COMILLAS
UNIVERSIDAD PONTIFICIA

Facultad de Ciencias Económicas y Empresariales

GUÍA PRÁCTICA DE IMPLEMENTACIÓN DE TOPIC MODELING EN R: ANALIZANDO ARTÍCULOS PERIODÍSTICOS SOBRE EL METAVERSO

Autor: María del Rocío Velilla Cadahía
Director: Lucía Barcos Redín

MADRID | Junio 2023

Resumen

El topic modeling es una técnica que, mediante algoritmos como el *Latent Dirichlet Allocation (LDA)*, ayuda a entender cuáles son los temas principales en una colección de documentos de texto. A pesar de su creciente uso en el ámbito de la investigación y de la industria, la literatura existente no cuenta con una guía práctica de implementación en R del LDA que profundice en los aspectos técnicos del algoritmo. Por tanto, el presente trabajo, tras una explicación profunda del topic modeling y del algoritmo de LDA, desarrolla una guía práctica que aborda todo el proceso de topic modeling y explica cuestiones teóricas que surgen al implementarlo. La guía se construye a través de un caso de análisis sobre el metaverso en los artículos del periódico *The Guardian* y se desarrolla en cinco fases principales. Tras las tres primeras fases de recolección de datos, preprocesamiento del texto y aplicación del LDA, se interpretarán y visualizarán en la cuarta fase los catorce tópicos obtenidos. Finalmente, en la quinta y última fase, se propondrán posibles usos de los tópicos que acentuarán el potencial del topic modeling. El trabajo finalizará con las principales conclusiones y una propuesta de tres posibles extensiones del estudio realizado.

Palabras clave: topic modeling, Latent Dirichlet Allocation (LDA), guía práctica, R, metaverso

Abstract

Topic modeling is a technique that, through algorithms such as Latent Dirichlet Allocation (LDA), helps understand what the most relevant topics in a collection of text documents are. Despite its increasing use in the field of academic research and in the industry, the existing literature does not include a practical guide of LDA implementation in R that delves into the technical aspects of this algorithm. As a result of this, after a profound explanation of topic modeling and of the LDA algorithm, a practical guide that addresses the entire topic modeling process and the theoretical questions that arise during its implementation is developed. The practical guide is built through a case study on the metaverse in news articles from the newspaper *The Guardian* and is constructed in five different phases. After the first three phases of data collection, preprocessing of text, and application of LDA, fourteen topics will be interpreted and visualized in the fourth phase. Finally, in the fifth and final phase, possible uses of the retrieved topics will be suggested and will accentuate the potential of topic modeling. The thesis will finish with the main conclusions and a proposal of three possible further extensions of the work conducted.

Keywords: topic modeling, Latent Dirichlet Allocation (LDA), practical guide, R, metaverse

Índice

Resumen **1**

Abstract **1**

Índice de figuras **4**

Índice de tablas **4**

1. Introducción **5**

1.1 Motivación y objetivos 5

1.2 Metodología 6

1.3 Estructura del trabajo 7

2. Topic modeling **8**

2.1 Descripción del topic modeling y casos de uso 8

2.2 Topic modeling mediante LDA 9

2.2.1 LDA: un algoritmo probabilístico y generativo *9*

2.2.2 Funcionamiento del LDA *13*

2.2.3 Gibbs sampling *14*

3. Guía práctica **18**

3.1 Descripción del caso de análisis utilizado en la guía práctica y de las fases del proceso de implementación del algoritmo 18

3.2 Fase 1: Recolección de datos y construcción de la base de datos 21

3.3 Fase 2: Preprocesamiento del texto 24

3.3.1 Tokenización *25*

3.3.2 Texto en minúsculas y eliminación de números y símbolos *27*

3.3.3 Eliminación de stopwords *28*

3.3.4 Lematización o stemming *29*

3.3.5 N-gramas y colocaciones *30*

3.3.6 Eliminación de lemas con muy alta y muy baja frecuencia *35*

3.4 Fase 3: Topic modeling mediante LDA 37

3.4.1 Número de tópicos a extraer: coherencia *38*

3.5 Fase 4: Interpretación y visualización de los tópicos 44

3.5.1 Interpretación de los tópicos *44*

3.5.2 Visualización de los tópicos *48*

3.6 Fase 5: Posibles usos de los tópicos encontrados 50

4. Conclusiones generales y posibles extensiones del trabajo 56

Bibliografía 58

Anexos 64

Anexo 1: Explicación del valor de la PMI ante una colocación 64

Anexo 2: Ejemplo de cálculo de la PMI 65

Anexo 3: Código R 66

Índice de figuras

Figura 1 Fases del proceso de implementación del algoritmo 19
Figura 2 Interés relativo del término "metaverse" en búsquedas de Google News..... 20
Figura 3 Pasos de preprocesamiento de textos 24
Figura 4 Colocaciones con bigramas..... 34
Figura 5 Coherencia UMass según número de tópicos 41
Figura 6 Catorce tópicos encontrados 44
Figura 7 Visualización de los tópicos con el paquete LDAVis..... 50
Figura 8 Distribución de los tópicos en los documentos..... 52
Figura 9 Relevancia de cada tópico dentro del corpus 54
Figura 10 Evolución de la prevalencia de los tópicos semana a semana 55

Índice de tablas

Tabla 1 Asignación de tópicos a cada palabra de cada documento..... 12
Tabla 2 Base de datos utilizada 23
Tabla 3 Tokenización de los documentos..... 26
Tabla 4 Colocaciones y PMI 33
Tabla 5 Extracto de "corpus_nostopwords" con las colocaciones incorporadas..... 35

1. Introducción

1.1 Motivación y objetivos

El topic modeling es, según Kherwa y Bansal (2019, p. 2), una técnica que "revela, descubre y anota la estructura temática de una colección de documentos". En definitiva, el topic modeling ayuda a entender cuáles son los temas, o tópicos, que prevalecen en una selección de documentos y, por tanto, permite sintetizar una gran cantidad de datos para extraer conclusiones semánticas de un conjunto de textos. Durante los últimos años, el uso del topic modeling en distintas áreas de conocimiento ha ido creciendo y popularizándose (Aranda et al., 2021). En efecto, encontramos en la literatura muchos artículos académicos explicando distintos temas de marketing, inversiones, educación y derecho mediante esta técnica (Reisenbichlet y Reutterer, 2018; Aranda et al., 2021). Para aplicar el topic modeling, es importante mencionar que existen varios algoritmos y el más utilizado es *Latent Dirichlet Allocation* (LDA), que se utilizará en el presente trabajo (Kherwa y Bansal, 2019).

De todo lo anterior, se deduce que el topic modeling cobra cada vez más relevancia en la investigación docente y en la industria. Dado el uso cada vez más frecuente, existen guías de implementación, como el famoso libro *Text Mining with R: A Tidy Approach*, de Silge y Robinson (2017). Este texto es un gran referente ampliamente utilizado en el ámbito del análisis de textos a nivel masivo, aportando información muy valiosa y didáctica. Sin embargo, al explicar varios aspectos de tratamiento y análisis de textos, no llega a profundizar en todos los detalles técnicos del topic modeling. En efecto, no aborda cuestiones relevantes a la hora de obtener buenos resultados, como son la utilización de la lematización, términos compuestos, número óptimo de tópicos a utilizar o la visualización como método importante de validación de los tópicos. Por todo ello, este trabajo pretende ser una guía práctica de aplicación del LDA en R que aborde todos estos aspectos que no se cubren en la gran redacción de Silge y Robinson (2017). Este objetivo general se va a conseguir a través de los siguientes objetivos específicos:

- Ofrecer una base conceptual y marco teórico sobre el algoritmo que dé un

conocimiento profundo del mismo para después poder abordar y aplicar correctamente la técnica.

- Desarrollar una guía práctica de implementación en R que aborde todo el proceso de topic modeling, desde la recolección de datos y el preprocesamiento hasta la visualización, incluyendo aspectos relevantes como la lematización, colocaciones y número de tópicos a extraer.
- Ofrecer ideas para mayor utilización de los tópicos extraídos.

1.2 Metodología

Para cumplir con los objetivos del trabajo, se pretende construir la guía práctica a través de un caso de análisis sobre el metaverso en artículos periodísticos. Para mostrar y detectar cuáles son los principales temas de conversación alrededor del metaverso en la prensa, se ha utilizado concretamente el periódico *The Guardian* debido a su accesibilidad. Gracias a la conexión con la API del periódico, perfeccionaremos nuestra base de datos no solo con los artículos que contengan la palabra *"metaverse",* sino también con los metadatos asociados a los artículos.

Para abordar el primer objetivo específico, se ha investigado la literatura existente, buscando información relevante en revistas académicas a través de bases de datos como *Web of Science* y *Scopus*. Tras la selección de la información más importante, se ha construido un marco teórico sobre el topic modeling y el LDA que ayuda a entender bien el algoritmo.

Para cubrir los dos últimos objetivos específicos, la guía de aplicación práctica en R se divide en las siguientes fases: recolección de datos, preprocesamiento, extracción de tópicos, interpretación y visualización de estos, y posibles usos. En cada una de las fases, se abordarán distintas cuestiones que surgen a la hora de implementar cada paso, acudiendo a fuentes académicas e ilustrando la aplicación a través del caso elegido. Todo el flujo de trabajo se realizará en R. Para cubrir cada paso de la aplicación, se explorarán distintas librerías y herramientas que ofrece R y se explicará cómo integrarlas construyendo un flujo de trabajo ágil.

1.3 Estructura del trabajo

Tras la presente introducción, el trabajo continuará en el Capítulo 2 con un marco teórico sobre el topic modeling y el LDA. Concretamente, se explicará esta técnica en profundidad y se presentarán casos generales de uso del topic modeling. Tras esta explicación, el trabajo continuará en el Capítulo 3 con la guía práctica. En primer lugar, se realizará una descripción del caso de análisis y de las fases del proceso de implementación del algoritmo de topic modeling. En segundo lugar, el trabajo seguirá con las distintas fases de implementación. Trataremos la recolección de datos en la primera fase y el preprocesamiento del texto en la segunda. En la tercera fase, identificaremos el número óptimo de tópicos a elegir y extraeremos dichos tópicos mediante Gibbs sampling. En la cuarta fase de la guía práctica, realizaremos la interpretación y visualización de los distintos tópicos encontrados. Finalmente, concluiremos la guía práctica con la quinta fase, en la cual se explicarán posibles usos de los tópicos extraídos. Posteriormente, el trabajo terminará en el Capítulo 4 con las conclusiones generales y las posibles extensiones del presente estudio.

2. Topic modeling

2.1 Descripción del topic modeling y casos de uso

En una era digital en la que la cantidad de datos tanto estructurados como no estructurados aumenta exponencialmente, resulta necesario contar con técnicas computacionales capaces de analizarlos y transformarlos en información valiosa para la toma de decisiones. Las técnicas de tratamiento y análisis de datos no estructurados, como son los textos, se engloban dentro del ámbito del *text mining*. Según Debortoli et al. (2016, p. 111), el text mining "permite extraer automáticamente información implícita, previamente desconocida y potencialmente útil a partir de grandes cantidades de datos textuales y de forma escalable y repetible". Dentro del text mining, contamos con técnicas como el análisis de sentimiento, que nos permite identificar las emociones presentes en un texto, o el topic modeling, objeto de estudio de nuestro trabajo (Dahal et al., 2019).

Como se ha comentado en la introducción del presente trabajo, el topic modeling ayuda a entender cuáles son los temas que prevalecen en una selección de documentos. Entrando en detalle, la primera clasificación de los modelos de topic modeling se realiza dividiéndolos entre modelos probabilísticos y modelos no probabilísticos (Kherwa y Bansal, 2019). Por un lado, los modelos probabilísticos, donde encontramos el LDA que estudiaremos en el presente trabajo, utilizan distribuciones probabilísticas dentro de la colección de documentos para determinar los temas principales de la misma (Blei et al., 2003). Por otro lado, los modelos no probabilísticos no se centran en la estructura estadística de la colección de documentos, sino en un análisis algebraico y matricial centrado especialmente en la frecuencia de las palabras dentro de los textos.

Es evidente que el potencial del topic modeling en la era digital con enormes cantidades de información es muy relevante. En efecto, su potencial se puede extender a diversos casos de uso, en los que encontramos, por ejemplo, "la recuperación de información, el análisis de redes sociales y el resumen de textos" (Kherwa y Bansal, 2019, p. 4). La recuperación de información se basa en devolver al usuario una serie de documentos según las palabras clave introducidas en un buscador. Gracias a la síntesis de varios

documentos en tópicos, el topic modeling resulta clave para poder asociar documentos a palabras clave y vice versa. A modo de ejemplo, los autores Zeng et al. (2017) desarrollaron un modelo mediante el cual pacientes de diabetes introducían preguntas con palabras clave y el modelo devolvía materiales educativos adecuados para responder las preguntas. En cuanto al análisis de redes sociales, el topic modeling permite analizar los temas principales que se comentan en las redes sociales acerca de una cuestión en particular. Por ejemplo, ante un lanzamiento de producto, una empresa podría identificar qué están comentando los usuarios sobre el nuevo producto y cuáles son las conversaciones más relevantes (Zhong y Schweidel, 2020). Finalmente, el caso de uso de resumen de textos que comentamos se ilustrará en nuestro caso de aplicación práctica, mediante el cual identificamos los temas más importantes sobre el metaverso mencionados en artículos periodísticos de *The Guardian.* En vez de tener que leer y estudiar todos los artículos que contengan la palabra "metaverse" en el periódico para entender qué se está comentando sobre este mundo virtual, los tópicos encontrados nos aportarán, de forma rápida y eficiente, información valiosa sobre los temas más relevantes.

2.2 Topic modeling mediante LDA

Tal y como se ha comentado, en esta guía de estudio se utilizará el algoritmo denominado Latent Dirichlet Allocation, acuñado en 2003 por los autores Blei, Ng y Jordan como algoritmo no supervisado de reducción de dimensionalidad. Además, es un modelo ampliamente utilizado en la literatura debido a su enorme potencial para descubrir tópicos con grandes textos.

2.2.1 LDA: un algoritmo probabilístico y generativo

El LDA parte de la suposición de que los documentos de un corpus (conjunto de documentos) son *bag-of-words,* o conjuntos de palabras cuyo orden no se considera relevante a la hora de utilizar el algoritmo. Además, esta premisa de "intercambiabilidad" también permite que el orden de los documentos dentro del corpus pueda obviarse al

realizar el topic modeling (Blei et al., 2003). De esta forma, se consigue simplificar el algoritmo y sus necesidades computacionales.

Como se puede entender tras esta introducción, en LDA, la palabra es la unidad mínima de estudio. Por tanto, un documento está compuesto por *N* palabras, y a su vez, un corpus está compuesto por *M* documentos. Al mismo tiempo, el algoritmo supone que un documento está compuesto por varios tópicos y que un tópico está compuesto por varias palabras. Teniendo en cuenta que el LDA es un algoritmo probabilístico, concluimos que (1) los documentos del corpus son mezclas probabilísticas de los tópicos y (2) los tópicos son mezclas probabilísticas de las palabras (Blei et al., 2003). Es decir, un documento está asociado a una distribución de probabilidades de tópicos y un tópico está asociado a una distribución de probabilidades de palabras. En este algoritmo, las distribuciones probabilísticas son distribuciones Dirichlet. Estas distribuciones son multinomiales (más de dos resultados, que en LDA son tópicos y palabras) y se utilizan para identificar la probabilidad de que una serie de sucesos simultáneos ocurra (Ponweiser, 2012).

El LDA se denomina también algoritmo generativo. Esto quiere decir que su base y fundamento está en la capacidad de generar las palabras de los documentos del corpus según las distribuciones de probabilidad arriba mencionadas. A modo de resumen y simplificando, el proceso de generación empieza viendo qué palabras se podrían asignar a cada tópico y qué tópicos se podrían asignar a cada documento. A continuación, para cada palabra que se quiere generar, tenemos en cuenta el documento en el que está para ver qué tópico sería más probable y, una vez elegido el tópico, vemos qué palabra sería más probable (Ponweiser, 2012).

Entrando en detalle, el proceso generativo en el que se basa el LDA se puede resumir en tres grandes pasos (Ponweiser, 2012):

1. Determinar la distribución (Φ) de las palabras dentro de cada tópico teniendo en cuenta que los tópicos siguen una distribución Dirichlet con hiperparámetro β
2. Determinar la distribución (θ) de los tópicos dentro de cada documento teniendo en cuenta que los documentos siguen una distribución Dirichlet con hiperparámetro α

3. Para cada una de las palabras (o posiciones reservadas a cada palabra) de cada documento se ha de:
    a. Determinar a qué tópico corresponde teniendo en cuenta que los tópicos para cada documento y palabra siguen una distribución multinomial con parámetro θ
    b. Generar una palabra teniendo en cuenta el tópico escogido en el punto 3a y suponiendo que las palabras siguen una distribución multinomial con parámetro Φ

A continuación, se va a explicar cada uno de los pasos en detalle y con ejemplos.

En primer lugar, se ha de asignar a cada tópico un vector con las probabilidades de que ese tópico contenga las palabras 1, 2, …, *V*, siendo *V* todo el vocabulario del que está compuesto el corpus. Este vector para cada tópico "k" se denomina Φ_k y sigue una distribución Dirichlet con parámetro β (que se explicará más adelante) (Ponweiser, 2012). A modo de ejemplo, suponiendo que hay tres tópicos dentro de los textos analizados y cinco palabras en el vocabulario:

Tópico 1: $\Phi_1 = \{0,2; 0,3; 0,1; 0,2; 0,2\}$ → El tópico 1 tiene una probabilidad del 20% de contener la palabra 1, del 30% de contener la palabra 2, del 10% de contener la palabra 3, del 20% de contener la palabra 4 y del 20% de contener la palabra 5.

Tópico 2: $\Phi_2 = \{0,1; 0,4; 0,2; 0,1; 0,2\}$ → El tópico 2 tiene una probabilidad del 10% de contener la palabra 1, del 40% de contener la palabra 2, del 20% de contener la palabra 3, del 10% de contener la palabra 4 y del 20% de contener la palabra 5.

