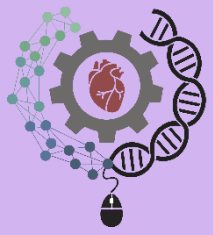
ESCUELA POLITÉCNICA SUPERIOR

Resumidor de artículos científicos del ámbito biomédico



Grado en Ingeniería Biomédica

Trabajo Fin de Grado

Autor:
Héctor Valdés Ferrer

Tutor/es:
Elena Lloret Pastor

Julio 2023

Universitat d'Alacant
Universidad de Alicante

RESUMEN

Vivimos en una época que experimenta cambios tecnológicos que evolucionan constantemente en gran parte por el crecimiento vertiginoso del ciberespacio, donde se disponen recursos de cualquier índole. Cada vez la información en forma textual se encuentra más digitalmente y es accesible a través de internet donde se almacenan enormes cantidades de datos, por lo que el tiempo es un bien preciado para cualquier persona que esté navegando e indagando la información deseada. En el campo sanitario, el tiempo todavía es más notorio y los tiempos de espera tienen que ser mínimos para ofrecer un buen servicio. Por este motivo, la importancia de que aparezcan métodos desarrollados los cuales permitan encontrar y averiguar de forma rápida la información comprendida en extensos documentos, ya que para el personal biomédico implica una gran optimización en tiempo de lectura. La generación automática de resúmenes mono documento de carácter extractivo desempeña esa función, puesto que consiste en un proceso mediante una aplicación informática la cual permite reducir del documento seleccionado en un texto breve la información, destacando las partes más relevantes del contenido y contribuyendo a disminuir la sobrecarga de información.

Centrándonos en la numerosa cuantía de documentos científicos y la extensión completa contenida en cada uno de ellos, sin resúmenes que identifiquen o destaquen el título del artículo, el autor, la editorial, revistas o diversas plataformas del ámbito biomédico sería prácticamente inviable acceder a la información utilizable que nos brinda internet.

En este trabajo fin de grado se presenta una arquitectura definiendo la herramienta Appenecum, desarrollando un método capaz de resumir artículos de textos del ámbito científico de forma automática, con la premisa de que los artículos deben estar relacionados con el covid y en formato PDF, prestando atención en el resumen extractivo. La finalidad de este trabajo es definir un procedimiento que sea capaz de ponderar cuán importante son las oraciones compuestas en el documento de forma similar a como lo haría un médico especialista de manera manual.

La utilidad de nuestra herramienta Appenecum ha sido demostrada por la realización de una evaluación y midiendo la eficacia del rendimiento del resumen generado frente a los resúmenes originales mediante las métricas Rouge, alcanzando unos porcentajes favorables.

Palabras clave: información, resúmenes, herramienta Appenecum

ABSTRACT

We live in an age of constantly evolving technological change, largely due to the rapid growth of cyberspace, where resources of all kinds are available. Information in textual form is increasingly available digitally and accessible through the internet where huge amounts of data are stored, making time a precious commodity for anyone browsing and searching for the desired information. In the healthcare field, time is even more of an issue and waiting times need to be kept to a minimum in order to provide a good service. For this reason, the importance of the appearance of developed methods which make it possible to find and find out quickly the information contained in extensive documents, for biomedical personnel, implies a great optimisation of reading time. The automatic generation of extractive single-document summaries performs this function, as it consists of a process using a computer application to reduce the information from the selected document to a short text, highlighting the most relevant parts of the content and helping to reduce the information overload.

Focusing on the large number of scientific documents and the full length contained in each of them, without summaries that identify or highlight the title of the article, the author, the publisher, journals or various platforms in the biomedical field, it would be practically unfeasible to access the usable information provided by the Internet.

In this final degree project we present an architecture defining the Appenecum tool, developing a method capable of summarising articles from scientific texts automatically, with the premise that the articles must be related to the covid and in PDF format, paying attention to the extractive summary. The aim of this work is to define a procedure that is able to weight how important the compound sentences are in the document in a similar way as a medical specialist would do it manually.

The usefulness of our Appenecum tool has been demonstrated by performing an evaluation and measuring the efficiency of the performance of the generated summary against the original summaries using Rouge metrics, reaching favourable percentages.

Keywords: information, summaries, Appenecum tool

AGRADECIMEINTOS

En primer lugar, me gustaría dar las gracias a mi tutora, Elena Lloret Pastor, por el gran apoyo realizado durante todo el periodo de realización de este proyecto con un gran mérito debido a las circunstancias que atraviesa. Sin ella este proyecto no se hubiera podido llevar a cabo.

Además, quiero gratificar a mi familia por todo el apoyo moral, sobre todo a mi actual pareja, Rosa Buitrón, por su contribución anímica.

Agradecer la aportación del doctor experto en medicina Felipe Navarro Cremades con número de colegiado 033834 por su colaboración en este trabajo que me ha servido como punto de partida para la realización de mi método en la herramienta Appenecum.

Por último, me gustaría dar las gracias a mis compañeros, Román Sumin, Mariola Navarro y Roberto Alcaraz por toda la ayuda recibida durante la carrera, y también mencionar a Joan Salas y Cristina Muñoz por este último tramo de la carrera.

