Universidad Nacional Autónoma de México

Facultad de Estudios Superiores Acatlán

Clasificador Automático de Artículos Web Utilizando Topic Modeling

T E S I S

Que para obtener el título de
Licenciado en Ciencia de Datos

Presenta
Ivan Yaotzin Velázquez Rocha
Tutor Dr. Christian Rubio Montiel

2023

Agradecimientos

A mi madre Nieves Marcela Rocha Castro, quien me ha brindado -con todo su amor incondicional- las oportunidades que me han permitido llegar hasta aquí.

A mis hermanos Ana Meztli, Ismael Izcoatl y Julio Ilhuicatl, que han sido mi motor para avanzar tan lejos.

Gracias a mi tutor Dr. Christian Rubio Montiel y miembros del jurado, Dr. Leonardo Martínez Sandoval, Mtra. Teresa Carrillo Ramirez, Mtro. Rubén Romero Ruiz y Dr. Eduardo Eloy Loza Pacheco; gracias también al Lic. Christian Carlos Delgado Elizondo por su apoyo durante mi estancia en la FES Acatlán, así como a todos mis profesores que tuve a lo largo de la carrera pues cada uno aportó una parte del conocimiento que hizo posible este escrito.

Objetivo

El objetivo de la presente tesis es mostrar el diseño, implementación y uso de un sistema clasificador automático de artículos web, así como explicar todos los conceptos teóricos que son clave para entender dicha implementación.

Así mismo, el presente texto está pensada para que las nuevas generaciones de la licenciatura en Ciencia de Datos tengan una guía práctica (con bases teóricas) de cómo desarrollar una solución tecnológica de inteligencia artificial aplicada.

Introducción

El sistema Clasificador Automático de Artículos Web es una solución tecnológica de uso personal. Su enfoque es la administración de fuentes web. Así que este software es una herramienta en proyectos de investigación.

El código del sistema es de acceso público, puede ser descargado, analizado y utilizado por cualquier persona que así lo desee. Se puede ver en la siguiente liga: `https://github.com/yaotzinvr/clasificador_articulos_web_topic_modeling`

En el capítulo "Presentación del sistema"se explica la funcionalidad de la aplicación así como el proceso de configuración en la computadora personal del interesado. El proceso del diseño e implementación es lo suficientemente detallado para aquellas personas con poca o ninguna experiencia de programación; además de que sirve como una guía básica para aquellos con mayor experiencia en programación y para aquellos que estén interesados en modificar el código puedan hacerlo y sepan dónde comenzar.

En el primer capítulo de este trabajo está destinado a desarrollar la base teórica del sistema, explicando los conceptos necesarios para el desarrollo tecnológico.

Índice general

Capítulo 1

Marco Teórico

La UNAM, en sus Unidades de Apoyo para el Aprendizaje, define un aparato crítico como "el conjunto de citas y notas presentes en un trabajo escrito o proyecto de investigación, además del fundamento teórico de las ideas planteadas. Se convierte también en una fuente de información para quien lee nuestro escrito" [1].

A continuación se muestra el aparato crítico de la presente tesis, a través de los conceptos mínimos requeridos para entender (a nivel teórico y técnico) el diseño, la construcción y el funcionamiento del sistema informático clasificador automático de artículos web.

El orden de los siguientes temas es importante ya que la mayoría dependen progresivamente; por ejemplo es fundamental entender lo que es un sistema informático antes de intentar entender lo que es una aplicación web: siendo esta última un tipo de sistema informático.

1.1. Sistema informático

Un *sistema informático* es un sistema que procesa información. Está compuesto por hardware y software. El hardware es un conjunto de componentes físicos (eléctricos, electrónicos, electromecánicos, cables; microprocesadores, pantallas, sensores, etc.) que siguen las instrucciones del software. El software es un conjunto de instrucciones escritas utilizando un lenguaje formal capaz de ser interpretado por el hardware [2].

Un sistema informático existe únicamente bajo la interacción de hardware y

software; o dicho de otra forma, es el software siendo ejecutado por el hardware. Siendo más preciso, un sistema informático se compone de un conjunto organizado de sistemas informáticos, cada uno con un software y hardware específico.

A partir de este momento nos referiremos a un sistema informático simplemente como sistema o aplicación.

1.2. Arquitectura de software

Es un conjunto de patrones, reglas y abstracciones que definen la estructura, funcionamiento, interacción e infraestructura de un sistema a alto nivel [3]. Para definir todas las características y componentes de un sistema, la arquitectura se basa en requerimientos funcionales, de mantenibilidad, flexibilidad, interacción, etc. Estos requerimientos los dictan las necesidades del usuario, las limitaciones de la tecnología, los procesos, y cualquier factor externo que intervenga en el desarrollo y uso del sistema.

1.3. Infraestructura

La infraestructura de las tecnologías de la información (la llamaremos simplemente infraestructura) se refiere al conjunto de hardware, software, componentes de red, sistema operativo, almacenamiento de datos, etc. que permiten operar y gestionar sistemas informáticos [4]. Algunos ejemplos de los componentes de infraestructura son:

- Hardware: de forma general se pueden ver como centros de datos, servidores, computadoras personales; y de propósito más específico pueden ser GPUs o TPUs.
- Redes: conexión a internet, firewall, enrutadores, conmutadores, software de gestión de tráfico de red, etc.

- SO: el sistema operativo es un conjunto de programas que gestionan los recursos del hardware y proporcionan servicios de recursos a software de propósito más general.

La infraestructura tiene dos importantes clasificaciones:

- Tradicional: Es aquella que existe físicamente: servidores, redes, espacio físico acondicionado. Cada vez es menos común por los altos costos que involucra.
- Nube: Se refiere al consumo de recursos de hardware a través de internet. Es decir, se rentan únicamente los recursos requeridos: procesamiento, almacenamiento y redes. Este modelo de infraestructura es cada vez más común, dada la facilidad de gestionar los costos.

1.4. Navegador web

Es un software especializado en obtener información de internet y mostrarla en una computadora personal. Siguen diferentes tipos de reglas y protocolos (el básico es HTTP), no sólo para extraer información sino para mantener un buen rendimiento y seguridad [5].

1.5. API Web

Una Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) es un conjunto de herramientas de programación que permiten comunicarse e interactuar con un sistema. Una API está construida, por lo general, para más de un lenguaje de programación dado que su objetivo es conectar dos sistemas distintos (con arquitecturas independientes).

Una API web es una API que sirve para controlar una aplicación web desde otros sistemas (no necesariamente web). El navegador web, por ejemplo, es un tipo de sistema que incluye varias APIs web.

1.6. Aplicación web

Es un sistema que está siendo ejecutado por un servidor web. Su principal característica es que la forma de acceder a él es a través de un navegador web o de una API web; lo cual lo diferencia de otro tipo de sistemas como los sistemas embebidos (o sistemas integrados, que se ejecutan dentro de un componente de hardware específico y no tienen contacto con otros componentes; por ejemplo, el software de un cajero automático) o los sistemas on-premises (aquellos que se ejecutan en un servidor sin conexión a internet; por ejemplo, los programas de paquetería de oficina en una laptop que pueden funcionar sin necesidad de conexión a internet) [5].

Las principales características de una aplicación web, son:

- La ejecución del software es independiente a la computadora desde donde se consume.
- Las solicitudes de procesamiento viajan a través de internet hasta el servidor web.
- El procesamiento de información se realiza en el servidor web y los resultados viajan a través de internet de regreso hacia la computadora que inició el consumo.

1.7. Virtualización

Es una técnica que consiste en utilizar software para imitar las características de hardware. Esto permite generar sistemas virtuales. El uso más común es tener un sistema operativo base, y sobre él montar sistemas virtuales que bien podrían ser diferentes sistemas operativos; de esta forma se pueden organizar y mantener varias aplicaciones de manera independiente; es decir, totalmente aisladas. Básicamente todas las aplicaciones y sistemas que ocupamos son el resultado de varias capas de virtualización.

Existen diferentes tipos de virtualización (de almacenamiento, memoria, procesamiento, recursos de red, etc.) pero el más utilizado es la virtualización de plataforma (también es llamada máquina virtual). Una máquina virtual es el resultado de un Hipervisor (VMM: Virtual Machine Monitor) el cual es un software que se ejecuta sobre el sistema operativo base y crea una capa de abstracción de los recursos de la computadora [6].
Es muy común las arquitecturas que mezclan diferentes tipos de virtualizaciones. Por ejemplo tener varias máquinas virtuales en servidores físicos independientes pero que comparten un mismo almacenamiento físico, como lo muestra la figura 1.1.

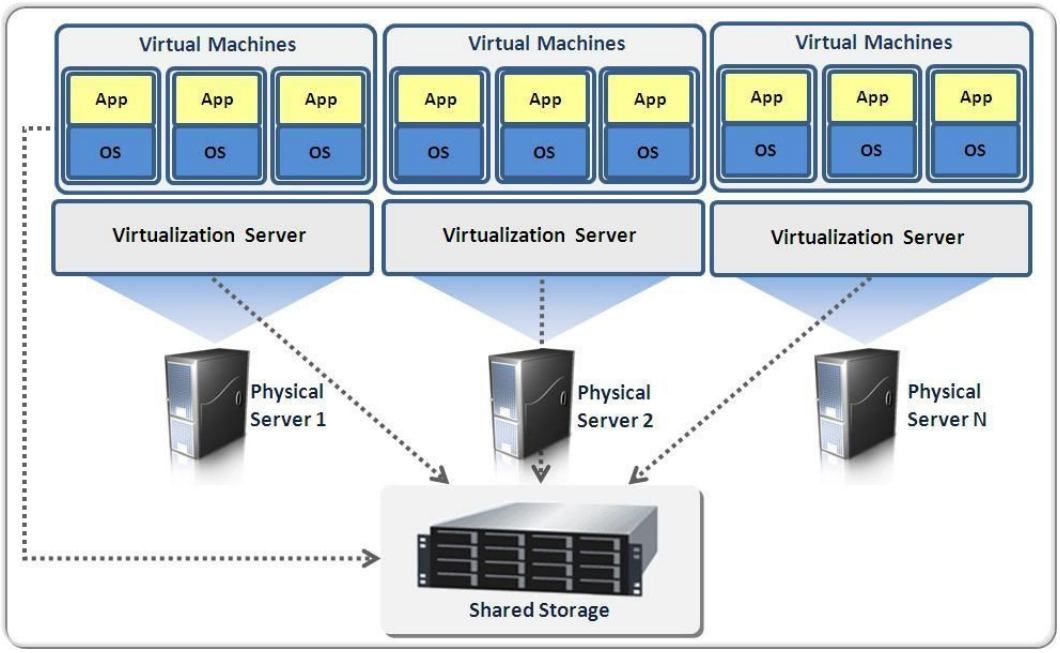


Figura 1.1: Virtualización de servidores. Fuente: tomado de [17].

1.8. Contenedores virtuales

Son sistemas independientes y empaquetados, que siguen una serie de estándares técnicos que permiten reutilizarse entre sí. Son una forma distinta de virtualización que las máquinas virtuales puesto que, a diferencia de ellas,

los contenedores no utilizan un hipervisor y abstraen los recursos del sistema operativo específicos para su funcionamiento, como lo muestra la figura 1.2. Esto brinda varias ventajas, como son la utilización óptima de los recursos del sistema operativo, mayor portabilidad y mayor control sobre la seguridad.

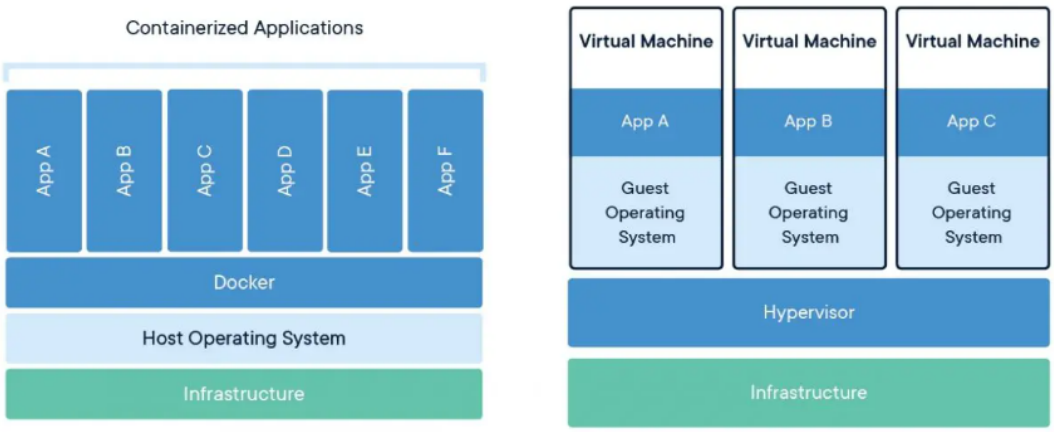


Figura 1.2: Docker vs Kubernetes: ventajas y desventajas. Fuente: tomado de [18].

El ciclo de vida de un contenedor virtual es el siguiente:

1. Primero existe una *imagen*, la cual es la abstracción de un contenedor y puede ser distribuida y reutilizada en otras imágenes. Contiene el código fuente y dependencias de la aplicación.

2. Una vez que a una imagen se le asigna valores específicos (como capacidad de memoria, variables de ambiente, etc.) se crea un contenedor el cual se puede ver como una instancia de una imagen; puesto que tiene un identificador único. De hecho se pueden crear contenedores independientes a partir de una misma imagen, cada uno con características únicas.

3. Un contenedor puede estar en ejecución o en pausa; cuando está en pausa no ocupa recursos del sistema (únicamente almacenamiento). Su ciclo de vida termina cuando es eliminado.

Los contenedores virtuales llegaron a revolucionar la infraestructura, por las ventajas que proveen:

- Portables: independientes del hardware y sistema operativo.
- Consistentes: funcionamiento predecible.
- Eficientes: fáciles de escalar y ligeros; se pueden ejecutar varios contenedores a la vez de una misma aplicación, además de que crearlos y eliminarlos es muy rápido.

1.9. Programación

La programación es el proceso de crear instrucciones específicas que indican a una computadora qué hacer, escritas en un lenguaje de programación. Existen diferentes formas de programación, cada una sigue un paradigma particular (un paradigma es una forma de entender un problema y plantear posibles soluciones). Algunos de los tipos de programación son imperativa, orientada a objetos, orientada a eventos, recursiva, etc.

1.10. Lenguaje de programación

Es un lenguaje formal (con símbolos y reglas bien definidas) informático que permite implementar un algoritmo (un algoritmo es una serie de instrucciones bien definidas, ordenadas y finitas que resuelve un problema matemático o lógico). No todo lenguaje informático es un lenguaje de programación (por ejemplo, el lenguaje de marcado, el declarativo, o los protocolos de comunicación). Los lenguajes de programación se pueden clasificar de diferentes maneras: lenguajes de alto y bajo nivel; lenguajes que soportan diferentes paradigmas de programación; lenguajes de propósito general y específico [7].

Ejemplo.
Python es un lenguaje de programación de alto nivel, de propósito general y que soporta varios paradigmas de programación, como es el funcional, imperativo y orientado a objetos. Su gran popularidad y amplia adopción se debe a varios puntos:

- Fácil lectura para el programador y fácil extensión: creación sencilla de nuevos módulos.
- Multiplataforma: es compatible con todos los sistemas operativos.

1.11. Web scraping

Es una técnica basada en programación, protocolos HTTP y APIs web que tiene como objetivo recolectar información de internet simulando el comportamiento de un humano, pero como un proceso automático y específico [8].

Los pasos generales del web scraping son:

1. Acceder a la información web.
2. Extraer la información.
3. Guardar información como datos estructurados o semiestructurados.

Se utilizan diferentes métodos para cada uno de esos pasos.