Tópico 3: $\Phi_3 = \{0,3; 0,1; 0,3; 0,2; 0,1\}$ → El tópico 3 tiene una probabilidad del 30% de contener la palabra 1, del 10% de contener la palabra 2, del 30% de contener la palabra 3, del 20% de contener la palabra 4 y del 10% de contener la palabra 5.

En segundo lugar, se debe asignar a cada documento del corpus un vector con las probabilidades de que ese documento pertenezca al tópico 1, 2, … *K*, siendo *K* el número de tópicos que hay en el corpus. Este vector para cada documento "d" se denomina θ_d y sigue una distribución Dirichlet con parámetro α (que también se explicará más adelante) (Ponweiser, 2012). Por ejemplo, suponiendo que un corpus contiene dos documentos y existen tres tópicos en los textos analizados:

> **Documento 1:** $\theta_1 = \{0,3; 0,1; 0,6\}$ → El documento 1 tiene una probabilidad del 30% de encontrarse en el primer tópico, del 10% de encontrarse en el segundo tópico, y del 60% de encontrarse en el tercer tópico.
>
> **Documento 2:** $\theta_2 = \{0,5; 0,4; 0,1\}$ → El documento 2 tiene una probabilidad del 50% de encontrarse en el primer tópico, del 40% de encontrarse en el segundo tópico, y del 10% de encontrarse en el tercer tópico.

A continuación, el próximo paso a seguir en la construcción de este algoritmo es el de asignar a cada palabra de cada documento un tópico según la distribución multinomial de los tópicos con parámetro θ_d, como vemos en la Tabla 1 a modo de ejemplo.

Tabla 1

Asignación de tópicos a cada palabra de cada documento

	Palabra 1	**Palabra 2**	**Palabra 3**	**Palabra 4**	**Palabra 5**
Doc. 1	Tópico 3	Tópico 3	Tópico 3	Tópico 2	Tópico 2
Doc. 2	Tópico 1	Tópico 1	Tópico 2	Tópico 2	Tópico 3

Fuente: elaboración propia a partir de Ponweiser (2012)

Una vez hecha esta asignación de tópicos, para cada palabra o término que ocupa la posición "N" dentro del documento "M", el algoritmo genera una palabra. Esto lo hace teniendo en cuenta el tópico asignado a ese término y basándose en la distribución multinomial con parámetro Φ_k de las palabras (Ponweiser, 2012).

Teniendo toda esta descripción en cuenta, se entiende que el LDA es un algoritmo generativo que busca aquellas palabras que con alta probabilidad generaron un corpus con un determinado número de tópicos.

2.2.2 Funcionamiento del LDA

El LDA utiliza un proceso iterativo mediante el cual, utilizando distribuciones de probabilidad, sugiere tópicos encontrados en el corpus teniendo siempre en cuenta que efectivamente esos tópicos puedan luego explicar el corpus objeto de estudio (Blei et al., 2003).

El LDA trabaja con variables observables y variables desconocidas, o latentes. Por tanto, lo que busca el LDA es aprender de lo observable para identificar la estructura latente de los documentos y poder identificar los tópicos. Además, cuenta con tres parámetros que han de determinarse. En primer lugar, se ha de establecer el hiperparámetro α que determina la distribución Dirichlet de los documentos. En segundo lugar, se ha de indicar al algoritmo el hiperparámetro β que determina la distribución Dirichlet de los tópicos. Finalmente, se ha de introducir en el algoritmo el número de tópicos que se encuentran en el corpus. Para ello, se utilizan distintas métricas (que veremos en el apartado 3.4.1), como la perplejidad y la coherencia (Blei et al., 2003).

Según Blei et al. (2003), la estimación de los parámetros α y β se realiza mediante un problema de optimización a través del cual se busca maximizar la *log likehood* del corpus. Es decir, se buscan aquellos parámetros que con mayor probabilidad hayan podido generar las palabras del corpus objeto de estudio. A modo de ejemplo, los autores explican que la función a optimizar sería:

$$\ell(\alpha, \beta) = \log p(w|\alpha, \beta)$$

Sin embargo, según la literatura existente, los valores de α y β que mejores resultados dan son α = (50 / número de tópicos) y β = 0,1 (Griffiths y Steyvers, 2004; Ponweiser, 2012; Yan et al., 2009). Por tanto, por motivos de simplificación matemática y computacional, utilizaremos esos valores para los dos hiperparámetros. Es importante mencionar que valores de α altos implican que cada documento contendrá una proporción variada de los

tópicos, sin centrarse la distribución en unos únicos tópicos. De forma similar, valores de β altos indican que los tópicos contendrán una proporción variada de palabras, sin centrarse la distribución en unas únicas palabras. Los valores de α y β que nos indica la literatura permiten variedad de tópicos dentro de los documentos al mismo tiempo que se busca que los tópicos sean específicos con las palabras a incluir (Griffiths y Steyvers, 2004).

La literatura existente, sin embargo, no sugiere un número exacto de tópicos a utilizar en los análisis, ya que dependerá de cada corpus objeto de estudio. Por tanto, es el analista quien debe determinar el número de tópicos (*K*) con los que trabajar. Para poder determinar este hiperparámetro *K*, la literatura recomienda ejecutar el algoritmo para valores diferentes del mismo y elegir aquel valor de *K* que proporcione un mejor resultado (Prabha y Sardana, 2023). Para evaluar la bondad de este resultado, suelen utilizarse, como veremos en el apartado 3.4.1, dos métricas alternativas: la coherencia y la perplejidad (Chang et al., 2009; Vangara et al., 2021). Una vez decidido el número de tópicos a utilizar en el análisis, se hará el proceso de generación de tópicos mediante inferencia estadística.

2.2.3 Gibbs sampling

En el presente estudio, realizaremos el proceso de generación de tópicos con LDA mediante *Gibbs sampling*. Sin embargo, es importante mencionar que este método no es la única alternativa, ya que existen otros como el *Variational Expectation Maximization* (o VEM) (Liu et al., 2011). El método Gibbs sampling para el LDA fue desarrollado como un algoritmo Monte Carlo por Griffiths y Steyvers en el año 2004. Como técnica de inferencia Bayesiana, para poder utilizarla se necesitan las probabilidades a priori determinadas por los hiperparámetros α y β. Como hemos comentado en el apartado anterior, los hiperparámetros tomarán los siguientes valores (Griffiths y Steyvers, 2004):

$$\alpha = 50 / \text{número de tópicos}$$

$$\beta = 0{,}1$$

El Gibbs sampling utiliza un proceso iterativo para encontrar la distribución (Φ) de las palabras dentro de cada tópico y la distribución (θ) de los tópicos dentro de cada documento. Es decir, el Gibbs sampling es una técnica que permite asignar las distintas palabras del corpus a los tópicos latentes (Griffiths y Steyvers, 2004). El primer paso que se ha de llevar a cabo es la determinación del número de tópicos que hay en el corpus. Una vez determinado dicho número, procederemos a utilizar los valores de α y β recomendados por la literatura y mencionados anteriormente. Tras esta determinación, el algoritmo empieza asignando un tópico a cada palabra del corpus. Esta asignación se realiza de forma aleatoria, teniendo en cuenta que todas las palabras tienen la misma probabilidad de ser asignadas a los distintos tópicos. Una vez que las palabras hayan sido asignadas a los tópicos, automáticamente los tópicos han sido a su vez asignados a documentos (Griffiths y Steyvers, 2004). Por tanto, tenemos una distribución inicial de palabras dentro de cada tópico y de tópicos dentro de cada documento.

Una vez terminada la asignación inicial, documento a documento se coge cada palabra y se calculan iterativamente dos probabilidades:

(1) En primer lugar, calculamos $P(w_i|z_i = j)$, donde w_i es la palabra que ocupa la posición "i" y z_i es el tópico asignado a la palabra que ocupa la posición "i". Teniendo en cuenta la palabra que ocupa la posición "i" dentro del documento, queremos ver cuántas veces (sin contar w_i) dentro del corpus ese mismo término se ha atribuido al tópico "j" con relación a la cantidad de tokens que ha sido atribuida al tópico "j" (también sin contar w_i). De este modo, calculamos la probabilidad de que la palabra que ocupa la posición "i" se encuentre dentro del tópico denominado "j". Matemáticamente, podemos expresar esta probabilidad de la siguiente forma (Griffiths y Steyvers, 2004):

$$P(w_i|z_i = j) = \frac{n_{-i,j}^{(w_i)} + \beta}{n_{-i,j}^{(\cdot)} + W\beta}$$

Donde:

$n_{-i,j}^{(w_i)}$ = número de veces que aparece la palabra que se sitúa en la posición "i" del documento en el tópico "j" (sin contar la palabra en la posición "i")

β = hiperparámetro cuyo valor ha sido valorado en 0,1

W = número de palabras únicas en los documentos

$n_{-i,j}^{(\cdot)} + W\beta$ = número total de palabras dentro del corpus (sin contar la palabra en la posición "i")

(2) En segundo lugar, calculamos $\mathrm{P}(\mathrm{z}_i = \mathrm{j})$. Teniendo en cuenta el tópico "j" que ha sido atribuido a la palabra en la posición "i", queremos ver cuántas palabras han sido atribuidas a ese tópico dentro del documento por el que se está iterando, con relación al número total de palabras dentro del documento (todo esto sin contar la palabra que ocupa la posición "i"). De esta forma, veremos la probabilidad de que el tópico denominado "j" se encuentre en el documento por el que se está iterando. Matemáticamente, podemos expresar esta probabilidad de la siguiente forma (Griffiths y Steyvers, 2004):

$$\mathrm{P}(\mathrm{z}_i = \mathrm{j}) = \frac{n_{-i,j}^{(d_i)} + \alpha}{n_{-i,\cdot}^{(d_i)} + \mathrm{T}\alpha}$$

Donde:

$n_{-i,j}^{(d_i)}$ = número de veces que el tópico "j" atribuido a la palabra en la posición "i" ha sido asignado dentro del documento (sin contar la palabra en la posición "i")

α = hiperparámetro cuyo valor equivale a 50 / número de tópicos

T = número de tópicos

$n_{-i,\cdot}^{(d_i)} + \mathrm{T}\alpha$ = número total de palabras dentro del documento (sin contar la palabra en la posición "i")

Estas probabilidades que acabamos de mencionar se calculan para cada palabra dentro de cada documento. Para cada palabra, se multiplican ambas probabilidades para llegar a una probabilidad condicionada representada de la siguiente forma (Griffiths y Steyvers, 2004):

$$\mathrm{P}(\mathrm{z_i} = \mathrm{j}|\mathrm{w_i}) = \mathrm{P}(\mathrm{w_i}|\mathrm{z_i} = \mathrm{j}) * \mathrm{P}(\mathrm{z_i} = \mathrm{j})$$

Esta probabilidad condicionada representa la probabilidad de que la palabra en la posición "i" esté dentro del tópico "j" teniendo en cuenta las asignaciones de las distintas palabras del corpus. Teniendo en cuenta las probabilidades condicionadas para todas las palabras, el Gibbs sampling funciona de forma iterativa. En cada iteración, se vuelve a realizar una asignación de palabras a tópicos según las probabilidades condicionadas halladas en la iteración previa y se calculan de nuevo las probabilidades condicionadas. El algoritmo continúa iterando hasta que se estabilizan las probabilidades condicionadas y las asignaciones dejan de cambiar en cada iteración. Esta convergencia indicará que se habrá llegado a una asignación de palabras a tópicos óptima (Griffiths y Steyvers, 2004).

Con este proceso de Gibbs sampling, la convergencia nos permite hallar la distribución (Φ) de las palabras dentro de cada tópico y la distribución (θ) de los tópicos dentro de cada documento (Griffiths y Steyvers, 2004). De esta forma, el algoritmo encontrará los tópicos latentes del corpus.

3. Guía práctica

3.1 Descripción del caso de análisis utilizado en la guía práctica y de las fases del proceso de implementación del algoritmo

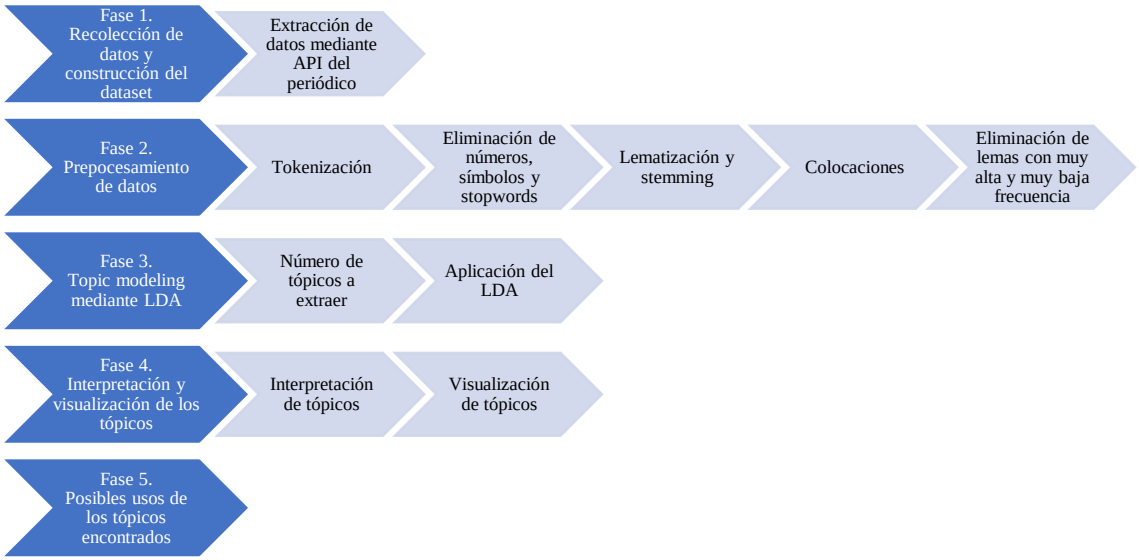
Como se ha comentado previamente, se pretende construir la guía práctica a través de un caso de estudio sobre el metaverso en el ámbito periodístico. En concreto, queremos identificar los tópicos más relevantes en los artículos del periódico *The Guardian* que incluyan el término "metaverse". De esta forma, podremos entender, mediante las distintas fases del proceso de implementación del algoritmo, qué se está comentando acerca de este mundo virtual y cuáles son los temas más relevantes.

Según Park y Kim (2022), el metaverso es "un mundo virtual tridimensional donde avatares participan en actividades políticas, económicas, sociales y culturales y donde coexiste lo real e irreal" (p. 4211). En definitiva, es un mundo interactivo e inmersivo que busca clarecer las barreras entre lo físico y lo virtual. Además, se caracteriza por sus *entornos* (que pueden ser reales, irreales o mixtos), sus *interfaces* (tridimensionales, inmersivas y/o físicas), su *interacción* (puede actuar como red social y permite la colaboración y comunicación entre usuarios) y su *valor social* (el metaverso capta y genera valor financiero, medioambiental, cultural, etc.) (Dwivedi et al., 2022). En cuanto a sus aplicaciones, el metaverso puede utilizarse en casi cualquier ámbito. Por ejemplo, puede utilizarse en videojuegos, en el mundo de la educación para realizar experimentos o viajar a la época medieval, en oficinas remotas para crear un ambiente de oficina real, en organismos al servicio de la ciudadanía para realizar simulaciones de desastres naturales, etc. Es importante mencionar que toda revolución digital conlleva también algunas desventajas, como posibles ataques de privacidad o problemas éticos ante falta de legislación (Park y Kim, 2022). En conclusión, el metaverso es un mundo incipiente y todavía muy desconocido, por lo que resulta interesante estudiar qué se está comentando sobre el mismo en los periódicos, y por tanto, en la sociedad en general.

El proceso de implementación del algoritmo de topic modeling (LDA) que seguiremos en esta guía práctica consta de cinco fases principales que quedan resumidas en la Figura 1.

Figura 1

Fases del proceso de implementación del algoritmo



Fuente: elaboración propia

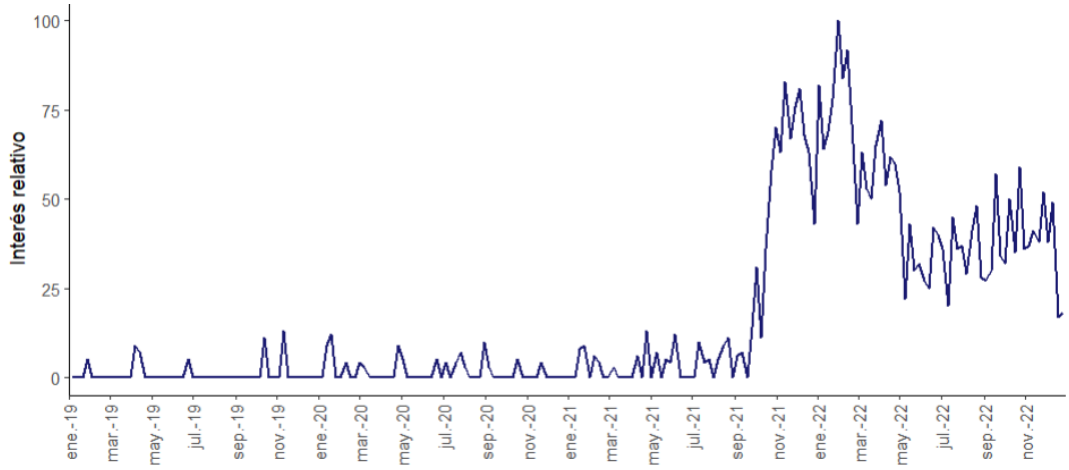
Para tener una visión amplia y global de los distintos tópicos, buscaremos en la primera fase del proceso artículos del periódico *The Guardian*, un periódico británico de gran relevancia. Accederemos a estos artículos mediante llamadas a la API (*application programming interface*) del periódico a través de R. Es importante enfatizar que se ha elegido este periódico ya que es el único que ofrece de manera gratuita la descarga de artículos completos a través de su API.