Índice de contenido

Índice de ilustraciones

Índice de tablas

1 INTRODUCCIÓN

Este punto introductorio se compone de tres subsecciones donde se expone el problema que vamos a abordar, la solución al problema en cuestión y el impacto que se espera una vez planteado el problema en un contexto biomédico.

1.1 Definición del problema

En la actualidad existe mucha variedad de información de prácticamente todo tipo que disponemos de fácil acceso para buscar y encontrar referencias a través de la web. A medida que transcurre el tiempo se almacenan grandes volúmenes de documentos y en estos últimos años hemos obtenido un crecimiento exponencial a causa de la masificación de disponibilidad de estos.

Este año 2023, seguimos con las secuelas de una de las peores pandemias recordadas a causa del COVID [1] y es por eso por lo que nuestro sistema nacional de salud (SNS) [2] debe estar preparado para afrontar con efectividad los nuevos desafíos en las distintas áreas de sanidad, como en biomedicina e investigación y sobre todo disponer con eficiencia herramientas para el procesamiento del lenguaje natural (PLN) [3], capaces de clasificar y recuperar documentos biomédicos, analizar y estructurar historiales clínicos, relacionar las bases de datos de pacientes con sistemas informáticos específicos de salud, entre otras muchas posibilidades que pueden ser llevados a cabo en centros hospitalarios.

La aportación de conocimiento en el ámbito sanitario viene dada mayormente a través de publicaciones científicas ya sea en revistas científicas, posts sanitarios, diccionarios médicos que son accedidos por medio de la web, bases de datos reconocidas como PubMed-Medline [4], ScienceDirect [5], Embase, Scopus o servidores como SciELO, sitios donde a diario se registran cientos y miles de publicaciones de artículos de investigación, que son fundamentales para el campo de la sanidad y permiten a los profesionales sanitarios mantenerse actualizados en los avances médicos, tratamientos y síntomas, optimizando su tiempo de lectura y mejorando la calidad de la investigación y atención a los pacientes.

Puesto que hay demasiada información, resulta complicado gestionar toda esta información, teniendo en cuenta la inversión de tiempo de lectura, analizar y procesar semejantes cantidades de archivos en estos últimos años debido a que el número de artículos médicos científicos se han disparado, sobre todo a partir de principios del 2020.

Desde el principio de la pandemia del covid-19 en el 2020, la sobrecarga de información y la velocidad con la que se ha propagado sobre todo en estos últimos años ha tenido un ritmo vertiginoso. Gran parte debido a que pasamos más tiempo conectados tecnológicamente que en cualquier otra época. Dadas las circunstancias que hemos atravesado la gran población mundial en general, por los múltiples confinamientos y medidas de seguridad, han posibilitado el teletrabajo a causa del

covid. Seguimos en periodo de adaptación y eso implica utilizar con más frecuencia los ordenadores en esta nueva era tecnológica en la que nos encontramos.

Podemos encontrar mucha información científica relacionada con el covid con motores de búsqueda como PubMed, ScienceDirect, Google Schoolar, ResearchGate, LibraryGenesis o Sci-Hub entre otros, que actualmente son las herramientas más utilizadas para recuperar información además de las bases de datos y los directorios que están interconectados entre sí. Estas herramientas de búsqueda permiten al usuario encontrar información disponible de forma interactiva y una vez realizada la exploración se muestra una ordenada lista de documentos que cumplen los criterios marcados.

Durante este trabajo nos hemos centrado sobre todo en buscar información del virus Sars-Cov-2 con los motores de búsqueda ScienceDirect, PubMed, y en menor medida Google Schoolar para contrastar y extraer la información más relevante abordando el tema principal de nuestro proyecto de resumir artículos científicos relacionados con la sanidad, más concretamente con el covid.

A continuación, vamos a mostrar un ejemplo del registro de artículos científicos relacionados con el covid extraído de la plataforma de acceso PubMed que contiene la base de datos de MEDLINE [6], todo ello del ámbito médico.

