
1 top_negativo['Review'].values
```

Out[99]:

```
array(['Awful customer service at reception. Reception manager was extremely judgmental and unhelpful, with a terrible attitude. Felt very
be booking again. Avoid at all costs.'],
      dtype=object)
```

Aquí podemos leer la review con el valor del sentimiento más alto y la que tiene el más bajo. El análisis es correcto ya que los dos comentarios van muy relacionados con s

Vamos ver lo mismo pero haciendo una comparación de varios años distintos para ver que se ha valorado más en cada año y que se ha valorado menos.

In [100]:

```
1 top_positivo = datos[datos['Año'] == 2023].loc[datos[datos['Año'] == 2023]['sentimiento'].idxmax()]
2 print(top_positivo['Review'])
```

We booked a residence and as ever was like going home..wonderful from start to finish..a must for anyone with kids that wants to be in the room..recomend the wonderful breakfast delivered to your room

In [101]:

```
1 top_negativo = datos[datos['Año'] == 2023].loc[datos[datos['Año'] == 2023]['sentimiento'].idxmin()]
2 print(top_negativo['Review'])
```

We are disappointed of the hotel. This cannot be a five star hotel in chilly London without any heating in the bathroom and such poor , no so, the rooms are not renovated. Not a value for money hotel! Does not worth the cost of ~ 450& a night!

In [102]:

```
1 top_positivo = datos[datos['Año'] == 2022].loc[datos[datos['Año'] == 2022]['sentimiento'].idxmax()]
2 print(top_positivo['Review'])
```

We like to stay Biltmore everytime we visit London which is very often and we are always treated wonderfully. Rooms are renovated and spacious is always there for to take care your comfort.

In [103]:

```
1 top_negativo = datos[datos['Año'] == 2022].loc[datos[datos['Año'] == 2022]['sentimiento'].idxmin()]
2 print(top_negativo['Review'])
```

Awful customer service at reception. Reception manager was extremely judgmental and unhelpful, with a terrible attitude. Felt very sorry for being again. Avoid at all costs.

In [104]:

```
1 top_positivo = datos[datos['Año'] == 2021].loc[datos[datos['Año'] == 2021]['sentimiento'].idxmax()]
2 print(top_positivo['Review'])
```

Sloane Sq Hotel, is tucked away very neatly on Sloane Sq. My stay here was delightful. Great staff and and hotel. The rooms have a lot of lights sleep! The location.... well what more could you ask for with Cartier on your door step and your spoilt for choice when it comes to cl

In [105]:

```
1 top_negativo = datos[datos['Año'] == 2021].loc[datos[datos['Año'] == 2021]['sentimiento'].idxmin()]
2 print(top_negativo['Review'])
```

Disgusting! I got shocked and surprised when walking through the corridor and saw that the hotel uses the same lift to transport foods and stuff was throwing dirty bed linen over the food trolley while the room service was coming out at the same time. I couldn't see health and s

In [106]:

```
1 top_negativo = datos[datos['Año'] == 2020].loc[datos[datos['Año'] == 2020]['sentimiento'].idxmin()]
2 print(top_negativo['Review'])
```

They found an unexploded WWII bomb under the hotel and we were all forced to leave at no notice. We weren't looked after well following the hotel. Terrible treatment of guests! No refund - nothing - not even a response after numerous emails!

In [107]:

```
1 top_negativo = datos[datos['Año'] == 2020].loc[datos[datos['Año'] == 2020]['sentimiento'].idxmin()]
2 print(top_negativo['Review'])
```

They found an unexploded WWII bomb under the hotel and we were all forced to leave at no notice. We weren't looked after well following the hotel. Terrible treatment of guests! No refund - nothing - not even a response after numerous emails!

7. Topic Modeling

LDA

- Vectorizamos el dataset

In [108]:

```

1 def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
2     """https://spacy.io/api/annotation"""
3     texts_out = []
4     for sent in texts:
5         doc = nlp(" ".join(sent))
6         texts_out.append(" ".join([token.lemma_ if token.lemma_ not in ['-PRON-'] else '' for token in doc if token.pos_ in allowed_pos]))
7     return texts_out

```

In [109]:

```

1 # Do Lemmatization keeping only Noun, Adj, Verb, Adverb
2 data_lemmatized = lemmatization(datos['normaliza'], allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV'])
3
4 print(data_lemmatized[:2])

```

['waterloo train station breakfast good room small clean bed cozy free wifi free coffee tea room free coffee hot choco tea lobby fridge mi
tel bar restaurant serve delicious pizza chicken wing overall recommend hotel', 'book ideally locate trip hotel exceptionally clean staff
room ready room ready panic lock storage room room view city shard good size really great walk shower complimentary breakfast adequate sho
moan least important issue difficult couple year chill definitely stay']

In [110]:

```

1 vectorizer = CountVectorizer(analyzer='word',
2                             min_df=10,                                     # minimum reqd occurrences of a word
3                             stop_words='english',                           # remove stop words
4                             token_pattern='[a-zA-Z0-9]{3,}',               # num chars > 3
5                             # max_features=50000,                            # max number of uniq words
6                             )
7
8 data_vectorized = vectorizer.fit_transform(data_lemmatized)

```

In [111]:

```

1 # Materialize the sparse data
2 data_dense = data_vectorized.todense()
3
4 # Compute Sparsicity = Percentage of Non-Zero cells
5 print("Sparsicity: ", ((data_dense > 0).sum() / data_dense.size) * 100, "%")

```

Sparsicity: 1.1053963444568053 %

- El modelo LDA

In [112]:

```

1 from sklearn.decomposition import LatentDirichletAllocation, TruncatedSVD
2 from sklearn.model_selection import GridSearchCV

```

In [113]:

```

1 # Build LDA Model
2 lda_model = LatentDirichletAllocation(n_components=20,                                # Number of topics
3                                       max_iter=10,                                 # Max Learning iterations
4                                       learning_method='online',                   # Random state
5                                       random_state=100,                            # n docs in each learning iter
6                                       batch_size=128,                            # compute perplexity every n iters, default: Don't
7                                       evaluate_every = -1,                         # Use all available CPUs
8                                       n_jobs = -1,
9                                       )
10 lda_output = lda_model.fit_transform(data_vectorized)
11
12 print(lda_model) # Model attributes

```

LatentDirichletAllocation(learning_method='online', n_components=20, n_jobs=-1, random_state=100)

In [114]:

```

1 # Log Likelihood: Higher the better
2 print("Log Likelihood: ", lda_model.score(data_vectorized))
3
4 # Perplexity: Lower the better. Perplexity = exp(-1. * log-Likelihood per word)
5 print("Perplexity: ", lda_model.perplexity(data_vectorized))
6
7 # See model parameters
8 print(lda_model.get_params())

```

Log Likelihood: -2626299.110248469
Perplexity: 731.426389214339
{'batch_size': 128, 'doc_topic_prior': None, 'evaluate_every': -1, 'learning_decay': 0.7, 'learning_method': 'online', 'learning_offset': 0, 'max_iter': 10, 'mean_change_tol': 0.001, 'n_components': 20, 'n_jobs': -1, 'perp_tol': 0.1, 'random_state': 100, 'topic_word_prior': None, 'use': 0}

- Realizamos un GridSearch del modelo LDA

In [115]:

```

1 # Define Search Param
2 search_params = {'n_components': [4, 8, 12, 15], 'learning_decay': [.5, .7, .9]}
3
4 # Init the Model
5 lda = LatentDirichletAllocation()
6
7 # Init Grid Search Class
8 model = GridSearchCV(lda, param_grid=search_params)
9
10 # Do the Grid Search
11 model.fit(data_vectorized)

```

Out[115]:

```
GridSearchCV(estimator=LatentDirichletAllocation(),
            param_grid={'learning_decay': [0.5, 0.7, 0.9],
                        'n_components': [4, 8, 12, 15]})
```

In [116]:

```

1 # Best Model
2 best_lda_model = model.best_estimator_
3
4 # Model Parameters
5 print("Best Model's Params: ", model.best_params_)
6
7 # Log Likelihood Score
8 print("Best Log Likelihood Score: ", model.best_score_)
9
10 # Perplexity
11 print("Model Perplexity: ", best_lda_model.perplexity(data_vectorized))

```

Best Model's Params: {'learning_decay': 0.9, 'n_components': 4}
Best Log Likelihood Score: -523132.09337497223
Model Perplexity: 557.6146240911819

- El tópico dominante de cada comentario

In [117]:

```

1 # Create Document - Topic Matrix
2 lda_output = best_lda_model.transform(data_vectorized)
3
4 # column names
5 topicnames = ["Topic" + str(i) for i in range(best_lda_model.n_components)]
6
7 # index names
8 docnames = ["Doc" + str(i) for i in range(len(datos))]
9
10 # Make the pandas dataframe
11 df_document_topic = pd.DataFrame(np.round(lda_output, 2), columns=topicnames, index=docnames)
12
13 # Get dominant topic for each document
14 dominant_topic = np.argmax(df_document_topic.values, axis=1)
15 df_document_topic['dominant_topic'] = dominant_topic
16
17 # Styling
18 def color_green(val):
19     color = 'green' if val > .1 else 'black'
20     return 'color: {col}'.format(col=color)
21
22 def make_bold(val):
23     weight = 700 if val > .1 else 400
24     return 'font-weight: {weight}'.format(weight=weight)
25
26 # Apply Style
27 df_document_topics = df_document_topic.head(15).style.applymap(color_green).applymap(make_bold)
28 df_document_topics

```

Out[117]:

	Topic0	Topic1	Topic2	Topic3	dominant_topic
Doc0	0.010000	0.010000	0.800000	0.180000	2
Doc1	0.010000	0.480000	0.510000	0.010000	2
Doc2	0.010000	0.270000	0.710000	0.010000	2
Doc3	0.070000	0.160000	0.760000	0.010000	2
Doc4	0.250000	0.330000	0.410000	0.010000	2
Doc5	0.010000	0.010000	0.970000	0.010000	2
Doc6	0.950000	0.020000	0.020000	0.020000	0
Doc7	0.010000	0.160000	0.660000	0.170000	2
Doc8	0.690000	0.040000	0.260000	0.010000	0
Doc9	0.010000	0.300000	0.680000	0.010000	2
Doc10	0.010000	0.300000	0.580000	0.120000	2
Doc11	0.010000	0.250000	0.740000	0.010000	2
Doc12	0.010000	0.010000	0.970000	0.010000	2
Doc13	0.250000	0.460000	0.280000	0.000000	1
Doc14	0.010000	0.180000	0.800000	0.010000	2

Aquí podemos ver la importancia de cada tópico según el comentario. En la mayoría de los casos en cada valoración existe un porcentaje de aparición de cada tópico. Observa con el tópico con más probabilidad de acierto.

Por ejemplo, en el Doc0, encontramos como predominan dos tópicos, el 1 y el 3, pero el 0 y el 2 también tienen representación. Claramente el tópico dominante es el 3 con

- Número de apariciones de cada tópico

In [118]:

```

1 df_topic_distribution = df_document_topic['dominant_topic'].value_counts().reset_index(name="Num Documents")
2 df_topic_distribution.columns = ['Topic Num', 'Num Documents']
3 df_topic_distribution

```

Out[118]:

	Topic Num	Num Documents
0	2	5010
1	0	3844
2	1	2052
3	3	1863

En esta tabla podemos apreciar el número de veces que ha salido un tópico en todo el corpus. Podemos decir que el tópico número 0 está en primera posición, seguido de

El primer tópico se queda el máximo de las apariciones de una forma significativa con unas 4670 veces de aparición, el tópico 1 está en segunda posición con 3016 y los tópicos respectivamente.

In [119]:

```

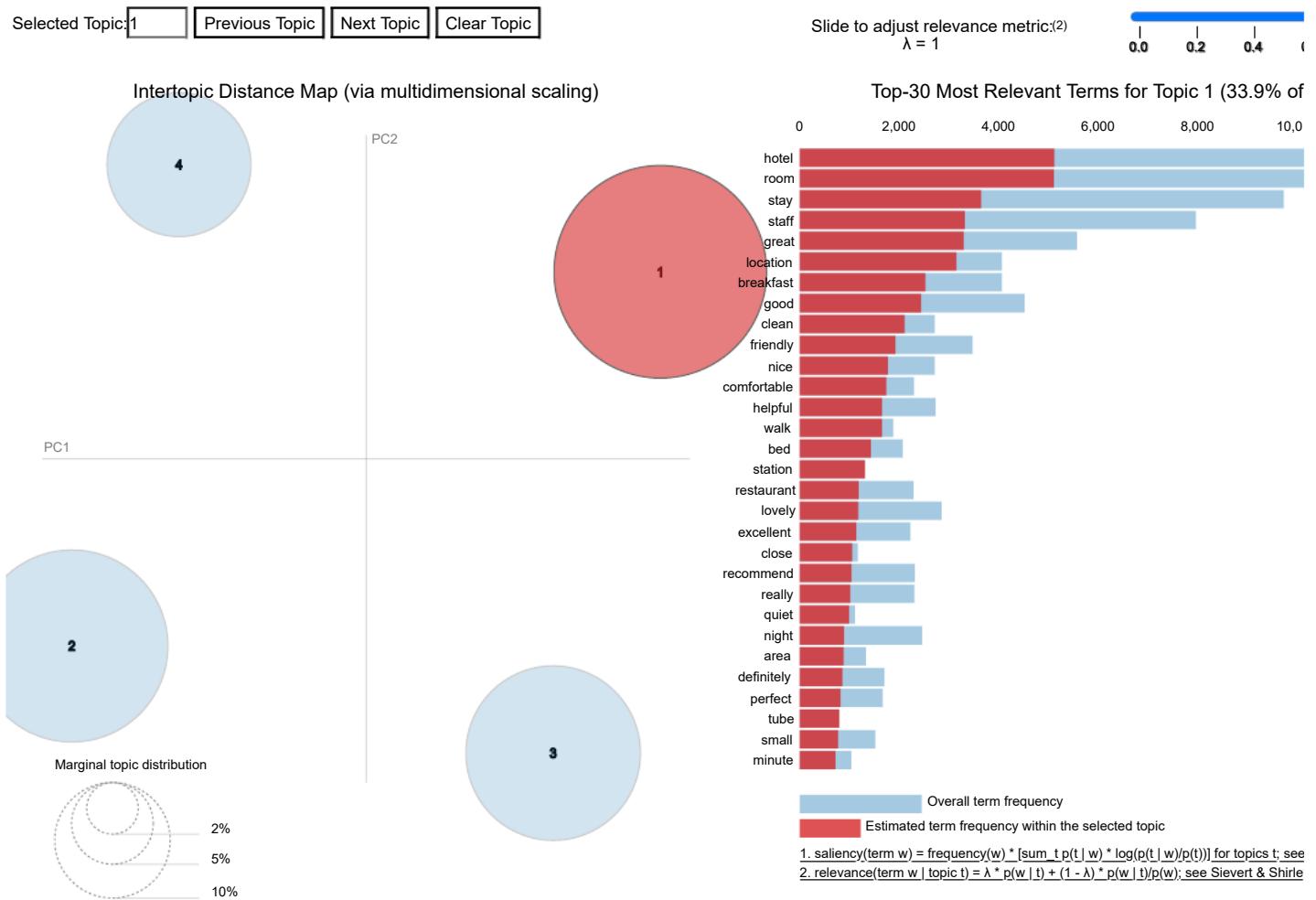
1 import pyLDAvis
2 import pyLDAvis.sklearn
3 import pyLDAvis.gensim_models
4
5 pyLDAvis.enable_notebook()
6 panel = pyLDAvis.sklearn.prepare(best_lda_model, data_vectorized, vectorizer, mds='tsne')
7 panel

```

C:\Users\xikix\anaconda3.2\lib\site-packages\past\builtins\misc.py:45: DeprecationWarning:

the imp module is deprecated in favour of importlib; see the module's documentation for alternative uses

Out[119]:



Aquí podemos ver dos gráficas complementarias y muy explicativas. En la izquierda encontramos el Intertopic Distance Map, donde cada burbuja pertenece a un tópico. Er salient Terms que nos muestra todas las palabras que tiene cada tópico y su frecuencia de aparición en el tópico determinado.

Podemos observar que el LDA realizado es un buen modelo ya que las burbujas de la gráfica de la izquierda están muy separadas. La intención es intentar encontrar grupo indicador de que al modelo le cuesta diferenciar entre los tópicos, ya que el solapamiento indica una similitud entre los tokens. Por otra parte, la dimensión de cada círculo 'en la review', con lo cual, cuanto más grande quiere decir que ese tópico ha salido en más reviews.

Apreciamos cuatro grandes grupos, formado por un conjunto de tokens. En el grupo 1, el que tiene la dimensión del círculo más grande, contiene el 31.9% de los tokens ha "hotel", "room", "stay", "staff" y "great" con un gran porcentaje de aparición de cada palabra. En el grupo dos, que contiene el 30.4% de los tokens, tiene una distribución muy similar de palabras como "stay", "staff", "hotel", "make" y "service" con mucho porcentaje de aparición. El grupo 3 tiene el 19.3% de tokens y solo tiene una palabra con un porcentaje "room". Contiene palabras como "room", "hotel", "check", "stay" y "breakfast". Aquí ya observamos como predominan otro tipo de palabras. En el tópico 4, con el 18.5% de tokens, sigue "hotel", seguido de palabras como "room", "stay", "service", "time"...

- En el tópico 1 se habla del hotel, la habitación y su localización. Encontramos las palabras que tienen más frecuencia de aparición y son los tokens más habituales en el tópico.
- En el tópico 2 se habla del equipo y el servicio que ofrece el hotel. Se relacionan con palabras muy positivas, con lo cual vamos a encontrar el tópico 2 en comentarios de reseñas.
- En el tópico 3 se habla de las habitaciones del hotel y cómo son. Encontramos "check", "bed", "clean" y "bathroom". Las reviews que hablan de cómo son las características.
- En el tópico 4 se habla de las habitaciones del hotel, el servicio, el equipo, la experiencia...

En general los tokens son muy similares unos con otros, hay elementos diferenciadores pero las "Top Words" son casi iguales para cada uno de ellos. Realmente me gusta entender mucho más a mi público objetivo. Por este motivo he decidido realizar otro modelo para profundizar más.

BERTopic

In [120]:

```
1 from bertopic import BERTopic
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\graph_objs\__init__.py:288: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\graph_objs\__init__.py:288: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\graph_objs\__init__.py:288: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\graph_objs\__init__.py:288: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.
```

In [121]:

```
1 from umap import UMAP
```

In [122]:

```
1 # Initiate UMAP
2 umap_model = UMAP(n_neighbors= 50,
3                     n_components= 10,
4                     min_dist=0.0,
5                     metric='cosine',
6                     random_state=100)
7 # Initiate BERTopic
8 topic_model = BERTopic(umap_model=umap_model, language="english", calculate_probabilities=True)
9 # Run BERTopic model
10 topics, probabilities = topic_model.fit_transform(datos['clean_text'])
11
12 topic_model.get_topic_info()
```

Out[122]:

Topic	Count	Name
0	-1	-1_room_hotel_stay_staff
1	0	0_great_staff_service_good
2	1	1_tea_afternoon_sandwich_scone
3	2	2_london_good_location_stay
4	3	3_birthday_celebrate_anniversary_special
...
65	64	64_apartment_130_gate_kensington
66	65	65_bed_comfy_comfortable_super
67	66	66_como_metropolitan_group_sutedja
68	67	67_quirky_suite_bolingbroke_batty
69	68	68_option_transport_london_location

70 rows × 3 columns

In [123]:

```

1 from hdbscan import HDBSCAN
2
3 # Clustering model
4
5 hdbscan_model = HDBSCAN(min_cluster_size=20, min_samples = 20, metric='euclidean', prediction_data=True)# Initiate BERTopic
6
7 topic_model = BERTopic(umap_model=umap_model, hdbscan_model=hdbscan_model, calculate_probabilities=True)# Run BERTopic model
8
9 topics, probabilities = topic_model.fit_transform(datos['clean_text'])# Get the list of topics
10
11 topic_model.get_topic_info()

```

Out[123]:

	Topic	Count	Name
0	-1	8601	-1_room_hotel_stay_staff
1	0	823	0_hotel_staff_location_great
2	1	405	1_london_good_location_great
3	2	375	2_tea_afternoon_sandwich_cake
4	3	297	3_birthday_savoy_special_celebrate
5	4	246	4_food_service_lunch_delicious
6	5	155	5_paddington_station_walk_hyde
7	6	143	6_covent_garden_trafalgar_square
8	7	136	7_tell_card_call_pay
9	8	118	8_marylebone_dorset_hotel_square
10	9	112	9_daughter_child_old_family
11	10	104	10_bed_pillow_comfortable_comfy
12	11	97	11_hyde_park_kensington_apartment
13	12	76	12_london_hotel_stay_experience
14	13	70	13_sloane_square_hotel_draycott
15	14	68	14_victoria_pimlico_station_breakfast
16	15	61	15_clermont_victoria_cross_char
17	16	59	16_cocktail_bar_drink_menu
18	17	55	17_shower_room_bath_bathroom
19	18	53	18_firmdale_knightsbridge_hotel_love
20	19	53	19_dorchester_promenade_afternoon_tea
21	20	52	20_buckingham_palace_westminster_park
22	21	51	21_beaumont_mayfair_stay_hotel
23	22	48	22_wife_husband_london_we
24	23	47	23_noise_noisy_sound_floor
25	24	44	24_hotel_room_beautiful_bath
26	25	44	25_berkeley_hotel_service_always
27	26	42	26_ritz_rivoli_experience_live
28	27	38	27_massage_spa_treatment_therapist
29	28	35	28_prince_akatoki_japanese_arch
30	29	34	29_waterloo_station_train_breakfast
31	30	33	30_rosewood_holborn_scarfes_london
32	31	32	31_henrietta_covent_garden_hotel
33	32	31	32_zetter_townhouse_clerkenwell_cocktail
34	33	30	33_room_smell_issue_night
35	34	30	34_langham_palm_artesian_court
36	35	26	35_mayfair_flemings_fleming_lorena
37	36	23	36_shower_sized_room_good
38	37	22	37_hilton_bankside_diamond_night

Podemos ver como el bertopic ha realizado muchos más tópicos que el LDA, en concreto hay 37 tópicos después del clustering. El elemento -1, el primero de todos, son los que el modelo no ha clasificado. Realmente son muchas, más de la mitad de la muestra.

Decir también que las veces que aparecen los tópicos es muy diversa, encontramos al primer elemento que tiene 1692 apariciones y el último que tiene 10 muestras. Esto agrupamientos de tópicos al hablar de la misma temática, pero espero que de forma distinta ya que por este motivo he desarrollado este modelo, para segmentar los temas mucho más.

- Tokens según el tópico

In [124]:

1 topic_model.get_topic(0)

Out[124]:

```
[('hotel', 0.02835508130850788),  
 ('staff', 0.02637198788803157),  
 ('location', 0.026108620470829288),  
 ('great', 0.02543446529187555),  
 ('helpful', 0.021626291811826036),  
 ('stay', 0.0215180116169612),  
 ('friendly', 0.021418190262895074),  
 ('good', 0.021222033300053244),  
 ('nice', 0.020079836024441976),  
 ('recommend', 0.01889440140678703)]
```

Podemos apreciar como el tópico con más aparición contiene las palabras que vemos. Ha agrupado "great", "staff", "good", "service", "hotel"... Esto nos determina que el tópico habla de temas relacionados con el hotel y cómo es, es decir, donde está situado o como es el equipo.

In [125]:

1 topic_model.get_topic(1)

Out[125]:

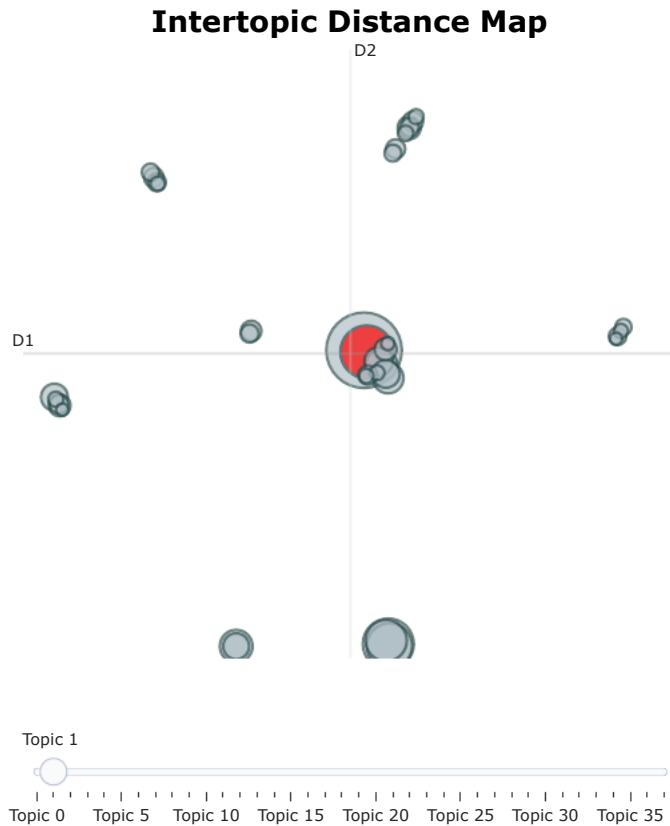
```
[('london', 0.05016294162359855),  
 ('good', 0.02977535263314355),  
 ('location', 0.0274399664495423),  
 ('great', 0.02673017652565895),  
 ('stay', 0.024178295660578115),  
 ('hotel', 0.023576064839623486),  
 ('recommend', 0.021163341632456457),  
 ('staff', 0.02018799216626245),  
 ('place', 0.01882549351271179),  
 ('would', 0.016759320785621274)]
```

En el segundo tópico observamos tokens como "London", "good", "location", "great"... Esto hace pensar que este tópico se encontrará en las reviews que hablan de forma generalizada sobre la ubicación.

- Visualización de la dimensión y la distancia de los tópicos

In [126]:

```
1 topic_model.visualize_topics()
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.
```



Podemos contemplar la distancia entre tópicos. Cada círculo es un tópico y observamos que hay un solapamiento importante, pero no me parece erróneo ya que mi intención es analizar y no podría llegar a realizarlo sin esta situación. Seguramente si hiperparametrizo mejor sacaría grupos más grandes y no sin solapamiento, ya que ahora mismo hice un análisis.

En la parte inferior del gráfico encontramos el tópico 0, el círculo más grande, junto a muchos de pequeños. En el tópico 0 se hablan de tokens como "hotel", "staff", "location", "London", "good", "location", "great" y "stay". Podemos ver como si que hablan de más o menos lo mismo pero de forma distinta, que es realmente lo que quería con los tópicos.

- Distribución de las palabras de cada tópico

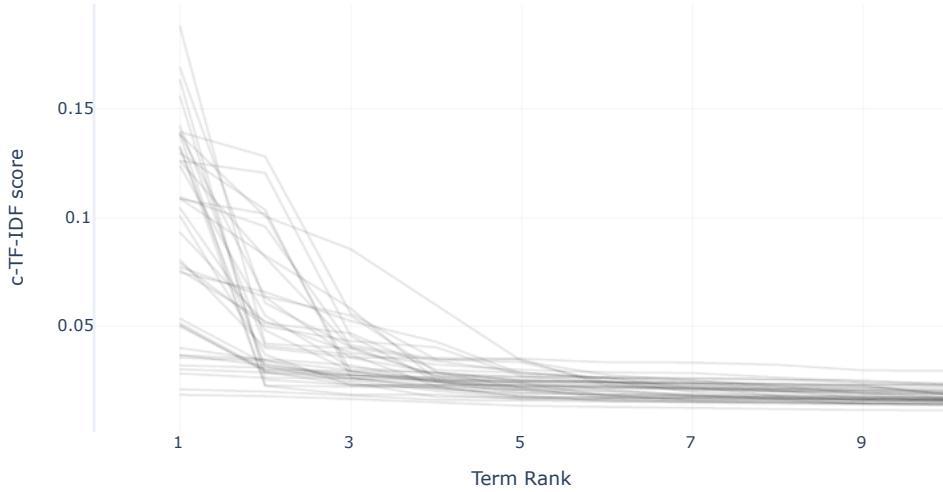
In [127]:

```
1 # Visualize term rank decrease
2 topic_model.visualize_term_rank()
```

```
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.
```

Term score decline per Topic



Al principio parece una visualización que es difícil de entender pero nos da la información para concluir con la duda del punto anterior.

Observamos como cada línea es un tópico y su comportamiento nos explica como es la contribución de cada palabra dentro del tópico. Nos muestra como de importantes : primera tiene todo el peso de la muestra, si está distribuida por toda la muestra...

- Como más recta sea la línea más distribución hay en el tópico, es decir, todas las palabras tienen un peso similar.
- Como más inclinada sea la representación más peso tendrá la palabra concreta del tópico.

Un ejemplo sería el tópico número 2, que está situado en lo más bajo de la gráfica. Confirmamos, gracias a la visualización anterior, con todas las palabras tienen el mismo

Otro ejemplo sería el tópico numero 61, que empieza en lo más alto. Esto quiere decir que la primera palabra tiene un gran peso. Al caer en picado podemos ver como el p

- Top palabras según el tópico

In [128]:

```

1 # Visualize top topic keywords
2 topic_model.visualize_barchart(top_n_topics=12)

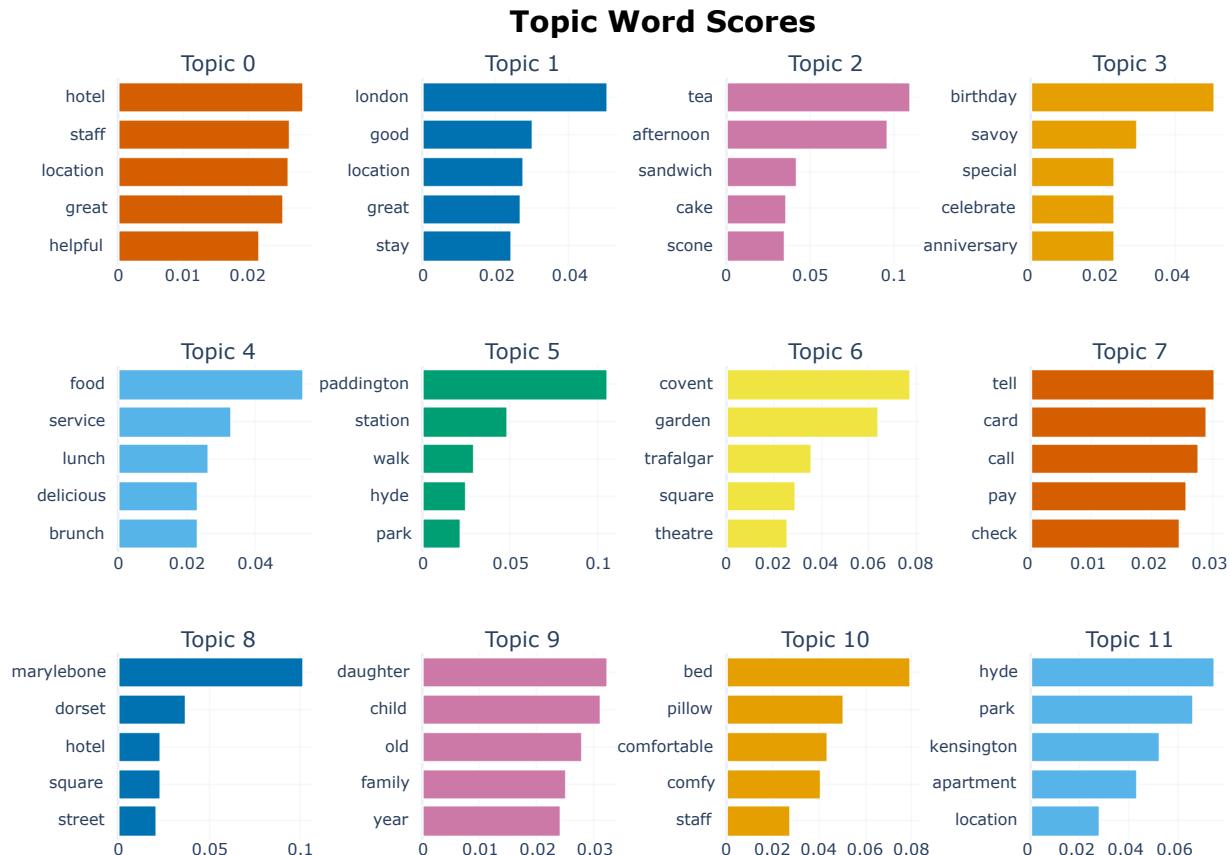
```

```

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

```



La información aportada es similar al punto anterior pero mucho más visual. Podemos observar los 12 mejores tópicos del modelo con sus cinco mejores palabras. Claramente podemos ver las palabras que forman cada tema pero hay elementos interesantes dentro de los histogramas. En algún tema hay palabras con mucha más representación que otras. En la normal en los temas o si son casos concretos.

Esta gráfica nos habla muy bien de como trabaja el modelo. Podemos observar como cada tema habla de forma similar sobre una temática independiente, como por ejemplo el tema 9 hablando de la familia.

- Visualización de las agrupaciones de los tópicos

In [129]:

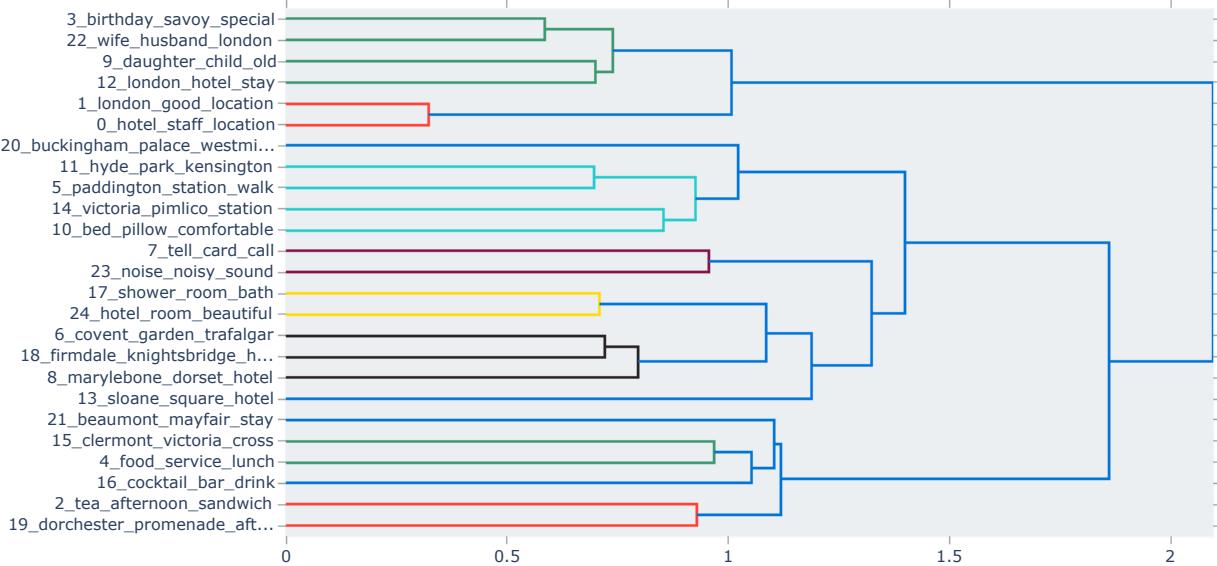
```

1 # Visualize connections between topics using hierarchical clustering
2 topic_model.visualize_hierarchy(top_n_topics=25)

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\figure_factory\_dendrogram.py:350: DeprecationWarning:
scipy.array is deprecated and will be removed in SciPy 2.0.0, use numpy.array instead
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\figure_factory\_dendrogram.py:351: DeprecationWarning:
scipy.array is deprecated and will be removed in SciPy 2.0.0, use numpy.array instead
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\figure_factory\_dendrogram.py:352: DeprecationWarning:
scipy.array is deprecated and will be removed in SciPy 2.0.0, use numpy.array instead
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\figure_factory\_dendrogram.py:353: DeprecationWarning:
scipy.array is deprecated and will be removed in SciPy 2.0.0, use numpy.array instead
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

```

Hierarchical Clustering



Aquí encontramos las relaciones entre los tópicos, donde cada color es un nivel de relación. Podemos observar como las conexiones tienen relación como, por ejemplo, el relacionados con la familia y celebraciones. Lo más probable es que los tópicos que hablan temas relacionados con la familia van relacionados con celebraciones o evento:

Cada relación pequeña va aumentando de tamaño al relacionarse cada vez más con grupos distintos. Por ejemplo el grupo de tópicos en color verde de arriba se relaciona como se relaciona la familia y los eventos festivos con el hotel, la localización, el staff...

- La similitud entre tópicos

In [130]:

```

1 # Visualize similarity using heatmap
2 topic_model.visualize_heatmap(n_clusters=5, top_n_topics=25)

```

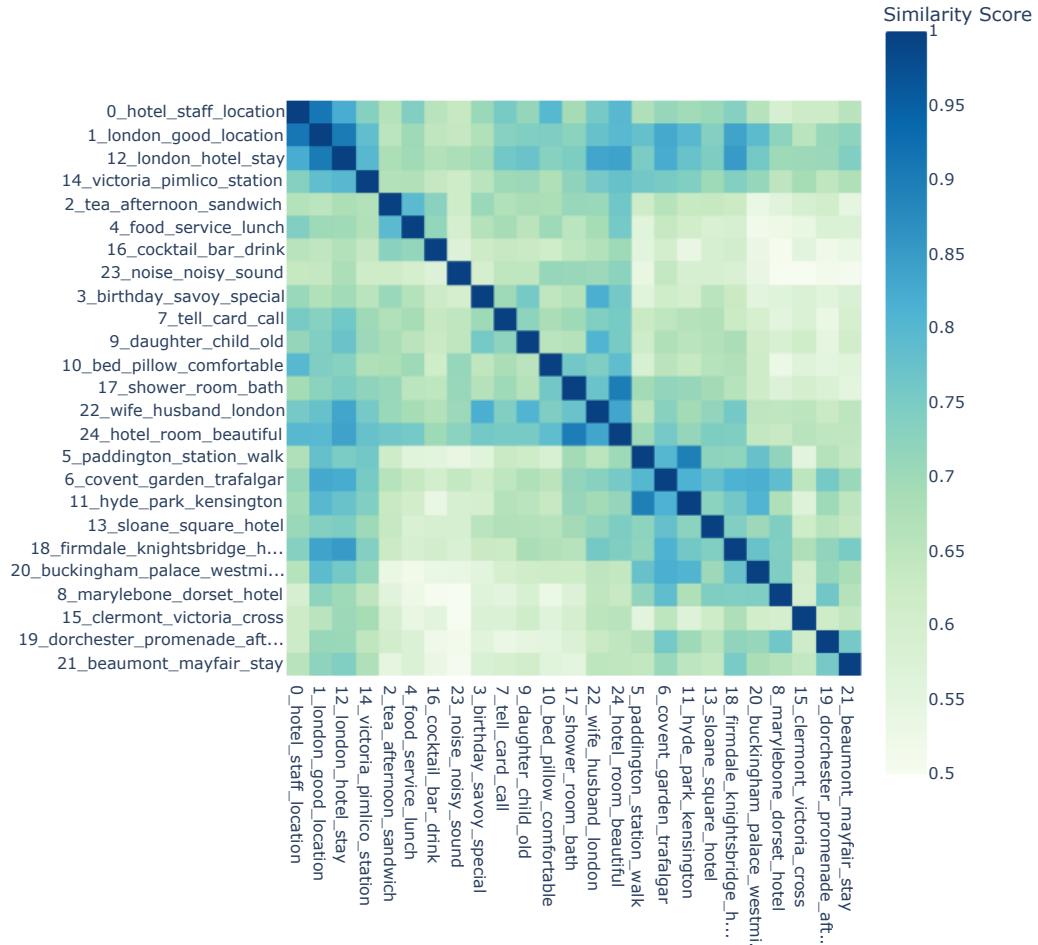
```

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

```

Similarity Matrix



En este corrplot podemos encontrar la relación de similitud que hay entre cada tópico, donde tanto si están en azul oscuro como en blanco estarán correladas, una en sentido contrario.

Esta visualización complementa muy bien con la anterior. Si nos fijamos en la gráfica anterior, observamos cómo el tópico 0 y 1 formaban una relación. Tienen una correlación de 0.89. Pasa lo mismo con el tópico 5 y el 11. En la gráfica anterior hay relación entre ellos y un 0.89.

- Topicos por tipo de clase según el tipo de sentimiento

In [131]:

```

1 topics_per_class = topic_model.topics_per_class(datos['clean_text'], classes=datos['tipo_sentimiento'])
2 fig_unsupervised = topic_model.visualize_topics_per_class(topics_per_class, top_n_topics=15)
3 fig_unsupervised

```

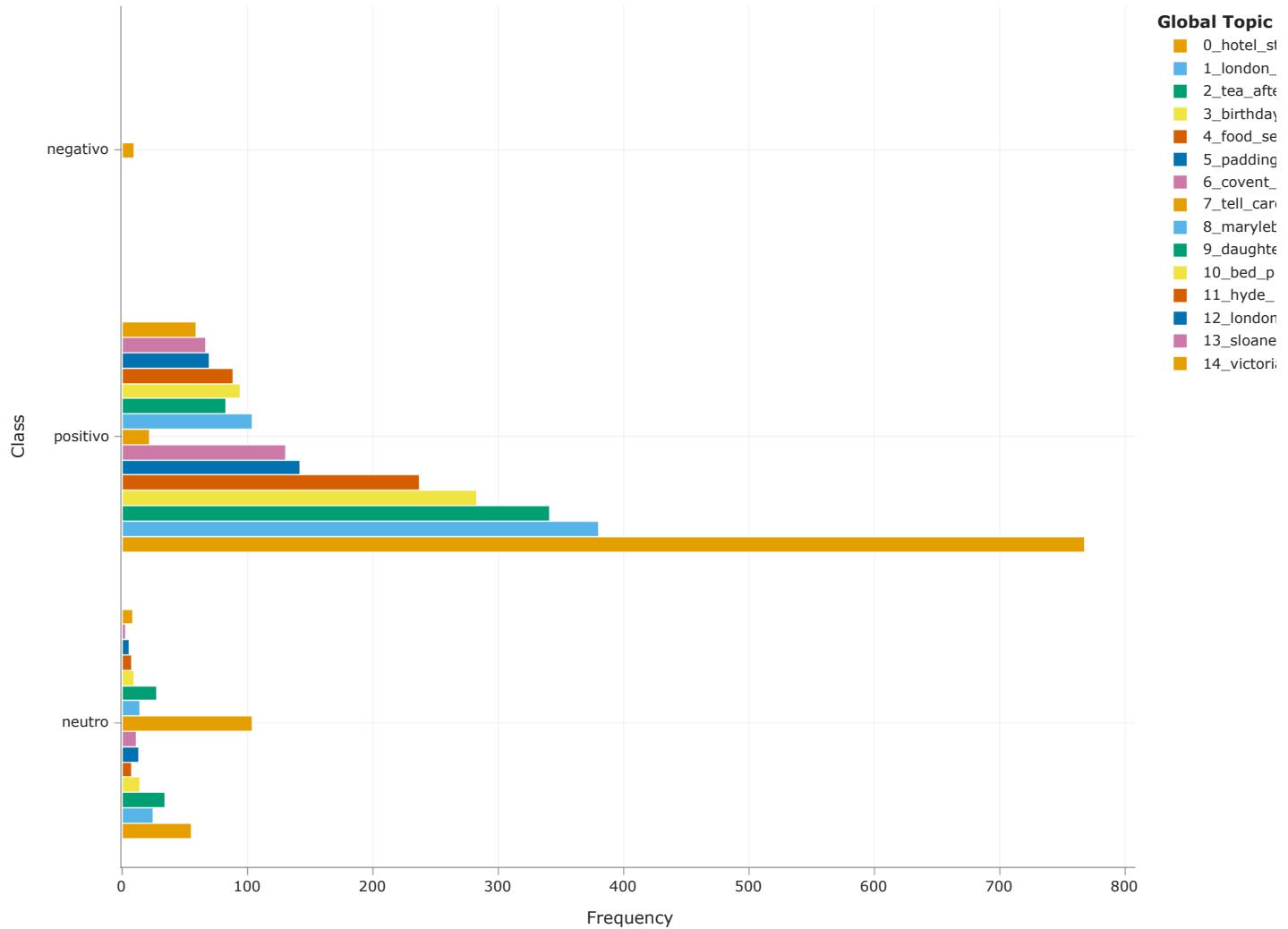
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io_renderers.py:395: DeprecationWarning:

distutils Version classes are deprecated. Use packaging.version instead.

Topics per Class



Aquí observamos la frecuencia de aparición de los 15 primeros tópicos según el tipo de sentimiento.

Podemos ver como en el sentimiento positivo lo que más aparece es el tópico relacionado con "hotel", "staff", "location", "great"... Lo que menos aparece es la forma de pago. La frecuencia está muy relacionada con elementos encontradas anteriormente. El segundo tópico con más aparición va relacionado con "London", "good", "location", "great"...

En el sentimiento neutro encontramos como lo más aparecido es el tópico relacionado con el pago. Los otros temas tienen una distribución parecida, pero destacamos este "staff", "location"...

En el sentido negativo, con muy poca representación, el elemento más aparecido se hablan de "tell", "manager", "terrible", "check" y "email". Si en el dataset hubiese más hablar mucho más de los temas que la gente habla sobre este tipo de sentimiento.

- Topic por año

In [132]:

```

1 timestamps = datos.Año.to_list()
2 topics_over_time = topic_model.topics_over_time(datos['clean_text'], timestamps)

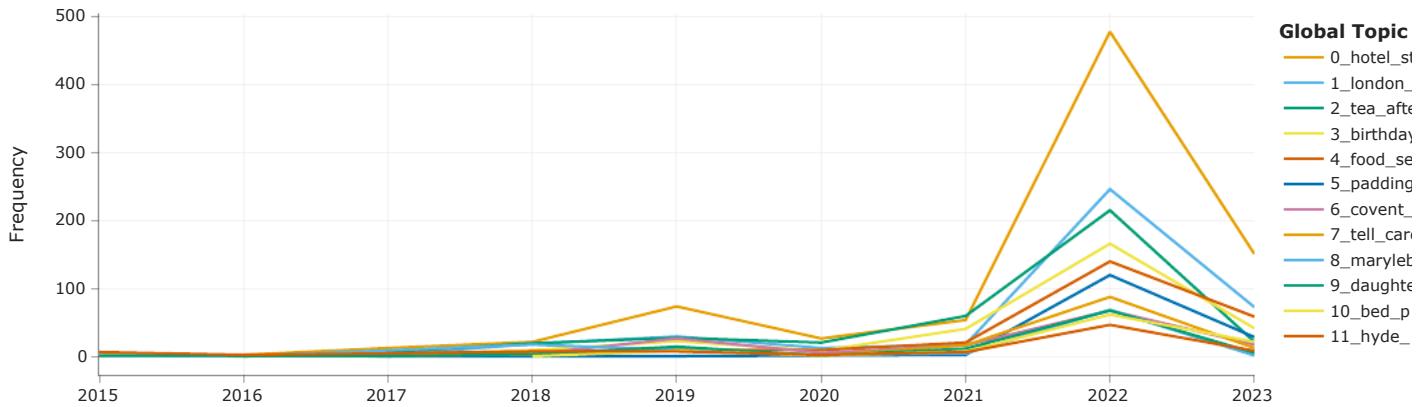
```

In [133]:

```
1 topic_model.visualize_topics_over_time(topics_over_time, topics=[0,1,2,3,4,5,6,7,8,9,10,11])
```

```
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:  
distutils Version classes are deprecated. Use packaging.version instead.  
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:  
distutils Version classes are deprecated. Use packaging.version instead.
```

Topics over Time



Observamos como es la distribución de los tópicos a medida que van pasando los años. Podemos observar como la mayoría sigue la misma representación año tras año p

El elemento más interesante es el salto del tópico número dos. Ha sufrido una gran evolución positiva pasando a ser el segundo tema más hablado en 2022, cuando en 20: tópicos. El tema de color verde en 2021 estaba en segunda posición y en 2022 pasa a estar en tercera posición. El tópico número 5 también ha tenido una evolución positiva

- Ejemplo para visualizar las probabilidades de un tópico

In [134]:

```
1 datos['Review'][0]
```

Out[134]:

```
'It is very near the waterloo train station. Breakfast was good, our room is small but its very clean and the bed is cozy. Free wifi, free  
y unlike the Hampton Inn in the US which has free coffee, hot choco and tea 24/7 at the lobby. There are no fridge and no microwave inside  
y have laundry services. They have a small gym but no pool. The hotel has its our bar restaurant that serves delicious pizzas and chicken  
tel.'
```

In [135]:

```
1 datos['clean_text'][0]
```

Out[135]:

```
'near waterloo train station breakfast good room small clean bed cozy free wifi free coffee tea inside room unlike hampton inn us free cof  
owave inside room coin laundry laundry services small gym pool hotel bar restaurant serve delicious pizza chicken wing overall recommend l
```

In [136]:

```
1 # Get probabilities for all topics
2 topic_model.probabilities_[0]
```

Out[136]:

```
array([5.57452522e-308, 1.27366777e-307, 3.70692288e-308, 5.41064308e-308,
       4.31727220e-308, 1.16340627e-307, 1.14627643e-307, 6.14163475e-308,
       6.94193128e-308, 1.15707922e-307, 6.45568874e-308, 1.15682782e-307,
       1.00155828e-307, 1.00540850e-307, 2.56552798e-307, 2.83166989e-308,
       4.95907892e-308, 7.69096195e-308, 1.36226126e-307, 4.04812506e-308,
       1.98097218e-307, 5.22547037e-308, 9.16642870e-308, 8.29717319e-308,
       7.24481209e-308, 4.06343854e-308, 4.11408025e-308, 4.20312832e-308,
       5.85904887e-308, 1.00000000e+000, 8.25422951e-308, 9.84354822e-308,
       6.03553549e-308, 7.16655191e-308, 7.76052490e-308, 8.20474583e-308,
       7.25086818e-308, 5.37560770e-308])
```

In [137]:

```

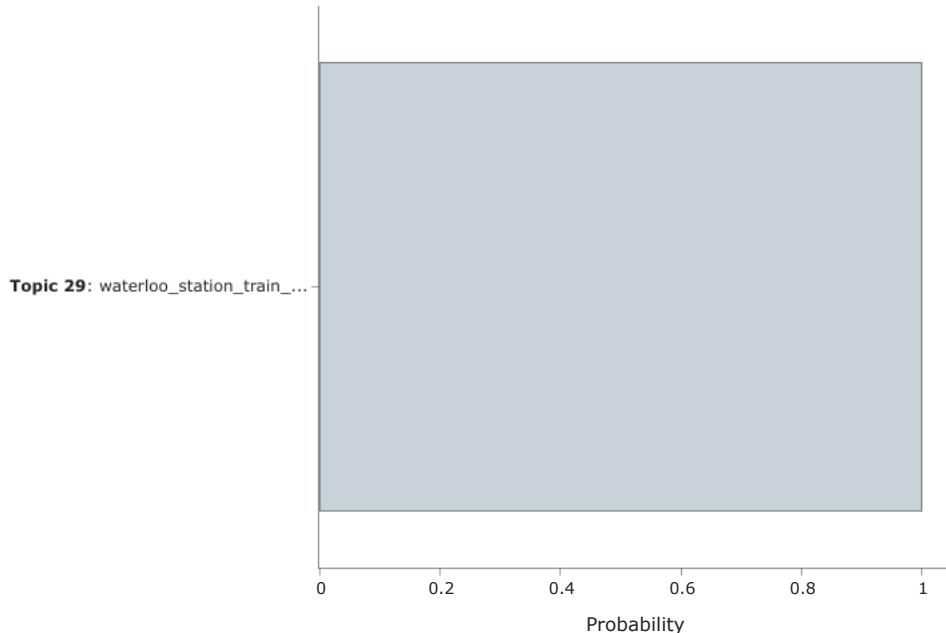
1 # Visualize probability distribution
2 topic_model.visualize_distribution(topic_model.probabilities_[0], min_probability=0.015)

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:
distutils Version classes are deprecated. Use packaging.version instead.

```

Topic Probability Distribution



Aquí observamos las probabilidades que tiene esta review de ser un tópico u otro. Observamos como esta valoración forma parte exclusivamente del tópico número 29.

In [138]:

```
1 datos['Review'][1]
```

Out[138]:

"We booked here as for us it was ideally located for our trip. Hotel is exceptionally clean and staff on reception so nice and friendly. E... If your room isn't ready don't panic they have a locked storage room. We had a room with a view of the city and the shard. Good size and had complimentary breakfast and was more than adequate. Shock horror no fried egg so what it's not the end of the World. Please why do people issues. We've had a difficult couple of years chill out. Will definitely stay here again"

In [139]:

```
1 datos['clean_text'][1]
```

Out[139]:

'book we ideally locate trip hotel exceptionally clean staff reception nice friendly easy check room ready room ready panic lock storage r... really great walk shower complimentary breakfast adequate shock horror fry egg end world please people moan least important issue difficul...y'

In [140]:

```

1 # Get probabilities for all topics
2 topic_model.probabilities_[20]

```

Out[140]:

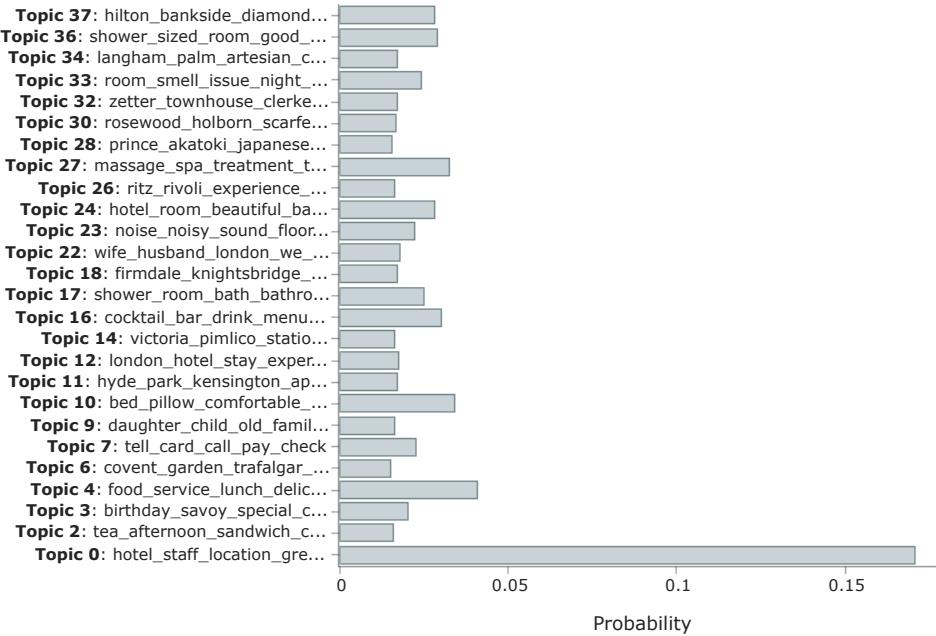
```
array([0.17091546, 0.01384576, 0.0157552 , 0.0201624 , 0.04107084,
       0.01135199, 0.01505544, 0.02251803, 0.01027657, 0.01640926,
       0.03409248, 0.01698734, 0.01761321, 0.01357791, 0.0161863 ,
       0.00764656, 0.0303608 , 0.02506986, 0.01726306, 0.01319633,
       0.01412339, 0.00951423, 0.0178503 , 0.02211001, 0.02813116,
       0.01213036, 0.01642023, 0.03242377, 0.01541058, 0.01396368,
       0.016657 , 0.01450412, 0.01728927, 0.02439428, 0.01695474,
       0.01492218, 0.02885273, 0.02820416])
```

In [141]:

```
1 topic_model.visualize_distribution(topic_model.probabilities_[20], min_probability=0.015)
```

```
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:  
distutils Version classes are deprecated. Use packaging.version instead.  
C:\Users\xikix\anaconda3.2\lib\site-packages\plotly\io\_renderers.py:395: DeprecationWarning:  
distutils Version classes are deprecated. Use packaging.version instead.
```

Topic Probability Distribution



Aquí podemos observar con mucha más claridad la visualización deseada. Vemos que según esta valoración tiene posibilidades de ser 26 tópicos distintos. Claramente veímos susceptible a clasificar en el tópico 0 ya que es el elemento que tiene más probabilidades de ser.

- Voy a crear una variable nueva para observar que número de tópico tiene

In [142]:

```
1 # Get the topic predictions
2 topic_prediction = topic_model.topics_[:]
3 # Save the predictions in the dataframe
4 datos['topic_prediction'] = topic_prediction
5 # Take a look at the data
6 datos.head(4)
```

Out[142]:

	Hotel	Nombre	Titulo	Review	Año	Mes	normaliza	clean_text	sent_subjetividad	sentimiento	tipo_sentimiento	token_len_limpio	bigrar
0	Hampton by Hilton London Waterloo	cresil	Good location	It is very near the waterloo train station. Br...	2023	1	[near, waterloo, train, station, breakfast, go...]	near waterloo train station breakfast good roo...	70	19	neutro	50	near_waterloo_train_stati...
1	Hampton by Hilton London Waterloo	cactusstud	City break	We booked here as for us it was ideally locate...	2021	8	[book, we, ideally, locate, trip, hotel, excep...]	book we ideally locate trip hotel exceptionall...	67	32	positivo	53	book_we_ideal...
2	Hampton by Hilton London Waterloo	Fredbear_Hull	great location	great location, great breakfast. clean. good s...	2023	1	[great, location, great, breakfast, clean, goo...]	great location great breakfast clean good staf...	62	42	positivo	25	great_location_great_breakfa...
3	Hampton by Hilton London Waterloo	Xtremepaul	Really great location and very clean throughout	Now when I'm staying in London I always choose...	2023	1	[stay, london, always, choose, great, location...]	stay london always choose great location min w...	61	55	positivo	23	stay_lond...