Además de incluir la restricción de que todos los artículos descargados tienen que contener la palabra “metaverse”, se ha decidido añadir una restricción temporal para analizar artículos desde el 1 de enero del 2019 hasta el 29 de abril de 2023. Con este caso de aplicación, veremos cómo varían los tópicos sobre el metaverso según el horizonte temporal que estemos analizando. Teniendo en cuenta que Meta anunció el 28 de octubre de 2021 la creación de su metaverso (Meta, 2021), se espera que el número de artículos

sobre el metaverso sea mucho mayor después de esa fecha. Sí que existirán artículos previos a esa fecha ya que el término "metaverso" se dio a conocer en 1992 en un relato de ciencia ficción del escritor Neil Stephenson titulado *Snow Crash* (Park y Kim, 2022; Stephenson, 1992). En efecto, buscando el término "metaverse" en Google Trends para ver cómo ha evolucionado el interés sobre dicha palabra en las búsquedas de Google News (agregador de noticias de Google), podemos concluir que antes del anuncio de Meta en octubre del 2021, el interés era escaso, mientras que llega a un máximo entre octubre del 2021 y marzo del 2022 (Google Trends, 2023). En el gráfico, el máximo interés sobre el término conlleva una puntuación de 100, mientras que el mínimo conlleva una puntuación de 0.

Figura 2

Interés relativo del término "metaverse" en búsquedas de Google News



Fuente: elaboración propia a partir de datos descargados de Google Trends (2023)

Una vez recolectados los artículos con sus respectivos metadatos (fecha de publicación, sección en el periódico, URL, etc.), se habrá recopilado una base de datos potente para realizar topic modeling.

En la segunda fase del proceso, se realizará el preprocesamiento de los datos. Por tanto, realizaremos una limpieza del texto, eliminando todo aquello que introduzca "ruido", para hallar los tópicos más relevantes. Por tanto, eliminaremos especialmente signos de puntuación, números, símbolos, y stopwords (notablemente artículos, determinantes y

pronombres usados con frecuencia en el texto sin aportar información importante). Tras la limpieza del texto, procederemos a la tokenización y normalización del mismo. Finalmente, detectaremos las colocaciones más importantes para identificar aquellas palabras que suelen estar juntas en los textos.

En la fase número tres, aplicaremos el algoritmo LDA y hallaremos los tópicos de nuestro estudio. Además, previamente estudiaremos distintas métricas de coherencia para determinar el número óptimo de tópicos a extraer.

En la cuarta fase, se procederá con la interpretación de los tópicos descubiertos. Es decir, concluiremos qué comunica cada tópico sobre el metaverso. Por tanto, podremos identificar sus usos, las percepciones que genera en la sociedad, etc. Además, para comprender mejor los tópicos, realizaremos visualizaciones de los mismos.

En la quinta y última fase, veremos cómo los tópicos encontrados pueden utilizarse para entender mejor qué se está comunicando sobre el metaverso. Podremos identificar los temas más importantes y entender su evolución en el tiempo.

3.2 Fase 1: Recolección de datos y construcción de la base de datos

Como ya se ha comentado, la extracción de los datos de los artículos se hará en R a través de la API del periódico. Una API, o *application programming interface*, es un programa mediante el cual diferentes aplicaciones pueden comunicarse entre sí (Jacobson et al., 2011). En nuestro caso, la aplicación de R se comunicará con *The Guardian* gracias a su API. R pedirá información sobre los artículos y la API permitirá que el periódico devuelva los datos solicitados. De esta forma, podemos descargar una gran cantidad de datos de manera rápida y eficiente.

Para descargar los datos de *The Guardian*, solicitamos una clave única y secreta al periódico para poder hacer la llamada a la API[1]. A continuación, puesto que R pone a disposición un paquete denominado "guardianapi" (Odell, 2019), procedemos a instalarlo y abrirlo. Este paquete permite el acceso a la API abierta de *The Guardian.*

[1] El enlace para solicitar la API es el siguiente: https://bonobo.capi.gutools.co.uk/register/developer

```
install.packages("guardianapi")

library(guardianapi)
```

Guardamos en el entorno de trabajo la clave API y la abrimos con la función "gu_api_key" del paquete "guardianapi". Por motivos de seguridad, nunca podremos hacer pública nuestra clave API.

```
Sys.setenv(GU_API_KEY = "tu_clave_secreta")

gu_api_key(check_env = TRUE)
```

Descargaremos todos los artículos que contengan el término "metaverse" y que hayan sido publicados entre una fecha de inicio y una de fin. En este caso, se tomó como fecha de inicio el 1 de enero de 2019 y como fecha de fin el 29 de abril de 2023[2]. Los datos descargados se guardarán en una base de datos denominada "datos".

```
termino <- "metaverse"

fecha_inicio <- "2019-01-01"

fecha_fin <- "2023-04-29"

datos <- gu_content(query = termino, from_date = fecha_inicio, to_date = fecha_fin)
```

Los datos descargados contienen 46 variables y 304 filas o publicaciones. Empezando por las variables, para centrar nuestro análisis y quedarnos únicamente con aquellas que son más relevantes, procederemos a utilizar y seleccionar solamente las siguientes tres variables:

- **web_publication_date:** fecha y hora de publicación en el periódico
- **body_text:** texto del cuerpo de la publicación
- **type:** tipo de publicación (blog o artículo)

En cuanto a las filas, es importante mencionar que la descarga contiene tanto artículos como blogs. Puesto que queremos analizar únicamente artículos periodisticos, se filtrarán

[2] Alternativamente, si se prefiere poner la fecha actual como fecha de fin, se recomienda el uso de la función "today()" del paquete "lubridate" de R (Spinu et al., 2023).

sólo aquellos registros correspondientes a artículos (variable “type” es igual a “article”) con la función “filter” del paquete “dplyr” (Wickham et al., 2022).

Para realizar la selección de variables y el filtro de los registros, ejecutamos el siguiente código, utilizando la función “head” al final para ver una muestra de nuestra base de datos (veáse la Tabla 2):

```
GUARDIAN_subset <- subset(datos, select = c("web_publication_date", "body_text", "type"))

library(dplyr)

GUARDIAN<-filter(GUARDIAN_subset, type== "article")

head(GUARDIAN)
```

Tabla 2

Base de datos utilizada

web_publication_date	body_text	type	section_name	web_title	standfirst	wordcount
2023-02-07 13:56:20	Pictures of 100 Birkin bags co[…]	article	Fashion	Are brands prot[…]	Luxury retailer and […]	620
2022-11-02 18:50:10	The other comedy in US tech la[…]	article	Business	Mark Zuckerberg[…]	Meta’s share price i[…]	773
2022-05-14 14:00:04	Psychotherapist Nina Jane Pate[…]	article	Technology	Can we create a[…]	In the increasingly […]	1785
2022-10-27 17:56:35	After shares in Facebook’s par[…]	article	Technology	Meta shares dip[…]	Virtual reality gamb[…]	571
2023-04-26 23:47:42	Meta revenue surpassed analyst[…]	article	Technology	Meta reports su[…]	The company posted $[…]	671
2022-10-20 13:00:55	Almost every December all arou[…]	article	Games	Lost in Roblox’[…]	This football-themed[…]	818

Fuente: elaboración propia a partir de los datos del caso de estudio

Para eliminar cualquier posibilidad de artículos duplicados o vacíos, aplicamos los siguientes comandos:

```
GUARDIAN <-GUARDIAN[!duplicated(GUARDIAN$body_text),]

GUARDIAN <-na.omit(GUARDIAN)
```

Tras los cambios aplicados a la base de datos, contaremos finalmente en nuestro estudio con unos datos compuestos por 286 artículos (filas) y 3 variables (columnas).

3.3 Fase 2: Preprocesamiento del texto

Descargados los textos, es importante realizar un preprocesamiento de los mismos. El preprocesamiento es una fase esencial que se realiza antes de aplicar el algoritmo LDA para limpiar y preparar los textos y poder así obtener mejores resultados. En efecto, multitud de estudios explican la importancia del preprocesamiento, y Denny y Spirling (2017) concluyen que un buen preprocesamiento aumenta la validez e interpretabilidad de los tópicos.

En el presente trabajo, nos centraremos en la metodología de preprocesamiento indicada en el artículo "*Applying LDA topic modeling in communication research: Toward a valid and reliable methodology*", altamente citado en el mundo académico (Maier et al., 2018). Concretamente, los autores del artículo consideran crucial seguir el orden de los siguientes pasos:

Figura 3

Pasos de preprocesamiento de textos

Fuente: elaboración propia a partir de Maier et al. (2018)

En R, existen varios paquetes que permiten realizar el preprocesamiento de textos. Por ejemplo, "tidytext" y "quanteda" son paquetes ampliamente utilizados y existen varias guías de implementación de estos dos famosos paquetes (Silge y Robinson, 2017; Weidmann, 2023). Sin embargo, en este trabajo, se ha decidido utilizar el paquete "udpipe" debido a su gran potencial (Wijffels, 2022). En efecto, los autores del mismo destacan su relevancia frente a otros paquetes, indicando que el paquete permite realizar etiquetado gramatical (*Part-of-Speech Tagging*, en inglés), hacer topic modeling con lemas y extraer palabras compuestas (Wijffels, 2023). Además, el paquete contiene el conjunto de algoritmos "UDPipe" diseñados por Straka y Straková en 2017 con redes neuronales. Estos algoritmos han sido entrenados con textos en más de 50 idiomas para

poder transformar un texto a una estructura CoNLL-U ("*Computational Natural Language Learning*") en la que se logra identificar tokens, estructuras gramaticales, lemas y dependencias, entre otros (Straka y Straková, 2017). Los potentes algoritmos de UDPipe consiguen de forma rápida, eficiente y con grandes resultados realizar la tokenización, etiquetado gramatical, lematización y análisis de dependencias necesarios para preprocesar textos para desarrollar topic modeling (Wijffels, 2022).

3.3.1 Tokenización

Un texto o artículo es un tipo de dato no estructurado. Sin embargo, para que pueda ser procesado por cualquier algoritmo, es necesario transformarlo y así dotarlo de cierta estructura. Para entender esta transformación, es importante tener en cuenta las siguientes definiciones:

Un *documento* es un texto y, en este estudio, será un artículo de periódico. Un conjunto de documentos se llama *corpus* y, en nuestro caso, será en concreto la totalidad de los 286 artículos contenidos en el conjunto de datos "GUARDIAN". Además, la unidad más pequeña de análisis se denomina *token.* Por tanto, el proceso de tokenización implica dividir los documentos en tokens que, de forma común, suelen ser términos o palabras (Silge y Robinson, 2017).

El paquete "udpipe" de R permite hacer la tokenización de forma rápida. En primer lugar, tendremos que descargar y abrir la librería.

```
install.packages("udpipe")

library(udpipe)
```

El paquete "udpipe" contiene varios modelos predeterminados con algoritmos capaces de realizar la tokenización de forma eficiente según el idioma del corpus. Teniendo en cuenta que todos nuestros artículos están en inglés, procederemos con la descarga y carga del modelo inglés (Wijffels, 2022).

```
ud_model_download <- udpipe_download_model(language = "english")

ud_model <- udpipe_load_model(ud_model_download$file_model)
```

Una vez cargado el modelo, continuamos con la tokenización. Para ello, utilizamos la función "udpipe_annotate", que nos devolverá los distintos tokens de cada documento, aportando también información sobre la categoría gramatical de cada token (sustantivo, adverbio, pronombre, etc.) y el lema de los mismos (que explicaremos más adelante). Esta función necesita que se le especifiquen dos argumentos: (1) el modelo predeterminado y descargado y (2) el corpus que queremos transformar a tokens. Tras ejecutar la función, convertiremos la respuesta devuelta en una tabla de datos con la función "as.data.frame". De esta forma, podremos visualizar mejor el resultado de la tokenización, como vemos en la Tabla 3 (Wijffels, 2022).

```
corpus<-GUARDIAN$body_text

corpus_tokenizado <- udpipe_annotate(ud_model, x = corpus)

corpus_tokenizado_data_frame <- as.data.frame(corpus_tokenizado)

head(corpus_tokenizado_data_frame)
```

Tabla 3

Tokenización de los documentos

doc_id	paragraph_id	sentence_id	sentence	token_id	token	lemma	upos	xpos
doc1	1	1	Pictures of 100 Birkin bags covered	1	Pictures	picture	NOUN	NNS
doc1	1	1	Pictures of 100 Birkin bags covered	2	of	of	ADP	IN
doc1	1	1	Pictures of 100 Birkin bags covered	3	100	100	NUM	CD
doc1	1	1	Pictures of 100 Birkin bags covered	4	Birkin	birkin	PROPN	NNP
doc1	1	1	Pictures of 100 Birkin bags covered	5	bags	bag	NOUN	NNS
doc1	1	1	Pictures of 100 Birkin bags covered	6	covered	cover	VERB	VBN

Fuente: elaboración propia a partir de los datos del caso de estudio.

Como podemos ver en la Tabla 3, cada término de las frases de los documentos del corpus se transforma en tokens (ver columna "token"). Además, en la columna "upos" podemos ver la categoría gramatical de cada token ya que el paquete "udpipe" realiza el importante proceso de *Part-of-Speech Tagging* en el procesamiento de textos. El *Part-of-Speech Tagging* asigna "a cada palabra … su respectiva categoría sintáctica" (Ekbal y Saha, 2013, p. 288). De este modo, cada token tendrá asignado su categoría gramatical, pudiendo así entender en el corpus qué tokens son adjetivos, sustantivos, números, símbolos, verbos,

nombres propios, etc. Esta identificación nos ayudará a reconocer aquellos tokens comprendidos en las categorías gramaticales que utilizaremos en el topic modeling. La función "udpipe_annotate" también devuelve los lemas (cuya definición y explicación se verá en el apartado 3.3.4) de los tokens en la columna denominada "lemma". Para obtener el lema correcto de cada token, el algoritmo se basa en la categoría gramatical, de tal forma que teniendo en cuenta las frases "él vino a casa" y "él bebió vino", el lema del primer "vino" sería "venir" y el lema del segundo sería el sustantivo s"vino".

La capacidad del algoritmo detrás de "udpipe_annotate" (una red neuronal, como se comentaba anteriormente) se demuestra fácilmente viendo ejemplos del *Part-of-Speech Tagging*. Por ejemplo, el algoritmo es capaz de identificar "US" como la abreviatura de "United States" y clasificarlo come nombre propio ("PROPN").

3.3.2 Texto en minúsculas y eliminación de números y símbolos

Según Maier et al. (2018), el siguiente paso en la fase de preprocesamiento de textos es el cambio del texto a minúsculas y la eliminación de números y símbolos. Para cambiar los lemas (tokens transformados, como explicaremos más adelante) a minúsculas, utilizamos el comando "tolower". De esta manera, palabras como "metaverse", "Metaverse" o "METAVERSE" se convierten en el mismo término, "metaverse".

```
corpus_tokenizado_data_frame$lemma<-
tolower(corpus_tokenizado_data_frame$lemma)
```

Para eliminar los números y símbolos, nos vamos a ayudar de la columna "upos" que hemos visto para identificar aquellos tokens cuya clasificación es de números o símbolos. Ejecutando el siguiente comando, podemos ver las distintas clasificaciones empleadas:

```
unique(corpus_tokenizado_data_frame$upos)

[1] "DET"   "ADJ"   "NOUN"  "ADP"   "PROPN" "PUNCT" "ADV"   "PART"  "PRON"
"AUX"   "SYM"   "NUM"   "VERB"  "CCONJ" "SCONJ" "X"     "INTJ"
```

"PUNCT" se refiere a puntuación, "SYM" a símbolos y "NUM" a números. Por tanto, con el próximo comando eliminaremos todos aquellos tokens cuya categoría gramatical sea una de las tres. Aprovechamos también para eliminar "X" y "PART", que incluyen

todo lo que el *Part-of-Speech Tagging* no ha sabido identificar por la rareza de los tokens (Universal Dependencies, 2022). De esta forma, conseguiremos empezar a limpiar el texto para quedarnos con lo esencial (Wijffels, 2022).

```
corpus_no_PUNCT_SYM_NUM <- subset(corpus_tokenizado_data_frame, !(upos %in% c("PUNCT", "SYM", "NUM", "X", "PART")))
```

Es posible que tras este último paso todavía queden algunos símbolos que UDPipe no haya identificado correctamente. En el corpus que se está analizando, por ejemplo, se han identificado símbolos como "-", "&" o "#" que no han sido eliminados. Por tanto, se procede a eliminarlos manualmente:

```
corpus_int<-subset(corpus_no_PUNCT_SYM_NUM,!grepl("^[&';@[:digit:]]", lemma))

corpus_int$lemma<-gsub("\\-", "", corpus_int$lemma)

corpus_int$lemma<-gsub("\\#", "", corpus_int$lemma)
```

3.3.3 Eliminación de stopwords

El tercer paso que indican Maier et al. (2018) es el de la eliminación de stopwords. Las stopwords son aquellos tokens que se repiten con alta frecuencia y no aportan información al texto. Ejemplos en español de stopwords serían "lo", "la", "un", "no", "el", etc. Paquetes de R como "tidytext", "stopwords" o "quanteda" contienen una función para eliminar una lista predeterminada de stopwords según el idioma del corpus (Silge y Robinson, 2017; Benoit, 2022; Benoit et al., 2023). Sin embargo, el paquete que estamos utilizando, "udpipe", no cuenta con esta lista, muy útil en el preprocesamiento de textos. Por tanto, procederemos a descargar el paquete "stopwords" para poder utilizar la función "data_stopwords_smart" indicando que queremos las stopwords del idioma inglés (Benoit, 2022). Además, podemos añadir nuestras propias stopwords a la lista predeterminada. Por ejemplo, en nuestro caso, como estamos buscando artículos que contengan la palabra "metaverso", la palabra en sí no aportará al topic modeling ya que la gran mayoría de los documentos la contendrá. Por tanto, la podemos añadir a la lista de stopwords. También podemos añadir a la lista los tokens "Facebook" y "Meta" ya que el "boom" del metaverso causado por el cambio de nombre de Facebook a Meta ha

generado una excesiva mención de estas palabras. De la misma manera, se incluyeron en la lista de stopwords los términos: "*year*", "*company*", "*time*", "*Guardian*", "*people*", "*thing*" y "*life*".

```
library(stopwords)

stopwords<-data_stopwords_smart$en

stopwords<-c(stopwords, "meta", "metaverse", "facebook", "year","company",
"time", "guardian", "people", "thing", "life")
```

Para eliminar las stopwords, utilizamos la función "subset" para quedarnos únicamente con aquellos tokens y lemas (cuya definición veremos en el próximo apartado) que no son stopwords.

```
corpus_nostopwords<-subset(corpus_int, !lemma %in% c(stopwords))

corpus_nostopwords<-subset(corpus_nostopwords, !token %in% c(stopwords))
```

3.3.4 Lematización o stemming

El cuarto paso que indican Maier et al. (2018) es el de la lematización o stemming, dos maneras de normalizar el corpus. Según Balakrishnan y Lloyd-Yemoh (2014), la lematización es una técnica que realiza un "análisis de vocabulario y morfológico de una palabra y busca eliminar sus terminaciones flexivas, devolviendo así las palabras a su forma de diccionario" (p. 263). Es decir, tras aplicar lematización, los tokens "metaversos", "inventó" e "innovaron" se convertirán respectivamente en los lemas "metaverso", "inventar" e "innovar". Un inconveniente que tiene esta técnica son los altos recursos computacionales requeridos ya que el algoritmo necesita identificar la categoría gramatical del token antes de poder lematizarlo. Por otra parte, el stemming es "un proceso que se utiliza para eliminar los sufijos derivativos y las inflexiones (es decir, sufijos que cambian la forma de las palabras y sus funciones gramaticales) para que las variantes de las palabras puedan tener las mismas raíces" (Balakrishnan y Lloyd-Yemoh, 2014, p. 262). Por ejemplo, las palabras "información" e "informe" comparten la raíz "inform" y un algoritmo de stemming convertirá estos tokens en esa raíz. La duda que surge al hacer el stemming gira en torno al nivel al que hay que llevar un token a su raíz.