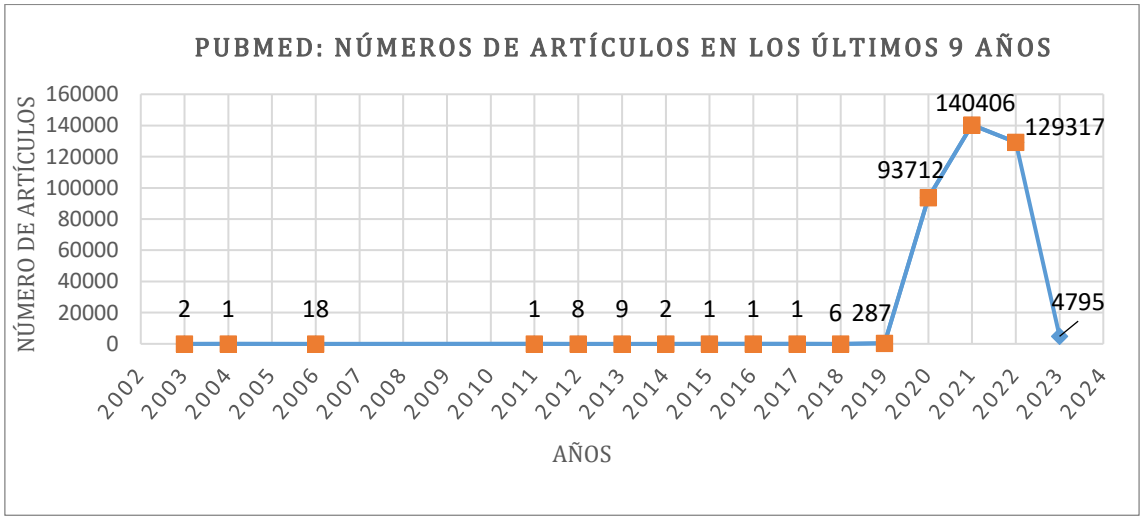


Ilustración 1. Gráfica de datos anuales recogidos hasta el mes de marzo del 2023 en el dataset de PubMed

Se puede apreciar un cambio brusco del número de artículos científicos asociados al covid del 2019 respecto a los años 2020, 2021, 2022 y 2023 por el surgimiento de un nuevo virus en el cual apenas se disponía de información y actualmente podemos encontrar bastantes informes, noticias, referencias y artículos que conciernen al covid.

1.2 Resolución del problema

El área de la ingeniería que recoge el proceso de creación de sistemas como el reconocimiento del habla, comprender y generar lenguaje, y sus aplicaciones para llevarlo a cabo es lo que conocemos como procesamiento del lenguaje natural, PLN. El papel que juega el PLN actualmente es muy relevante en el dominio biomédico por el análisis, extracción y estructuración de la información que contienen los repertorios de textos biomédicos, y en la sociedad española se observa en casi todos los sectores, incluyendo a la sanidad cada vez más, como puede ser un médico buscando en el navegador a través del buscador Google documentos o webs potencialmente relevantes en relación a un virus nuevo como es el Sars-cov-2 o también llamado covid-19 parar contrastar la información más importante con los síntomas que presenta el paciente mediante la consulta que le ha especificado.

Una de las tareas del PLN es la generación automática de resúmenes que surge con la intención de hacer frente a la sobrecarga de información, como una manera de seleccionar automáticamente aquella que realmente interesa al usuario y descartar la que es irrelevante. A partir de uno o varios documentos que conservan la información importante, de los originales, eliminando la información insignificante presentando una longitud significativamente inferior a los iniciales. Es una disciplina que cuenta con más de 50 años de investigación, pero hasta finales del siglo XX no se popularizó debido al auge experimentado por internet y al aumento exponencial de la información disponible, que hace imposible al usuario procesar toda la información en un tiempo razonable.

El interés por el desarrollo de sistemas de apoyo en el ámbito sanitario está teniendo un progresivo crecimiento, tanto empresas privadas como públicas invierten cantidades muy grandes de dinero para el aumento de herramientas como es la generación de resúmenes, algo así como el proyecto TextDigester de Francesco Ronzano en 2017 [https://github.com/fra82/textdigester] [7]. Nuestro TFG está enfocado en el ámbito biomédico y exploramos nuevas técnicas para mejorar en el acceso de la información biomédica a través de la incorporación de generación de resúmenes.

1.3 Impacto esperado

Estamos experimentando un cambio socioeconómico que en líneas de investigación tratan de mejorar la asistencia médica con progresos tecnológicos como la telemedicina o el Big Data, éste último avance está comprobando un fuerte impacto en cuanto al tratamiento de los datos en sanitarios y pacientes porque ofrece la posibilidad de gestionar cantidades enormes de información de forma eficiente y práctica, garantizando protección y confidencialidad. Por tanto, la creación de una herramienta competente para generar resúmenes bien estructurados es un apoyo en favor de la investigación y mejoras sanitarias.

Centrándonos en el ámbito biomédico, el control a través de estas herramientas innovadoras puede implicar un adelanto tecnológico para optimizar el tiempo en la toma de decisiones de los especialistas puesto que en el ámbito sanitario es de vital importancia, al igual que el tiempo que se emplee en ella. Por eso, la importancia de disponer métodos automáticos que permitan sintetizar el contenido informativo de mayor relevancia con el objetivo de producir un texto más corto para producir un resumen es de suma importancia.

La organización de textos biomédicos supone un gran avance para el crecimiento de estas tecnologías y sobre todo ayudaría a reducir y clasificar el volumen de datos con los que los expertos pudieran trabajar, contribuyendo a la rentabilidad de la medicina.

Con el desarrollo de este proyecto trataremos de ayudar sobre todo a investigadores y médicos a mostrar, recuperar e integrar la información biomédica. A través de la aplicación Appenecum los especialistas sanitarios podrán obtener de un extenso documento científico en formato PDF, otro más corto resumiendo las partes más importantes logrando con ello agilizar el tiempo de lectura y por consiguiente los tiempos de atención a los pacientes.

El impacto que puede producir es en mejorar el sistema sanitario, facilitando la labor de los profesionales y mejorando el servicio a los pacientes. La progresión de estas técnicas se puede trasladar a cualquier campo sanitario, desde el campo de la investigación, hasta la atención en el paciente.