- Métodos para acceder a la información web:
    - Haciendo peticiones HTTP (Protocolo de Transferencia de Hipertexto o HTTP por sus siglas en inglés) de forma directa al servidor web. Para esto se requiere saber exactamente cuáles son los métodos y configuración necesaria que espera el servidor. Se utiliza cualquier aplicación que pueda hacer peticiones HTTP; por ejemplo un cliente HTTP, navegador de internet, bibliotecas de código en casi cualquier lenguaje de programación, etc.
    - Usando un navegador web que sea controlado por un proceso de automatización. El ejemplo más común es Selenium, la cual es una aplicación que simula ser un humano para interactuar con un navegador web.

- Métodos para extraer la información
    - Usando expresiones regulares, las cuales son una técnica para crear patrones de búsqueda en texto.
    - Analizando y usando el DOM (Modelo de Objetos del Documento o DOM por sus siglas en inglés) del propio recurso web. DOM es un estándar de organización de los recursos web para representar documentos escritos en los lenguajes HTML y XML, y que permite acceder a los datos de forma dinámica.
    - Usando bibliotecas en lenguajes de programación para manipular lenguaje HTML y XML.
    - Algoritmos de aprendizaje automático o Procesamiento de Lenguaje Natural (NLP por sus siglas en inglés) específicas para la extracción del texto requeridos.
- Guardar información
    - Guardar la información como datos estructurados; por ejemplo en texto plano o en alguna base de datos relacional.
    - Guardar la información como datos semi estructurados; por ejemplo en texto plano o en alguna base de datos no relacional.

1.12. Inteligencia artificial

> *"Artificial intelligence is the science and engineering of making intelligent machines, especially intelligent computer programs."*
>
> *John McCarthy* [9]

Como se muestra en la figura 1.3, la inteligencia artificial se divide en campos de conocimiento más especializados.

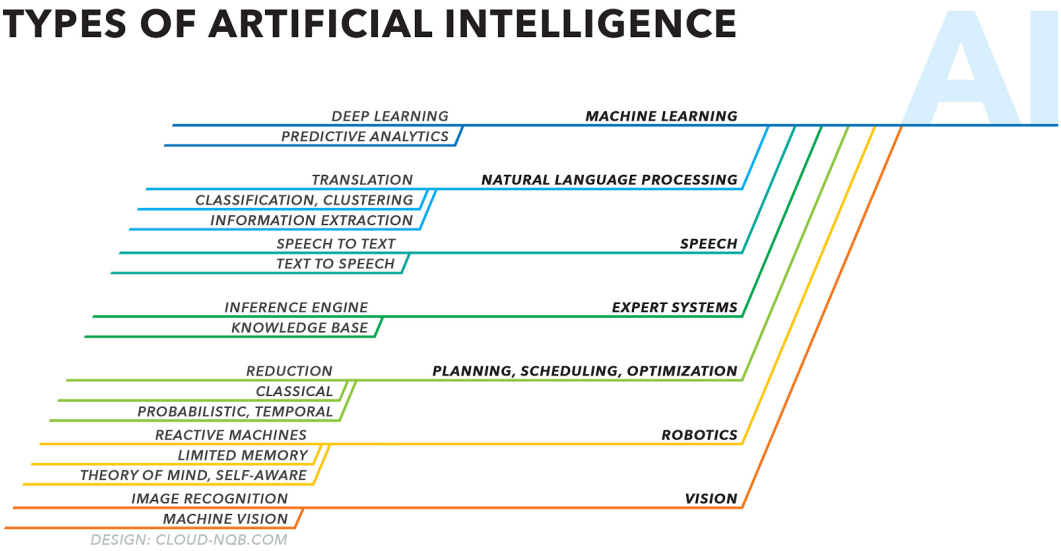


Figura 1.3: Types of Artificial Intelligence. Fuente: tomado de [19].

1.13. Aprendizaje automático

El aprendizaje automático - también conocido como aprendizaje de máquina (Machine Learning en inglés) - es un campo de conocimiento que forma parte de la Inteligencia Artificial. Tiene como objetivo generar modelos de predicción y clasificación de datos. Combina la capacidad computacional con métodos numéricos, matemáticos y estadísticos para crear modelos con capacidad de aprender (y evolucionar) de acuerdo a los datos recibidos [10].

Los modelos de aprendizaje automático se adaptan de acuerdo a los datos y eso los diferencia de los modelos matemáticos tradicionales que utilizan reglas explícitas para predecir y clasificar. Esto se logra por la automatización de algoritmos de métodos numéricos, estadísticos y matemáticos por medio de un lenguaje de programación. Para que un modelo de aprendizaje automático pueda describir y/o predecir el comportamientos de ciertos datos, debe de ser entrenado con una gran cantidad de datos.

El aprendizaje automático, según se observa en la figura 1.4, se divide en campos de conocimiento especializados.

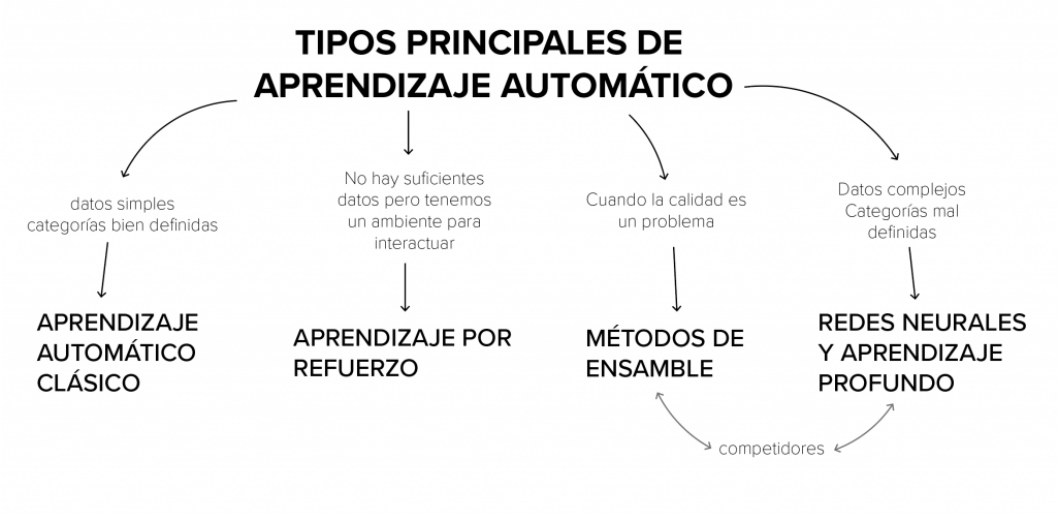


Figura 1.4: Inteligencia Artificial y Machine Learning para todos. Fuente: tomado de [20].

1.14. Procesamiento de Lenguaje Natural (PLN)

Es un campo interdisciplinario que tiene su origen en la Inteligencia Artificial, las ciencias de la computación y la lingüística. Se enfoca en el estudio y desarrollo de métodos para la comunicación entre computadores y humanos utilizando lenguaje natural.

El lenguaje natural es aquel que es hablado por humanos, el cual se contrapone con el lenguaje artificial, el cual es el hablado por máquinas. El resultado buscado por el PLN es que las computadoras puedan entender el lenguaje natural y responder a él de forma casi instantánea. Para lograr esto no sólo hace falta métodos de traducción, sino que deben de ser computacionalmente eficaces [11]. Existen varios subcampos al interior de PLN cada uno con objetivos muy puntuales, como se observa en la figura 1.5.

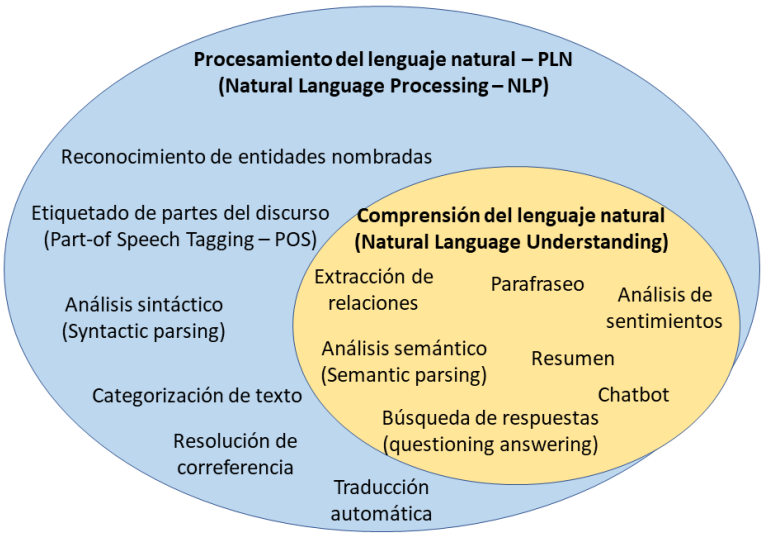


Figura 1.5: Procesamiento del lenguaje natural y Comprensión del lenguaje natural. Fuente: tomado de [21].

Debido a la complejidad del lenguaje humano, el PLN tiene varios retos:

- Ambigüedad léxica: en el lenguaje natural el significado de cada palabra lo da el contexto de una forma muy amplia.
- Ambigüedad semántica: en el lenguaje natural, el significado de una oración depende de contextos culturales, históricos, materiales, emocionales, etc.
- Incompletud de información: el proceso humano de entender el lenguaje es siempre a partir de datos incompletos y con ruido.

Dada la complejidad del lenguaje natural y la dificultad para crear algoritmos con sus reglas explícitas, el PLN se ha visto altamente beneficiado por el aprendizaje automático.

1.15. Chatbot

Un bot conversacional (chatbot) es un software que simula mantener una conversación con un humano, utilizando una serie de procesos reiterativos y siguiendo reglas específicas [12]. El surgimiento de chatbots funcionales de forma masiva se dio gracias al aprendizaje automático, y la sofisticación del chatbot depende de la complejidad de dicho modelo o modelos que hay detrás.

El funcionamiento de un chatbot es el siguiente:

- Se captura o recibe texto en lenguaje natural, que se envía a su vez a módulos ocultos de procesamiento de texto.
- Se aplica técnicas de limpieza de texto.
- Se aplica técnicas de PLN para predecir la respuesta adecuada.
- Se envía la respuesta de vuelta.

Todo este proceso debe de suceder lo suficientemente rápido como para simular una conversación humana.

1.16. Topic modeling

El topic modeling o modelado de temas es un método estadístico y probabilístico que identifica tópicos o temas en textos. Esto se logra detectando patrones en una colección de documentos (un documento es una unidad de texto) llamado corpus y agrupando las palabras que aparecen en dichos documentos en temas (o tópicos). Los temas se definen como una función de densidad; dicha función representa la probabilidad de la ocurrencia de una palabra cualquiera dado cierto tema [13].

Es decir, un tema o tópico es la representación semántica de un grupo de palabras estadísticamente significativas dentro de un corpus.

Es muy utilizado en aprendizaje automático y PLN para análisis de texto, dado que la semántica también es representada en el modelo. Topic modeling es un concepto general, y varias técnicas específicas entran en esta clasificación. Algunos de estos algoritmos son los siguientes:

- Latent Semantic Analysis (LSA, por su siglas en inglés) o análisis semántico latente.
- Probabilistic Latent Semantic Analysis (PLSA, por su siglas en inglés) o análisis semántico probabilístico latente.
- Latent Dirichlet Allocation (LDA, por su siglas en inglés) o asignación latente de Dirichlet.
- Correlated Topic Model (CTM, por su siglas en inglés) o modelo de temas correlacionados.

1.17. Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation (LDA) es un modelo de tópicos de tipo generativo. Se puede entender a partir de su acrónimo:

- *Latente* significa oculto, algo que aún no se ha encontrado.
- *Dirichlet* es un tipo de distribución probabilística la cual (según el algoritmo) describe la distribución de las palabras en los temas. En este punto es esencialmente distinto al método LSA, por ejemplo, el cual asume otro tipo de distribución para manejar los datos.
- *Asignación* se refiere a la creación de los temas.

Topic modelling generativo es aquel que permite la clasificación de documentos aún no procesados una vez que el modelo fue entrenado (sin necesidad de utilizar el corpus completo). Este enfoque es especialmente útil para la incorporación de un corpus infinito (por ejemplo, el streaming de texto).

El algoritmo asume que cada palabra en un documento es generada a partir de un tópico que es tomado de una distribución de tópicos para cada documento. La distribución de tópicos para cada documento sigue el supuesto de una distribución de Dirichlet; es decir que la distribución de Dirichlet describe el patrón de frecuencia de cada una de las combinaciones de las palabras [14].

Ejemplo. En la figura 1.6 se puede ver cómo un documento está conformado de 10 palabras, cada una clasificada en un tópico específico de acuerdo al modelo LDA. Y se tiene una relación de porcentaje de relevancia a nivel de documento, tópico y palabra.

El algoritmo LDA hace dos suposiciones clave:

- Los documentos son una mezcla de temas.
- Los temas son una mezcla de palabras.

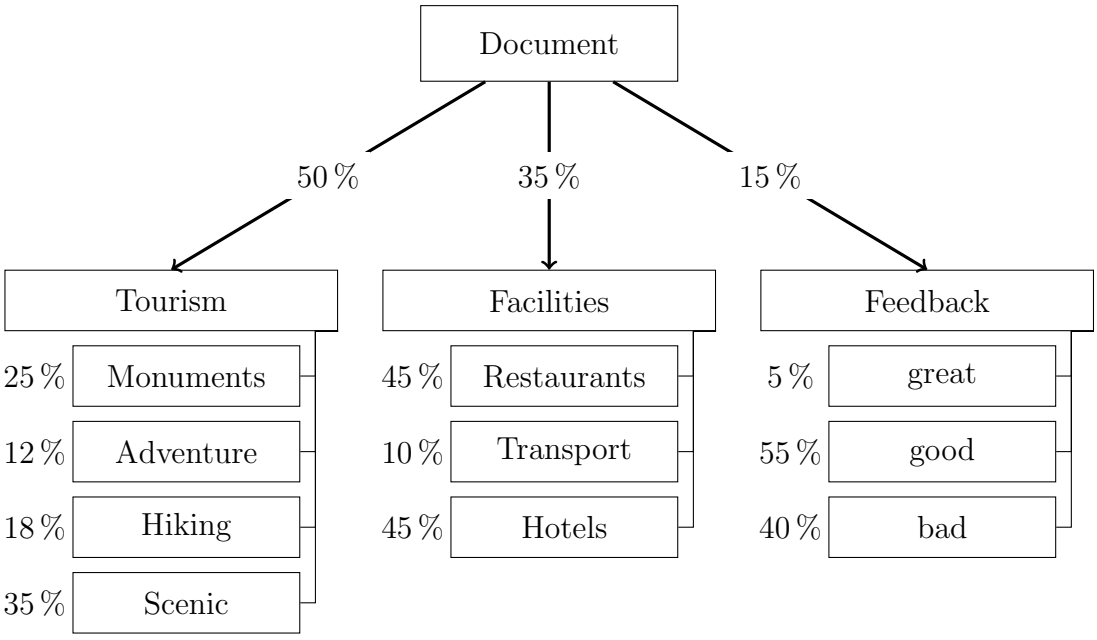


Figura 1.6: Topic Modelling using LDA. Fuente: tomado de [22].

