

MÁSTER EN BIG DATA Y DATA SCIENCE

FACULTAD DE COMERCIO Y TURISMO

UNIVERSIDAD COMPLUTENSE DE MADRID



**TRABAJO FIN DE MÁSTER
CURSO 2022-2023**

**Análisis de valoraciones mediante la extracción
del sentimiento y del topic modeling**

GERARD CHICOT NAVALLS

TUTORES: Santiago Mota Herce

Carlos Ortega

Agradecimientos

Este punto es muy significativo ya que sin ninguna de las personas comentadas en este apartado no podría haber llegado a realizar este máster y, consecuentemente, no podría haber disfrutado de un año formándome en Big Data y Data Science concluyendo con este trabajo.

Tengo que dar las gracias a toda mi familia ya que siempre me han apoyando y motivado en mis ilusiones. Siempre me han dicho que apostara por mis estudios, así lo he hecho, hasta finalizar con este máster. Realmente no sé qué me depara mi futuro, pero lo que sí que sé es que voy a seguir formándome en esta materia para llegar a ser un gran profesional. Resaltar también los ánimos y compañía de mi pareja y amigos con los cuales no sería la persona que soy a día de hoy.

Por supuesto no me quiero olvidar de agradecer a todos los profesores que me han aportado mucho en mi desarrollo en el campo del Data Science.

Resumen

Este trabajo final de máster que voy a presentar trata sobre la realización de un análisis sobre cómo son las valoraciones de 86 hoteles situados en Londres. Los datos del estudio son obtenidos mediante un *Web Scrapping* de TripAdvisor. He buscado hoteles de Londres con cuatro estrellas, de esta forma he obtenido negocios con el mismo público, objetivo y características. Seguidamente he realizado un análisis de los datos obtenidos para desarrollar un preprocesamiento y una transformación para la normalización del corpus. Una de las partes fundamentales del trabajo es el estudio del sentimiento. He extraído el sentimiento de cada valoración para poder hacer un análisis segmentado y más detallado. Por último, pero no menos importante, es que para el foco del trabajo he realizado un *Topic modeling* para extraer los tópicos de los que se hablan en el corpus y, así, poder saber cuáles son los puntos clave de las valoraciones del público objetivo.

Abstract

The master thesis I am going to present is about conducting an analysis on the ratings of 86 hotels located in London. The data from the study is obtained through a Web Scrapping of TripAdvisor. I have looked for four-star London hotels, this way I have obtained businesses with the same target audience and characteristics. Then I have made an analysis of the data obtained to develop a pre-processing and a transformation for the normalization of the corpus. One of the fundamental parts of the work is the study of the feelings. I have extracted the feelings of each review to be able to make a segmented and more detailed analysis. Last but not least, is that for the focus of the study I have conducted a Topic modeling to extract the topics that are spoken in the corpus and, thus, be able to know what are the key points of the assessments of the target audience.

Índice

1.	Introducción	1
1.1	Motivación	1
1.2	Objetivos	2
2.	Ingesta de datos	3
2.1	Web Scrapping	3
3.	Análisis exploratorio.....	4
3.1	EDA y Transformación de datos	4
3.2	Normalización	5
4.	Extracción del sentimiento.....	6
5.	Análisis del corpus.....	8
5.1	Análisis cuantitativo	8
5.1.1	Data set general	8
5.1.2	Data set positivo.....	10
5.1.3	Data set negativo	10
5.2	Análisis cualitativo.....	11
5.2.1	Data set general	11
5.2.2	Valoraciones según el año.....	11
6.	Desarrollo de los modelos de Topic Model.....	13
6.1	LDA	13
6.2	BerTopic	15
7.	Conclusiones.....	19
7.1	Resultados	19
7.2	¿Qué puede mejorar?	20
7.3	Futuro trabajo	20
8.	Anexos	21
8.1	Anexo I – Gráficos y tablas	21
8.2	Anexo II – Web scrapping script.....	27
8.3	Anexo III – Script del análisis	39
9.	Bibliografía	84

1. Introducción

Este proyecto se basa en el desarrollo y aplicación de tecnologías Data Science en el área del Natural Language Processing (NLP). La principal intención es extraer información para hacer un análisis cuantitativo y cualitativo de 86 hoteles situados en Londres. Para poder realizar este estudio, he realizado un análisis de sentimiento y un *Topic modeling* para poder extraer los tópicos y el sentimiento de cada *review*.

En primer lugar, he hecho un *Web Scrapping* para conseguir los datos pertinentes al estudio. He creado un data set con distintos elementos de cada comentario de TripAdvisor que ha hecho un usuario. A cada paso hecho, he tratado los datos conseguidos para poder obtener con más facilidad los elementos posteriores. Gracias a todo esto he logrado adquirir de forma automática todos los nombres de los hoteles en Londres que he decidido analizar. Seguidamente, de cada hotel, he extraído el título del comentario, el texto del comentario, la fecha en la que se realizó y el nombre del usuario.

En segundo lugar, he comenzado a tratar los datos para entender qué y cómo tengo que preprocesar. En el apartado del preprocesado he limpiado todo el corpus para tener los datos optimizados. Para conseguir esto, he tratado temas relacionados con la eliminación de espacios en blanco, he convertido todo a minúsculas, he quitado las contracciones, los signos de puntuación y las *stop words*, he tokenizado y he lematizado. No obstante, más adelante entrará en más detalle sobre cómo he obtenido los datos optimizados.

Una vez tengo el corpus tratado he procedido a realizar el análisis de sentimiento y el *Topic modeling*. He extraído el sentimiento de cada *review* para poder clasificar cada texto, de esta forma puedo concretar mucho más mi estudio y realizar grupos de análisis. Todo esto es necesario para llegar a la parte del estudio cuantitativo y cualitativo donde se entiende cuáles son las palabras más utilizadas para los comentarios más positivos y negativos, ver la evolución en el tiempo de las *Top words*, qué palabras valoraban hace un año y ahora, etc.

Con el *Topic modeling* tendría que llegar a las mismas conclusiones que en el apartado anterior pero aquí vemos cuáles son los tópicos más recurrentes, y anteriormente hemos observado las palabras más utilizadas. De esta forma he podido entender cuáles son los temas que más hablan los clientes, las agrupaciones de tópicos, el porcentaje de muestra de cada palabra según el clúster en el que se encuentra, entre otros.

Finalmente podemos encontrar las conclusiones al estudio realizado y una representación de todas las visualizaciones mediante un *dashboard* interactivo.

1.1 Motivación

El objetivo final de este estudio es aportar información real y objetiva de los usuarios que han utilizado los servicios de hostelería de Londres. En los últimos años se ha hablado mucho del marketing, de cómo llegar mejor a tu cliente y de cómo conocer mejor a tu target. Mi intención es ver si es posible, mediante un análisis NLP, lo que los clientes de dichos hoteles valoran más y valoran menos.

Todas estas preguntas y conclusiones anteriores vienen a causa de mi formación previa, ya que soy graduado en Marketing y Comunicación Online dónde el foco principal ha sido siempre el

cliente y todo tiene que girar en torno a él. Gracias a este trabajo vamos a entender cómo se comportan los usuarios que consumen este producto y cuáles tendrían que ser los elementos de inversión para poder conseguir más ventas, mejores valoraciones y, consecuentemente, mejor posicionamiento en el mercado.

1.2 Objetivos

Voy a exponer una serie de objetivos que quiero satisfacer con cada punto de este trabajo:

- Objetivo 1: *Scraping Web* y la extracción de datos para el estudio.

Crear un script donde de forma automática me extraiga todos los nombres de las páginas que yo crea necesario para después, extraer, de cada uno, toda la información necesaria.

- Objetivo 2: Estudio y preprocesado de la Ingesta de Datos.

Observar cual es la forma en la que tengo que tratar los datos para poder dejar y/o extraer los elementos que dificultan el estudio.

- Objetivo 3: Extraer el sentimiento de cada comentario.

Calcular el sentimiento en forma de valor extraído de cada comentario gracias a la suma y la resta del valor de cada palabra que tiene, y así poder separar las *reviews* según si son valoraciones muy positivas, positivas, neutras, negativas o muy negativas.

- Objetivo 4: Realizar un estudio cuantitativo y cualitativo de los distintos grupos según el sentimiento.

Hacer un estudio de los grupos realizados según el sentimiento que tienen para poder entender cuáles son las palabras más utilizadas según la valoración, cómo han evolucionado en el tiempo, cuál es la *review* más positiva y qué es lo más valorado.

- Objetivo 5: Conseguir los temas más hablados por los consumidores.

Realizar un *Topic modeling* para extraer los temas más hablados en cada comentario y así poder entender que es de lo que más se habla, hacer *clusters* de temas, entender mucho mejor a tu target...

- Objetivo 6: Obtener conclusiones a partir de los objetivos 4 y 5.

Las conclusiones serán la forma en que voy a plasmar la información sacada del análisis cuantitativo y cualitativo de los comentarios según el sentimiento y su relación con los tópicos más hablados. De esta forma vamos a ver que es el que más importancia le da el consumidor a día de hoy para que las empresas del sector que no tienen los resultados esperados puedan entender dónde van a tener que invertir para mejorar.

2. Ingesta de datos

En este apartado voy a tratar todo lo relacionado con la ingesta de los datos y cómo he conseguido la información del estudio. Voy a tratar elementos relacionados con el *Web Scrapping* para ver qué es, aspectos legales y cómo he hecho el código para obtener los datos de TripAdvisor.

2.1 Web Scrapping

Actualmente internet contiene una ingente cantidad de datos y cada día es más grande. Esta información es accesible y en muchos casos gratuita, por este motivo he decidido nutrirme de esta tecnología para poder obtener mis datos. Por otra parte, el contenido de los usuarios que consumen tu producto es algo muy valioso, ya que es el *feedback* de cómo estás haciendo tu trabajo, cómo de bueno es tu producto y cómo se siente el consumidor. Su análisis es primordial para mejorar.

El uso más frecuente del *web scrapping* es para generar inteligencia de negocio y así sacar conclusiones objetivas mediante el análisis de las *reviews* de los productos de la competencia, conocer las tendencias del momento o crear comparaciones de precios.

El *web scrapping* en España es legal si los datos extraídos son de acceso público, pero deja de serlo si estás incurriendo en un delito des del punto de vista de la propiedad intelectual o si accedemos a datos de terceros sobre los que no se tiene consentimiento para su almacenamiento o tratamiento (Ley Orgánica de Protección de Datos). Tengo que recalcar que los datos almacenados no se van a divulgar y tienen un uso exclusivamente académico.

El *web scrapping* te permite descargar datos procedentes de internet de forma automática. De esta forma he automatizado un script gracias a la librería BeautifulSoup que ayuda a extraer los datos de internet. He realizado un bucle para entrar en distintas páginas de la web en concreto para poder extraer los nombres de los hoteles de Londres que quiero estudiar. Seguidamente he aprovechado estos nombres para realizar otro bucle y así poder obtener de cada nombre de hotel los nombres de los usuarios, los títulos de los comentarios, los comentarios y la fecha en que se publicó.

La forma en la que obtengo los nombres de los hoteles está compuesta por dos partes. He necesitado dos URL ya que la distinción entre una y otra era tan grande que no he podido realizar un solo bucle. Los dos scripts son iguales lo único que en el segundo hay un bucle que recorre 4 páginas en busca de nombres, y el primero sólo recorre la primera página.

Una vez he conseguido los nombres de los hoteles de Londres, los he tenido que tratar para dejar el nombre tal y como indican las URL del sitio web. Cuando he extraído la información me salía “1. El Nombre del Hotel”, pero yo necesitaba “El_Nombre_del_Hotel” ya que la URL en cuestión utiliza este formato para identificar cada hotel.

Al tener todo de una forma concreta, pude proceder a realizar la segunda parte en la que obtengo la información deseada de cada hotel. He realizado un bucle que recorre todas las páginas necesarias y extrae la información relacionada con cada comentario.

3. Análisis exploratorio

Una de las partes más importantes del trabajo es el preprocesado del corpus, ya que sin un buen preprocesado el valor, la calidad y la robustez del estudio disminuye. Para poder realizar bien este apartado he tenido que observar en detalle como es el texto para determinar qué tengo que hacer.

En general este apartado trata de dejar el data set sin elementos extraños para luego realizar la normalización del corpus. Se trata de dejar el texto uniformado para así poder generar los vectores que nos van a servir para modelar.

3.1 EDA y Transformación de datos

En el análisis del corpus he observado ciertos elementos que se van a tener que tratar. Por ejemplo, he encontrada NaN, duplicados, comentarios en distintas lenguas, variables a tratar...

- Si hablamos de los duplicados podemos ver como existen dos filas idénticas. Esto no me parecía normal, entonces decidí prescindir de una de ellas.
- En los valores perdidos he encontrado dos filas que no tienen *review*. No tiene ningún sentido que haya dos comentarios iguales, con lo cual eliminé estas filas.
- He observado que hay clientes recurrentes, es decir, hay usuarios que han hecho más de un comentario en un mismo hotel, pero también hay otros que han valorado hoteles distintos. En el estudio hay usuarios que consumen un solo hotel, clientes que consumen hoteles distintos y otros que solo han estado en un hotel del estudio.
- En los comentarios hay varias valoraciones con distintas lenguas, en concreto hay 24 comentarios que no son en inglés. Por lo tanto, tuve que decidir si hacer una traducción o eliminarlas directamente.
- También he decidido tratar dos variables en concreto: el nombre del hotel y la variable data.
 - El nombre del hotel lo he decidido tratar para que se entienda mejor. He pasado de separaciones con “_” a separación con espacio y también he eliminado la parte final de cada nombre, ya que había “-London_England”. De esta manera se entiende mucho mejor.
 - La variable data también está formada por elementos prescindibles y, aparte, se tiene que transformar para poder utilizar la información de la columna. Entonces, he eliminado “Date of stay:” para que me quede con “el mes en concreto” + “el año”. “El año” está en número, pero “el mes en concreto” está escrito y es lo que he transformado a numérico.

Una vez realizada toda la fase exploratoria, tengo claro qué hacer y cómo voy a proceder a la transformación. He decidido eliminar los elementos duplicados, los NaN y los comentarios que están en otras lenguas que no son la del inglés. Al ser pocos elementos no va a afectar al modelo o al análisis del sentimiento ya que tengo 12794 filas y eliminé 27 que es igual al 0.021% de los datos. Por otra parte, en la variable año, he convertido a numérico el nombre del mes y he separado en dos columnas distintas el año y el mes.

Ahora mismo está todo tratado para poder proceder con la normalización del corpus.

3.2 Normalización

En el *Text Mining* es necesario transformar el corpus con características numéricas para que los algoritmos puedan hacer su trabajo. La normalización se divide en dos grandes partes: la primera sirve para identificar qué tengo que eliminar o cambiar para normalizar el corpus y la segunda se trata de transformar dichos elementos para tener el corpus listo para vectorizar.

La intención de la normalización es uniformizar el texto para reducir su dimensionalidad y, así, puede favorecer a la velocidad de construcción de modelos y su eficiencia.

En el análisis de las valoraciones me he fijado en que hay muchos elementos a tratar como podrían ser los signos de puntuación, hay texto en mayúsculas y en minúsculas, las palabras vacías de sentimiento, las contracciones y todo aquello que dificulte el análisis o esté presente en algunas valoraciones y en otras no. Por supuesto, también voy a tokenizar y lematizar el corpus procesado. Un token es una palabra, un signo de puntuación, un número... Cuando tokenizamos el texto procesado nos quedamos solo con esas palabras que nos interesan, que aportan valor al estudio. En la lematización la intención es quedarse con la raíz de la palabra para quedarnos con el significado principal.

Por todo lo explicado anteriormente, he transformado todas las palabras a minúsculas, he quitado las contracciones, he realizado la tokenización, he eliminado las palabras vacías (*stopwords*), he eliminado todos los signos de puntuación y he realizado la lematización.

Una vez he realizado, mediante las funciones pertinentes, todos los cambios en el corpus me quedaron de esta forma:

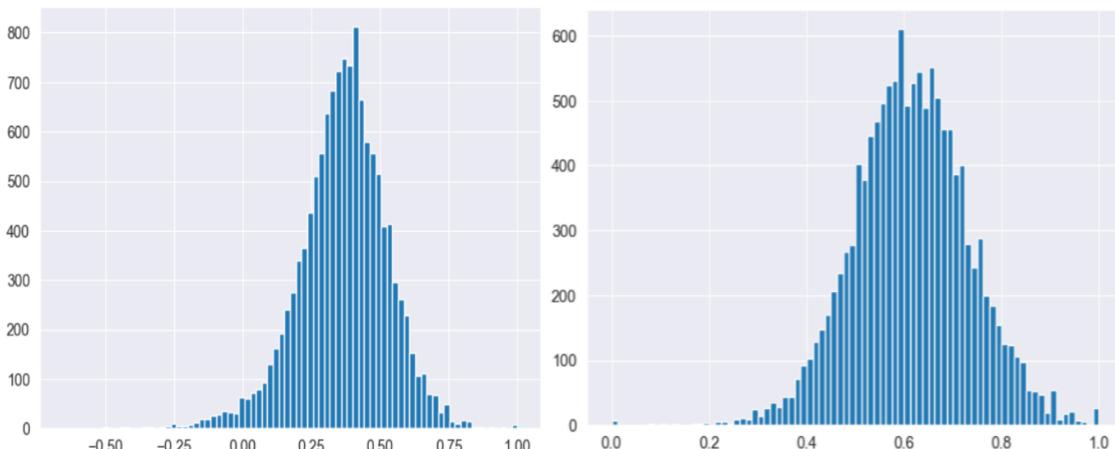
Review	normaliza	clean_text
I stayed at the Resident Hotel Covent Garden f...	[stay, resident, hotel, covent, garden, 6, nig...	stay resident hotel covent garden 6 night coul...
This was our first trip to London, so we wante...	[first, trip, london, want, pick, hotel, would...	first trip london want pick hotel would spend ...
My wife and I stayed 4 nights at the Resident ...	[wife, stay, 4, night, resident, covent, garde...	wife stay 4 night resident covent garden last ...
The hotel reps were Amazing! helpful and patie...	[hotel, rep, amazing, helpful, patient, anytim...	hotel rep amazing helpful patient anytime day ...
Came to see a show as a birthday treat. Great ...	[come, see, show, birthday, treat, great, loca...	come see show birthday treat great location 5 ...

4. Extracción del sentimiento

El análisis del sentimiento sirve para detectar y analizar las opiniones que la gente expresa. Para realizar este trabajo me interesa extraer el sentimiento a nivel de frase, que mediante las partículas de sentimiento puedo clasificar en positivas, negativas o neutras. Las partículas de sentimiento son aquellas palabras que ayudan a determinar si es positivo o negativo. En mi análisis he utilizado la librería TextBlob para que, mediante operaciones aritméticas, me salga un valor para determinar si una valoración es positiva, negativa o neutra.

Gracias a TextBlob he podido conseguir dos valores a partir de un comentario:

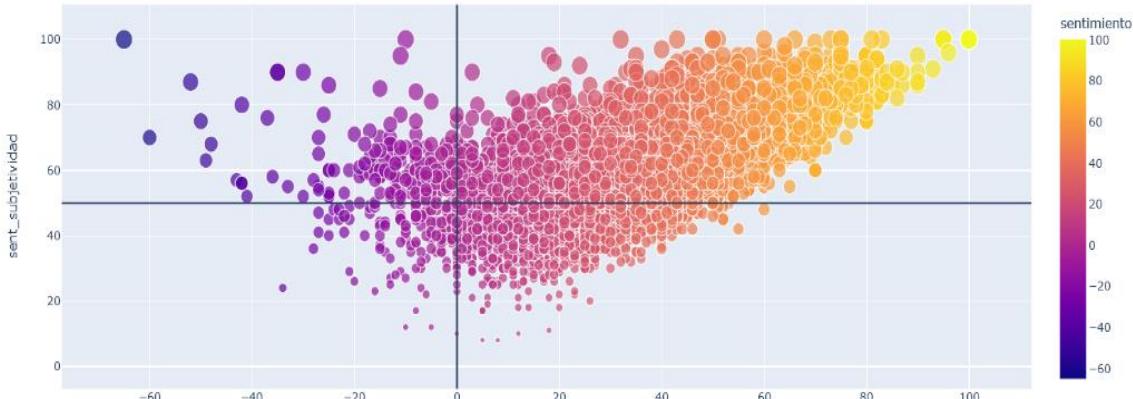
- `TextBlob.sentiment.polarity`: esto nos devuelve un valor entre el -1 y el 1. El -1 es el valor más negativo y el 1 es el valor más positivo. Para el estudio, me interesaba dividir en tres el sentimiento: positivo, negativo y neutro. Los valores positivos tienen un rango de 1 y 0.21, los valores neutros van del 0.20 al -0.20 y los valores negativos van del -0.21 al -1.
- `TextBlob.sentiment.subjective`: esto nos devuelve un valor entre 0 y 1. Este valor nos habla de la objetividad y la subjetividad, cuanto más se acerca al 0 más objetivo es y cuanto más se acerca a 1 es más subjetivo.



En la gráfica de la izquierda podemos observar la distribución del sentimiento. Claramente se ve como la mayoría de los comentarios son más positivos que negativos. En la derecha apreciamos distribución de la subjetividad. Vemos como el pico está en 0.6, los comentarios tienden a ser un poco más subjetivos que objetivos.

