



ugr

Universidad
de Granada

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA

Uso de herramientas de Text Mining para extraer información asociada al Covid-19

Identificación de temáticas para publicaciones científicas

Autor

Jesús Medina Taboada

Director

Juan Francisco Huete Guadix



ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍAS INFORMÁTICA
Y DE TELECOMUNICACIÓN



DEPARTAMENTO DE CIENCIAS
DE LA COMPUTACIÓN E
INTELIGENCIA ARTIFICIAL

GRANADA, SEPTIEMBRE DE 2021

Uso de herramientas de Text Mining para extraer información asociada al Covid-19: Identificación de temáticas para publicaciones científicas.

Jesús Medina Taboada

Palabras clave: Minería de Textos, Covid-19, Inteligencia Artificial, Agrupamiento de Textos, Publicaciones Científicas.

Resumen

En los tiempos que corren el mundo lleva casi 2 años azotado por una pandemia a escala mundial, conocida como COVID-19 o SARS-CoV-2.

Vivimos en la era del big data e inmensas cantidades de información sobre el tema se publica cada día. Tanta que es imposible digerirla toda.

Según estudios del pasado año 2020, las publicaciones científicas sobre COVID-19 crecían de manera exponencial con el transcurso de los meses llegando a las 160.000 publicaciones a finales de ese año. No hay estudios recientes que documenten el número, pero todos los indicios apuntan a que la cantidad podría haberse multiplicado por diez.

Hasta la fecha, en pos de encontrar factores relacionados con su contagio, su comportamiento y cura, se han realizado estudios de todo tipo y desde distintas perspectivas, dando como resultado publicaciones sobre numerosas temáticas.

En este trabajo se propone utilizar herramientas de text mining para poder identificar esas temáticas partiendo de una base de documentos.

En concreto se entrenará de un modelo Latent Dirichlet Allocation(LDA), mediante el cual se infieren un número de temáticas realizando un agrupamiento de todos los documentos de la base.

El propósito último del proyecto es la creación de una aplicación software a partir del modelo generado, que facilite a los usuarios la tarea de documentación para un proyecto. Permitiendo identificar a qué tema o temas en concreto pertenece una publicación, acción muy conflictiva que se agrava ante la magnitud de documentos que existen.

Use of Text Mining tools to extract information associated with Covid-19: Identification of topics for scientific publications

Jesús Medina Taboada

Keywords: Text Mining, Covid-19, Artificial Intelligence, Text Clustering, Scientific Publications.

Abstract

In this day and age, the world has been plagued by a worldwide pandemic, known as COVID-19 or SARS-CoV-2, for almost 2 years.

We live in the age of big data and immense amounts of information on the subject are published every day. So much that it is impossible to digest it all.

According to studies from last year 2020, scientific publications on COVID-19 grew exponentially over the months reaching 160,000 publications by the end of that year. There are no recent studies documenting the number, but all indications are that the number could have increased tenfold.

To date, in pursuit of finding factors related to its contagion, behavior and cure, studies of all kinds and from different perspectives have been carried out, resulting in publications on numerous topics.

In this work we propose to use text mining tools to identify these topics from a document base.

Specifically, a Latent Dirichlet Allocation (LDA) model will be trained, through which a number of topics will be inferred by grouping all the documents in the database.

The ultimate purpose of the project is the creation of a software application from the generated model, which facilitates the users the task of documentation for a project. It allows to identify to which specific topic or topics a publication belongs, a very conflictive action that is aggravated by the magnitude of existing documents.

Yo, **Jesús Medina Taboada**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77560498J, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jesús Medina Taboada

Granada a Septiembre de 2021 .

D. **Juan Francisco Huete Guadix**, profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Uso de herramientas de Text Mining para extraer información asociada al Covid-19: Identificación de temáticas para publicaciones científicas*, ha sido realizado bajo su supervisión por **Jesús Medina Taboada**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a X de mes de 201 .

Los directores:

Juan Francisco Huete Guadix

Agradecimientos

Quiero empezar dando las gracias a todos los contribuidores a desarrollar las bibliotecas gensim y pyLDAvis que he usado para la implementación del trabajo.

A mi tutor Juan Francisco Huete Guadix por darme la oportunidad en este proyecto y por toda la ayuda y atención a lo largo de su desarrollo.

Quiero agradecer y dedicar este trabajo a todas las personas que aportan su pequeño grano de arena para que el aprendizaje libre, gratis y de calidad sea posible.

Para todos los científicos que han obtenido la base de datos con la que he trabajado. Y a la inestimable labor de todos los profesionales y personas que han trabajado tan duro en la crisis sanitaria.

Para mi amigo Juancho, que ha sido un gran apoyo durante este duro año y sé que lo seguirá siendo.

Índice de figuras

1. Introducción	1
1.1. Motivación y contexto del trabajo	1
1.2. Objetivos	6
1.3. Concepción y planificación	7
1.4. Trabajos anteriores y estado del arte	8
1.5. Línea de desarrollo proyecto	9
1.6. Estimación de costes el proyecto	10
1.6.1. Justificación	11
1.7. Organización de la memoria	11
2. Extracción de información	13
2.1. Base de datos	13
2.2. Análisis de estructura	14
2.3. Preprocesamiento	17
3. Cálculo de las temáticas	21
3.1. Algoritmo LDA:Latent Dirichlet Allocation	21
3.2. Implementación LDA	26
3.2.1. Lematización	26
3.2.2. Listado de términos y stopwords	26
3.2.3. Creación del diccionario	27
3.2.4. Eliminación palabras vacías	27
3.2.5. Creación del corpus	28
3.2.6. Creación del modelo LDA y ajustes	29
3.3. Visualización resultados	30
3.3.1. Datos sobre elección del número de temáticas	32

3.3.2.	Datos sobre la eliminación de palabras frecuentes	33
3.3.3.	Datos sobre interpretación de los términos de una temática	33
4.	Asignación de temáticas	37
4.1.	Asignación de la temática para texto completo	38
4.2.	Asignación de la temática por frases	40
4.2.1.	Calculo probabilidad $p(x t,d)$	41
4.3.	Generación de archivos	45
5.	Desarrollo software	47
5.1.	Análisis	48
5.2.	Diseño	50
5.3.	Servidor web	54
5.4.	Guía usuario	57
5.5.	Estructura del proyecto	57
6.	Conclusiones y vías futuras	65
6.1.	Vías futuras y mejoras	66
6.1.1.	Reflexión personal	68
6.2.	Valoración ética y social	68
	Bibliografía	71

Índice de figuras

1.1. Principales temáticas en publicaciones de 2020 [4]	3
1.2. Número acumulativo de publicaciones por meses en 2020	6
1.3. Pipeline del proyecto [16]	10
2.1. Dataset COVID-19 Open Research. Plataforma Kaggle	14
2.2. Metadata del dataset: Abstract	18
2.3. Archivo JSON modo texto completo	19
2.4. Archivo JSON modo frases	19
3.1. Imagen genérica usada para explicar el funcionamiento de LDA . .	23
3.2. Grafo del modelo LDA [16]	24
3.3. Relación ejemplo y Plate Notation del LDA	25
3.4. Barra de navegación	31
3.5. Comparación en la elección del número de tópicos	32
3.6. Comparación refinamiento del modelo	33
3.7. Comparación palabras relevantes de las temáticas	35
4.1. Relación tópicos-colores	38
4.2. Fórmula cálculo de la probabilidad[5]	42
5.1. Modelo lineal secuencial (Cascada)	48
5.2. Index Page	51
5.3. Document List Page	52
5.4. Document Page	53
5.5. Statistics Page	54
5.6. Levantar web server	58
5.7. Página de inicio	59
5.8. Barra de navegación	59
5.9. Lista de documentos disponibles	60

Índice de figuras

5.10. Visualización de documento completo	60
5.11. Botones funcionales	61
5.12. Visualización de documento por frases	61
5.13. Lista de temáticas	62
5.14. Estadísticas	63

CAPÍTULO 1

Introducción

Como parte inicial de la memoria en este capítulo se pretende exponer de forma clara y concisa el propósito del trabajo llevado a cabo, de forma que el lector pueda formarse una idea básica que le ayude a entender el conjunto del trabajo que va a leer.

El tema del proyecto es el uso de herramientas de Text Mining para extraer información asociada al covid-19, pero, ¿qué es el text mining exactamente? ¿qué herramientas existen? y ¿qué buscamos obtener de ellas en este proyecto?

1.1. Motivación y contexto del trabajo

El Text Mining o Minería de Textos, es una rama de la ciencia de datos formada por un conjunto de técnicas y tecnologías que se utilizan para analizar grandes cantidades de texto, con el fin de inferir información no explícita en el texto. Esta información puede ser tendencias, correlaciones, o reglas que expliquen el comportamiento del texto que se obtienen por medio de estadísticas y algoritmos de búsqueda relacionados a la Inteligencia Artificial.

Dentro del Text Mining existen muchas subramas relacionadas con la informa-

ción que se desea extraer de los documentos, las más relevantes están relacionadas con el procesamiento del lenguaje natural(NPL) cuyo objetivo podría ser extraer datos clave o resúmenes. Sin embargo, la funcionalidad que nosotros aprovecharemos será la de agrupamiento.

Esta técnica permite agrupar textos de acuerdo a diferentes categorías, identificando características en común entre los textos, lo que facilita la búsqueda o navegación entre extensas cantidades.

La característica por la que queremos agrupar la colección de documentos es la temática, es decir, sobre qué trata el documento. Durante el trabajo nos referiremos a ella como temática, tema o tópico.

Si por ejemplo agrupásemos la colección en 3 temáticas obtendríamos algo como, temática 1:"causas del COVID-19", temática 2:"síntomas del COVID-19" temática 3:"la vacuna COVID-19". Podemos usar de ejemplo algunas de las temáticas principales en publicaciones entre enero y noviembre de 2020.

CORONAVIRUS PAPER TOPICS

After an early focus on modelling the spread of the pandemic, researchers are now turning to other topics, an analysis of PubMed papers and preprints suggests.

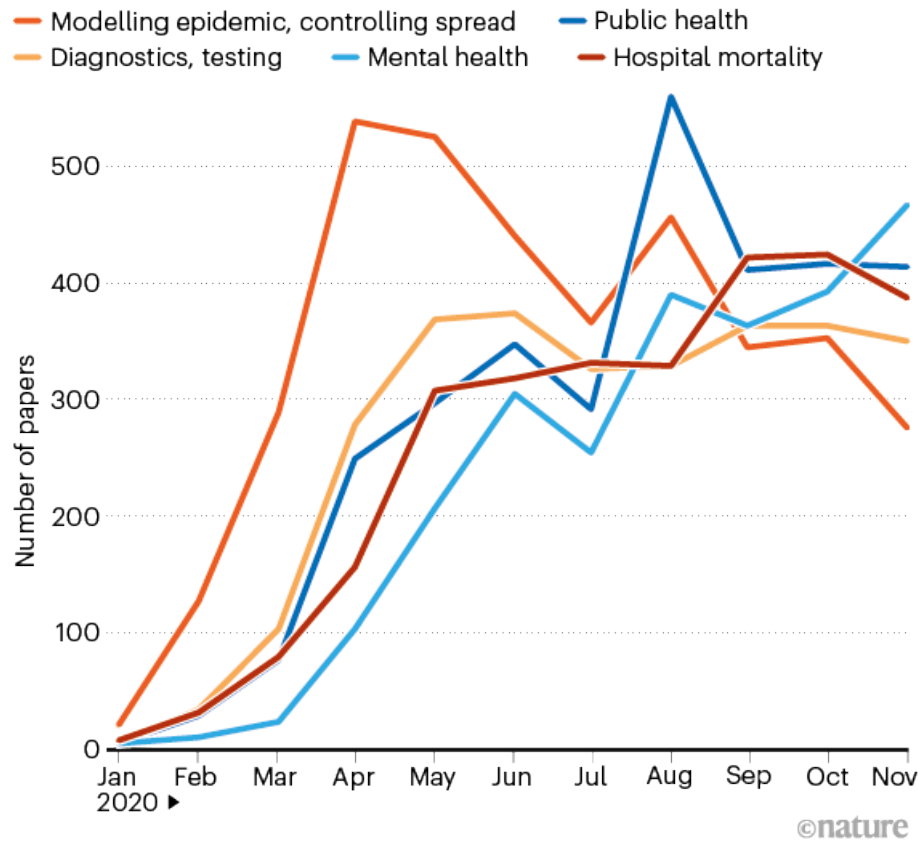


Figura 1.1: Principales temáticas en publicaciones de 2020 [4]

Esta agrupación la llevaremos a cabo usando el algoritmo LDA (Latent Dirichlet Allocation) el cual utiliza la relevancia de los términos que conforman el documento para descubrir las temáticas.

Una vez determinadas las distintas temáticas de la colección, realizaremos un proceso de asignación individual a cada documento dándole la temática a la que pertenece. Para luego dar un paso más y asignar la temática a la que pertenece cada frase del texto.

En este proceso podemos ver indicios de 2 técnicas básicas de la inteligencia

artificial, el clustering o agrupación y la clasificación, aunque esta última realmente no puede entenderse bajo la definición de la técnica de inteligencia artificial, sino más bien como un proceso de etiquetamiento manual.

El objetivo final del trabajo es, una vez hecha la asignación de temáticas, diseñar una aplicación web donde podremos visualizar los documentos y a qué temática corresponden a través de colores, de forma que a cada temática del modelo le corresponde un color. Temática1-rojo, temática2-azul, temática3-verde etc.

Como hemos dicho los documentos que usaremos son publicaciones sobre el COVID-19 pero antes de continuar pongámonos un poco en contexto.

La pandemia de COVID-19 (conocida popularmente como pandemia de coronavirus o simplemente como el coronavirus) a septiembre de 2021 es una pandemia mundial y actualmente en curso derivada de la enfermedad ocasionada por el virus SARS-CoV-2.

Su primer caso fue identificado en diciembre de 2019 en la ciudad de Wuhan, China. La Organización Mundial de la Salud (OMS) la reconoció como una pandemia el 11 de marzo de 2020 (cuando informó que había 4291 muertos y 118.000 casos en 114 países)

A día 9 de julio de 2021, se ha informado de más de 186.8 millones de casos de la enfermedad en 258 países y territorios en el mundo, y 4.035.527 de fallecidos.[15]

Si juntamos estas dos partes encontramos un término curioso, la cienciometría o bibliometría que consiste en el análisis cuantitativo de la producción científica (en especial los artículos científicos), para investigar el desarrollo, estructura, dinámica, tendencias y relaciones de la práctica científica. Su desarrollo es uno de los efectos de la revolución digital en la ciencia, su uso en la medicina ha aumentado y las aplicaciones bibliométricas en el área médica pueden clasificarse en cinco categorías: recuperación de literatura, obtención de nuevo conocimiento, revisiones bibliográficas, análisis de las ciencias médicas, evaluación, gestión y política para medicina [2]

Nosotros nos situamos en la obtención de nuevo conocimiento, las temáticas. Sobre ellas gira todo el propósito de este proyecto, donde queremos conseguir una herramienta software para ayudar a discernir entre la gran cantidad de publicaciones que existen sobre el tema. Ayudando a las personas que deseen buscar publicaciones sobre un temática en concreto. Facilitando labores de búsqueda de referencias y documentación de estudios.

Hay un porcentaje en todo texto conocido como uso del lenguaje, donde se usan términos conectores y palabras frecuentes que realmente no nos transmiten ninguna información. Una de las aplicaciones de nuestro proyecto sería ayudar a identificar estos fragmentos para evitarlos y poder realizar una síntesis mas eficiente del texto. Por lo tanto una de las principales motivaciones que nos ha traído hasta aquí es la ingestionable cantidad de publicaciones que existen a día de hoy y que siguen aumentando. Si actualmente una persona se propone buscar publicaciones acerca de un campo concreto del covid, no tiene más herramientas que su juicio a la hora de leer cada uno de los resúmenes de los que trata una publicación, tarea tediosa y que consume una cantidad de tiempo de la que no se dispone.

Desde que se iniciara la pandemia allá por finales del año 2019, profesionales de todo el mundo y expertos han puesto su conocimiento a servicio de la sociedad realizando estudios y publicando artículos de libre acceso.

Según un estudio realizado en diciembre de 2020 [4] tras 10 meses de pandemia los artículos habían ido experimentando un incremento meteórico alcanzando los 200.000 publicaciones.

De eso hace ya casi un año y a día de hoy no se conoce la cifra exacta pero podría haberse multiplicado por 5 en este periodo. La base de datos que usaremos cuenta en su totalidad actualmente con más de 500.000 publicaciones recogidas.

CORONAVIRUS CASCADE

One estimate suggests that more than 200,000 coronavirus-related journal articles and preprints had been published by early December.