2 OBJETIVOS

2.1 Principal

Este trabajo tiene como objetivo principal crear un resumidor de artículos científicos en el campo biomédico utilizando técnicas de PLN y nuestro conocimiento en esta área, basándonos en fuentes confiables y acreditadas de información.

2.2 Secundarios

Para lograr con éxito el objetivo principal, será necesario tener en cuenta los siguientes objetivos específicos:

- Analizar recursos léxicos y semánticos de plataformas biomédicas como ScienceDirect o PUBMED en el desarrollo de generación del resumen.

- Estudiar la aplicabilidad de técnicas extractivas del PLN y la generación automática de resúmenes extractivos mono documento para nuestro método.

- Proponer y desarrollar un método en el que su diseño e implementación extraiga conjuntos de oraciones para el sistema de generación automática de artículos científicos del campo biomédico, fundamentado según la importancia del documento.

- Determinar experimentalmente la forma efectiva de integrar los resúmenes automáticos en nuestra arquitectura del sistema mono documento, consiguiendo que el usuario obtenga una experiencia positiva.

- Evaluar y valorar la calidad de los resúmenes generados por nuestra herramienta Appenecum.

3 ESTADO DE LA CUESTIÓN

En esta sección, nos enfocaremos en definir y presentar las funcionalidades principales del PLN, desde la creación de resúmenes hasta su evaluación, y el uso de diversas herramientas y recursos para facilitar esta tarea. Estos son los conceptos fundamentales de investigación que están estrechamente relacionados con el objetivo de este trabajo de fin de grado.

3.1 ¿Qué es el procesamiento del lenguaje natural (PLN)?

Para comprender cómo se genera un resumidor de artículos primeramente debemos tener en cuenta que el resumidor de artículos dentro del ámbito biomédico forma parte de una de las tareas del PLN entre las muchas otras utilidades que tiene. Esta tecnología de producir resúmenes surge a comienzos de los años cincuenta a la par que el nacimiento de la Inteligencia Artificial (IA) por lo que están muy vinculadas siendo el PLN una disciplina dentro de la IA. A día de hoy, la tecnología está sufriendo una progresión vertiginosa en estos últimos años a causa de la magnitud de información disponible en la web, los avances tecnológicos en el área de algoritmia y la capacidad resolutiva de computación que existe actualmente, como los clusters.

La definición de PLN [8]: "Es una parte esencial de la Inteligencia Artificial que investiga y formula mecanismos computacionalmente efectivos que faciliten la interrelación hombre/maquina y permitan una comunicación mucho más fluida y menos rígida que los lenguajes formales". Específicamente, está enfocado en el desarrollo de las comunicaciones ser humano-máquina, separándolas en partes y reconociendo los componentes más importantes del mensaje.

El PLN se puede dividir en dos grandes áreas como son la comprensión y generación del lenguaje natural, para poder realizar el resto de las tareas pretendiendo conseguir que las máquinas puedan interpretar, entender y manipular el lenguaje humano.

En primer lugar, la comprensión del lenguaje natural (CLN o NLU) es el proceso de acción para resolver suposiciones realizadas a partir de unos datos como base, para comenzar el desarrollo de investigaciones, dada una entrada (un texto) se ha determinar cuál de sus posibles interpretaciones es la adecuada (Reiter y Dale, 1995). Aclarando la definición anterior, podemos decir que es la parte del PLN que se encarga de asimilar el mensaje escrito y comprender su contexto e intención, de manera parecida a como lo realizaría un humano. Para que el sistema de CLN funcione es necesario el uso de reglas gramaticales, la teoría del campo semántico y pragmático, un idioma específico para los conjuntos de datos, etc.

En segundo lugar, la generación del lenguaje natural (GLN o NLG) provee a la máquina la virtud de producir un nuevo mensaje en lenguaje humano de forma independiente. Recapitulando, la función de estos modelos es: seleccionar la

información a reproducir (dependiendo de la interpretación del mensaje a contestar), decidir cómo organizarla y cómo reproducirla (léxico y recursos gramaticales, morfología, estructuras sintácticas, etc.). Estos modelos generan frases nuevas palabra a palabra y tienen que ser entrenados para que funcionen correctamente (Power, 1995 y Lavoie, 1996).

A partir de aquí las aplicaciones del PLN pueden ser muchas como recuperación de información, traducción automática, reconocimiento y síntesis del habla, etc. Pero nos vamos a centrar en la clasificación de textos [9] y el resumen [10] que es el tema principal que vamos a abordar a lo largo de este TFG.

3.1.1 Tipos de textos

Antes de clasificar los tipos de textos definimos según (DRAE, 22ª edición) [11] qué es un texto. "Es un conjunto de enunciados con unidad y coherencia, que se transmiten de forma oral o escrita". Para la clasificación de textos existen muchas posturas que han seguido diversas pautas con más de dos milenios de reflexiones sobre los tipos de textos. Desde el primer modelo manifestado por Aristóteles en su Retórica para el análisis del discurso mediante textos, considerado pionero de la lingüística textual, hasta la actualidad que todavía no hay una aprobación unánime para la tipología textual. Sin entrar en demasiada profundidad en el estudio de los tipos textos, vamos a seguir la propuesta de Werlich Egon [12] siendo de las más divulgadas por ser precursor en cuanto a la división de tipos textuales básicos agrupados según su enfoque contextual. Propone una clasificación basada en las estructuras cognitivas, es decir combina lo que corresponde estrictamente al orden cognitivo ("modos de abordar la realidad") con el orden lingüístico ("modos de representar la realidad").