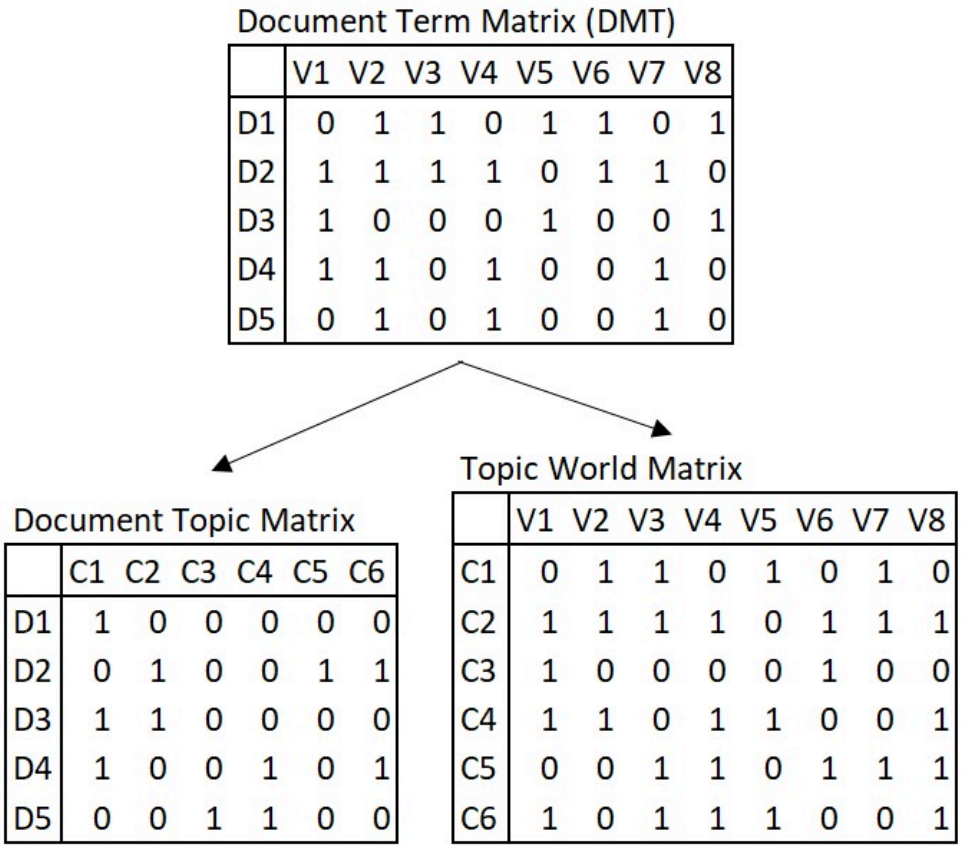
Entonces los temas que utilizan la distribución Dirichlet generan las palabras. En estadística eso significa que los documentos son la densidad de probabilidad de los temas y los temas son la densidad de probabilidad de las palabras.

¿Cuáles son los pasos de éste método?

Primero se debe hacer una limpieza y procesamiento de los datos. En este proceso se limpia el texto, se homologa y se generan tokens (que son las palabras del texto en diferentes combinaciones).

Luego se generan 3 tipos de representaciones de los datos, que se pueden ver como 3 diferentes matrices:

- Document Term Matrix (DTM): representa la relación entre los documentos (filas) y las palabras (columnas).
- Document Topic Matrix: representa la relación que hay entre los documentos (filas) y los temas (columnas).
- Topic World Matrix: representa la relación entre los temas (filas) y las palabras (columnas).



	V1	V2	V3	V4	V5	V6	V7	V8
D1	0	1	1	0	1	1	0	1
D2	1	1	1	1	0	1	1	0
D3	1	0	0	0	1	0	0	1
D4	1	1	0	1	0	0	1	0
D5	0	1	0	1	0	0	1	0

	C1	C2	C3	C4	C5	C6
D1	1	0	0	0	0	0
D2	0	1	0	0	1	1
D3	1	1	0	0	0	0
D4	1	0	0	1	0	1
D5	0	0	1	1	0	0

	V1	V2	V3	V4	V5	V6	V7	V8
C1	0	1	1	0	1	0	1	0
C2	1	1	1	1	0	1	1	1
C3	1	0	0	0	0	1	0	0
C4	1	1	0	1	1	0	0	1
C5	0	0	1	1	0	1	1	1
C6	1	0	1	1	1	0	0	1

Figura 1.7: Relación entre DTM, Document Topic Matrix y Topic World Matrix.

Dadas las tres estructuras, que se representan en la figura 1.7, el algoritmo puede generar relaciones entre documentos, temas y palabras. Éste es un proceso iterativo, y para la primera iteración se genera una asociación aleatoria. Por ejemplo:

Los documentos se representan en función de los temas y palabras de cada

tema (D. Blei y et. al, 2003).

$$
\begin{aligned}
D1 =&(v1(c5), v2(c3), v3(c1), v4(c2), v5(c5), v6(c4), v7(c7), v8(c1))\\
D2 =&(v'1(c2), v'2(c4), v'3(c2), v'4(c1), v'5(c2), v'6(c1), v'7(c5), v'8(c3),\\
&v'9(c7), v'10(c1))\\
D3 =&(v''1(c3), v''2(c1), v''3(c5), v''4(c3), v''5(c4), v''6(c1), \ldots, v''13(c1),\\
&v''14(c3), v''15(c2))\\
D4 =&(v'''1(c4), v'''2(c5), v'''3(c3), v'''4(c6), v'''5(c5), v'''6(c3), \ldots, v'''10(c3),\\
&v'''11(c7), v'''12(c1))\\
D5 =&(en''''1(c1), en''''2(c7), en''''3(c2), en''''4(c8), en''''5(c1), en''''6(c8), ...,\\
&en''''32(c3), en''''33(c6), en''''34(c5))
\end{aligned}
$$

Se pueden representar los documentos únicamente en función de los temas.

$$
\begin{aligned}
D1 =&c5 + c3 + c1 + c2 + c5 + c4 + c7 + c1\\
D2 =&c2 + c4 + c2 + c1 + c5 + c2 + c1 + c5 + c3 + c7 + c1\\
D3 =&c4 + c5 + c3 + c6 + c5 + c3 + \ldots + c3 + c7 + c1\\
D3 =&c1 + c7 + c2 + c8 + c1 + c8 + \ldots + c3 + c6 + c5
\end{aligned}
$$

También los temas en función de las palabras.

$$
\begin{aligned}
C1 =&v3 + v8 + v'4 + v'6 + v'10 + v''2 + v''6 + \ldots + v''13 + v''12 + v''''1 + v''''5\\
C2 =&v4 + v'1 + v'3 + v''15 + \ldots. + en''''3 + \ldots\\
C3 =&v2 + v'8 + v''1 + v''4 + v''14 + v'''3 + v'''6 + \ldots + v'''10 + v''''32 + \ldots
\end{aligned}
$$

A partir de una primera asignación, se genera una próxima iteración utilizando esta lógica:

- Se hace la suposición de que la asignación de todos los temas han sido correctos excepto para la palabra actual (siendo que el algoritmo va recorriendo palabra por palabra del texto y va asignando y corrigiendo en el recorrido del corpus).
- Calcula dos tipos de probabilidades diferentes, que son (p1) que es la proporción de palabras en el documento que están asignadas a cierto tema, y (p2) que es la proporción de asignaciones al tema de todos los documentos relacionadas a cierta palabra.
- Se reasigna la palabra actual a un nuevo tema, utilizando la probabilidad del tema más relevante para dicha palabra; esta probabilidad se obtiene por producto de las 2 probabilidades calculadas (p1 * p2).

El proceso iterativo converge al optimizar las matrices documento-tema y tema-palabra.

El proceso general de LDA se puede observar en la figura 1.8.

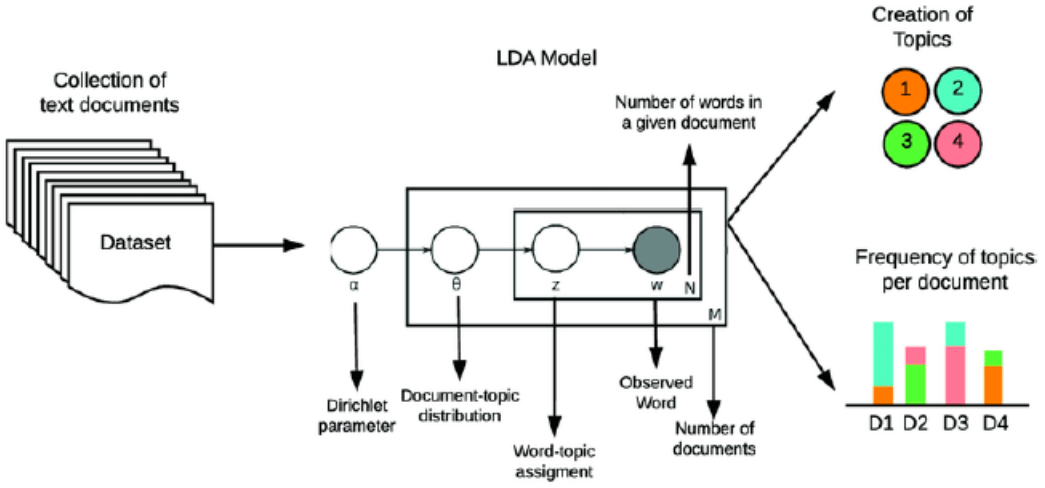


Figura 1.8: Text Mining of Open-Ended Questions in Self-Assessment of University Teachers: An LDA Topic Modeling Approach. Fuente: tomado de [23].

La implementación del método de LDA está disponible en distintas bibliotecas de lenguajes de programación. Una de las más populares es Gensin [25] la cual

es un conjunto de librerías escritas en lenguaje Python con implementaciones específicas de Topic Modeling para PLN.

1.18. Base de datos

Una base de datos es una colección bien organizada de información (datos con significado) que normalmente se guardan en un sistema informático. Una base de datos es gestionada por un sistema de gestión (o administración) de bases de datos (DBMS por sus siglas en inglés).

Las bases de datos requieren un lenguaje de consulta, por ejemplo el lenguaje de consulta estructurado (SQL por su siglas en inglés) para manejar los datos.

Básicamente existen dos formas de clasificar las bases de datos. La relacional y la no relacional. La primera es la base de datos tradicional.

- Relacional. La base de datos relacional se estructura (u organiza) como un conjunto de tablas (con columnas y filas); a los datos que almacena se les conoce como datos estructurados.
- No relacional. Una base de datos no relacional se utiliza para guardar y gestionar datos no estructurados y semiestructurados [15].

El modelo ACID (Atomicity, Consistency, Isolation and Durability) que es el acrónimo en español de Atomicidad, Consistencia, Aislamiento y Durabilidad; indican las reglas que debe de cumplir una base de datos relacional de calidad:

- Atomicidad: se refiere a ejecutar todas las instrucciones de forma satisfactoria o de lo contrario no ejecutar ninguna; es decir, una operación debe de ser completa y nunca quedarse a la mitad del proceso.
- Consistencia: propiedad que asegura que sólo comienzan aquellas operaciones que pueden terminar.

- Aislamiento: propiedad que asegura que una operación no puede afectar a otras. Así se pueden realizar dos operaciones sobre la misma estructura de forma independiente.

- Durabilidad: propiedad que asegura que una vez realizada una operación, persista la información a pesar del tiempo o fallas del sistema.

1.19. Base de datos no relacional

Una base de datos no relacional (también llamada noSQL) es aquella que no almacena datos estructurados.

El modelo CAP (el cual es un acrónimo que en español significa Disponibilidad, Consistencia y Tolerancia a Particiones) indica los principios que siguen los diferentes tipos bases de datos no relacionales [16].

- Disponibilidad: es la capacidad del sistema de estar conectado y disponible para la lectura y escritura de la información.

- Consistencia: es la capacidad de que la información que entregue el sistema esté libre de errores, que se mantenga la integridad de los datos y no haya contradicciones.

- Tolerancia a particiones: (también llamada tolerancia a fallas) es la capacidad del sistema de funcionar correctamente mientras tenga instancias varias en diferentes lugares físicos o virtuales, esto hará que sea más escalable y tolerante a fallas; pero el reto es la sincronización entre las distintas instancias.

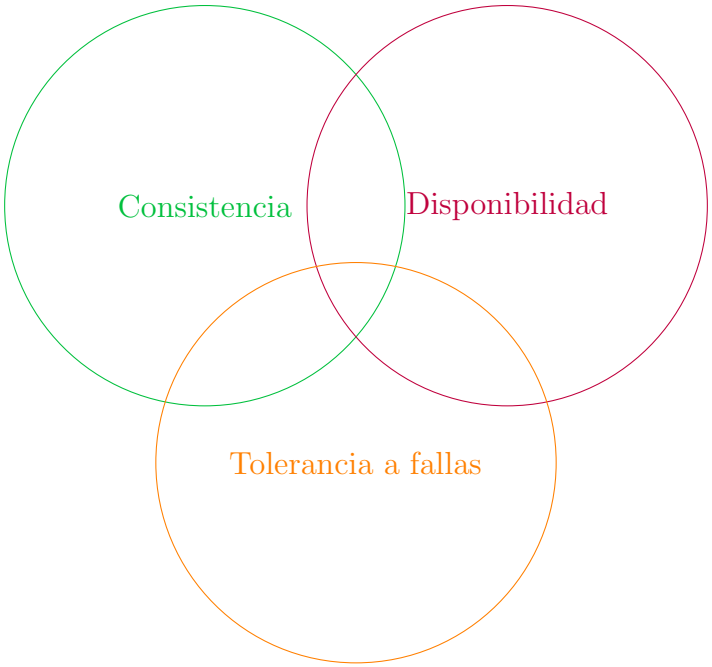


Figura 1.9: O Teorema CAP! Fuente: tomado de [24].

El modelo CAP, como se observa en la figura 1.9, establece que una base de datos no relacional no puede cubrir de manera completa las tres características.

Algunos tipos de bases de datos no relacionales, son las siguientes [17]:

- Documental: son bases de datos semi estructuradas, pues almacenan la información en unidades llamadas documentos, los cuales están en formato semi estructurado como es xml o json. Son muy flexibles y escalables. Ejemplos: MongoDB, CouchDB.
- Clave-valor: son bases de datos no estructuradas, que se basan en tener un conjunto de clave valor, donde la clave es un valor único y el valor es texto libre. Se basa en el modelo Random Access Machine (RAM por sus siglas en inglés) el cual tiene un tiempo constante de acceso a la información; es principalmente este el motivo por el que las base de datos de clave-valor dan prioridad a la velocidad de acceso. Ejemplos: Redis, Voldemorte.

- Columnar: son bases de datos que se basan en una variante del modelo entidad relación; donde el manejo principal es agrupar por atributos en vez de tuplas; lo cual provoca que se puedan obtener grandes cantidades de información de forma rápida sobre una misma tabla; y por el contrario es muy lento para hacer relaciones complejas con otras tablas y para escribir información. Ejemplos: Cassandra, Project Gemini.

- Grafos: son aquellas bases de datos que se basan en modelos de gráficas dirigidas; donde se tiene una serie de nodos que se relacionan entre sí por medio de aristas. Su principal ventaja es que el modelo es sencillo pero muy poderoso para encontrar relaciones complejas. Ejemplos: Neo4j, InfoGrid, Trinity.

Capítulo 2

Presentación del sistema

El objetivo del sistema *Clasificador automático de artículos web* es ayudar a gestionar una biblioteca personal de artículos web, la cual será organizada de forma automática dependiendo de su propio contenido. Es decir, el usuario debe de ir agregando artículos web de su interés al sistema, y éste se encarga de guardar y clasificar dichos artículos de acuerdo al contenido de los mismos.

En un principio la biblioteca se encuentra vacía. Se recomienda que los artículos sean de un tema relacionado (entre más relacionado mejor) y en un mismo idioma (inglés), de esta forma se tendrá una mayor probabilidad de que el algoritmo de clasificación funcione adecuadamente. El sistema funciona correctamente sólo con contenido en inglés; si se utiliza otro idioma la calidad dicha clasificación será baja.

¿Cómo funciona la clasificación de artículos?

Cada artículo se clasifica en 10 categorías diferentes, y las categorías se pueden compartir entre artículos; de tal forma que entre más artículos sean clasificados mayor será la probabilidad de que existan categorías que compartan artículos.

Ejemplo.

- Se agrega el artículo *A Simple Guide to Machine Learning Visualisations* (`https://medium.com/towards-data-science/a-simple-guide-to-machine-learning-visualisations-6c808ac925dd`).
- El sistema lo clasifica en las categorías: *model, positive, diabetes, yellowbrick, learning, machine, visualisations, dataset, negative, curve.*

- Se agrega el artículo *Machine Learning Rules of Thumb* (https://towardsdatascience.com/machine-learning-rules-of-thumb-b50232b4b2f8).