Con la siguiente gráfica podemos ver la distribución del sentimiento con su propia subjetividad. Podemos ver cómo hay más valores neutros y positivos que negativos, hay muy poca representación de elementos negativos. También vemos como las valoraciones más positivas y negativas tienden a ser subjetivas. Podemos concluir que nuestro corpus tiene:

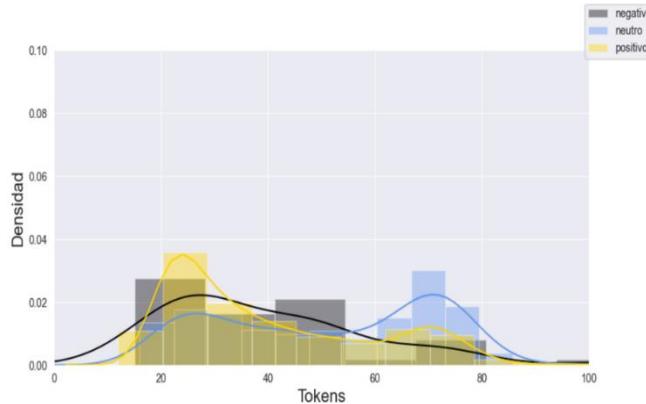
Sentiment Analysis



- 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 *reviews* neutras.

Siguiendo con el análisis del sentimiento he creado una gráfica para entender cuántos valores hay de cada clase de sentimiento y que distribución de palabras tienen las valoraciones según la clase pertinente.

Claramente los hoteles del análisis están haciendo bien su trabajo, ya que las observaciones del tipo positivo despuntan mucho más que las otras clases. Podemos ver con claridad como las valoraciones negativas tienen muy poca representación.



En la parte derecha de la gráfica 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 20-30 tokens tienen la review hecha. 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 los comentarios negativos, aunque haya muy poca representación, podríamos decir que las reviews tienen entre 60 y 80 palabras, es decir, cuando no les ha gustado han explicado detalladamente el porqué.

5. Análisis del corpus

He dividido este punto en cuatro estudios independientes para observar cómo se comportan las palabras en cada estado del sentimiento. Para cada uno he observado lo mismo. Mi intención es hacer un análisis cuantitativo y cualitativo para observar cuales son las palabras que salen con más frecuencia, cuantas veces salen dichas palabras, ver cuáles son los comentarios más positivos y negativos y observar cómo son los bigramas y los trigramas para entender mucho más el corpus.

El estudio cuantitativo va relacionado con el tipo de sentimiento. He hecho un estudio general para entender qué valora el target inglés, pero lo interesante de este análisis es ver donde se reproducen estas palabras y entender si tienen un sentido más positivo o negativo.

En el estudio cualitativo muestro cuales son los mejores y peores comentarios para observar cómo y de qué forma son las quejas de las *reviews*.

5.1 Análisis cuantitativo

5.1.1 Data set general

Si analizamos las palabras que salen con más frecuencia en el estudio podemos encontrar *hotel, room, breakfast, great, good, London...* Con ese primer contacto ya podemos intuir de qué van a tratar las valoraciones. También podemos observar la frecuencia de aparición de cada palabra, esto nos da más información, así podemos saber cuántas veces ha salido cada *token*.

En el gráfico 1 (vea Anexo I) encontramos como cuatro palabras tienen una aparición mucho más elevada que las otras. *Hotel, room, stay* y *staff* tienen una frecuencia superior a las 8000 apariciones. Entre 6000 y 4000 encontramos palabras con mucha importancia como *staff, great, London, good, service, location* y *breakfast*. Son palabras que tienen mucho significado y que pueden llegar a ser temas de los que los usuarios hablan detalladamente. Todo lo que procede, en gran medida, son palabras que podrían acompañar los *tokens* comentados anteriormente. Podemos observar *lovely, helpful, comfortable, excellent...* Palabras con un carácter positivo, hecho que nos confirma que tienen más aparición las palabras positivas que las negativas. En frecuencias de aparición bajas, también hay palabras que pueden ser interesantes como *clean, restaurant* o *experience*.

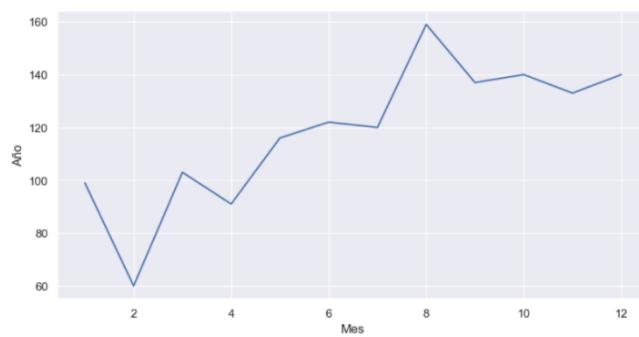
En los bigramas y trigramas ya se puede entender mucho mejor las asociaciones de palabras. Son elementos muy explicativos y podemos ver hacia donde se encamina el sentimiento de los *tokens* mencionados anteriormente.



En los bigramas podemos observar elementos muy positivos como *highly recommend*, *staff friendly*, *great location* o *friendly helpful*. Aquí ya podemos ver cómo se asocian las palabras con más y menos frecuencia logrando *tokens* muy explicativos y positivos. Solo hay un *token* que me llama mucho la atención: *can not*. Esta asociación tiene un sentido negativo, pero en los trigramas observamos como en realidad es *can not wait* que tiene un sentido positivo. En los trigramas se puede apreciar como realmente la gente está muy contenta con el servicio y el producto. Hay palabras como *would highly recommend*, *staff friendly helpful* o *hotel great location*.

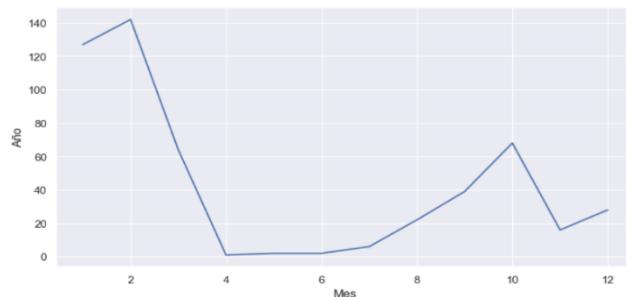
El análisis cuantitativo del *data set* general nos da mucha información valiosa, pero me quedo con las palabras más usadas, su frecuencia de aparición y saber con qué sentimiento se relaciona. Claramente las valoraciones tienen un sentido positivo, consecuencia la poca representación de comentarios negativos. Esto me hace pensar que los hoteles del estudio saben muy bien lo que necesita su público objetivo.

Al centrarme en los comentarios a lo largo de los años he encontrado datos muy interesantes relacionado con la Covid-19 y la reducción de la demanda por causa del confinamiento y las restricciones mundiales.



La gráfica superior corresponde al año 2019, donde podemos ver cómo tiene una tendencia ascendiente des del mes de febrero, pero con pequeñas bajas en abril y setiembre. El mes que tiene más comentarios es el agosto, dato lógico ya que verano y navidades son los meses con más ocupación y, por lo tanto, es normal que tengan más comentarios. Podemos apreciar cómo después de verano la gráfica se mantiene hasta diciembre, donde hay navidades y 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 gráficas.

En la siguiente 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 enero todo era normal y, a partir de aquí, hay una bajada a 0 comentarios de una forma directa. Hasta en julio no vemos ningún comentario y en octubre hay un pequeño pico que



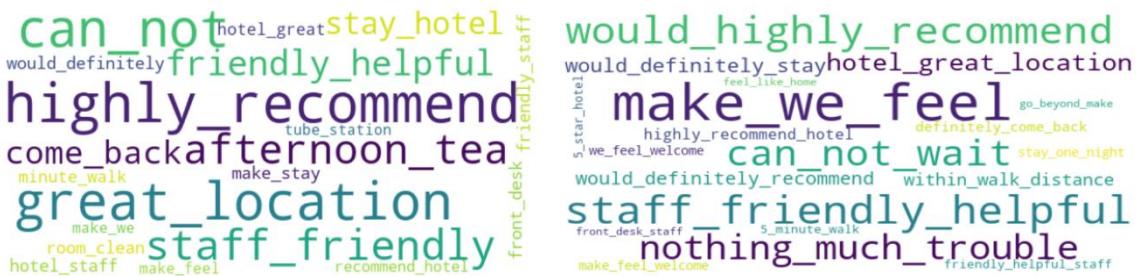
nada se puede comparar al año anterior.

Generalmente hay una correlación entre los meses de mayor ocupación y el número de comentarios. Es verdad que, en la gráfica del 2022, año en el que hay más valoraciones, el pico está en octubre y la gráfica tiene una tendencia positiva des de enero. Hecho que no confirma la teoría anterior, pero lógicamente cuantas más personas consumen un producto más probabilidad hay que tengan valoraciones.

5.1.2 Data set positivo

Lógicamente el *data set* positivo tiene una gran similitud con el análisis cuantitativo del *data set* general, ya que la mayoría de valoraciones son positivas. Los *tokens* que aparecen son muy similares y en la gráfica 2 (vea Anexo I) observamos como la frecuencia de aparición es muy similar.

Si entramos en más profundidad observamos que en los bigramas y trigramas hay las mismas agrupaciones de palabras.



Esto me permite concluir en que todo el *data set* va relacionado con *reviews* positivas. Este hecho estaba claro en el análisis inicial al ver las representaciones de las clases, pero haciendo un análisis más detallado y profundo vemos como se representan los mismos elementos.

5.1.3 Data set negativo

Lo interesante de realizar un análisis cuantitativo del *data set* negativo es que por mucho que haya poca representación no va a dar información muy útil para poder entender el público objetivo y en donde se centran las valoraciones negativas.

En el gráfico 3 (vea Anexo I) vemos la distribución de la frecuencia de los *tokens*. Encontramos las mismas palabras que en los estudios anteriores, pero con características un poco distintas. Observamos como las palabras con más representación son *hotel* y *room* con una diferencia importante. A diferencia de los otros encontramos palabras como *check*, *work*, *manager*, *dirty* o *bad*. Esto me hace pensar que hay personas que han comentado temas relacionados con el pago, la limpieza y cómo ha sido su mala experiencia.

Si entramos en más profundidad encontramos en los bigramas y trigramas asociaciones de *tokens* similares a los anteriores y otros de nuevos. Observamos como hay *star hotel, room ready, bad experience, five stars...* Me hace pensar que las valoraciones han sido en relación a la mala experiencia estando en hoteles de alta categoría. En los trigramas hay elementos como *room blood find, cleanliness room blood, room city view, five star hotel, ask hotel manager...* Todo va relacionado sobre la limpieza y como las habitaciones no cumplen con los requisitos.



5.2 Análisis cualitativo

5.2.1 Data set general

Las valoraciones más positivas son aquellas que tienen la extracción del sentimiento más alta y, lógicamente, la más negativa es aquella que se acerca más al menos uno.

La valoración más positiva:

Comfortable, superbly located, well managed and with a delicious breakfast. What else could one wish for? It was better than expected. Thanks for all the staff. we would stay there again without any doubts.

La valoración más negativa:

Awful customer service at reception. Reception manager was extremely judgmental and unhelpful, with a terrible attitude. Felt very sorry for her staff. We will never be booking again. Avoid at all costs.

5.2.2 Valoraciones según el año

Vamos a ver cuáles han sido las mejores y peores valoraciones en el rango de años comprendidos entre el 2023 y el 2021.

Las valoraciones más positivas:

2021: Sloane Sq Hotel, is tucked away very neatly on Sloane Sq. My stay here was delightful. Great staff and hotel. The rooms have a lot of character and make for a perfect night's 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 choosing where to eat too!

2022: We like to stay Biltmore every time we visit London which is very often and we are always treated wonderfully. Rooms are renovated and

spacious. For breakfast or lunch Asif Bajwa is always there for to take care your comfort.

2023: 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 heart of London in a wonderful apartment. Recommend the wonderful breakfast delivered to your room

Las valoraciones más negativas:

2021: Disgusting! I got shocked and surprised when walking through the corridor and saw that the hotel uses the same lift to transport foods and dirty clothes, a lady member of staff 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 safety policy!

2022: Awful customer service at reception. Reception manager was extremely judgmental and unhelpful, with a terrible attitude. Felt very sorry for her staff. We will never be booking again. Avoid at all costs.

2023: 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, noisy heating system in the room. Also, the rooms are not renovated. Not a value for money hotel! Does not worth the cost of ~ 450& a night!

6. Desarrollo de los modelos de Topic Model

Para obtener los resultados deseados en la creación de temas de los comentarios de los hoteles de Londres me he ayudado del *topic modeling*.

Esta técnica de *machine learning* no supervisado tiene como función detectar patrones de *tokens* y frases para agrupar automáticamente grupos de palabras y expresiones similares. Lo que hace es contar palabras y agrupar patrones de palabras similares para encontrar temas en datos no estructurados. Detecta patrones como la frecuencia de aparición de los *tokens* y la distancia entre las palabras para asociar similitudes.

He utilizado dos modelos para lograr sacar los temas de cada comentario: un LDA y un BerTopic. El LDA es un modelo más clásico con una velocidad de procesamiento mucho más rápido que el BerTopic pero con una menor profundización. En cambio, el BerTopic saca resultados más específicos y profundos de los temas del corpus. He querido ver la diferencia entre los resultados de los dos modelos y ver si las conclusiones son similares.

6.1 LDA

El LDA o Latent Dirichlet Allocation es un modelo generativo¹ que nos ayuda en la explicación de grupos observados a partir de conjuntos no observados. De esta forma podemos explicar qué datos son similares. Fue creado por David Blei, Andrew Ng y Michael Jordan en 2002 y lo presentaron como un ejemplo de modelo grafo².

Cada documento se puede ver como una combinación de categorías. El elemento clave del LDA es que supone que el uso de una palabra es ser parte de un tema y que da siempre la misma información se encuentre donde se encuentre. El LDA tiene un enfoque más probabilístico y me da un vector de probabilidades de cada *review*.

Para poder llevar a cabo este modelo he realizado distintos objetos para poder ejecutar y entrenar un modelo LDA.

Una vez tienes todo el corpus normalizado y limpio de elementos que no aportan nada, tienes que tener un objeto lematizado para poder crear un vector. Este vector nos va a servir para entrenar el modelo. Seguidamente he creado el modelo en cuestión con unos parámetros iniciales. Para terminar el entrenamiento del LDA he realizado un Grid Search para optimizar el modelo y ver los mejores parámetros con los que tengo que entrenar el modelo.

Mis resultados me han creado cuatro tópicos, cada uno más grande que el otro, donde mediante la asociación de *tokens*, hay la explicación de cuatro temas con los que clasifica cada comentario del *data set*.

Observamos en las tablas 1 y 2 (vea el Anexo I) cuantas valoraciones según el tópico ha creado el LDA y la importancia de cada tópico según el comentario. Podemos comentar como el primer tópico, el número 2, se queda con la máxima distribución. Tiene 5010 veces de

¹ Un modelo generativo es un modelo para generar valores aleatorios de un dato observable. Este especifica una distribución conjunta sobre una secuencia de etiqueta.

² Representa las dependencias entre variables aleatorias como un grafo en el que cada variable aleatoria es un nodo.

aparición, los tópicos 0 y 1 tienen 3844 y 2052 apariciones respectivamente. Por último, tenemos el tópico 3 con 1863. En la otra tabla se puede apreciar el porcentaje de probabilidad de acierto de cada tópico para crear, finalmente, la variable dominant_topic que se queda con el tópico con más probabilidad de acierto.

Las gráficas 4 y 5 (vea Anexo I) son complementarias y muy explicativas. Primero encontramos el Intertopic Distance Map, donde cada burbuja pertenece a un tópico. Seguidamente encontramos los Top-30 Most salient Terms que nos muestra todas las palabras que tiene cada tópico y su frecuencia de aparición en el tópico determinado.

Podemos observar cómo realmente es un buen modelo ya que la burbuja de la gráfica de la izquierda está muy separada. La pretensión es intentar encontrar grupos que no se solapen ya que sería un 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 va relacionado con la aparición del tópico 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* hablados. Encontramos palabras como *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 a la del tópico 1. Contiene 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 considerable de aparición, la palabra *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*, la palabra con más frecuencia es *hotel*, seguido de palabras como *room*, *stay*, *service* y *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 mundo de la hostelería.
- 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 con comentarios positivos.
- En el tópico 3 se habla de las habitaciones del hotel y cómo son. Encontramos *check*, *bed*, *clean* y *bathroom*. Las *reviews* que hablen de cómo son las características de las habitaciones serán del tópico 3.
- 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 gustaría segmentar más los tópicos para poder entender mucho más a mi público objetivo. Por este motivo he decidido realizar otro modelo para profundizar más.

6.2 BerTopic

BerTopic es una técnica de modelado de tópicos que se nutre de BERT y TF-IDF para crear grupos densos de temas fáciles de interpretar manteniendo palabras importantes en las descripciones de los temas. Realmente lo que hace es aprovecharse de Transformers para modelar temas de formas muy distintas, entre ellas la forma manual, supervisada y semisupervisada.

Este modelo es más lento que otros si lo comparo con un LDA o un NMF pero tiene mucha más profundidad. Esto me va a permitir poder encontrar muchos más tópicos y poder entender mejor al público objetivo. Obtiene los resultados a partir de crear *embeddings* del corpus para luego realizar un *clustering* de los tópicos.

Gracias a todo lo comentado anteriormente podemos representar documentos de un corpus con *embeddings* mediante técnicas de *clustering*, para agrupar palabras que semánticamente son similares.

Para crear este modelo he creado una reducción de la dimensionalidad mediante UMAP. El algoritmo funciona mucho mejor ya que en dimensiones bajas da mejores resultados. El Uniform Manifold Approximation & Projection, UMAP, ayuda a mantener la estructura local y global de los datos, aunque se reduzca su dimensionalidad.

Seguidamente he utilizado el algoritmo HDBSCAN. Este modelo de *clustering* está basado en la agrupación según la densidad. Otros elementos que me han hecho decidir por este modelo es que admite ruido en los datos y no necesita que le diga previamente el número de *clústeres* que necesito.

Con lo cual me quedo finalmente con un modelo con el que le he reducido la dimensionalidad con UMAP y he agrupado según la densidad con HDBSCAN. Este es mi modelo final para obtener los temas más hablados en las valoraciones.

1	topic_model.get_topic(0)
	[('hotel', 0.02835508130850788), ('staff', 0.02637198788803157), ('location', 0.026108620470829288), ('great', 0.02543446529187555), ('helpful', 0.021626291811826036), ('stay', 0.0215180116169612), ('friendly', 0.021418190262895074), ('good', 0.02122203330053244), ('nice', 0.020079836024441976), ('recommend', 0.01889440140678703)]

1	topic_model.get_topic(1)
	[('london', 0.05016294162359855), ('good', 0.02977535263314355), ('location', 0.0274399664495423), ('great', 0.02673017652565895), ('stay', 0.024178295660578115), ('hotel', 0.02357664839623486), ('recommend', 0.021163341632456457), ('staff', 0.02018799216626245), ('place', 0.01882549351271179), ('would', 0.016759320785621274)]

Aquí vemos los primeros resultados del BerTopic. A la izquierda encontramos agrupado *hotel*, *staff*, *location*, *great*, *helpfu*, etc. Esto nos determina que el tópico 0, el que tiene más representación, habla de temas relacionados con el hotel y cómo es, es decir, donde está situado, como es el equipo, etc. En tópico de la derecha observamos *tokens* como *London*, *good*, *location*, *great*... Esto hace pensar que este tópico se encontrará en las *reviews* que hablan de forma positiva de temas relacionados directamente con la ubicación.

En la gráfica 6 (vea Anexo I) 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 entrar en más profundidad de análisis 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 hay tópicos que comparten *tokens* en común.

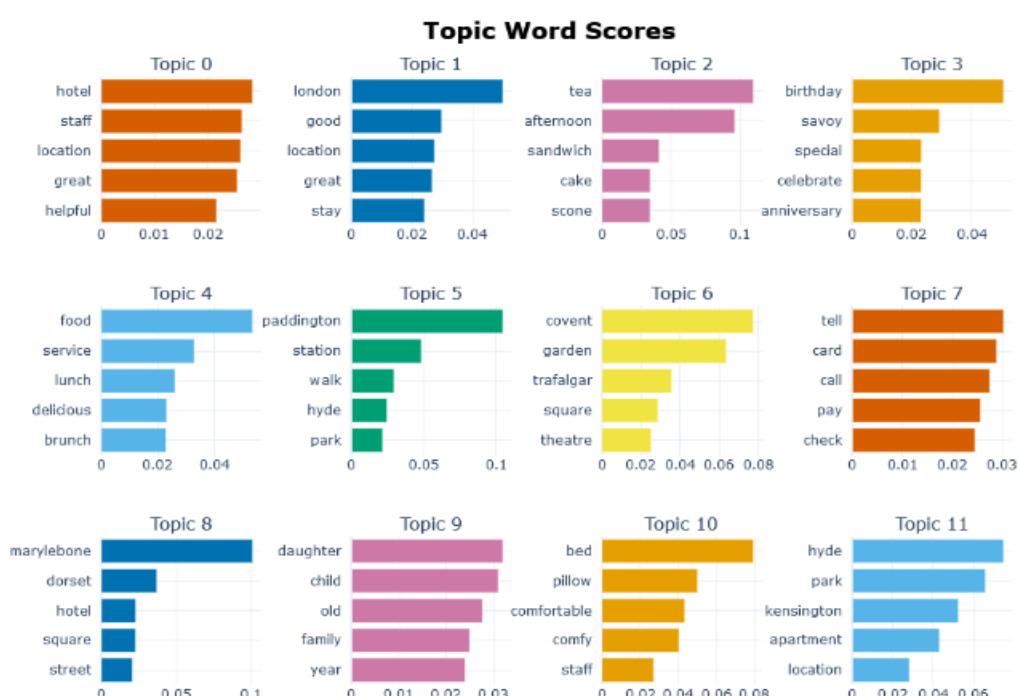
En la parte inferior del gráfico encontramos el tópico 0, el círculo más grande, junto a muchos de pequeños. En el tópico 0 se hablan de *tokens* como *hotel*, *staff*, *location*, *great* o *helpful*, en el tópico 1 se habla de *london*, *good*, *location*, *great* y *stay*. Podemos ver como sí que hablan de más o menos lo mismo, pero de forma distinta, que es realmente lo que quería conseguir, segmentar mucho más los tópicos.

