REUE | Artículo especial

El Procesamiento de Lenguaje Natural en la revisión de literatura científica

Natural language processing for reviewing search results from PubMed

Melanie Calero Sánchez[1], José Carlos González González[2], Isabel Sánchez Berriel[1], Guillermo Burillo-Putze[3], José Luis Roda García[1]

Filiación de los autores:
[1]Departartamento de Ingeniería Informática y de Sistemas, Universidad de La Laguna, Tenerife, España.
[2]Servicio de Informática, Universidad de La Laguna, Tenerife, España.
[3]Servicio de Urgencias. Hospital Universitario de Canarias, Universidad de La Laguna,Tenerife, España.

Correspondencia:
Melanie Calero Sánchez.
Dpto de Ingeniería Informática y de Sistemas, Universidad de La Laguna.
Camino San Francisco de Paula, 19.
38200, La Laguna, Tenerife, España.

E-mail:
melaniecalero.5@gmail.com

Información del artículo:
Recibido: 19-6-2024.
Aceptado: 28-6-2024.
Online: 2-7-2024.

Editor responsable:
Manuel Pardo Ríos.

DOI:
10.55633/s3me/REUE030.2024

Introducción

El Procesamiento del Lenguaje Natural (PLN) es una disciplina con una larga historia. Nació en los años 50 como una subárea de la Inteligencia Artificial (IA) y la Lingüística, con el objetivo de estudiar los problemas derivados de la generación y comprensión automática del lenguaje natural. Aunque se pueden encontrar trabajos de épocas anteriores, fue en 1950 cuando Alan Turing publicó un artículo titulado "Intelligence" en el que proponía lo que hoy se denomina el test de Turing como criterio de inteligencia[1].

En la actualidad existe una gran cantidad de información de calidad disponible en repositorios especializados para diferentes campos de conocimiento. Sin embargo, cuando un especialista se enfrenta a la búsqueda de información, la relevancia de las respuestas obtenidas debe determinarla mediante la lectura y análisis del contenido de los documentos. Para simplificar y hacer más eficiente este proceso, podemos encontrar numerosos trabajos sobre recuperación de información en textos científicos aplicando técnicas de PLN.

En el trabajo de Tommasel *et al.*[2], los autores presentan una aplicación web para la recuperación de información enriquecida con un recomendador de documentos médicos en danés. Los documentos que proporciona a los usuarios se determinan aplicando técnicas de PLN mediante búsquedas léxicas y semánticas. También guía al usuario en su búsqueda, recomendando nuevas consultas relacionadas, así como proponiendo conceptos médicos que dirigen la búsqueda.

Podemos encontrar una comparación entre diferentes técnicas de representación de documentos aplicado al filtrado y etiquetado de documentos para la búsqueda de evidencias en medicina, encontrando resultados prometedores cuando se utilizan algoritmos basados en modelos del lenguaje[3]. La representación mediante técnicas de PLN de las búsquedas en sistemas de recuperación de información del paciente, pero también de literatura médica se analiza en el trabajo de Peikos *et al.*[4]. Estos autores llegan a la conclusión que la expansión de las consultas con entidades médicas y otros datos del paciente mejoran los resultados respecto a las búsquedas planteadas por asesores médicos cuando se realizan sobre las historias clínicas, pero no en la literatura científica.

Otros ámbitos de conocimiento en los que se han aplicado técnicas avanzadas de PLN para la recuperación de información son el ámbito legal[5] o incluso el propio PLN, como presentan los autores de la herramienta NLPEXPLORER[6], un portal no sólo para la búsqueda de documentos, sino también para la indexación y visualización de los resultados de las consultas. Una idea del interés de la investigación de estas técnicas en el campo de la medicina es la inclusión en la CLEF *eHealth Evaluation Lab,* de una sección temática en recuperación de información en la literatura científica médica[7]. Gracias a la precisión de la información que se obtiene con estas técnicas se pueden construir soluciones confiables que funcionan como filtros altamente especializados y así reducir notablemente la cantidad de información que al investigador se le presenta. Además, basados en ellos, se pueden construir puntos de consulta para usuarios con conocimientos básicos en Informática, poniendo a su disposición la tecnología de última generación en recuperación de información.

Extraer conocimiento de un *corpus* de documentos de forma automática o semiautomática empieza a ser posible. Cada vez más, investigadores de diferentes ámbitos adquieren habilidades relacionadas con la programación en determinados lenguajes informáticos. Con un conjunto simple de instrucciones y con el apoyo de librerías especializadas, los investigadores pueden obtener información de documentos de manera práctica y sencilla. Así, el uso de herramientas como *Jupyter Notebooks o Colab* ha facilitado el desarrollo de la escritura de código en diferentes lenguajes para poder realizar diversas tareas, entre ellas las de recuperación y análisis de la bibliografía.

© SEMES

En noviembre de 2022 aparece *ChatGPT,* un sistema conversacional basado en la IA generativa. En este caso, el tratamiento de documentos se puede realizar sin necesidad de escribir una sola línea de código sino a través de preguntas que realiza el usuario en su propio lenguaje al sistema y este, una vez realizadas las búsquedas oportunas (recuperación de información), devuelve una respuesta elaborada en lenguaje natural (generación de respuesta).

PubMed[8] es una base de datos gratuita que incluye principalmente la base de datos MEDLINE de referencias y resúmenes sobre ciencias de la vida y temas biomédicos. La Biblioteca Nacional de Medicina (NLM) de los Institutos Nacionales de Salud de Estados Unidos mantiene la base de datos como parte del sistema *Entrez* de recuperación de información.

Desde 1971 hasta 1997, el acceso en línea a la base de datos MEDLINE se realizaba principalmente a través de instalaciones institucionales, como las bibliotecas universitarias. PubMed, publicado por primera vez en enero de 1996, marcó el comienzo de la era de la búsqueda privada, gratuita en MEDLINE. El sistema PubMed se ofreció gratuitamente al público a partir de junio de 1997.

A 23 de mayo de 2023, PubMed cuenta con más de 35 millones de citas y resúmenes que se remontan a 1966, selectivamente al año 1865, y muy selectivamente a 1809. En la misma fecha, 24,6 millones de registros de PubMed aparecen con sus resúmenes, y 26,8 millones de registros tienen enlaces a versiones de texto completo (de los cuales 10,9 millones de artículos están disponibles, a texto completo y de forma gratuita). En los últimos 10 años, se han añadido una media de casi un millón de registros nuevos cada año.

En este trabajo, aplicaremos diversas técnicas de procesamiento del lenguaje natural para procesar las publicaciones de PubMed y analizar en qué medida están vinculadas con nuestro tema de investigación: el suicidio en jóvenes y los algoritmos de *Machine Learning*. Nuestro planteamiento es ofrecer un conjunto de estrategias, técnicas y herramientas de PLN que faciliten al investigador la fase inicial de revisión de documentos de una temática específica.

Para ello en primer lugar se presentan las principales estrategias y técnicas de PLN, mostrando en el cuarto apartado un ejemplo de uso de algunas de estas técnicas para nuestro caso de revisión. A continuación se muestran nuevas estrategias y técnicas basadas en la IA Generativa para realizar el mismo proceso de revisión anterior, mostrando en el apartado siguiente, un ejemplo para su comprensión. En el apartado siguiente se ofrece la descripción e implementación de los sistemas actuales de *chatbots* que facilitan la interacción entre el investigador y el *corpus* de artículos encontrados a través de sistemas preguntas-respuestas.

Objetivos

El objetivo principal de este trabajo es demostrar cómo el uso del procesamiento de lenguaje natural facilita y mejora la recopilación de bibliografía científica. Específicamente, se busca:

a) Analizar cómo las técnicas de PLN nos ayudan en la recuperación de bibliografía científica.

b) Describir los principales métodos y estrategias de PLN aplicables a la búsqueda y selección de bibliografía.

c) Proveer ejemplos prácticos de los pasos a seguir en las revisiones bibliográficas haciendo uso de PLN.

Técnicas Clásicas de Procesamiento de Lenguaje Natural

El PLN es el campo de conocimiento de la IA que se ocupa de investigar la manera de comunicar las máquinas con las personas mediante el uso de lenguas naturales, como el español, el inglés o el chino. Cualquier lengua humana puede ser tratada por los ordenadores. Lógicamente, limitaciones de interés económico o práctico hacen que solo las lenguas más habladas o utilizadas en el mundo digital tengan aplicaciones informáticas en uso.