Si acortamos demasiado la raíz, estaríamos cometiendo el error conocido como *overstemming.* Por ejemplo, si la raíz de "planta", "planeta" y "planificar" se establece en "plan", se alteraría el significado de los dos primeros tokens. Si por el contrario no acortamos lo suficiente la raíz, estaríamos cometiendo *understemming*. A modo de ilustración, si la raíz de "librería" y "librito" es "libr", el token "librar" también se reduciría a "libr". Por tanto, estaríamos dando un significado incorrecto a este último token.

Aunque ninguna de las dos técnicas es perfecta, la literatura prefiere la lematización, ya que deriva en mejores resultados y permite que los tópicos tengan un mayor grado de interpretabilidad (Balakrishnan y Lloyd-Yemoh, 2014; Maier et al., 2018; Syed, 2018). Teniendo además en cuenta que el paquete "udpipe" solo permite la lematización, procederemos el análisis teniendo en cuenta los lemas.

Como vimos en la Tabla 3 al tokenizar el corpus, la función "udpipe_annotate" también nos devolvió los lemas en la columna "lemma".

3.3.5 N-gramas y colocaciones

El último paso en el preprocesamiento consiste en eliminar aquellos lemas con muy alta y muy baja frecuencia en el corpus (Maier et al., 2018). Sin embargo, antes de continuar con estas eliminaciones, proponemos realizar previamente la inclusión de colocaciones en nuestra base de datos para no perder información relevante. Como hemos estado viendo, un documento se puede dividir en tokens mediante la tokenización. La tokenización que hemos hecho se ha realizado con unigramas, es decir, cada token estaba compuesto por un único término. Al dividir el corpus en unigramas, el orden de los términos se pierde, y grupos de palabras como "inteligencia artificial" pierden su sentido al separarse en "inteligencia" por un lado, y "artificial" por otro. Esta inconveniencia puede resolverse con los denominados "n-gramas": *n* términos consecutivos dentro del texto. "Tecnología barata" es un bigrama al contener dos términos, y "el ordenador nuevo" es un trigrama compuesto por tres términos. Al juntarse los términos en n-gramas, se convierten en tokens compuestos por varios términos.

Cuando hablamos de n-gramas, surge también la idea de las colocaciones, aquellos tokens que con frecuencia mayor al 50% aparecen juntos en el texto y pueden empezar a sugerir varias ideas sobre el corpus (Pecina, 2005). Mientras que una colocación es un n-grama, un n-grama no tiene por qué ser una colocación. En nuestro corpus, el bigrama "tecnología barata" puede simplemente aparecer por azar una única vez. Sin embargo, "Silicon Valley", en el contexto del metaverso, es un bigrama cuyas dos palabras aparecen juntas con frecuencia y tiene sentido considerarlas como una única entidad. Por ello, "tecnología barata" podría considerarse únicamente un bigrama, mientras que "Silicon Valley" es tanto un bigrama como una colocación. Para asegurarnos de que captamos toda la información relevante del texto, la identificación de colocaciones resulta de gran importancia a la hora de realizar topic modeling.

Las colocaciones en un corpus se extraen mediante métricas de asociación, como pueden ser la *pointwise mutual information* (PMI), *mutual dependency* (MD) o *log-frecuency biased mutual dependency* (LFMD). Varios estudios han demostrado que la PMI es la métrica que mejores resultados genera, por lo que será la que utilizaremos en nuestro caso de estudio (Paperno et al., 2014; Pecina, 2005). La PMI mide cómo de dependientes son los términos que conforman un n-grama (n > 1) dentro del corpus, teniendo en cuenta la frecuencia con la que aparecen esos n-gramas en comparación con la frecuencia con la que aparecen los términos individuales del n-grama dentro del texto. Para entenderlo mejor, teniendo en cuenta un bigrama, la PMI se mediría de la siguiente forma (Church y Hanks, 1990):

$$\mathrm{PMI} = \log_2\left(\frac{\mathrm{P}(w_1, w_2)}{\mathrm{P}(w_1) \cdot \mathrm{P}(w_2)}\right)$$

donde:

w_1 y w_2 corresponden al primer y segundo término del bigrama, respectivamente

$P(w_1, w_2)$ es la probabilidad de que w_1 y w_2 aparezcan juntos en el corpus

$P(w_1)$ es la probabilidad de que w_1 aparezca en el corpus

$P(w_2)$ es la probabilidad de que w_2 aparezca en el corpus

El valor más bajo que puede tener la PMI es cero (Church y Hanks, 1990). Si los términos w_1 y w_2 son independientes, y por tanto, no forman una colocación, la probabilidad de que aparezcan juntos en el corpus será igual a la multiplicación de la probabilidad de que ocurra w_1 y de la probabilidad de que ocurra w_2 (es decir, $\mathrm{P}(\mathrm{w}_1, \mathrm{w}_2) = \mathrm{P}(\mathrm{w}_1) \cdot \mathrm{P}(\mathrm{w}_2)$). Teniendo en cuenta la ecuación de PMI, el resultado de la misma será igual a cero.

$$\mathrm{PMI} = \log_2\left(\frac{\mathrm{P}(\mathrm{w}_1, \mathrm{w}_2)}{\mathrm{P}(\mathrm{w}_1) \cdot \mathrm{P}(\mathrm{w}_2)}\right) = \log_2\left(\frac{\mathrm{P}(\mathrm{w}_1) \cdot \mathrm{P}(\mathrm{w}_2)}{\mathrm{P}(\mathrm{w}_1) \cdot \mathrm{P}(\mathrm{w}_2)}\right) = \log_2(1) = 0$$

En el caso de que se trate de una colocación real, los términos que la constituyen aparecerán juntos en los corpus con mayor frecuencia que si fuesen términos independientes. Por tanto, $\mathrm{P}(\mathrm{w}_1, \mathrm{w}_2)$ sería mayor que $\mathrm{P}(\mathrm{w}_1) \cdot \mathrm{P}(\mathrm{w}_2)$ y la PMI sería mayor que cero. Para una explicación matemática más detallada de este razonamiento, véase el Anexo 1.

Mientras que la PMI tiene una cota inferior de cero (en el caso de independencia), no existe cota superior y depende de las probabilidades o frecuencias con las que aparecen los términos. Si las colocaciones son poco frecuentes, la matemática nos indica que la PMI será muy alta. Para eliminar el ruido que puede generar el hecho de que dominen bigramas que aparecen muy pocas veces dentro del corpus, solo utilizaremos aquellos bigramas que aparecen por lo menos 10 veces (Phelps, 2021).

Para identificar las colocaciones en nuestro caso de estudio, utilizaremos la función "keywords_collocation" del paquete de "udpipe". Esta función necesita varios inputs:

(1) el corpus,

(2) los términos del corpus que se quieren utilizar (en nuestro caso, los lemas),

(3) en qué agrupación del corpus queremos buscar los n-gramas (nosotros buscaremos dentro de las frases de los distintos documentos),

(4) el tipo máximo de n-gramas que queremos (nosotros elegiremos bigramas), y finalmente,

(5) la frecuencia mínima que debe tener un n-grama dentro del corpus para ser considerado como posible colocación (en nuestro caso, se establecerá una frecuencia mínima de 10).

Empezaremos pidiendo a la función los bigramas del corpus:

```
bigramas <- keywords_collocation(corpus_nostopwords, term = "lemma", group =
c("doc_id", "sentence_id"), ngram_max = 2, n_min = 10)

head(bigramas)
```

Tabla 4

Colocaciones y PMI

keyword	ngram	left	right	freq	freq_left	freq_right	pmi	md	lfmd
las vegas	2	las	vegas	11	11	13	13,22	-0,24	-13,71
gran turismo	2	gran	turismo	14	14	14	13,12	0,00	-13,12
neal stephenson	2	neal	stephenson	11	11	15	13,02	-0,45	-13,91
tik tok	2	tik	tok	15	15	16	12,93	-0,09	-13,11
elden ring	2	elden	ring	12	12	20	12,60	-0,74	-14,08
sci fi	2	sci	fi	15	15	21	12,53	-0,49	-13,50

Fuente: elaboración propia a partir de los datos del caso de estudio.

En nuestro caso, viendo la imagen del resultado de la función en la Tabla 4, vemos que el bigrama “gran turismo” aparece en el corpus 14 veces, “gran” aparece 14 veces y “turismo” aparece 14 veces. Todas las veces que aparece el término “gran” va acompañado de “turismo”, y viceversa. Por tanto, podemos concluir que estamos ante términos completamente dependientes y que la PMI será alta, tratándose así de una colocación potente. Por otra parte, el bigrama “tik tok” aparece 15 veces, “tik” aparece 15 veces y “tok” aparece 16 veces. Por tanto, podemos concluir que “tik tok” también es una colocación potente ya que todas las veces (menos una) que aparece el término “tok” está junto a “tik”. En efecto, la PMI de 12,93 de este bigrama indica que es una colocación importante. Para una explicación matemática más detallada de este resultado de PMI, véase el Anexo 2.

Para poder determinar si un n-grama es una colocación importante (es decir, si su PMI es suficientemente alta), es necesario recurrir a la literatura existente. Algunos autores mencionan que una PMI de 3 es suficiente para indicar que una colocación es buena,

mientras que otros explican que una PMI de 4 para los bigramas y una PMI de 5 para los trigramas es necesaria (Phelps, 2021; Petrovic et al., 2006). En nuestro estudio, para intentar no restar importancia a algunos n-gramas, utilizaremos un mínimo de 3. En R, por tanto, solo nos quedaremos con aquellos n-gramas con PMI mayor o igual a 3:

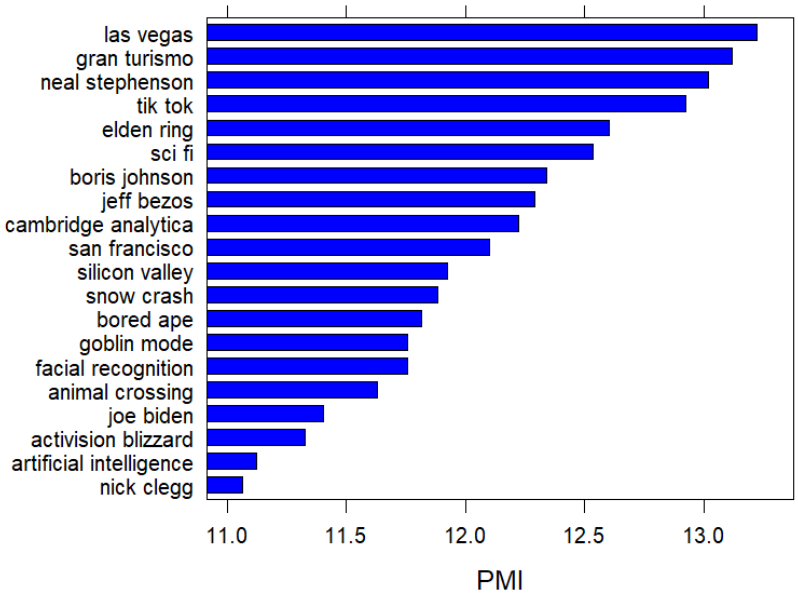
```
bigramas <- bigramas[bigramas$pmi >= 3,]
```

Para visualizar las colocaciones más fuertes dentro del corpus, es importante ordenar de mayor a menor PMI las distintas colocaciones y seleccionar aquellas con mayor PMI. En nuestro caso, proponemos visualizar las 20 mejores colocaciones. Para ello, ejecutamos el siguiente código (NLP with R and UDPipe, 2020):

```
bigramas$key <- factor(bigramas$keyword, levels = rev(bigramas$keyword))

library(lattice)[3]

barchart(key ~ pmi, data = head(subset(bigramas, freq>3), 20), col = "blue",

         main = "Colocaciones con bigramas", xlab = "PMI")
```

Figura 4

Colocaciones con bigramas



Fuente: elaboración propia a partir de los datos del caso de estudio y del código de NLP with R and UDPipe (2020)

[3] (Sarkar, 2023)

Al visualizar las colocaciones, podemos empezar a entender algunos mensajes transmitidos en el corpus. Por ejemplo, podemos entender que Silicon Valley ha sido posiblemente un catalizador del metaverso, que el escándalo de Cambridge Analytica y lo que conlleva está dentro de las discusiones del metaverso, y que la inteligencia artificial es importante en este contexto.

Para enriquecer nuestro corpus, queremos que las colocaciones estén incluidas dentro del mismo. Por tanto, con la función "txt_recode_ngram" de "udpipe", remplazaremos los unigramas por las colocaciones identificadas (Wijffels, 2022). Por ejemplo, si en el texto aparece el término "*nonfungible*" seguido de "*token*", entonces, el primer término se sustituirá por la colocación "*nonfungible token*" y el segundo se quedará en blanco. Al quedarse en blanco, no se duplicará la colocación. En la Tabla 5, un extracto de la base "corpus_nostopwords" que incluye las colocaciones, podemos ver en la columna "term" el resultado del ejemplo comentado.

```
corpus_nostopwords$term <- corpus_nostopwords$lemma

corpus_nostopwords$term<-txt_recode_ngram(corpus_nostopwords$lemma, compound
= bigramas$keyword, ngram = bigramas$ngram, sep = " ")
```

Tabla 5

Extracto de "corpus_nostopwords" con las colocaciones incorporadas

doc_id	paragraph_id	sentence_id	sentence	token_id	token	lemma	upos	xpos	[…]	term
doc1	1	2	In the case, being heard	35	non-fungible	nonfungible	ADJ	JJ	[…]	nonfungible token
doc1	1	2	In the case, being heard	36	tokens	token	NOUN	NNS		*NA*

Fuente: elaboración propia a partir de los datos del caso de estudio

3.3.6 Eliminación de lemas con muy alta y muy baja frecuencia

Según varios autores (Maier et al., 2018; Denny y Spirling, 2017), se recomienda eliminar aquellos lemas que aparezcan en menos del 0,5% de los documentos y aquellos lemas que aparezcan en más del 99% de los documentos. Al eliminar aquellas palabras que no aparecen casi en ningún documento del corpus, estaremos quitando ruido para mantener lo importante para hacer LDA. Asimismo, al eliminar los lemas que se repiten mucho en

el corpus, no dejaremos que esos lemas resten importancia al resto y protagonicen los tópicos sin poder extraer buenas interpretaciones y conclusiones. Además, según Maier et al. (2020), al eliminar estos lemas, la necesidad de alta capacidad computacional disminuye, haciendo que el algoritmo de LDA pueda funcionar de forma más confiable, rápida, económica y eficiente.

En "udpipe", existe la función "dtm_remove_sparseterms" que elimina automáticamente aquellos lemas que aparecen en muy pocos documentos (Wijffels, 2022). Sin embargo, no existe una función similar para eliminar aquellos lemas que aparecen en muchos documentos. Por tanto, recurriremos a la función "dfm_trim" del paquete "quanteda", ya que nos permite realizar las eliminaciones que queremos (Benoit et al., 2023). Antes de ello, realizaremos dos cambios en el corpus. En primer lugar, como indica la literatura, eliminaremos aquellas palabras que contengan tres letras o menos, ya que la aportación que hacen estas palabras al topic modeling es muy escasa (Parra et al., 2016). En segundo lugar, seleccionaremos para el topic modeling únicamente aquellos lemas que sean sustantivos ("NOUN"), adjetivos ("ADJ") y nombres propios ("PROPN"). En efecto, es una práctica común en artículos académicos realizar esta selección y, en este contexto, los sustantivos, adjetivos y nombres propios ofrecen una mayor aportación semántica (Jacobi et al., 2015; Chen et al., 2013).

```
corpus_nostopwords <- corpus_nostopwords[nchar(corpus_nostopwords$term)>3, ]

corpus_nostopwords <- subset(corpus_nostopwords, upos %in% c("NOUN", "ADJ",
"PROPN"))
```

Puesto que debemos asegurar que el corpus se lee bien con el paquete "quanteda", antes de utilizar la función "dfm_trim" habrá que realizar una serie de pasos de conversión de formatos de "udpipe" a formatos de "quanteda" mediante la función "as.dfm", tal y como se muestra a continuación (Benoit et al., 2023).

```
library(quanteda)

dtf <- document_term_frequencies(corpus_nostopwords, document = "doc_id",
term = "term")

dtm <- document_term_matrix(dtf)
```

```
dfm_quanteda<-as.dfm(dtm)
```

Finalmente, eliminamos los términos que aparezcan en menos del 0,5% y más del 99% de los documentos, y cambiamos el resultado a un formato que pueda posteriormente entender la función "LDA" del paquete "topicmodels" (Grun y Hornik, 2023).

```
dfm_trimmed<-dfm_trim(dfm_quanteda, min_docfreq = 0.005, max_docfreq = 0.99,
docfreq_type="prop")

dtm=convert(dfm_trimmed, to="topicmodels")
```

3.4 Fase 3: Topic modeling mediante LDA

Una vez que tenemos el corpus preprocesado y una idea sobre las colocaciones, el próximo paso es empezar a realizar el topic modeling. En R, utilizamos la función "LDA" del paquete "topicmodels" para extraer los tópicos mediante LDA y con la técnica de Gibbs sampling (Grun y Hornik, 2023). La función LDA contiene, entre otros, los siguientes argumentos:

- El argumento "x" es la *Document Term Matrix*[4] que contiene todos los tokens del corpus.
- El argumento "k" representa el número de tópicos que se han de extraer. En el próximo apartado, se indicará cómo determinar el número óptimo de tópicos.
- El argumento "method" nos permite elegir entre el método de Gibbs sampling y el de VEM. En este apartado, seleccionaremos el método de Gibbs sampling.
- El argumento "control" nos permite en primer lugar establecer el valor de los hiperparámetros α y β (este último denominado delta en R). Asimismo, este argumento también nos permite indicar una semilla para poder reproducir los mismos resultados cada vez que se ejecute el código (puesto que la asignación inicial es aleatoria, cada vez obtendríamos resultados distintos si no se incluye una

[4] La *Document Term Matrix* es una matriz que muestra la frecuencia con la que aparece cada término del corpus dentro de cada uno de los documentos (Afzali y Kumar, 2019).

semilla).