Según Werlich las bases textuales de los textos se pueden reducir en cinco modelos básicos:

1. Base descriptiva (textos descriptivos). Relacionado con la percepción de los hechos y variación en el espacio. Describen cómo es un objeto, una persona, un animal, un sentimiento o una situación. (Declaración de un testigo, folleto turístico, etc.).
2. Base narrativa (textos narrativos). Asociados con la percepción de los hechos y del tiempo. A grandes rasgos los textos narrativos son aquellos que informan de acontecimientos vividos. (Cuentos, informes, etc).
3. Base expositiva (textos expositivos). Transmiten información de manera objetiva, son referidos al análisis y síntesis de ideas y representaciones conceptuales (sintéticas o analíticas) como definiciones, ensayos, etc.
4. Base argumentativa (textos argumentativos). Expresan una posición o un juicio de valor. Están vinculados a las relaciones entre ideas y conceptos. En

los textos argumentativos el que habla normalmente manifiesta una opinión, debate un argumento o expresa ciertas dudas. (Comentario, tratado científico, etc.)

5. Base instructiva (textos instructivos). Indica acciones para el comportamiento del que habla. Están relacionados con las conductas futuras. Pretenden provocar un comportamiento determinado en la actitud del lector, algunas de sus funciones son aconsejar, advertir, proponer, etc. (Manual de instrucciones, leyes, normativas, etc).

Para este trabajo vamos a centrar nuestra atención en los textos expositivos o explicativos porque tienen como finalidad dar a conocer e informar de una serie de conceptos específicos para aportar conocimiento claro y directo sobre un tema. Los textos expositivos no reflejan la opinión del autor, tan sólo muestran un tema basándose en evidencias y fuentes respaldadas, teniendo como objetivo presentar e informar al lector, como es nuestro caso, artículos científicos relacionados con el covid.

3.2 ¿Qué es la generación automática de resúmenes?

Una vez mencionado los conceptos básicos que están relacionados con el tema principal de este TFG, es de gran interés entender la tarea conocida como generación automática de resúmenes, por el proceso de originar y evaluar un resumen. Durante la elaboración de este trabajo prestaremos nuestra atención en investigar y proponer métodos y técnicas para producir el resumen en sí, aunque otra opción de la que sólo vamos a mencionar es abordar la tarea de evaluación de un resumen una vez que ya está generado. Ambas son tareas complicadas, debido al alto contenido de subjetividad que tiene un resumen.

El resumen y la clasificación de textos se utilizan en el PLN para resumir extensiones largas de manera automática o extraer palabras claves para poder clasificarlos.

Para iniciar cómo generar resúmenes automáticamente es imprescindible saber qué se entiende por resumen y en qué consiste la generación automática de resúmenes. Según (DRAE, 22ª edición) [11] resumir es reducir a términos breves y precisos, o considerar tan solo y repetir, abreviadamente lo esencial de un asunto o tema. Esto implica, por un lado, que debemos saber lo más importante del tema tratado en cuestión y, por consiguiente, saber expresarlo de manera abreviada. Teniendo estas dos ideas claras, la generación de resúmenes no es más que producir un resumen de manera automática, sin que nosotros tengamos que preocuparnos de qué aspectos son los más importantes y cómo los queremos contar o presentar.

Los aspectos a tener en cuenta al generar un resumen de forma automática podemos dividirlo en:

- Fases del proceso [13]. Propone la división más popular, señalando tres etapas en la realización de la tarea: en la primera etapa la identificación del tópico identifica el núcleo del tema central del documento origen, es decir de qué trata el documento a través de la selección de las unidades (palabras, oraciones o párrafos) más importantes. Acto seguido en la etapa de interpretación, se lleva a cabo un análisis y comprensión del texto más profundos, mediante el uso de sofisticadas herramientas y recursos lingüísticos, obteniéndose una representación intermedia de la información. El resultado de la etapa anterior se traduce al texto final del resumen, en un formato y lenguaje adecuado para su lectura por parte del usuario contenida en la fuente. Por último, durante la etapa de generación produciremos el resumen final y si queremos utilizar palabras distintas de los documentos de entrada, tendremos que utilizar también técnicas de generación del lenguaje natural (GLN).

- Tipos de resúmenes [14]. Al tener dificultad para encontrar un solo tipo de resúmenes a causa de la multitud de ellos y dependiendo del resumen adquirido, es decir según a quién va dirigido según el medio podemos encontrar texto, audio, vídeo o hipertexto. Según el número de documentos que intervienen, es decir la entrada del documento que puede ser mono, si en su elaboración todo el material procede de una única fuente o multi documento, si proceden de distintos documentos que tratan sobre un mismo tema, o según cómo lo vamos a presentar, en este caso la salida del documento pudiendo ser como un titular, como un extracto o un abstracto. Según en qué idioma se encuentra el resumen, monolingüe, multilingüe, crosslingüe. Por último, según la finalidad del resumen puede ser genérico, personalizado, indicativo, informativo, de opción, de actualización y orientado a un tema. Vemos que existe una amplia variedad de tipos de resúmenes, de los cuales hemos citado los más importantes.