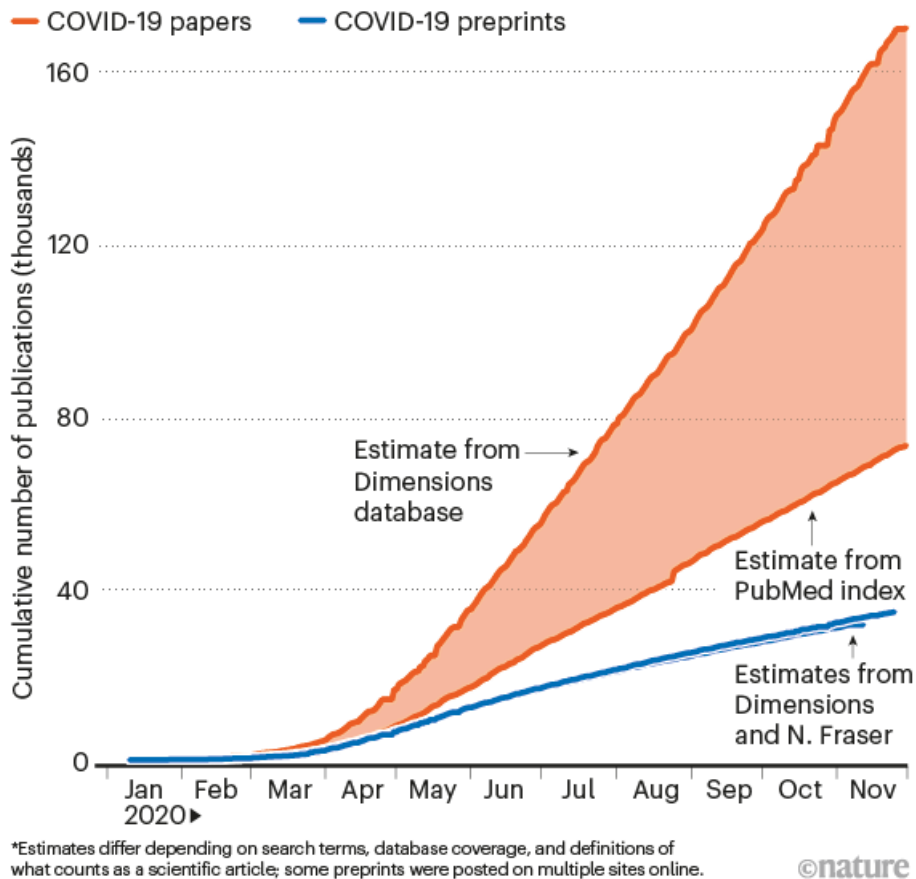


Figura 1.2: Número acumulativo de publicaciones por meses en 2020

1.2. Objetivos

- **Objetivo principal:** desarrollo de una aplicación software para el visionado de documentos por temáticas

La funcionalidad principal que ofrece el proyecto como se ha introducido, es la de ofrecer un servicio web para la visualización de publicaciones científicas aportando una visión informativa de cómo se organiza la publicación según su temática. A partir de este punto, se despliegan objetivos iniciales que la

aplicación debe cumplir:

1. Listado de todas las publicaciones disponibles
 2. Correcto visionado de los documentos
 3. Funcionalidad de intercambio de modo de visionado de temáticas: Texto Completo / Frases
 4. Representación correcta de los datos obtenidos
 5. Incorporación de gráficas dinámicas
 6. Elementos visuales para la comprensión de los datos(leyenda de temáticas)
 7. Diseño intuitivo y elegante
 8. Rapidez en la respuesta y sin fallos
- **Objetivos del modelo:** desarrollo y puesta en marcha de un modelo que usa técnicas de text mining para inferir información de una base de datos.
 1. Correcto análisis y procesado de la base de datos
 2. Comprensión profunda del funcionamiento del algoritmo utilizado(LDA)
 3. Calculo preciso de los parámetros del modelo para su refinamiento
 4. Obtención fiable y representativa del modelo LDA mediante ajustes y pruebas
 - **Objetivos personales:**
 1. Aprender y estudiar sobre el campo del text mining y sus herramientas
 2. Obtener experiencia tratando con grandes bases de datos
 3. Incrementar mis conocimientos sobre tecnologías de desarrollo web
 4. Trasladar los datos obtenidos de un modelado complejo de datos a una aplicación software visualmente interactiva
 5. Mejorar mi habilidad de redacción y documentación de un proyecto complejo

1.3. Concepción y planificación

Cuando solicité este proyecto pensé desde un principio que era la oportunidad de poner en práctica los conocimientos aprendidos en el grado y en especial a los de mi mención de computación sistemas inteligentes, a la vez que podría aplicarlos sobre un tema tan relevante en los días que vivimos el cual se aleja un poco de otros temas más teóricos y que a menudo distan de una realidad cercana, haciendo

que se pierda el interés sobre los resultados.

Realmente el data mining y en concreto el text mining es un área de la computación que no había estudiado en la carrera, pero por la cual había desarrollado interés. Sin ir más lejos de haber leído un poco al respecto, antes de comenzar me encontraba con un tema totalmente desconocido para mí.

Al inicio no podía imaginar la complejidad que el trabajo podría llegar a abarcar, ya que el tema del text mining podía enfocarse desde muchas perspectivas.

Al principio junto al tutor, se plantearon varias formas de trabajar con la base de datos de documentos covid-19 y qué técnicas usar. Debatimos cómo atacar el objetivo principal y decidimos enfocarlo hacia el concepto de temáticas.

Concretamos 2 posibles enfoques sobre cómo tratar los documentos: uno en el que los clasificábamos y otro en el que los agrupábamos (clustering).

Tras esto optamos por coger lo mejor de cada técnica dividiendo el proceso en 3 etapas:

1. Etapla agrupamiento: consiste en agrupar por temáticas los documentos a partir de los términos del texto, las cuales se infieren usando el algoritmo LDA.
2. Etapla clasificación: con las temáticas obtenidas, determinamos a partir del modelo generado mediante un estudio probabilístico, a qué temática pertenece el documento, y dentro de este, a qué temática pertenecen cada una de las frases que lo componen. Entendiendo como pertenecer la temática más probable.
3. Desarrollo aplicación software: confección de un visor de documentos donde presentar los resultados, mostrando con colores la temática a la que pertenece un documento a 2 granularidades distintas, frase y texto completo.

Respecto al diseño del visor de documentos definimos algunos puntos relevantes como la representación visual colores/temáticas y qué había que hacer énfasis en la funcionalidad de poder cambiar el modo de visualización de documento entre texto completo y frases.

1.4. Trabajos anteriores y estado del arte

Aunque he estudiado la rama de computación y sistemas inteligentes, el campo del text mining no se ha llegado a ver. Sin embargo mis numerosos trabajos en

otras asignaturas como aprendizaje automático o visión por computador me han ayudado a comprender con mayor facilidad los conceptos de este campo. También me ayudó la asignatura desarrollo de aplicaciones web donde realicé un software de gestión de una biblioteca, y gracias a la cual he podido crear la aplicación que presento. Aunque mi experiencia en este campo es escasa creo que he realizado un buen trabajo.

Cuanto realizaba la fase de estudio de la base de datos en la plataforma Kaggle, la cual explicaré más tarde, tuve la oportunidad de leer varios trabajos que se realizaron a partir de la misma base de datos.

En esta plataforma personas de todo el mundo comparten sus proyectos y estudios y fue una buena primera toma de contacto, aunque no encontré ningún proyecto sobre agrupamiento de temáticas.

Sin embargo actualmente el uso del algoritmo LDA es lo último en temáticas asociadas a text mining y se pueden encontrar numerosas implementaciones de como usarlo por internet.

Aparte hay algunos papers muy interesantes que tratan el tema de las temáticas y la visualización de datos obtenidos del modelo LDA que me han ayudado a realizar el trabajo. Principalmente han sido 3:

LDAvis: A method for visualizing and interpreting topics[3] y *Termite: Visualization Techniques for Assessing Textual Topic Models*[1] que tratan sobre la interpretación visual del modelo LDA.

LDA-based term profiles for expert finding in a political setting[5] que trata sobre la subdivisión de documentos y como calcular sus temáticas.

Por último añadir que tras una exhaustiva búsqueda no encontré ninguna aplicación web que plantease los objetivos y funciones de este trabajo como la que he desarrollado. En text mining la mayoría de aplicaciones son sobre resumir y identificar sentimientos de los documentos.

1.5. Línea de desarrollo proyecto

En un plano de abstracción un poco mayor, podríamos ver el proceso que recorren los documentos científicos, desde que se obtienen hasta que realizamos su clasificación, de la siguiente manera.

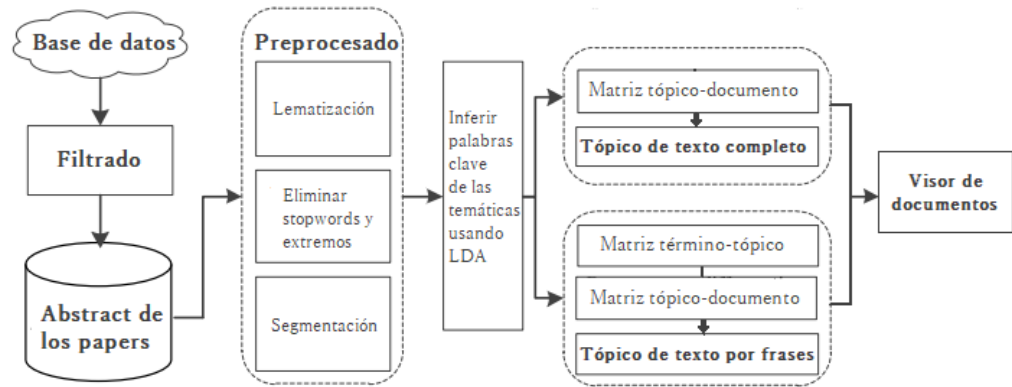


Figura 1.3: Pipeline del proyecto [16]

1. Descarga de la base de datos
2. Análisis de su estructura y extracción de información relevante (abstract)
3. Técnicas de preprocesamiento para transformar en entrada de algoritmo LDA
4. Cálculo de parámetros y confección del modelo LDA
5. Asignación de temáticas modo 'Texto Completo'
6. Asignación de temáticas modo 'Frases'
7. Presentación visual e interactiva de resultados en aplicación web

Cada uno de estos pasos se desarrollarán en profundidad a lo largo de la memoria.

1.6. Estimación de costes el proyecto

A continuación se va a realizar una estimación de los costes del proyecto. Para ello se tendrán en cuenta varios factores como las horas de trabajo empleadas, las infraestructuras usadas o equipo usado en el desarrollo.

Factores	Coste	Total
Ordenador	1500€	1500€
Espacio de trabajo(3 meses)	120€/mes	360€
Trabajo autónomo realizado(3 meses)	14€/hora	5460€
Bibliografía	150€	150€
COSTE TOTAL	-	7.770€

1.6.1. Justificación

- **Equipo usado:** ordenador con prestaciones de gama media-alta para la gran carga de computo que necesita la generación del modelo. Además de un segundo monitor para desarrollo del proyecto. Todo ello con un precio que ronda los 1500€.
- **Espacio de trabajo:** se supone que no se dispone de un espacio adecuado para trabajar y se requiere el alquiler de un estudio con comodidades como internet, electricidad, agua, etc. rondando los 120€ mensuales.
- **Trabajo autónomo:** se ha trabajado durante 3 meses, suponemos una estimación de una jornada de 6 horas al día de lunes a viernes, con un total de 65 días de trabajo dando un total de 390 horas de trabajo autónomo. El salario de un ingeniero informático Junior ronda los 14€ por lo que obtenemos un coste de 5.460€.
- **Bibliografía:** se requiere documentar fuentes privadas que no están en internet generando un coste adicional de 150€.

El coste total del proyecto asciende a **7.470,00 €**.

1.7. Organización de la memoria

La memoria se divide en un total de 6 capítulos:

- **Introducción:** el capítulo actual, donde se describe globalmente el proyecto para que el usuario tome una idea general antes de profundizar más.
- **Extracción de información:** explicamos la base de datos que usaremos, su estructura y filtrado.
- **Cálculo de las temáticas:** fase en la que se explica como funciona el algoritmo LDA y como realizamos el modelo para extraer las temáticas.

- **Asignación de las temáticas:** en esta parte explicamos los métodos que usamos para asignar las temáticas a los documentos en sus 2 modos.
- **Desarrollo software:** capítulo que documenta todo el proceso de desarrollo de la aplicación web, desde su análisis y diseño hasta su implementación.
- **Conclusiones:** capítulo que cierra la memoria donde vemos si hemos cumplido los objetivos y las conclusiones a las que hemos llegado.

Al final del todo encontramos la bibliografía del trabajo.

Esta memoria ha sido escrita en LaTeX a partir de la plantilla estándar para trabajos de fin de grado que ofrece la Universidad de Granada.

CAPÍTULO 2

Extracción de información

La primera etapa del trabajo consiste en obtener la información que usaremos para realizar el estudio. Puesto que esta información que necesitamos son documentos, necesitamos una base muy numerosa de estos, fiable y bien estructurada. El proceso comenzará descargando la base de datos de internet. Hecho esto, se llevará a cabo un proceso de análisis de la estructura que compone cada archivo para finalmente realizar un preprocesamiento donde se decide qué partes de la información usaremos y cuales descartaremos.

2.1. Base de datos

Para descargar la base de datos nos valdremos de **Kaggle** que es una plataforma subsidiaria de Google LLC donde podemos encontrar una comunidad en línea de científicos de datos y profesionales del aprendizaje automático.[12] En ella se encuentran miles de bases de datos listas para descargar y al alcance de cualquier persona que desee trabajar con ella. A partir de estas, se crean los *'challenges'* que son competiciones que invitan a los usuarios a realizar estudios y pruebas con el dataset además de incentivos económicos. En concreto haremos uso de la base de datos **COVID-19 Open Research**[10], que es un recurso de más de 500.000 artículos académicos, en su mayoría escritos

Capítulo 2. Extracción de información

en inglés, y que incluye más de 200.000 con texto completo sobre COVID-19 o SARS-CoV-2.

Este conjunto de datos de libre acceso, se pone a disposición de la comunidad investigadora a nivel mundial para que se apliquen los recientes avances en el procesamiento del lenguaje natural y otras técnicas de inteligencia artificial, con el fin de generar nuevos conocimientos en apoyo de la lucha actual contra el virus.

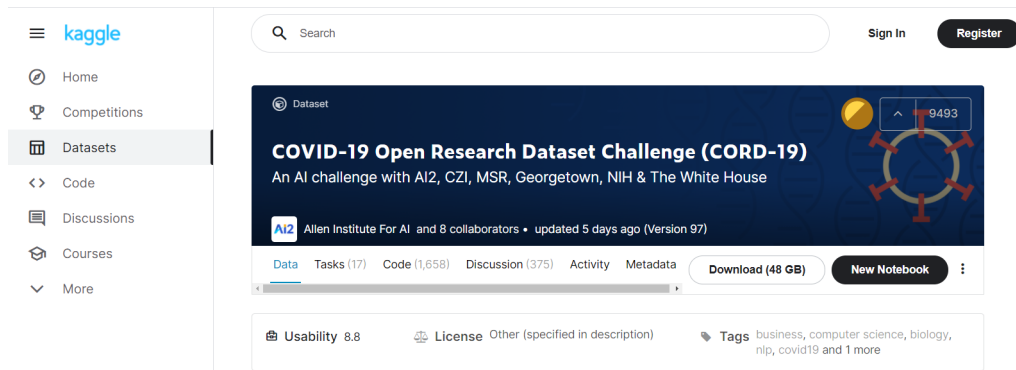


Figura 2.1: Dataset COVID-19 Open Research. Plataforma Kaggle

Este conjunto de datos ha sido creado por el *Allen Institute for AI* en colaboración con la *Chan Zuckerberg Initiative*, el *Center for Security and Emerging Technology* de la Universidad de Georgetown, Microsoft Research, IBM y la *National Library of Medicine - National Institutes of Health*, en coordinación con la Oficina de Política Científica y Tecnológica de la Casa Blanca.

2.2. Análisis de estructura

El análisis de la estructura que conforman los documentos con los que vamos a trabajar es crucial. Estos se encuentran en formato JSON, que es un formato de texto sencillo para el intercambio de datos, y están estructurados de forma que nos aportan muchísima información de todo tipo sobre el artículo: como título, identificador del artículo, fecha de publicación, autores o referencias entre otros. En nuestro caso lo más importante y sobre lo que girará todo nuestro procedimiento será el texto del artículo.

```
1 # JSON schema of full text documents
2
3 {
4     "paper_id": <str>,
5     "metadata": {
6         "title": <str>,
7         "authors": [
8             {
9                 "first": <str>,
10                "middle": <list of str>,
11                "last": <str>,
12                "suffix": <str>,
13                "affiliation": <dict>,
14                "email": <str>
15            },
16            ...
17        ],
18        "abstract": [
19            {
20                "text": <str>,
21                "cite_spans": [
22                    {
23                        "start": 151,
24                        "end": 154,
25                        "text": "[7]",
26                        "ref_id": "BIBREF3"
27                    },
28                    ...
29                ],
30                "ref_spans": <list of dicts>,
31                "section": "Abstract"
32            },
33            ...
34        ],
35        "body_text": [
36            {
37                "text": <str>,
38                "cite_spans": [],
39                "ref_spans": [],
40                "eq_spans": [],
41                "section": "Introduction"
42            },
43            ...
44            {
45                ...,
46                "section": "Conclusion"
47            }
48        ],
```

```
49     "bib_entries": {
50         "BIBREF0": {
51             "ref_id": <str>,
52             "title": <str>,
53             "authors": <list of dict>
54             "year": <int>,
55             "venue": <str>,
56             "volume": <str>,
57             "issn": <str>,
58             "pages": <str>,
59             "other_ids": {
60                 "DOI": [
61                     <str>
62                 ]
63             }
64         },
65         "BIBREF1": {},
66         ...
67         "BIBREF25": {}
68     },
69     "ref_entries":
70         "FIGREF0": {
71             "text": <str>,
72             "type": "figure"
73         },
74         ...
75         "TABREF13": {
76             "text": <str>,
77             "type": "table"
78         }
79     },
80     "back_matter": <list of dict>
81 }
82 }
83 }
```

Principalmente hay 2 secciones de texto en el archivo JSON y sobre las que podríamos aplicar el algoritmo LDA.