El PLN abarca algoritmos tanto para la comprensión como para la generación del lenguaje. Estos algoritmos se utilizan en aplicaciones informáticas con un objetivo específico. Algunas de ellas requieren exclusivamente la comprensión del lenguaje, otras son tareas generativas y otras tienen componentes de ambas[9]. Algunos ejemplos de aplicaciones del PLN son:

Categorización de textos

La clasificación de textos generalmente asigna una etiqueta o categoría a un documento o fragmento de texto. Según el fin de la clasificación nos estaremos refiriendo a un proceso de análisis de sentimiento, detección de spam, identificación del tópico o tema del que versa un texto, o simplemente clasificarlo según otro tipo de categorías libres[10].

El análisis de sentimiento determina el sentimiento (positivo, negativo, neutral) expresado en un texto. Por ejemplo, "¡Me encantó la película!" debe asignarse sentimiento positivo. Algoritmos más complejos evalúan el sentimiento que expresa un texto respecto a aspectos concretos[11]. En este caso, si los aspectos a valorar son "restaurante" y "menú", en la expresión: "El restaurante era caro, pero el menú barato" deben recibir la etiqueta negativa y positiva respectivamente.

En cuanto a la extracción de tópicos, podremos establecer de forma automática si un determinado artículo versa sobre epidemiología, pediatría o algún otro tema. Estos algoritmos son de especial importancia para la recuperación de información.

© SEMES

Generación automática de resúmenes

Las técnicas englobadas en este grupo tienen como objetivo resumir un documento de texto o conjuntos de ellos, presentando al usuario la información principal de forma precisa y completa. El objetivo principal es, en un documento concreto, generar un texto con las ideas principales y con la menor repetición posible[12].

Extracción de información

Se conoce como extracción de información al conjunto de técnicas que permiten identificar menciones en el texto a conceptos semánticos o entidades y también relaciones entre ellos[13]. Las entidades y relaciones a extraer dependerán del dominio de los textos. Por ejemplo, en un dominio general podemos hablar de personas, organizaciones y lugares en el rol de entidades y tipos de relaciones entre ellas: estar casados, empleados por, o vivir en. Si hablamos de un dominio biomédico, podríamos referirnos a entidades: "proteína", "enfermedad", "tratamiento" y relaciones "proteína-proteína", "enfermedad-tratamiento", etc. En un nivel más complejo, el tratamiento en un modelo de grafos de esta información permite la generación de grafos de conocimiento que constituyen mapas conceptuales de la información implícita en colecciones de documentos.

Respuestas a preguntas

El objetivo de los sistemas pregunta-respuesta es proporcionar respuestas pertinentes a preguntas que realiza el usuario en lenguaje natural. Las respuestas se proporcionan a partir de la información contenida en la base de conocimiento del sistema, generalmente creada a partir de documentos de relevancia en el dominio de la aplicación. La arquitectura de estos sistemas incluye 3 módulos: análisis de la pregunta, recuperación de los fragmentos que determinan la respuesta y construcción de la respuesta[14].

Traducción automática

El PLN potencia los sistemas de traducción automática como Google Translate. Estos sistemas alinean palabras en todos los idiomas, considerando el contexto y las expresiones idiomáticas.

Agentes conversacionales

Un agente conversacional es una aplicación que produce conversaciones humanas[15]. Su principal tarea es generar respuestas en lenguaje natural a preguntas realizadas por un humano también en lenguaje natural. Otros nombres que reciben estos sistemas son: *chatbots,* asistentes virtuales, agentes interactivos, asistentes digitales, etc. Podemos encontrar sistemas basados en reglas que analizan el mensaje introducido por el usuario y construyen la respuesta mediante plantillas preestablecidas según el resultado que obtienen al procesar dicho mensaje. En otra línea están los sistemas modernos basados en redes neuronales de aprendizaje profundo. En este caso encontramos los que se basan en recuperación, la respuesta del agente se produce computando la respuesta más relevante. Otro tipo son los agentes conversacionales generativos, que construye la repuesta proporcionando en cada instante la palabra del vocabulario que mejor se ajustaría hasta tener la construcción completa. También existen sistemas híbridos que combinan tanto la recuperación como la generación[16]. Un ejemplo de agente conversacional generativo es *ChatGPT.*

A continuación, vemos algunos de los niveles de análisis lingüísticos que son aplicados en el procesamiento del lenguaje natural. No todos los análisis que se describen se aplican en cualquier tarea de PLN, sino que depende del objetivo de la aplicación.

1. Análisis morfológico o léxico. Consiste en el análisis interno de las palabras que forman oraciones para extraer lemas, rasgos flexivos, unidades léxica compuestas. Es esencial para la información básica: categoría sintáctica y significado léxico.

2. Análisis sintáctico. Consiste en el análisis de la estructura de las oraciones de acuerdo con el modelo gramatical empleado (lógico o estadístico).

3. Análisis semántico. Proporciona la interpretación de las oraciones, una vez eliminadas las ambigüedades morfosintácticas.

4. Análisis pragmático. Incorpora el análisis del contexto de uso a la interpretación final. Aquí se incluye tanto el tratamiento del lenguaje figurado (metáfora e ironía) como el conocimiento del mundo específico necesario para entender un texto especializado.

Un análisis morfológico, sintáctico, semántico o pragmático se aplicará dependiendo del objetivo de la aplicación. Por ejemplo, un conversor de texto a voz no necesita el análisis semántico o pragmático. Pero un sistema conversacional requiere información muy detallada del contexto y del dominio temático.

Por otra parte, alguno de estos niveles de análisis, se subdividen en tareas, algunas de ellas requeridas por otras. El flujo de tareas que se necesitan en cada aplicación se denomina *pipeline*. Entre las tareas más utilizadas destacan[17]:

1. Preprocesamiento: Constituye el paso previo a cualquier procesamiento que se haga al texto. Abarca tareas de limpieza como eliminación de *urls, hashtags,* conversión a minúsculas, etc. Se trata de eliminar fragmentos que generan ruido en el texto. En ocasiones también se hace referencia a determinadas tareas orientadas a la estandarización del texto como la transformación a minúsculas, eliminación de puntuación y derivación. Por ejemplo, la derivación reduce las palabras a su forma raíz (por ejemplo, "corre" se convierte en "correr"). Estos pasos garantizan una entrada más limpia para tareas posteriores de PLN.

2. Identificación del idioma: Cada idioma contiene características propias que en determinados análisis requerirán un tratamiento específico. Es el primer análisis que se debe llevar a cabo, nos servirá por ejemplo para elegir qué ficheros de datos como diccionarios se usarán en el procesamiento.

3. Tokenización: La tokenización es una tarea clave en el PLN. Implica dividir un texto en unidades más pequeñas

© SEMES

Yo	**me**	**siento**	**herida**	**,**	**mucho**	**dolor**	**en**	**mi**	**corazón**
yo	me	sentar	herida	,	mucho	dolor	en	mi	corazón
PP1C	*PP1C*	*VMIP1S*	*NCFS00*	*Fc*	*DI0MS0*	*NCMS*	*SP*	*DP1CS*	*VMIS3P0*
SN0	*S00*	*0*	*0*			*000*		*S*	

Figura 1. Análisis del discurso sobre el dataset[20].

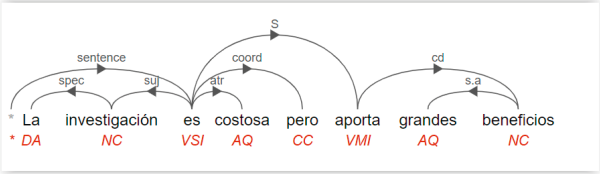


Figura 2. Análisis de dependencia obtenido con la demo online Freenling[19].

que pueden ser palabras, signos de puntuación o subpalabras. Según la aplicación que se vaya a dar al tokenizador será importante incluir los signos de puntuación, por ejemplo, en análisis de sentimientos serán importantes los signos de admiración "¡!". La inclusión de subpalabras resuelve problemas de falta de casos de palabras flexionadas en los conjuntos test[18]. Un tokenizador al nivel dada la frase: "Fui diagnosticado con deficiencia de vitamina B12.". Tokenizarlo produce: ["Fui", "diagnosticado", "con", "deficiencia", "de", "vitamina", "B12", "."].

4. Segmentación de oraciones: Del mismo modo que es necesario conocer la unidad mínima que tratarán los algoritmos mediante tokenizadores, para muchos problemas en PLN se necesita identificar las frases en el texto. Esta tarea se realiza con los algoritmos de segmentación de oraciones.