- Técnicas y enfoques [15]. Es importante conocer qué técnicas y enfoques podemos utilizar para poder generar un resumen de forma automática. Cabe destacar la distinción entre resúmenes que distingue los resúmenes por extracción de los por abstracción ya que es especialmente relevante cuando se quieren generar resúmenes de manera automática.

  La generación de resúmenes por extracción son los que se generan a base de extraer y concatenar oraciones completas del texto original.

  Un método automático de generación de resúmenes por extracción actuará de acuerdo con el siguiente proceso general. En primer lugar, se divide el texto en oraciones, a continuación, se puntúan individualmente las oraciones de acuerdo con algún criterio o criterios que midan la importancia relativa de las oraciones. En tercer lugar, se seleccionan las oraciones mejor

puntuadas (en un número que dependerá de la longitud que se desee que tenga el resumen). Finalmente, se ordenan las oraciones y se concatenan para componer el resumen final.

En cuanto a las técnicas y soluciones empleadas para generar resúmenes por extracción, cabe agruparlas en tres categorías.

- Los enfoques superficiales [15]. Puntúan las oraciones utilizando heurísticas sencillas que tratan de estimar la relevancia de las oraciones para la audiencia objetivo del resumen. Entre las heurísticas más utilizadas están la frecuencia de los términos del documento y la posición de las oraciones en el mismo. La heurística de la frecuencia de los términos en el documento se basa en el uso de técnicas estadísticas y de recuperación de información para estimar la importancia relativa de las palabras que aparecen en el documento. De esta manera, posteriormente, se seleccionan para el resumen aquellas oraciones que contengan las palabras más importantes. La heurística de la localización de las oraciones en el texto (posición de las oraciones) se basa en la evidencia empírica de que, en algunos tipos de documentos, la posición de una oración en el texto es indicativa de su relevancia. Así, por ejemplo, las noticias periodísticas se escriben siguiendo una estructura de pirámide invertida, según la cual los datos de mayor interés se incluyen en primer lugar y, a continuación, se desarrollan aspectos secundarios. Siguiendo esta idea, se asigna mayor puntuación a las oraciones que ocupan las primeras posiciones del documento.

- Técnicas discursivas [16]. Un poco más complejas que las anteriores, se basan en un análisis de la estructura del discurso y en la detección de distintos tipos de relaciones entre palabras. Dentro de estas técnicas, destaca por su difusión el enfoque de las cadenas léxicas. Este método consiste en recorrer el texto buscando enlaces entre oraciones, de tal modo que las oraciones que se encuentren enlazadas conforman una cadena léxica. Cada cadena se identifica con un tema dentro del documento. Dos oraciones se consideran enlazadas si entre sus palabras se establecen relaciones, pudiendo ser estas relaciones muy diversas (por ejemplo, sinonimia, homonimia o repetición). Finalmente, y en lugar de extraer oraciones para el resumen, se extraen cadenas completas. Para ello, se puntúan las cadenas de acuerdo a distintos criterios como su longitud o la frecuencia de sus palabras. La más popular es la llamada teoría de la estructura retórica formulada por Mann y Thompson [17].

- Las técnicas basadas en grafos [18]. Representan el texto como una red compleja en la que los nodos distinguidos cada una de las unidades textuales en las que se divide el texto y las aristas representan algún tipo de relación entre estas unidades, generalmente de naturaleza léxica, sintáctica o incluso semántica. La idea subyacente en este tipo de enfoques es la emergencia en la red de grupos de unidades que guardan estrecha relación entre sí y que determinan la información relevante del documento. Para ello, se aplica un algoritmo de clustering encaminado a localizar los nodos centrales del grafo en función del número y peso de las aristas que confluyen en los distintos nodos. Finalmente, se seleccionan para el resumen las oraciones más centrales del documento.

Generación de resúmenes abstractivos [19]. Implican la realización de un proceso de abstracción, comprensión y reescritura. Generan un resumen a partir de cero, utilizando palabras y frases nuevas que no están presentes en el texto original. Sin embargo, su generación automática es más compleja y únicamente aplicable a dominios muy concretos. Es un área de investigación activa, y en constante evolución por el surgimiento de nuevas técnicas y enfoques. Un método automático de generación de resúmenes por abstracción actuará de acuerdo con el siguiente proceso general:

En primer lugar, se divide el texto en oraciones y a continuación se construye una representación semántica de estas oraciones. Se realizan operaciones de selección del contenido relevante, agregación y generalización de dicho contenido. Finalmente, se traduce la representación resultante a lenguaje natural mediante el uso de técnicas de generación de lenguaje [20].

En cuanto a las técnicas y soluciones empleadas para generar resúmenes por abstracción, pueden ser las técnicas de deep learning que son las más empleadas en la actualidad y han ganado popularidad en los últimos años debido a los avances en el campo del PLN como son:

- Modelos basados en reglas [21]. Estos enfoques utilizan reglas lingüísticas y heurísticas predefinidas para generar resúmenes abstractivos. Las reglas pueden incluir la selección de oraciones, la reformulación de frases, la sustitución de palabras clave y la estructuración del resumen. Los modelos basados en reglas requieren un conocimiento lingüístico profundo y una definición precisa de las reglas.