El apartado **abstract** y el **body_text**, que serían el resumen del artículo y el contenido del artículo respectivamente.

Tras una evaluación, teniendo en cuenta la gran cantidad de información y las técnicas que se emplearían, se concluyó que la sección abstract sería suficiente para el proyecto, teniendo ya que tiene una longitud significativamente reducida, reduciría el tiempo de computación.

Por una parte el modelo podría verse perjudicado por la falta de información,

no obstante, se determinó que al usar el abstract ,que no es otra cosa que un resumen del texto principal, ganaríamos en compactación del significado global del documento. Que a resumidas cuentas comparte un gran porcentaje del objetivo del trabajo: determinar la temática del documento, es decir, de qué trata.

Esto se traduce en el modelo, a una reducción del diccionario total y un incremento en palabras clave que ayudan a un mejor agrupamiento, como se explica en el capítulo 3.

Una vez decidido que este era el campo más importante y el cual extraeríamos, se decide conservar también el identificativo único del documento **paper_id** y el título del documento **title** (aunque este campo en algunos documentos se encuentra vacío).

El resultado final y de forma organizada queda así:

```
1 # JSON esquema del documento filtrado
2
3 {
4     "paper_id": "",
5     "title": "",
6     "abstract": [
7         {}
8     ]
9 }
```

2.3. Preprocesamiento

En este paso se busca obtener, como se acaba de mostrar, un filtrado de toda la información que el documento nos proporciona, almacenando únicamente lo esencial.

Esto, aparte de una mayor rapidez computacional, nos brinda una eficiencia en almacenamiento.

La plataforma Kaggle permite a sus usuarios trabajar directamente desde la página importando la base de datos directamente de sus servidores. Debido a esto en los inicios del proyecto se planteó realizar toda la programación en la plataforma. Pero debido a su lentitud entre otros inconvenientes, se buscaron alternativas como Jupyter notebook o nootbok en colab.

Finalmente por comodidad decidí descargar directamente la base de datos y trabajar en mi computador.

Capítulo 2. Extracción de información

La base de datos total descargada serían de **234.004** archivos en formato JSON con un peso de **20,3 GB**.

El proceso que se sigue para el filtrado es simple. Para cada archivo de la base de datos se lee el formato JSON con ayuda de la librería homónima y se acceden a los atributos que se desean: `paper_id`, `title` y `abstract`.

Los 2 primeros se copian directamente, mientras que al tercero tendremos que trabajarlo un poco más.

El abstract en el archivo viene dado por una lista de diccionarios, los cuales a su vez tienen 3 campos: `'text'`, `'cite_spans'` y `'ref_spans'`.

De estos solo nos interesa el campo `text` por lo que nos quedaría una lista con distintos `'text'` que son los párrafos que conforman el abstract. En este punto aplicaremos 2 procedimientos distintos, para obtener por una parte los documentos con texto junto **text** y documentos con texto por frases **phrases**.

En el primer tipo lo que haremos será juntar todos los campos `'text'` en un solo texto e incluirlo en el archivo final, para que posteriormente el algoritmo lo procese en conjunto y le asigne una temática a todo el texto.

Filtro abstract

Al llegar a este punto, debemos parar a mirar la documentación que nos proporciona la base de datos Kaggle. Ya que en el metadata se nos especifica que un **26 %** de los documentos tiene el campo `abstract` vacío.

A abstract

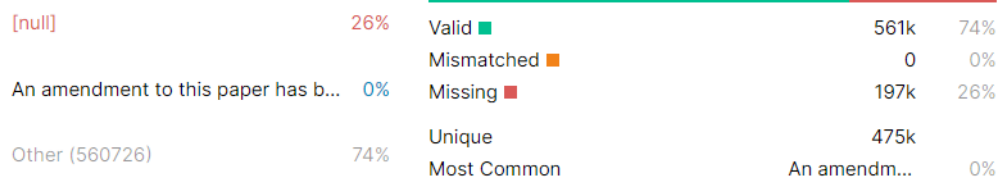


Figura 2.2: Metadata del dataset: Abstract

Esto se traduce a archivos que no aportan nada por lo que hay que realizar una comprobación y descartarlos.

Pero en proceso de ejecución nos percatamos que, no solo es que en algunos documentos el campo abstract esté vacío, sino que en muchos casos está incompleto, es muy corto, incongruente o directamente un número.

También se descubre que muchos abstracts son citaciones de autores y no son realmente resúmenes por lo que no aportan información (Estos abstract comienzan siempre por 'Citation:').

A causa de este factor se toman las siguientes reglas para descartar un abstract:

- Si el texto comienza por citation se descarta.
- Si el texto tiene longitud menor a 150 caracteres se descarta.

Una vez salvada esta complicación nos adentraremos en el segundo tipo de transformación.

En la segunda forma usaremos el texto completo del anterior paso para hacer una división por frases. Usaremos el módulo 'punkt' de la biblioteca NLTK que sirve para detectar automáticamente separaciones en texto por frases, y es mucho más sofisticado que realizar nosotros una búsqueda.

Para acabar se copian todos los campos seleccionados a un nuevo archivo JSON que se guarda, obviando de esta forma el resto de atributos inútiles.

```
{
  "paper_id": "ee84ef4d7c3283914205963d3bef0e0880586fff",
  "title": "A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gr\u00f6bner Bases",
  "abstract": [
    {
      "text": "The solution and a portable implementation of the inverse kinematics computation of a 3 degree-of-freedom (DOF) robot manipulator using Gr\u00f6bner bases are presented. The main system was written Python with computer algebra system SymPy. Gr\u00f6bner bases are computed with computer algebra system Risa/Asir, called from Python via OpenXM infrastructure for communicating mathematical software systems. For solving a system of algebraic equations, several solvers (both symbolic and numerical) are used from Python, and their performance has been compared. Experimental results with different solvers for solving a system of algebraic equations are shown."
    }
  ]
}
```

Figura 2.3: Archivo JSON modo texto completo

```
{
  "paper_id": "ee84ef4d7c3283914205963d3bef0e0880586fff",
  "title": "A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gr\u00f6bner Bases",
  "abstract": [
    {
      "text": "The solution and a portable implementation of the inverse kinematics computation of a 3 degree-of-free"
    },
    {
      "text": "The main system was written Python with computer algebra system SymPy."
    },
    {
      "text": "Gr\u00f6bner bases are computed with computer algebra system Risa/Asir, called from Python via OpenXM"
    },
    {
      "text": "For solving a system of algebraic equations, several solvers (both symbolic and numerical) are used fr"
    },
    {
      "text": "Experimental results with different solvers for solving a system of algebraic equations are shown."
    }
  ]
}
```

Figura 2.4: Archivo JSON modo frases

Algorithm 1 Proceso de preprocesamiento y división de los archivos en text y phrases

```
1: function PREPROCESAMIENTO
2:   for each documento  $\in$  database do
3:     document_text  $\leftarrow$  Dictionary()
4:     document_phrases  $\leftarrow$  Dictionary()
5:     whole_text  $\leftarrow$  String()
6:     phrase_list  $\leftarrow$  List()
7:     document_text += titulo
8:     document_phrases += titulo
9:     document_text += paper_id
10:    document_phrases += paper_id
11:    for each parrafo  $\in$  documento do
12:      whole_text += parrafo
13:    end for
14:    if  $\text{len}(\text{whole\_text}) > 150$  and  $\neg \text{whole\_text.comienza}(\text{"CITATION"})$ 
15:      then
16:        document_text += whole_text
17:        for each frase  $\in$  documentno do
18:          document_phrases += frase
19:        end for
20:        save(document_text)
21:        save(document_phrases)
22:      end if
23:    end for
24:  end function
```

Tras el preprocesamiento se obtiene una base de datos, de **679 MB**, que supone una reducción del **99,67 %** de la base inicial.

Con una dimensión de **309.886** archivos, teniendo en cuenta que los documentos están duplicados para texto completo y texto por frases. Lo que nos dejaría con un total de **154,939** documentos con los que trabajar.

CAPÍTULO 3

Cálculo de las temáticas

Tras la etapa de preprocesamiento procedemos a trabajar con la materia prima de nuestro proyecto, los textos.

Usando todos ellos generaremos un modelo de agrupamiento usando el algoritmo Latent Dirichlet Allocation que relaciona las publicaciones según temáticas que el propio algoritmo infiere.

Sin embargo, para que el algoritmo pueda tomar de entrada el conjunto de textos, aun tenemos que prepararlos un poco más y transformarlos de diferentes formas para que el algoritmo pueda usarlos de entrada.

3.1. Algoritmo LDA: Latent Dirichlet Allocation

El Latent Dichlet Allocation que se podría traducir como Asignación Relevante de Dirichlet es un modelo generativo implementado por David Blei, Andrew Ng y Michael I. Jordan en 2003, que permite que conjuntos de datos puedan ser relacionados en grupos bajo relaciones no observadas.

En este trabajo se usará lo que se conoce como **Descubrimiento de temáticas**, que es un subproblema del procesamiento de lenguaje natural (NLP). En nuestro contexto plantearemos una base de documentos agrupada bajo temáticas no observadas a priori.

El LDA es una generalización del antiguo enfoque del '*Probabilistic latent semantic analysis (pLSA)*'. El modelo **pLSA** es equivalente al LDA bajo una distribución a priori de Dirichlet uniforme.

La **distribución de Dirichlet**, en honor a Peter Gustav Lejeune Dirichlet y que da nombre al algoritmo generalmente denotada $\text{Dir}(\alpha)$ es una familia de distribuciones de probabilidad continuas multivariada, parametrizadas por un vector alfa perteneciente al conjunto de los números reales positivos. Es la generalización multivariada de la distribución beta. Las distribuciones de Dirichlet frecuentemente se utilizan como distribuciones a priori en estadística Bayesiana, y de hecho la distribución de Dirichlet es el conjugado a priori de la distribución categórica y la distribución multinomial. [11]

La **intuición** tras el algoritmo para el modelado de temas consiste en que considera cada documento como una colección de temas en una determinada proporción. Y cada tema como una colección de palabras clave, de nuevo, en una determinada proporción.

Una vez que se le proporciona el **número de temas**, todo lo que hace es reorganizar la distribución de temas dentro de los documentos y la distribución de palabras clave dentro de los temas para obtener una buena composición de la distribución de temas y palabras clave.

Pero cuando hablamos de temática, ¿a qué nos referimos realmente y cómo se representa?
Entenderemos **temática o tópico** en este contexto como una colección de palabras clave relevantes, que son representantes típicos de un tema general.

Sólo con leer las palabras clave, se puede identificar de qué trata el tema. Por ejemplo, una temática inferida por el modelo podría ser el siguiente conjunto de palabras: cachorro, ladrar, hueso y paseo. Que se podría identificar como 'temática relacionada a perros'.

Por lo tanto, se debe tener claro que el algoritmo no genera las temáticas definidas directamente como tal y se requiere de una **interpretación humana** para determinarlas.

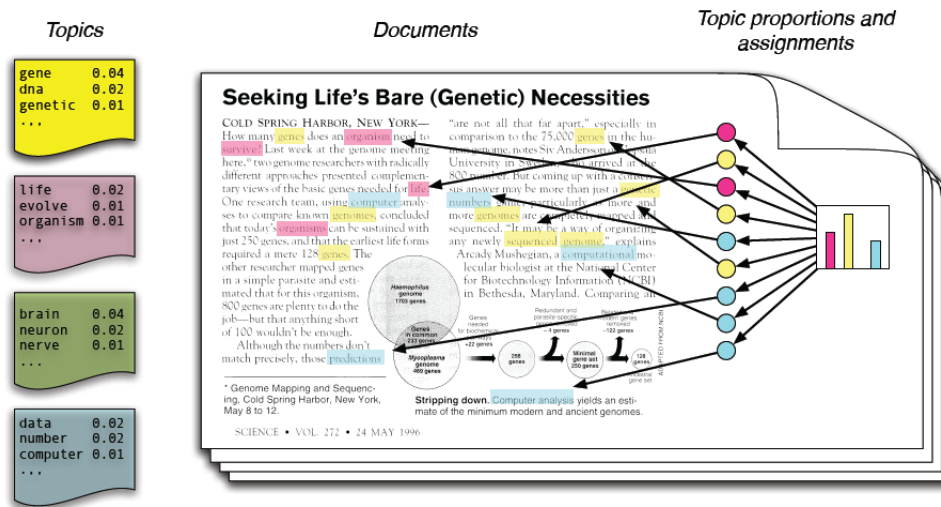


Figura 3.1: Imagen genérica usada para explicar el funcionamiento de LDA

Si la colección de documentos es lo suficientemente grande, el algoritmo descubrirá los temas(conjuntos de términos) basándose en la co-ocurrencia de términos individuales. Aunque la tarea de asignar una etiqueta significativa a un tema individual, dicho en otras palabras, que todos los términos estén relacionados con la bolsa de palabras, depende del usuario. Y esto a menudo requiere conocimientos especializados. Por ejemplo en nuestro proyecto necesitaríamos de un especialista o profesional sanitario.

El enfoque LDA asume que[14]:

1. El contenido semántico de un documento está compuesto por la **combinación** de uno o más términos de uno o más temas.
2. Ciertos términos son **ambiguos**, pertenecen a más de un tema, con diferente probabilidad. (Por ejemplo, el término adiestramiento puede aplicarse tanto a los perros como a los gatos, pero es más probable que se refiera a los perros, ya que se utilizan como animales de trabajo o participan en competiciones de obediencia o habilidad). Sin embargo, en un documento, la presencia conjunta de términos vecinos específicos (que pertenecen a un solo tema) desambiguará su uso.
3. La mayoría de los documentos sólo contienen un número **relativamente pequeño** de temas. En la colección, por ejemplo, los temas individuales aparecerán con distintas frecuencias. Es decir, tienen una distribución de probabilidad, de modo que es más probable que un documento determinado contenga algunos temas que otros.

Capítulo 3. Cálculo de las temáticas

4. Dentro de un tema, algunos términos se utilizarán con **mucha más frecuencia** que otros. En decir, los términos de un tema también tendrán su propia distribución de probabilidad.

Es muy común encontrar el funcionamiento de LDA en '*Notación de Placas*' que se suele usar para representar modelos gráficos probabilísticos (MGP), donde las dependencias entre las numerosas variables pueden captarse de forma concisa.

Las cajas son 'placas' que representan réplicas, las cuales son entidades repetidas. La placa exterior representa los documentos, mientras que la placa interior representa las posiciones de las palabras repetidas en un documento determinado. Cada posición está asociada a una elección de tema y palabra.

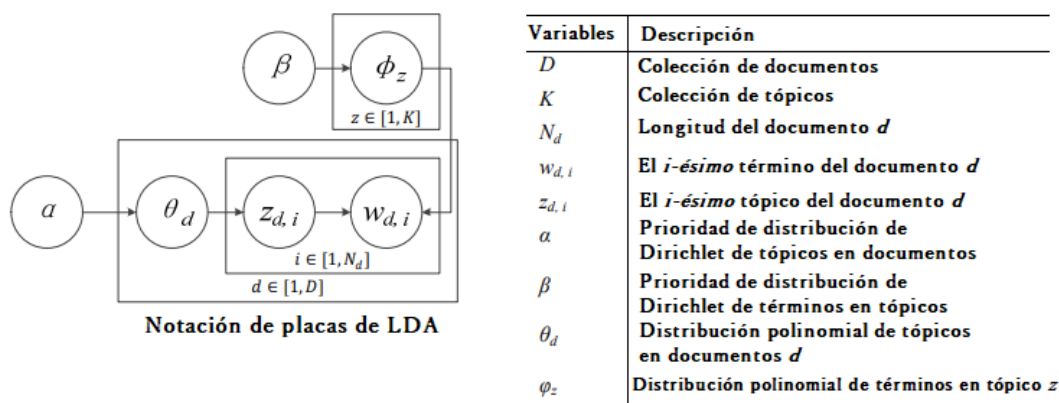


Figura 3.2: Grafo del modelo LDA [16]

A su vez los factores clave para obtener buenos tópicos bien segregados serían:

- La calidad del tratamiento del texto.
- La variedad de temas de los que habla el texto.
- El número de temas introducidos en el algoritmo.
- Los parámetros de ajuste del algoritmo.

Los 2 primeros nos han sido asegurados por la fiable base de datos que usamos y los 2 últimos serán temas a tratar en la implementación.

La clave del Latent Dirichlet Allocation reside en la independencia de los términos respecto de los documentos. Este enfoque conocido como bolsa de palabras enuncia que no importa el orden de los términos. Por lo que un término siempre será visto como parte de un tema, transmitiendo la misma información independiente del lugar que ocupe dentro del documento.