El gráfico 7 (vea Anexo I) puede parecer una visualización 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 cómo es la contribución de cada palabra dentro del tópico. Nos muestra como de importantes son las palabras dentro del tópico, si la primera tiene todo el peso de la muestra, si está distribuida por toda la muestra, etc.

- 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 peso. Otro ejemplo sería el tópico número 61, que empieza en lo más alto. Esto quiere decir que la primera palabra tiene un gran peso. Al caer empicado podemos ver como el peso de la segunda palabra es muy bajo.

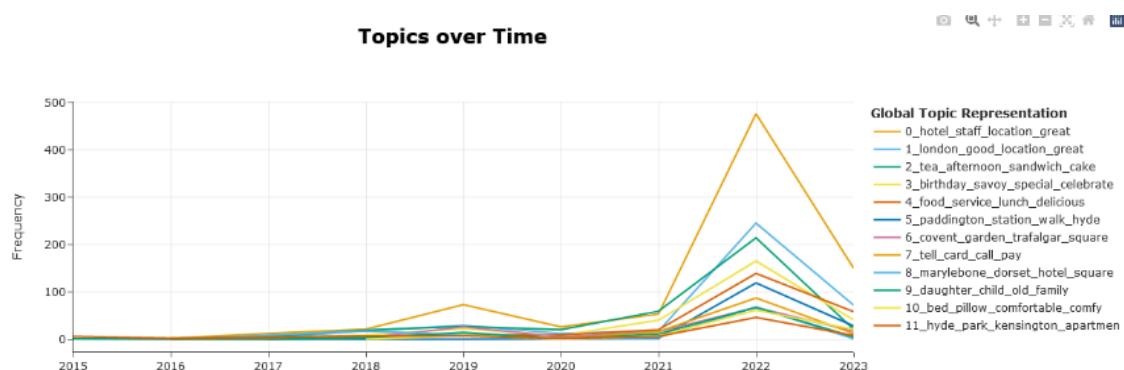


En esta gráfica “Topic Word Scores” podemos observar los 12 mejores tópicos del modelo con sus cinco mejores palabras. Claramente vemos una gran relación de las palabras que forman cada tópico, pero hay elementos interesantes dentro de los histogramas. En algún tópico hay palabras con mucha más representación que otras. En la gráfica posterior analizaré si es algo normal en los tópicos o si son casos concretos. Esta gráfica nos habla muy bien de cómo trabaja el modelo. Podemos observar cómo cada tópico habla de forma similar sobre una temática independiente, como por ejemplo el tópico 4 hablando de la comida o el tópico 9 hablando de la familia.

En la gráfica 8 (vea Anexo I) encontramos las relaciones entre los tópicos, donde cada color es un nivel de relación. Podemos observar cómo las conexiones tienen relación como por ejemplo el verde de arriba contiene elementos relacionados con la familia y celebraciones. Lo más probable es que los tópicos que hablan de temas relacionados con la familia van relacionados con celebraciones o eventos festivos. 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 con el rojo situado debajo. Observamos como se relaciona la familia y los eventos festivos con el hotel, la localización, el *staff*, etc.

En este *corrplot* de la gráfica 9 (vea Anexo I) podemos encontrar la relación de similitud que hay entre cada tópico, donde tanto si están en azul oscuro como en blanco estarán corraladas, una en sentido positivo y otra en sentido negativo. 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 del 0.91, están muy corraladas, puedo concluir que por mucho que hablen de *tokens* un poco distintos hay multicolinealidad. Pasa lo mismo con el tópico 5 y el 11. En la gráfica anterior hay relación entre ellos y en esta gráfica hay una correlación de un 0.89.

En la gráfica 10 (vea Anexo I) 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. Apreciamos como la distribución de la frecuencia está muy relacionada con elementos encontrados 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 tópico y otro que habla sobre *hotel, staff, location...* En el sentido negativo, con muy poca representación, el elemento más aparecido se habla de *tell, manager, terrible, check, email...* Si en el *data set* hubiese más representación negativa podríamos hablar mucho más de los temas que la gente habla sobre este tipo de sentimiento.



Observamos cómo 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, pero hay otros que no.

Un elemento interesante es el tópico número 2. En 2021 estaba en primera posición, pero en un año ha pasado a estar en tercera posición. El tema de color azul es el que ha evolucionado de forma más positiva. En 2021 estaba casi a bajo del todo y en 2022 está en segunda posición. El tópico número 3 ha pasado de ser número 2 en 2021 a ser número 4 en 2022.

7. Conclusiones

7.1 Resultados

Los resultados de los modelos tienen un sentido diferente en cada modelo. En el LDA he obtenido cuatro tópicos con los que clasifica todas las valoraciones y, en cambio, en el BerTopic he conseguido 37 temas. He podido observar como uno profundiza mucho más que el otro con lo cual puedo entender mucho mejor de lo que hablan los consumidores por mucho que haya temas que se podrían relacionar muy directamente con otros. Para este estudio he priorizado los resultados del modelo profundo que no los del otro modelo, ya que mi intención es saber al detalle donde hay las valoraciones más positivas y negativas.

Después de analizar todo el *data set* y entender el volumen de comentarios por usuario, el año de la publicación y el corpus, he obtenido unos resultados ciertamente interesantes. Podemos decir que los tópicos más utilizados por los consumidores, según el modelo BerTopic, son los relacionados con los trabajadores del hotel, la localización del hotel, la comida, eventos festivos, zonas turísticas cercanas, características de las habitaciones y temas relacionados con el pagamento.

Con lo cual, si una persona me pregunta, objetivamente, cuáles son los puntos clave para crear un negocio hostelero en Londres según el público objetivo serían:

- Los trabajadores y los servicios del hotel son el elemento clave. El tópico más importante nos habla concretamente de esto. Si ofrecen un buen servicio con gente amable y simpática la gente lo valora mucho.
- La ubicación hotel. Elemento crucial para cualquier negocio hostelero. Si tu ubicación está en el centro de la ciudad o en un punto clave rodeado de elementos turísticos vas a estar muy buscado. No olvidar que también es muy importante como te vas a mover por la ciudad. Tienes que estar rodeado de sitios que facilitan la comunicación entre lugares.
- Los productos alimentarios. Palabras como *tea*, *sandwich*, *cake*, *scone* y *afternoon* configuran este tópico. No puedes olvidar la tradición por este tipo de productos.
- Tienes que tener una sala o un lugar donde poder celebrar fiestas de aniversario. ¡Hay muchas valoraciones sobre este tema!
- La comodidad de la cama, la almohada y los elementos que configuran la calidad de la habitación tienen que satisfacer al público objetivo. Vas a tener muy malas valoraciones si no inviertes en este tema.

Realmente solo me he quedado en la mitad de lo que quería conseguir. Mi intención era aportar valor en relación a lo más y menos que valora el público objetivo, pero solo he obtenido resultados de lo más positivo. A causa del *data set* concreto, que tiene muy poca representación negativa. Este hecho ha hecho que solo obtenga conclusiones objetivas de las valoraciones positivas. Con más representación de esta clase desbalanceada hubiera podido obtener conclusiones de cuáles son los aspectos que hacen valorar más negativamente los hoteles de Londres.

Concluir también que si más adelante tengo que hacer un análisis de tópicos de unos comentarios de hoteles, restaurantes o ubicaciones, utilizaría el modelo BerTopic

directamente para poder obtener tópicos más segmentados y específicos. Ayuda mucho más a entender el target este modelo que un LDA. Con el LDA he visto de forma rápida cuales son los *tokens* que salen en los temas con más aparición, pero no me ha permitido entrar en profundidad como quería.

7.2 ¿Qué puede mejorar?

Para obtener las conclusiones deseadas claramente se tienen que buscar elementos de estudio que tengan características de todas las clases, no se puede pretender dar información positiva y negativa sin ellas.

Por otra parte, un análisis temporal más exhaustivo podría ser muy interesante. Recopilando más datos podríamos llegar a tener mucha más información y ver cómo han evolucionado los tópicos para ver las tendencias del sector.

Una hiperparametrización más detallada podría llegar a dar mejores resultados. Estoy satisfecho con los datos obtenidos ya que podemos ver como todo tiene sentido, pero como mejor parametrizado esté el modelo mejores conclusiones se extraen, con lo cual, mejores conclusiones.

Otro elemento a tener en cuenta, es que existen muchos más modelos que el LDA y el BerTopic. Existen modelos como el Non Negative Matrix Factorization (NMF), Latent Semantic Analysis (LSA), Parallel Latent Dirichlet Allocation (PLDA), Pachinko Allocation Model (PAM)... Sería interesante ver el comportamiento de todos estos modelos para determinar cuál va mejor.

Sería muy interesante configurar un sistema de análisis en *streaming* con Hadoop para ver día a día como son las evoluciones de las tendencias y los tópicos de distintos sectores de interés.

7.3 Futuro trabajo

Gracias a mi pasado no tecnológico, ya que mis estudios anteriores son en relación al marketing online, veo la gran importancia que tiene la aplicación del NLP en el mundo de los negocios y como puedes mejorar de una forma clara y objetiva. Tengo claro que voy a seguir mis pasos por este camino y voy a seguir formándome en este sector.

Decir también, que otro de los puntos fuertes de este trabajo es que este *script* funciona para analizar cualquier conjunto de valoraciones. Si mi intención es abrir un restaurante italiano en Barcelona, lo primero que voy a hacer es ejecutar este *script* con valoraciones de todos los restaurantes italianos de Barcelona o, incluso, segmentar directamente por ubicación y ver que es lo que valoran de este tipo de negocios en el barrio de Gracia. De esta forma voy a tener información objetiva muy útil para iniciar mi nuevo negocio y aplicar las inversiones necesarias a cada tópico para satisfacer directamente al público objetivo. Siempre se tiene que tener la mirada en el consumidor para progresar y prosperar.

8. Anexos

8.1 Anexo I – Gráficos y tablas

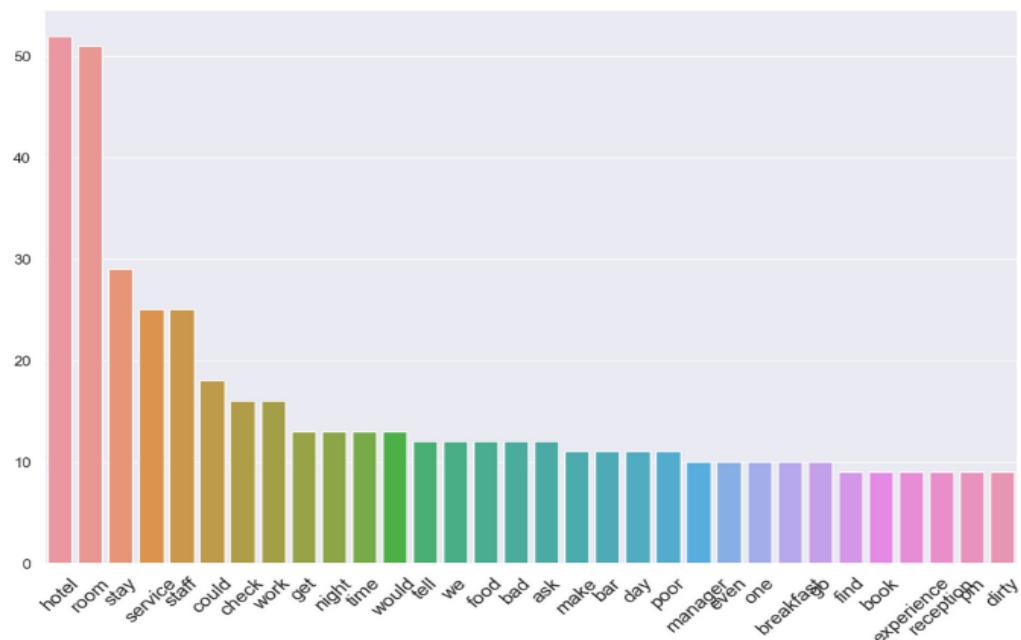


Gráfico 1

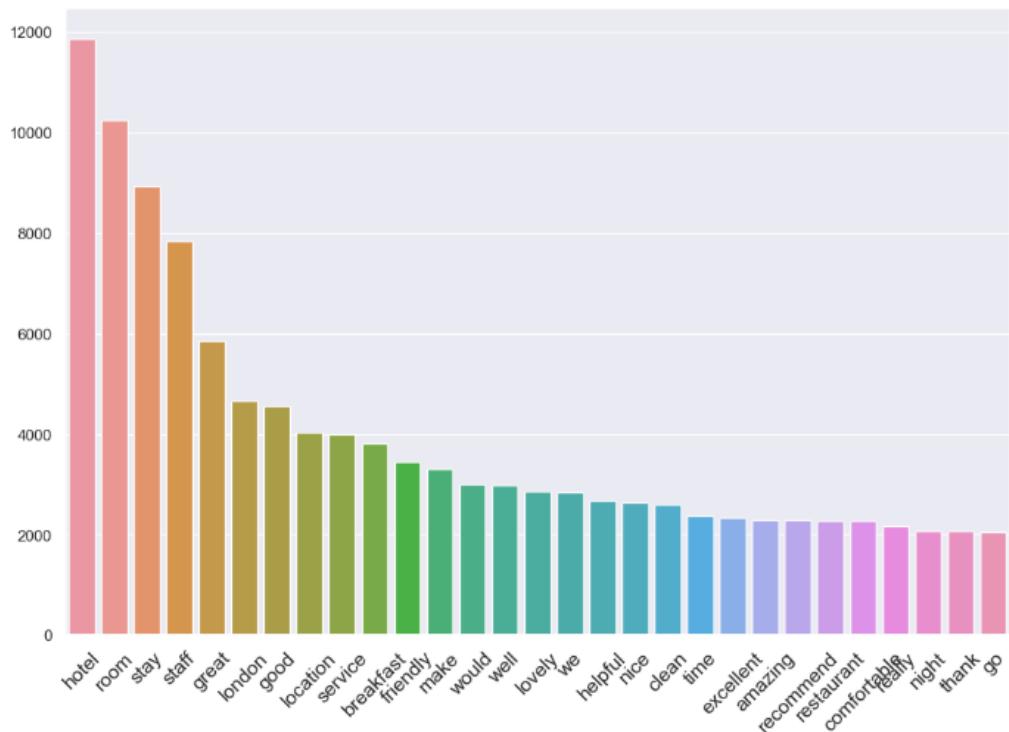


Gráfico 2

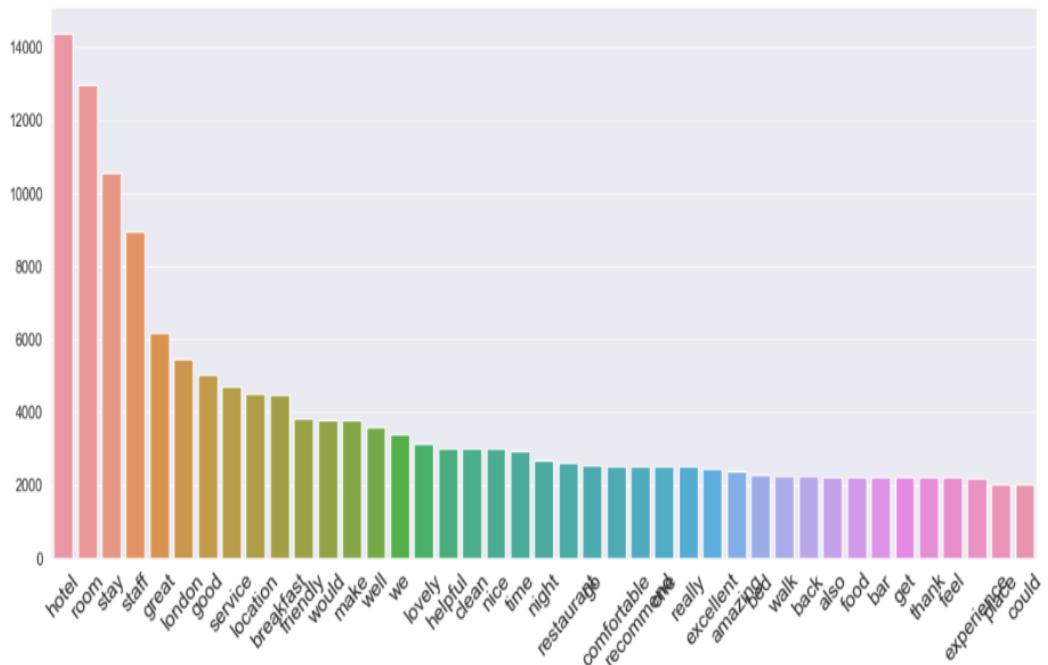


Gráfico 3

	Topic0	Topic1	Topic2	Topic3	dominant_topic
Doc0	0.080000	0.290000	0.010000	0.620000	3
Doc1	0.080000	0.520000	0.010000	0.410000	1
Doc2	0.010000	0.200000	0.010000	0.770000	3
Doc3	0.010000	0.160000	0.080000	0.760000	3
Doc4	0.210000	0.330000	0.050000	0.400000	3
Doc5	0.010000	0.340000	0.010000	0.640000	3
Doc6	0.020000	0.020000	0.950000	0.020000	2
Doc7	0.110000	0.240000	0.010000	0.640000	3
Doc8	0.010000	0.200000	0.670000	0.120000	2
Doc9	0.130000	0.180000	0.010000	0.680000	3
Doc10	0.010000	0.530000	0.010000	0.460000	1
Doc11	0.010000	0.290000	0.010000	0.700000	3
Doc12	0.010000	0.010000	0.010000	0.980000	3
Doc13	0.000000	0.520000	0.180000	0.300000	1
Doc14	0.010000	0.280000	0.010000	0.710000	3