- Modelos de lenguaje estadísticos [22]. Se basan en modelos estadísticos que aprenden a partir de grandes cantidades de datos de entrenamiento. Utilizan técnicas como la generación de secuencias o modelos de Markov ocultos para generar resúmenes abstractivos. Estos modelos pueden aprender la probabilidad de generar una secuencia de palabras coherente y contextualmente relevante como resumen. Algunos ejemplos de técnicas estadísticas utilizadas en la generación de resúmenes son el modelo de

lenguaje n-gram, el modelo de Markov oculto (HMM) y el modelo de Markov de saltos múltiples (MMSJ).

- Los modelos de lenguaje basados en redes neuronales [23]. Pueden aprender representaciones de palabras y frases contextualizadas, lo que les permite capturar la semántica y la coherencia del texto. Han demostrado ser efectivos para generar resúmenes coherentes y de calidad, como GPT (Genereative Pre-trained Transformer) o BERT (Bidirectional Encoder Representations from Transformers). Se entrenan utilizando pares de texto y resumen, y aprenden a generar resúmenes coherentes y de calidad. Sin embargo, requieren grandes cantidades de datos, de entrenamiento y recursos computacionales significativos para entrenar y ejecutar los modelos.

La evaluación es el proceso para determinar la calidad del resumen generado, dividiéndose en:

- Tipos de evaluación por Mani (2001) [24] pueden ser:
  - Intrínseca: el resumen se evalúa de forma individual, en base a la información que contiene o ciertas características que pueden tener.
  - Extrínseca: el resumen es evaluado aplicándolo a una tarea concreta, por lo que es lo mismo en el contexto de otra tarea (recuperación de información, búsqueda de respuestas, clasificación de textos, etc). De esta manera, podemos comprobar si el resumen es útil para mejorar el rendimiento de la tarea en cuestión.

- Criterios de calidad y formas de evaluar [15]:
  - Criterios de calidad: el contenido informativo que comprobaremos si el resumen preserva el contenido relevante del documento original. La legibilidad que se ha de evaluar su forma, es decir su calidad gramatical.
  - Formas de evaluar: se pueden realizar de forma manual llevada a cabo por humanos que es más exacta y fiable, pero sin embargo es muy costosa y normalmente inviable debido al volumen de resúmenes. También puede evaluarse de forma automática, es decir de manera aproximada al contenido informativo de los resúmenes, pero no su legibilidad.

3.3 Análisis de herramientas que utilizan técnicas de PLN

A continuación, vamos a analizar individualmente varias herramientas que no todas forman parte del campo de la biomedicina, pero que son utilizadas en este ámbito como instrumentos de apoyo a la investigación, en la toma de decisiones y en general para facilitar los tiempos de lectura.

La tabla 1 presenta una recopilación de diferentes softwares comparando las características más relevantes y que en cierta manera constituyen buena parte del dominio biomédico, asemejándose a lo que va a consistir nuestro software que explicaremos con más detalle posteriormente.

Nombre	**Descripción**	**Idioma demostrativo**	**Tipo de enfoque**	**Información adicional**
AbreMES-X: **(Ander Intxaurrondo, 2019)** https://github.com/PlanTL-SANIDAD/AbreMES-X	Extractor de abreviaturas médicas españolas que se emplea para generar la base de datos de abreviaturas médicas españolas	Monolingüe, sólo en español	Utiliza un algoritmo por extracción ya que une acrónimos con oraciones completas de los textos originales (multigénero) y emplea un *enfoque superficial* donde realiza una estimación según la dependencia de veces que la información se repita y sea relevante.	La base de datos se genera detectando abreviaturas y sus posibles definiciones explícitamente mencionadas en una misma frase, extraídas de los metadatos de diferentes publicaciones biomédicas escritas en español que contienen los títulos y resúmenes. El algoritmo alcanza un 96% de precisión y un 82% de recall en una colección de pruebas estándar y no requiere de datos de entrenamiento.

TextDigester: **(Francesco Ronzano, 2017)** https://github.com/fra82/textdigester	Genera resúmenes de texto libre a gran escala de manera automática. La aplicación analiza textos que pueden provenir de páginas HTML, documentos XML u objetos JSON que son los más utilizados en las páginas web o redes sociales.	En diferentes idiomas (multilingüe).	De carácter extractivo que puede analizar textos planos, contenidos textuales de páginas web, objetos JSON y ejecuta análisis léxico-semántico de los documentos. Utiliza técnicas basadas en grafos por el método de resumen que calcula el centroide mediante sus vectores y clasifica las oraciones respecto a su similitud.	Integra distintos recursos de PLN, es un generador de resúmenes automático ubicuo, e implementa varios métodos para la generación de resúmenes.
Diccionario ConTexto: **(Mikel Artetxe, 2017)** https://github.com/artetxem/contexto	Alternativa libre que permite la creación automática de diccionarios contextuales ofreciendo ejemplos de uso reales extraídos automáticamente	En diferentes idiomas, pero limitado (multilingüe)	Este proyecto permite buscar la traducción de palabras y expresiones sobre grandes corpus bilingües, ofreciendo así ejemplos de su uso real en contexto. Tiene un enfoque basado en grafos por la red compleja de nodos que fracciona el contenido y precisa la información relevante del documento.	Es un diccionario contextual de código abierto. Incluye herramientas para construir automáticamente dichos modelos de diccionario a partir de corpus paralelos, una biblioteca central de Java para usarlos.
Spaccc_Post-Tagger: **(Felipe Soares, 2019)** https://github.com/PlanTL-SANIDAD/SPACCC_POS-TAGGER	Es un repositorio etiquetador morfosintáctico para corpus de dominio médico en español basado en FreeLing	Monolingüe, sólo en español.	Herramienta extractiva que tiene un enfoque discursivo, el cual nos permite analizar la sintaxis de la oración y detectar distintos	Obtiene un rendimiento de actuación de: 98,85% de segmentado, 99,47% de Tokenización