- El sistema lo clasifica en las categorías: *regression, size, learning, weights, linear, neural, data, model, machine, models.*

Como se puede ver, las categorías que se tienen en común son: *model*, *learning*, *machine.*

Entonces la biblioteca tiene 2 artículos y 17 categorías. Donde 3 de esas categorías tienen 2 artículos asociados y las restantes sólo 1.

Esta asociación sencilla permite una búsqueda de artículos basada en categorías. Es decir, que el usuario puede acceder a los artículos guardados en su biblioteca a partir de la relación de categorías que hay entre ellos.

Ejemplo.

- Se agrega el artículo "*Machine Learning and Progressive Organizing — NOW Is The Time*" (https://medium.com/@TCWstrategies/machine-learning-and-progressive-organizing-now-is-the-time-e9ce2a429b1a).

- El sistema lo clasifica en las categorías: *union, workers, organizing, people, new, learning, machine, scale, wages, moment.*

- Se puede ver que hay una coincidencia con los otros 2 artículos anteriores de sólo 2 categorías: *machine* y *learning.*

Ahora se tienen 3 artículos diferentes en la biblioteca, y entre ellos se puede ver una relación entre categorías, que se describe en la Tabla 2.1.

	machine	learning	model
A Simple Guide to Machine Learning Visualisations	Sí	Sí	Sí
Machine Learning Rules of Thumb	Sí	Sí	Sí
Machine Learning and Progressive Organizing	Sí	Sí	No

Tabla 2.1: Relación entre artículos y categorías comunes

En realidad cada artículo tiene más categorías, pero basándonos en estas tres se pueden hacer las siguientes combinaciones de búsqueda:

- Si se busca por *machine* y *learning* el resultado son 3 artículos.
- Si se busca por *machine* y *model* el resultado son 2 artículos.
- Si se busca por *learning* y *model* el resultado son 2 artículos.
- Si se busca por *machine*, *learning* y *model* el resultado son 2 artículos.

Además de la búsqueda a partir de categorías, también se pueden buscar los artículos a través de su título. Es decir, que se puede utilizar alguna palabra o frase para que el sistema identifique a todos los artículos de la biblioteca cuyo título coincide parcial o totalmente con la palabra o frase, sin importar mayúsculas ni minúsculas.

Ejemplo.

- Si se busca la frase *machine learning* el sistema arroja 3 artículos resultantes.
- Si se busca la palabra *visual* el sistema arroja 1 artículo resultante.
- Si se busca la palabra *RULES* el sistema arroja 1 artículo resultante.
- Si se busca la palabra *time* el sistema arroja 1 artículo resultante.

¿Cómo se gestiona la biblioteca?

El sistema clasifica y guarda artículos en una misma acción, es decir siempre que se agrega un artículo se agrega al menos una categoría (a menos que todas las categorías del articulo nuevo ya existan previamente). Cada vez que se clasifica un artículo, el sistema le asigna un identificador único, el cual es indicado al usuario. Este identificador puede ser consultado en cualquier momento.

Ejemplo.

- Cuando se guardó el artículo *Machine Learning and Progressive Organizing — NOW Is The Time* el sistema le asigna el identificador 6236e484993b5b83f84b8616.

El límite de artículos permitidos en la biblioteca está dado por un número constante que se configura en el código del sistema. El valor predeterminado es 200, y este número se puede modificar a preferencia del usuario.

Cuando se llega al límite y no se pueden agregar más artículos el sistema devuelve un mensaje indicando que se tienen dos opciones:

- Aumentar el límite de artículos permitidos.
- Eliminar algún artículo de la biblioteca.

Para eliminar un artículo se debe de indicar su identificador, y esta acción sólo se puede hacer de un artículo a la vez. Cada vez que se elimina un artículo, la cantidad de artículos en la biblioteca se disminuye en 1, así como todas las categorías asociadas a dicho artículo.

Ejemplo.

- Se elimina el artículo con el identificador 6236e484993b5b83f84b8616.
- Las categorías *machine* y *learning*, que tenían relacionados 3 artículos, ahora tendrán solo 2.
- Las demás categorías del artículo eliminado desaparecen de la biblioteca, dado que no hay más artículos relacionados a ellas.

También se pueden obtener las categorías de la biblioteca, de dos formas diferentes.

- Obtener las categorías con mayor número de artículos relacionados, ordenadas de mayor a menor.
- Obtener las categorías con menor número de artículos relacionados, ordenadas de menor a mayor.

En ambos casos, el usuario puede elegir la cantidad de categorías que quiere ver, y el sistema provee dicha información con el nombre de la categoría y la cantidad de artículos relacionados.

Ejemplo.
Existen en la biblioteca las siguientes categorías con sus respectivas cantidades de artículos relacionados.

data (34), Python (23), machine (12), learning (10), visualization (5), math (2), graph (1)

Y se realizan las 2 formas de búsqueda de categoría:

- Primeras 3 categorías con mayor cantidad de artículos: *data (34), Python (23), machine (12).*
- Primeras 2 categorías con menor cantidad de artículos: *graph (1), math (2).*

¿Cómo se accede al sistema?

Toda la interacción con el sistema se hace por medio de un chatbot integrado a la cuenta personal de usuario en la plataforma de Telegram, de tal forma que cada acción que se quiera realizar se hace por medio de comandos escritos que se envían por medio de mensajes de texto al chatbot, y éste a su vez responde a cada comando con un mensaje de vuelta que contiene el resultado de la operación.

Para que el chatbot esté a la escucha de comandos se debe de ejecutar el código del sistema en la computadora personal del usuario.

2.1. Configuración del sistema

Los requisitos de infraestructura que se requieren para la ejecución de sistema son los siguientes.

- Tener instalado Python 3.8 (`https://www.Python.org/downloads/release/Python-380/`).
- Tener instalado el navegador Google Chrome en alguna de sus ultimas versiones estables (`https://www.google.com/chrome/`).
- Tener instalado Docker en su ultima versión (`https://docs.docker.com/engine/install/`).

2.1.1. Creación y configuración del ambiente virtual

El ambiente virtual de Python es una capa intermedia de infraestructura que separa los recursos propios de la computadora con los recursos de Python. Es una forma de aislar las bibliotecas específicas de Python, sin afectar la instalación principal. Los pasos son los siguientes.

- Descarga del código del sistema.
  - Descarga el código desde `https://github.com/yaotzinvr/clasificador_articulos_web_topic_modeling.git`
  - Guárdalo en tu ubicación de preferencia.
  - Ejemplo: */clasificador_web_topic_modeling*.
- Creación de ambiente virtual.
  - Sitio oficial: `https://docs.Python.org/es/3.8/library/venv.html`

- Crea el ambiente virtual dentro de la carpeta del proyecto. (El proceso de creación del ambiente virtual se puede encontrar en el enlace anterior).
- Ejemplo: */clasificador_web_topic_modeling/venv*.

- Instalación de paquetes (en el ambiente virtual).
  - Instala los paquetes del archivo */clasificador_web_topic_modeling/requirements.txt*
  - Ejecución de ejemplo: pip install -r /requirements.txt

  Para asegurarse de que el ambiente virtual esté funcionando correctamente, realiza lo siguiente:

  - Activa el ambiente virtual (viene indicado aquí: `https://docs.Python.org/es/3.8/library/venv.html`).
  - Ejecuta el comando *Python –version* y la salida debe de ser la versión 3.8 de Python. Ejemplo: Python 3.8.10
  - Ejecuta el comando *Python* para entrar a la terminal interactiva del ambiente virtual.
  - Ejecuta el comando *import telebot, pymongo, selenium, webdriver_manager, gensim* y no debe de indicar algún error, lo que significa que las bibliotecas se han instalado correctamente en el ambiente virtual.
  - Ejecuta *exit()* para salir de la terminal interactiva.

2.1.2. Creación de chatbot

El chatbot de Telegram se ejecuta sobre la plataforma personal del usuario. Para esto se debe tener una cuenta de Telegram (`https://telegram.org/`). Para crear un bot personal se deben seguir las instrucciones del sitio oficial (`https://core.telegram.org/bots/features#creating-a-new-bot`).

Al finalizar el proceso se obtiene un token de acceso al API del chatbot. Ejemplo: 110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw.

2.1.3. Creación y configuración de la BD

Para guardar la información de la biblioteca el sistema utiliza MongoDB (https://www.mongodb.com/) la cual es una base de datos no relacional de tipo documental. Para utilizarla de manera local la forma más sencilla es usar Docker, que es una tecnología de virtualización que ya se instaló previamente.

La descarga, creación y configuración se hará desde comandos de consola.

Ejecuta el comando:
docker run -d -p 27017:27017 –name mongodb -v mongodb-data:/data/db
-e MONGODB_INITDB_ROOT_USERNAME=usuario -e
-e MONGODB_INITDB_ROOT_PASSWORD=pass mongo:5.0.6

No es necesario hacer algún cambio en los datos de ejemplo, pero si se quiere configurar de manera distinta se puede hacer.

El comando *docker run* (https://docs.docker.com/engine/reference/commandline/run/) indica la creación de un contenedor a partir de una imagen, con las siguientes características:

- *-d* : indica que el contenedor se ejecuta en segundo plano.
- *-p* : indica el mapeo entre el puerto de la computadora con el del contenedor, siendo el 27017 el puerto predeterminado para MongoDB. También se puede mapear a un puerto distinto de la computadora; por ejemplo 500:27017, entonces la base de datos se accede en el puerto 500.
- *–name* : indica el nombre que se le asigna al contenedor, que además es un identificador único. Así que si se quiere tener un contenedor diferente se debe elegir otro nombre.
- *-v* : indica el volumen del contenedor; es decir donde se guardarán los datos del contenedor, lo que permite que no se pierdan los datos al

reiniciar su ejecución. Se hace un mapeo entre el almacenamiento local y el del contenedor; a la izquierda indica el nombre del volumen. (*mongo-data*) y a la derecha la dirección interna del contenedor. Si se quiere tener un segundo contenedor se recomienda asignarle un volumen distinto, de lo contrario se compartirán datos entre ambos contenedores

- *-e* : indica una variable de entorno. Donde *MONGODB_INITDB_ROOT_USERNAME* y *MONGODB_INITDB_ROOT_PASSWORD* son variables de entorno que la imagen usa de manera predeterminada.

- *mongo:5.0.6* : indica el nombre de la imagen de MongoDB (`https://hub.docker.com/_/mongo`) que se descarga así como su versión.

Con dicho comando comenzará a descargarse la imagen docker oficial de MongoDB (a menos que ya se haya hecho previamente), posteriormente se creará un contenedor y finalmente se ejecutará dicho contenedor.

Para comprobar que el contenedor se está en ejecución se utiliza: *docker ps*

La salida de dicho comando indica todos los contenedores que se están ejecutando.

Ejemplo.

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
bdfe2074fd62 mongo:5.0.6 "docker-entrypoint.s. . . "10 seconds ago Up 8 seconds
0.0.0.0:27017->27017/tcp, :::27017->27017/tcp mongodb

Para detener el contenedor se debe reiniciar la compuradora, reiniciar el servicio de docker o ejecutar el comando *docker stop <identificador del contenedor>*. Ejemplo: *docker stop mongodb*

El comando *docker ps -a* muestra una lista completa de los contenedores (detenidos y en ejecución) y ahí debe de aparecer el nombre del contenedor cuando no se está en ejecución.

Para ejecutar un contenedor existente se ejecuta *docker start <identificador del contenedor>*. Ejemplo: *docker start mongodb*

Si se quiere eliminar el contenedor, se ejecuta *docker rm <identificador del contenedor>*. Ejemplo: *docker rm mongodb*

Si se quiere eliminar el volumen del contenedor, se ejecuta *docker volume rm <identificador del volumen>*. Ejemplo: *docker volume rm mongodb-data*. Se puede validar la existencia del volumen con el comando *docker volume ls*. Hay que tener en cuenta que al eliminar el volumen se elimina la base datos; es decir que aunque se elimine un contenedor, se puede crear otro que utilice el mismo volumen y los datos no se perderán.

Para configurar la base de datos, primero se debe asegurar que el contenedor se está ejecutando. Ejecuta el comando *docker exec -it mongodb mongo*.

El comando *docker exec* (`https://docs.docker.com/engine/reference/commandline/exec/`) ejecuta un comando específico sobre un contenedor en ejecución, con las siguientes características:

- *-it* : indica que se ejecute una terminal interactiva.
- *mongodb* : indica el nombre del contenedor.
- *mongo* : indica el comando a ejecutar sobre el contenedor. El comando mongo (`https://www.mongodb.com/docs/manual/reference/program/mongo/`) inicia la terminal de la base de datos MongoDB.

Este comando abre una segunda terminal, con la cual se puede interactuar directamente con el motor de MongoDB dentro del contenedor.

Dentro de la segunda terminal interactiva ejecuta el comando *db.getSiblingDB('topic_modeling').createCollection('articulos')* con lo cual se crea una nueva base de datos llamada *topic_modeling* y a su interior una colección llamada *articulos*. La salida de la ejecución debe ser { *"ok": 1* } que indica su correcta creación.

Para comprobar la creación de la base de datos y su colección, se ejecuta en orden lo siguiente:

- *show dbs* : muestra las bases de datos existentes y el tamaño de cada una.
- *use topic_modeling* : indica que se usa como referencia la base de datos emphtopic_modeling para los próximos comandos.
- *show collections* : muestra las colecciones de la base de datos seleccionada previamente.

Luego ejecuta el comando:

```
db.getSiblingDB('topic_modeling').createUser(
  user: 'usuario',
  pwd: 'pass',
  roles: [
    {
      role: 'root',
      db: 'admin',
    },
  ],
)
```

el cual crea un nuevo usuario (de nombre *usuario* y contraseña *pass*) y le asigna el rol de administrador para la base de datos *topic_modeling*. El resultado del comando debe de ser *Successfully added user ...*

Por último ejecuta el comando *exit* para salir de la terminal interactiva del contenedor.

Ahora que se ha creado y configurado la base de datos, puedes hacer una prueba de conexión usando Python. Realiza lo siguiente:

- Ejecuta el comando *Python* dentro del ambiente virtual para entrar a la terminal interactiva de Python.
- Ejecuta el comando *import pymongo*; *pymongo.MongoClient('mongodb://usuario:pass@127.0.0.1:27017/topic_modeling').server_info()['version'] ; exit()*, donde:
  - *import pymongo* : indica que se utilizará la biblioteca pymongo.
  - *pymongo.MongoClient* : método que obtiene una nueva conexión a la base de datos. Se obtiene un dato de tipo diccionario, y una de las llaves es versión.
  - *mongodb://usuario:pass@127.0.0.1:27017/topic_modeling* : cadena de conexión a la base de datos MongoDB (`https://www.mongodb.com/docs/manual/reference/connection-string/`), que se usa como argumento para el método *MongoClient*.
  - *exit()* : indica la salida de la terminal interactiva de Python.

La salida del comando anterior debe de ser la versión de la base de datos MongoDB ejecutándose dentro del contenedor (ejemplo: '5.0.6'), con lo cual se comprueba que desde el ambiente virtual se tiene acceso a la base de datos *topic_modeling*.

2.2. Variables globales del sistema

El archivo de configuración es *config.py* el cual contiene las variables globales que se usan por todos los módulos del sistema. Las variables se pueden ver en

la Tabla 2.2.

Variable	Descripción
LIMITE_MAX_ARTICULOS	Es la cantidad máxima de artículos permitida en tu biblioteca
CATEGORIAS_EXCLUIR	Son las categorías a excluir de la clasificación de los artículos
BOT_TELEGRAM_URI	Es el token de conexión a la API del chatbot de Telegram
MONGO_URI	Es la cadena de conexión a la base de datos de MongoDB
BD	Es el nombre de la base de datos de MongoDB
COLLECTION	Es el nombre de la colección de MongoDB

Tabla 2.2: Variables del sistema

2.3. Ejecución del sistema

Una vez que se ha configurado de manera correcta la infraestructura, el ambiente virtual, la base de datos y el archivo de configuración se puede ejecutar el sistema, de la siguiente forma.