Tabla 1

Topic Num	Num Documents
0	0
1	1
2	3
3	2

Tabla 2

Selected Topic 1 Previous Topic Next Topic Clear Topic

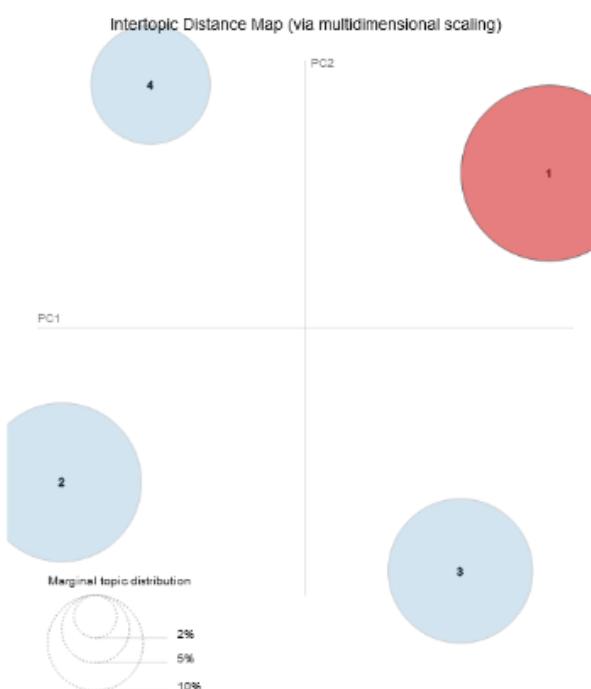


Gráfico 4

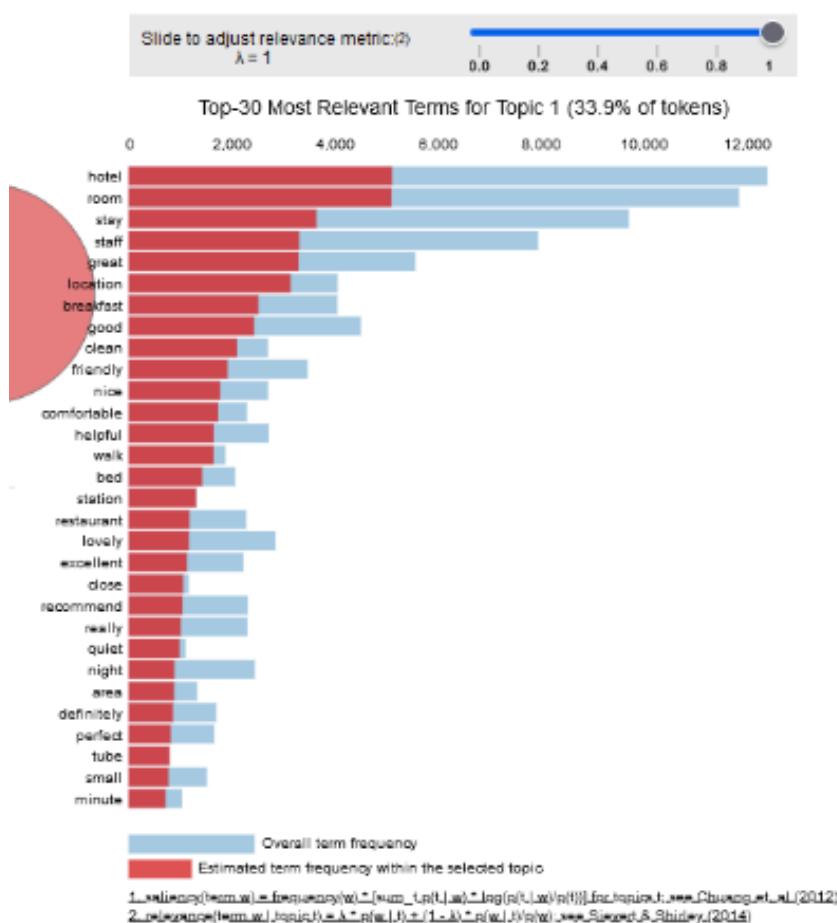


Gráfico 5

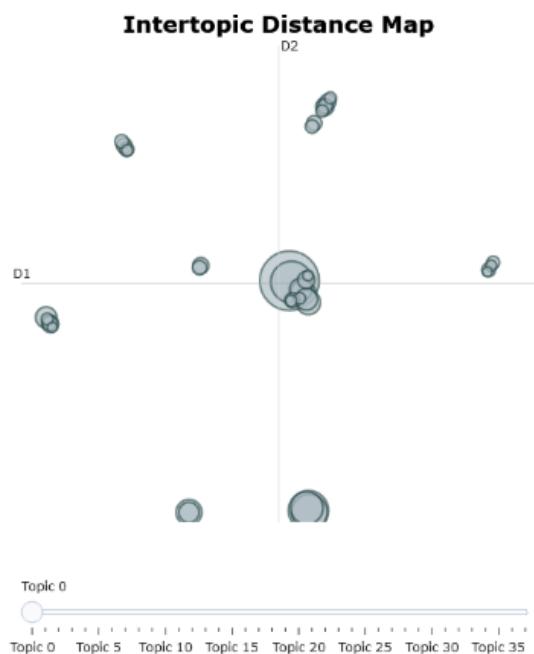


Gráfico 6

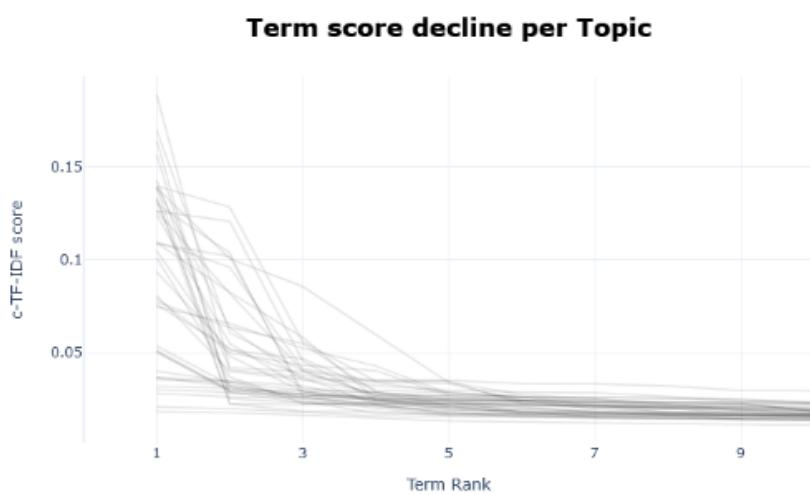


Gráfico 7

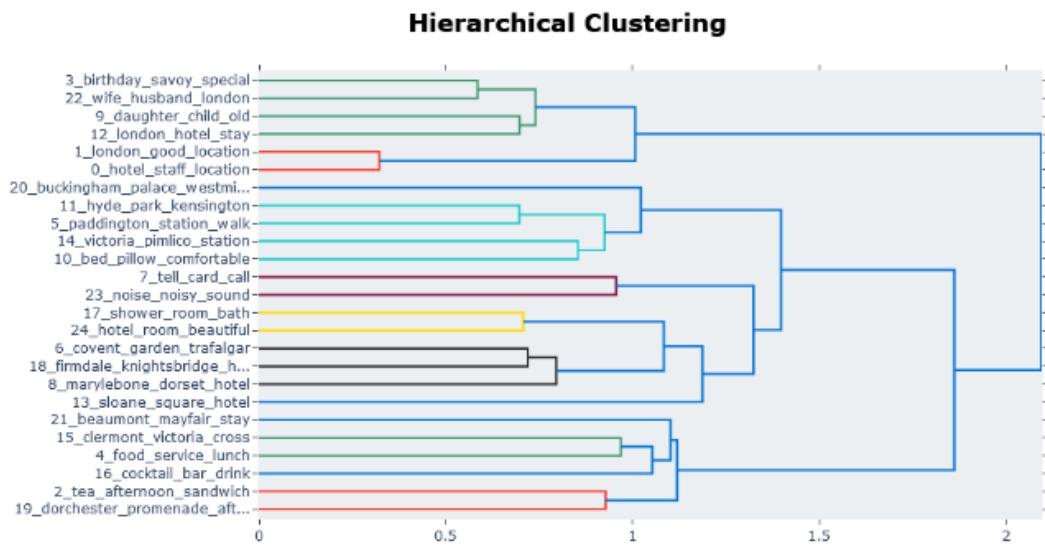


Gráfico 8

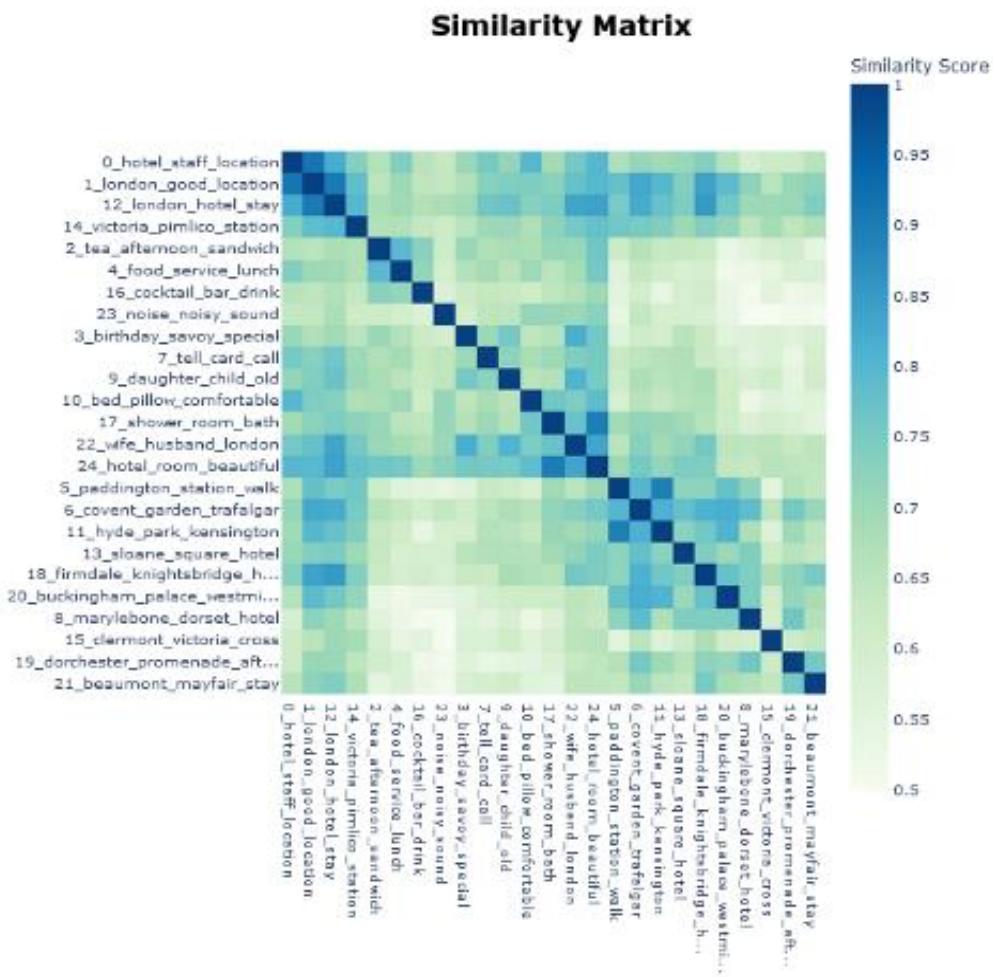


Gráfico 9

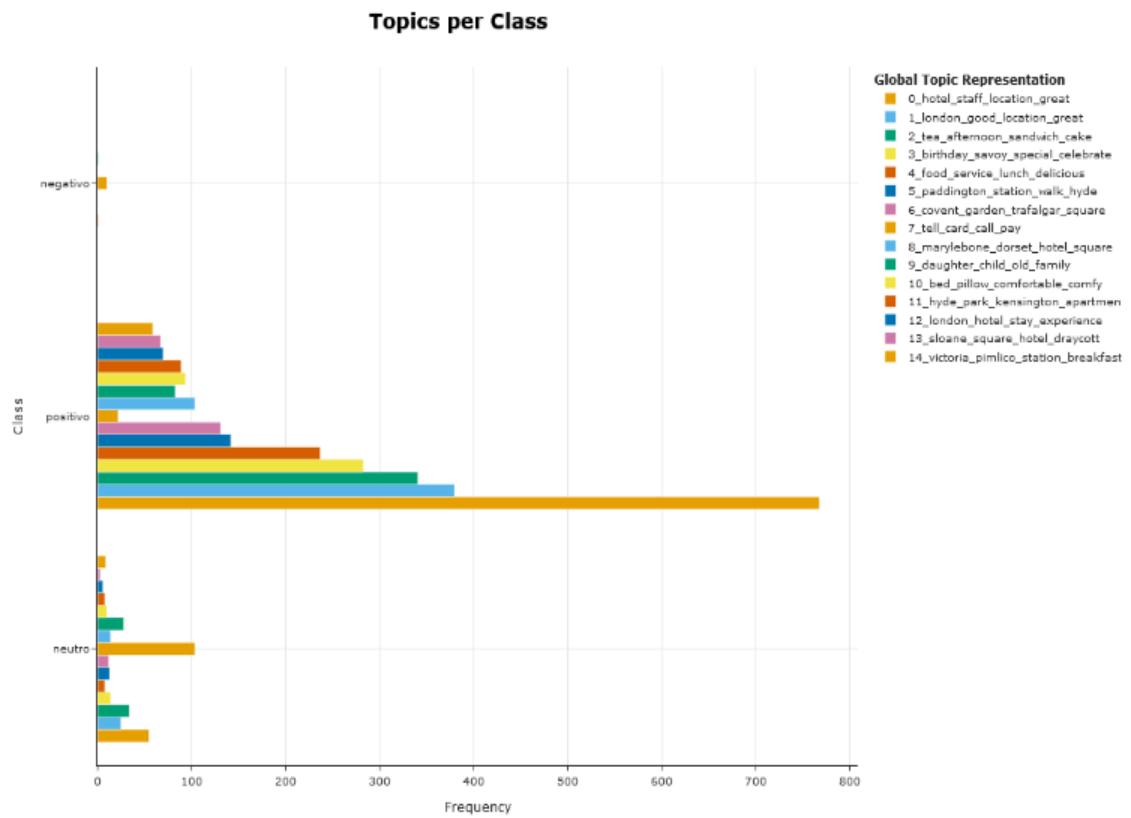


Gráfico 10

8.2 Anexo II – Web scrapping script

19/2/23, 22:11

Web Scrapping TripAdvisor - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [1]:

```
1 import requests
2 from bs4 import BeautifulSoup
3 import pandas as pd
4 import time
5 import re
6 import time
```

In [2]:

```
1 import warnings
2
3 warnings.filterwarnings("ignore")
```

In [3]:

```
1 headers = ({'User-Agent':
2     'Mozilla/5.0 (Windows NT 10.0; Win64; x64) \
3     AppleWebKit/537.36 (KHTML, like Gecko) \
4     Chrome/90.0.4430.212 Safari/537.36',
5     'Accept-Language': 'en-US, en;q=0.5'})
```

- Conseguir los nombres de los hoteles

In [4]:

```
1 url = "https://www.tripadvisor.co.uk/Hotels-g186338-zfc4-zfd21371-a_ufe.true-a_sort.POPULARITY-London_England-Hotels.html"
2 hotel_page = requests.get(url, headers=headers)
3 page = str(hotel_page.text)
4
5 names = re.findall(r'{"@type": "Hotel", "name": "(.*?)"', page)
6
7 df = pd.DataFrame(names, columns=['Hotel name'])
8
9 print(df)
```

	Hotel name
0	The Resident Covent Garden
1	The Montague on The Gardens
2	The Resident Victoria
3	The Chesterfield Mayfair
4	The Resident Soho
5	The Bloomsbury Hotel
6	Collingham Serviced Apartments
7	Wilde Aparthotels By Staycity London - Paddington
8	Vintry & Mercer
9	Canopy by Hilton London City
10	Holmes Hotel London
11	The Resident Kensington
12	The Clermont, Charing Cross
13	St. Ermin's Hotel, Autograph Collection
14	Artist Residence London
15	Sonder The Henry
16	Roseate House London
17	Park Grand London Kensington
18	The Clermont, Victoria
19	K+K Hotel George
20	Page 8
21	Lost Property St Paul's London - Curio Collect...
22	Hilton London Tower Bridge
23	Hyde Park International
24	Inhabit, Queen's Gardens
25	The Dixon, Tower Bridge, Autograph Collection
26	Wilde Aparthotels by Staycity London Aldgate T...
27	One Hundred Shoreditch
28	The Hide London
29	Apex London Wall Hotel

19/2/23, 22:11

Web Scrapping TripAdvisor - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [5]:

```

1 url = "https://www.tripadvisor.co.uk/Hotels-g186338-oa30-zfc4-zfd21371-a_sort.POPULARITY-London_England-Hotels.html"
2 hotel_page = requests.get(url, headers=headers)
3 page = str(hotel_page.text)
4
5 names = re.findall(r'{"@type": "Hotel", "name": "(.*?)"', page)
6
7 df2 = pd.DataFrame(names, columns=['Hotel name'])
8
9 print(df2)

```

	Hotel name
0	Mama Shelter London - Shoreditch
1	Radisson Blu Edwardian Mercer Street Hotel, Lo...
2	Park Grand London Hyde Park
3	Hard Rock Hotel London
4	The Laslett
5	Montagu Place Hotel
6	The Trafalgar St. James
7	The Henrietta Hotel
8	Hilton London Canary Wharf
9	Park Plaza County Hall London
10	The Windermere Hotel, London
11	The Zetter Townhouse, Marylebone
12	Thistle London Marble Arch
13	Club Quarters Hotel London, Trafalgar Square
14	The Hoxton Southwark
15	Sloane Square Hotel
16	Dorset Square Hotel
17	Flemings Mayfair
18	The Zetter Townhouse Clerkenwell
19	Hazlitt's Hotel
20	Treehouse Hotel London
21	The Rathbone Hotel Fitzrovia
22	Radisson Blu Edwardian New Providence Wharf Ho...
23	Lansbury Heritage Hotel
24	Apex Temple Court Hotel
25	The Caesar Hotel
26	Hotel Indigo London - Tower Hill, an IHG Hotel
27	Park Grand Paddington Court
28	Hotel nhnow London
29	The Zetter Hotel

In [6]:

```

1 url = "https://www.tripadvisor.co.uk/Hotels-g186338-oa60-zfc4-zfd21371-a_sort.POPULARITY-London_England-Hotels.html"
2 hotel_page = requests.get(url, headers=headers)
3 page = str(hotel_page.text)
4
5 names = re.findall(r'{"@type": "Hotel", "name": "(.*?)"', page)
6
7 df3 = pd.DataFrame(names, columns=['Hotel name'])
8
9 print(df3)

```

	Hotel name
0	Hotel Indigo London - Kensington, An IHG Hotel
1	Clayton Hotel City of London
2	InterContinental London - the O2, an IHG Hotel
3	Apex City of London Hotel
4	Morton Hotel
5	The Queen's Gate Hotel
6	St Paul's Hotel
7	London Bridge Hotel
8	St. James\u2019 Court, A Taj Hotel
9	The Hoxton, Shoreditch
10	The Level at Melia White House
11	Mercurie London Bridge
12	The Chamberlain Hotel
13	Ashburn Hotel
14	No Ten Manchester Street
15	Hilton London Metropole
16	citizenM London Shoreditch
17	Signature Townhouse Hyde Park
18	Hotel Xenia, Autograph Collection
19	The Hoxton, Holborn
20	Pullman London St. Pancras Hotel
21	Hart Shoreditch Hotel London, Curio Collection...
22	Thistle Piccadilly Hotel
23	Pestana Chelsea Bridge
24	Novotel London Bridge Hotel
25	Novotel London Tower Bridge
26	Thistle Holborn Hotel
27	Holiday Inn London - Whitechapel
28	Leonardo Royal Hotel London St. Paul's
29	Park Plaza Victoria London

In [7]:

```

1 url = "https://www.tripadvisor.co.uk/Hotels-g186338-oa90-zfc4-zfd21371-a_sort.POPULARITY-London_England-Hotels.html"
2 hotel_page = requests.get(url, headers=headers)
3 page = str(hotel_page.text)
4
5 names = re.findall(r'{"@type":"Hotel","name":"(.?)"', page)
6
7 df4 = pd.DataFrame(names, columns=['Hotel name'])
8
9 print(df4)

```

	Hotel name
0	The Megaro
1	The Gyle
2	The Exhibitionist Hotel
3	New Road Hotel
4	The Grand at Trafalgar Square
5	The Mandeville Hotel
6	H10 London Waterloo
7	Space Apart Hotel
8	The Gore London \u2013 Starhotels Collezione
9	citizenM Tower of London Hotel
10	The Westbridge Hotel
11	Malmaison London
12	Middle Eight
13	CitizenM London Bankside
14	Novotel London Excel
15	DoubleTree by Hilton London Greenwich
16	Novotel London Blackfriars
17	Leonardo Royal London Tower Bridge
18	Residence Inn by Marriott London Tower Bridge
19	Hyatt Place London City East
20	Novotel London Canary Wharf
21	Hotel 55
22	The Sumner Hotel
23	Leonardo Royal Hotel London City - Tower of Lo...
24	The Beaufort Hotel
25	The Rockwell
26	The Westminster London, Curio Collection by Hi...
27	Z at Gloucester Place
28	The Cavendish London Hotel
29	NYX Hotel London Holborn

In [8]:

```

1 url = "https://www.tripadvisor.co.uk/Hotels-g186338-oa120-zfc4-zfd21371-a_sort.POPULARITY-London_England-Hotels.html"
2 hotel_page = requests.get(url, headers=headers)
3 page = str(hotel_page.text)
4
5 names = re.findall(r'{"@type":"Hotel","name":"(.?)"', page)
6
7 df5 = pd.DataFrame(names, columns=['Hotel name'])
8
9 print(df5)

```

	Hotel name
0	Hotel Xanadu
1	Radisson Blu Edwardian Bloomsbury Street Hotel...
2	Danubius Hotel Regents Park
3	Park Plaza London Riverbank
4	Club Quarters Hotel London Covent Garden Holborn
5	The Z Hotel Piccadilly
6	Hotel Indigo London - Paddington
7	Boundary London
8	London Marriott Hotel Kensington
9	Radisson Blu Edwardian Vanderbilt Hotel, London
10	Karma Sanctum Soho
11	Lincoln Plaza London, Curio Collection by Hilton
12	Holiday Inn London - Camden Lock, an IHG Hotel
13	Mornington Hotel London Victoria
14	The Rembrandt
15	The Bailey's Hotel London Kensington
16	196 Bishopsgate
17	Crowne Plaza London - Ealing, an IHG Hotel
18	Hilton London Angel Islington
19	The Wellington by Blue Orchid
20	Novotel London Paddington
21	Thistle Trafalgar Square
22	Staunton Hotel
23	The Premier Notting Hill
24	London Marriott Hotel Regents Park
25	YOTEL London City
26	NH London Kensington
27	Good Hotel London
28	The Bermondsey Square Hotel
29	Grand Royale London Hyde Park

- Unión de los data frames creados

In [9]:

```
1 frames = [df, df2, df3, df4, df5]
2
3 result = pd.concat(frames)
```

In [10]:

```
1 result = result.reset_index(drop=True)
```

- Transformación

In [11]:

```
1 result = result['Hotel name'].tolist()
```

In [12]:

```
1 punctuation = ["-", ",", "!", ".", ";"]
2
3 for i in range(len(result)):
4     for symbol in punctuation:
5         result[i] = result[i].replace(symbol, "")
6
7 print(result)
```