5. Lematización: Es el proceso que convierte las palabras al lema o forma canónica que las representa. Por ejemplo, "investigado"," investiga", "investigue" son flexiones del lema "investigar". La lematización se puede resolver mediante un análisis morfológico, o con algoritmos de derivación más simples que sólo extraen raíz del resto de la palabra.

6. Reconocimiento de entidades (NER): identifica entidades, típicamente, nombres de personas, organizaciones o ubicaciones en el texto. Por ejemplo:

Entrada: "Apple Inc. Tiene su sede en Cupertino".

Salida: "Apple Inc. (ORG) tiene su sede en Cupertino (LOC)".

Se pueden aplicar los algoritmos NER para la extracción de otros tipos de entidades, como en[19], donde se presenta una herramienta para el alemán cuyos algoritmos son capaces de identificar las entidades: "medicamento", "concentración", "vía", "forma", "dosis", "frecuencia" y "duración", relevantes para la comprensión de textos que incluyen tratamientos médicos.

NER es crucial para los sistemas de extracción de información y respuesta a preguntas.

7. Etiquetado de partes del discurso (POS): El etiquetado POS asigna etiquetas gramaticales (como sustantivo, verbo, adjetivo) a cada palabra de una oración. Ayuda en el análisis sintáctico y la desambiguación. En la Figura 1 se muestra el resultado que obtiene Freeling[19] para el texto "Yo me siento herida, mucho dolor en mi corazón":

8. Análisis sintáctico: Extrae la estructura sintáctica de las oraciones.

9. Análisis de dependencias: Se determinan las relaciones gramaticales entre las palabras de una oración: sujeto, complemento directo, complemento circunstancial. Se suelen representar gráficamente, conectando las palabras relacionadas e indicando con una etiqueta el tipo de relación. Por ejemplo, para la oración "La investigación es costosa pero aporta grandes beneficios", se obtiene el resultado que presentamos en la Figura 2.

10. Resolución de la correferencia: Identifica fragmentos de texto que se refieren a la misma entidad. Por ejemplo, en el texto: "Fui diagnosticado con deficiencia de vitamina D, el médico me lo diagnosticó. El doctor me la recetó". Se resaltan las correferencias: "lo" → "deficiencia de vitamina", "la" → "vitamina D".

La sucesión de tareas que se deben aplicar depende del problema en cuestión y de la tipología de la lengua. En algunos idiomas el orden de las palabras es muy relevante, otros son morfológicamente complejos. La configuración del *pipeline* y los algoritmos que se aplican darán mejores resultados si se adapta a las características del idioma. Por ejemplo, el inglés no es morfológicamente complejo por lo que en ese idioma la unidad básica puede ser la palabra, en lugar del lema o la raíz. Hay idiomas que tienden a la omisión de pronombres, como el español, sin embargo, en inglés esto no es así, por lo que en español es importante el tratamiento de la correferencia[17].

El resultado de aplicar estos análisis son textos enriquecidos con información respecto a la estructura y semántica del lenguaje. Sobre ellos se aplican algoritmos de aprendizaje automático que generan las clasificaciones, la extracción de información o cualquier otra aplicación objetivo. Los algoritmos de aprendizaje pueden ser supervisados o no supervisados, según se tengan datos de ejemplo de los que ya se conoce su categoría o la predicción que hay que hacer con ellos. En el caso del lenguaje, son necesarios enormes cantidades de ejemplos para configurar una muestra representativa de las infinitas posibilidades de expresiones que producen los hablantes de una lengua para garantizar un aprendizaje que conlleve la generalización de la información.

Actualmente, gracias a la potencia computacional que se ha desarrollado, es posible aplicar redes neuronales altamente complejas a volúmenes de información que abarquen cantidades ingentes de documentos. Los modelos generados se conocen como grandes modelos del lenguaje (en inglés LLM) con un alto grado de generalización en las capacidades de comprensión y generación de la lengua. Por ejemplo, el modelo GPT 3 se entrenó con *Commom Crawl* (repositorio de datos de todas las páginas en la web), WebText2 (información obtenida de la web por *OpenAI*), Books1 y Book2 (repositorios de libros) además de todos los artículos de la Wikipedia en inglés, sumando un total de aproximadamente 500.000 millones de *tokens*[21]. El entrenamiento de estos modelos tiene un alto

© SEMES

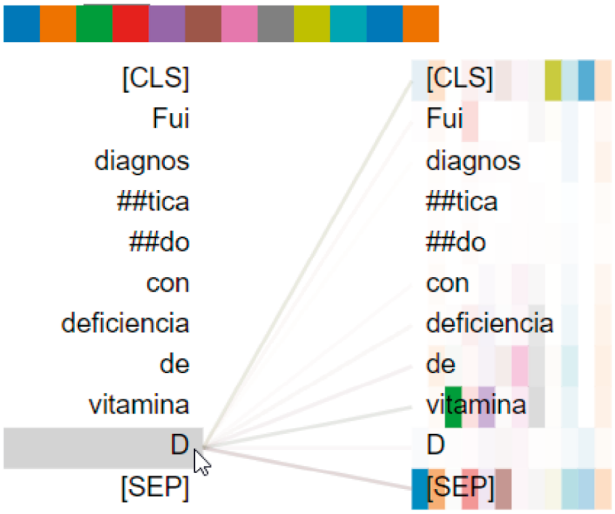


Figura 3. Atención computada por el modelo del lenguaje bert-base-spanish-wwm-cased para la palabra "D". Visualización generada con BertViz[25].

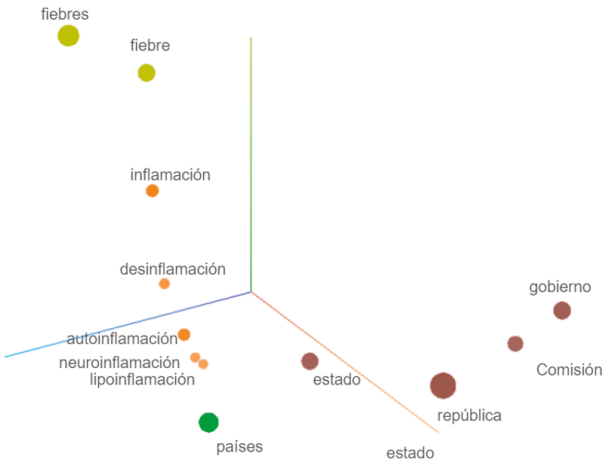


Figura 4. Representación de un conjunto de palabras en un espacio numérico tridimensional mediante incrustaciones de palabras obtenidas con el algoritmo Word2Vec[27].

coste en dinero, energía y tiempo, por lo que los modelos ya entrenados se adaptan a los dominios o tareas específicas en caso de que la estructura general captada en ellos no sea adecuada para el registro, la terminología, etc que se use en nuestros textos o el objetivo del aprendizaje. El procedimiento de adaptación de estas redes neuronales a nuestros textos se conoce el ámbito de los *LLMs* como *fine-tuning*.

Los modelos de lenguaje en su entrenamiento aprenden representaciones contextuales de palabras, transformándolas a vectores numéricos. Se destacan en tareas como completar texto, responder preguntas y generar texto. Entre los modelos más conocidos están BERT[21] o GPT[22]. El entrenamiento de un modelo se refiere al cómputo que realizan los algoritmos para ajustar la distribución de probabilidades de secuencias de palabras. Se busca que la probabilidad de la secuencia sea lo más alta posible según las colecciones de textos o *corpus* con los que se entrenan. La arquitectura denominada transformadores de estas redes neuronales ha supuesto un salto sustancial en la calidad de los modelos generados dando un alto grado de fiabilidad a las aplicaciones de PLN.

Ejemplo: "Había una vez un valiente caballero llamado \[MASK\]." (BERT completa la palabra enmascarada).

De forma intuitiva estamos trasladando el lenguaje escrito a un espacio vectorial, donde diferentes propiedades quedan representadas como números reales. En este campo podemos destacar los siguientes conceptos fundamentales para perfilar la tecnología subyacente:

1. Modelos de secuencia a secuencia: Son redes neuronales que manejan secuencias de entrada y salida de longitud variable con el objetivo de predecir palabras desconocidas en la entrada que se generan en la salida. Las predicciones pueden referirse a la traducción a otra lengua, o a completar la siguiente parte de una oración en la lengua propia, o bien a rellenar palabras faltantes en un fragmento de texto. Las redes neuronales recurrentes (RNN) y los modelos basados en transformadores entran en esta categoría.