- El argumento “initialize” determina cómo se realiza la primera asignación de palabras a tópicos. Teniendo en cuenta el estudio de Griffiths y Steyvers (2004) en este caso también, utilizaremos una asignación inicial aleatoria (denominada “random” en R).

Antes de ejecutar el algoritmo de LDA, procederemos a determinar en el próximo apartado el número óptimo de tópicos a extraer.

3.4.1 Número de tópicos a extraer: coherencia

Es importante mencionar que la elección del número de tópicos es un paso importante para evitar tanto el infra-ajuste como el sobreajuste del modelo de LDA. Por un lado, un número escaso de tópicos llevará al modelo a estar infra-ajustado, haciendo que se pierda información al generar tópicos muy generales. Por otro lado, un número demasiado elevado de tópicos hará que el modelo de LDA esté sobreajustado, desgranando demasiado los tópicos y haciendo que los temas principales y más importantes del corpus sean difíciles de entender (Vangara et al., 2021). Para evitar ambos casos, procederemos a realizar un estudio minucioso utilizando métricas de coherencia.

Tal y como explica la literatura existente, las métricas de perplejidad que se utilizaban anteriormente para evaluar el número óptimo de tópicos no se consideran actualmente adecuadas (Pasquali, 2016). La perplejidad fue evaluada por Blei, Ng y Jordan (2003) en la famosa publicación en la que explican el algoritmo de LDA. La perplejidad es una métrica que determina cómo de bien un modelo representa los datos. Es decir, cómo de bien es capaz de predecir los datos o palabras de un texto (Jacobi et al., 2015). Sin embargo, “esta métrica no considera la coherencia semántica de los tópicos encontrados, haciendo difícil evaluar el rendimiento del modelo” (Pasquali, 2016, p. 21). Al no tener en cuenta la coherencia semántica, la métrica de perplejidad llegaba a contradecirse con la interpretación humana. Por tanto, se ha decidido utilizar la métrica de coherencia en lugar de la perplejidad en el presente trabajo.

La coherencia es una métrica que se centra en las similitudes semánticas de las palabras de un tópico. Cuanto más similares semánticamente sean estas palabras, mayor será la coherencia e interpretabilidad del tópico (Mei et al., 2007). De forma resumida, la coherencia de un tópico es la "media del nivel de coherencia de cada par de palabras" (Pasquali, 2016, p. 23). Es importante mencionar que existen distintas métricas de coherencia y que estas pueden clasificarse como intrínsecas o extrínsecas. Una métrica de coherencia intrínseca utiliza el propio corpus objeto de estudio para evaluar la coherencia de las palabras dentro de los tópicos. Sin embargo, las métricas extrínsecas utilizan corpus externos (como puede ser Wikipedia) para evaluar la coherencia (Pasquali, 2016). Una métrica de coherencia intrínseca es la denominada coherencia "UMass", mientras que una métrica extrínseca es la coherencia "UCI" (Stevens et al., 2012). Teniendo en cuenta que en el presente trabajo no se va a utilizar ningún corpus externo, utilizaremos la métrica de coherencia UMass, desarrollada por Mimno et al. (2011).

Mimno et al. (2011) explican que su métrica de coherencia se basa en la probabilidad condicionada de que dos palabras de un tópico estén dentro de un mismo documento dado que una de ellas lo está. En efecto, la ecuación que determina el nivel de coherencia UMass puede expresarse de la siguiente manera (Pasquali, 2016):

$$UMass(w_i, w_j) = \log\left(\frac{p(w_i, w_j) + \epsilon}{p(w_i)}\right)$$

El numerador tiene en cuenta la probabilidad de que las palabras w_i y w_j se encuentren en un mismo documento, el denominador considera la probabilidad con la que la palabra w_i aparece en los documentos y ϵ es un factor de "smoothing" que los autores establecen en 1 para que se evite tener que calcular el logaritmo de cero. Si las dos palabras aparecen con mucha frecuencia juntas teniendo en cuenta la probabilidad con la que aparece la primera, el nivel de coherencia de estas dos palabras será alto. Cuanto mayor sea la probabilidad de que dos palabras aparezcan en el mismo documento, mayor será su coherencia y mayor será el numerador. Al incorporar un logaritmo en la ecuación de coherencia UMass, cuanto mayor sea el numerador teniendo en cuenta el denominador, el nivel de coherencia será más próximo a cero. Para evaluar la coherencia de un tópico

entero, se realiza una media de la coherencia de los pares de palabras más importantes. Cuanto mayor sea el nivel de coherencia, mejor será el tópico y su interpretación. Calculando la media de los niveles de coherencia de cada tópico, llegamos al nivel de coherencia del modelo entero. A la hora de decidir cuántos tópicos queremos que tenga nuestro modelo, queremos elegir aquel número de tópicos que generen la mejor coherencia del modelo con todos sus tópicos.

Para el caso concreto que estamos analizando, se explorarán valores de *K* (número de tópicos) comprendidos entre 2 y 40. Para cada valor de *K*, se ejecutará el algoritmo LDA con Gibbs sampling y se evaluará la coherencia del modelo. Entre todos los valores de *K* evaluados, se elegirá aquel con mayor coherencia (más próxima a cero). Para generar este proceso en R, se implementará un bucle en el que se irán ejecutando modelos LDA (con la función "lda" del paquete "topicmodels" de Grun y Hornik (2023)) con distintos números de tópicos y se irán guardando las coherencias conseguidas. Dado el carácter aleatorio del algoritmo y que en cada ejecución pueden obtenerse resultados diferentes, cada modelo LDA con un número específico de tópicos se ejecutará cinco veces. Posteriormente, se calculará la media de la coherencia para estas cinco ejecuciones. Ejecutando el siguiente código adaptado de la información práctica del paquete "text2vec" de Selivanov et al. (2022), llegamos a un gráfico (véase la Figura 5) en el que, de forma visual, conseguimos identificar un número adecuado de tópicos. La función de coherencia "coherence" se consigue gracias al paquete "text2vec". En esta función, indicando que la métrica que queremos calcular es "mean_logratio" con un factor de "smooth" de 1, calculamos una métrica muy similar a la coherencia UMass.

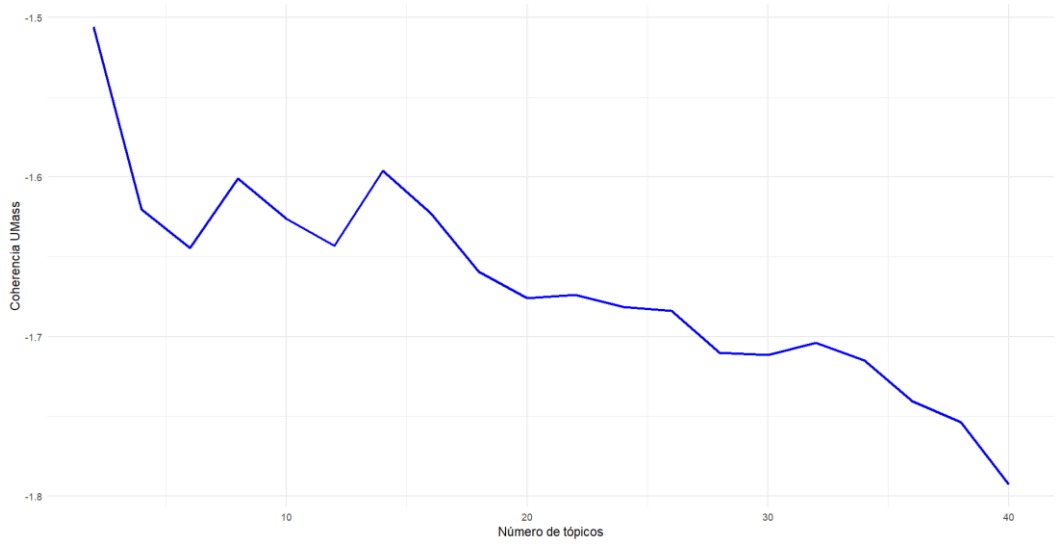
```
library(text2vec)

library(ggplot2)[5]

range<-seq(2, 40, by=2)

tcm=crossprod((as.matrix(dtm)))

coherence_logratio=data.frame()
```

[5] (Wickham et al., 2023)

```
for(i in range){

  resultsloop_logratio=data.frame()

  for (j in seq(1,5,by=1)){

    topicmodeling <- LDA(dtm, method="Gibbs", k=i,iter=200,  burnin=100,
         control=list(alpha = 50/i, delta=0.1), initialize="random")

    words_topic<-terms(topicmodeling, k=10)

    words_topic_matrix<-as.matrix(words_topic)

    coherenceloop_logratio[j,1]=mean(coherence(words_topic_matrix, tcm,
       metrics = c("mean_logratio"), smooth=1, n_doc_tcm = nrow(tcm)))}

  coherence_logratio[i,1]=i

  coherence_logratio[i,2]=mean(coherenceloop_logratio$V1)}

ggplot() + geom_line(data= coherence_logratio, aes(x = V1, y = V2), color =
'blue', stat="identity", size=1)+labs(x="Número de tópicos", y="Coherencia
UMass")+theme_minimal()
```

Figura 5

Coherencia UMass según número de tópicos



Fuente: elaboración propia a partir de los datos del caso de estudio e información del paquete "text2vec" de Selivanov et al. (2022)

Según la literatura, se debe coger el nivel de coherencia más próximo a cero (Prabha y Sardana, 2023). En este caso, puesto que no tendría sentido resumir un corpus tan extenso

en únicamente dos tópicos, el número de tópicos que hace que la coherencia UMass sea más próxima a cero es 14. Por tanto, procederemos con nuestro análisis teniendo en cuenta 14 tópicos. En el código, indicamos mediante el comando "i = 14" que queremos extraer 14 tópicos.

```
i=14

modelo <- LDA(dtm, method = "Gibbs", k =i, control = list(alpha = 50/i,
delta=0.1, seed=58), initialize="random")
```

Una vez ejecutado el algoritmo, queremos entender qué palabras se han asignado a cada tópico. Para poder verlo de forma ordenada, nos basamos en la distribución (Φ, "phi") de las palabras dentro de cada tópico. Para cada uno de ellos, elegimos las 15 palabras con mayor "phi", para así poder identificar los 15 términos más relevantes de cada tópico. En efecto, estas 15 palabras son aquellas que ayudarán a interpretar la temática del tópico encontrado. Con el siguiente código, adaptado de Foley (2020), conseguimos la salida que se observa en la Figura 6.

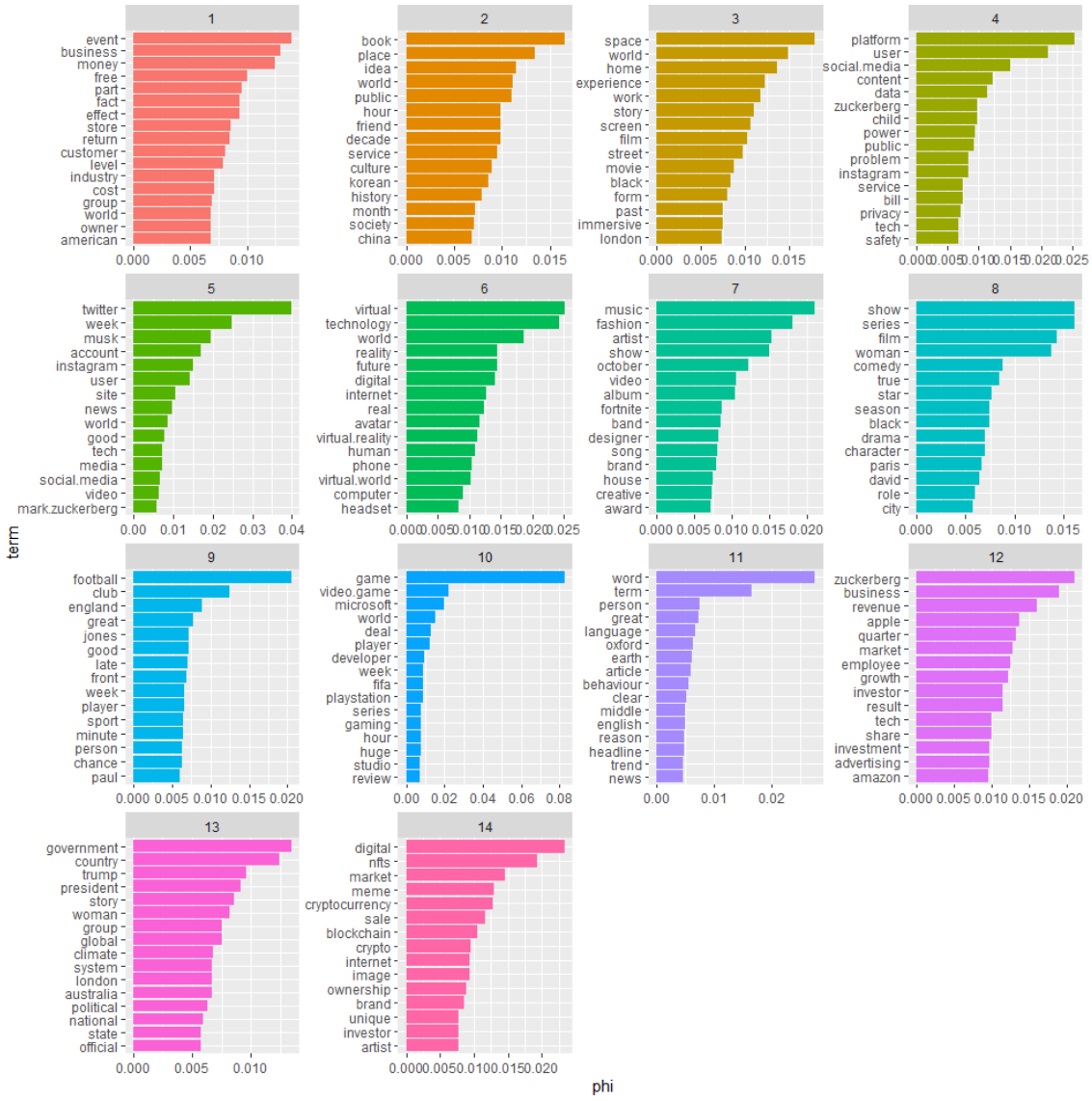
```
library(reshape2)

library(tidytext)[6]

phi <- data.frame(as.matrix(posterior(modelo)$terms))

phi$row_name <- (rownames(phi))

topicos <- melt(phi)

colnames(topicos) <- c("topic", "term", "phi")

topicos$topic<-as.numeric(topicos$topic)



top_terms <- topicos %>%

  group_by(topic) %>%

  top_n(15,phi) %>%

  ungroup() %>%
```

[6] (Robinson y Silge, 2023)

```
  arrange(topic,-phi)



plot_topic <- top_terms %>%

  mutate(term = reorder_within(term, phi, topic)) %>%

  ggplot(aes(term, phi, fill = factor(topic))) +

  geom_col(show.legend = FALSE) +

  facet_wrap(~ topic, scales = "free") +

  coord_flip() +

  scale_x_reordered()

plot_topic
```

Figura 6

Catorce tópicos encontrados



Fuente: elaboración propia a partir de los datos del caso de estudio. Visualización ejecutada con adaptación de código de Foley (2020)

3.5 Fase 4: Interpretación y visualización de los tópicos

3.5.1 Interpretación de los tópicos

Tras haber extraído los tópicos y los lemas más relevantes de cada uno de ellos, podemos proceder a la interpretación de los mismos. Esta fase del proceso de topic modeling es de gran importancia ya que con ella se utiliza la interpretación humana para entender cuáles

son los temas principales expresados dentro del corpus. Para esta interpretación, nos centraremos en la Figura 6 e iremos tópico a tópico tratando de entender qué comunica cada uno. Asimismo, se dará un nombre a cada tópico, que posteriormente se añadirá en R para las siguientes visualizaciones.