- Ejecutar el contenedor que contiene la base de datos.
- Ejecutar el ambiente virtual.
- Ejecutar, dentro del ambiente virtual y en la ruta donde se encuentra el código fuente, el comando *Python chatbot.py*.

Este proceso de debe hacer cada vez que se quiera iniciar el chatbot.
El sistema comienza a ejecutarse y en la terminal se puede ver una serie de mensajes, como el siguiente: ... *TeleBot: "Started polling."*

```
2022-04-05 18:00:26,490 (__init__.py:663 MainThread) INFO - TeleBot: "Started polling."
2022-04-05 18:00:26,490 (util.py:84 PollingThread) DEBUG - TeleBot: "Received task"
2022-04-05 18:00:26,490 (apihelper.py:88 PollingThread) DEBUG - TeleBot: "Request: method=get url=https://api.telegram.org/bot5083545517
:{TOKEN}/getUpdates params={'offset': 1, 'timeout': 20, 'long_polling_timeout': 20} files=None"
```

Figura 2.1: Inicialización del chatbot

En el chatbot escribe el mensaje */info* el sistema debe responder un mensaje indicando los comandos para interactuar con el sistema. Esto indica que el sistema se está ejecutando correctamente, como se muestra en la figura 2.2.

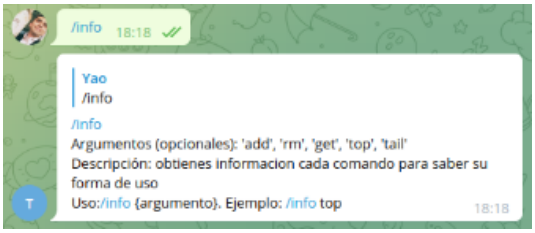


Figura 2.2: Comando /info

2.4. Uso del sistema

Los comandos para interactuar con el sistema son los siguientes

- */add*: agrega un nuevo artículo a la biblioteca.
- */rm*: elimina un artículo de la biblioteca.
- */get*: busca artículos de acuerdo a un filtrado por título.
- */getc*: busca artículos de acuerdo a un filtrado por categorías.
- */top* y */tail*: visualiza categorías de la biblioteca.

A cada comando el sistema responde un mensaje específico, indicando que se ha ejecutado correctamente. En caso de haber un error, el sistema responde un mensaje con el formato:

> ¡Lo siento! Hubo un error inesperado *~*
> ERROR <mensaje de error específico>

En donde el <*mensaje de error específico*> depende del método específico dentro del código donde ocurrió la excepción. Los motivos para alguna excepción no controlada del sistema son los siguientes:

- Algún paso erróneo o incompleto en la configuración de la infraestructura.
- Algún paso erróneo o incompleto en la configuración del ambiente virtual.
- Algún paso erróneo o incompleto en la configuración de la base de datos.

- Algún paso erróneo o incompleto en la configuración del archivo de variables globales.
- Algún problema con la conexión de internet.
- Algún problema con el servidor del artículo web que se intenta agregar.

Ejemplo:

> ¡Lo siento! Hubo un error inesperado *~*
> ERROR (servicio_add)(get_categorias_all)(get)(get_client)Authentication failed., full error: {'ok': 0.0, 'errmsg': 'Authentication failed.', 'code': 18, 'codeName': 'AuthenticationFailed'}

El cual indica que hay un error en los métodos anidados *(servicio_add)(get_categorias_all)(get)(get_client)*. Aquí el método de la izquierda consume al de la derecha, así que la excepción ocurrió en el último método. En este caso el error específico es *Authentication failed...* que indica que la autenticación a la base de datos no fue exitosa, por lo que se debe de revisar en el archivo *config.py* la cadena de conexión a la base de datos y posiblemente repetir los pasos de creación y configuración de base de datos.

2.4.1. Agregar artículos

Se debe agregar un par de artículos para iniciar a poblar la biblioteca de información. Ejecuta el comando */add <url del artículo>*.

Ejemplo, observar la figura 2.3:

/add `https://towardsdatascience.com/introduction-to-linear-programming-in-Python-9261e7eb44b`

La respuesta del sistema es la siguiente:

Guardado correctamente.
Título: Introduction to Linear Programming in Python I.
Solvers II.
Variables
ID: 624e73d965bc9cb0f3f23175
Categorías:
solver,linear,variables,solution,optimal,power,tools,constraints,
programming,army

Figura 2.3: Comando /add: registro de nuevo artículo

Como se muestra en la figura 2.4, si el mismo artículo se quiere volver a agregar, el sistema indica que ya se ha agregado previamente.

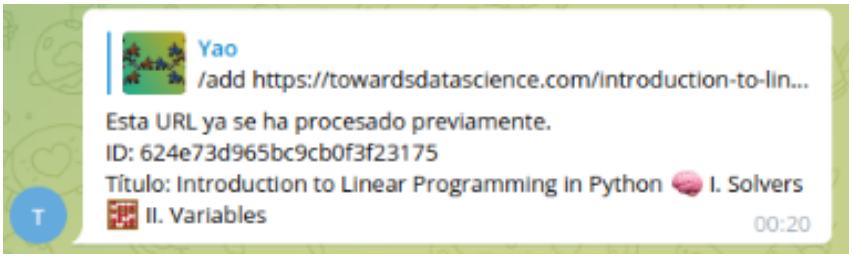


Figura 2.4: Comando /add: artículo ya existente

Una vez que se han agregado un par de artículos a la biblioteca, se puede ver en funcionamiento los siguientes comandos.

2.4.2. Visualiza categorías

Ejecuta el comando */top <n>* para obtener las n categorías con mayor número de artículos relacionados. Ejemplo: */top 5*. Los resultados son los siguientes:

(4) data
(2) like
(1) entropy
(1) distribution
(1) cross

Ejecuta el comando */tail <n>* para obtener las n categorías con menor número de artículos relacionados. Ejemplo: */tail 5*. Los resultados son lo siguientes:

(1) entropy
(1) distribution
(1) cross
(1) true
(1) predicted

2.4.3. Buscar artículos

Para poder obtener los artículos de la biblioteca se tienen 2 comandos diferentes: */get* y */getc*

El comando */get <texto>* realiza una búsqueda del texto indicado sobre el título de todos los artículos de la biblioteca. Se obtienen todos los artículos con al menos una coincidencia del texto en su título. La búsqueda no hace distinción entre minúsculas y mayúsculas.

Ejecuta el comando */get <texto>* para obtener algunos artículos. El resultado tiene el siguiente formato:

<N> Resultado(s)

- (<identificador del artículo>) (<Título del artículo>) <url del artículo>

Donde <N> es el número de artículos coincidentemente con la búsqueda, seguido de una lista de los artículos cada uno con su identificador, título y url.

Ejemplo.
Ejecuta */get Python* y se obtiene:

2 Resultado(s)

- (624e73d965bc9cb0f3f23175)(Introduction to Linear Programming in Python I. Solvers II. Variables) https://towardsdatascience.com/introduction-to-linear-programming-in-Python-9261e7eb44b

- (625111b91d97a89545cb4260) (Python : *args and **kwargs) https://medium.com/@sunilrana123/Python-args-and-kwargs-2b8c9d774ce5

El siguiente comando es */getc <categorías>* que realiza una búsqueda de todos aquellos artículos que coincidan con las categorías indicadas, separadas por comas. Es decir, que se obtienen todos los artículos que coincidan con todas las categorías especificadas. En caso de que alguna de las categorías indicadas no exista el sistema devuelve el siguiente mensaje:

Estas categorías no con válidas: <categorías>

dónde *<categorías>* son las categorías indicadas en el comando que no existen en la biblioteca.

Una ejecución exitosa da como resultado el mismo que el comando */get <texto>*

Ejemplo.

/getc like y se obtiene como resultado:

> 2 Resultado(s)
> - (624a63794742c6bd66bf2ef3)(Dash is Deeper than Dashboards Why is Dash so revolutionary? How does Dash work?) `https://aiqc.medium.com/dash-is-deeper-than-dashboards-5ab7414f121e`
> - (625111b91d97a89545cb4260) (Python : *args and **kwargs) `https://medium.com/@sunilrana123/Python-args-and-kwargs-2b8c9d774ce5`

/getc data,like y se obtiene como resultado:

> 1 Resultado(s)
> - (624a63794742c6bd66bf2ef3)(Dash is Deeper than Dashboards Why is Dash so revolutionary? How does Dash work?) `https://aiqc.medium.com/dash-is-deeper-than-dashboards-5ab7414f121e`

2.4.4. Elimina artículos

Para eliminar un artículo se debe de ejecutar el comando */rm <identificador>*. El identificador del artículo se puede obtener con alguno de los comandos */add*, */get* o */getc*.

Ejecuta el comando */rm <identificador>*. La respuesta debe decir *Eliminado correctamente* como en la figura 2.5.

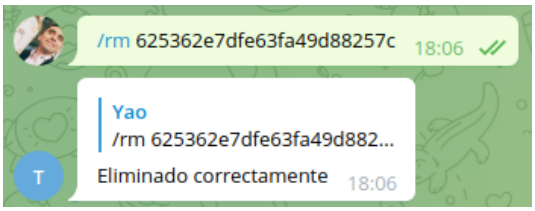


Figura 2.5: Comando /rm

2.4.5. Reorganiza biblioteca

El comando */build* hace una recategorización de todos los artículos de la biblioteca. El sistema toma a cada artículo guardado y le asigna 10 categorías de acuerdo a su contenido. Este comando se utiliza cuando se quiere excluir una o varias categorías.

En el archivo de configuración del sistema se tiene la variable *CATEGORIAS_EXCLUIR* la cual es una lista de categorías. Dichas categorías no serán tomadas en cuenta cuando el sistema asigne las categorías a los artículos; es decir, es una lista negra.

La forma correcta de utilizar el comando */build* es en combinación de la variable de configuración *CATEGORIAS_EXCLUIR.* Así que cada vez que se quiera ignorar una o varias categorías de la biblioteca se sigue el siguiente proceso:

1. Detener el sistema (en caso de que se esté ejecutando).
2. Agrega la o las categorías a ignorar a la variable global. *CATEGORIAS_EXCLUIR* del archivo de configuración.
3. Ejecuta el sistema.
4. Ejecuta el comando */build*.
5. Verifica que la o las categorías hayan desaparecido de la biblioteca (con el comando */top* o */tail*).

El comando /build tiene efecto sobre todos los artículos de la biblioteca y sobre las categorías. Es decir, no solo afecta a aquellos artículos que se relacionen con alguna de las categoría a excluir, sino que la clasificación de todos los artículos será distinta, por la propia naturaleza del algoritmo de de clasificación.

Ejemplo.
Ejecuta el comando */top 5* para obtener las 5 categorías con mayor número de artículos relacionados. Ejemplo de resultado:

(4) data
(2) like
(1) entropy
(1) distribution
(1) cross

Ahora, detén la ejecución del sistema y agrega dentro del archivo de configuración del sistema 2 categorías a excluir. Ejemplo:

$$CATEGORIAS\_EXCLUIR = [\text{'entropy','cross'}]$$

Ejecuta el sistema nuevamente y luego el comando */build*. El resultado del comando será: *Todos los artículos se han categorizado correctamente*

Para confirmar que se han excluido dichas categorías, ejecuta nuevamente el comando */top 5*. Ejemplo de resultado:

(4) data
(2) function
(1) distribution
(1) predicted
(1) probability

Además, ejecuta el comando */getc entropy,cross*

Lo cual debe de responder el mensaje: *Estas categorías no con válidas: entropy, cross*

De lo anterior se pueden observar 2 detalles.

- Las dos categorías que se han excluido ya no existen en la biblioteca.
- La cantidad de artículos relacionados de las demás categorías ha sido afectada.

Esto se debe a que con el comando */build* todos los artículos se han clasificado de forma distinta.

Capítulo 3

Análisis y diseño

Nombre del sistema

- Clasificador automático de artículos web.

Objetivo del sistema

- Clasificar automáticamente artículos web con base en su contenido, con el fin de construir una biblioteca personal, por medio una interfaz de usuario sencilla y de fácil acceso.

Características requeridas

- Alta usabilidad: la forma de interactuar con el sistema debe de ser intuitiva y sencilla.
- Configuración mínima: la configuración inicial debe de ser lo más accesible posible.
- Acceso rápido: la forma de acceso debe ser lo más sencilla posible.
- Mínima administración: el mantenimiento y administración debe de ser mínimo y centralizado.
- Personalizable: debe de ser lo suficientemente flexible para adaptarse a preferencias y necesidades particulares de cada usuario
- Escalable: el usuario debe tener la opción de extender la funcionalidad del sistema.

Definición de alcances

- Ejecución: de manera local en una computadora personal.

- Acceso: personal, el usuario crea y controla todas sus cuentas y contraseñas.
- Interacción con usuario: por medio de un bot de Telegram utilizando comandos simples.
- Base de datos: MongoDB como motor de base de datos, implementada con una imagen de Docker.
- Densidad de información: se pueden agregar como máximo 200 artículos al sistema de manera predeterminada, y se puede modificar la cantidad.
- Fuente de información: el sistema puede clasificar cualquier artículo de la página `https://medium.com` (Medium incluye una basta colección: 1,385,000 nuevos artículos publicados por mes) y de `https://www.springer.com` (Springer Science+Business Media es una de las editoriales científicas más importantes del mundo, con más de 6500 nuevos libros publicados cada año).
- Uso público: el código del sistema es de uso público y puede ser extendido y usado para agregar funcionalidades específicas.

Casos de uso

- El usuario solicita saber cuáles son los comandos que puede utilizar y la forma de usarlos.
    - Ejecución: el sistema obtiene la información del uso de los comandos.
    - Salida: comandos disponibles y forma de uso.
- El usuario solicita agregar a la biblioteca un artículo web.
    - Entrada: liga del artículo.
    - Ejecución: el sistema lee el contenido del artículo, genera el modelo de clasificación, asigna categorías y guarda toda la información en la base de datos.

    - Salida: identificador del artículo guardado y modelo de clasificación.
    - Excepciones: (1) el artículo no puede ser leído; (2) ya no se pueden agregar más artículos a la biblioteca; (3) el artículo ya se ha agregado previamente.

- El usuario solicita eliminar de la biblioteca un artículo web.
    - Entrada: identificador del artículo.
    - Ejecución: el sistema elimina de la base de datos el artículo.
    - Salida: notificación de que se eliminó correctamente.
    - Excepciones: (1) el identificador del artículo no existe.

- El usuario solicita la búsqueda de artículos web en la biblioteca.
    - Entrada: texto o lista de categorías.
    - Ejecución: el sistema busca a todos los artículos cuyo título coincide parcial o totalmente con el texto indicado; o con las categorías indicadas.
    - Salida: identificador, título y liga de los artículos resultantes de la búsqueda.
    - Excepciones: (1) alguno de las categorías indicadas no existe.

- El usuario solicita saber las categorías con mayor o menor cantidad de artículos relacionados.
    - Entrada: orden y número de categorías a mostrar.
    - Ejecución: el sistema recolecta todos las categorías existentes de la base de datos e identifica la cantidad de artículos que se relacionan a cada una.
    - Salida: categorías ordenadas con su cantidad de artículos.

- El usuario solicita categorizar los artículos de su biblioteca.

    - Ejecución: el sistema obtiene el contenido de cada artículo, genera el modelo de clasificación, asigna las nuevas categorías y guarda la información.
    - Salida: notificación de que se ha categorizado correctamente.

Comandos del sistema

- Todos los comandos comienzan con una diagonal (/) .
- Algunos comandos permiten más de un parámetro y deberán estar separados por comas (,) .
- El tiempo de ejecución de cada comando varía de acuerdo a factores como: conexión a internet, velocidad de servicios de Telegram, recursos de la computadora donde se ejecuta, etc.
- En cada comando de entrada el sistema regresa un mensaje de salida una vez que finalice la ejecución.
- Se pueden ejecutar nuevos comandos aunque no haya terminado de ejecutar el anterior y el sistema los irá encolando y ejecutando en orden; aunque se recomienda esperar a que cada comando termine de ejecutarse para mandar otros nuevos.