2. Mecanismo de Atención: Introducida por el modelo *Transformer,* la atención permite que el modelo se centre en partes relevantes de la secuencia de entrada. Para ello computa valores indicando, para una palabra concreta, la importancia que tienen las restantes sobre ella en la comprensión de la secuencia[23]. En la Figura 3 se puede ver una representación de estos valores en los grosores de las líneas que unen la palabra "D" en la oración "Fui diagnosticado con deficiencia de vitamina D" con el modelo del lenguaje español *bert-base-spanish-wwm-cased*[24]. Destaca la división en *tokens* que fragmenta la palabra "diagnosticado" en los *tokens* "diagnos", "tica", "do". Se observa la importancia que tiene el *token* vitamina para comprender el significado del *token* D.

3. Incrustaciones de palabras: Las incrustaciones de palabras representan palabras o documentos como vectores densos en un espacio continuo, haciendo referencia al espacio vectorial numérico en el que queda representado el lenguaje. Los métodos populares incluyen Word2Vec, doc2vec, GloVe y FastText para obtener transformaciones semánticas. Los *Transformers* también generan representaciones vectoriales que mejoran a las anteriores por su capacidad de identificar los diferentes sentidos de las palabras polisémicas. Los sistemas actuales de PLN que usen técnicas de aprendizaje automático en su mayoría incluyen en su *pipeline* la generación de estas incrustaciones. Son especialmente útiles para capturar diversas relaciones semánticas, por ejemplo, la analogía: vector("rey") - vector("hombre") + vector("mujer") ≈ vector("reina"). En la Figura 4 se representan incrustaciones de algunas palabras generadas con Word2Vec[26].

4. Similitud textual: La similitud textual nos da un indicador de lo similares que son dos textos, bien sea en su escritura o en su significado. La similitud en la grafía habitualmente se calcula utilizando la distancia de Levenshtein, se obtiene del recuento de cambios que hay que realizar

© SEMES

para transformar una palabra en otra. Cuanto mayor es la distancia menor similitud. Por ejemplo: "inflamación" e "influencia" tienen una distancia de *Leveshtein* de 5, mientras que "inflamar" e "inflama" tienen una distancia de 1. Por otra parte, la similitud semántica es un indicador de lo similares que son los significados de dos palabras. Este concepto no se restringe a la sinonimia, sino que abarca cualquier tipo de relación semántica, como la analogía, la afinidad semántica entre las palabras, la hiperonimia, qué verbos predican de un sustantivo y muchas más. Existen diferentes enfoques respecto al cálculo de la similitud semántica.

4.1. Medidas de similitud que se calculan utilizando recursos léxicos como *thesaurus* u ontologías. La similitud entre dos palabras se mide en función de el camino que es necesario recorrer en la red para alcanzar el nodo de una palabra desde otra en una red semántica como WordNet o EuroWordNet[28]. WordNet es una red semántica de la lengua inglesa, en la que las palabras se organizan en conjuntos de sinónimos denominados *synsets*. Los nodos de la red son *synsets* representando conceptos entre los que se establecen relaciones semánticas como la hiperonimia, hiponimia, metonimia, holonimia y antonimia representadas como arcos de la red[29]. EuroWordNet es la versión de WordNet para un conjunto de lenguas europeas donde se establece las relaciones interlenguas de los *synsets* en cada idioma. Las estructuras de cada idioma se recogen en la versión de EuroWordNet[30] correspondiente, además incorpora una ontología común a todos los idiomas y las relaciones entre ellos.

4.2. Por otro lado, los modelos del lenguaje son capaces de generar espacios numéricos semánticos mediante las incrustaciones de palabras, en los que palabras con relaciones semánticas están próximas. Por esta razón se pueden computar medidas de similitud basadas en la distancia que separa sus incrustaciones. La medida más habitual en este caso es la similitud del coseno. En la Figura 5 se marcan los ángulos entre las incrustaciones de las palabras "inflamación", "gobierno" y "república". El coseno de cada uno de ellos será la similitud entre las palabras: inflamación-gobierno (ángulo rojo) e gobierno-república (ángulo azul). A menor distancia, menor ángulo, más cercano será su coseno a 1 y por tanto mayor similitud.

El concepto de similitud semántica es un componente fundamental en las siguientes aplicaciones de PLN: la recuperación de información, los sistemas pregunta-respuesta, la clasificación de texto, la generación automática de resúmenes.

Recuperación de bibliografía usando técnicas de PLN clásicas

En este apartado haremos uso de algunas de las diferentes técnicas antes mencionadas aplicándolas a la recuperación bibliográfica de la investigación del suicidio en jóvenes de la base de documentos de PubMed. Las líneas de código que se exponen al usar cada una de las técnicas solo pretenden mostrar que no son necesarios grandes

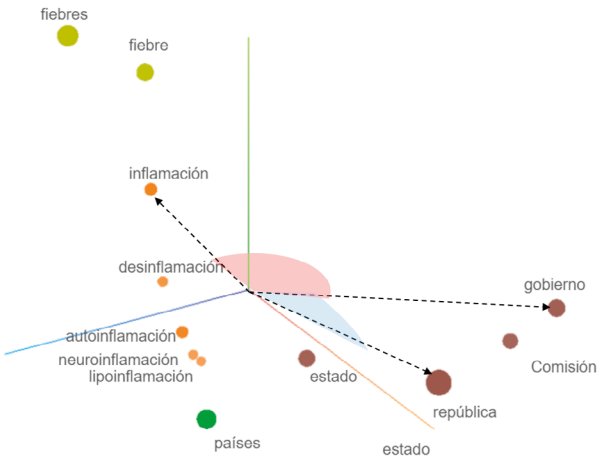


Figura 5. Ángulos que forman los vectores de incrustaciones de las palabras inflamación, gobierno y república[27].

conocimientos de programación. Las herramientas tipo Notebooks, como los Jupyter Notebooks o Google Colab, nos permiten ejecutar directamente estas instrucciones una vez le hayamos dado los datos correspondientes.

El objetivo de este ejemplo es mostrar al lector como utilizando técnicas de PLN se pueden acelerar algunas tareas del proceso de revisión bibliográfica. Concretamente los pasos seguidos son los siguientes:

1) Realizar una búsqueda de todos los artículos de la base de datos PubMed que tengan en su título la palabra "suicide".

Para ello, PubMed cuenta con una manera sencilla de extraer tus búsquedas en un fichero con formato csv preconfigurado. Las búsquedas permiten una serie de filtros tales como el año de publicación, la revista e incluso búsqueda por palabras clave.

En nuestro caso, buscamos publicaciones desde 2015 que contuviesen la palabra suicidio, extrayendo un csv con los campos ID, título, autor, revista, año de publicación y DOI.

En el resto del ejemplo, se usarán palabras y frases en español por facilidad de lectura de estos apartados.

2) Realizar el preprocesado y obtener cada una de las palabras o *tokens* del título (tokenización).

En el apartado anterior, se establece un *pipeline* generalizado cuando trabajamos técnicas de procesamiento de lenguaje natural. Este paso, incluye los apartados 1 y 3 mencionados en el *pipeline*.

Se convierten todos los títulos a minúsculas, quitamos comillas y signos tales como @,#, etc. y, a partir de este preprocesado, tokenizamos y obtenemos cada una de las palabras del título.

3) Eliminar las palabras comunes y no informativas de cada título (por ejemplo, "y", "el", "de") para centrarse en las palabras clave relevantes (eliminación de *stop words*).