- **Tópico 1 – Adaptación al cliente:** las palabras "*business*", "*money*", "*customer*", "*event*", "*store*" e "*industry*" hacen alusión a un nuevo paradigma en el que la industria – para generar ganancias y mantener el negocio – debe adaptarse a un nuevo tipo de cliente que requiere experiencias de compra distintas en las tiendas y mediante eventos.
- **Tópico 2 – Cultura virtual:** las palabras "*book*", "*culture*" y "*korean*" (referencia a pop coreano) forman parte de un tópico que nos revela una nueva cultura virtual, en la que los libros cobran vida virtual, los cantantes son avatares y los grupos de fanes se comunican en el metaverso.
- **Tópico 3 – Experiencia inmersiva:** los lemas "*experience*", "*screen*", "*film*", "*movie*" e "*immersive*" hacen referencia a las nuevas experiencias cinematográficas que ofrece el metaverso. La tecnología del metaverso permite tener experiencias inmersivas que transforman medios artísticos tradicionales que poco a poco se quedarán anticuados.
- **Tópico 4 – Privacidad en las plataformas:** las palabras "*platform*", "Instagram", "Zuckerberg", "*social media*", "*data*", "*privacy*", "*power*", "*safety*" hacen alusión al poder que tiene el metaverso en las redes sociales y las preocupaciones que nacen del mismo. En efecto, la privacidad de datos es un tema que alarma tras el anuncio del desarrollo del metaverso por parte de Meta. Teniendo en cuenta que fue Meta quien empezó a impulsar el metaverso como nueva herramienta tecnológica, este tópico nos ayuda a entender quién impulsa la expansión del metaverso.
- **Tópico 5 – Comunicación en redes:** los tokens "Twitter", "Instagram", "*user*", "*news*" y "*social media*" hacen referencia a la comunicación sobre el metaverso que están haciendo distintos usuarios en varias redes sociales. Particularmente,

Twitter (con su accionista Elon Musk) es una gran fuente de información con constantes conversaciones sobre las últimas novedades.

- **Tópico 6 – Realidad virtual:** las palabras "*virtual reality*", "*virtual world*", "*avatar*", "*headset*" y "*computer*" explican que el metaverso es un mundo virtual mediante el cual, a través de herramientas como pueden ser cascos y ordenadores, uno puede convertirse en un avatar y descubrir una nueva realidad y mundo virtual. Con este tópico vemos la importancia de los bigramas dentro del corpus, que nos permiten identificar que no solo estamos hablando de una realidad, sino de una realidad que es virtual.
- **Tópico 7 – Conciertos y eventos:** las palabras "*music*", "*fashion*", "*show*", "*creative*" y "*awards*" nos explican que el metaverso está empezando a incluir conciertos, eventos de moda y entregas de premio. Por tanto, se entiende que es una nueva plataforma para reunir a gente y transformar eventos tradicionales en eventos creativos y digitales.
- **Tópico 8 – Contenido cinematográfico sobre el metaverso:** las palabras "*show*", "*series*", "*film*" y "*season*" hacen referencia a películas y series. Tras ver qué documentos contienen este tópico (como veremos más adelante), este tópico hace referencia a menciones del metaverso en películas y series. A diferencia del tópico 3, no se está haciendo referencia a las nuevas experiencias cinematográficas que posibilita el metaverso, sino a menciones del metaverso en medios artísticos tradicionales.
- **Tópico 9 – Mundo del deporte:** los tokens "*football*", "*club*", "*player*" y "*sport*" hacen alusión al mundo del deporte. En efecto, este tópico nos explica que el deporte tiene un rol importante en el metaverso, en el cual se pueden crear comunidades de aficionados, comprar y vender equipamiento deportivo virtual y jugar en competiciones virtuales.
- **Tópico 10 – Videojuegos:** las palabras "*video game*", "*player*", "*developer*", "*fifa*" y "*playstation*" explican el universo de los videojuegos dentro del metaverso. En efecto, el metaverso ofrece oportunidades para crear entornos de juegos que suponen nuevas experiencias tanto para desarrolladores como para

jugadores.

- **Tópico 11 – Palabra del año:** las palabras "Oxford", "*language*", "*english*", "*trend*", "*word*" y "*term*" nos indican que este tópico está relacionado con la importancia de la palabra "metaverso" en el idioma inglés. Explorando los artículos que contienen un alto porcentaje de este tópico (método que veremos más adelante), se descubre que este tópico hace referencia a la famosa "Palabra del año" de Oxford. En efecto, los artículos de nuestro corpus revelan que la palabra "metaverso" fue nominada por Oxford en 2022, aunque no llegó a ganar.
- **Tópico 12 – Inversión y riqueza:** las palabras como "*business*", "*investment*", "*growth*", "*market*", "*result*" y "*revenue*" se refieren a los resultados financieros de las empresas en el mercado. Con innovaciones tecnológicas como el metaverso, siempre surge la duda de cómo estas se van a poder rentabilizar y cómo las empresas que las impulsan se van a ver beneficiadas en los mercados tras realizar inversiones relevantes.
- **Tópico 13 – Política:** las palabras "*government*", "*president*", "*political*", "*state*", "*country*" y "*official*" hacen alusión a las ideas y problemas políticos que pueden surgir a raíz del metaverso. Al ser un mundo todavía muy desconocido, se plantean cuestiones políticas y gubernamentales ya que inevitablemente toda la población se verá afectada.
- **Tópico 14 – Inversiones descentralizadas:** los tokens "*nfts*", "*meme*", "*crypto*", "*blockchain*" e "*investor*" hacen alusión a una nueva forma de inversión descentralizada que se hace posible gracias al blockchain. En efecto, esta tecnología posibilita el intercambio en el metaverso de nuevos activos (memes e imágenes digitales, por ejemplo) que generan nuevas formas de inversión y de riqueza.

Una vez que tenemos los tópicos identificados e interpretados, podemos proceder a nombarlos en R con los títulos atribuidos. Para ello, ejecutamos las siguientes líneas de código:

```
topicNames<-c("Adaptación al cliente", "Cultura virtual", "Experiencia
inmersiva", "Privacidad en las plataformas", "Comunicación en redes",
"Realidad virtual", "Conciertos y eventos", "Contenido cinematográfico sobre
el metaverso", "Mundo del deporte", "Videojuegos", "Palabra del año",
"Inversión y riqueza", "Política", "Inversiones descentralizadas")
```

3.5.2 Visualización de los tópicos

Para entender mejor nuestros tópicos y cómo se diferencian los unos de los otros, es importante poder visualizarlos en un espacio bidimensional. En efecto, al tratarse de varios tópicos con distintos términos, una visualización en forma de síntesis de todos los tópicos puede ser de gran utilidad. Para ello, podemos hacer uso del paquete "LDAVis", presentado en el artículo "*LDAVis: A method for visualizing and interpreting topics*" (Sievert y Shirley, 2014). Según los autores, su herramienta de visualización de tópicos tiene tres objetivos principales. En primer lugar, se quiere entender qué nos quiere decir cada uno de los tópicos. Como veremos, este objetivo se cumple observando la parte derecha de la visualización de la Figura 7 (que equivale a los gráficos de los términos y tópicos mostrados en la Figura 6). En segundo lugar, queremos entender cómo de prevalente es cada uno de los tópicos. Finalmente, el último objetivo es el de entender cómo de similares o distintos son los tópicos. Estos dos últimos objetivos se observan en la parte izquierda de la visualización (Sievert y Shirley, 2014). La parte izquierda de la visualización posiciona los tópicos en dos dimensiones tras haber realizado internamente un proceso de reducción de dimensionalidad mediante "*Principal Component Analysis*" (Sievert y Shirley, 2014). Cuanto más lejos estén los unos de los otros en el plano, más distintos serán esos tópicos y mejor se habrá realizado el análisis. Cabe destacar que la visualización que ofrece el paquete "LDAVis" es interactiva y el usuario puede interactuar con ella, por ejemplo, seleccionando los tópicos en el gráfico de la izquierda y viendo los términos más relevantes de los mismos a la derecha.

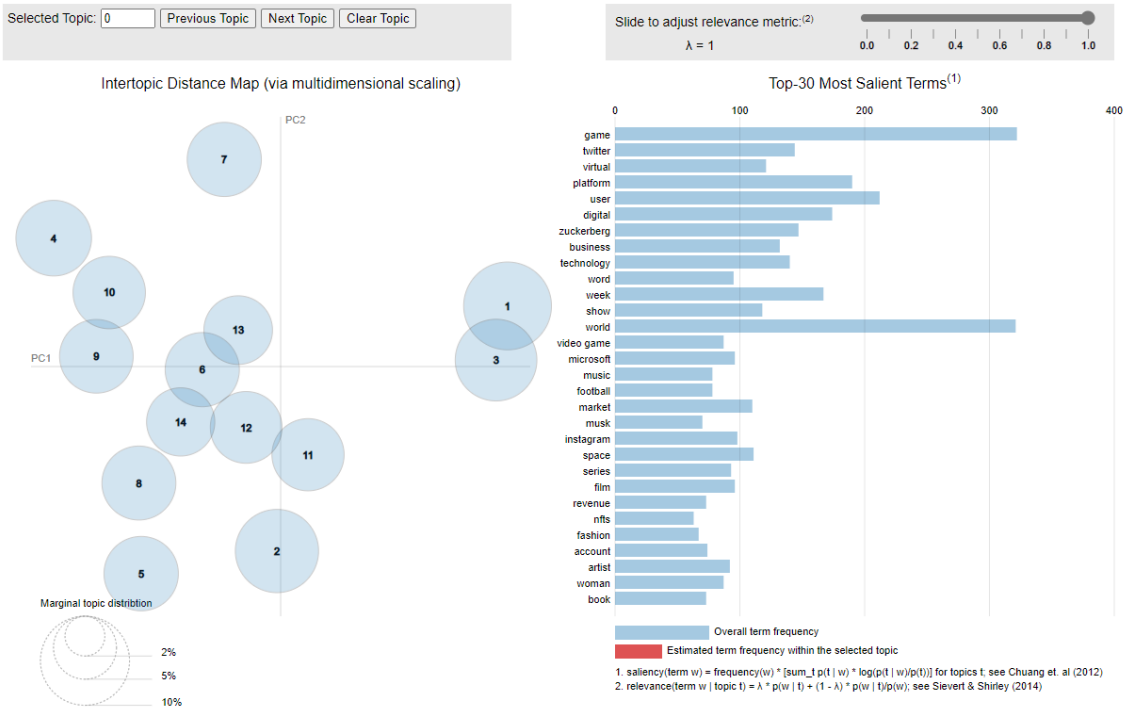
Para poder realizar la visualización, el paquete "LDAVis" exige la creación de un documento JSON. En este documento, tendremos que indicar la distribución (Φ) de las palabras dentro de cada tópico, la distribución (θ) de los tópicos dentro de cada

documento, las palabras del corpus, sus frecuencias, y el número de palabras dentro de cada documento (gracias al paquete “slam” (Hornik et al., 2022)). Posteriormente, utilizaremos la librería “servr" para poder visualizar el documento JSON con la función “serVis” (Xie, 2023). A continuación se muestra el código utilizado para esta visualización (basado en Holster, 2022). La ejecución de este código da como resultado la Figura 7 ya explicada anteriormente.

```
library(LDAvis)

library(slam)

phi <- as.matrix(posterior(modelo)$terms)

theta <- as.matrix(posterior(modelo)$topics)

vocab <- colnames(phi)

doc.length <- as.vector(table(dtm$i))

term.frequency <- col_sums(dtm)

json_lda <- createJSON(phi = phi, theta = theta, vocab = vocab, doc.length =
doc.length, term.frequency = term.frequency)

library(servr)

serVis(json_lda)
```

Figura 7

Visualización de los tópicos con el paquete LDAVis



Fuente: elaboración propia a partir de los datos del caso de estudio. Visualización ejecutada con código de Holster (2022)

En la Figura 7, podemos ver a la izquierda que los catorce tópicos han sido repartidos en un espacio bidimensional. Si bien es verdad que algunos tópicos se solapan ligeramente, el gráfico indica que los tópicos son efectivamente distintos entre sí. Por tanto, se ha logrado crear tópicos relevantes que puedan expresar ideas diferentes y múltiples sobre el metaverso. Además, el tamaño similar de los círculos de los tópicos nos indica que la prevalencia de cada uno es más o menos similar, siendo por tanto todos los tópicos prácticamente igual de relevantes.

3.6 Fase 5: Posibles usos de los tópicos encontrados

Tras la interpretación y visualización de los tópicos, resulta interesante ir más allá en el análisis e identificar más usos de los tópicos encontrados. En este trabajo se presentan tres posibles usos de estos tópicos: (1) la importancia de los tópicos en cada documento,

(2) la importancia de cada tópico dentro del corpus total y, finalmente, (3) la evolución de la relevancia de cada tópico a lo largo del tiempo.

Para evaluar la importancia de los tópicos en cada documento, se ha procedido a realizar un *heatmap* mediante el cual, a mayor intensidad de color, mayor probabilidad de que el tópico esté presente en un documento (véase Figura 8). En efecto, el gráfico se ha realizado con la distribución (θ) de los tópicos dentro de cada documento. A simple vista, podemos observar en la Figura 8 que, por ejemplo, un grupo de artículos o documentos del final del corpus mencionan el metaverso dentro de las áreas de los conciertos y eventos, inversiones descentralizadas e inversión y riqueza. Al mismo tiempo, se puede analizar el gráfico de la Figura 8 por filas. Por ejemplo, se puede observar que el tópico sobre inversión y riqueza es más o menos relevante en gran parte de los documentos. El código utilizado para realizar este *heatmap* aparece abajo (basado en Niekler y Wiedemann, 2017). Como se puede observar, se ha utilizado el paquete "plotly" (Sievert et al., 2023) con el objetivo de hacer el gráfico interactivo. Gracias a ello, el usuario puede pasar el cursor sobre cualquier documento y ver con qué probabilidad aparece cada tópico en el documento. La utilidad de este tipo de gráfico es relevante ya que un usuario que quiera ver en qué artículos se menciona especialmente el metaverso dentro de, por ejemplo, el ámbito de la realidad virtual, podrá identificarlos rápidamente.

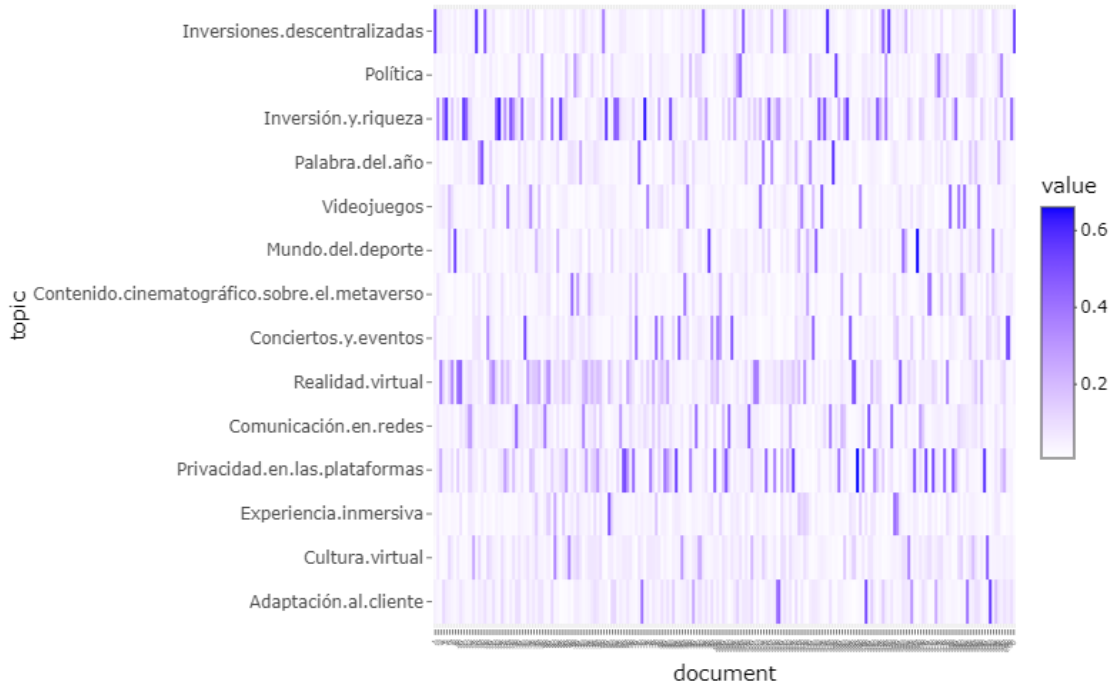
```
library(reshape2)[7]

library(plotly)

body<-GUARDIAN

exampleIds<-1:nrow(body)

N <- length(exampleIds)

topicProportionExamples <- theta[exampleIds,]

colnames(topicProportionExamples) <- topicNames

vizDataFrame <- melt(cbind(data.frame(topicProportionExamples), document =
factor(1:N)), variable.name = "topic", id.vars = "document")
```

[7] (Wickham, 2022)

```
plot<-ggplot(vizDataFrame, aes(document, topic,
fill=value))+geom_tile()+scale_fill_gradient(low="white",
high="blue")+theme(axis.ticks.x = element_blank(), axis.text.x =
element_blank())

ggplotly(plot)
```

Figura 8

Distribución de los tópicos en los documentos



Fuente: elaboración propia a partir de los datos del caso de estudio. Visualización ejecutada con código adaptado de Niekler y Wiedermann (2017)

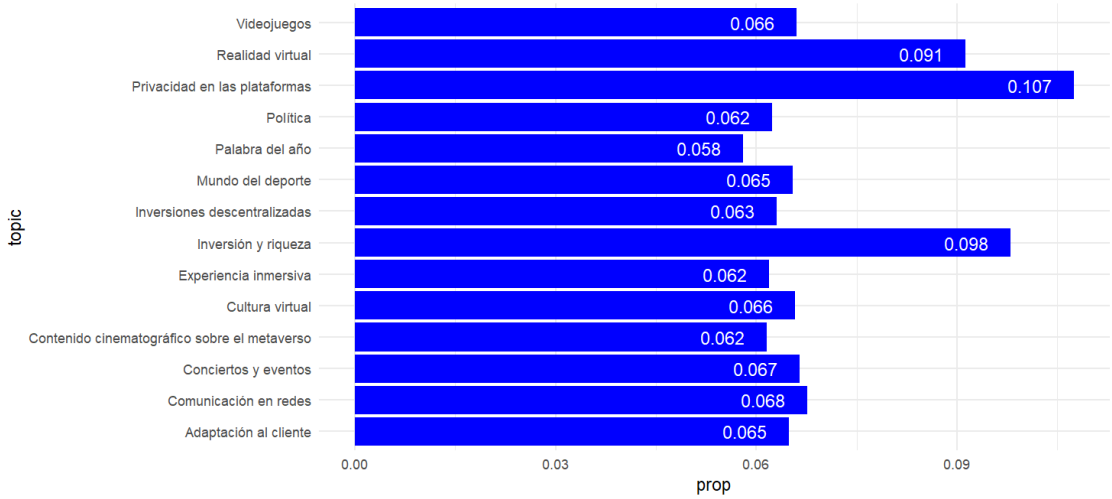
En segundo lugar, podemos también analizar la importancia de los tópicos dentro del corpus entero. Para ver cuáles son los tópicos más frecuentes en el corpus, tendremos primero en cuenta cuáles son las frecuencias de cada uno de ellos dentro de cada documento utilizando “theta”, la distribución a posteriori de los tópicos dentro de los documentos. Aquellos tópicos con mayores frecuencias dentro de todos los documentos serán aquellos que se consideran más relevantes en el corpus entero. Como podemos observar en la Figura 9, el tópico con mayor relevancia dentro de nuestro corpus es el de “Privacidad en las plataformas”, seguido por “Inversión y riqueza” y “Realidad virtual”.