Ejemplificando vendría a decir que el conjunto de términos 'Juan contrató a Luis' es lo mismo que 'Luis contrató a Juan'. En ambos casos el conjunto de palabras y su frecuencia sería el mismo.

Esta propiedad es necesaria para que las probabilidades sean intercambiables permitan una mayor aplicación de métodos matemáticos.

A pesar de que LDA de vez en cuando trata frases semánticamente diferentes como la misma cosa, funciona bien en un documento general.

El modelo interpreta esto como un todo y encuentra las temáticas a partir de esta característica. En el caso de que los documentos se comparasen de forma individual, algunos temas podrían no ser recogidos ya que, solo cuando el cuerpo entero es visto en su conjunto, comienzan a deducirse ciertos tópicos.

Por tanto los términos que aparecen con menos frecuencia en documentos únicos pero que en cambio son comunes en muchos documentos, significan que existe un tema en común entre todos los textos. En nuestro caso como veremos en la implementación, encontraremos varios términos referentes a la índole de la base de datos, el covid-19.[13]

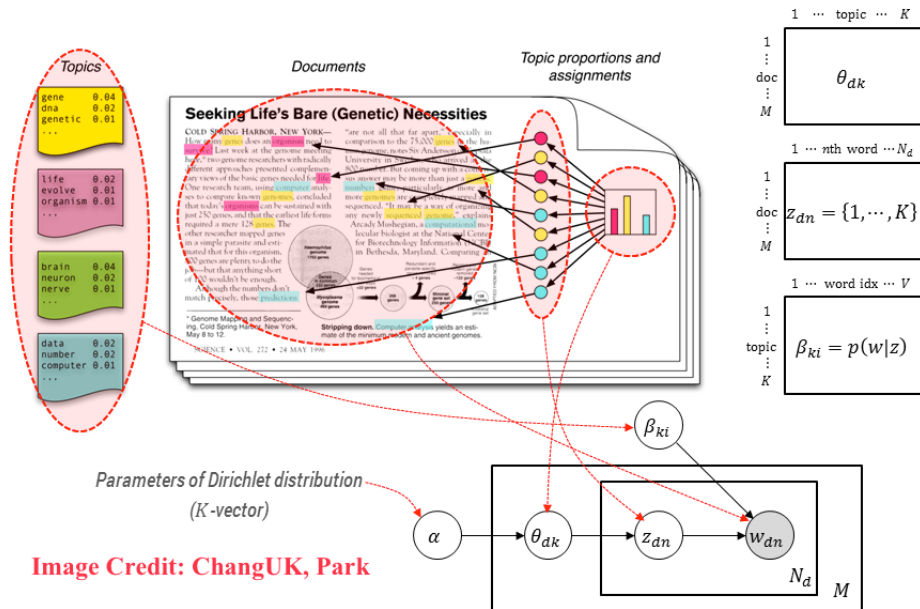


Figura 3.3: Relación ejemplo y Plate Notation del LDA

3.2. Implementación LDA

Realizaremos la implementación del algoritmo en el lenguaje de programación Python. Usaremos principalmente la biblioteca Gensim[9] pero también otras como Spacy, NLTK.

Partiendo de nuestra base de documentos ya filtrada, aún necesitaremos realizar otra etapa de procesamiento de los datos y prepararlos para que el algoritmo pueda tomarlos como entrada en la generación del modelo.

Esta preparación consiste en una serie de transformaciones del texto que se realiza para cada documento.

3.2.1. Lematización

La lematización es el primer proceso y consiste en la transformación de términos(palabras) del texto a una forma simplificada. Podría verse como un proceso en el que nos quedamos con la raíz de las palabras, eliminando las terminaciones, sufijos, prefijos y signos gramaticales, convirtiéndolas en palabras clave.

Por ejemplo todas las formas verbales 'somos', 'soy', 'serás', 'éramos' se lematizarían en el término 'ser'.

En la lematización solo nos quedaremos con los nombres, los adjetivos, los verbos y los adverbios, desechando así conjunciones y determinantes que solo sirven de conectores entre las palabras que realmente dotan de significado a los textos.

La transformación se realiza a partir de un diccionario especializado de el módulo spacy llamado 'en_core_web'[7] que recoge términos clave frecuentes en blogs, noticias, y publicaciones.

Este proceso traduce el texto a los términos con los que el modelo trabajará.

Texto inicial: ['The covid-19, is a very dangerous infectious disease']

Texto lematizado:['covid be very danger infection disease']

Podemos ver por ejemplo como 'is' se transforma en 'be', 'dangerous' en 'danger' y las palabras 'a' y 'the' al ser determinantes desaparecen.

3.2.2. Listado de términos y stopwords

Con la ayuda de la función `simple_preprocess` convertimos el texto de términos lematizados a una lista de términos.

En el proceso de obtención de los términos encontramos que algunas palabras tienen una frecuencia desmesurada en comparación al resto. Son palabras que se usan mucho en el lenguaje, como verbos y adjetivos descriptivos, denominadas 'stopwords' o 'palabras de parada'.

Son términos que aparecerán muchas veces y que no aportarán información relevante al modelo ya que aparecerán en prácticamente todos los documentos. Por lo que debemos eliminarlos. Descargaremos la colección de stopwords en inglés con la ayuda de la biblioteca NLTK y eliminaremos estas palabras de la lista de términos.

Texto lematizado:['covid be very danger infection disease']

Texto listado:['covid', 'be', 'very', 'danger', 'infection', 'disease']

Lista sin stopwords:['covid', 'danger', 'infection', 'disease']

Comprobamos como 'very' y 'be' desaparecen pues son 2 términos de uso del lenguaje que suelen aparecer mucho.

3.2.3. Creación del diccionario

A partir de este punto crearemos ya las dos estructuras de datos que tomará el algoritmo LDA como entrada, el diccionario y el corpus.

El diccionario que llamaremos 'id2word'(identificado a palabra) es, como su nombre indica, una estructura de datos de tipo diccionario, la cual mapea los términos únicos de la lista con un identificador numérico.

Lo crearemos mediante el método Dictionary que toma como argumento la lista y devuelve el diccionario con clave identificador numérico y valor el término.

Lista sin stopwords:['covid', 'danger', 'infection', 'disease']

Diccionario:{1:'covid', 2:'danger', 3:'infection', 4:'disease'}

3.2.4. Eliminación palabras vacías

A pesar de eliminar las stopwords comprobaremos que al computar el modelo seguirán apareciendo palabras con una gran frecuencia, que se han escapado a la colección de stopwords.

En cambio, otras palabras aparecen un número de veces ínfimo en comparación a la media, ambos grupos nos causarán problemas.

Y lo harán porque a la hora de interpretar el modelo causarán que las temáticas se creen de manera muy poco representativa. Por un lado las palabras que aparecen mucho estarán en todas las temáticas y estas perderán independencia espacial. Esto quiere decir que varias temáticas serán parecidas entre si. Además de que las

temáticas al contener términos genéricos serán menos interpretables. Por otro lado las palabras que tienen muy baja incidencia, al punto de que aparezcan en unos pocos documentos, nos acarrearán grandes problemas cuando ejecutemos el algoritmo. Este problema se basa en que el algoritmo internamente genera una matriz de probabilidades, en la que cada término tiene una probabilidad de aparecer. La suma de las probabilidades de todos los términos tiene que ser 1. Pero la probabilidad de estos términos que solo aparecen en un par de documentos es tan baja que se cuenta como cero. Dando lugar a que al realizar la suma el resultado sea ligeramente inferior a 1 lo que se interpreta como un error en el algoritmo.

Para eliminarlas usaremos sobre el diccionario la función `filter_extremes` que descarta los términos que aparecen en menos de un número de documentos (extremo inferior) y descarta también los tokens que aparecen en más del un tanto por uno del total de documentos.

Tras varias pruebas y contrastar con otros proyectos, se decidió que el umbral inferior serían aparecer en menos de 20 documentos y el superior aparecer en más de un 10 % de los documentos totales.

3.2.5. Creación del corpus

Por último crearemos el corpus de bolsa de palabras, a partir del diccionario. Este será otro de los argumentos que necesita el modelo.

Consiste en una colección que representa la frecuencia con la que aparece cada término en un documento. De manera que para un documento x tendremos una lista de pares (identificador, frecuencia) para cada término del diccionario que se aparezca en el documento.

Diccionario: {1:'covid', 2:'danger', 3:'infection', 4:'disease'}

Corpus: [(1,1),(2,1),(3,1),(4,1)]

Lo que se traduce a que el término 'danger' con identificador 2 aparece 1 vez en el documento.

Este corpus se crea con el método `doc2bow()` a partir de `Dicctionary`.

3.2.6. Creación del modelo LDA y ajustes

El modelo se genera con la función `LdaModel` que toma como argumentos hasta 8 parámetros según mi implementación, aunque puede tomar muchos más. De esos 8 la mayoría están relacionados con factores de como se entrenará el modelo (`chunksize`, `passes`, `random_state`) y que se han puesto siguiendo referencias de otras implementaciones similares.

Sin embargo hay 3 argumentos que si son muy importantes:

- `common_corpus`: el corpus que hemos generado
- `id2word`: el diccionario que hemos calculado
- `num_topics`: el número de tópicos que infiere el modelo

La **elección del número de tópicos** es la decisión más determinante en el desarrollo del proyecto, porque todo gira y evoluciona a partir de ellas.

Lamentablemente y como se ha comentado antes, se necesita de una interpretación humana para darle un verdadero sentido a las temáticas, es decir, no existe un método matemático para calcular el número idóneo de ellas.

Desde el punto de vista de la máquina, nosotros le podemos pedir el número de temáticas que queramos, pero cuando las calcule y las observemos nos daremos cuenta que no están bien distribuidas o que no son interpretables.

Por ello el único modo de elegir un número de temáticas adecuado ha sido a base de pruebas y ejecuciones. Cabe destacar que ha sido un proceso un tanto tedioso debido a que con la gran cantidad de datos que el modelo usa cada ejecución del LDA requería bastante tiempo computacional.

Por lo tanto a partir de este método empírico fui acotando el número. En un extremo teníamos 10 y en el otro 100.

Por un lado con 10 temáticas obteníamos buenos resultados pero no habían una muy buena distribución de los términos para cada tópico. Es de esperar que si hay 10 temáticas, cada una este compuesta por cerca del 10% total de los términos, pero en la práctica esto no se cumple. Al haber tantos documentos pensé que 10 eran muy pocos.

Por otra parte con 100 tópicos, nos damos cuenta que a partir de la temática 96 los temas contienen un número de términos cercano al 0% por lo que encontramos un límite en la representación.

En estas ejecuciones si intentamos interpretar la mayoría de temáticas, son muy concretas en algunos casos pero en otras no tienen sentido.

Tras estas observaciones y teniendo en cuenta que el objetivo primordial del

trabajo es la aplicación de estos datos en el visor web de documentos, para ayudar a la gente a identificar los temas con colores, tenía que enfocarlo desde el punto de vista de que no hay un gran número de colores únicos.

Según los colores que se usan para el desarrollo de paginas web, hay calificados unos 24 colores fácilmente distinguibles por el ser humano, a partir de ese número ya cuesta mucho diferenciarlos.

Así que finalmente hice pruebas con un número intermedio y me quedé con 15 temáticas. Que como veremos en la siguiente sección nos dio bastantes buenos resultados.

3.3. Visualización resultados

Tras la generación del modelo utilizaremos la biblioteca `pyLDavis`[8] para la visualización del modelo que hemos entrenado. Con esta biblioteca, creada por Ben Mabey y Paul English, podremos interactuar con las temáticas encontradas y los términos relevantes que las forman entre otras cuantas características.

Para ejecutarlo una vez guardado el modelo solo hay que usar la función `prepare()` del algoritmo y se nos guardará un objeto que contiene toda la información para pintar los datos.

Esta información se puede exportar en diferentes formatos como notebooks o en formato html, que es el que usaremos nosotros.

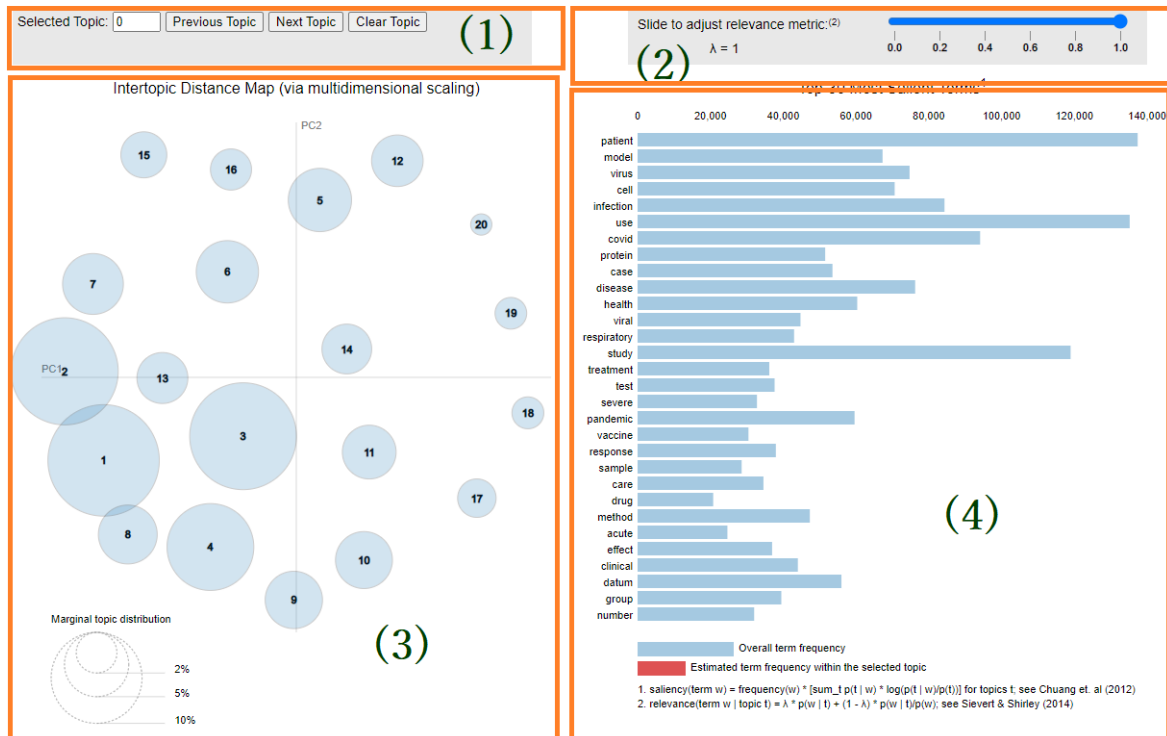


Figura 3.4: Barra de navegación

El resultado es el siguiente, donde diferenciamos las siguientes partes:

1. Barra para movernos entre las temáticas
2. Mapa que representa como las temáticas se relacionan
3. Ajuste de la métrica de relevancia de los términos
4. Representación gráfica de la distribución de los términos relevantes que forman una temática.

Hay mucho trabajo detrás de esta biblioteca para lograr esta representación tan buena. Pero nosotros no aprovecharemos esas características y solo nos servirá para guiarnos de como de bueno es el modelo que estamos entrenando.

Lo que nos interesa saber es que existirá dependencia entre las temáticas si están separadas unas de otras por una distancia determinada en la sección(3) de la representación. Si están muy juntas a otras significarán que las temáticas se parecen mucho y compartirán varios términos en común.

Capítulo 3. Cálculo de las temáticas

Por otra parte cuanto más grande sea la circunferencia significará que contiene mayor número de términos totales del diccionario del modelo.

En modelo perfecto buscaríamos que los círculos estuvieran bien separados y desvariación típica respecto al tamaño no fuera muy grande, es decir cada uno contuviera más o menos la misma proporción de términos.

En otros trabajos donde se trabaja con colecciones de documentos pequeñas, se pueden realizar estudios muy interesantes como 'Tópico dominante y contribución en el documento', 'La frase más representativa de cada tópico' o 'Frecuencia de la distribución de longitud de palabras en documentos'. Pero como nosotros trabajamos con una colección de documentos a gran escala me ha resultado imposible llevarlos a cabo.

3.3.1. Datos sobre elección del número de temáticas

Como comentaba en la anterior sección podemos visualizar una comparación de la elección en el número de temáticas.