Este apartado de limpieza de palabras comunes es un paso muy útil en nuestro caso de uso, pero no siempre es necesario. Para el filtrado de documentos por similitud de palabras, será necesario vectorizar y también contar las ve-

© SEMES

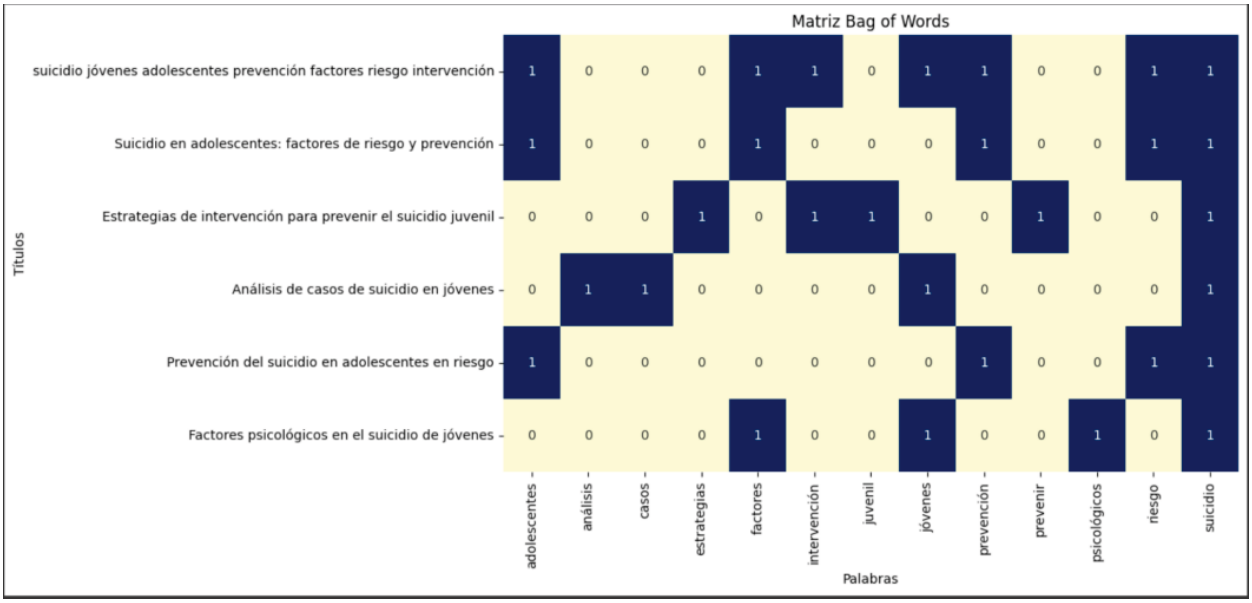


Figura 6. Matriz *Bag of Words* ejemplificando el conteo de palabras exactas.

ces que aparecen ciertas palabras. Es por ello, que la limpieza de palabras repetitivas que no aportan valor contextual será clave para no distorsionar los resultados obtenidos.

4) Reducir las palabras de cada título a su forma raíz para mejorar la consistencia de la búsqueda (lematización).

Este paso se corresponde con la quinta fase del *pipeline* que detallamos en el apartado 3. Tras su aplicación, tendremos todas las frases de los títulos que contienen la palabra "suicidio" tokenizadas y lematizadas.

5) Selección manual de palabras a relacionar con "suicidio" y que deben estar en las frases de los títulos procesados encontrados en PubMed.

Para el caso de uso en curso, se seleccionan las palabras más frecuentes y esclarecedoras en un subconjunto de las publicaciones relacionadas con el tema analizado, obteniendo el conjunto de palabras:

adolescentes | jóvenes | estudiantes | tendencias | medios | niños | escuelas | máquinas | redes | adolescencia | internet | tendencia | usuarios | móviles | redes | twitter | artificial | nota.

Para una mayor cobertura, estas se expanden con sinónimos obtenidos de EuroWordNet en la versión del español. El siguiente trozo de código en Python muestra esta tarea:

Código:

```
# Paso : Lista de palabras clave iniciales
keywords = ['suicidio', 'jóvenes', 'adolescentes', 'prevención', 'factores de riesgo', 'intervención']
# Función para encontrar sinónimos usando WordNet
def get_synonyms(word):
    synonyms = set()
    for syn in wn.synsets(word, lang='spa'):
        for lemma in syn.lemmas(lang='spa'):
            synonyms.add(lemma.name())
    return synonyms
get_synonyms(word='adolescente')
```

Salida:

```
{'adolescente',
 'chavea',
 'imberbe',
 'joven',
 'juvenil',
 'mozo',
 'muchacho',
 'pollo',
 'zagal'}
```

Y el código ampliando la lista de palabras con Wordnet es:

```
# Paso 2: Expansión de la lista de palabras clave
expanded_keywords = set(keywords)
for keyword in keywords:
    synonyms = get_synonyms(keyword)
    expanded_keywords.update(synonyms)
print(f"Palabras clave expandidas: {expanded_keywords}")
```

Salida:

{'cautela', 'factores de riesgo', 'operación', 'cuidado', 'adolescentes', 'suicidio', 'precaución', 'prevención', 'intervención', 'jóvenes', 'previsión'}

6) Comprobar qué palabras manuales aparecen en los títulos encontrados en PubMed.

Este paso se puede hacer directamente con una comprobación exacta de los textos implicados o utilizando métricas de similitudes que nos podrán cuantitativamente dar un valor de cuánto se parecen ambas palabras.

Si optamos por la comprobación exacta, podemos mostrar un ejemplo visual de cómo podemos testarlo en la Figura 6.

En este caso, para el conteo se puede utilizar la librería *sklearn*, muy reconocida en el mundo del *Machine Learning*, en concreto, el paquete CountVectorizer.

7) Optando por una estrategia de similitud, las frases de los títulos obtenidas del paso 4 las convertimos en

© SEMES

```
suicidio jóvenes adolescentes prevención factor...    1.000000
Suicidio en adolescentes: factores de riesgo y ...    0.714286
Estrategias de intervención para prevenir el su...    0.267261
Análisis de casos de suicidio en jóvenes              0.251976
Prevención del suicidio en adolescentes en riesgo     0.503953
Factores psicológicos en el suicidio de jóvenes       0.428571
```

Figura 7.

colecciones de vectores de frecuencias (números) que nos permitan calcular la distancia semántica.

En el apartado 3 de la presente publicación, se hablaba de similitud textual, será en este paso donde aplicaremos diversas técnicas mencionadas en dicho apartado. En concreto hablamos del apartado 10 del *pipeline*. Tenemos el título de cada publicación convertido en *tokens* y tenemos nuestra lista de palabras clave, lo que haremos será calcular la similitud entre esa lista de *tokens* del título con nuestra lista de palabras para detectar su aparición.

Veamos un ejemplo:

Si tenemos los siguientes títulos:

titles = [

"Suicidio en adolescentes: factores de riesgo y prevención",

"Estrategias de intervención para prevenir el suicidio juvenil",

"Análisis de casos de suicidio en jóvenes",

"Prevención del suicidio en adolescentes en riesgo",

"Factores psicológicos en el suicidio de jóvenes"

]

Vectorizamos para crear la matriz de frecuencias de palabras:

Vectorización utilizando Bag of Words

vectorizer = CountVectorizer()

bow_matrix = vectorizer.fit_transform(filtered_titles)

Utilizamos la distancia coseno explicada en el apartado 3:

Similitud de coseno

cosine_similarities = cosine_similarity(bow_matrix)

Crear un DataFrame para visualizar las similitudes

cosine_sim_df=pd.DataFrame(cosine_similarities,index=filtered_titles, columns=filtered_titles)

Visualización de la matriz de similitud de coseno

print(cosine_sim_df)

Para obtener los títulos más similares a un título específico:

title_to_compare = "suicidio jóvenes adolescentes prevención factores riesgo intervención"

similar_titles=cosine_sim_df[title_to_compare].sort_values(ascending=False)

print(similar_titles)

Obtenemos lo que observamos en la Figura 7.

Como vemos, la similitud del coseno nos indica cuánto de parecidas son las palabras seleccionadas manualmente con las frases de los títulos de las publicaciones. A mayor número, mayor similitud. Normalmente seleccionamos aquellas por encima de un umbral establecido.

8) De las publicaciones obtenidas en el punto 7, nos interesan aquellas relacionadas con "AI Model" y con "Machine Learning". Volvemos a calcular la distancia de estos dos términos individualmente y calculamos la similitud con las frases de todos los títulos obtenidos del paso 7.

9) Finalmente se hace lo mismo para "nota suicidio".

En resumen, el *pipeline* o flujo de proceso seguido en este ejemplo permite pasar de 9.212 publicaciones iniciales con la palabra "suicidio" a las 74 publicaciones finales que además tienen las palabras consideradas relevantes y manualmente seleccionadas. La Figura 8 muestra el *pipeline* o flujo del proceso.