En efecto, estos tres tópicos tienen una probabilidad de aparecer dentro del corpus de 10,7%, 9,8% y 9,1%, respectivamente. En nuestro caso de estudio, esto nos indica que lo más mencionado en las noticias sobre el metaverso tiene que ver con las preocupaciones alrededor de la privacidad en este nuevo entorno, la riqueza que el metaverso puede llegar a generar con los nuevos métodos de inversión y la inmersión que se puede obtener gracias a la realidad virtual. Viendo la Figura 8, vemos que estos tres tópicos con mayor relevancia dentro del corpus son, evidentemente, también aquellos que aparecen más dentro de los documentos. En efecto, las filas correspondientes a estos tres tópicos son las que muestran en la Figura 8 una mayor tendencia al color azul. El código que permite visualizar un gráfico de barras con las proporciones de cada tópico es el siguiente (Niekler y Wiedemann, 2017):

```
topicproportions <- colSums(theta) / nrow(dtm)

names(topicproportions) <- topicNames

topicproportions<-sort(topicproportions, decreasing = TRUE)

proportions <- data.frame(topic = names(topicproportions), prop =
as.numeric(topicproportions))

ggplot(proportions, aes(x=prop, y=topic))+geom_bar(stat="identity",
fill="blue")+  geom_text(aes(label = round(prop,3)), hjust =
1.5,color="white") +theme_minimal()
```

Figura 9

Relevancia de cada tópico dentro del corpus



Fuente: elaboración propia a partir de los datos del caso de estudio. Visualización ejecutada con código adaptado de Niekler y Wiedermann (2017)

Finalmente, resulta también importante identificar cómo evoluciona la prevalencia de cada tópico a lo largo del tiempo. De esta forma, conseguiremos ver cómo han evolucionado las noticias sobre el metaverso en estos últimos años, qué tópicos han ganado o perdido relevancia, y qué tópicos son actualmente los más relevantes. Con el siguiente código, podemos obtener la visualización de la Figura 10 para poder ver la evolución de los tópicos semana a semana (código basado en Niekler y Wiedemann, 2017):

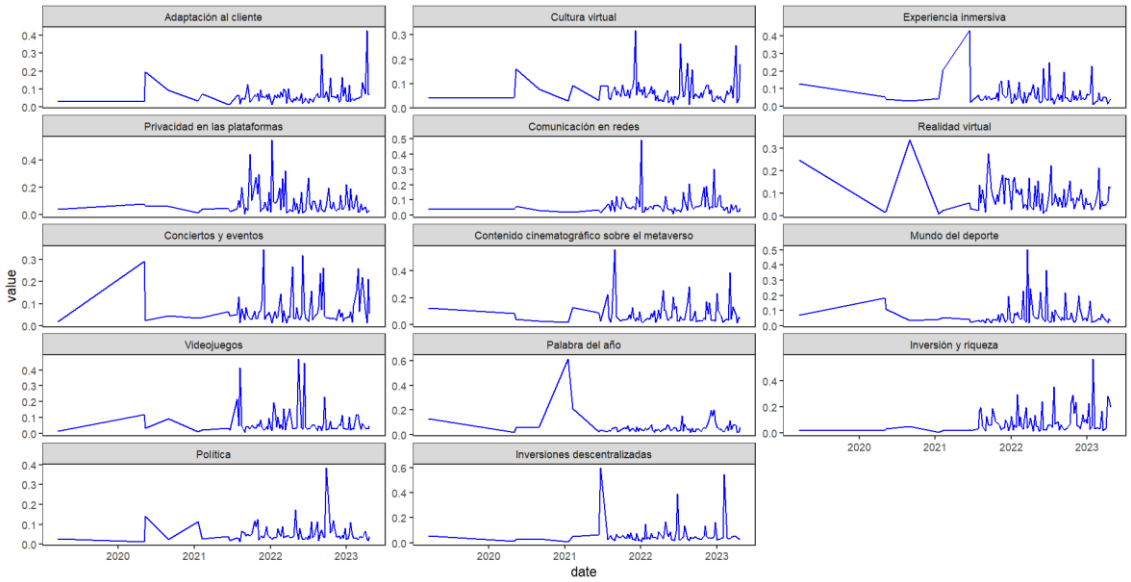
```
library(lubridate)[8]

body$date<-floor_date(body$web_publication_date, unit="week")

topic_proportion <- aggregate(theta, by = list(date = body$date), mean)

colnames(topic_proportion)[2:15] <- topicNames

vizDataFrame <- melt(topic_proportion, id.vars = "date")

ggplot(vizDataFrame, aes(x = date, y = value)) + geom_line(color="blue") +
facet_wrap(~ variable, scales = "free_y", ncol = 3) + theme_bw() +
```

[8] (Spinu et al., 2023)

```
theme(panel.grid.major = element_blank() ,
panel.grid.minor = element_blank())
```

Figura 10

Evolución de la prevalencia de los tópicos semana a semana



Fuente: elaboración propia a partir de los datos del caso de estudio. Visualización ejecutada con código adaptado de Niekler y Wiedermann (2017)

Viendo los artículos semana a semana, podemos ver que la relevancia de cada uno ha ido cambiando a lo largo del tiempo. Cogiendo, por ejemplo, el tópico de "Inversión y riqueza", vemos que empezó a ganar relevancia a finales de 2021 y fue experimentando picos a lo largo de 2022 y 2023. Previamente, se trataba de un tópico que no se mencionaba debido a que todavía se desconocía el metaverso en el ámbito de las inversiones. En las últimas semanas, vemos que el tópico que ha prevalecido ha sido el de "Adaptación al cliente", enfatizando que últimamente se está dando importancia al potencial que ofrece el metaverso de adaptarse a un nuevo tipo de cliente que requiere experiencias de compra distintas en las tiendas y mediante eventos. Por el contrario, el tópico de "Comunicación en redes" parece tener menos prevalencia últimamente, dando paso a temas sobre el metaverso más disruptivos que las redes sociales.

4. Conclusiones generales y posibles extensiones del trabajo

El objetivo general del trabajo era el de crear una guía de implementación del LDA que cubriera todos los aspectos del topic modeling. Este objetivo se ha logrado en el presente trabajo mediante la consecución de los tres objetivos específicos planteados.

En cuanto al primer objetivo específico, se ha logrado a través de la construcción de un marco teórico que profundiza en la técnica del LDA. Esta explicación teórica consigue ofrecer al usuario de esta guía una base teórica fundamental y necesaria para poder posteriormente aplicar el algoritmo. Tratando temas como, por ejemplo, la necesidad de identificar hiperparámetros y el aspecto generativo del algoritmo, se presenta una importante base teórica.

El segundo objetivo específico, consistente en desarrollar paso a paso una guía práctica de implementación en R que aborde todo el proceso de topic modeling, se ha conseguido mediante las cinco fases principales que se han descrito en el tercer capítulo. Se ha logrado explicar paso a paso cómo implementar en R de forma eficiente, ágil y robusta el topic modeling y se han dado respuestas a las distintas cuestiones prácticas que surgen a la hora de aplicar el topic modeling. En efecto, se han tratado temas como la preferencia de utilizar la lematización al stemming y el número óptimo de tópicos a extraer mediante la coherencia.

Finalmente, el último objetivo específico consistía en sugerir ideas para la mayor utilización de los tópicos. Mostrando la distribución de los tópicos dentro de los documentos, la frecuencia de los mismos dentro del corpus y su evolución a lo largo del tiempo, se ha logrado demostrar el potencial de utilización que tiene el topic modeling.

Teniendo en cuenta los límites de extensión del presente trabajo, nuestro estudio cuenta con ciertas limitaciones que podrían resolverse con posibles extensiones del trabajo. En primer lugar, puesto que se han utilizado únicamente artículos periodísticos de *The Guardian*, se propone la inclusión de artículos de otros periódicos para evitar sustentar el análisis en un único periódico en el cual podrían aparecer ciertos niveles de sesgo. En efecto, se podría contemplar la descarga de artículos de *News API* o de *Financial Times*.

De esta forma, se realizaría la extracción de tópicos sobre el metaverso con una base más amplia y variada, pudiendo incluso encontrar tópicos nuevos y distintos. En segundo lugar, se ha utilizado la métrica de coherencia UMass como métrica para determinar el número de tópicos a utilizar. Sin embargo, existen otras métricas, como la denominada "UCI" basada en la PMI, que podrían aportar distintas conclusiones sobre el número óptimo de tópicos (Stevens et al., 2012). Al utilizar varias métricas, podríamos tomar una decisión sobre el número de tópicos a extraer más sustentada y correcta. Finalmente, otra posible extensión del trabajo sería la de realizar un análisis de sentimiento de los distintos tópicos. Tal y como explican Jeong et al. (2019), esto se podría realizar asignando un nivel de sentimiento a cada palabra del corpus. Una vez creados los tópicos, el nivel de sentimiento de un tópico se calcularía según los niveles de sentimiento de las palabras comprendidas dentro del mismo. De esta forma, podríamos entender si lo que se comenta sobre el metaverso en los distintos tópicos tiene un tono positivo, negativo o neutro. Esto nos ayudaría a comprender mejor las opiniones sobre el metaverso para poder tener un entendimiento más completo sobre el mismo.

Bibliografía

Afzali, M., & Kumar, S. (2019). Text Document Clustering: Issues and Challenges. *International Conference on Machine Learning, Big Data, Cloud and Parallel Computing*, 263-268. doi:10.1109/COMITCon.2019.8862247

Ahmad, R., Pervaiz, A., Mannan, H., & Zaffar, F. (2017). Aspect Based Sentiment Analysis for Large Documents. *Twenty-third Americas Conference on Information Systems*, 1-10. doi:https://core.ac.uk/download/pdf/301372008.pdf

Aranda, A., Sele, K., Etchanchu, H., Guyt, J., & Vaara, E. (2021). From Big data to Rich Theory: integrating critical discourse analysis with structural topic modeling. *European Management Review, 18*, 197-214. doi:10.1111/emre.12452

Balakrishnan, V., & Lloyd-Yemoh, E. (agosto de 2014). Stemming and lemmatization: A comparison of retrieval performances. *Lecture Notes on Software Engineering, 2*, 262-267. doi:10.7763/LNSE.2014.V2.134

Benoit, K. (2022). Package "stopwords". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/stopwords/stopwords.pdf

Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Muller, S., . . . Lowe, W. (2023). Package "quanteda". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/quanteda/quanteda.pdf

Blei, D., Ng, A., & Jordan, M. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research, 3*, 993-1022. doi:0.5555/944919.944937

Bouma, G. (2009). Normalized (Pointwise) Mutual Information in Collocation Extraction. *Proceedings of the Biennial GSCL Conference*. Obtenido de https://svn.spraakdata.gu.se/repos/gerlof/pub/www/Docs/npmi-pfd.pdf

Chang, J., Boyd-Graber, J., Gerrish, S., Wang, C., & Blei, D. (2009). Reading Tea Leaves: How Humans Interpret Topic Models. *Neural Information Processing Systems*, 288-296. Obtenido de https://proceedings.neurips.cc/paper/2009/file/f92586a25bb3145facd64ab20fd554ff-Paper.pdf

Chen, Z., Mukherjee, A., Liu, B., Hsu, M., Castellanos, M., & Ghosh, R. (2013). Leveraging Multi-Domain Prior Knowledge in Topic Models. *International Joint Conference on Artificial Intelligence*, 2071-2077. doi:10.5555/2540128.2540426

Church, K., & Hanks, P. (1990). Word Association Norms, Mutual Information, and Lexicography. *Computational Linguistics, 16*(1), 22-29. Obtenido de https://aclanthology.org/J90-1003.pdf

Dahal, B., Kumar, S., & Li, Z. (2019). Topic modeling and sentiment analysis of global climate change tweets. *Social Network Analysis and Mining*, 1-20. doi:10.1007/s13278-019-0568-8

Debortoli, S., Junglas, I., Müller, O., & vom Brocke, J. (2016). Text mining for information systems researchers: an annotated topic modeling tutorial. *Communications of the Association for Information Systems, 39*, 110-135. doi:10.17705/1CAIS.03907

Denny, M., & Spirling, A. (27 de septiembre de 2017). Text Preprocessing For Unsupervised Learning: Why It Matters, When It Misleads, And What To Do About It. (168-189, Ed.) *Political Analysis, 26*. doi:http://dx.doi.org/10.2139/ssrn.28491

Dwivedi, Y., Hughes, L., Baabdullah, A., Ribeiro-Navarrete, S., Giannakis, M., & Al-Debei, M. (2022, julio 16). Metaverse beyond the hype: Multidisciplinary perspectives on emerging challenges, opportunities, and agenda for research, practice and policy. *International Journal of Information Management, 66*. doi:https://doi.org/10.1016/j.ijinfomgt.2022.102542

Ekbal, A., & Saha, S. (2013). Simulated annealing based classifier ensemble techniques: Application to part of speech tagging. *Information Fusion*, 288-300. doi:http://dx.doi.org/10.1016/j.inffus.2012.06.002

Foley, M. (2020). *Topic Modeling*. Obtenido de https://bookdown.org/mpfoley1973/data-sci/

Google Trends. (9 de enero de 2023). *Google Trends for Search Item "Metaverse" and Date Range "1/1/19-12/21/22"*. Obtenido de Google: https://trends.google.com/trends/explore?date=2019-01-01%202022-12-31&q=metaverse

Griffiths, T., & Steyvers, M. (2004). Finding Scientific Topics. *Proceedings of the National Academy of Sciences of the United States of America, 101*(1), 5228-5235. doi:10.1073/pnas.0307752101

Grun, B., & Hornik, K. (marzo de 2023). Package ‘topicmodels’. *Repositorio CRAN*. Obtenido de https://cran.r-project.org/web/packages/topicmodels/topicmodels.pdf

Holster, J. (2022). Analysis, Text. En *Introduction to R for Data Science: A LISA 2020 Guidebook* (p. Bookdown). Obtenido de https://bookdown.org/jdholster1/idsr/

Hornik, K., Meyer, D., & Buchta, C. (2022). Package "slam". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/slam/slam.pdf

Jacobi, C., Atteveldt, W., & Welbers, K. (2015). Quantitative analysis of large amounts of journalistic texts using topic modelling. *Digital Journalism, 4*, 1-18. doi:10.1080/21670811.2015.1093271

Jacobson, D., Brail, G., & Woods, D. (2011). The API Opportunity. In *APIs: A Strategy Guide* (pp. 1-6). Sebastopol, California, Estados Unidos: O'Reilly.

Jeong, B., Yoon, J., & Lee, J.-M. (2019). Social media mining for product planning: a product opportunity mining approach based on topic modeling and sentiment

analysis. *International Journal of Information Management, 48*, 280-290. doi:10.1016/j.ijinfomgt.2017.09.009

Kherwa, P., & Bansal, P. (2019). Topic Modeling: A Comprehensive Review. *EI Endorsed Transactions on Scalable Information Systems, 7*(24), 1-16. doi:10.4108/eai.13-7-2018.159623

Kwartler, T. (2017). Text Sources. En *Text Mining in Practice with R* (págs. 287-290). Wiley.

Liu, Z., Zhang, Y., Chang, E., & Sin, M. (2011). PLDA+: Parallel Latent Dirichlet Allocation with Data Placement and Pipeline Processing. *ACM Transactions on Intelligent Systems and Technology, 2*(3). doi:10.1145/1961189.1961198

Maier, D., Niekler, A., Wiedemann, G., & Stoltenberg, D. (2020). How Document Sampling and Vocabulary Pruning Affect the Results of Topic Models. *Computational Communication Research, 2*(2), 139-152. doi:10.5117/CCR2020.2.001.MAIE

Maier, D., Waldherr, A., Miltner, P., Wiedemann, G., Niekler, A., Keinert, A., . . . Adam, S. (2018). Applying LDA topic modeling in communication research: Toward a valid and reliable methodology. *Communication Methods and Measures: Special Issue on Computational Methods*, *12*, 93-118.

Marlin, B. (2003). Modeling User Rating Profiles for Collaborative Filtering. *Advances in Neural Information Processing Systems*, *16*. Obtenido de https://papers.nips.cc/paper_files/paper/2003/file/269d837afada308dd4aeab28ca2d57e4-Paper.pdf

Meta. (28 de octubre de 2021). *Introducing Meta: A Social Technology Company*. Obtenido de Meta: https://about.fb.com/news/2021/10/facebook-company-is-now-meta/

Mimno, D., Wallach, H., Talley, E., Leenders, M., & McCallum, A. (2011). Optimizing Semantic Coherence in Topic Models. *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, 262–272. Obtenido de https://aclanthology.org/D11-1024.pdf

Newbold, P., Carlson, W., & Thorne, B. (2008). Probabilidad. In *Estadística para administración y economía* (Sexta Edición ed., pp. 102-112). Madrid: Pearson Prentice Hall.