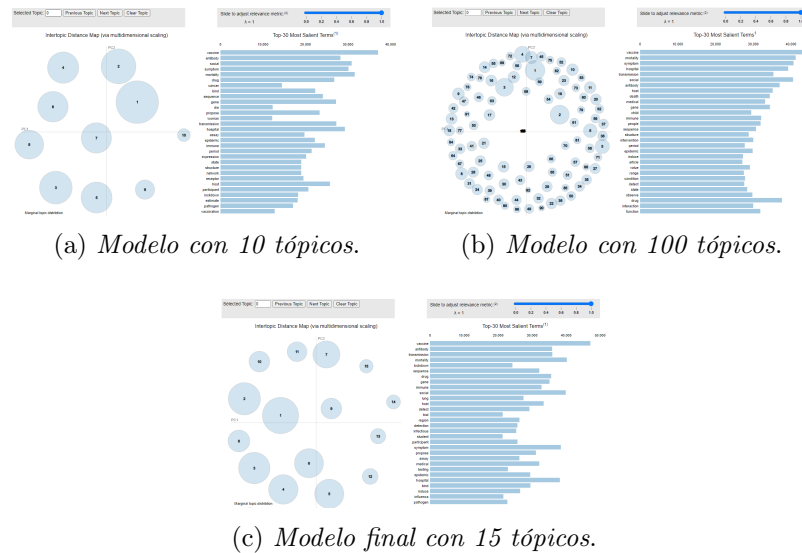


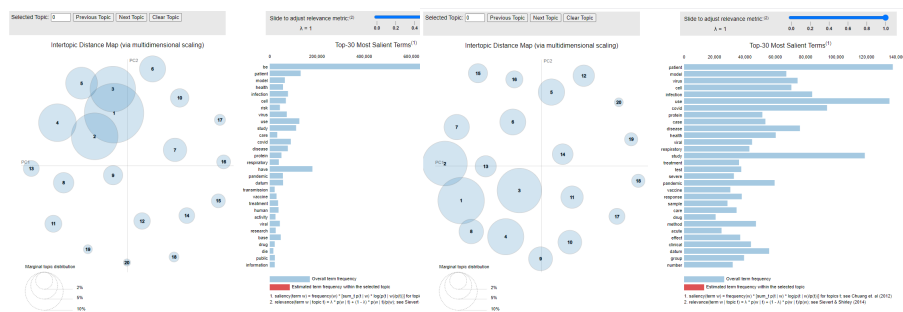
Figura 3.5: Comparación en la elección del número de tópicos

3.3.2. Datos sobre la eliminación de palabras frecuentes

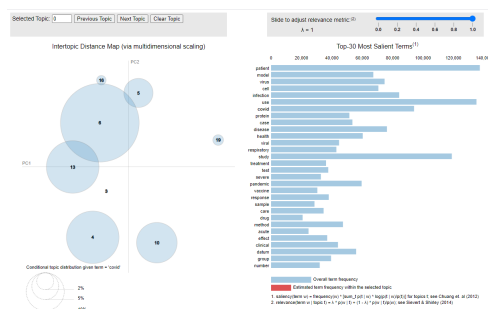
La primera y más notoria corrección del modelo se realizó tras ver que en las estadísticas encontrábamos palabras muy frecuentes en todos los tópicos y que no dejaban lugar a una correcta clasificación.

Los más comunes fueron 'be' y 'have' ya que entre estos dos términos casi abarcaban la mitad de incidencias totales del modelo.

Se corrigió haciendo un filtrado tras la lematización y añadiendo otros cuantos términos que se encontraban en casi todas las temáticas y que eran genéricos como 'use' o 'covid' y que por lo tanto no aportaban mucho.



(a) *Modelo con palabras frecuentes.* (b) *Modelo sin palabras frecuentes.*



(c) *Ejemplo de palabras frecuente 'covid'.*

Figura 3.6: Comparación refinamiento del modelo

3.3.3. Datos sobre interpretación de los términos de una temática

Ahora hablaremos brevemente de la interpretación de las temáticas que hemos obtenido, podría decirse que tenemos 3 tipos.

En primer lugar como en el ejemplo la temática 8, es una **temática que fácil-**

mente se identifica de que va con palabras como participante, encuesta, comportamiento, mental y varias problemas como estrés, depresión y ansiedad. Se podría definir como 'síntomas ocasionados por el corona virus en la población'. En el modelo generado la mayoría de temáticas son de este estilo.

En segundo lugar, tenemos de ejemplo la temática 12 en la que nos encontramos palabras como estudiante, programa, entrenamiento, energía, dispositivo, tarea o sensor. En este caso ya **cuesta más encontrar una temática que lo defina** y es un ejemplo de que el algoritmo ha descubierto una temática que habla sobre cuestiones menos relacionadas con la enfermedad. Es temática la interpretaría como: "Tecnologías relacionadas con el estudio del covid".

En último lugar encontramos la temática 7 que tiene términos como gen, receptor, molecular, tejido, inhibidor, encima y otras cuantas palabras que desconozco su significado. Es un claro ejemplo de cómo encontramos **temáticas muy específicas y que no somos capaces de interpretar** forman parte de un vocabulario muy técnico. Sería aquí donde intervendría un especialista en el tema para ayudar a dar una interpretación del tema consensuada y decirnos si tiene sentido.

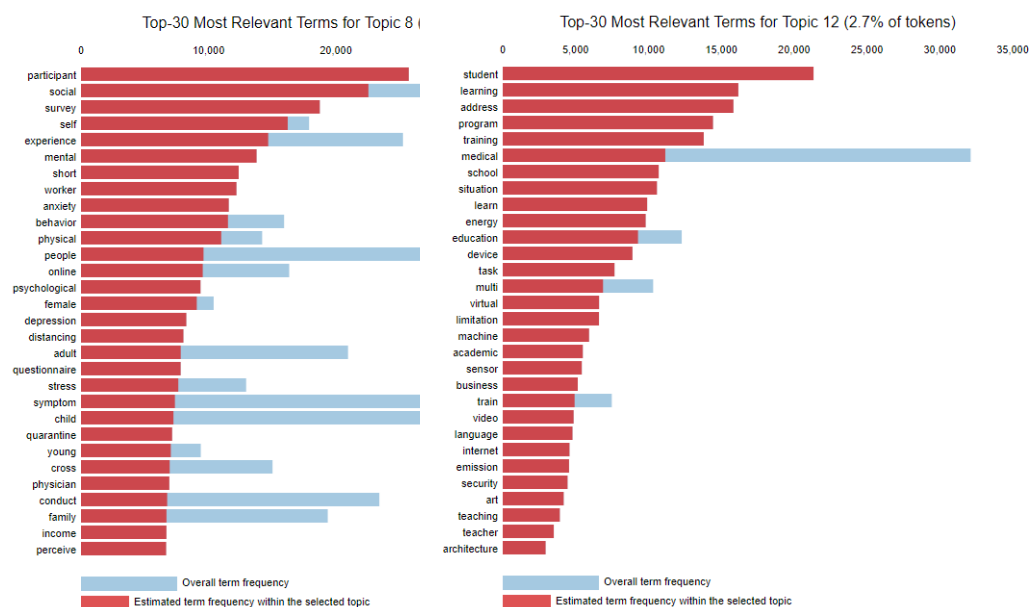
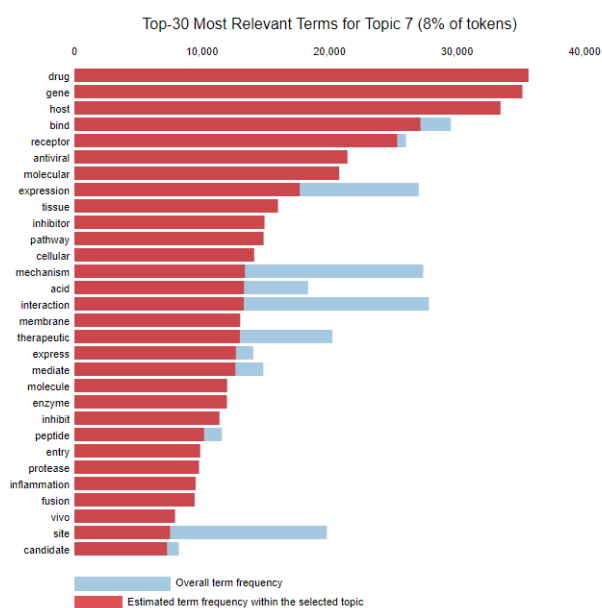
(a) *Términos de la temática 8.*(b) *Términos de la temática 12.*(c) *Términos de la temática 7.*

Figura 3.7: Comparación palabras relevantes de las temáticas

CAPÍTULO 4

Asignación de temáticas

La clasificación de los documentos consiste en evaluar de nuevo cada documento, esta vez con la ayuda del modelo obtenido, y calcular a qué temática es más probable que pertenezca.

Tras este cálculo le asignaremos a cada documento un identificador de la temática más probable y un color asociado a esa temática de entre una lista previamente preparada.

Esta lista o tabla relaciona el identificador de un tópico(int) con identificador de color(string de color en formato hexadecimal) en el documento **topic_colors.json**. Luego de calcular la temática de un texto lo usaremos para asignarle un color. Más adelante veremos el motivo de añadir un tópico 0 como desconocido debido a un problema que en ocasiones surge en la clasificación y que indicaría una 'no clasificación'.

0		11	
1		12	
2		13	
3		14	
4		15	
5		16	
6		17	
7		18	
8		19	
9		20	
10			

Figura 4.1: Relación tópicos-colores

En cuanto al calculo de la temática tenemos que tener en cuenta 2 puntos muy importante en esta etapa:

- Hasta el momento cada documento de la base de datos inicial se ha transformado en 2 distintos, uno en el que el texto esta agrupado y se ve como un conjunto único, y otro en el que el texto esta agrupado por frases por lo que el texto total se ve como una subdivisión de textos.
- Para cada documento que se ha usado para generar los clusters del modelo, se ha usado un enfoque de multi-clase, o adaptado al problema podría usar el término 'multi-temática'. Esto quiere decir que a un documento no se le ha etiquetado a una única temática, sino que debido a la aplicación del algoritmo LDA, usando las probabilidades de cada término, se ha obtenido que cada documento pertenece a cada temática en una probabilidad de forma que la suma de todas estas probabilidades es 1.

Partiendo de estas premisas vamos a calcular y decidir bajo los siguientes criterios cual el la temática asignada a cada texto o en otras palabras, que temática tiene la probabilidad más alta para un texto dado.

4.1. Asignación de la temática para texto completo

La primera asignación será sobre el conjunto de todo el texto que conforma el documento.

El cálculo de el tópico asignado a este documento es sencillo. Ya que podemos obtener la distribución de tópicos para un documento a partir del modelo calculado y el corpus del texto que queremos evaluar.

Algorithm 2 Asignación de la temática para texto por texto

```

1: function ASIGNATEMATICA_TEXTO
2:    $lda \leftarrow \text{load}(lda\_model)$ 
3:   for each  $file \in database\_text$  do
4:      $obj \leftarrow \text{open}(file)$ 
5:      $text \leftarrow obj['abstract']$ 
6:      $corpus \leftarrow \text{getCorpus}(text)$ 
7:      $topic\_distribution \leftarrow lda.get\_document\_topics(corpus)$ 
8:      $main\_topic \leftarrow \max(topic\_distribution)$ 
9:      $obj['abstract'][0]['topic'] \leftarrow main\_topic$ 
10:     $obj['abstract'][0]['color'] \leftarrow \text{getColor}(main\_topic)$ 
11:     $\text{save}(obj)$ 
12:   end for
13: end function

```

Lo único que tenemos que hacer es calcular el corpus y pasarlo de argumento a la función. Calcularemos el corpus siguiendo el mismo procedimiento que seguimos al ejecutar el algoritmo LDA.

Primero lematizamos el texto, le aplicamos `preprocess_simple`, lo convertimos en diccionario, filtramos los extremos y creamos el corpus convirtiéndolo en bolsa de palabras.

`get_document_topics(corpus,min_probability)`

Recibe el corpus del documento(formato bolsa de palabras) y una probabilidad mínima para mostrar tópicos del documento. Los tópicos con probabilidad inferior a $1e-08$ se descartarán para la salida.

Devuelve una lista de int y float, donde int es el identificador del tópico y el float la distribución asociada.

EJ: salida de un documento para un modelo con 10 temáticas

$$((1, 0.35), (4, 0.14), (9, 0.27), (10, 0.08))$$

Se interpreta como que los tópicos más probables son el 1, 4, 9 y 10, el resto tienen probabilidad tan baja que se descartan directamente.

Capítulo 4. Asignación de temáticas

De esta salida nos quedamos con la mayor probabilidad, en este ejemplo sería el tópico con identificador 1.

Buscamos en el documento topic-colors el código de color asociado a ese identificador, y añadimos ambos campos(topic y color) a los del texto en el json original.

La clasificación y modificación consecuente en el json que representa el documento sería de la siguiente forma:

```
1 # JSON antes de asignar de tema
2 {
3
4     "paper_id": "44590abf59c031019734dfb921b3e6b34e629293",
5     "title": "Air-pollutant_mass_concentration(...)",
6     "abstract": [
7         {
8             "text": "To_curb_the_spread_of(...)",
9         }
10    ]
11 }
```

```
1 # JSON tras asignar de tema
2 {
3
4     "paper_id": "44590abf59c031019734dfb921b3e6b34e629293",
5     "title": "Air-pollutant_mass_concentration(...)",
6     "abstract": [
7         {
8             "text": "To_curb_the_spread_of(...)",
9             "topic": "8",
10            "color": "#9f42c4"
11        }
12    ]
13 }
```

4.2. Asignación de la temática por frases

Llegamos al punto donde convergen todas las ideas que hemos mostrado hasta ahora, dando paso a la aplicación realmente novedosa en el uso del algoritmo LDA: **la asignación de temáticas inferidas a subfragmentos de un documento.**

Para entenderlo mejor tendremos en cuenta que:

- \mathbf{x} =tópico
- \mathbf{t} =término
- \mathbf{d} =subdocumento(frase)
- \mathbf{k} =número de tópicos del modelo

A diferencia de la anterior asignación de texto completo, que obteníamos directamente a partir de la probabilidad por tópico del modelo, en esta variante se requiere un proceso más elaborado.

Para obtener la temática más probable tendremos que hacer una serie de cálculos a partir, eso sí, de las matrices nos devolvía el modelo LDA generado.

La probabilidad de que la frase o subdocumento pertenezca a una temática X_1 será el producto de las probabilidad de cada término del subdocumento de pertenecer a la temática X_i .

Si repetimos esto para cada temática del modelo obtenemos la distribución del documento de pertenecer a una temática, donde asignaríamos al documento la temática con mayor probabilidad.

La base pues, de este nuevo cálculo es la probabilidad de temática de un término dado el documento que denotaremos como $p(\mathbf{x}|\mathbf{t},\mathbf{d})$.

Para calcularla haremos uso de una fórmula matemática obtenida de un estudio para subdividir documentos en el uso del algoritmo LDA[5].

Esta fórmula obtiene la probabilidad que necesitamos haciendo uso de otras 2 probabilidades que sí podemos extraer del modelo, la probabilidad de término por tópico $\mathbf{p}(\mathbf{t}|\mathbf{x})$ y la probabilidad de tópico por documento $\mathbf{p}(\mathbf{x}|\mathbf{d})$.

4.2.1. Cálculo probabilidad $p(\mathbf{x}|\mathbf{t},\mathbf{d})$

La expresión matemática se formula bajo la asunción de la independencia de los términos respecto de los documentos dados los tópicos, que significa que la relación que pueda existir entre términos y temática es independientes del documento en el que aparezca ese término. De forma que $p(\mathbf{t}|\mathbf{x},\mathbf{d})$ puede tratarse como $p(\mathbf{t}|\mathbf{x})$ usando ese enunciado, que es de donde parte la fórmula.

En resumen estas son las probabilidades que intervienen:

- **$p(\mathbf{x}|\mathbf{t}, \mathbf{d})$** : Probabilidad de tópico dado un termino de un documento (la que queremos hallar).
- **$p(\mathbf{t}|\mathbf{x})$** : Probabilidad un término dado un tópico (podemos obtenerla de `get_topics()` que nos devuelve la matriz term-topic del modelo).
- **$p(\mathbf{x}|\mathbf{d})$** : Probabilidad de tópico dado un documento (nos la devuelve la función `get_document_topics()`)

$$p(t, x|d) = p(t|x, d)p(x|d) = p(t|x)p(x|d)$$

$$p(t|d) = \sum_{l=1}^k p(t, x_l|d) = \sum_{l=1}^k p(t|x_l)p(x_l|d)$$

$$p(x|t, d) = \frac{p(t, x|d)}{p(t|d)} = \frac{p(t|x)p(x|d)}{\sum_{l=1}^k p(t|x_l)p(x_l|d)}$$

Figura 4.2: Fórmula cálculo de la probabilidad[5]

Obtención $p(\mathbf{t}|\mathbf{x})$

Usaremos la función `get_topics()` que nos devuelve directamente del modelo una matriz con la probabilidad de cada término para cada tópico.

Solo tendremos que acceder al término que queremos consultar y nos devolverá su probabilidad para cada temática.

Obtención $p(\mathbf{x}|\mathbf{d})$

Usaremos la función `get_document_topics()` la cual toma como argumento el corpus del documento a evaluar, en este caso la frase. Por ello tendremos que realizar todo el proceso que ya conocemos para obtener el corpus de la frase, lematización, lista de términos, creación del diccionario y finalmente obtención del corpus.

Como resultado la función nos devuelve una lista de int y float, donde int es el identificador de la temática y el float la probabilidad de que esa temática pertenezca a la frase.

Una vez obtenidas, procedemos por fin a calcular $p(\mathbf{x}|\mathbf{t}, \mathbf{d})$.

Usando la frase lematizada calculamos para cada término que la conforma el producto de las probabilidades para cada temática y normalizando ese producto, es

decir, dividiéndolo por la suma de las probabilidades de ese mismo producto para cada una de las temáticas posibles.

De esta forma obtenemos una matriz de dimensión número de términos de la frase por temáticas del modelo, donde cada columna es un término y cada fila una temática.

El último paso es multiplicar para cada temática las probabilidades de todos sus términos, y es aquí donde encontramos un inconveniente.