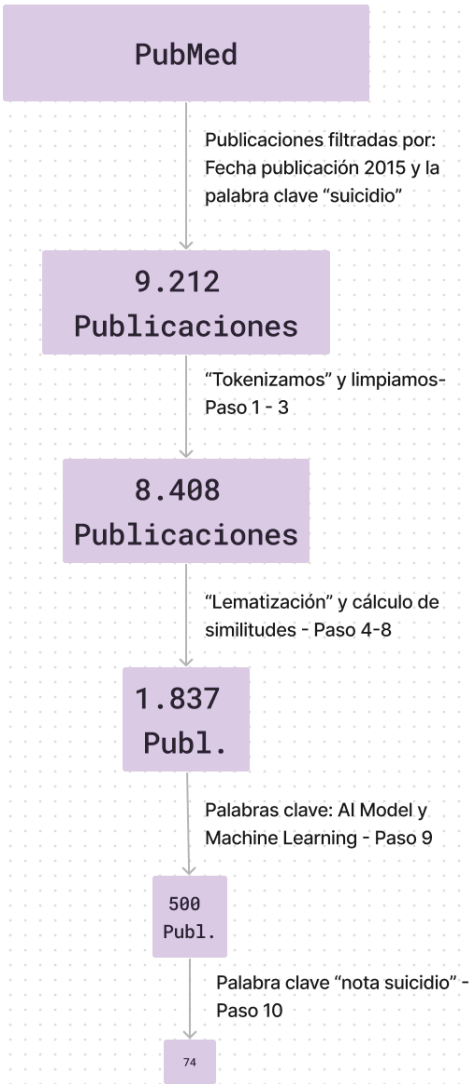


Figura 8. *Pipeline* o flujo del proceso seguido y resultados obtenidos.

© SEMES

Revisión de bibliografía usando IA generativa

El avance de la IA generativa ha hecho que el proceso indicado en el apartado anterior, que implica una ardua limpieza de datos y el uso de varios modelos avanzados de aprendizaje automático, hoy sea tan sencillo como preguntarle a un sistema conversacional tipo ChatGPT, Llama, Gemini, etc. Un sistema de IA generativa recibe una pregunta del usuario, busca información en el *corpus* sobre el que ha sido entrenado y es capaz de "generar" una respuesta en lenguaje natural de fácil comprensión por un humano. Uno de los componentes más importantes son las instrucciones o *prompts,* hasta el punto de que ha aparecido una disciplina denominada ingeniería de *prompts* que consiste en estrategias y técnicas para hacer las preguntas al sistema de IA. En este apartado, mostramos un ejemplo sencillo relacionado con nuestra recuperación de bibliografía del tema del suicidio en los jóvenes.

Veamos un caso con diferentes *prompts* que mantienen coherencia entre las preguntas que se hacen secuencialmente (es lo que denominados contexto). Los ejemplos son similares a los expuestos en el apartado anterior, pero haciendo uso del sistema ChatGPT:

Usuario: Si adjunto un archivo de publicaciones, me puedes decir aquellas relacionadas con el suicidio en jóvenes donde aparezcan las palabras suicidio, nota, inteligencia artificial y aprendizaje automático.

ChatGPT: Claro, puedo ayudarte con eso. Por favor, adjunta el archivo de publicaciones y procederé a buscar menciones de palabras clave relacionadas con el suicidio en jóvenes, concretamente buscaré las palabras "suicidio", "nota", "inteligencia artificial" y "machine learning".

Una vez adjuntado el archivo con los títulos de las publicaciones, la respuesta obtenida del agente conversacional es la siguiente:

ChatGPT: Gracias por facilitarnos la lista de publicaciones. Ahora revisaremos los títulos para identificar aquellos relacionados con el suicidio en jóvenes y que contengan las palabras clave "suicidio", "nota", "inteligencia artificial" y "aprendizaje automático".

Sin embargo, parece que ninguna de las entradas proporcionadas contiene explícitamente las palabras "inteligencia artificial" o "aprendizaje automático". Aun así, enumeraré las publicaciones que se centran en el suicidio juvenil:

32699044 - "Acceso a las armas de fuego y suicidio adolescente"

35156223 - "Intentos de suicidio de adolescentes en Estados Unidos: Cuando interactúan la ideación suicida y la capacidad suicida".

30771807 - "Suicidio en jóvenes indígenas: una crisis sin paliativos"

36822834 - "Ciberacoso y suicidio adolescente"

Ninguna de estas publicaciones menciona "nota", "inteligencia artificial" o "aprendizaje automático" en sus títulos, pero se centran en el tema del suicidio en jóvenes. Si necesita una búsqueda más detallada o específica dentro del texto completo de estas publicaciones, necesitaría tener acceso al contenido íntegro de los artículos.

Si puede proporcionarme más detalles o fragmentos de texto adicionales, estaré encantado de ayudarle más a fondo.

Ahora, el sistema puede solicitar más información de las publicaciones a analizar:

ChatGPT: Gracias por proporcionarnos estos títulos adicionales. Aquí hay publicaciones relacionadas con el suicidio en jóvenes que mencionan "suicidio", "nota", "inteligencia artificial" y "aprendizaje automático":

31079565 - "Using Machine Learning to Identify Suicide Risk: A Classification Tree Approach to Prospectively Identify Adolescent Suicide Attempters".

Esta publicación se ajusta perfectamente a sus criterios, ya que menciona "suicidio", "aprendizaje automático" y se centra en los jóvenes (adolescentes).

Si necesita más detalles sobre esta publicación en concreto o cualquier información adicional, no dude en comunicármelo.

Vemos como el usuario puede realizar las peticiones de forma simple, sin importar el nivel de conocimientos informáticos que tenga. También podemos apreciar que las respuestas serán acordes a la colección de documentos de los que dispone el sistema. No sólo esto, sino que devolverá información más precisa y fiable cuanto mayor sea el contexto que proporcionemos. Este tipo de peticiones que hacemos a agentes conversacionales basados en *LLMs* son los *prompts,* y pueden variar desde una simple pregunta directa, sin más información (por ejemplo, "Quiero una colección textos sobre suicidio y *machine learning*") a otros más específicos en los que se agregue contexto: "Necesito saber, de esta colección de ficheros que incluye artículos científicos, cuáles aplican técnicas de *Machine Learning,* por ejemplo, Lematización, POS, NER, incrustaciones, similitud semántica y clasificación. En el artículo se debe analizar el texto de notas de suicidio, por lo que es probable que también se trate el tema del preprocesamiento y además se expongan métricas de resultados". Se aprecia en estos ejemplos la facilidad con la que se pueden establecer filtros complejos a información que no está estructurada, poniendo de manifiesto la potencia de estas herramientas para construir sistemas de consulta que no supongan una barrera para el usuario. En el siguiente capítulo se muestra cómo se ha usado el PLN para la construcción de un *bot* de *Telegram* con capacidades similares a los *prompts.*

Consultas al *corpus* bibliográfico seleccionado a través los *chatbots*

Los sistemas conversacionales *(chatbots)* bien conocidos como ChatGPT, Copilot, etc., ofrecen herramientas e interfaces para poder entablar diálogos entre un humano y la IA correspondiente. Los Modelos de Lenguajes que hay detrás de estos sistemas son generalistas y, por tanto, si queremos personalizarlos con documentos/información propia se debe adoptar alguna estrategia para que el sistema adquiera ese conocimiento y pueda responder a nuestras preguntas.

© SEMES

Existen diferentes plataformas que ofrecen librerías de programación con funciones especializadas para extraer texto de documentos propios y ofrecerlas al usuario a través de la IA generativa[31-33]. El conocimiento propio se ofrece al modelo de lenguaje grande y él es capaz de integrarlo con su propio modelo generalista para que nos ofrezca respuestas concretas de nuestro *corpus*. Estos sistemas se denominan de Generación de Recuperación Aumentada (RAG)[34]. En este apartado mostramos un ejemplo de *chatbot* de *Telegram* para crear un sistema conversacional del ámbito de la investigación usado a lo largo de este trabajo: el suicidio juvenil.

Los trabajos seleccionados según la metodología expuesta, forman un *corpus* de conocimiento sobre el que un investigador comenzará a realizar su investigación. La consulta de estos documentos normalmente se realiza de forma manual para poder profundizar en los detalles de cada uno. Crear un sistema de preguntas y respuestas sobre ese *corpus* de conocimiento resulta eficiente y en un ahorro de tiempo que puede ser considerable.

Basándonos de nuevo en los Modelos de Lenguajes Grandes, hemos construido un sistema conversacional para la consulta de cualquier contenido del *corpus* de investigación. La facilidad de uso del sistema es la principal característica, ya que se basa en interfaces que todos los usuarios suelen estar acostumbrados. Concretamente, se ha diseñado un *bot* de *Telegram* al que se le hacen las preguntas de investigación correspondientes y, una vez consultados los documentos del repositorio previamente seleccionado, tendríamos la información requerida. El siguiente paso es responder al usuario en su propio lenguaje los hallazgos que responden a sus preguntas.

En la Figura 9 se muestra un diagrama de interacción del sistema conversacional. Los requisitos básicos de este sistema son:

– Servidor de preguntas de *Telegram*.