Niekler, A., & Wiedemann, G. (2017). *Tutorial 6: Topic Models*. Obtenido de Topic Distributions: https://nballier.github.io/tm4ss.github.io/Tutorial_6_Topic_Models.html#3_topic_distributions

NLP with R and UDPipe. (2020). Obtenido de Basic Analytical Use Cases I - UDPipe - Basic Analytics: https://bnosac.github.io/udpipe/docs/doc5.html

Nuser, M., & Al-Horani, E. (2020). Medical documents classification using topic modeling. *Indonesian Journal of Electrical Engineering and Computer Science, 17*(3), 1524~1530. doi:10.11591/ijeecs.v17.i3.pp1524-1530

Odell, E. (23 de junio de 2019). Package ‘guardianapi’. *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/guardianapi/guardianapi.pdf

Ooms, J. (6 de diciembre de 2022). Package "jsonlite". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/jsonlite/jsonlite.pdf

Paperno, D., Marelli, M., Tentori, K., & Baroni, M. (2014). Corpus-based estimates of word association predict biases in judgement of word co-occurrence likelihood. *Cognitive Psychology, 74*, 66-83. doi:10.1016/j.cogpsych.2014.07.001.

Park, S.-M., & Kim, Y.-G. (2022 de enero de 2022). A Metaverse: Taxonomy, Components, Applications, and Open Challenges. *IEEE Access, 10*, 4209-4251. doi:10.1109/ACCESS.2021.3140175

Parra, D., Trattner, C., Gómez, D., Hurtado, M., Wen, X., & Lin, Y.-R. (2016). Twitter in academic events: a study of temporal usage, communication, sentimental and topical patterns in 16 computer science conferences. *Computer Communications*, 301-314. doi:10.1016/j.comcom.2015.07.001

Pasquali, A. (2016). Automatic Coherence Evaluation Applied to Topic Models. *Universidad de Oporto*, 21. Obtenido de https://core.ac.uk/download/pdf/143407343.pdf

Pecina, P. (2005). An extensive empirical study of collocation extraction methods. *Proceedings of the ACL Student Research Workshop*, 13-18. doi:10.5555/1628960.1628964

Petrovic, S., Snajder, J., Dalbelo-Basic, B., & Kolar, M. (2006). Comparison of Collocation Extraction Measures for Document Indexing. *28th International Conference on Information Technology Interfaces*, 451-456. doi:10.1109/ITI.2006.1708523

Phelps, J. (2021). All "Spun Up": Findings from Familiar and Unfamiliar Methods. En *Engaging Research Communities in Writing Studies* (págs. 91-113). Routledge.

Ponweiser, M. (2012). *Latent Dirichlet Allocation in R.* WU Vienna University of Economics and Business - Institute for Statistics and Mathematics. Obtenido de https://research.wu.ac.at/ws/files/18975608/main.pdf

Prabha, S., & Sardana, N. (2023). Question Tags or Text for Topic Modeling: Which is better. *Procedia Computer Science*, 2172-2180. doi:10.1016/j.procs.2023.01.193

Reisenbichlet, M., & Reutterer, T. (2018). Topic modeling in marketing: recent advances and research opportunities. *Journal of Business Economics, 89*, 327-356. doi:10.1007/s11573-018-0915-7

Robinson, D., & Silge, J. (2023). Package "tidytext". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/tidytext/tidytext.pdf

Sarkar, D. (2023). Package "lattice". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/lattice/lattice.pdf

Selivanov, D., Bickel, M., & Wang, Q. (2022). Package "text2vec". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/text2vec/text2vec.pdf

Sievert, C., & Shirley, K. (2014). LDAvis: A method for visualizing and interpreting topics. *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*, 63–70. Obtenido de https://nlp.stanford.edu/events/illvi2014/papers/sievert-illvi2014.pdf

Sievert, C., Parmer, C., Hocking, T., Chamberlain, S., Ram, K., Corvellec, M., & Despouy, P. (2023). Package "plotly". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/plotly/plotly.pdf

Silge, J., & Robinson, D. (2017). *Text Mining with R.* O'Reilly.

Spinu, V., Grolemund, G., & Wickham, H. (2023). Package "lubridate". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/lubridate/lubridate.pdf

Stephenson, N. (1992). *Snow Crash.* Nueva York: Bantam Books.

Stevens, K., Kegelmeyer, P., Andrzejewski, D., & Buttler, D. (2012). Exploring Topic Coherence over many models and many topics. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 952-961. Obtenido de https://aclanthology.org/D12-1087.pdf

Straka, M., & Straková, J. (2017). Tokenizing, POS Tagging, Lemmatizing and Parsing UD 2.0 with UDPipe. *CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 88-99. doi:10.18653/v1/K17-3009

Syed, S. (2018). Topic Discovery from Textual Data.

Tholpadi, G., Kanti Das, M., Bansal, T., & Bhattacharyya, C. (2015). Relating Romanized Comments to News Articles by Inferring Multi-Glyphic Topical Correspondence. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 311-317. doi:https://nmrinl.github.io/mrinalkd_aaai2015.pdf

Universal Dependencies. (2022). *Universal POS Tags*. Obtenido de Universal Dependencies: https://universaldependencies.org/u/pos/

van Atteveldt, W., & Welbers, K. (November de 2019). *Github*. Obtenido de Choosing the number of topics (and other parameters) in a topic model: https://github.com/ccs-amsterdam/r-course-material/blob/master/tutorials/R_text_LDA_perplexity.md

van Atteveldt, W., Trilling, D., & Arcila, C. (2022). Text as Data. In W. van Atteveldt, D. Trilling, & C. Arcila, *Computational Analysis of Communication.* Reino Unido: Wiley.

Vangara, R., Bhattarai, M., Skau, E., Chennupati, G., Djidjev, H., Tierney, T., . . . Alexandrov, B. (2021). Finding the Number of Latent Topics with Semantic Non-Negative Matrix Factorization. *IEEE Access, 9*, 117217-117231. doi:10.1109/ACCESS.2021.3106879

Weidmann, N. (2023). Natural Language Processing with Quanteda. En N. Weidmann, *Data Management for Social Scientists* (págs. 175-179). Reino Unido: Cambridge University Press.

Wickham, H. (2022). Package "reshape2". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/reshape2/reshape2.pdf

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Woo, K., . . . Dunnington, D. (2023). Package "ggplot2". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf

Wickham, H., François, R., Henry, L., & Müller, K. (2022, septiembre 1). Package "dplyr". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/dplyr/dplyr.pdf

Wijffels, J. (2022). *Package "udpipe"*. CRAN Repository. Obtenido de https://cran.r-project.org/web/packages/udpipe/udpipe.pdf

Wijffels, J. (2023). UDPipe Natural Language Processing - Topic Modelling Use Cases. *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/udpipe/vignettes/udpipe-usecase-topicmodelling.html

Xie, Y. (2023). Package "servr". *CRAN Repository*. Obtenido de https://cran.r-project.org/web/packages/servr/servr.pdf

Yan, F., Xu, N., & Qi, Y. (2009). Parallel Inference for Latent Dirichlet Allocation on Graphics Processing Units. *Advances in Neural Information Processing Systems, 22*. Obtenido de https://proceedings.neurips.cc/paper/2009/hash/ed265bc903a5a097f61d3ec064d96d2e-Abstract.html

Zeng, Y., Liu, X., Wang, Y., Shen, F., Liu, S., Mojarad, M., . . . Liu, H. (2017). Recommending education materials for diabetic questions using information retrieval approaches. *Journal of Medical Internet Research, 19*(10). doi:10.2196/jmir.7754

Zhong, N., & Schweidel, D. (2020). Capturing changes in social media content: a multiple latent changepoint topic model. *Marketing Science, 39*(4), 827-846. doi:10.1287/mksc.2019.1212

Anexos

Anexo 1: Explicación del valor de la PMI ante una colocación

Si cada vez que aparecen los términos w_1 y w_2 aparecen juntos, estaremos ante un caso de dependencia completa en el que $P(w_1) = P(w_2) = P(w_1, w_2)$. Ante esta situación, la PMI sería igual a (Church & Hanks, 1990):

$$\mathrm{PMI} = \log_2\left(\frac{P(w_1, w_2)}{P(w_1) \cdot P(w_2)}\right) = \log_2\left(\frac{1}{P(w_1)}\right) = \log_2\left(\frac{1}{P(w_2)}\right) = \log_2\left(\frac{1}{P(w_1, w_2)}\right)$$

En este caso, puesto que P(w1) = P(w2) = P(w1, w2) ha de estar entre 0 y 1, la división de 1 entre esta probabilidad forzará a la PMI a ser superior a 0. Cuanto más baja sean estas probabilidades, mayor será la PMI.

Anexo 2: Ejemplo de cálculo de la PMI

En nuestro caso de estudio, como veíamos en la Tabla 4, el bigrama “tik tok” aparece 15 veces, “tik” aparece 15 veces y “tok” aparece 16 veces. Teniendo en cuenta que nuestro corpus está compuesto por 124.455 tokens, llegamos a una PMI de 12,93 de la colocación “tik tok” de la siguiente forma:

$$\mathrm{PMI}(\mathrm{tik},\mathrm{tok}) = \log_2\left(\frac{\mathrm{P}(\mathrm{tik},\mathrm{tok})}{\mathrm{P}(\mathrm{tik})\cdot\mathrm{P}(\mathrm{tok})}\right) = \log_2\left(\frac{\frac{15}{124.455}}{\frac{15}{124.455}\cdot\frac{16}{124.455}}\right)$$

$$= 12{,}93$$

Anexo 3: Código R

```
####FASE 1: RECOLECCIÓN DE DATOS Y CONSTRUCCIÓN DE LA BASE DE DATOS####

library(guardianapi)

Sys.setenv(GU_API_KEY="tu_clave_secreta ")

gu_api_key(check_env = TRUE)

termino <- "metaverse"

fecha_inicio <- "2019-01-01"

fecha_fin <- "2023-04-29"

datos <- gu_content(query = termino, from_date = fecha_inicio, to_date = fecha_fin)

GUARDIAN_subset <- subset(datos, select = c("web_publication_date", "body_text", "type"))

library(dplyr)

GUARDIAN<-filter(GUARDIAN_subset, type== "article")

head(GUARDIAN)

GUARDIAN <-GUARDIAN[!duplicated(GUARDIAN$body_text),]

GUARDIAN <-na.omit(GUARDIAN)


####FASE 2: PREPROCESAMIENTO DEL TEXTO####

#####TOKENIZACIÓN#####

library(udpipe)

ud_model_download <- udpipe_download_model(language = "english")

ud_model <- udpipe_load_model(ud_model_download$file_model)

corpus<-GUARDIAN$body_text

corpus_tokenizado <- udpipe_annotate(ud_model, x = corpus)

corpus_tokenizado_data_frame <- as.data.frame(corpus_tokenizado)

head(corpus_tokenizado_data_frame)


#####TEXTO EN MINÚSCULA Y ELIMINACIÓN DE NÚMEROS Y SÍMBOLOS#####

corpus_tokenizado_data_frame$lemma<-
tolower(corpus_tokenizado_data_frame$lemma)

unique(corpus_tokenizado_data_frame$upos)
```

```
corpus_no_PUNCT_SYM_NUM <- subset(corpus_tokenizado_data_frame, !(upos %in% c("PUNCT", "SYM", "NUM", "X", "PART")))

corpus_int<-subset(corpus_no_PUNCT_SYM_NUM,!grepl("^[&';@[:digit:]]", lemma))

corpus_int$lemma<-gsub("\\-", "", corpus_int$lemma)

corpus_int$lemma<-gsub("\\#", "", corpus_int$lemma)


#####ELIMINACIÓN DE STOPWORDS#####

library(stopwords)

stopwords<-data_stopwords_smart$en

stopwords<-c(stopwords, "meta", "metaverse", "facebook", "year","company", "time", "guardian", "people", "thing",  "life")

corpus_nostopwords<-subset(corpus_int, !lemma %in% c(stopwords))

corpus_nostopwords<-subset(corpus_nostopwords, !token %in% c(stopwords))


#####N-GRAMAS Y COLOCACIONES#####

bigramas <- keywords_collocation(corpus_nostopwords, term = "lemma", group = c("doc_id", "sentence_id"), ngram_max = 2, n_min = 10)

bigramas <- bigramas[bigramas$pmi >= 3,]

bigramas$key <- factor(bigramas$keyword, levels = rev(bigramas$keyword))

library(lattice)

barchart(key ~ pmi, data = head(subset(bigramas, freq>3), 20), col = "blue", main = "Colocaciones con bigramas", xlab = "PMI")

corpus_nostopwords$term <- corpus_nostopwords$lemma

corpus_nostopwords$term<-txt_recode_ngram(corpus_nostopwords$lemma, compound = bigramas$keyword, ngram = bigramas$ngram, sep = " ")


#####ELIMINACIÓN DE LEMAS CON MUY ALTA Y MUY BAJA FRECUENCIA#####

corpus_nostopwords <- corpus_nostopwords[nchar(corpus_nostopwords$term)>3, ]

corpus_nostopwords <- subset(corpus_nostopwords, upos %in% c("NOUN", "ADJ", "PROPN"))

library(quanteda)
```

```
dtf <- document_term_frequencies(corpus_nostopwords, document = "doc_id", term = "term")

dtm <- document_term_matrix(dtf)

dfm_quanteda<-as.dfm(dtm)

dfm_trimmed<-dfm_trim(dfm_quanteda, min_docfreq = 0.005, max_docfreq = 0.99, docfreq_type="prop")

dtm=convert(dfm_trimmed, to="topicmodels")


####FASE 3: TOPIC MODELING MEDIANTE LDA####

#####NÚMERO DE TÓPICOS A EXTRAER#####

library(text2vec)

library(ggplot2)

range<-seq(2, 40, by=2)

tcm=crossprod((as.matrix(dtm)))

coherence_logratio=data.frame()

for(i in range){

 resultsloop_logratio=data.frame()

 for (j in seq(1,5,by=1)){

  topicmodeling <- LDA(dtm, method="Gibbs", k=i,iter=200, burnin=100,control=list(alpha = 50/i, delta=0.1), initialize="random")

  words_topic<-terms(topicmodeling,k=10)

  words_topic_matrix<-as.matrix(words_topic)

  coherenceloop_logratio[j,1]=mean(coherence(words_topic_matrix, tcm,
       metrics = c("mean_logratio"), smooth=1, n_doc_tcm = nrow(tcm)))

 }

 coherence_logratio[i,1]=i

 coherence_logratio[i,2]=mean(coherenceloop_logratio$V1)

}

ggplot() + geom_line(data= coherence_logratio, aes(x = V1, y = V2), color = 'blue', stat="identity", size=1)+labs(x="Número de tópicos", y="Coherencia UMass")+theme_minimal()


#####IMPLEMENTACIÓN DE LDA#####

i=14
```

```
modelo <- LDA(dtm, method = "Gibbs", k =i, control = list(alpha = 50/i, delta=0.1, seed=58), initialize="random")

library(reshape2)

library(tidytext)

phi <- data.frame(as.matrix(posterior(modelo)$terms))

phi$row_name <- (rownames(phi))

topicos <- melt(phi)

colnames(topicos) <- c("topic", "term", "phi")

topicos$topic<-as.numeric(topicos$topic)

top_terms <- topicos %>%
  group_by(topic) %>%
  top_n(15,phi) %>%
  ungroup() %>%
  arrange(topic,-phi)

plot_topic <- top_terms %>%
  mutate(term = reorder_within(term, phi, topic)) %>%
  ggplot(aes(term, phi, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip() +
  scale_x_reordered()

plot_topic


####FASE 4: INTERPRETACIÓN Y VISUALIZACIÓN DE LOS TÓPICOS####

topicNames<-c("Adaptación al cliente", "Cultura virtual", "Experiencia inmersiva", "Privacidad en las plataformas", "Comunicación en redes", "Realidad virtual", "Conciertos y eventos", "Contenido cinematográfico sobre el metaverso", "Mundo del deporte", "Videojuegos", "Palabra del año", "Inversión y riqueza", "Política", "Inversiones descentralizadas")

library(LDAvis)

library(slam)

phi <- as.matrix(posterior(modelo)$terms)

theta <- as.matrix(posterior(modelo)$topics)
```

```
vocab <- colnames(phi)

doc.length <- as.vector(table(dtm$i))

term.frequency <- col_sums(dtm)

json_lda <- createJSON(phi = phi, theta = theta, vocab = vocab, doc.length =
doc.length, term.frequency = term.frequency)

library(servr)

serVis(json_lda)


####FASE 5: POSIBLES USOS DE LOS TÓPICOS ENCONTRADOS####

#####IMPORTANCIA DE LOS TÓPICOS DENTRO DE LOS DOCUMENTOS#####

library(reshape2)

library(plotly)

body<-GUARDIAN

exampleIds<-1:nrow(body)

N <- length(exampleIds)

topicProportionExamples <- theta[exampleIds,]

colnames(topicProportionExamples) <- topicNames

vizDataFrame <- melt(cbind(data.frame(topicProportionExamples), document =
factor(1:N)), variable.name = "topic", id.vars = "document")

plot<-ggplot(vizDataFrame, aes(document, topic,
fill=value))+geom_tile()+scale_fill_gradient(low="white",
high="blue")+theme(axis.ticks.x = element_blank(), axis.text.x = element_blank())

ggplotly(plot)


#####IMPORTANCIA DE LOS TÓPICOS DENTRO DEL CORPUS#####

topicproportions <- colSums(theta) / nrow(dtm)

names(topicproportions) <- topicNames

topicproportions<-sort(topicproportions, decreasing = TRUE)

proportions <- data.frame(topic = names(topicproportions), prop =
as.numeric(topicproportions))

ggplot(proportions, aes(x=prop, y=topic))+geom_bar(stat="identity", fill="blue")+
geom_text(aes(label = round(prop,3)), hjust = 1.5,color="white") +theme_minimal()


#####EVOLUCIÓN DE LOS TÓPICOS A LO LARGO DEL TIEMPO#####
```

```
library(lubridate)

body$date<-floor_date(body$web_publication_date, unit="week")

topic_proportion <- aggregate(theta, by = list(date = body$date), mean)

colnames(topic_proportion)[2:15] <- topicNames

vizDataFrame <- melt(topic_proportion, id.vars = "date")

ggplot(vizDataFrame, aes(x = date, y = value)) + geom_line(color="blue") +
facet_wrap(~ variable, scales = "free_y", ncol = 3) + theme_bw() +
theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank())
```