[The Resident Covent Garden', 'The Montague on The Gardens', 'The Resident Victoria', 'The Chesterfield Mayfair', 'The Resident Soho', 'The Bloomsbury Hotel', 'Collingham Serviced Apartments', 'Vintry & Mercer', 'Wilde Aparthotels By Stay city London Paddington', 'Canopy by Hilton London City', 'Holmes Hotel London', 'The Resident Kensington', 'The Clermont Charing Cross', 'Artist Residence London', 'St Ermin's Hotel Autograph Collection', 'Sonder The Henry', 'Roseate House London', 'Park Grand London Kensington', 'The Clermont Victoria', 'Page 8', 'K+K Hotel George', 'Hilton London Tower Bridge', 'The Dixon Tower Bridge Autograph Collection', 'Inhabit Queen's Gardens', 'Wilde Aparthotels by Staycity London Aldgate Tower Bridge', 'Apex London Wall Hotel', 'Hyde Park International', 'One Hundred Shoreditch', 'The Hide London', "Lost Property St Paul's London Curio Collection by Hilton", 'Mama Shelter London Shoreditch', 'Radisson Blu Edwardian Mercer Street Hotel London', 'Park Grand London Hyde Park', 'Hard Rock Hotel London', 'The Laslett', 'Montagu Place Hotel', 'The Trafalgar St James', 'The Henrietta Hotel', 'Hilton London Canary Wharf', 'Park Plaza County Hall London', 'The Windermere Hotel London', 'The Zetter Townhouse Marylebone', 'Thistle London Marble Arch', 'Club Quarters Hotel London Trafalgar Square', 'The Hoxton Southwark', 'Sloane Square Hotel', 'Dorset Square Hotel', 'Flemings Mayfair', 'The Zetter Townhouse Clerkenwell', 'Hazlitt's Hotel', 'Treehouse Hotel London', 'The Rathbone Hotel Fitzrovia', 'Radisson Blu Edwardian New Providence Wharf Hotel London', 'Lansbury Heritage Hotel', 'Apex Temple Court Hotel', 'The Caesar Hotel', 'Hotel Indigo London Tower Hill an IHG Hotel', 'Park Grand Paddington Court', 'Hotel nhow London', 'The Zetter Hotel', 'Hotel Indigo London Kensington An IHG Hotel', 'Clayton Hotel City of London', 'InterContinental London the O2 an IHG Hotel', 'Apex City of London Hotel', 'Morton Hotel', 'The Queen's Gate Hotel', 'St Paul's Hotel', 'London Bridge Hotel', 'St James\u2019s Court A Taj Hotel', 'The Hoxton Shoreditch', 'The Level at Melia White House', 'Mercure London Bridge', 'The Chamberlain Hotel', 'Ashburn Hotel', 'No Ten Manchester Street', 'Hilton London Metropole', 'citizenM London Shoreditch', 'Signature Townhouse Hyde Park', 'Hotel Xenia Autograph Collection', 'The Hoxton Holborn', 'Pullman London St Pancras Hotel', 'Hart Shoreditch Hotel London Curio Collection by Hilton', 'Thistle Piccadilly Hotel', 'Pestana Chelsea Bridge', 'Novotel London Bridge Hotel', 'Novotel London Tower Bridge', 'Thistle Holborn Hotel', 'Holiday Inn London Whitechapel', 'Leonardo Royal Hotel London St Paul's', 'Park Plaza Victoria London', 'The Megaro', 'The Gyle', 'The Exhibitionist Hotel', 'New Road Hotel', 'The Grand at Trafalgar Square', 'The Mandeville Hotel', 'H10 London Waterloo', 'Space Apart Hotel', 'The Grove London \u2019s Starhotels Collezione', 'citizenM Tower of London Hotel', 'The Westbridge Hotel', 'Maison London', 'Middle Eight', 'CitizenM London Bankside', 'Novotel London Excel', 'DoubleTree by Hilton London Greenwich', 'Novotel London Blackfriars', 'Leonardo Royal London Tower Bridge', 'Residence Inn by Marriott London Tower Bridge', 'Hyatt Place London City East', 'Novotel London Canary Wharf', 'Hotel 55', 'The Summer Hotel', 'Leonardo Royal Hotel London City Tower of London', 'The Beaufort Hotel', 'The Rockwell', 'The Westminster London Curio Collection by Hilton', 'Z at Gloucester Place', 'The Cavendish London Hotel', 'NYX Hotel London Holborn', 'Hotel Xanadu', 'Radisson Blu Edwardian Bloomsbury Street Hotel London', 'Danubius Hotel Regents Park', 'Park Plaza London Riverbank', 'Club Quarters Hotel London Covent Garden Holborn', 'The Z Hotel Piccadilly', 'Hotel Indigo London Paddington', 'Boundary London', 'London Marriott Hotel Kensington', 'Radisson Blu Edwardian Vanderbilt Hotel London', 'Karma Sanctum Soho', 'Lincoln Plaza London Curio Collection by Hilton', 'Holiday Inn London Camden Lock an IHG Hotel', 'Mornington Hotel London Victoria', 'The Rembrandt', 'The Bailey's Hotel London Kensington', '196 Bishopsgate', 'Crowne Plaza London Ealing an IHG Hotel', 'Hilton London Angel Islington', 'The Wellington by Blue Orchid', 'Novotel London Paddington', 'Thistle Trafalgar Square', 'Staunton Hotel', 'The Premier Notting Hill', 'London Marriott Hotel Regents Park', 'YOTEL London City', 'NH London Kensington', 'Good Hotel London', 'The Bermondsey Square Hotel', 'Grand Royale London Hyde Park']

In [13]:

```

1 for i in range(len(result)):
2     result[i] = result[i].replace(" ", "_")
3
4 print(result)

```

['The_Resident_Covent_Garden', 'The_Montague_on_The_Gardens', 'The_Resident_Victoria', 'The_Chesterfield_Mayfair', 'The_Resident_Soho', 'The_Bloomsbury_Hotel', 'Collingham_Serviced_Apartments', 'Vintry_&_Mercer', 'Wilde_Aparthotels_By_Sta
ycity_London_Paddington', 'Canopy_by_Hilton_London_City', 'Holmes_Hotel_London', 'The_Resident_Kensington', 'The_Clerm
ont_Charing_Cross', 'Artist_Residence_London', 'St_Ermin's_Hotel_Autograph_Collection', 'Sonder_The_Henry', 'Roseate_Ho
use_London', 'Park_Grand_London_Kensington', 'The_Clermont_Victoria', 'Page_8', 'K+K_Hotel_George', 'Hilton_London_Towe
r_Bridge', 'The_Dixon_Tower_Bridge_Autograph_Collection', 'Inhabit_Queen's_Gardens', 'Wilde_Aparthotels_by_Staycity_Lon
don_Aldgate_Tower_Bridge', 'Apex_London_Wall_Hotel', 'Hyde_Park_International', 'One_Hundred_Shoreditch', 'The_Hide_Lon
don', 'Lost_Property_St_Paul's_London_Curio_Collection_by_Hilton', 'Mama_Shelter_London_Shoreditch', 'Radisson_Blu_Ed
wardian_Mercer_Street_Hotel_London', 'Park_Grand_London_Hyde_Park', 'Hard_Rock_Hotel_London', 'The_Laslett', 'Montagu_P
lace_Hotel', 'The_Trafalgar_St_James', 'The_Henrietta_Hotel', 'Hilton_London_Canary_Wharf', 'Park_Plaza_County_Hall_Lon
don', 'The_Windermere_Hotel_London', 'The_Zetter_Townhouse_Marylebone', 'Thistle_London_Marble_Arch', 'Club_Qarters_Ho
tel_London_Trafalgar_Square', 'The_Hoxton_Southwark', 'Sloane_Square_Hotel', 'Dorset_Square_Hotel', 'Flemings_Mayfair',
'The_Zetter_Townhouse_Clerkenwell', 'Hazlitt's_Hotel', 'Treehouse_Hotel_London', 'The_Rathbone_Hotel_Fitzrovia', 'Radiss
on_Blu_Ewardian_New_Providence_Wharf_Hotel_London', 'Lansbury_Heritage_Hotel', 'Apex_Temple_Court_Hotel', 'The_Caesars
Hotel', 'Hotel_Indigo_London_Tower_Hill_an_IHG_Hotel', 'Park_Grand_Paddington_Court', 'Hotel_nhow_London', 'The_Zetter
_Hotel', 'Hotel_Indigo_London_Kensington_An_IHG_Hotel', 'Clayton_Hotel_City_of_London', 'InterContinental_London_the
_O2_an_IHG_Hotel', 'Apex_City_of_London_Hotel', 'Morton_Hotel', 'The_Queen's_Gate_Hotel', 'St_Paul's_Hotel', 'London_Bri
dge_Hotel', 'St_James\u2019_Court_A_Taj_Hotel', 'The_Hoxton_Shoreditch', 'The_Level_at_Melia_White_House', 'Mercurie_Lo
ndon_Bridge', 'The_Chamberlain_Hotel', 'Ashburn_Hotel', 'No_Ten_Manchester_Street', 'Hilton_London_Metropole', 'citizen
M_London_Shoreditch', 'Signature_Townhouse_Hyde_Park', 'Hotel_Xenia_Autograph_Collection', 'The_Hoxton_Holborn', 'Pulm
an_London_St_Pancras_Hotel', 'Hart_Shoreditch_Hotel_London_Curio_Collection_by_Hilton', 'Thistle_Piccadilly_Hotel', 'Pe
stana_Chelsea_Bridge', 'Novotel_London_Bridge_Hotel', 'Novotel_London_Tower_Bridge', 'Thistle_Holborn_Hotel', 'Holiday
Inn_London_Whitechapel', 'Leonardo_Royal_Hotel_London_St_Paul's', 'Park_Plaza_Victoria_London', 'The_Megaro', 'The_Gyl
e', 'The_Exhibitionist_Hotel', 'New_Road_Hotel', 'The_Grand_at_Trafalgar_Square', 'The_Mandeville_Hotel', 'H10_London_W
aterloo', 'Space_Apart_Hotel', 'The_Gore_London_\u2019Starhotels_Collezione', 'citizenM_Tower_of_London_Hotel', 'The
Westbridge_Hotel', 'Malmaison_London', 'Middle_Eight', 'CitizenM_London_Bankside', 'Novotel_London_Excel', 'DoubleTree
by_Hilton_London_Greenwich', 'Novotel_London_Blackfriars', 'Leonardo_Royal_London_Tower_Bridge', 'Residence_Inn_by_Marr
iott_London_Tower_Bridge', 'Hyatt_Place_London_City_East', 'Novotel_London_Canary_Wharf', 'Hotel_55', 'The_Sunner_Hote
l', 'Leonardo_Royal_Hotel_London_City_Tower_of_London', 'The_Beaufort_Hotel', 'The_Rockwell', 'The_Westminster_London
Curio_Collection_by_Hilton', 'Z_at_Gloucester_Place', 'The_Cavendish_London_Hotel', 'NYX_Hotel_London_Holborn', 'Hotel
Xanadu', 'Radisson_Blu_Ewardian_Bloomsbury_Street_Hotel_London', 'Danubius_Hotel_Regents_Park', 'Park_Plaza_London_Riv
erbanks', 'Club_Qarters_Hotel_London_Covent_Garden_Holborn', 'The_Z_Hotel_Piccadilly', 'Hotel_Indigo_London_Paddingto
n', 'Boundary_London', 'London_Marriott_Hotel_Kensington', 'Radisson_Blu_Ewardian_Vanderbilt_Hotel_London', 'Karma_San
ctum_Soho', 'Lincoln_Plaza_London_Curio_Collection_by_Hilton', 'Holiday_Inn_London_Camden_Lock_an_IHG_Hotel', 'Morning
ton_Hotel_London_Victoria', 'The_Rembrandt', 'The_Bailey's_Hotel_London_Kensington', '196_Bishopsgate', 'Crown_Plaza_L
ondon_Ealing_an_IHG_Hotel', 'Hilton_London_Angel_Islington', 'The_Wellington_by_Blue_Orchid', 'Novotel_London_Padding
ton', 'Thistle_Trafalgar_Square', 'Staunton_Hotel', 'The_Premier_Nutting_Hill', 'London_Marriott_Hotel_Regents_Park', 'Y
OTEL_London_City', 'NH_London_Kensington', 'Good_Hotel_London', 'The_Bermondsey_Square_Hotel', 'Grand_Royale_London_Hy
e_Park']

In [14]:

```

1 for i in range(len(result)):
2     result[i] = result[i].replace("", "_")
3
4 print(result)

```

['The_Resident_Covent_Garden', 'The_Montague_on_The_Gardens', 'The_Resident_Victoria', 'The_Chesterfield_Mayfair', 'The_Resident_Soho', 'The_Bloomsbury_Hotel', 'Coillingham_Serviced_Apartments', 'Vintry_&_Mercer', 'Wilde_Aparthotels_By_Sta
ycity_London_Paddington', 'Canopy_by_Hilton_London_City', 'Holmes_Hotel_London', 'The_Resident_Kensington', 'The_Clerm
ont_Charing_Cross', 'Artist_Residence_London', 'St_Ermin_s_Hotel_Autograph_Collection', 'Sonder_The_Henry', 'Roseate_Ho
use_London', 'Park_Grand_London_Kensington', 'The_Clermont_Victoria', 'Page_8', 'K&K_Hotel_George', 'Hilton_London_Tow
r_Bridge', 'The_Dixon_Tower_Bridge_Autograph_Collection', 'Inhabit_Queen_s_Gardens', 'Wilde_Aparthotels_by_Stacycity_Lon
don_Aldgate_Tower_Bridge', 'Apex_London_Wall_Hotel', 'Hyde_Park_International', 'One_Hundred_Shoreditch', 'The_Hide_Lon
don', 'Lost_Property_St_Paul_s_London_Curio_Collection_by_Hilton', 'Mama_Shelter_London_Shoreditch', 'Radisson_Blu_Ed
wardian_Mercer_Street_Hotel_London', 'Park_Grand_London_Hyde_Park', 'Hard_Rock_Hotel_London', 'The_Laslett', 'Montagu_P
lace_Hotel', 'The_Trafalgar_St_James', 'The_Henrietta_Hotel', 'Hilton_London_Canary_Wharf', 'Park_Plaza_County_Hall_Lon
don', 'The_Windermere_Hotel_London', 'The_Zetter_Townhouse_Marylebone', 'Thistle_London_Marble_Arch', 'Club_Qarters_Ho
tel_London_Trafalgar_Square', 'The_Hoxton_Southwark', 'Sloane_Square_Hotel', 'Dorset_Square_Hotel', 'Flemings_Mayfair',
'The_Zetter_Townhouse_Clerkenwell', 'Hazlitt_s_Hotel', 'Treehouse_Hotel_London', 'The_Rathbone_Hotel_Fitrovia', 'Radiss
on_Blu_Ewardian_New_Providence_Wharf_Hotel_London', 'Lansbury_Heritage_Hotel', 'Apex_Temple_Court_Hotel', 'The_Caesar_
Hotel', 'Hotel_Indigo_London_Tower_Hill_an_IHG_Hotel', 'Park_Grand_Paddington_Court', 'Hotel_nhow_London', 'The_Zetter
_Hotel', 'Hotel_Indigo_London_Kensington_An_IHG_Hotel', 'Clayton_Hotel_City_of_London', 'InterContinental_London_the_02_an_IHG_Hotel', 'Apex_City_of_London_Hotel', 'Morton_Hotel', 'The_Queen_s_Gate_Hotel', 'St_Paul_s_Hotel', 'London_Bri
dge_Hotel', 'St_James\u2019_Court_A_Taj_Hotel', 'The_Hoxton_Shoreditch', 'The_Level_at_Melia_White_House', 'Mercure_Lo
ndon_Bridge', 'The_Chamberlain_Hotel', 'Ashburn_Hotel', 'No_Ten_Manchester_Street', 'Hilton_London_Metropole', 'citizen
M_London_Shoreditch', 'Signature_Townhouse_Hyde_Park', 'Hotel_Xenia_Autograph_Collection', 'The_Hoxton_Holborn', 'Pullm
an_London_St_Pancras_Hotel', 'Hart_Shoreditch_Hotel_London_Curio_Collection_by_Hilton', 'Thistle_Piccadilly_Hotel', 'Pe
stana_Chelsea_Bridge', 'Novotel_London_Bridge_Hotel', 'Novotel_London_Tower_Bridge', 'Thistle_Holborn_Hotel', 'Holiday
Inn_London_Whitechapel', 'Leonardo_Royal_Hotel_London_St_Paul_s', 'Park_Plaza_Victoria_London', 'The_Megaro', 'The_Gyl
e', 'The_Exhibitionist_Hotel', 'New_Road_Hotel', 'The_Grand_at_Trafalgar_Square', 'The_Mandeville_Hotel', 'H10_London_W
aterloo', 'Space_Apart_Hotel', 'The_Gore_London_\u20192013_Starhotels_Collezione', 'citizenM_Tower_of_London_Hotel', 'The
Westbridge_Hotel', 'Malmaison_London', 'Middle_Eight', 'CitizenM_London_Bankside', 'Novotel_London_Excel', 'DoubleTree
by_Hilton_London_Greenwich', 'Novotel_London_Blackfriars', 'Leonardo_Royal_London_Tower_Bridge', 'Residence_Inn_by_Marr
iott_London_Tower_Bridge', 'Hyatt_Place_London_City_East', 'Novotel_London_Canary_Wharf', 'Hotel_55', 'The_Summer_Hote
l', 'Leonardo_Royal_Hotel_London_City_Tower_of_London', 'The_Beaufort_Hotel', 'The_Rockwell', 'The_Westminster_London
Curio_Collection_by_Hilton', 'Z_at_Gloucester_Place', 'The_Cavendish_London_Hotel', 'NYX_Hotel_London_Holborn', 'Hotel_X
anadu', 'Radisson_Blu_Ewardian_Bloomsbury_Street_Hotel_London', 'Danubius_Hotel_Regents_Park', 'Park_Plaza_London_Riv
erbanks', 'Club_Qarters_Hotel_London_Covent_Garden_Holborn', 'The_Z_Hotel_Piccadilly', 'Hotel_Indigo_London_Paddingto
n', 'Boundary_London', 'London_Marriott_Hotel_Kensington', 'Radisson_Blu_Ewardian_Vanderbilt_Hotel_London', 'Karma_San
ctum_Soho', 'Lincoln_Plaza_London_Curio_Collection_by_Hilton', 'Holiday_Inn_London_Camden_Lock_an_IHG_Hotel', 'Morning
ton_Hotel_London_Victoria', 'The_Rembrandt', 'The_Bailey_s_Hotel_London_Kensington', '196_Bishopsgate', 'Crownie_Plaza_L
ondon_Ealing_an_IHG_Hotel', 'Hilton_London_Angel_Islington', 'The_Wellington_by_Blue_Orchid', 'Novotel_London_Paddingt
on', 'Thistle_Trafalgar_Square', 'Staunton_Hotel', 'The_Premier_Notting_Hill', 'London_Marriott_Hotel_Regents_Park', 'Y
OTEL_London_City', 'NH_London_Kensington', 'Good_Hotel_London', 'The_Bermondsey_Square_Hotel', 'Grand_Royale_London_Hyd
e_Park']

In [15]:

```
1 result.remove('The_Gore_London_\u20192013_Starhotels_Collezione')
```

In [70]:

```
1 Nombres_hoteles_TFM = pd.DataFrame(result)
```

In [71]:

```
1 Nombres_hoteles_TFM.shape
```

Out[71]:

(149, 1)

In [19]:

```

1 from google.colab import files
2
3 Nombres_hoteles_TFM.to_csv('nombres_hoteles_TFM.csv', index=True)
4
5 files.download('nombres_hoteles_TFM.csv')

```

<IPython.core.display.Javascript object>
<IPython.core.display.Javascript object>

- Conseguir la información deseada de cada hotel

Tengo un error, que no se identificar el motivo, seguramente hay un nombre o un valor que no permite la ejecución global. He decidido tratar en listas de 10 para poder solucionar este error.

In [16]:

```
1 result_final = result[0:100]
```

19/2/23, 22:11

Web Scrapping TripAdvisor - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [28]:

```
1 result1 = result_final[0:9]
```

In [32]:

```
1 result2 = result_final[10:19]
```

In [33]:

```
1 result3 = result_final[20:29]
```

In [34]:

```
1 result4 = result_final[30:39]
```

In [35]:

```
1 result5 = result_final[40:49]
```

In [36]:

```
1 result6 = result_final[50:59]
```

In [37]:

```
1 result7 = result_final[60:69]
```

In [38]:

```
1 result8 = result_final[70:79]
```

In [39]:

```
1 result9 = result_final[80:89]
```

In [40]:

```
1 result10 = result_final[90:100]
```

In [29]:

```

1 start_time = time.time()
2
3 d = []
4
5 for hotel in result1:
6     time.sleep(3)
7
8     for i in range(0, 15):
9         offset2 = i*10
10        url = f"https://www.tripadvisor.co.uk/Hotel_Review-g186338-d15134890-Reviews-or{offset2}-{hotel}-London_England.html#REVIEWS"
11
12        result = requests.get(url,headers=headers,timeout=5,verify=False)
13        soup = BeautifulSoup(result.content, 'html.parser')
14        all_reviews = soup.find_all("div", {"class": "YibK1 MC R2 Gi z Z BB pBbQr"})
15
16        for div in all_reviews:
17            Nombre = div.find("a", {"class": "ui_header_link uyyBf"}).text
18            Titulo = div.find("div", {"class": "KgQgP MC _S b S6 H5 _a"}).text
19            Review = div.find("div", {"class": "fIrgE _T"}).text
20            Data = div.find("span", {"class": "teHYY _R Me S4 H3"}).text
21            d.append(
22                {
23                    'Nombre': Nombre,
24                    'Titulo': Titulo,
25                    'Review': Review,
26                    'Data': Data,
27                }
28            )
29
30 end_time = time.time()
31 print(f"Scraping time was: {end_time-start_time}")
32 df = pd.DataFrame(d)
33
34 df.to_csv("test.csv", index=False, sep=',')
35 df.head()

```

Scraping time was: 348.03094506263733

Out[29]:

	Nombre	Titulo	Review	Data
0	nimrodk803	Small room, no service	On the one hand, very friendly and welcoming s...	Date of stay: February 2023
1	NicolasM807	Charming Hotel in a Perfect Location	This was our first trip to London, so we wanted...	Date of stay: August 2022
2	LouisaBean	Fantastic location, friendly staff, comfortabl...	Great location, room was so comfortable and cle...	Date of stay: January 2023
3	Eastcoastwil	Wonderful Location for Theatre	We returned to The Resident Covent Garden beca...	Date of stay: February 2023
4	yenalubu	Great memories in London	So happy to stay here for my first solo travel...	Date of stay: February 2023

In [30]:

1 df.shape

Out[30]:

(1350, 4)

19/2/23, 22:11

Web Scrapping TripAdvisor - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [31]:

1 df.head(15)

Out[31]:

	Nombre	Titulo	Review	Data
0	nimrodk803	Small room, no service	On the one hand, very friendly and welcoming s...	Date of stay: February 2023
1	NicolasM807	Charming Hotel in a Perfect Location	This was our first trip to London, so we wante...	Date of stay: August 2022
2	LouisaBean	Fantastic location, friendly staff, comfortabl...	Great location, room was so comfortable and cl...	Date of stay: January 2023
3	Eastcoastwil	Wonderful Location forTheatre	We returned to The Resident Covent Garden beca...	Date of stay: February 2023
4	yenalubu	Great memories in London	So happy to stay here for my first solo travel...	Date of stay: February 2023
5	Boss-man56	Excellent	Just spent 3 days at The Resident Covent Garde...	Date of stay: February 2023
6	Discover647381	Great Place	Everything was perfect everything you could wa...	Date of stay: January 2023
7	gufton	Customer Service Excellence	The best hotel I've stayed at in London. The c...	Date of stay: February 2023
8	meltmetopuz	Perfect	The best hotel i have ever stayed in London. T...	Date of stay: January 2023
9	FrancesParis	Reliable comfort and convenience	Very comfortable and well equipped room in a h...	Date of stay: January 2023
10	Ella W	Wonderful Hotel, Book it!	I've read soo many reviews before selecting th...	Date of stay: January 2023
11	c11ffc	Birthday celebration	We needed somewhere to stay in Covent Garden a...	Date of stay: January 2023
12	Gabriel R	Simple but excellent hotel in the heart of Lon...	If you don't need unnecessary luxury but love ...	Date of stay: January 2023
13	vaksdenise	Great hotel in Covent Garden	Great room and bathroom, very attentive staff....	Date of stay: January 2023
14	Kevin	Impeccable Service	We stayed here in a King room for 2 nights in ...	Date of stay: January 2023

In [41]:

```

1 start_time = time.time()
2
3 g = []
4
5 for hotel in result2:
6     time.sleep(3)
7
8     for i in range(0, 15):
9         offset2 = i*10
10        url = f"https://www.tripadvisor.co.uk/Hotel_Review-g186338-d15134890-Reviews-or{offset2}-{hotel}-London_England.html#REVIEWS"
11
12        result = requests.get(url,headers=headers,timeout=5,verify=False)
13        soup = BeautifulSoup(result.content, 'html.parser')
14        all_reviews = soup.find_all("div", {"class": "YibK1 MC R2 Gi z Z BB pBbQr"})
15
16        for div in all_reviews:
17            Nombre = div.find("a", {"class": "ui_header_link uyyBf"}).text
18            Titulo = div.find("div", {"class": "KgQgP MC_S b S6 H5_a"}).text
19            Review = div.find("div", {"class": "fIrGe _T"}).text
20            Data = div.find("span", {"class": "teHYY _R Me S4 H3"}).text
21            g.append(
22                {
23                    'Nombre': Nombre,
24                    'Titulo': Titulo,
25                    'Review': Review,
26                    'Data': Data,
27                }
28            )
29
30    end_time = time.time()
31    print(f"Scraping time was: {end_time-start_time}")
32    df2 = pd.DataFrame(g)
33
34    df2.to_csv("test.csv", index=False, sep=',')
35    df2.head()

```

Scraping time was: 357.06658720970154

Out[41]:

	Nombre	Titulo	Review	Data
0	nimrodk803	Small room, no service	On the one hand, very friendly and welcoming s...	Date of stay: February 2023
1	NicolasM807	Charming Hotel in a Perfect Location	This was our first trip to London, so we wante...	Date of stay: August 2022
2	LouisaBean	Fantastic location, friendly staff, comfortabl...	Great location, room was so comfortable and cl...	Date of stay: January 2023
3	Eastcoastwil	Wonderful Location forTheatre	We returned to The Resident Covent Garden beca...	Date of stay: February 2023
4	yenalubu	Great memories in London	So happy to stay here for my first solo travel...	Date of stay: February 2023

19/2/23, 22:11

Web Scrapping TripAdvisor - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [45]:

```
1 df2.shape
```

Out[45]:

(1340, 4)

In [46]:

```
1 df2.head(15)
```

Out[46]:

	Nombre	Titulo	Review	Data
0	nimrodk803	Small room, no service	On the one hand, very friendly and welcoming s...	Date of stay: February 2023
1	NicolasM807	Charming Hotel in a Perfect Location	This was our first trip to London, so we wanted...	Date of stay: August 2022
2	LouisaBean	Fantastic location, friendly staff, comfortable...	Great location, room was so comfortable and cle...	Date of stay: January 2023
3	Eastcoastwil	Wonderful Location for Theatre	We returned to The Resident Covent Garden beca...	Date of stay: February 2023
4	yenalubu	Great memories in London	So happy to stay here for my first solo travel...	Date of stay: February 2023
5	Boss-man56	Excellent	Just spent 3 days at The Resident Covent Garde...	Date of stay: February 2023
6	Discover647381	Great Place	Everything was perfect everything you could wa...	Date of stay: January 2023
7	gufton	Customer Service Excellence	The best hotel I've stayed at in London. The c...	Date of stay: February 2023
8	meltemtopuz	Perfect	The best hotel i have ever stayed in London. T...	Date of stay: January 2023
9	FrancesParis	Reliable comfort and convenience	Very comfortable and well equipped room in a h...	Date of stay: January 2023
10	Ella W	Wonderful Hotel, Book it!	I've read so many reviews before selecting th...	Date of stay: January 2023
11	c11ffc	Birthday celebration	We needed somewhere to stay in Covent Garden a...	Date of stay: January 2023
12	Gabriel R	Simple but excellent hotel in the heart of Lon...	If you don't need unnecessary luxury but love ...	Date of stay: January 2023
13	vaksdenise	Great hotel in Covent Garden	Great room and bathroom, very attentive staff...	Date of stay: January 2023
14	Kevin	Impeccable Service	We stayed here in a King room for 2 nights in ...	Date of stay: January 2023

In [42]:

```

1 start_time = time.time()
2
3 k = []
4
5 for hotel in result3:
6     time.sleep(3)
7
8     for i in range(0, 15):
9         offset2 = i*10
10        url = f"https://www.tripadvisor.co.uk/Hotel_Review-g186338-d15134890-Reviews-or{offset2}-{hotel}-London_England.html#REVIEWS"
11
12        result = requests.get(url,headers=headers,timeout=5,verify=False)
13        soup = BeautifulSoup(result.content, 'html.parser')
14        all_reviews = soup.find_all("div", {"class": "YibK1 MC R2 Gi z Z BB pBbQr"})
15
16        for div in all_reviews:
17            Nombre = div.find("a", {"class": "ui_header_link uyyBf"}).text
18            Titulo = div.find("div", {"class": "KgQgP MC _S b S6 H5 _a"}).text
19            Review = div.find("div", {"class": "fIRGe _T"}).text
20            Data = div.find("span", {"class": "teHYY _R Me S4 H3"}).text
21            k.append(
22                {
23                    'Nombre': Nombre,
24                    'Titulo': Titulo,
25                    'Review': Review,
26                    'Data': Data,
27                }
28            )
29
30        end_time = time.time()
31    print(f"Scraping time was: {end_time-start_time}")
32 df3 = pd.DataFrame(k)
33
34 df3.to_csv("test.csv", index=False, sep=',')
35 df3.head()

```

Scraping time was: 347.3837511539459

Out[42]:

	Nombre	Titulo	Review	Data
0	nimrodk803	Small room, no service	On the one hand, very friendly and welcoming s...	Date of stay: February 2023
1	NicolasM807	Charming Hotel in a Perfect Location	This was our first trip to London, so we wante...	Date of stay: August 2022
2	LouisaBean	Fantastic location, friendly staff, comfortabl...	Great location, room was so comfortable and cl...	Date of stay: January 2023
3	Eastcoastwil	Wonderful Location forTheatre	We returned to The Resident Covent Garden beca...	Date of stay: February 2023
4	yenalubu	Great memories in London	So happy to stay here for my first solo travel...	Date of stay: February 2023

In [43]:

1 df3.shape

Out[43]:

(1350, 4)

19/2/23, 22:11

Web Scrapping TripAdvisor - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [44]:

```
1 df3.head(15)
```

Out [44]:

	Nombre	Titulo	Review	Data
0	nimrodk803	Small room, no service	On the one hand, very friendly and welcoming s...	Date of stay: February 2023
1	NicolasM807	Charming Hotel in a Perfect Location	This was our first trip to London, so we wante...	Date of stay: August 2022
2	LouisaBean	Fantastic location, friendly staff, comfortabl...	Great location, room was so comfortable and cl...	Date of stay: January 2023
3	Eastcoastwil	Wonderful Location forTheatre	We returned to The Resident Covent Garden beca...	Date of stay: February 2023
4	yenalubu	Great memories in London	So happy to stay here for my first solo travel...	Date of stay: February 2023
5	Boss-man56	Excellent	Just spent 3 days at The Resident Covent Garde...	Date of stay: February 2023
6	Discover647381	Great Place	Everything was perfect everything you could wa...	Date of stay: January 2023
7	gufton	Customer Service Excellence	The best hotel I've stayed at in London. The c...	Date of stay: February 2023
8	meltemtopuz	Perfect	The best hotel i have ever stayed in London. T...	Date of stay: January 2023
9	FrancesParis	Reliable comfort and convenience	Very comfortable and well equipped room in a h...	Date of stay: January 2023
10	Ella W	Wonderful Hotel, Book it!	I've read soo many reviews before selecting th...	Date of stay: January 2023
11	c11ffc	Birthday celebration	We needed somewhere to stay in Covent Garden a...	Date of stay: January 2023
12	Gabriel R	Simple but excellent hotel in the heart of Lon...	If you don't need unnecessary luxury but love ...	Date of stay: January 2023
13	vaksdenise	Great hotel in Covent Garden	Great room and bathroom, very attentive staff...	Date of stay: January 2023
14	Kevin	Impeccable Service	We stayed here in a King room for 2 nights in ...	Date of stay: January 2023

In []:

```
1 df_final = [df,df2,df3]
```

In []:

```
1 df_final = pd.concat([df,df2])
```

In []:

```
1 len(df_final)
```

In []:

```
1 df_final = df_final.reset_index(drop=True)
```

In []:

```
1 df_final.to_csv("datos_final_TFM.csv", index=False, sep=',')
2 files.download('datos_final_TFM.csv')
```

8.3 Anexo III – Script del análisis

17/2/23, 20:53

UCM - TFM - Gerard Chicot Navalls - Jupyter Notebook

TFM - NLP - Gerard Chicot Navalls

In [1]:

```
1 import warnings  
2  
3 warnings.filterwarnings("ignore")
```

1. Imports

In [2]:

```

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

Requirement already satisfied: nltk in c:\users\xikix\anaconda3.2\lib\site-packages (3.7)
Requirement already satisfied: click in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk) (8.0.4)
Requirement already satisfied: tqdm in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk) (4.64.1)
Requirement already satisfied: joblib in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk) (1.1.0)
Requirement already satisfied: regex==2021.8.3 in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk) (2022.7.9)
Requirement already satisfied: colorama in c:\users\xikix\anaconda3.2\lib\site-packages (from click->nltk) (0.4.6)
Requirement already satisfied: contractions in c:\users\xikix\anaconda3.2\lib\site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\xikix\anaconda3.2\lib\site-packages (from contractions) (0.0.24)
Requirement already satisfied: pyahocorasick in c:\users\xikix\anaconda3.2\lib\site-packages (from textsearch>=0.0.21->contractions) (2.0.1)
Requirement already satisfied: anyascii in c:\users\xikix\anaconda3.2\lib\site-packages (from textsearch>=0.0.21->contractions) (0.3.1)
Requirement already satisfied: TextBlob in c:\users\xikix\anaconda3.2\lib\site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in c:\users\xikix\anaconda3.2\lib\site-packages (from TextBlob) (3.7)
Requirement already satisfied: tqdm in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk>=3.1->TextBlob) (4.64.1)
Requirement already satisfied: regex>=2021.8.3 in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk>=3.1->TextBlob) (2022.7.9)
Requirement already satisfied: joblib in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk>=3.1->TextBlob) (1.1.0)
Requirement already satisfied: click in c:\users\xikix\anaconda3.2\lib\site-packages (from nltk>=3.1->TextBlob) (8.0.4)
Requirement already satisfied: colorama in c:\users\xikix\anaconda3.2\lib\site-packages (from click->nltk>=3.1->TextBlob) (0.4.6)

```

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_contraccion(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

```

Requirement already satisfied: contractions in c:\users\xikix\anaconda3.2\lib\site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in c:\users\xikix\anaconda3.2\lib\site-packages (from contractions) (0.0.24)
Requirement already satisfied: pyahocorasick in c:\users\xikix\anaconda3.2\lib\site-packages (from textsearch>=0.0.21->contractions) (2.0.1)
Requirement already satisfied: anyascii in c:\users\xikix\anaconda3.2\lib\site-packages (from textsearch>=0.0.21->contractions) (0.3.1)

3. Obtención del corpus

In [4]:

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

In [5]:

1 datos

Out[5]:

	Hotel	Nombre	Titulo	Review	Date of stay
0	Hampton_by_Hilton_London_Waterloo-London_England	cresil	Good location	It is very near the waterloo train station. Br..	Date of stay
1	Hampton_by_Hilton_London_Waterloo-London_England	cactusstud	City break	We booked here as for us it was ideally locate...	Date of stay
2	Hampton_by_Hilton_London_Waterloo-London_England	Fredbear_Hull	great location	great location, great breakfast. clean. good s...	Date of stay
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...	Date of stay
4	Hampton_by_Hilton_London_Waterloo-London_England	Honest Reviewer	Would stay again	I booked this hotel with trepidation mainly be...	Date of stay
...
12789	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	JamesByers78	Holiday in London	An excellent hotel for the price! Great contin...	Date of stay
12790	Holiday_Inn_Express_London_Earl_s_Court_an_IHG...	Rowena D	Weekend away	Visited with a friend on Royal Wedding/footbal...	Date of stay
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, considerin...	Date of stay
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...	Date of stay
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...	Date of stay

12794 rows x 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 de 12794 reseñas.

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 todos 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 greata 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_Berkley-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_Apartehotels_by_StacyCity_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'
'Ihabit_Queen_s_Gardens-London_England'
'Wilde_Apartehotels_by_StacyCity_London_Aldgate_Tower_Bridge-London_England'
'The_Westminster_London_Curio_Collection_by_Hilton-London_England'
'The_Piccadilly_London_West_End-London_England'
'Draycott_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_Akatsuki_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 Reviewers tengan el mismo nombre.

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 consigue 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.

17/2/23, 20:53

UCM - TFM - Gerard Chicot Navalls - Jupyter Notebook

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

17/2/23, 20:53

UCM - TFM - Gerard Chicot Navalls - Jupyter Notebook

In [23]:

```

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

```

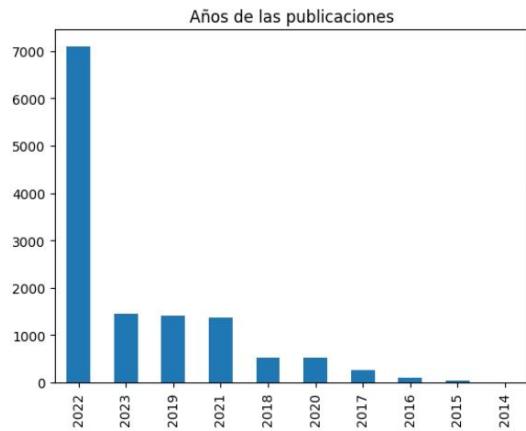
¿Cómo se comportan las variables del tiempo?

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 val-

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	Título	Review	Año	Mes	norme
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, h...
2	Hampton by Hilton London Waterloo	Fredbear_Hull	great location	great location, great breakfast. clean. good s...	2023	1	[great, location, great, breakfast, cl...
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, gi...
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, ai...

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.

Quiero saber en qué tipo de sentimiento aparecen los unigramas, bigrams y trigrams del estudio general. De esta forma podré afirmar si las palabras que salen con más frecuencia son neutras o negativas.

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)

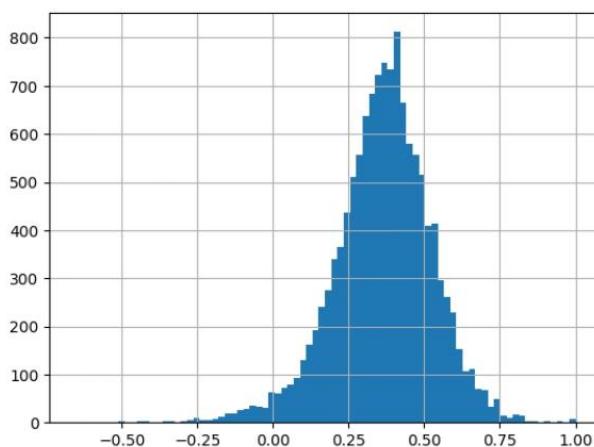
0% | 0/12769 [00:00<?, ?it/s]
0% | 0/12769 [00:00<?, ?it/s]
```

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 0 y 1.0. Particularmente he añadido una sección de neutro para poder dividir mucho mejor el análisis y poder segmentar más.

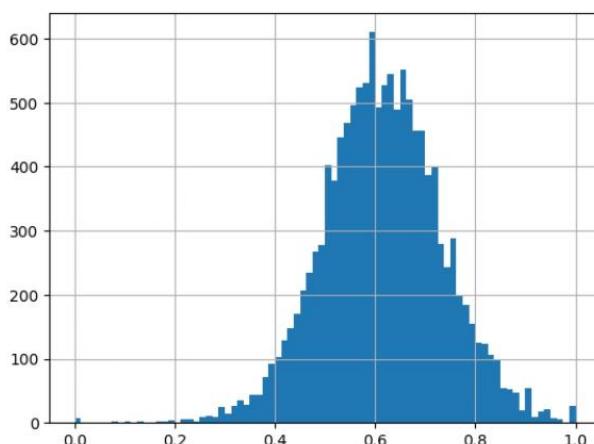
Con lo cual podemos concluir que las reviews tienden a ser neutras-positivas, con un rango de entre 0.15 a 0.60.

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	Título	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]:

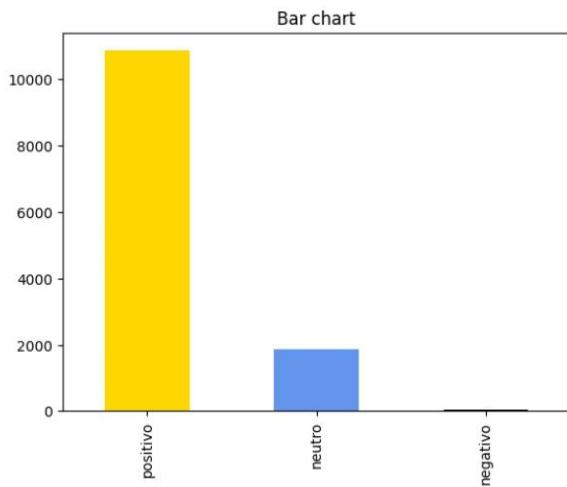
```
1 datos['tipo_sentimiento'].value_counts()
```

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 tienen poca representación.

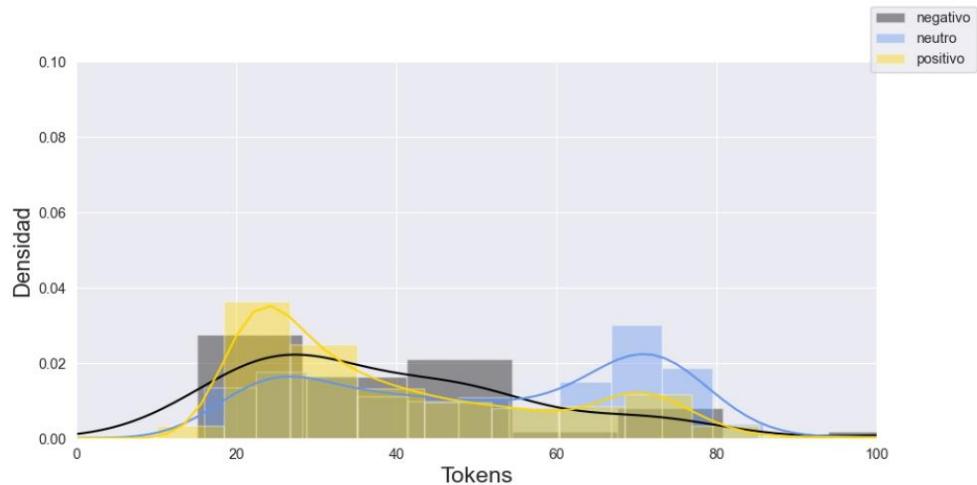
Distribución de longitud del texto

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 negativos, vemos que tienen una representación más amplia, con un pico entre 15 y 30 palabras y otro entre 40 y 50. Si hablamos de neutros, vemos que tienen una representación más amplia, con un pico entre 15 y 30 palabras y otro entre 40 y 50. Si hablamos de positivos, vemos que tienen una representación más amplia, con un pico entre 15 y 30 palabras y otro entre 40 y 50.

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'

```

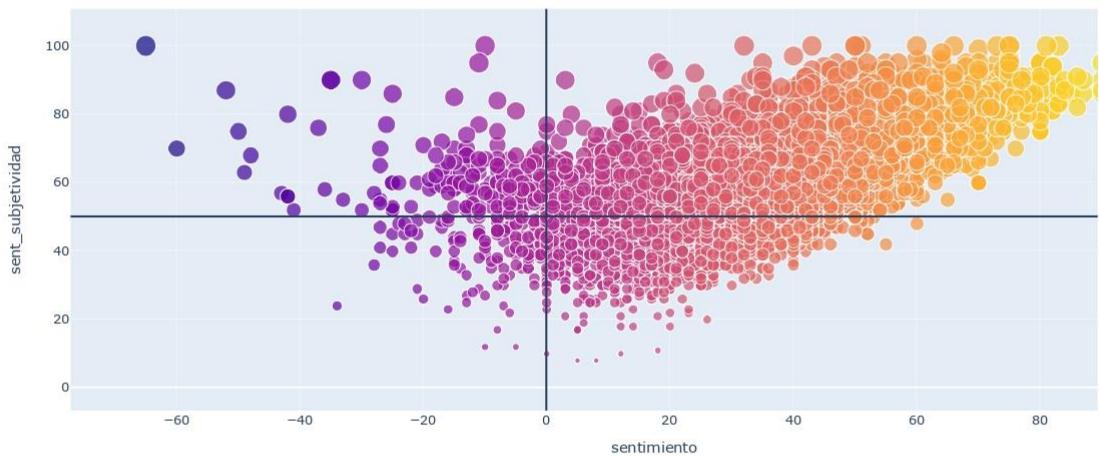
In [41]:

```

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=[dict(
11                           type= 'line',
12                           yref= 'paper', y0= 0, y1= 1,
13                           xref= 'x', x0= 0, x1= 0),
14                           dict(
15                               type='line',
16                               xref='paper', x0=0, x1=1,
17                               yref='y', y0=50, y1=50,
18                           )])
19
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, hay muy pocas negativas. 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 revolución.
- 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 específico para las tres categorías segmentadas.