– Base de datos de documentos de investigación.

– Servidor de IA generativa mediante modelo de lenguaje grande para las respuestas.

– Servidor de respuestas mediante recuperación aumentada.

El servidor de *Telegram* se instala en una máquina que estará esperando a que el usuario realice una pregunta. Cualquier persona puede realizar las preguntas si conoce el nombre del *bot*. Las preguntas son almacenadas en una cola pendiente de procesar. Otro servidor posee la base de datos con los trabajos de investigación que han pasado por algunas transformaciones para poder calcular similitudes entre los documentos. La transformación más importante es la conversión de un documento concreto dónde lo dividiremos en trozos semánticamente similares. Estos trozos, denominados *chunks* se guardan en una base de datos en formatos numéricos gracias a diferentes métodos proporcionados por librerías de incrustaciones *(embeddings)*. Cuando un usuario realiza una pregunta, el *bot* pasa el control al programa principal que transforma dicha pregunta a un formato numérico *(embeddings)*. En este momento hace una consulta a la base de datos de documentos en formato numérico buscando aquellos *chunks* más cercanos a la representación numérica de la pregunta realizada. El resultado de la consulta serán los n *chunks* más cercanos por similitud semántica. Finalmente, se pasan los trozos de textos de los artículos encontrados al proceso generativo para que basado en el *LLMs* se genere una respuesta bien enunciada con la información requerida. Esta respuesta se envía de nuevo a *Telegram* para que sea entregada al usuario en el chat correspondiente. En la Figura 10 se muestra en detalle el flujo de información dentro de todo el proceso.

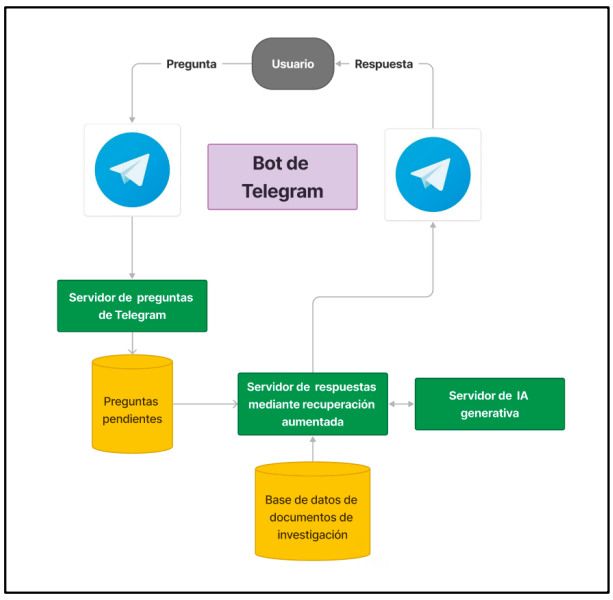


Figura 9. Esquema básico del sistema basado en *Telegram*.

Veamos un ejemplo de la experiencia del usuario.

1. Realicemos la pregunta al *bot* de *Telegram* (Figura 11).

2. El *bot* realiza las operaciones internas y ofrece como respuesta, aquellos artículos que disponen de contenido relacionado con la pregunta (Figura 12).

El bot ha revisado los trozos de textos de estos artículos relacionados con la pregunta y se los ha pasado al LLM que genera la respuesta en lenguaje natural mostrada en la Figura 13.

Este sistema nos permite hacer preguntas de forma coloquial al sistema sobre los artículos que hemos seleccionado a través de las técnicas anteriormente descritas. Sobra decir que, si bien el sistema es generalista en cuanto a que sirve para todo tipo de ámbitos científicos, la base de datos de artículos determinará concretamente ese ámbito. Por tanto, cada investigador o grupo de investigación podrían tener uno o más *bots* para las diferentes investigaciones que realizan.

Conclusiones

La metodología propuesta reduce significativamente los tiempos y las tareas para la selección inicial de trabajos de investigación de bases de datos bibliográficas.

© SEMES

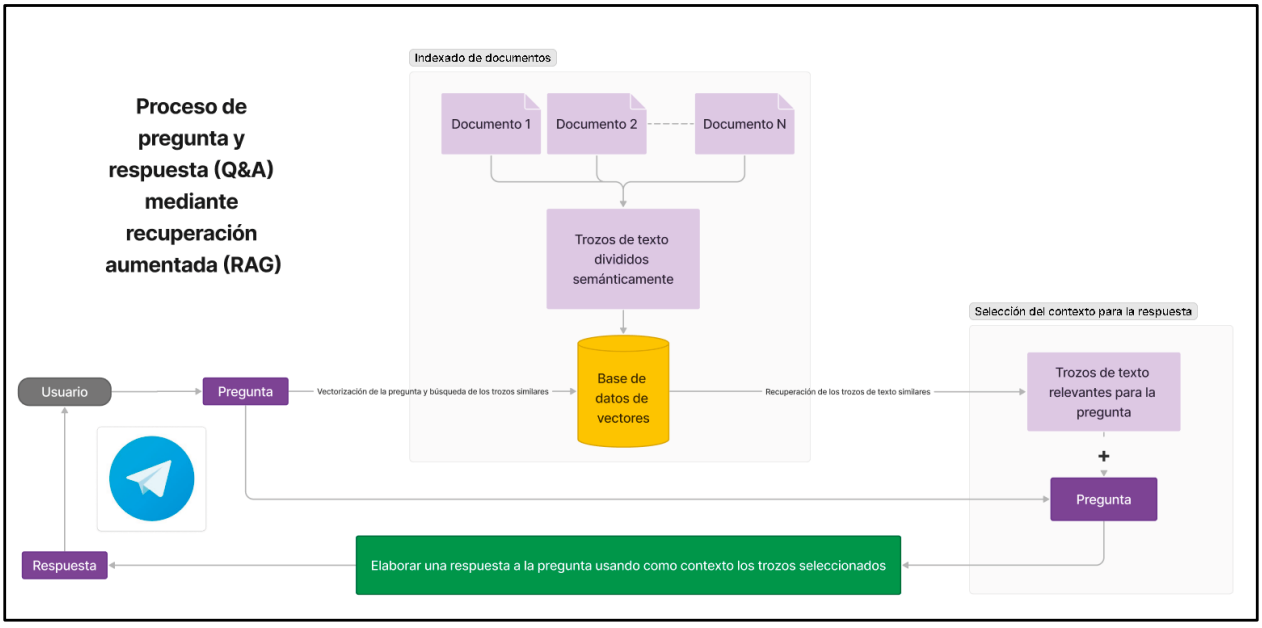


Figura 10. Flujo de interacción del sistema de preguntas y respuestas (Q&A).

Es complementaria a las metodologías tipo PRISMA, ya que se encarga de las primeras tareas de búsqueda a través de una selección automática de la bibliografía de una temática específica propuesta por un investigador.

Si bien las técnicas clásicas de aprendizaje automático y PLN son muy efectivas, es necesario tener unos mínimos conocimientos de programación o de uso de *notebooks* tipo *Jupyter*.

Las técnicas basadas en los *LLMs* para buscar y selección de trabajos de investigación son muy cercanas a los no iniciados en la programación ya que permiten interactuar con un sistema a través de preguntas vía ChatGPT u otros.

Un sistema de pregunta respuesta personalizado puede hacer más eficiente las búsquedas de información dentro de nuestro *corpus* de conocimiento.

En resumen, las técnicas de IA, concretamente de PLN y *LLMs*, prometen ser una potente herramienta de ayuda a los investigadores en las tareas de búsqueda y selección de trabajos de investigación de las bases de datos de bibliografía.

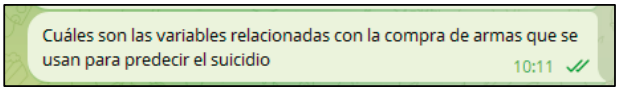


Figura 11.

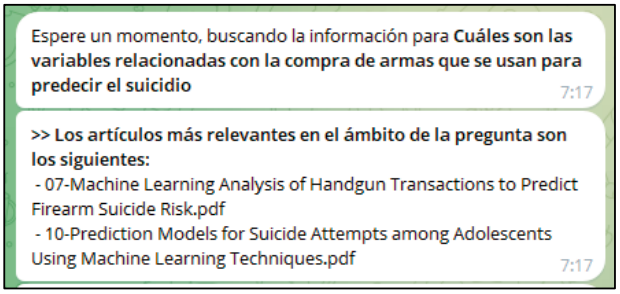


Figura 12.