Al multiplicar la probabilidad de cada término nos va quedando un número decimal cada vez menor, de tal manera que llegado a cierto punto de multiplicaciones podríamos sufrir desbordamiento.

La solución a esto es aplicar logaritmos a cada probabilidad de manera que se convierte en una suma por las propiedades de los logaritmos.

Como queremos calcular la probabilidad de que una secuencia de texto t_1, t_2, \dots, t_n , por ejemplo una frase, en un documento d pertenezca a una determinada temática, x_l , esto es, $P(x_l | t_1, t_2, \dots, t_n, d)$. Esto se puede calcular como

$$P(x_l | t_1, t_2, \dots, t_n, d) = \frac{P(x_l, t_1, t_2, \dots, t_n | d)}{P(t_1, t_2, \dots, t_n | d)}$$

En el caso del denominador lo podemos calcular como

$$P(t_1, t_2, \dots, t_n | d) = \sum_{i=1}^k P(x_i, t_1, t_2, \dots, t_n | d)$$

Por lo que todo se reduce a conocer cómo podemos calcular

$$P(x_l, t_1, t_2, \dots, t_n | d) = P(t_1, t_2, \dots, t_n | x_l, d) P(x_l | d)$$

En este caso, podemos asumir la independencia entre términos y documentos dado que conocemos las temáticas, la expresión se reduce a

$$P(x_l, t_1, t_2, \dots, t_n | d) = P(t_1, t_2, \dots, t_n | x_l) P(x_l | d)$$

y si asumimos un modelo de bolsa de palabras, muy común en lo que es el procesamiento de términos en lenguaje natural, en el que se asume la independencia entre los términos en una determinada temática tenemos que

$$P(t_1, t_2, \dots, t_n | x_l) = \prod_{j=1}^n P(t_j | x_l)$$

Resumiendo, la expresión que queremos calcular se reduce a

$$P(x_l | t_1, t_2, \dots, t_n, d) = \frac{P(x_l, t_1, t_2, \dots, t_n | d)}{P(t_1, t_2, \dots, t_n | d)} = \frac{\prod_{j=1}^n P(t_j | x_l) P(x_l | d)}{\sum_{i=1}^k \prod_{j=1}^n P(t_j | x_i) P(x_i | d)}$$

Teniendo en cuenta que los productos de probabilidades en la expresión anterior, al ser cada uno de ellos valores en el intervalo $[0, 1]$, nos pueden dar problemas de underflow, deberemos utilizar un truco matemático para realizar estos cálculos. Para ello, trataremos de computar el logaritmo de las probabilidades, esto es,

$$\log(P(x_l, t_1, t_2, \dots, t_n | d)) = \log\left(\prod_{j=1}^n P(t_j | x_l) P(x_l | d)\right) = \sum_{j=1}^n \log(P(t_j | x_l)) + \log(P(x_l | d))$$

Una vez que lo tenemos calculado, podemos deshacer el cambio y recuperar los valores de las probabilidades originales, esto es,

$$P(x_l, t_1, t_2, \dots, t_n | d) = e^{\log(P(x_l, t_1, t_2, \dots, t_n | d))} = e^{\sum_{j=1}^n \log(P(t_j | x_l)) + \log(P(x_l | d))}$$

Finalmente nos queda una lista de suma de probabilidades por cada tópico de la cual asignaremos a la frase la temática con mayor puntuación en esa suma.

A la hora de clasificar en el formato frase, como la división por frases no es 100 % exacta, en ocasiones da lugar a frases cortas que por sus términos no logran probabilidad suficiente para ningún tópico o directamente tras la lematización de esa corta frase se descartan todos los términos. Por ello se crea el tópico 0 para estos casos y colorear la división por frases, problema que no ocurre con el texto completo.

También al calcular la asignación en formato frases, cuando calculamos la probabilidad de cada término necesitamos lematizar la frase, pero al no poder ejecutar la eliminación de extremos (porque no es proporcional a un solo fragmento que al conjunto de todos los textos), habrá palabras de la frase que no pertenezcan al corpus del modelo.

Para solucionar este inconveniente tuve que modificar el código fuente de la implementación de LDA de la biblioteca gensim, poniendo una condición de que si se intentaba evaluar una palabra que no estaba en el corpus, se le asignara probabilidad 1 es decir que no contase en el calculo final.

Algorithm 3 Asignación de la temática para texto por frases

```

1: function ASIGNATEMATICAFRASES
2:   for each documento  $\in$  database do
3:     for each frase  $\in$  documento do
4:       lista_prob  $\leftarrow$  list(float)
5:       matrix  $\leftarrow$  FloatMatrix(n_topics,len_frase)
6:       for each term  $\in$  frase do
7:         for each topic  $\in$  n_topics do
8:           matrix[topic][term] = calculaProbabilidad(topic,term)
9:         end for
10:      end for
11:      for each topic  $\in$  n_topics do
12:        lista_prob[topic] = suma(matrix[topic])
13:      end for
14:      main_topic = max(lista_prob)
15:      frase.asignaTopic(main_topic)
16:    end for
17:  end for
18: end function

```

4.3. Generación de archivos

Antes de cerrar definitivamente la etapa de generación de archivos e información en la que he hemos preprocesado la base de datos, hemos generado el modelo con sus estadísticas y ahora modificado de nuevo los documentos para añadirles la temática y su color, nos resta un último archivo que generar.

Para la siguiente parte del proyecto que es la presentación de los datos tenemos que generar un archivo html en el que se una la información del modelo (tópicos inferidos con sus correspondientes términos relevantes que lo definen) y la información de la tabla de colores para identificarlos visualmente.

Uniendo esta información creamos el archivo **topic_list.html** que muestra una lista de las temáticas, con su color correspondiente y sus términos relevantes.

Así cerramos la modificación y creación de archivos y no la tocaremos más.

CAPÍTULO 5

Desarrollo software

En esta fase del proyecto trasladaremos a un plano visual e interactivo todo lo que hemos ido consiguiendo hasta el momento.

Crearemos una aplicación web donde el usuario pueda hacer uso de un visor de documentos en el cual puede interactuar con cada texto y la asignación de temática realizada.

El objetivo de esta aplicación es que sirva como herramienta al usuario para identificar de que trata el texto en distintos grados de granularidad.

Lo usos partiendo de esta base de datos serían consultar bibliografías para la realización de proyectos y estudios. Agilización del proceso de documentación de un estudio, o apoyo en simple lectura de documentos.

Primero de todo hay que aclarar que la aplicación no tiene excesiva complejidad. Se buscaba un funcionamiento simple que cumpliera la función sin excentricismos ni características fuera del marco del proyecto. Está pensada para ser una aplicación gratuita y para todo el mundo, por lo que no hay usuarios registrados y al estar montada sobre el propio computador la lógica de acceso de datos es mínima. Por lo tanto no tiene un gran número de acciones con las que interactuar en la aplicación y las que hay son simples y están bien definidas.

Por todo esto, el proceso de análisis y diseño como se verá ahora es escueto e intuitivo. Debido a esto en algunas etapas del análisis no se pueden llevar acabo algunos pasos, como diagrama de clases.

Seguiremos un modelo lineal secuencial, más conocido como modelo en cascada que es un enfoque sistemático del desarrollo software, que comienza desde el plano conceptual y va progresando con las etapas de análisis, diseño, codificación y pruebas.

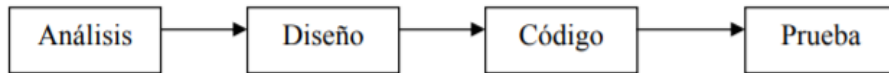


Figura 5.1: Modelo lineal secuencial (Cascada)

5.1. Análisis

En el proceso de análisis se reúnen requisitos, centrados principalmente en el software. Recogiendo requerimientos funcionales y no funcionales conseguiremos comprender completamente la naturaleza del programa.

REQUISITOS FUNCIONALES

RF-1 : Listar documentos

Descripción: Lista todos los documentos disponibles de la base de datos en forma de enlaces clicables

Salidas: Lista de documentos ordenados por su identificador de publicación.

RF-2 : Visualizar un documento

Descripción: Tras hacer clic en un enlace de documento se muestra el visor donde puede leer el texto.

Salidas: Una publicación científica.

RF-3 : Visualizar estadísticas del modelo LDA

Descripción: Hace clic en el barra de navegación en 'Statistics' y muestra por pantalla las estadísticas relacionadas al modelo.

Salidas: Gráfico interactivo sobre el modelo LDA entrenado

RF-4 : Cambiar a modo Texto de temática

Descripción: Pulsa el botón Mode y cambia la visualización de las temáticas del texto pasando a representar la temática del texto completo.

Precondiciones: Estar visualizando un documento y que el modo actual sea Frases

Salidas: Temática del documento completo.

RF-5 : Cambiar a modo Frases de temática

Descripción: Pulsa el botón Mode y cambia la visualización de las temáticas del texto pasando a representar la temática de las frases del texto.

Precondiciones: Estar visualizando un documento y que el modo actual sea Texto

Salidas: Temáticas de cada frase

RF-6 : Visualizar página inicio

Descripción: Hace clic en el barra de navegación 'DocViewer' y se muestra la página de inicio.

Salidas: Página inicio

CASOS DE USO DEL SISTEMA

Los casos de uso permiten especificar los requisitos del sistema y se suelen realizar antes de los requisitos funcionales. Su objetivo es mostrar la interacción entre la aplicación y el usuario, mostrando los pasos que se han de seguir para llevar a cabo cualquier tarea que la aplicación ofrezca.

CU-01 Leer un documento y ver a que temática pertenece

Tipo: servicio al usuario

Descripción: el usuario puede elegir de entre todos los documentos uno para visualizarlo

Secuencia pasos de acción

- 1 Clicka en la pestaña de lista de documentos
- 2 El sistema muestra una lista de documentos disponibles
- 3 Elige y clicka el documento que quiere
- 4 El sistema muestra el artículo con su texto y su temática asociada

CU-02 Conocer la lista de las temáticas disponibles

Tipo: servicio al usuario

Descripción: El usuario desea saber las temáticas que hay y las consulta

Secuencia pasos de acción

- 1 Clicka en la pestaña de lista de documentos
- 2 El sistema muestra una lista de documentos disponibles
- 3 Elige y clicka el documento que quiere
- 4 Se muestra el documento con las funcionalidad mode y topic list
- 5 Clicka el botón topic list
- 6 El sistema muestra la lista de temáticas con sus colores y sus términos relevantes

CU-03 Cambiar la granularidad de temática

Tipo: servicio al usuario

Descripción: El usuario esta visualizando un documento en un modo de temática y quiere cambiar a otro

Secuencia pasos de acción

- 1 Clicka en la pestaña de lista de documentos
- 2 El sistema muestra una lista de documentos disponibles
- 3 Elige y clicka el documento que quiere
- 4 El sistema muestra el artículo con su texto y su temática asociada además de activar botón de modo
- 5 El usuario clicka en el botón mode
- 6 El sistema recarga el documento con la nueva granularidad de temáticas

5.2. Diseño

Para lograrlo ideé un diseño sencillo dividido en 3 partes.

1. Un listado de todos los documentos de disponibles.
2. El visor de documentos donde leemos los elementos del mismo y podemos interactuar con las propiedades inferidas durante la clasificación.
3. Un apartado donde vemos las estadísticas del modelo obtenido ya mencionado en el anterior capítulo.

Todas estas partes en conjunto estarían conectadas por una interfaz manejada por una barra superior en la que podemos movernos con facilidad.

Cabe destacar también que en este proceso de diseño tome la decisión de que toda la aplicación web estaría escrita en inglés porque al ser el idioma de los artículos científicos lo veía más coherente.

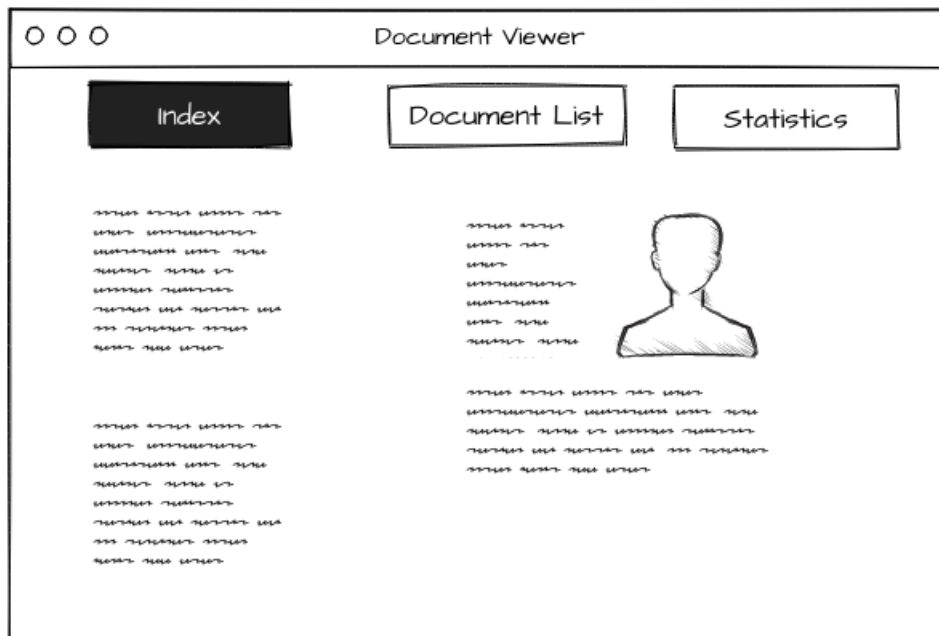


Figura 5.2: Index Page

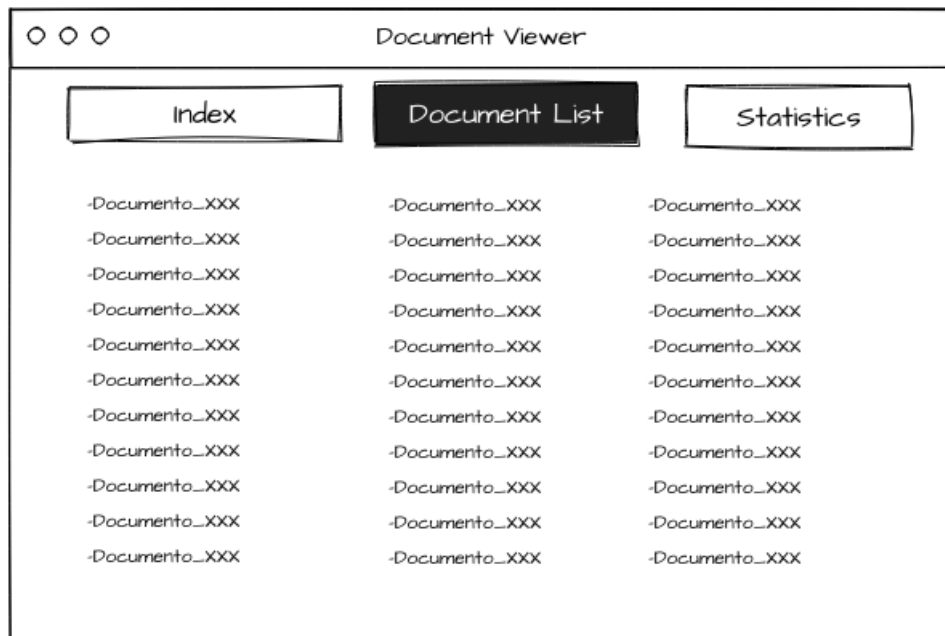


Figura 5.3: Document List Page

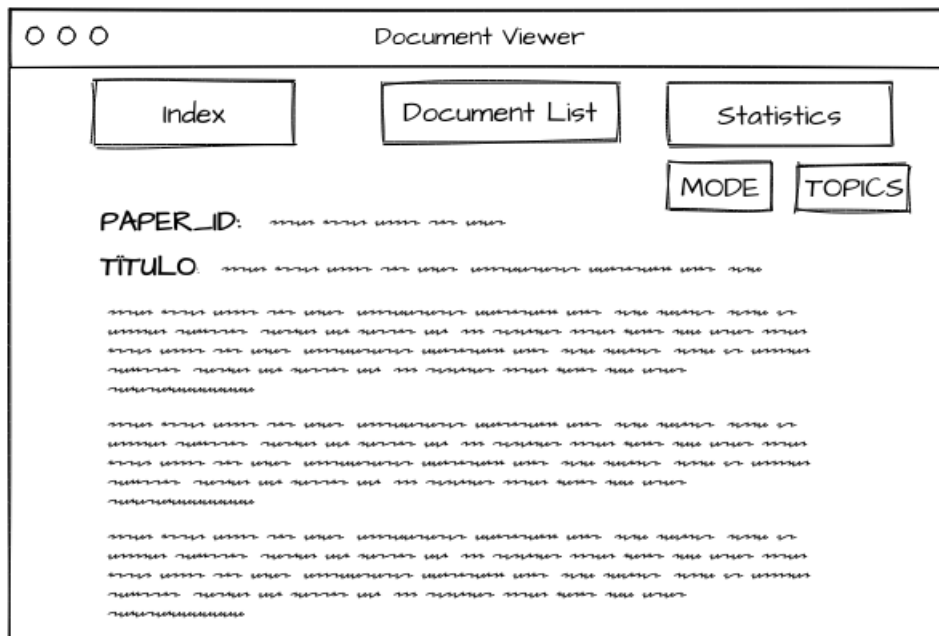


Figura 5.4: Document Page

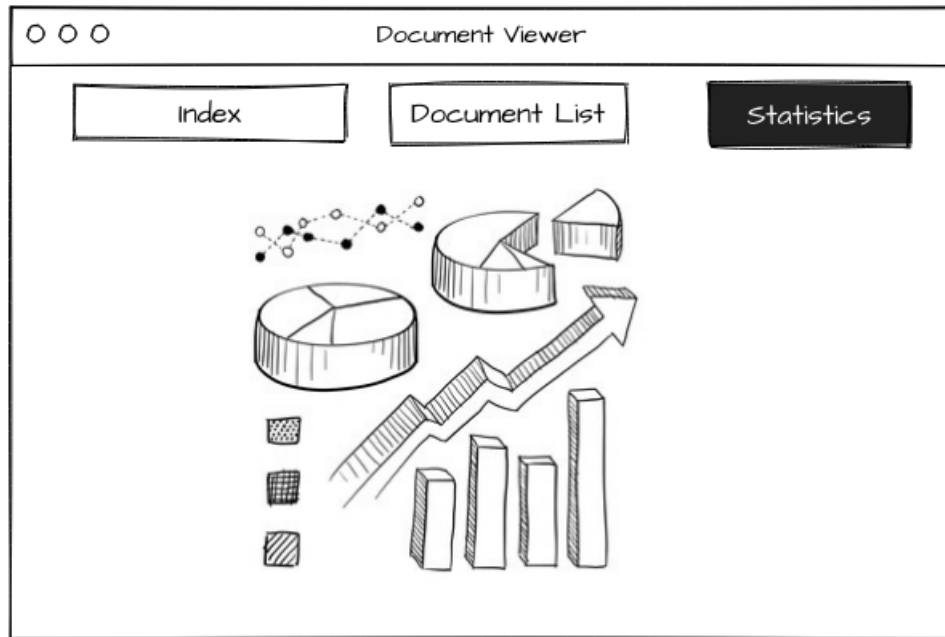


Figura 5.5: Statistics Page

5.3. Servidor web

El servidor web esta montado en Django[6] que es un framework de desarrollo web de código abierto, escrito en Python y que respeta el patrón de diseño conocido como modelo–vista–controlador.