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
```

In [49]:

```
1 %time  
2 dict_freq = Counter(lista_tokens)
```

Wall time: 70.8 ms

In [50]:

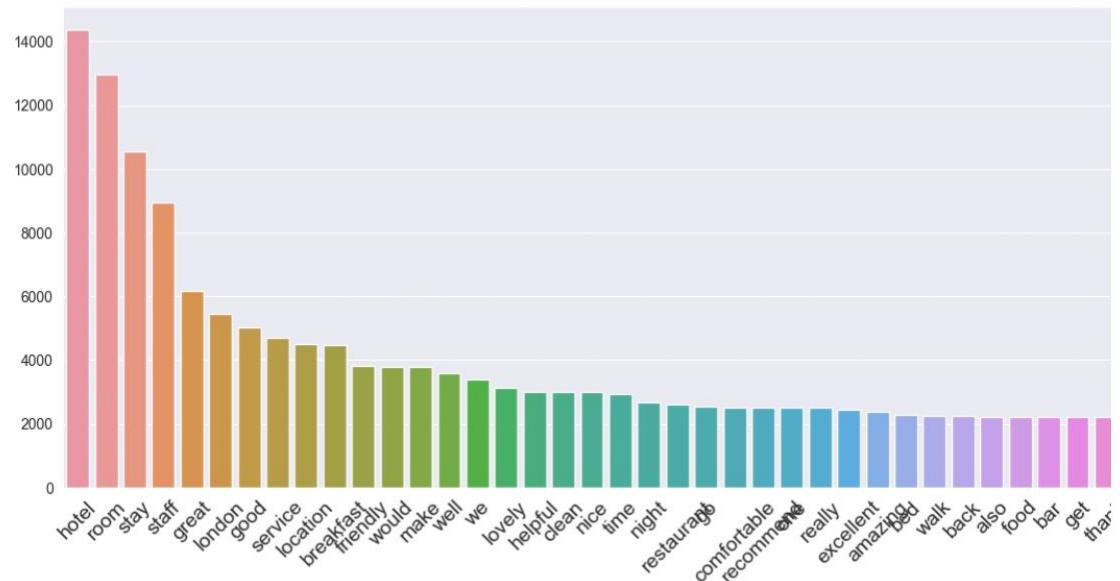
```
1 # Ordenamos el diccionario por la frecuencia de sus palabras
2 dict_freq_order = sorted(dict_freq.items(), key=lambda x: x[1], reverse=True)
3 token_names = list()
4 token_freqs = list()
5 for i in dict_freq_order:
6     if i[1] > 2000:
7         token_names.append(i[0])
8         token_freqs.append(i[1])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

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 dándonos pistas sobre los temas principales. 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...

* Trigrama

In [58]:

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

In [59]:

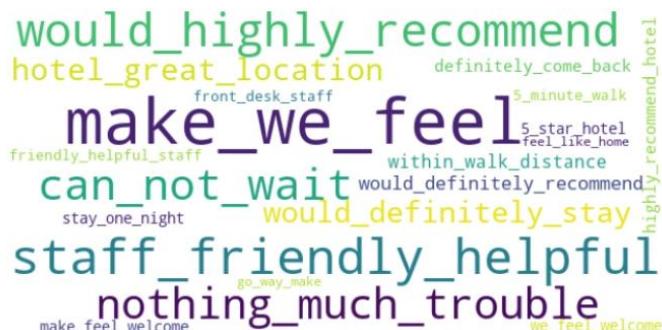
```
1 trigram_list = datos["trigrama"].tolist()
2 review_str2 = ' '.join(trigram_list)
```

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



Encontramos como los trigramas 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 qué valora más el target.

- Aparición de comentarios en el tiempo

Voy a mostrar 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 distribución.

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]:

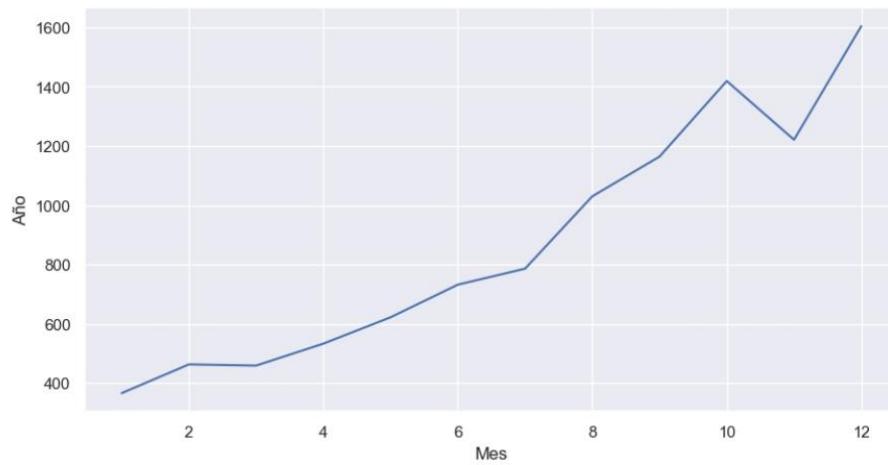
```
1 años = [2019, 2020, 2021, 2022]
2 df_19_22 = datos[datos['Año'].isin(años)]
```

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 1600.

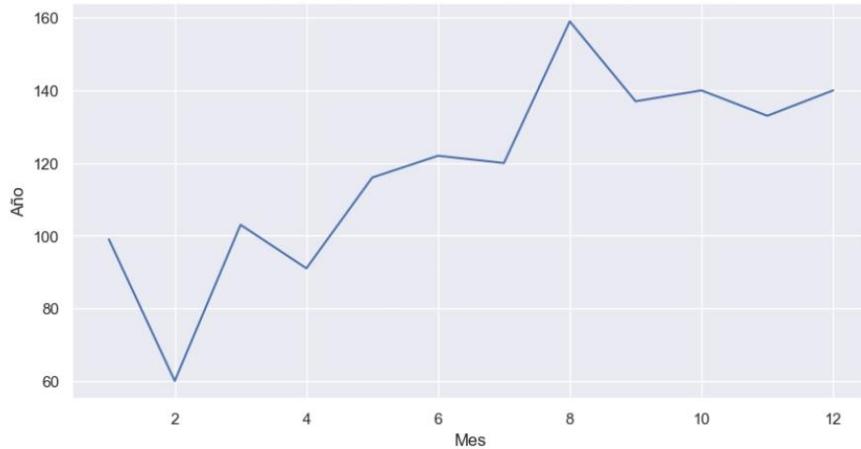
Esta gráfica no se explica muy bien ya que habla sobre la suma de las valoraciones de cuatro años seguidos observado según cada mes. Nos encontramos en un evento r a os, el confinamiento de la Covid-19. Estoy seguro que en las gráficas posteriores vamos a observar meses enteros sin ninguna valoración.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

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.

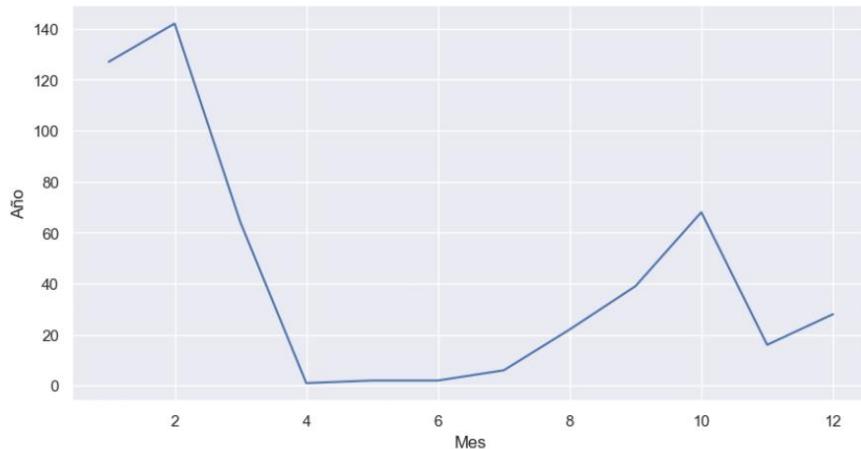
Podemos apreciar una correlación entre la demanda y el número de comentarios, este hecho lo voy a confirmar a medida que vaya viendo el comportamiento de las otras cosas.

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 enero 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 año anterior.

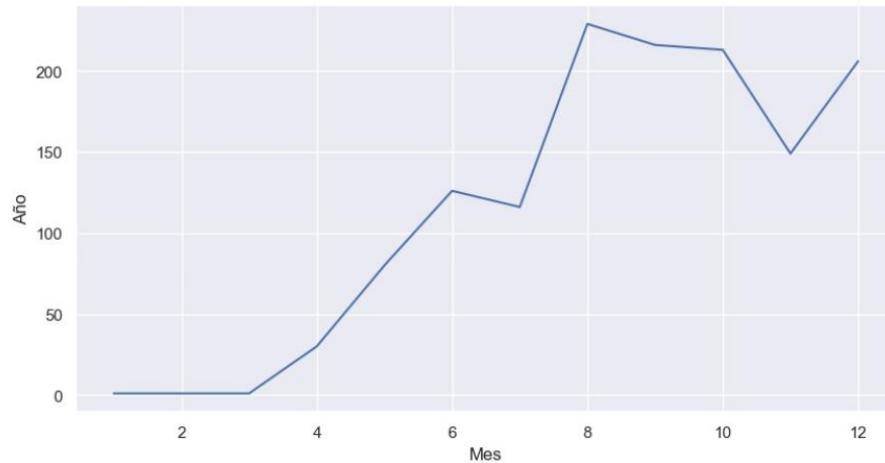
Otra vez podemos concluir que la correlación entre ocupación y número de comentarios tiene mucha lógica.

In [68]:

```

1 comentarios_año = df_21.groupby(['Mes']).count()
2 plt.figure(figsize=(10, 5))
3 sb.set()
4 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 contra el sector.

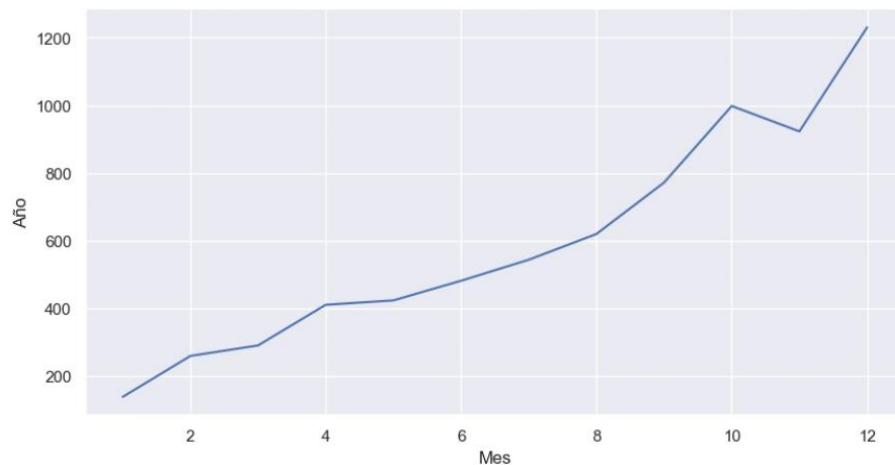
Una vez solucionado el rebrote las valoraciones aumentaron y vemos un gran pico positivo en el mes de agosto. El número de comentarios del mes de agosto del año 2021 supone un aumento de la ocupación por culpa de las medidas contra la Covid-19.

In [69]:

```

1 comentarios_año = df_22.groupby(['Mes']).count()
2 plt.figure(figsize=(10, 5))
3 sb.set()
4 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 rr 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)
```

In [72]:

```
1 %time  
2 dict_freq = Counter(lista_tokens)
```

Wall time: 48.9 ms

In [73]:

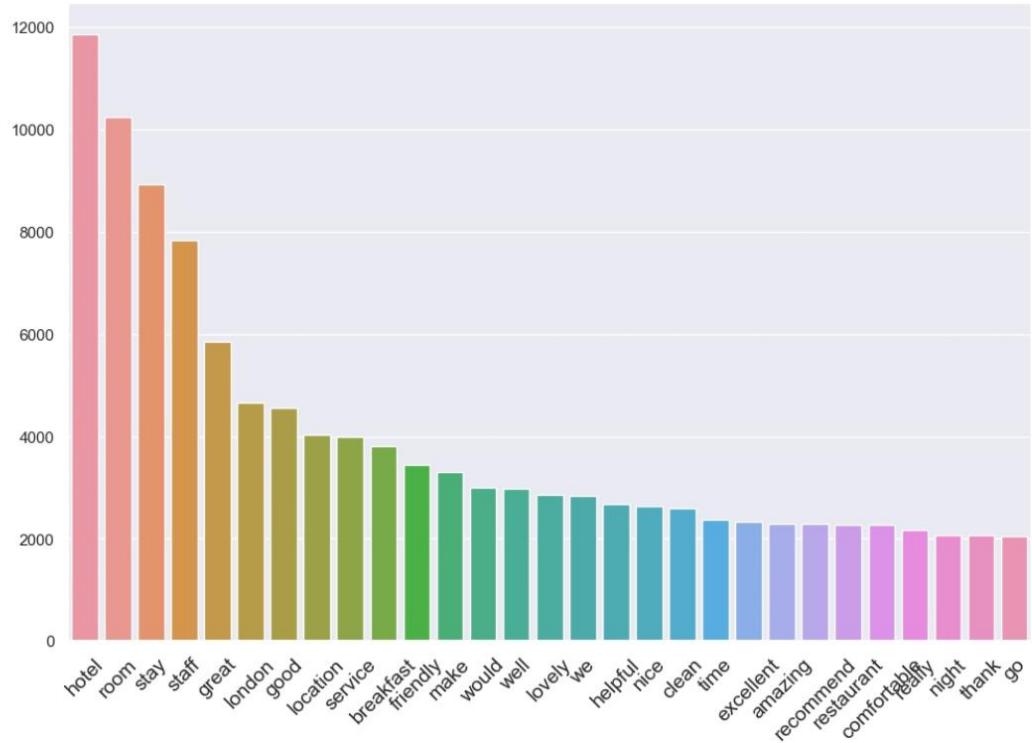
```
1 # Ordenamos el diccionario por la frecuencia de sus palabras
2 dict_freq_order = sorted(dict_freq.items(), key=lambda x: x[1], reverse=True)
3 token_names = list()
4 token_freqs = list()
5 for i in dict_freq_order:
6     if i[1] > 2000:
7         token_names.append(i[0])
8         token_freqs.append(i[1])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

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_wordcloud(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_wordcloud(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", "I 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_documento = 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_documento)
```

In [85]:

```
1 %%time  
2 dict_freq = Counter(lista_tokens)
```

Wall time: 996 µs

In [86]:

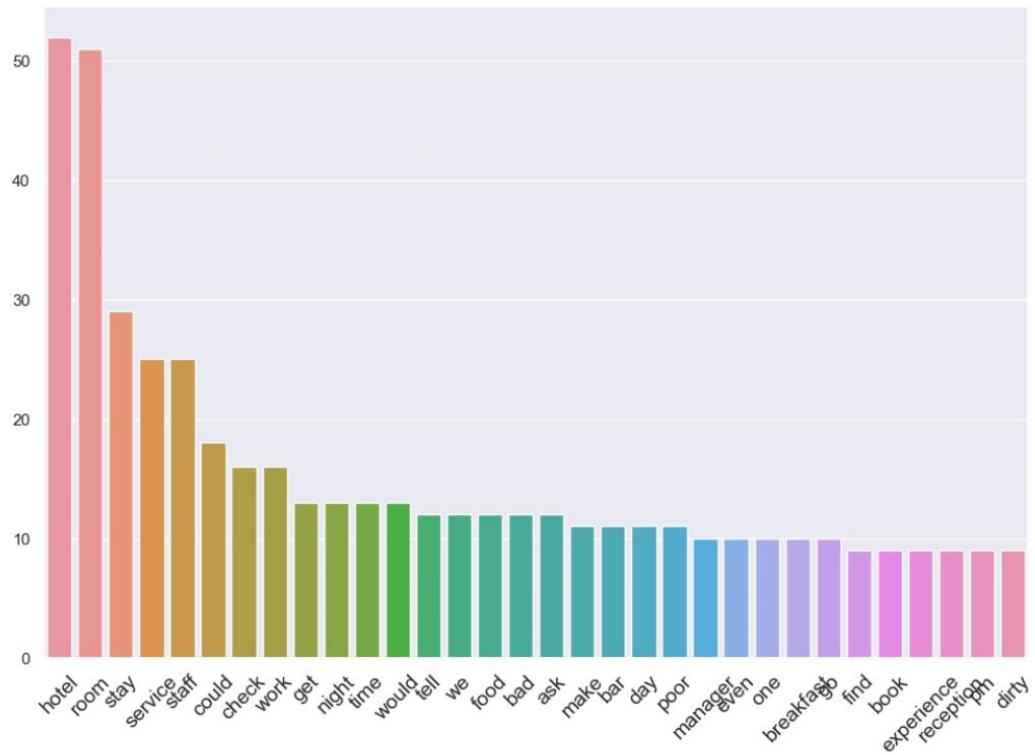
```
1 # Ordenamos el diccionario por la frecuencia de sus palabras
2 dict_freq_order = sorted(dict_freq.items(), key=lambda x: x[1], reverse=True)
3 token_names = list()
4 token_freqs = list()
5 for i in dict_freq_order:
6     if i[1] > 8:
7         token_names.append(i[0])
8         token_freqs.append(i[1])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

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 si 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.

* Trígrama

In [92]:

```
1 datos_negativo["trígrama"] = 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["trígrama"].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 trígramas nos encontramos elementos muy interesantes como por ejemplo five_star_hotel. Me hace pensar que las valoraciones relacionadas con este trígrama expresa una buena experiencia en 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.

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
rtment..recommend 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 spac
awa 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
ing 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 hotel. The rooms have a lot of
ghts 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
uff 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 th
otel. 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_pc
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',                   # Learning method
5                                       random_state=100,                            # Random state
6                                       batch_size=128,                            # n docs in each learning iter
7                                       evaluate_every = -1,                      # compute perplexity every n iters, default: Don't
8                                       n_jobs = -1,                                # Use all available CPUs
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: -2626299.1102484697
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_idf': 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 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 porque

- 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
1	0
2	1
3	3

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]:

```

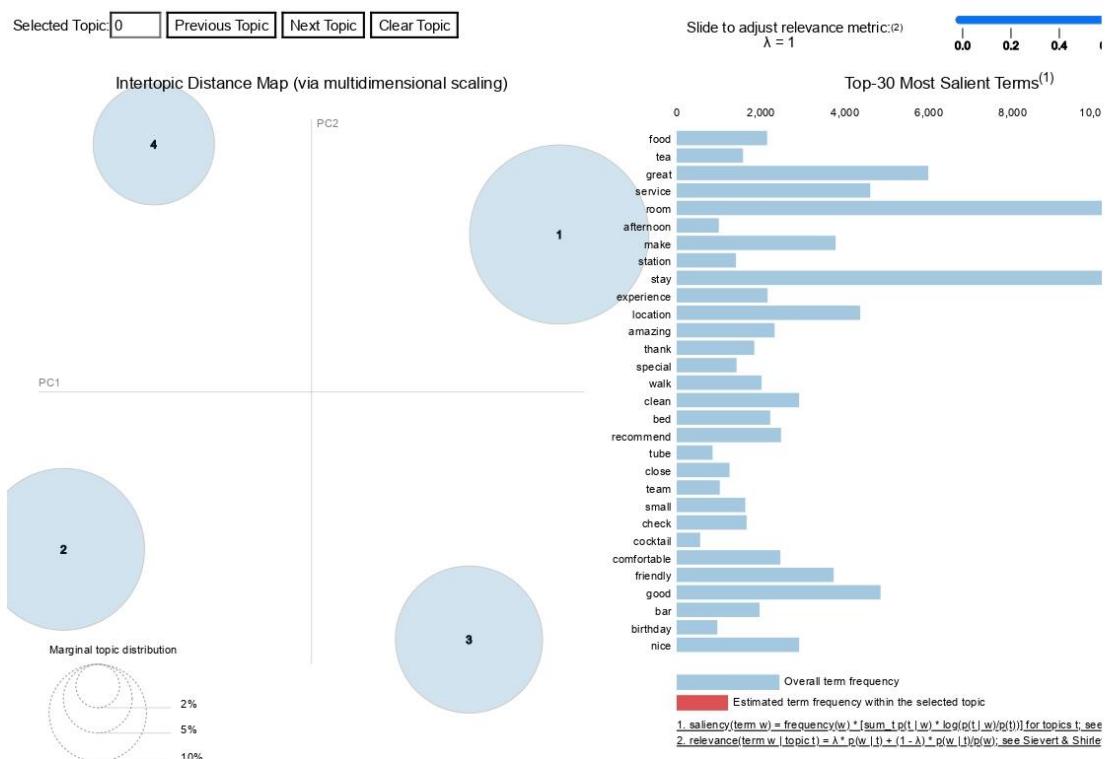
1 import pyLDAvis
2 import pyLDAvis.sklearn
3 import pyLDAvis.gensim_models
4
5 pyLDAvis.enable_notebook()
6 panel = pyLDAvis.sklearn.prepare(best_lda_model, data_vectorized, vectorizer, mds='tsne')
7 panel

```

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 tema. El Topical Terms que nos muestra todas las palabras que tiene cada tema y su frecuencia de aparición en el tema 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 un indicador de que el modelo le cuesta diferenciar entre los temas, 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 tema ha salido en más reviews.

Apreciamos cuatro grandes grupos, formados 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: "hotel", "room", "stay", "staff" y "great" con un gran porcentaje de aparición de cada palabra. En el grupo 2, que contiene el 30.4% de los tokens, tiene una distribución muy similar con 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 de "room". Contiene palabras como "room", "hotel", "check", "stay" y "breakfast". Aquí ya observamos como predominan otro tipo de palabras. En el tema 4, con el 18.5% de los tokens, seguido de palabras como "room", "stay", "service", "time"...

- En el tema 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 la revisión.
- En el tema 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 tema 2 en comentarios de revisión.
- En el tema 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 tema 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	7461 1_room_hotel_stay_staff
1	0	1930 0_great_staff_service_good
2	1	382 1_tea_afternoon_sandwich_scone
3	2	320 2_london_good_location_stay
4	3	200 3_birthday_celebrate_anniversary_special
...
65	64	11 64_apartment_130_gate_kensington
66	65	11 65_bed_comfy_comfortable_super
67	66	10 66_como_metropolitan_group_sutedja
68	67	10 67_quirky_suite_bolingbroke_batty
69	68	10 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	-1_room_hotel_stay_staff
1	0	0_hotel_staff_location_great
2	1	1_london_good_location_great
3	2	2_tea_afternoon_sandwich_cake
4	3	3_birthday_savoy_special_celebrate
5	4	4_food_service_lunch_delicious
6	5	5_paddington_station_walk_hyde
7	6	6_covent_garden_trafalgar_square
8	7	7_tell_card_call_pay
9	8	8_marylebone_dorset_hotel_square
10	9	9_daughter_child_old_family
11	10	10_bed_pillow_comfortable_comfy
12	11	11_hyde_park_kensington_apartment
13	12	12_london_hotel_stay_experience
14	13	13_sloane_square_hotel_draycott
15	14	14_victoria_pimlico_station_breakfast
16	15	15_clermont_victoria_cross_char
17	16	16_cocktail_bar_drink_menu
18	17	17_shower_room_bath_bathroom
19	18	18_firmdale_knightsbridge_hotel_love
20	19	19_dorchester_promenade_afternoon_tea
21	20	20_buckingham_palace_westminster_park
22	21	21_beaumont_mayfair_stay_hotel
23	22	22_wife_husband_london_we
24	23	23_noise_noisy_sound_floor
25	24	24_hotel_room_beautiful_bath
26	25	25_berkeley_hotel_service_always
27	26	26_ritz_rivoli_experience_live
28	27	27_massage_spa_treatment_therapist
29	28	28_prince_akatoki_japanese_arch
30	29	29_waterloo_station_train_breakfast
31	30	30_rosewood_holborn_scarfes_london
32	31	31_henrietta_covent_garden_hotel
33	32	32_zetter_townhouse_clerkenwell_cocktail
34	33	33_room_smell_issue_night
35	34	34_langham_palm_alesian_court
36	35	35_mayfair_flemings_fleming_lorena
37	36	36_shower_sized_room_good
38	37	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 o 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', 0.02007983602441976),  
 ('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', 0.0297753526314355),  
 ('location', 0.0274399664495423),  
 ('great', 0.02673017652565895),  
 ('stay', 0.024178295660578115),  
 ('hotel', 0.023576064839623486),  
 ('recommend', 0.021163341632456457),  
 ('staff', 0.02018799216626245),  
 ('place', 0.01882549351271179),  
 ('would', 0.016759320785621274)]
```

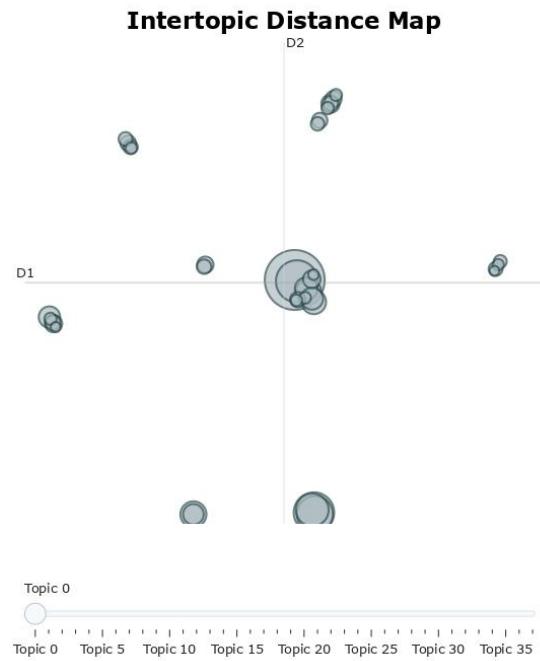
En el segundo tópico observamos tokens como "London", "good", "location", "great"... Esto hace pensar que este tópico se encontrará en las reviews que hablan de forma generalizada con la ubicación.

- 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 es común.

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" habla de "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]:

```

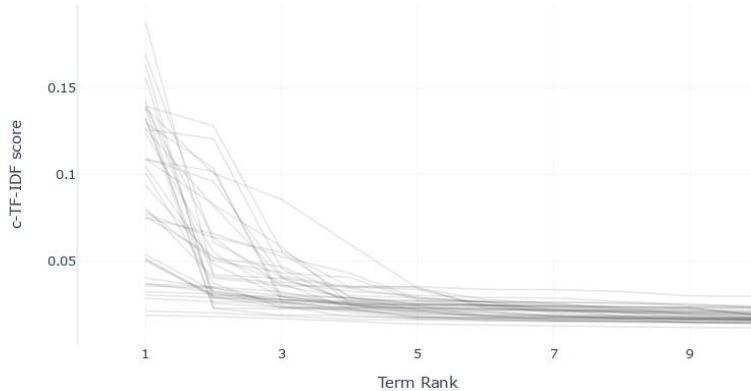
1 # Visualize term rank decrease
2 topic_model.visualize_term_rank()

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 peso. 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 peso disminuye.

- 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 observar 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. Es lo normal en los tópicos 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 tema 9 hablando de la familia.

- Visualización de las agrupaciones de los tópicos

In [129]:

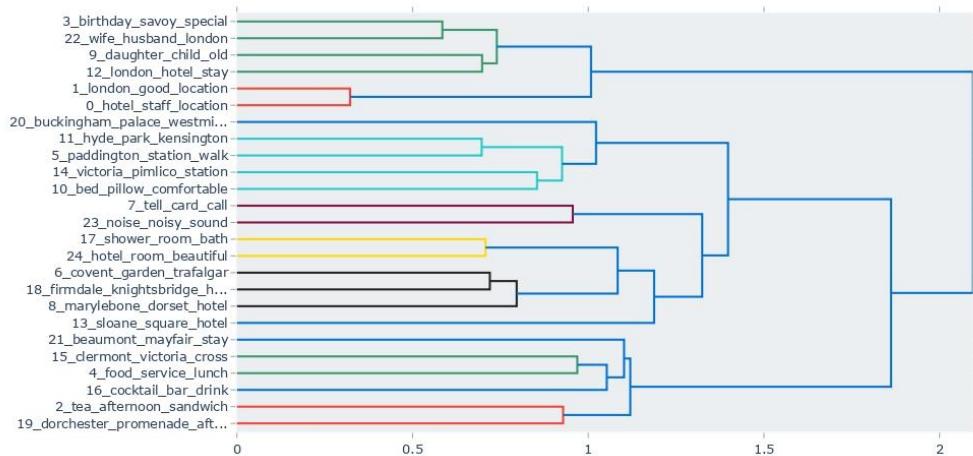
```

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\figure_factory\_dendrogram.py:354: 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)

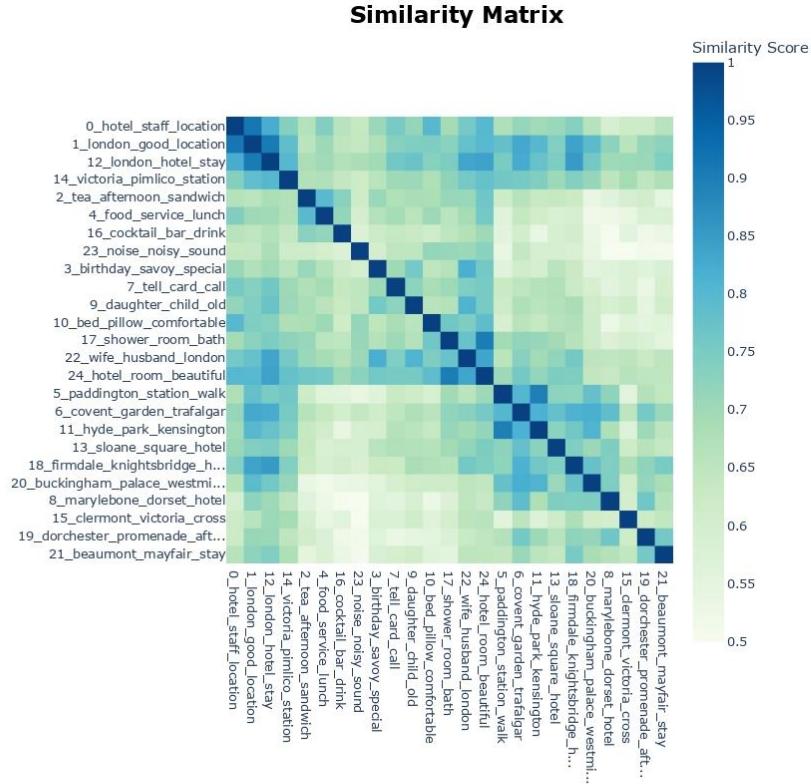
```

```

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.

```



En este complot 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 al otro.

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 positiva. Si nos fijamos en esta gráfica, podemos ver que el tópico 0 y 1 tienen una correlación negativa. Pueden haber errores en la ejecución del código o en la interpretación de los resultados.

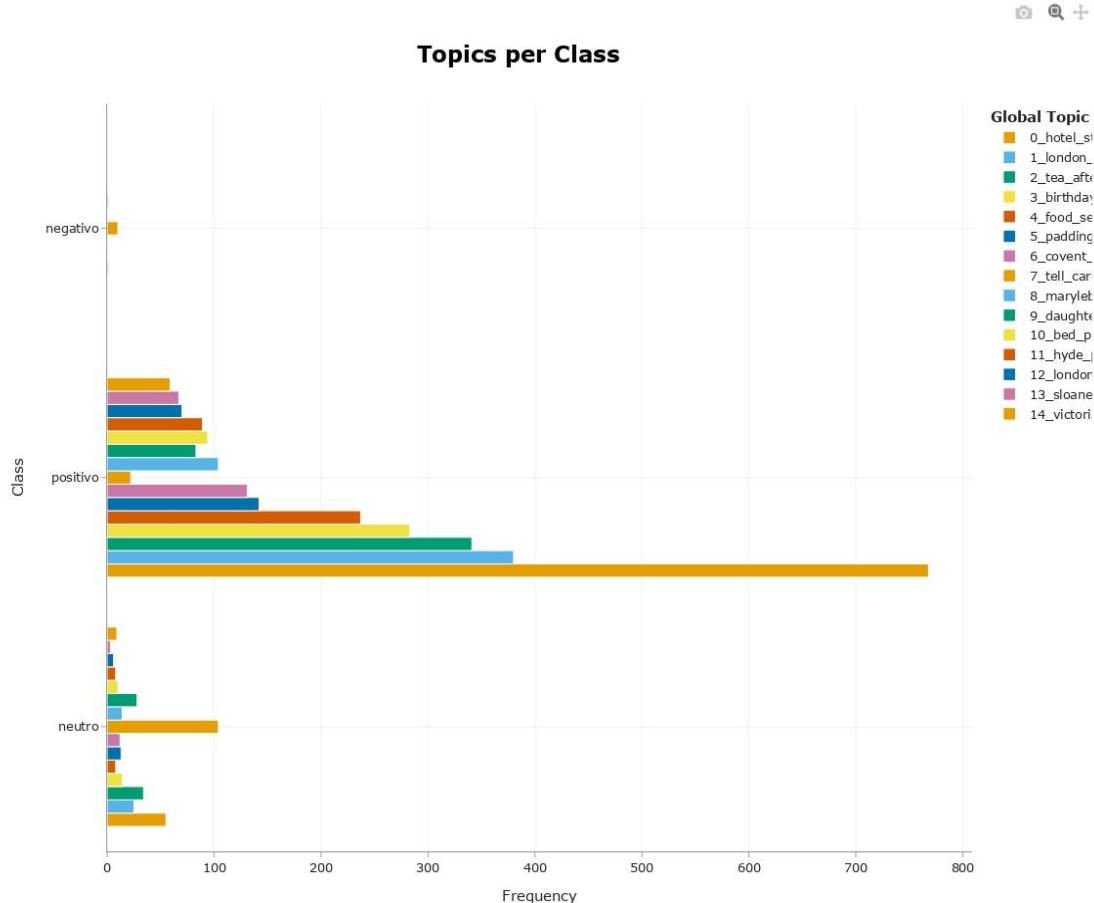
- Tópicos por tipo de clase según el tipo de sentimiento

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.

```



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]:

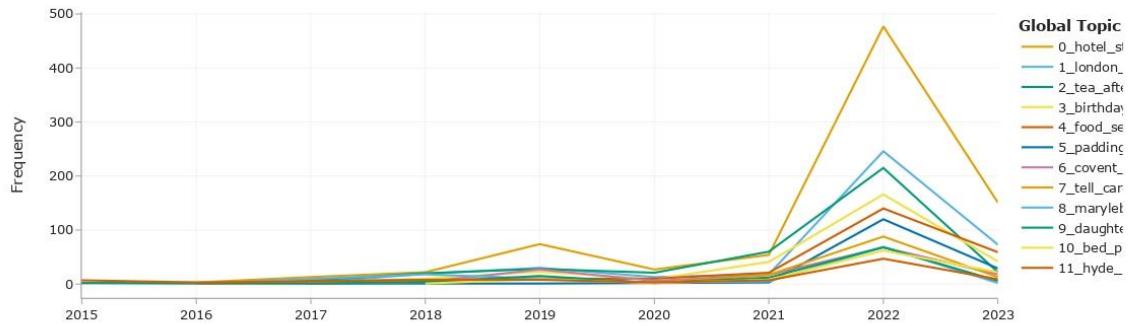
```

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

- Ejemplo para visualizar las probabilidades de un tema

```
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 owe 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,
       7.250866818e-308, 5.37560770e-308])
```

In [137]:

```

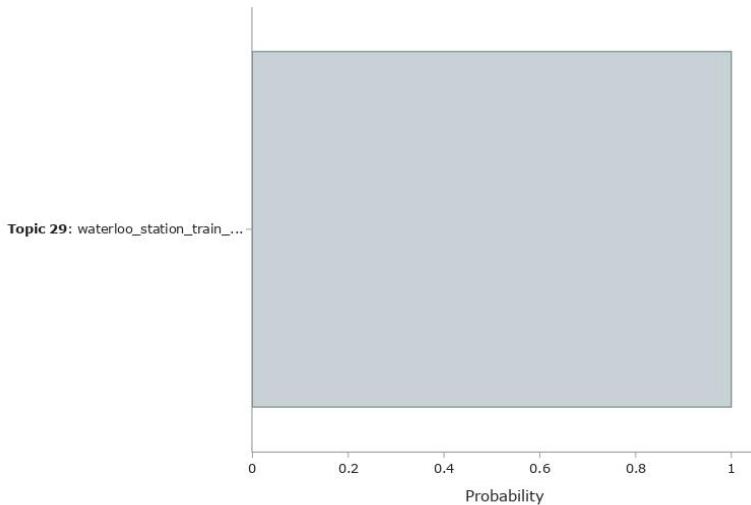
1 # Visualize probability distribution
2 topic_model.visualize_distribution(topic_model.probabilities_[0], min_probability=0.015)

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. Ey. 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 rey really great walk shower complimentary breakfast adequate shock horror fry egg end world please people moan least important issue difficuly'
```

In [140]:

```

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

```

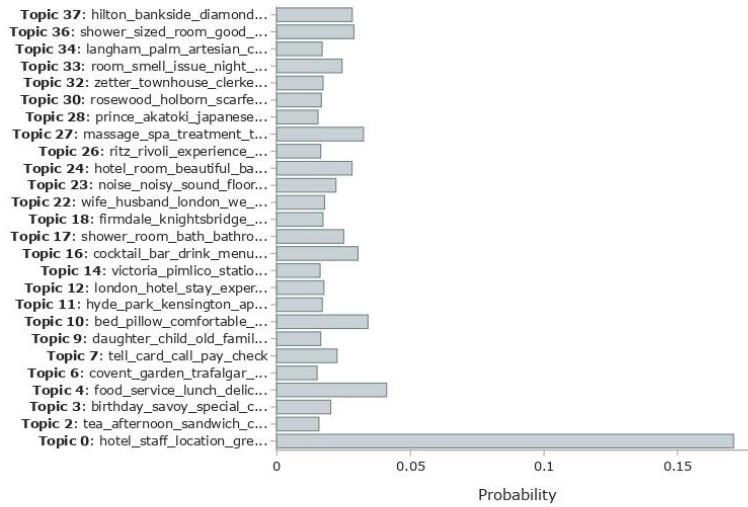
Out[140]:

```
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 que el tema 0 es el más probable.

- 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	Título	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...	near waterloo train station breakfast good roo...	70	19	neutro	50
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...	book we ideally locate trip hotel exceptionall...	67	32	positivo	53
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...	great location great breakfast clean good staf...	62	42	positivo	25
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...	stay london always choose great location min w...	61	55	positivo	23

9. Bibliografía

Topic modeling guide (GSDM, LDA, LSI) | Kaggle

<https://www.kaggle.com/code/ptfrwrd/topic-modeling-guide-gsdm-lda-lsi#LSI-model>

Topic modeling with LSA, PSLA, LDA & lda2Vec | NanoNets

<https://medium.com/nanonets/topic-modeling-with-lsa-psla-lda-and-lda2vec-555ff65b0b05>

Sentiment Analysis using Transfer Learning #1 | Kaggle

<https://www.kaggle.com/code/tamoghna96saha/sentiment-analysis-using-transfer-learning-1>

Sentiment Analysis on TripAdvisor Hotel Reviews with Python and NLP | by Dennis Niggli | Python in Plain English

<https://python.plainenglish.io/sentiment-analysis-on-tripadvisor-hotel-reviews-with-python-and-nlp-68b3555d816c>

Topic modeling with LSA, pLSA, LDA, NMF, BERTopic, Top2Vec: a Comparison | by Nicolo Cosimo Albanese | Towards Data Science

<https://towardsdatascience.com/topic-modeling-with-lsa-plsa-lda-nmf-bertopic-top2vec-a-comparison-5e6ce4b1e4a5>

Topic Visualization - BERTopic

https://maartengr.github.io/BERTopic/getting_started/visualization/visualization.html#visualize-topics

Topic modeling visualization - How to present results of LDA model? | ML+

<https://www.machinelearningplus.com/nlp/topic-modeling-visualization-how-to-present-results-lda-models/>

BERTopic.ipynb - Colaboratory

<https://colab.research.google.com/drive/1FieRA9fLdkQEGDIMYl0I3MCjSUKVF8C-?usp=sharing#scrollTo=UbT2Bd9gqaJ3>

Topic modeling: An Introduction

<https://monkeylearn.com/blog/introduction-to-topic-modeling/>

BERTopic - BERTopic

<https://maartengr.github.io/BERTopic/api/bertopic.html>

Parameter Selection for HDBSCAN* — hdbscan 0.8.1 documentation

https://hdbscan.readthedocs.io/en/latest/parameter_selection.html

Topic modeling with Deep Learning Using Python BERTopic | by Amy @GrabNGoInfo | GrabNGoInfo | Medium

<https://medium.com/grabngoinfo/topic-modeling-with-deep-learning-using-python-bertopic-cf91f5676504>

Hyperparameter Tuning for BERTopic Model in Python | by Amy @GrabNGoInfo | GrabNGoInfo | Medium

<https://medium.com/grabngoinfo/hyperparameter-tuning-for-bertopic-model-in-python-104445778347>

HDBScan clustering and BERT for Topic modeling | by NaturalTech | Medium

<https://naturaltech.medium.com/hdbscan-clustering-and-bert-for-topic-modeling-453ef4184cc9>

Hyperparameter Tuning for BERTopic Model in Python - Grab N Go Info

<https://grabngoinfo.com/hyperparameter-tuning-for-bertopic-model-in-python/>

LDA - How to grid search best topic models? (With examples in python)

<https://www.machinelearningplus.com/nlp/topic-modeling-python-sklearn-examples/>

A Beginner's Guide to Latent Dirichlet Allocation (LDA) | by Ria Kulshrestha | Towards Data Science

<https://towardsdatascience.com/latent-dirichlet-allocation-lda-9d1cd064ffa2>

Latent Dirichlet Allocation - Wikipedia, la enciclopedia libre

https://es.wikipedia.org/wiki/Latent_Dirichlet_Allocation

BERTopic: Neural topic modeling with a class-based TF-IDF procedure

<http://arxiv.org/abs/2203.05794>

Web scraping Laia Subirats Maté Mireia Calvo González

Gascó Sánchez, Luis

Text Mining Curso 2022 Estos apuntes se distribuyen bajo una licencia Creative Commons de Reconocimiento-NoComercial-SinObraDerivada