Caso de uso	Comando	Opciones	Salida
Solicitar información	/info <comando>	comando: (opcional) nombre del comando del que se solicita información. Si no se proporciona se solicita información general	Información general de todos los comandos, o específica de uno. Ejemplo: /info add
Agregar artículo	/add <articulo>	artículo : (obligatorio) la liga del artículo web	Identificador del artículo en el sistema y los tópicos asignados a él
Eliminar artículo	/rm <id>	id : (obligatorio) el identificador del artículo en el sistema	Notificación de operación exitosa
Buscar artículos por título	/get <texto>	texto : (obligatorio) fragmento de título de los artículos que se buscan	Artículos relacionados cuyo título contenga el texto indicado
Buscar artículos por categorías	/getc <topico1>, <topico2>, . . . , <topicoN>	topicoN : (obligatorio al menos una categoría) categorías (separadas por coma) de los artículos que se buscan	Artículos relacionados a todas las categorías indicadas
Ver tópicos con mayor número de artículos	/top <n>	n: (opcional, valor predeterminado: 10) número entero mayor a cero que indica los primeros N tópicos a mostrar.	Lista de las N categorías con la mayor cantidad de artículos relacionados a cada uno, ordenados de forma descendente
Ver tópicos con menor número de artículos	/tail <n>	n: (opcional, valor predeterminado: 10) número entero mayor a cero que indica las primeras N categorías a mostrar	Lista de las N categorías con la menor cantidad de artículos relacionados a cada uno, ordenados de forma ascendente
Generar categorías	/build		Notificación de operación exitosa

Extracción de información

- Cada vez que se agrega un artículo a la biblioteca se extrae la información de la liga proporcionada.
- Este proceso requiere siempre conexión a internet. En caso de que falle la conexión el proceso se interrumpe y fallará.
- Se requiere utilizar una técnica de web scraping de la liga web proporcionada.

Algoritmo de clasificación

- Cada vez que se agrega un artículo a la biblioteca o se solicita la clasificación explícita se requiere el contenido del artículo. A dicha información se le aplica un modelo de aprendizaje automático que clasifique la información.
- El modelo de aprendizaje automático debe de ser de tipo Topic Modeling pues debe obtener los principales tópicos (o categorías) a partir de un texto recibido.
- El algoritmo ideal para ser implementado es LDA.

Modelo de datos

- Cada vez que se ejecuta un comando el sistema debe utilizar una base de datos para extraer e insertar información.
- El modelo de datos será lo más sencillo posible.
- Se elige un modelo no-relacional de tipo documental con formato JSON
- Se elige a MongoDB como motor de base de datos.
- La estructura de datos tiene el siguiente formato:

Documento	Atributo	Descripción	Tipo
Artículo			
	id	Identificador del artículo	texto
	título	Título del artículo	texto
	subtítulo	Subtítulo del artículo	texto
	lda	Modelo LDA que contiene los tópicos relacionados al artículo cada uno con un valor numérico	objeto
	url	Liga de artículo	texto
	contenido	Contenido del artículo	texto

Capítulo 4

Implementación

Para el desarrollo del sistema se configuró el ambiente de desarrollo con las siguientes características.

- Sistema operativo: Ubuntu 18.04
- Docker Engine - Community 20.10.14
- Python 3.8
- Ambiente virtual (virtualenv 15.1.0) con las bibliotecas:
  - pyTelegramBotAPI 4.5.0
  - pymongo[srv] 4.1.1
  - selenium 4.1.3
  - webdriver_manager 3.5.4
  - gensim 4.1.3

4.1. Arquitectura del sistema

Si observamos la figura 4.1, podemos ver que el sistema se divide en diferentes módulos, cada uno con una función en específico.

1. Core: se encarga de implementar la lógica del sistema y organizar la interacción de los otros módulos.
2. Chatbot: se encarga de la interacción con el usuario.
3. WebScraping: se encarga de la extracción de información del artículo.
4. Model: se encarga de la clasificación de la información del artículo.

5. DataBase: se encarga de guardar y acceder a la información del sistema.

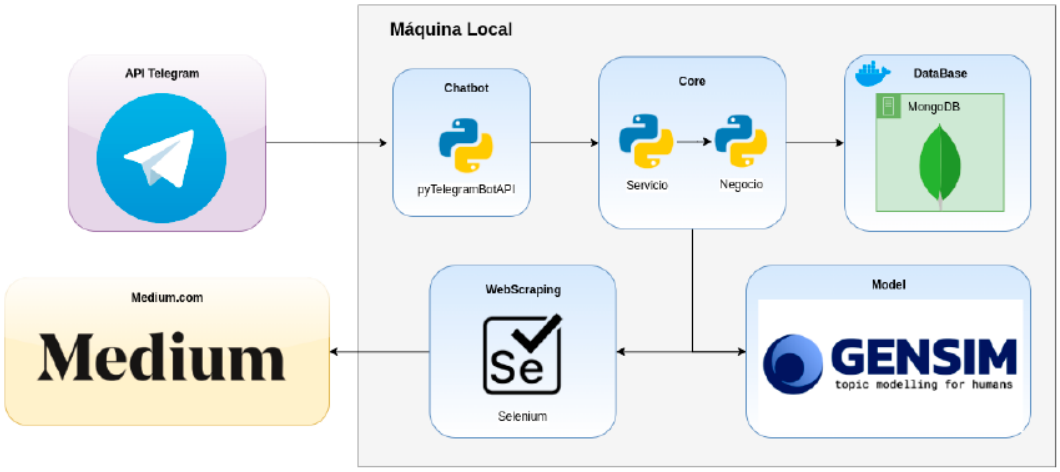


Figura 4.1: Módulos del sistema

En la figura 4.1 se puede observar la interacción general de cada módulo y el flujo de información. Todos los módulos consumen los recursos de una computadora personal y tienen interacción únicamente con recursos locales; a excepción de Chatbot y WebScraping, que interactúan con servicios en internet.

A continuación se describe la implementación de cada uno de los módulos del sistema.

4.2. Módulo ChatBot

Es el módulo que interactúa directamente con el usuario, por medio de la API de Telegram. El usuario escribe los comandos del sistema en forma de mensajes de textos que envía al bot personal que proporciona la propia plataforma de Telegram.

Tecnología: Telegram es una plataforma de mensajería instantánea líder en el mercado, con un servicio para desarrollo de chatbots llamado Telegram Bot

API, el cual es bastante intuitivo y fácil de implementar; además está disponible para cualquier usuario.

La biblioteca de python seleccionada para el uso de Telegram Bot API es pyTelegramBotAPI (`https://github.com/eternnoir/pyTelegramBotAPI`)

Diagrama de secuencia, representado en la figura 4.2: muestra el flujo de información que va en el siguiente orden.

1. El usuario hace una petición al API de Telegram.
2. La petición llega hasta el módulo Chatbot.
3. El módulo Chatbot consume a la capa de Servicio (módulo Core).
4. La información viaja de regreso.

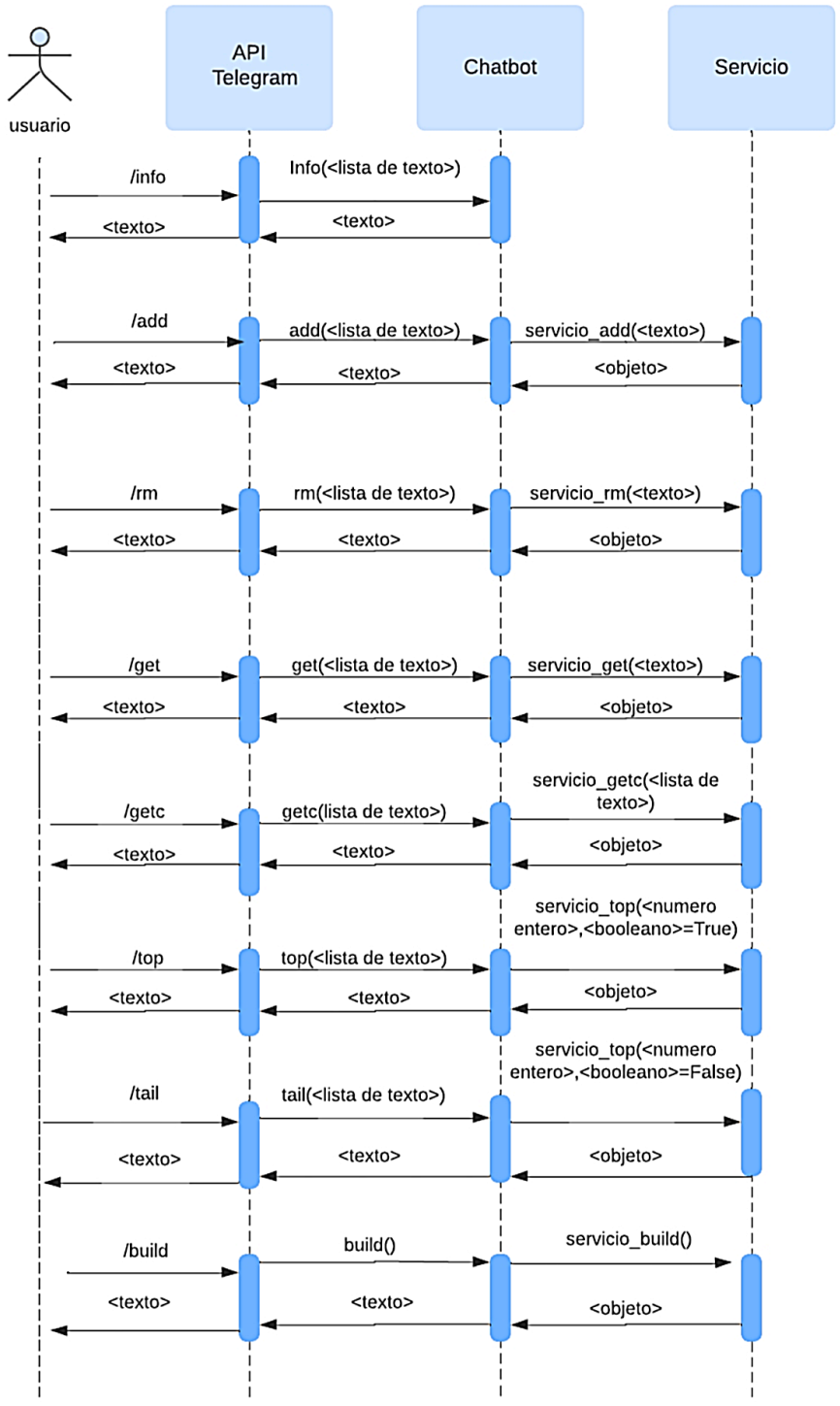


Figura 4.2: Diagrama de secuencia del módulo Chatbot

Descripción de los métodos.

- info
  - Procesa el comando */info*.
  - Recibe un parámetro opcional, el cual indica el comando de que se requiere información.
  - Retorna un mensaje que corresponde a la forma correcta de usar el comando indicado.
- add
  - Procesa el comando */add*.
  - Recibe un parámetro obligatorio, el cual debe de ser una url válida de un artículo web de la plataforma medium.com o springer.com.
  - Retorna un mensaje que indica la clasificación asignada al artículo y su identificador en base de datos.
  - Excepciones:
    - La liga del artículo indicado no es válida.
    - Excepciones anidadas del método *servicio_add* del módulo Core.
- rm
  - Procesa el comando */rm*.
  - Recibe un parámetro obligatorio, el cual debe de ser el identificador de un artículo existente en la biblioteca.
  - Retorna un mensaje que indica que se ha eliminado correctamente.
  - Excepciones:
    - Argumento requerido.
    - Excepciones anidadas del método *servicio_rm* del módulo Core.
- get

- Procesa el comando */get*.
- Recibe un parámetro obligatorio, el cual debe de ser texto que coincida con el fragmento del título de un artículo.
- Retorna una lista de artículos cuyo t ıtulo contenga el texto indicado.
- Excepciones:
  - Argumento requerido.
  - Excepciones anidadas del método *servicio_get* del módulo Core.

- getc
  - Procesa el comando */getc*.
  - Recibe un parámetro obligatorio, el cual debe de ser una o varias categorías existentes separadas por coma.
  - Retorna una lista de artículos que coinciden con las categorías indicadas.
  - Excepciones:
    - Argumento requerido.
    - Excepciones anidadas del método *servicio_getc* del módulo Core.

- top
  - Procesa el comando */top*.
  - Recibe un parámetro opcional, el cual debe de ser un número entero positivo. Si no se indica parámetro o se indica el número cero, se toma al número 10.
  - Retorna una lista de categorías de longitud no mayor al parámetro indicado.
  - Excepciones:
    - Argumento inválido.
    - Excepciones anidadas del método *servicio_top* del módulo Core.

- tail
    - Procesa el comando */tail*.
    - Recibe un parámetro opcional, el cual debe de ser un número entero positivo. Si no se indica parámetro o se indica el número cero, se toma al número 10.
    - Retorna una lista de categorías de longitud no mayor al parámetro indicado.
    - Excepciones:
        - Argumento inválido.
        - Excepciones anidadas del método *servicio_top* del módulo Core.
- build
    - Procesa el comando */build*.
    - No recibe parámetros de entrada y retorna un mensaje de ejecución exitosa.
    - Excepciones:
        - Argumento inválido.
        - Excepciones anidadas del método *servicio_build* del módulo Core.

4.3. Módulo Core

Es el módulo que se encarga de generar la lógica de cada uno de los casos de uso. Gestiona y consume todos los módulos del sistema, a excepción de Chatbot.

Tecnología: todo el desarrollo utiliza únicamente Python.
Está dividido en dos capas.

- Servicio: capa encargada de proporcionar al módulo Chatbot la lógica requerida para cada comando.

- Negocio: capa que contiene toda la lógica de cada comando y gestiona a los diferentes módulos del sistema.

Diagrama de secuencia de la capa Servicio, representado en la figura 4.3: muestra el flujo de información que va en el siguiente orden.

- El módulo Chatbot solicita la lógica de procesamiento de un comando.
- La capa de Servicio es consumida por el módulo Chatbot.
- La capa de Servicio consume la capa de Negocio.
- La información viaja de regreso.