La concepción del visor web esta pensada para cumplir su función de manera elegante y directa, sin elementos que acaparen mas importancia que la que verdaderamente tiene: mostrar los documentos y su clasificación.

Implementación

La aplicación se implementó en un PC con prestaciones estándar, con sistema operativo Windows, y fue desarrollada con varios lenguajes de programación (Python, HTML y JavaScript) sobre el framework Django.

A partir de un comando de Django creamos un proyecto simple el cual ya trae toda la lógica principal que necesitaremos para montar la aplicación. Las dos partes más importantes del servidor son el directorio **templates** y el archivo con toda la lógica de las vistas **views.py**.

La idea en este tipo de modelos es que la vista web se genere a partir de un template base (**base.html**) al que se van añadiendo de forma dinámica contenido, consiguiendo de esta forma que toda la aplicación tenga la misma estética y organización.

Este contenido que se añade parte, a su vez de otros templates anidados que servirán por ejemplo para mostrar las estadísticas (**statistics.html**) o listar los documentos disponibles para visionar (**doclist.html**).

Toda esta generación de contenido y la sincronización de toda la lógica subyacente está gestionado por el controlador, el archivo **views.py**.

En él encontramos numerosas funciones que se encargan entre otras cosas de abrir los archivos procesados en la clasificación, interpretar los datos en ellos y transformarlos en una presentación web.

Listado de documentos

Views.py

Se listan los archivos del directorio `final_database/tex` y se guardan en una lista de strings eliminando su extensión `.json`.

Esta lista se divide en 3 sublistas para poder mostrarlo luego por pantalla de manera eficiente, ya que son cientos de miles de documentos y se listan de una en el navegador, podría quedarse colgado. **Template**

Estas 3 sublistas se pasan al template donde en un bucle se van mostrando cada archivo en forma de link. Si se hace clic en el enlace del documento la web redirige a una página donde se puede leer el texto del documento gracias a que en la url se envían el nombre del documento y su formato que por defecto siempre sera `text`.

Visor de documentos

Views.py

Esta vista de la aplicación se muestra cuando clicamos en un enlace de la lista de documentos disponible. De esta obtenemos mediante la url el paper id y el modo.

En primer lugar leemos el archivo `json`(gracias al paper id) que modificamos en la parte de clasificación y que tienen toda la información necesaria para mostrar correctamente la clasificación del texto por colores.

De este archivo extraeremos título, identificador del documento y el abstract el cual contiene el texto completo ya que el modo que obtenemos de la url por defecto es TEXT.

De forma análoga entran en juego el archivos **topics.html** que la clasificación generó.

También en esta función leemos el archivo colors.json en el que se indica un listado que relaciona número de la temática con código de color asociado. El cual se lee en formato json (diccionario).

Todos estas variables una vez leídas se envían al template.

Template

Lo primero se muestran el identificador del documento y acto seguido el título del artículo el cual puede estar vacío.

A continuación se procede a mostrar el texto en modo TEXT que es el texto completo con un subrayado del color correspondiente. Para ello se muestra cada párrafo que compone el abstract(en este caso al ser texto completo solo es 1 con todo junto). Este elemento del abstract como se explicó anteriormente contiene 3 campos, text, color y topic. El template accede a estos campos y muestra el texto con fondo del color asociado.

Independientemente en la barra superior aparecen accesibles 2 botones, **topics list** y **mode**. El primero nos mostrará un ventana emergente donde podemos consultar los diferentes topics que existen en el modelo, su color y las 10 palabras más relevantes que lo conforma.

Este página no es otra que el archivo topics.html que generamos en la clasificación. El botón mode es el que nos permitirá visionar el documento seccionado por frases mostrando independientemente la temática más probable de cada frase.

El botón reenvía la misma url al controlador pero con argumento mode phrase, lo que hace que cargue el documento correspondiente pero del directorio final_database, que hará que se repita el proceso explicado con la diferencia que en este caso el abstract estará compuesto por numerosos elementos reverenciando a cada frase, cada uno con su campo text y color.

Visualización de estadísticas

Views.py

Simplemente lee el archivo donde el algoritmo almacenó las estadísticas en formato html y renderiza al template.

Template

En el template se recibe el html de las estadísticas y se muestra.

5.4. Guía usuario

El proyecto se ha desarrollado en sistema operativo Windows. Con una jerarquía de carpetas y archivos que se mostrará en la siguiente sección.

Se puede reproducir por completo todo el trabajo de manera correcta, simplemente a partir de la base de datos y la implementación del servidor web.

Ejecutando en correcto orden los 3 programas que he desarrollado, preprocesamiento, LDA, y clasificación de temáticas.

Prácticamente todo el código ha sido desarrollado en el lenguaje Python, a excepción de unas pocas cosas relacionadas con la presentación de los datos en el servidor web, las cuales se desarrollan en html, css y javascript. Como IDE de soporte a la hora de programar he usado Spyder para el text mining y Pycharm para la parte del servidor web que ha sido montada sobre Django.

Dentro de Spyder he creado un entorno virtual para este proyecto en concreto y en el cual he ido instalando todas las dependencias necesarias para poder realizar el proyecto.

Aparte de tener instalado Python y sus respectivas bibliotecas relacionadas básicamente solo tendremos que instalar las bibliotecas específicas relacionadas con la implementación del algoritmo LDA que usaremos, las cuales se desarrollarán más adelante.

Sin embargo algo de suma importancia a resaltar es que la implementación del algoritmo LDA que usaremos no funciona correctamente con las últimas versiones de algunas bibliotecas comunes como sería numpy, por lo que para el correcto funcionamiento de la aplicación en conjunto deberemos desinstalarlo e instalarlo de nuevo en una determinada versión. En concreto para numpy debemos tener instalada la versión 1.20.3 o de lo contrario obtendremos un error en tiempo de ejecución.

5.5. Estructura del proyecto

Partiendo de un directorio principal TFG tenemos 3 grupos de directorios y archivos.

1. Grupo directorios sobre procesamiento documentos

'pdf_json': documentos json directos de la base de datos

'final_database': documentos json con la necesaria de textos

'text': documentos json con texto todo junto

'phrases': documentos json con texto dividido por frases

'statistics': donde se guardan estadísticas del modelo de clustering

'topic_color.json': json con relación número temática-color

2. Archivos necesarios para obtener modelo y resultados

'Preprocesamiento.py': programa que crea la carpeta `final_database` a partir de `pdf_json/` con text y phrases con los archivos preprocesados.

'LDA.py': programa que crea el modelo de clustering usando el algoritmo LDA a partir de la carpeta `final_database->text`

'topic_clasification': programa que modifica la `final_database` añadiendo para texto y frases el tópicos al que pertenece y el color para luego mostrarlo.

'create_topic_html': crea un archivo para luego mostrar los distintos tópicos del modelo y sus palabras más significativas.

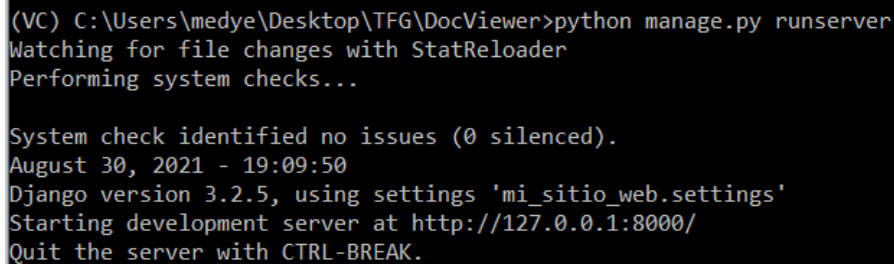
3. Directorio de servidor web

'DocViewer': directorio donde tengo montado el web server bajo Django

Para lanzar el visor de documentos debemos abrir la terminal de nuestro computador y ejecutar 2 sencillos comandos.

1. Colocarnos en el directorio del visor de documentos dentro del directorio de trabajo con **cd TFG/DocViewer**.
2. Levantar el servidor web con el comando **python manage.py runserver**.