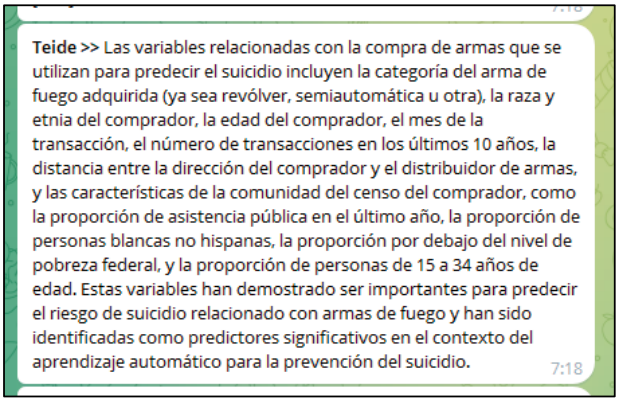


Figura 13.

INFORMACIÓN DEL ARTÍCULO

Conflicto de intereses: Los autores declaran no tener conflicto de interés en relación con el presente artículo.

Financiación: Los autores declaran la no existencia de financiación en relación con el presente artículo.

Responsabilidades éticas: Todos los autores han confirmado el mantenimiento de la confidencialidad y respeto de los derechos de los pacientes, acuerdo de publicación y cesión de derechos de los datos a la Revista Española de Urgencias y Emergencias.

Artículo no encargado por el Comité Editorial y con revisión externa por pares.

AGRADECIMIENTOS

Los autores quieren agradecer a la Cátedra Cajasiete-ULL de Big data, Open Data y Blockchain, a la Cátedra en Telemedicina, Robótica y Telecirugía Fundación Canaria Kishoo-Universidad de La Laguna su apoyo en el desarrollo de este trabajo.

BIBLIOGRAFÍA

1. Turing A. Computing Machinery and Intelligence. Mind LIX. 1950;236:433–60 doi:10.1093/mind/LIX.236.433

© SEMES

2. Tommasel A, Pablos Sarabia R, Assent I. Re2Dan: Retrieval of Medical Documents for e-Health in Danish. In Proceedings of the 17th ACM Conference on Recommender Systems, 2023; 1208-1211. doi: https://doi.org/10.1145/3604915.3610655
3. Carvallo A, Parra D, Lobel H, Soto A. Automatic document screening of medical literature using word and text embeddings in an active learning setting. Scientometrics. 2020;125:3047-84.doi: https://doi.org/10.1007/s11192-020-03763-4
4. Peikos G, Alexander D, Pasi G, de Vries AP. Investigating the Impact of Query Representation on Medical Information Retrieval. En: European Conference on Information Retrieval, pp. 512-521. Cham: Springer Nature Switzerland, 2023. doi: https://doi.org/10.1007/978-3-031-28238-6_42
5. Ganguly D, Conrad JG, Ghosh K, Ghosh S, Goyal, et al. Legal IR and NLP: the history, challenges, and state-of-the-art. En: European Conference on Information Retrieval, pp. 331-340. Cham: Springer Nature Switzerland; 2023. doi: https://doi.org/10.1007/978-3-031-28241-6_34
6. Parmar M, Jain N, Jain P, Jayakrishna Sahit P, Pachpande S, Singh S, et al. NLPExplorer: exploring the universe of NLP papers. En: Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal Proceedings, Part II 42, pp. 476-480. Springer International Publishing; 2020. doi: https://doi.org/10.1007/978-3-030-45442-5_61
7. Kelly L, Suominen H, Goeuriot L, Neves M, Kanoulas, et al. Overview of the CLEF eHealth evaluation lab 2019. En: Experimental IR Meets Multilinguality, Multimodality, and Interaction: 10th International Conference of the CLEF Association, CLEF 2019, Lugano, Switzerland, Proceedings 10, pp. 322-339. Springer International Publishing; 2019. doi: https://doi.org/10.1007/978-3-030-28577-7_26
8. PubMed, National Center for Biotechnology Information (NCBI). (Consultado 12 Junio 2024). Disponible en: https://pubmed.ncbi.nlm.nih.gov/
9. Reiter E, Dale R. (1997). Building applied natural language generation systems. Natural Language Engineering. 1997;3:57-87. doi: 10.1017/S1351324997001502
10. Sebastiani F. (2005). Text categorization. In Encyclopedia of database technologies and applications, pp. 683-687. IGI Global. doi: 10.4018/978-1-59140-560-3.ch112
11. Schouten K, Frasincar F. Survey on aspect-level sentiment analysis. IEEE transactions on knowledge and data engineering. 2015;28:813-30. doi: 10.1109/TKDE.2015.2485209
12. El-Kassas WS, Salama CR, Rafea AA, Mohamed HK. Automatic text summarization: A comprehensive survey. Expert systems with applications. 2021; 165:113679. doi: https://doi.org/10.1016/j.eswa.2020.113679
13. Grishman R. Information extraction. IEEE Intelligent Systems. 2015;30:8-15. doi: https://doi.org/10.1109/MIS.2015.68
14. Caballero,M. A brief survey of question answering systems. International Journal of Artificial Intelligence & Applications (IJAIA). 2021;12(5). doi: https://doi.org/10.1109/MIS.2015.68
15. Kusal S, Patil S, Choudrie J, Kotecha K, Mishra S, Abraham A. AI-based conversational agents: A scoping review from technologies to future directions. IEEE Access. 2022;10:92337-56. doi: 10.1109/ACCESS.2022.3201144
16. Agarwal R, Wadhwa M. (2020). Review of state-of-the-art design techniques for chatbots. SN Computer Science. 2020;1:46. doi:https://doi.org/10.1007/s42979-020-00255-3
17. Jurafsky D, Martin JH. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition (3ª ed draft). 2024. (Consultado 17 Junio 2024). Disponible en: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf
18. Frei J, Kramer F. GERNERMED: An open German medical NER model. Software Impacts. 2022;11:100212. doi: https://doi.org/10.1016/j.simpa.2021.100212
19. Padró L, Stanilovsky E. Freeling 3.0: Towards wider multilinguality. In LREC2012, 2012.
20. Paul Mooney, 2019. Medical Symptoms Text and Audio Classification in Kaggle. (Consultado 17 Junio 2024). Disponible en: https://www.kaggle.com/code/paultimothymooney/medical-symptoms-text-and-audio-classification
21. Devlin J, Chang MW, Lee K, Toutanova KB. Pre-training of deep bidirectional transformers for language understanding in: Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers). Minneapolis, MN: Association for Computational Linguistics. 2019;4171-86. doi: 10.18653/v1/N19-1423
22. Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language models are few-shot learners. Advances In Neural Information Processing Systems. 2020;33:1877-901.
23. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, et al. Attention is all you need. Advances In Neural Information Processing Systems. 2017;30.
24. BETO: Spanish Bert dccuchile/bert-base-spanish-wwm-cased in Hugging Face. (Consultado 17 Junio 2024). Disponible en: https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased
25. Vig J. BertViz, herramienta para la visualización de la atención en modelos BERT. (Consultado 18 Junio 2024). Disponible en: https://github.com/jessevig/bertviz
26. Mikolov T, Chen K, Corrado G, Dean J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. doi: https://doi.org/10.48550/arXiv.1301.3781
27. Tatman, R. Pre-trained Word Vectors for Spanish (Dataset Over 1 million 300-dimensional word vectors for Spanish) in Kaggle. (Consultado 14 Junio 2024). Disponible en: https://www.kaggle.com/datasets/rtatman/pretrained-word-vectors-for-spanish?resource=download
28. Budanitsky A. Lexical semantic relatedness and its application in natural language processing. technical report CSRG-390, Department of Computer Science, University of Toronto, 1999.
29. Miller GA. WordNet: a lexical database for English. Communications of the ACM. 1995;38:39-41. doi: https://doi.org/10.1145/219717.219748
30. EuroWordNet Project. (Consultado 18 Junio 2024). Disponible en: https://archive.illc.uva.nl/EuroWordNet/data/sampleData.htm.
31. Open AI. (Consultado 12 Junio 2024). Disponible en: https://platform.openai.com/
32. Meta. (Consultado 12 Junio 2024). Disponible en: https://llama.meta.com/
33. Langchain. (Consultado 12 Junio 2024). Disponible en: https://www.langchain.com/
34. Lewis PSH, Perez E, Piktus A, Petroni F, Karpukhin V, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. CoRR. 2020;vol. abs/2005.11401. [Online]. doi: https://doi.org/10.48550/arXiv.2005.11401

© SEMES