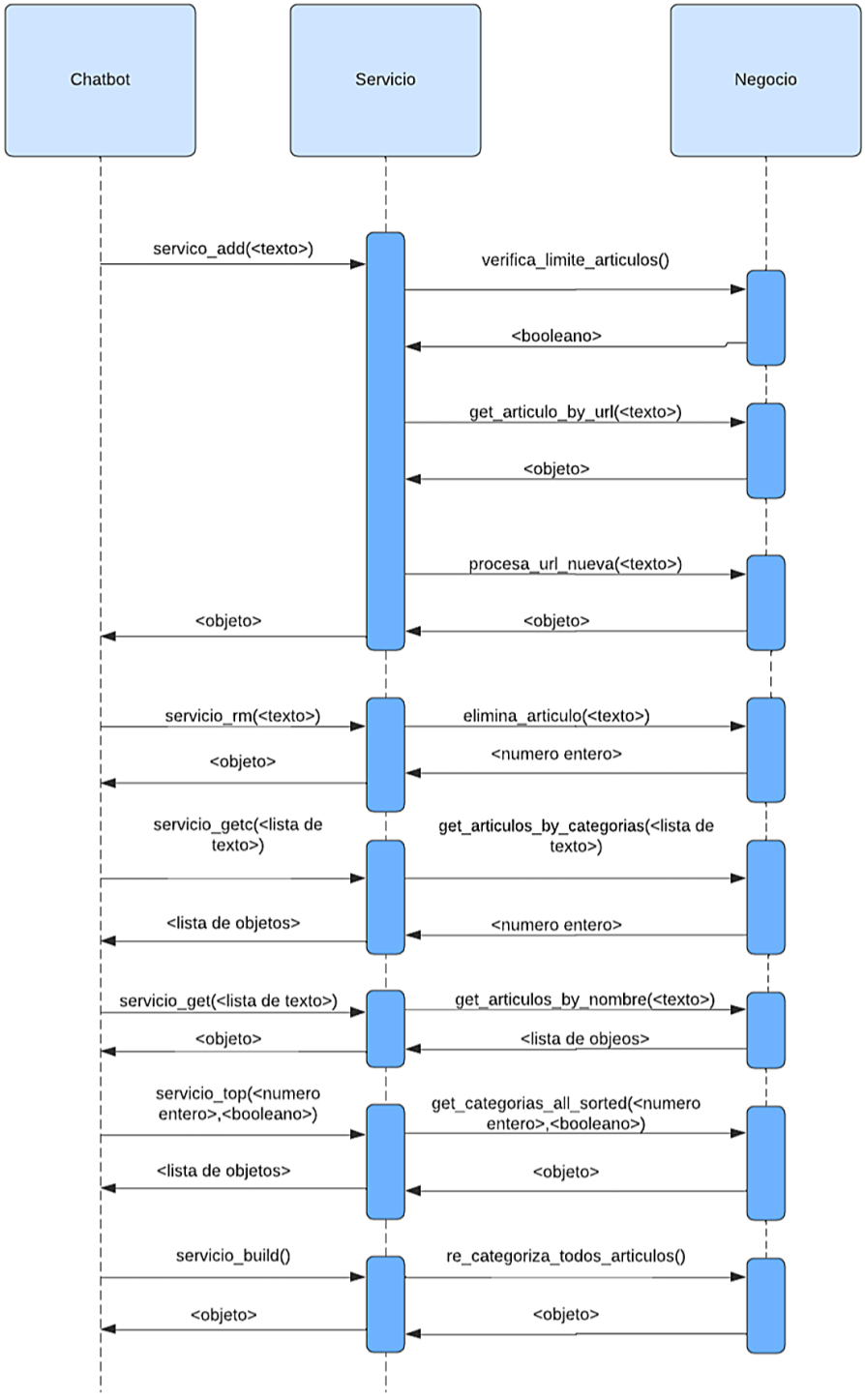


Figura 4.3: Diagrama de secuencia de la capa Servicio del módulo Core

Métodos de la capa Servicio.

- servicio_add
  - Método intermedio entre Chatbot y la capa Negocio que especifica la lógica del comando */add*.
  - Recibe un parámetro obligatorio, que es la url del artículo a agregar a la base de datos.
  - Retorna la clasificación asignada al artículo y su identificador en base de datos.
  - Excepciones:
    - ◇ No se pueden agregar más artículos porque se ha alcanzado el límite de categorías permitidas.
    - ◇ El artículo ya se ha procesado previamente.
    - ◇ Excepciones anidadas de los métodos de la capa Negocio.
- servicio_rm
  - Método intermedio entre Chatbot y la capa Negocio que especifica la lógica del comando */rm*.
  - Recibe un parámetro obligatorio, que es el identificador del artículo a eliminar de la base de datos.
  - Retorna un mensaje de eliminación exitosa.
  - Excepciones:
    - ◇ El identificador no es válido.
    - ◇ Excepciones anidadas de los métodos de la capa Negocio.
- servicio_getc
  - Método intermedio entre Chatbot .y la capa Negocio que especifica la lógica del comando */getc*
  - Recibe un parámetro obligatorio, que es la o las categorías para buscar los artículos.

    - Retorna una lista de artículos cuyo título contenga el texto indicado.
    - Excepciones:
        - Se requiere al menos una categoría.
        - Algunas categorías indicadas no existen.
        - Excepciones anidadas de los métodos de la capa Negocio.

- servicio_get
    - Método intermedio entre Chatbot y la capa Negocio que especifica la lógica del comando */get*.
    - Recibe un parámetro obligatorio, que es texto para buscar en los títulos de los artículos.
    - Retorna una lista de los artículos encontrados con las coincidencias de las categorías dadas.
    - Excepciones:
        - Excepciones anidadas de los métodos de la capa Negocio.

- servicio_top
    - Método intermedio entre Chatbot y la capa Negocio que especifica la lógica de los comandos */top* y */tail*.
    - Recibe dos parámetros obligatorios. El primero indicando la cantidad de categorías a mostrar y el segundo indica si el comando es */top* (predeterminado) o */tail*.
    - Retorna una lista de categorías con la longitud indicada, y si son las primeras o últimas (ordenadas por cantidad de artículos asociados) de acuerdo a si se indicó el comando */top* o */tail*
    - Excepciones: la anidadas de los métodos de la capa Negocio.

- servicio_build
    - Método intermedio entre Chatbot y la capa Negocio que especifica la lógica de los comandos */build*.

- No recibe parámetros y devuelve un mensaje de éxito.
- Excepciones: la anidadas de los métodos de la capa Negocio.

Diagrama de secuencia de la capa Negocio, mostrado en la figura 4.4: muestra el flujo de información que va en el siguiente orden.

1. La capa Servicio consume la capa Negocio.
2. La capa Negocio consume los módulos WebScraping, Model y DataBase, según se requiera.
3. La información viaja de regreso.

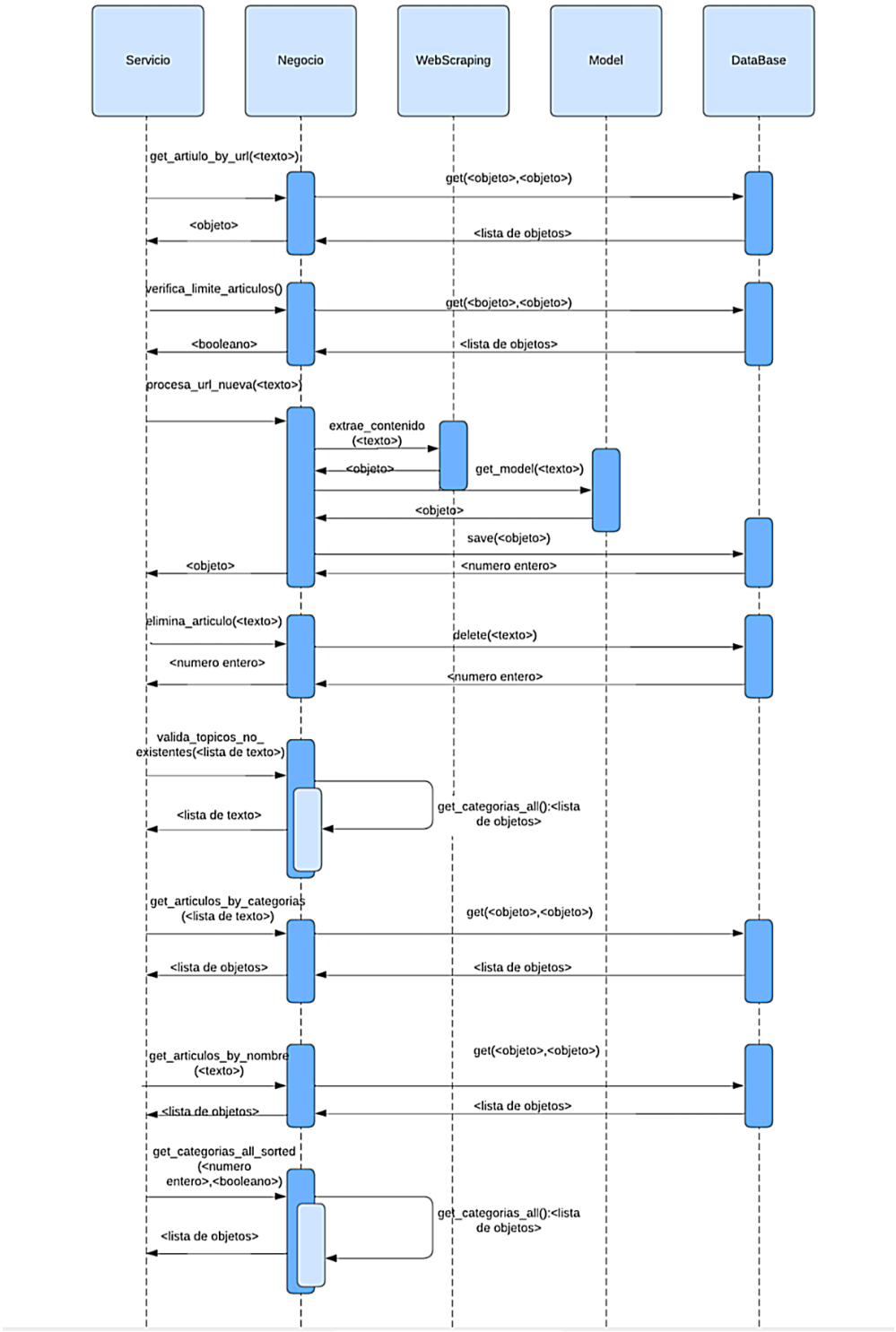


Figura 4.4: Diagrama de secuencia de la capa Negocio del módulo Core

Métodos de la capa Negocio.

- get_articulo_by_url
  - Método que obtiene un artículo de la base de datos a partir de su url.
  - Recibe un parámetro obligatorio, que es la url del artículo a obtener.
  - Retorna el artículo asociado a la url indicada.
  - Excepciones:
    - ◇ Aquellas anidadas de los métodos del módulo DataBase.
- verifica_limite_articulos
  - Método que verifica si ya se ha llegado al límite de artículos en la biblioteca.
  - No recibe parámetro y devuelve un dato de tipo booleano. *Verdadero* significa que aún no se llega al límite y se pueden agregar más artículos.
  - Retorna un mensaje de eliminación exitosa.
  - Excepciones:
    - ◇ Aquellas anidadas de los métodos del módulo DataBase.
- procesa_url_nueva
  - Método que se encarga de extraer el contenido de un artículo web, obtiene luego sus categorías y las guarda en la base de datos.
  - Recibe un parámetro, que es la url del artículo a procesar.
  - Retorna el identificador, el título y la categoría del artículo ya guardado en la base de datos.
  - Excepciones:
    - ◇ Aquellas anidadas de los métodos de los módulos WebScraping, Model y DataBase.

- elimina_articulo
    - Método que se encarga de eliminar de la base de datos un artículo específico.
    - Recibe un parámetro obligatorio, que es el identificador del artículo en la base de datos.
    - Retorna 0 o 1, donde 0 significa que el identificador es inválido, y 1 significa que se eliminó correctamente.
    - Excepciones:
        - ◇ Aquellas anidadas de los métodos del módulo DataBase.
- valida_topicos_no_existentes
    - Método que se encarga de verificar una lista de categorías para identificar cuáles de ellas no existen en la base de datos.
    - Recibe un parámetro obligatorio, que es una lista de las categorías a validar.
    - Retorna una lista que contiene aquellas categorías (recibidas como parámetro) que no existen en la base de datos; si todas existen entonces la lista se devuelve vacía.
    - Excepciones:
        - ◇ Aquellas anidadas del método *get_categorias_all*.
- get_urls_by_categorias
    - Método que se encarga de obtener de la base de datos todos los artículos que coinciden con una lista de categorías.
    - Recibe un parámetro obligatorio, que es una lista de categorías.
    - Retorna una lista de los artículos que son parte de todas las categorías a la vez.
    - Excepciones:

        - ◇ Aquellas anidadas de los métodos del módulo DataBase.

- get_urls_by_nombre
    - Método que se encarga de obtener de la base de datos todos los artículos cuyo título coincide parcialmente con cierto texto.
    - Recibe un parámetro obligatorio que es un texto para buscar en los títulos de todos los artículos.
    - Retorna una lista de los artículos coincidentes por título.
    - Excepciones:
        - ◇ Aquellas anidadas de los métodos del módulo DataBase.

- get_categorias_all_sorted
    - Método que se encarga de obtener todas las categorías de la base de datos, de forma ordenada.
    - Recibe dos parámetros obligatorios. El primero indicando la cantidad de categorías a obtener (de manera predeterminada indica que todas) y el segundo indica el orden (de manera predeterminada descendente).
    - Retorna una lista de categorías, con la longitud y orden indicados.
    - Excepciones:
        - ◇ Aquellas anidadas del método *get_categorias_all*.

- get_categorias_all
    - Método que se encarga de obtener todas las categorías de la base de datos.
    - Retorna una lista desordenada de todas las categorías.
    - Excepciones:
        - ◇ Aquellas anidadas de los métodos del módulo DataBase.

- re_categoriza_todos_articulos

- Método que se encarga de volver a calcular el modelo LDA de todos los artículos de la biblioteca.
- No recibe ningún parámetro ni devuelve un valor.
- Excepciones:
  - Aquellas anidadas de los métodos del módulo Model.

4.4. Módulo WebScraping

Es el encargado de extraer la información de un artículo web de la plataforma `medium.com` o `springer.com`

Tecnología: Selenium es un ambiente de diferentes tecnologías que se enfoca en la automatización de pruebas de aplicaciones web, por lo que incluye su propia API para diferentes lenguajes de programación, controlador web, entre otros.

Selenium WebDriver (`https://www.selenium.dev/documentation/webdriver/`) es una API de selenium que sirve para automatizar tareas específicas de un navegador. Es ampliamente aceptado en la comunidad de desarrollo de software e incluso está en proceso de convertirse en el estándar de su tipo, pues es parte de las recomendaciones de W3C (`https://www.w3.org/TR/webdriver1/`).

Selenium WebDriver se usa comúnmente para realizar web scraping ya que tiene formas avanzadas de interactuar con páginas web, simulando ser un humano.

El proceso de web scraping del sistema se implementó utilizando la biblioteca de python *selenium* (`https://selenium-python.readthedocs.io/`) para web scraping y la bilbioteca *webdriver-manager* (`https://pypi.org/project/webdriver-manager/`) para controlar el navegador Chrome.

Diagrama de secuencia del módulo WebScraping, como se muestra en la figura 4.5

Muestra el flujo de información que va en el siguiente orden:

1. La capa Negocio del módulo Core consume el módulo WebScraping.
2. La información viaja de regreso.

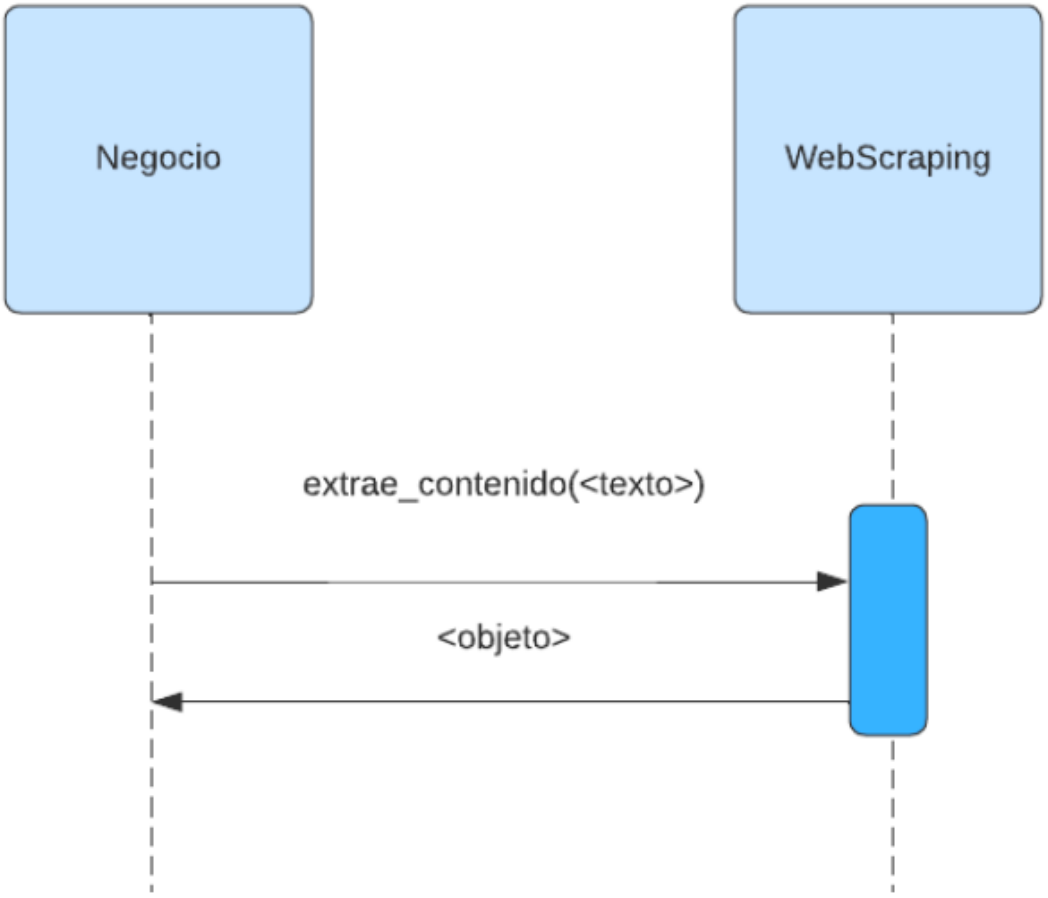


Figura 4.5: Diagrama de secuencia del módulo WebScraping

Métodos del módulo WebScraping.