Una vez ejecutados nos aparecerán los siguientes mensajes indicando el correcto arranque de la aplicación en la dirección local `http://127.0.0.1:8000`
Para finalizar el proceso como se indica solo hay que presionar 'ctrl + C'.



```
(VC) C:\Users\medye\Desktop\TFG\DocViewer>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 30, 2021 - 19:09:50
Django version 3.2.5, using settings 'mi_sitio_web.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figura 5.6: Levantar web server

En este momento si **abrimos** nuestro navegador web e introducimos la dirección donde el servidor se aloja podremos acceder a la aplicación.

De inicio veremos esta página donde se puede ver un resumen del proyecto y

arriba la barra de navegación.



Figura 5.7: Página de inicio

En esta barra tenemos 3 opciones, la primera 'Document viewer' donde volvemos a la página de inicio, la segunda 'Document List' donde se nos muestra los documentos disponibles y por último la pestaña 'Statistics' donde podemos ver las estadísticas relacionadas.



Figura 5.8: Barra de navegación

Si nos movemos a Document list nos encontramos un listado en 3 columnas de todos los documentos disponibles identificados por su paper_id. Haciendo uso del buscador del navegador (Ctrl+F) podemos buscar de entre todos los miles de documentos el que deseamos abrir o similares.

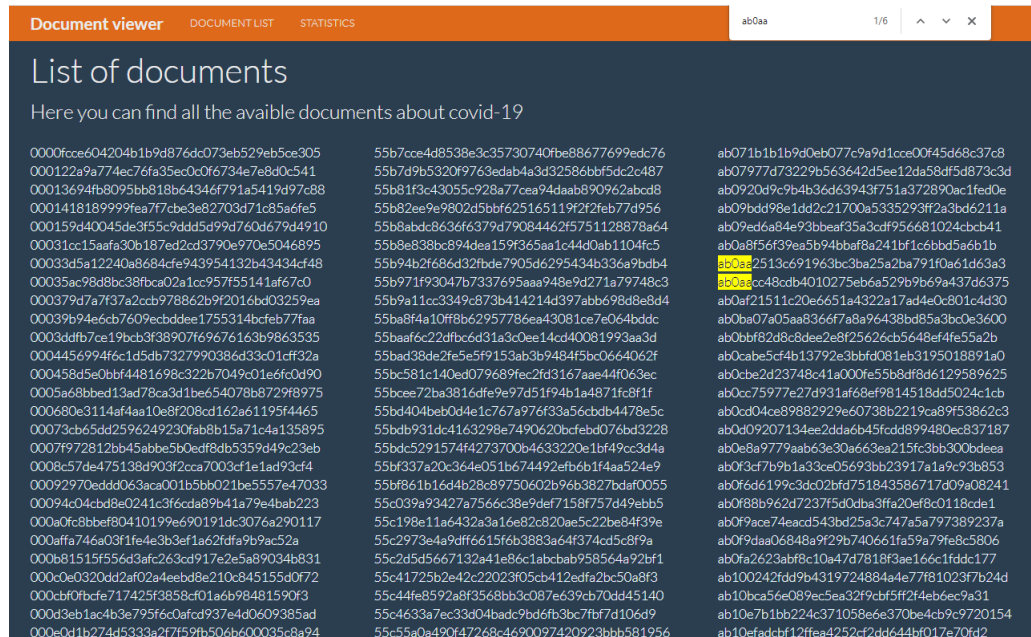


Figura 5.9: Lista de documentos disponibles

Una vez que hacemos clic sobre un documento de la lista, nos dirigimos al visor de este, donde se nos muestra en orden descendente: el identificador del artículo, el título y por último el texto que se encuentra con un fondo de color.

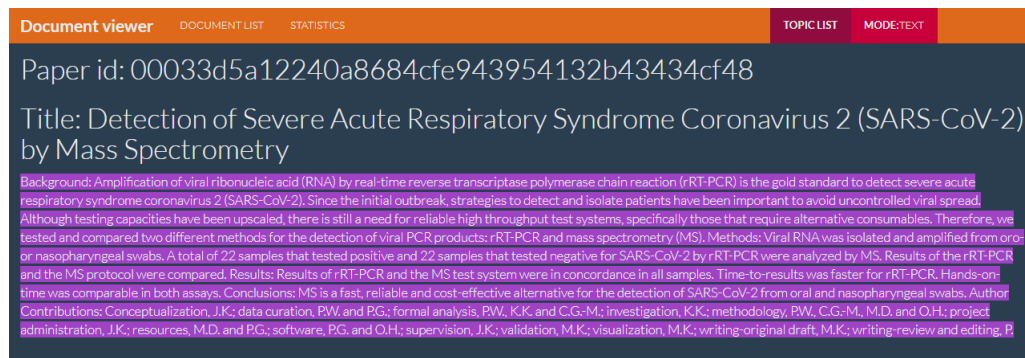


Figura 5.10: Visualización de documento completo

Nos damos cuenta que en la barra de navegación han aparecido 2 botones nuevos, Topic List y Mode.



Figura 5.11: Botones funcionales

Por defecto al abrir el documento desde la lista se nos abrirá en modo TEXT que es la clasificación total del texto.

Si clicamos en el botón mode nos cambiara la presentación del texto a modo frases, donde se determina la temática más probable por cada frase.

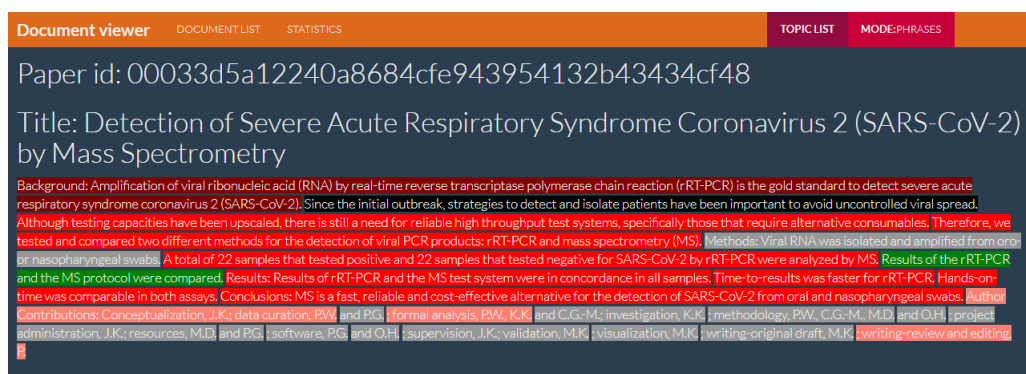


Figura 5.12: Visualización de documento por frases

Volviendo a clicar en el botón mode podemos seguir alternando entre estos dos modos.

Si pulsamos el botón topic list se nos despliega una lista con todas las temáticas del modelo en las que pueden clasificarse los textos acompañadas por el color que las identifica y las 10 palabras que definen la temática.

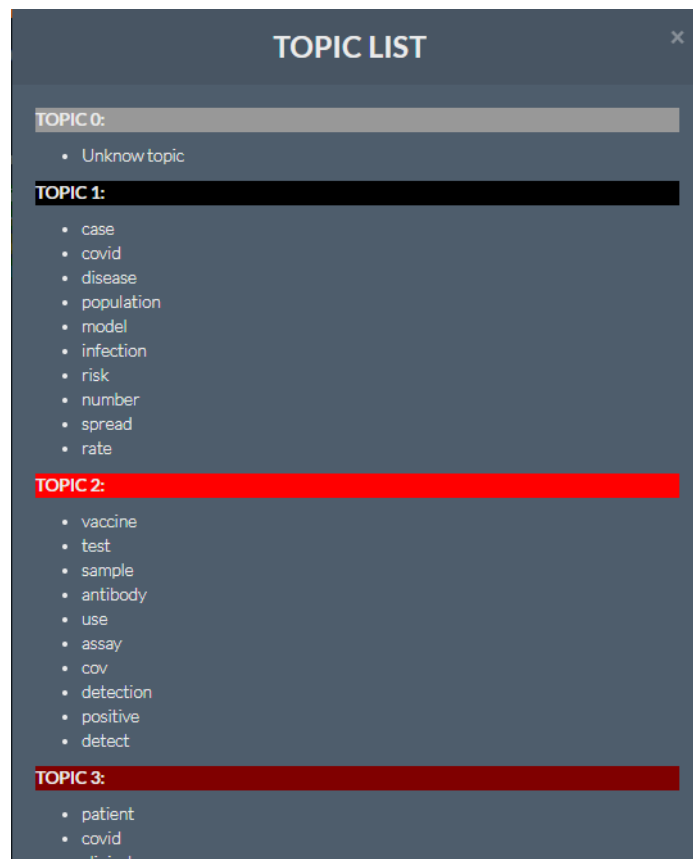


Figura 5.13: Lista de temáticas

Si salimos de la visualización de documentos los botones topic list y mode se ocultarán, sin embargo en cualquier momento de la interacción con la aplicación podemos acceder al apartado de estadísticas donde se nos muestran los datos del modelo generado con toda la infamación que nos generaba el modelo y que vimos en el capítulo 4.

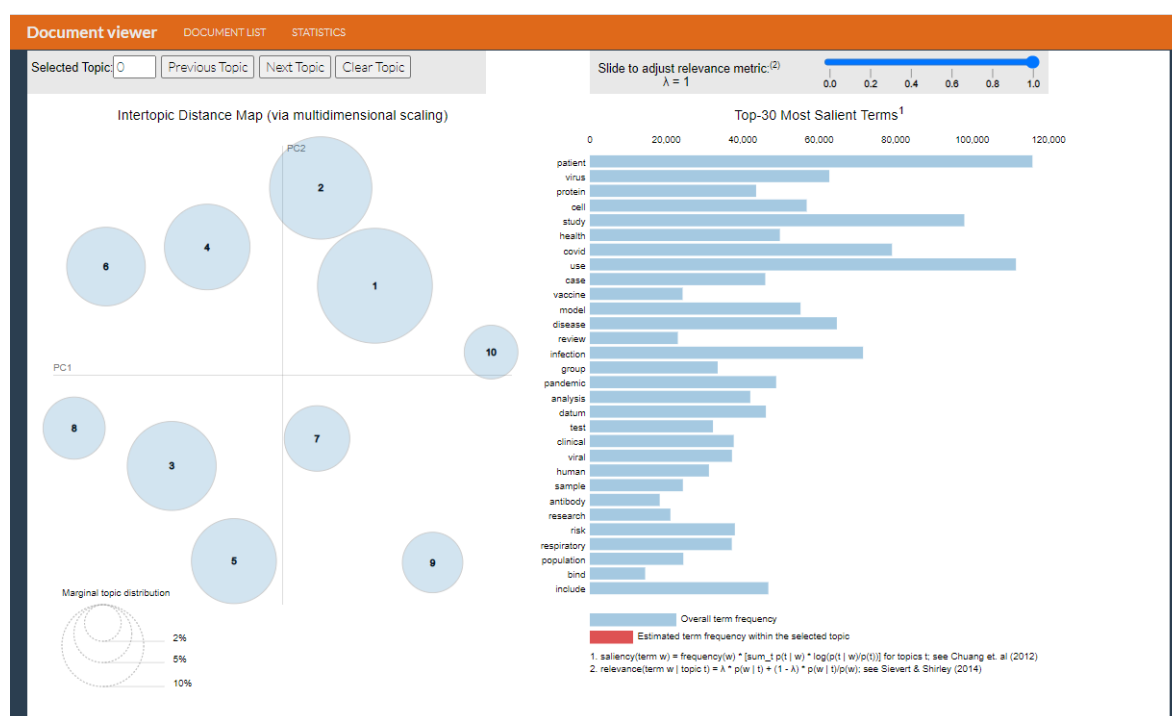


Figura 5.14: Estadísticas

CAPÍTULO 6

Conclusiones y vías futuras

Llegamos así a la parte final del proyecto donde veremos si hemos cumplido los objetivos que nos propusimos al principio, a que conclusiones podemos llegar y llevaremos a cabo un estudio de las vías de futuro del proyecto.

- En primer lugar podemos decir que hemos cumplido todos los objetivos que nos propusimos al principio de manera correcta y con un gran resultado. Quizás el único objetivo que ha quedado con un gran margen de mejor ha sido el de fluidez de la aplicación, porque al tener tantísimos documentos a la hora de listarlos en la aplicación estos se cargan lentos.
- Aunque el proyecto se realiza a una escala de extracción de información pequeña(*abstract*),no quita que las conclusiones inferidas a partir de los datos sean de gran interés.
- Podemos hacer una reflexión sobre las tecnologías que hemos usado y darnos cuenta de lo poderosas que son estas herramientas y el partido que podemos sacarle. Hace apenas unos años era impensable llevar a cabo estas tareas y menos que personas con poca experiencia con la debida documentación sean capaces de lograr estos resultados.
- Hay que recordar que era un total inexperto en el tema del text mining y aunque me queda mucho por aprender, siento que he aprendido muchísimo y

he ganado mucha experiencia profesional.

- La tarea de extracción de temáticas del campo del natural processing language tiene mucho margen por evolucionar aún y ha quedado demostrado que no es una tarea trivial, requiere de amplios conocimientos tanto en estadística y matemáticas como en programación para, a la hora de la implementación, poder exprimirlo al máximo y obtener los mejores resultados.
- Respecto a la aplicación que he desarrollado, conforme iba obteniendo resultados, me daba cuenta de que realmente es una herramienta muy útil y que me hubiera gustado tener en el ámbito académico cuando buscaba las bibliografías de mis trabajos en el grado.
- Es un gran inconveniente el necesitar de una interpretación humana para determinar parámetros como el número de tópicos y por otra parte saber si las temáticas de la aplicación tienen un sentido. Porque quizás alguien con conocimientos limitados podría tener dificultades a la hora de usar la aplicación, debido a que lee las palabras más relevantes de una temática y no es capaz de relacionarlas. Aunque estoy seguro que con el paso de los años estos modelos mejorarán y solventaran estos problemas.
- Me ha parecido muy enriquecedor e interesante trabajar con tecnologías tan a la orden del día, realmente hemos trabajado con el estado del arte en este ámbito y es una oportunidad que no siempre se tiene.

6.1. Vías futuras y mejoras

Tras estas conclusiones podemos hablar de algunas vías futuras que tendría la aplicación.

Aunque lo más importante es la aplicación final, debemos estudiar los puntos flojos del proyecto desde el principio. Es cierto que hay mejoras que solo repercuten en la aplicación final y que tienen que ver con el diseño o la implementación, pero hay otras que parten del modelo generado y que repercuten directamente sobre la aplicación final.

Desde un punto de vista de la herramienta de text mining, es decir de la obtención del modelo LDA, aunque hemos obtenido muy buenos resultados con este trabajo y teniendo en cuenta que en este tipo de tecnologías la escalabilidad siempre está presente, podemos asegurar que aún existe un margen de mejora considerable.

- El primer punto de mejora, si poseyésemos de capacidad computacional razonablemente mayor, sería realizar el mismo estudio usando el body text de los documentos en vez del abstract o quizás ambos. Esto resultaría en un diccionario de palabras mucho más concreto y específico, a costa claro está, de un incremento en su tamaño muy significativo. Pero con un filtrado y ajuste más exhaustivo podría obtener resultados muy interesantes.
- A su vez también podría refinarse un poco más eliminando cierta cantidad de documentos que contienen textos en idiomas que no sean el inglés. Como hemos podido ver en los resultados. Una de las formas sería identificar para documento en que idioma está escrito con la ayuda de un diccionario de cada idioma.
- Por otro lado, como se ha comentado, se utilizaba una lematización con un diccionario de artículos web, donde se pueden perder terminaciones del argot médico. Una mejora sustancial sería el desarrollo de un diccionario específico médico-científico para realizar la lematización. De esta forma podríamos quizás encontrar en el modelo palabras específicas que se habían saltado.
- Una vía de futuro sería ampliar la aplicación añadiéndole una escala más de granularidad a nivel de párrafos, aparte del que ya tenemos(texto completo y frases). Para ello habría que reestructurar toda la etapa de preprocesado y extracción de información. Pero de cara a la aplicación web obtendríamos un gran atractivo y podría ser un gran incentivo a la hora de ayudar a los usuarios a identificar las temáticas, ya que los párrafos es un camino intermedio entre texto completo y frase.
- Podría mejorarse la interfaz de la aplicación añadiéndole funciones para variar el número de temáticas entre varias opciones, por ejemplo 10, 20 o 30 tópicos y ver como influye a su vez en todo el conjunto del proyecto. Podríamos también incorporar un motor de búsqueda de documentos más sofisticado. Y diseñar una manera más fluida de moverse por los documentos.
- Desplegar la aplicación online en un dominio donde los usuarios puedan acceder y descargar el documento que quisieran tras elegir las publicaciones que más le interesen gracias a las temáticas.
- Realizar una abstracción de la aplicación y volverla funcional no solo para la base de datos covid. Añadir una funcionalidad en la que tu cargas tu propia base de datos y se genera todo, desde el modelo hasta la visualización y donde el usuario podría establecer algunos parámetros como el número de tópicos.

Ya hemos comentado que en ocasiones resulta difícil encontrar textos que incluyan un único tema y los aspectos tratados se entremezclan entre sí, por lo que lo habitual es enfrentarnos a textos en los que los temas se contaminan entre sí, pudiendo incluirse en más de un grupo a la hora de clasificarlos. Es un problema que asumimos desde el principio y tras obtener los resultados, es muy difícil valorar cuanto de bueno es o podría llegar a ser la clasificación que hemos hecho.

Creo personalmente que el text mining y el natural precessing language aún tienen mucho que evolucionar y que sorprendernos como se está viendo en los últimos años con otros campos del NLP. Quizá la agrupación por temáticas no tenga el peso e importancia que otros temas en la actualidad, pero estoy seguro que en unos años se obtendrán avances muy superiores y fascinantes, donde obtengamos resultados que no difieran tanto del que podría hacer un profesional humano.

6.1.1. Reflexión personal

La segmentación de un grupo tan grande de textos y donde dentro del mismo tema en común, el covid, hay tantísima cantidad de subtemas de los que se pueden hablar, estudios muy concretos o un gran número de opiniones y análisis, es una tarea de suma dificultad.

En el trabajo hemos logrado entrenar un modelo a partir de la base de datos y usarlo para clasificar los mismos documentos que sirvieron para entrenarlo. Es una tarea distinta a la de clasificar documentos con un modelo ya preentrenado que sería un simple etiquetado de documentos.

Nosotros hemos ido un paso más allá, creando un modelo customizado donde se ha intentado maximizar la independencia y relevancia entre los clusters y se ha ido reajustando para obtener los mejores resultados posibles. Todo ello con el fin de crear una aplicación realista y práctica, que pueda ayudar de verdad a las personas.

6.2. Valoración ética y social

Hay que destacar, que en los tiempos que nos ha tocado vivir, la increíble labor de todos los profesionales dedicados a la investigación ha sido incalculable. Y no solo investigadores sanitarios sobre el virus que nos ha tenido encerrados tanto tiempo, sino también estudios como este.

Cada día se es más consciente en el mundo el poder que tiene la información, los datos son el nuevo bien más valioso en la sociedad y estos datos esconden informa-

ción de gran interés. En este trabajo hemos conseguido extraer de qué tema tratan un sinfín de documentos, tarea que nos llevaría años leyendo uno por uno y ni decir tiene la impensable tarea de obtener unos resultados similares a los nuestros mediante un procedimiento manual.

Hoy hemos conseguido asignar una temática a un documento a partir aparentemente 'de la nada' de forma automática, ¿qué seremos capaces de hacer dentro de unos años?

Está en la labor de los ingenieros informáticos y de datos realizar estudios y en los ingenieros software el desarrollo de aplicaciones que mejoren y faciliten el trabajo de otras, creando así una cadena de valor.

En la que el trabajo de varios campos de la informática confluyen en un mismo propósito. No es ningún secreto que desde que se empezó a invertir en nuestra profesión, en la investigación, en el desarrollo. La vida de las personas ha ido evolucionando hacia un presente lleno de facilidades. Es cierto que mi aplicación tendría un perfil de usuarios con grandes conocimientos sobre el tema y a cualquier persona no le serviría de nada. Es un sesgo que he tenido en mente desde el comienzo, sabiendo que no va a tener nunca un gran impacto social ni va a cambiar la vida de nadie. Quizás en un futuro no muy lejano, en que se pudiera extrapolar a muchos otros ámbitos y obviamente con un funcionamiento y resultados válidos, estaríamos ante una herramienta que ayudaría a mucha gente como yo, una herramienta hecha por un estudiante para facilitar en algún tipo de medida la vida y labor de otros estudiantes.

Bibliografía

- [1] Jason Chuang, Christopher Manning y Jeffrey Heer. “Termite: Visualization Techniques for Assessing Textual Topic Models”. En: (mayo de 2012). DOI: 10.1145/2254556.2254572.
- [2] Layla Michan e Israel Munoz-Velasco. “Cienciometría para ciencias médicas: definiciones, aplicaciones y perspectivas”. es. En: *Investigación en educación médica* 2 (jun. de 2013), págs. 100-106. ISSN: 2007-5057. URL: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S2007-50572013000200006&nrm=iso.
- [3] Carson Sievert y Kenneth Shirley. “LDAvis: A method for visualizing and interpreting topics”. En: jun. de 2014. DOI: 10.13140/2.1.1394.3043.
- [4] Nicholas Fraser y Bianca Kramer. *covid19_preprints*. Mar. de 2020. DOI: 10.6084/m9.figshare.12033672.v54. URL: https://figshare.com/articles/software/covid19_preprints/12033672/54.
- [5] Luis Miguel Campos Ibáñez y col. *LDA-based term profiles for expert finding in a political setting*. This work has been funded by the Spanish Ministerio de Economía y Competitividad under projects TIN2016-77902-C3-2-P, PID2019-106758GB-C31, y the European Regional Development Fund (ERDF-FEDER)., mar. de 2021. DOI: 10.1007/s10844-021-00636-x. URL: <http://hdl.handle.net/10481/67746>.
- [6] *Documentación de Django*. 2021. URL: <https://docs.djangoproject.com/es/3.2/1>.
- [7] *Documentación de pyLDAvis*. 2021. URL: <https://spacy.io/models/en>.

- [8] *Documentación de pyLDavis*. 2021. URL: <https://pyldavis.readthedocs.io/en/latest/>.
- [9] *Gensim ldamodel Documentation*. 2021. URL: <https://radimrehurek.com/gensim/models/ldamodel.html>.
- [10] *Kaggle Database*. 2021. URL: <https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge>.
- [11] Wikipedia. *Distribución de Dirichlet* — *Wikipedia, The Free Encyclopedia*. <http://es.wikipedia.org/w/index.php?title=Distribuci%C3%B3n%20de%20Dirichlet&oldid=135785350>. [Online; accessed 05-September-2021]. 2021.
- [12] Wikipedia. *Kaggle* — *Wikipedia, The Free Encyclopedia*. <http://es.wikipedia.org/w/index.php?title=Kaggle&oldid=133814715>. [Online; accessed 05-September-2021]. 2021.
- [13] Wikipedia. *Latent Dirichlet Allocation* — *Wikipedia, The Free Encyclopedia*. <http://es.wikipedia.org/w/index.php?title=Latent%20Dirichlet%20Allocation&oldid=124355588>. [Online; accessed 05-September-2021]. 2021.
- [14] Wikipedia. *Latent Dirichlet allocation* — *Wikipedia, The Free Encyclopedia*. <http://en.wikipedia.org/w/index.php?title=Latent%20Dirichlet%20allocation&oldid=1033241492>. [Online; accessed 05-September-2021]. 2021.
- [15] Wikipedia. *Pandemia de COVID-19* — *Wikipedia, The Free Encyclopedia*. <http://es.wikipedia.org/w/index.php?title=Pandemia%20de%20COVID-19&oldid=138104246>. [Online; accessed 04-September-2021]. 2021.
- [16] Maoran Zhu, Xiaopeng Zhang y Hongwei Wang. “A LDA Based Model for Topic Evolution: Evidence from Information Science Journals”. En: *Proceedings of 2016 International Conference on Modeling, Simulation and Optimization Technologies and Applications (MSOTA2016)*. Atlantis Press, 2016/12, págs. 49-54. ISBN: 978-94-6252-284-8. DOI: <https://doi.org/10.2991/msota-16.2016.12>. URL: <https://doi.org/10.2991/msota-16.2016.12>.