- extrae_contenido
  - Método que se encarga de extraer el texto (identificando un título, subtítulo y contenido) de un artículo de la plataforma `medium.com` o `springer.com`
  - Recibe un parámetro obligatorio, que es la url del artículo a procesar.
  - Retorna el título, subtítulo y contenido del artículo.
  - Excepciones:
    - La url no es válida.
    - Aquellas no controladas de Selenium.

4.5. Módulo Model

Es el encargado de aplicar un modelo de aprendizaje automático de tipo Topic Modeling al texto del artículo. El método elegido es LDA (Latent Dirichlet Allocation) el cual genera diez categorías distintas a partir de un texto dado. El texto es el contenido del artículo, y las categorías resultantes son las representación semántica del artículo.

Tecnología: Gensim (`https://radimrehurek.com/gensim/`) es la biblioteca de python de código libre (tipo de licencia comunitaria) más popular enfocada a PLN que implementa varios algoritmos de Topic Modeling, incluyendo LDA.

Diagrama de secuencia del módulo Model, representado en la figura 4.6: muestra el flujo de información que va en el siguiente orden.

1. La capa Negocio del módulo Core consume el módulo Model.
2. La información viaja de regreso.

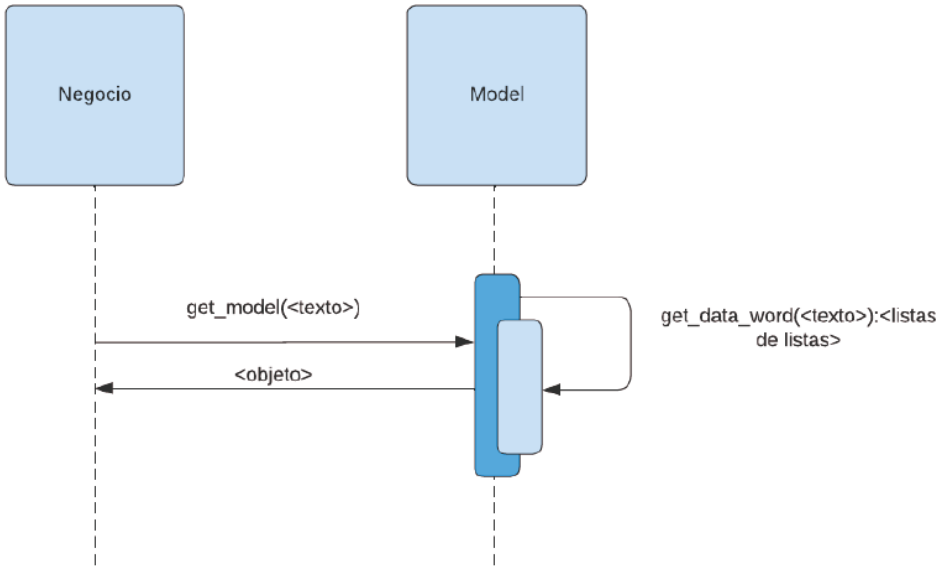


Figura 4.6: Diagrama de secuencia del módulo Model

Métodos del módulo Model.

- get_model
  - Método que se encarga de generar el modelo de tipo Topic Modeling implementado por el algoritmo LDA. Usa la biblioteca Gensin que contiene algoritmos con configuraciones predeterminadas.
  - Recibe un parámetro obligatorio, que es una sola cadena de texto.
  - Retorna 10 categorías, cada una con un valor específico
  - Excepciones:
    - Aquellas anidadas del método *get_data_words*.
    - Aquellas no controladas de Gensin.
- get_data_words
  - Método que se encarga de limpiar el texto a tratar (de signos de puntuación y *stop words*; las *stop words* son palabras que no aportan información a la semántica de una oración para los algoritmos de PLN, por ejemplo: *el, la, los, del, es*) y genera la estructura que requiere la biblioteca Gensin.

- Recibe un parámetro obligatorio, que es una sola cadena de texto.
- Retorna la estructura que requiere el algoritmo LDA.
- Excepciones:
  - Aquellas no controladas de Gensin.

4.6. Módulo DataBase

Es el encargado de comunicarse con la base de datos, para guardar y consumir la información del sistema.

Tecnología: MongoDB es un motor de base de datos de tipo documental, el cual tiene como principales características la flexibilidad de implementación y velocidad tanto de escritura como lectura.

Diagrama de secuencia del módulo DataBase, representado en la figura 4.7: muestra el flujo de información que va en el siguiente orden.

1. La capa Negocio del módulo Core consume el módulo DataBase.
2. La información viaja de regreso.

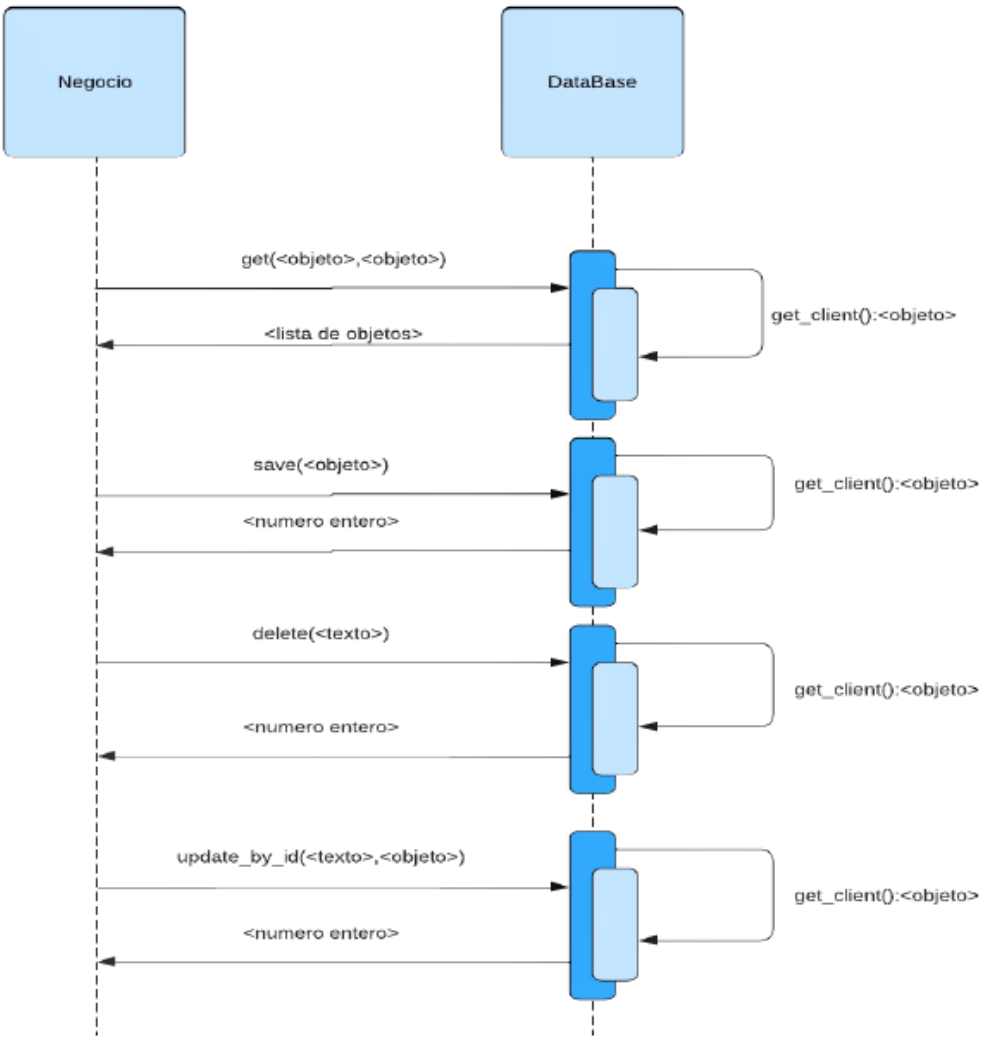


Figura 4.7: Diagrama de secuencia del módulo DataBase

- get_client
  - Método que se encarga de generar una conexión a la base de datos. Utiliza la biblioteca PyMongo.
  - Retorna una conexión nueva a la base de datos.
  - Excepciones: aquellas no controladas de PyMongo.
- get

- Método que se encarga de realizar una búsqueda en la base de datos y obtener los artículos que coincidan con los parámetros indicados.
- Recibe dos parámetros obligatorios. El primero es una estructura de filtros que utiliza MongoDB para realizar la búsqueda. El segundo es una lista de los atributos del modelo que interesan, de manera predeterminada trae todos los atributos.
- Retorna una lista de artículos que coinciden con los filtros proporcionados y cada uno con los atributos indicados.
- Excepciones:
  - Aquellas anidadas del método *get_client*.
  - Aquellas no controladas de PyMongo.

- save
  - Método que se encarga de guardar un nuevo artículo en la base de datos.
  - Recibe un parámetros obligatorio, que es el artículo a guardar.
  - Retorna el identificador asignado al artículo guardado.
  - Excepciones:
    - Aquellas anidadas del método *get_client*.
    - Aquellas no controladas de PyMongo.

- delete
  - Método que se encarga de eliminar un artículo de la base de datos.
  - Recibe un parámetros obligatorio, que es el identificador del artículo a eliminar.
  - Retorna el artículo que se ha eliminado.
  - Excepciones:
    - Aquellas anidadas del método *get_client*.
    - Aquellas no controladas de PyMongo.

- update_by_id
    - Método que se encarga de actualizar un artículo de la base de datos.
    - Recibe dos parámetros obligatorios, que son el identificador del artículo a actualizar y los nuevos datos a insertar.
    - Retorna el número de artículos actualizados.
    - Excepciones:
        - Aquellas anidadas del método *get_client*.
        - Aquellas no controladas de PyMongo.

Conclusiones

El objetivo de la presente tesis fue cumplido a lo largo del desarrollo, pues se mostró claramente el proceso de diseño e implentación de un sistema clasificador automático de artículos web. Además se desarrollaron puntualmente los conceptos teóricos que soportan dicha implementación tecnológica.

El presente sistema está abierto a extensión, es decir que está listo para agregar una mayor cantidad de funcionalidades, así como agregar mayor complejidad a las funcionalidades ya existentes.

Un siguiente paso a seguir en el proceso de desarrollo de este sistema es subir el código a un servidor web; sin embargo esto involucra costos.
Otro paso a seguir es la ampliación de bases de datos editoriales que acepta el sistema. Una opción podría ser ScienceDirect, por ejemplo, la cual es una base de datos de artículos científicos y médicos.

Bibliografía

[1] Barahona, Abel. *Metodologia de trabajos cientificos*, 2nd ed. Bogota: Ipler, 1979

[3] A. Arias, ed. *Ingeniería y Arquitectura del Software*, 2016

[6] T. Scheepers. *Virtualization and Containerization of Application Infrastructure: A Comparison*. University of Twente Enschede, 2014

[7] D. Kusnetzky. Virtualization A Managers' Guide, O'Reily, 2011

[9] Gunther Reinhart The International Academy for Production Engineering Luc Laperriére. *CIRP Encyclopedia of Production Engineering*, SpringerReference, 2014

[10] E. Alpaydin. *Introduction to Machine Learning*, Massachusetts Institute of Technology, 2014

[11] A. Farzirdar. *Natural Language Processing for Social Media*, Morgan Claypool, 2016

[12] P. Vanichvasin. "Chatbot Development as a Digital Learning Tool to Increase Students' Research Knowledge". En: *Canadian Center of Science and Education*, 2021, pp. 44-53

[13] A. Dieng y et. al. "Topic Modeling Embedding Spaces". En: *Transactions of the Association for Computational Linguistics*, 2020, pp. 439-453

[14] D. Blei y et. al. "Latent Dirichlet Allocation". En: *Journal of Machine Learning Research*, 2003, pp. 993-1022

[15] A. Castro y et. al. "Utilidad y funcionamiento de las bases de datos NoSQ". En: *Revista de la Facultad de Ingeniería de la Universidad Pedagógica y Tecnológica de Colombia*, 2012, pp. 21-32

[17] P. Sadalage. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*, Pearson Education, Inc., 2013

Mesografía

[2] J. Llamas, ed. (2021) *Sistema informático*. [Online] Disponible: https://economipedia.com/definiciones/sistema-informatico.html

[4] Red Hat, ed. (2019) *¿Qué es la infraestructura de TI?*. [Online] Disponible: https://www.redhat.com/es/topics/cloud-computing/what-is-it-infrastructure

[5] I. Flores, ed. (2020) *HTTP*. [Online] Disponible: https://developer.mozilla.org/es/docs/Web/HTTP

[8] M. Martí, ed. (2016) *¿Qué es el Web scraping? Introducción y herramientas*. [Online] Disponible: https://web.archive.org/web/20170729001446/https://sitelabs.es/web-scraping-introduccion-y-herramientas/

[16] Whittake, ed. (2022) *An Illustrated Proof of the CAP Theorem. Retrieved on May 5th*. [Online] Disponible: https://mwhittaker.github.io/blog/an_illustrated_proof_of_the_cap_theorem/

[17] O.S. Group. (2022) *Virtualización de servidores*. [Online] Disponible: https://www.osgroup.co/virtualizacion-de-servidores/

[18] Desde Linux. (2022) *Docker vs Kubernetes: ventajas y desventajas*. [Online] Disponible: https://blog.desdelinux.net/docker-vs-kubernetes/

[19] Dave Costenaro. (2018) *Preparing for Artificial Intelligence*. [Online] Disponible: https://becominghuman.ai/preparing-for-artificial-intelligence-d0b644087537

[20] Juan Ignacio Barrios Arce. (2020) *Inteligencia Artificial y Machine Learning*

para todos. [Online] Disponible: https://www.juanbarrios.com/inteligencia-artificial-y-machine-learning-para-todos/

[21] Victor J. Vallejo. (2022) *Procesamiento del lenguaje natural y Comprensión del lenguaje natural*. [Online] Disponible: https://es.wikipedia.org/wiki/Archivo:Procesamiento_del_lenguaje_natural_y_Comprensi%C3%B3n_del_lenguaje_natural.png

[22] Vallejo, V. (2021) *Topic Modelling using LDA*. [Online] Disponible: https://medium.com/analytics-vidhya/topic-modelling-using-lda-aa11ec9bec13

[23] Buenaño-Fernández, D. (2020) *Text Mining of Open-Ended Questions in Self-Assessment of University Teachers: An LDA Topic Modeling Approach*. [Online] Disponible: https://www.researchgate.net/publication/339368709_Text_Mining_of_Open-Ended_Questions_in_Self-Assessment_of_University_Teachers_An_LDA_Topic_Modeling_Approach

[24] Juan Rivillas. (2020) *O Teorema CAP!*. [Online] Disponible: https://www.juanrivillas.com/blog/cap_theorem/

[25] Radim Řehůřek. (2022) *GENSIM topic modelling for humans*. [Online] Disponible: https://radimrehurek.com/gensim/index.html