			tipos de elementos que componen las palabras (forma, lema, etiqueta y porcentajes).	y un 99,87% por parte del discurso.
SciELO-Spain-Crawler: **(Ander Intxaurrondo, 2018)** https://github.com/PlanTL-SANIDAD/SciELO-Spain-Crawler	Es un software creado para descargar todas las publicaciones escritas en español desde el servidor SciELO	Monolingüe, sólo en español	Enfoque basado en grafos que utiliza un algoritmo para encontrar los nodos centrales del grafo que convergen en los distintos nodos para escoger del resumen las oraciones más centrales del documento.	Esta biblioteca virtual electrónica abarca una colección de revistas científicas de salud españolas que se seleccionan siguiendo criterios de calidad preestablecidos
Changefreq: (Reeve, Han y Brooks, 2007): The use of domain-specific concepts in biomedical text summarization. Information Processing and Management 43, 1765-1776. 2007. https://www.redalyc.org/pdf/5157/515751742007.pdf	Se usa en el dominio biomédico para identificar las sentencias del texto. Utiliza relaciones semánticas entre conceptos vecinos en texto, centrándose en el uso de las distribuciones de frecuencia, formando un resumen similar al original.	En diferentes idiomas (multilingüe)	Híbrido entre dos enfoques. Por un lado, está basado en enfoques superficiales, y por otro lado, emplea técnicas discursivas para extraer los conceptos específicos del dominio biomédico e identificar las decisiones del enunciado original.	Se pueden usar conceptos en vez de términos, para que den soporte a la identificación de los conceptos y son capaces de determinar relaciones semánticas.

Tabla 1. Comparativa de diferentes herramientas a tener en cuenta en el ámbito sanitario

4 CUERPO DEL TRABAJO

Antes de profundizar en el desarrollo y diseño de nuestra herramienta llamada "Appenecum", es importante destacar la fundamentación proporcionada por un especialista médico, que nos ha llevado a considerar como punto de partida la creación de un método de generación automática de resúmenes. Desde su perspectiva, hemos tomado la decisión de iniciar este proyecto con el objetivo de sintetizar un artículo científico del campo de la biomedicina seleccionado de la plataforma ScienceDirect. Nuestro método generará un resumen con una estructura similar al artículo científico seleccionado en formato PDF, y esta iniciativa nos permitirá avanzar en la creación de nuestra herramienta.

4.1 Pensamiento en la forma de proceder a resumir de un médico

En general los artículos científicos son la mayoría expositivos como hemos mencionado anteriormente en la tipología textual porque transmiten conocimientos sobre un tema en particular y abordan una investigación con una metodología bien definida, finalmente se comentan los resultados obtenidos y se transmiten a la comunidad científica para avanzar en el conocimiento. Los textos expositivos están fundamentados por Adam, J. M. ,1992 [25]. Se componen principalmente por la introducción donde el autor expone el tema y la idea a explicar brevemente en qué va consistir el tema para despertar el interés del lector, el desarrollo del texto o cuerpo central dividido en párrafos los cuales el escritor manifiesta las ideas principales del tema que sirven para confirmar o reafirmar la idea expuesta, y suelen ir acompañados de ejemplos para mejorar su comprensión, y la conclusión que presenta un breve resumen de los puntos tratados en el cuerpo del texto mostrando una valoración de la introducción o sencillamente una sinopsis de los argumentos. También tienen como objetivo informar al receptor de forma clara mediante una explicación sobre algún tema específico donde destaca el conocimiento intelectual sobre el tema, el rigor, la exactitud, la claridad y el orden.

Después de mantener una conversación con un médico especialista sobre su interpretación y la forma de proceder cuando terminan de leer un artículo científico, utiliza un razonamiento de manera estructurada y organizada. La reflexión inicial por el experto sanitario es extraer del artículo científico las ideas principales y secundarias, centrándose en las primeras para dividirlas en fases, es decir primero abrevia de todos los párrafos construyendo un resumen de cada párrafo para recapitular la información que considera más relevante, en segundo lugar extrae de cada resumen de cada párrafo la idea principal y por último, renombra el título del artículo por otro más escueto que simplifique el nuevo resumen originado. Otra cuestión a destacar es la forma o métodos de extraer las ideas principales de cada párrafo que llegados a este punto pueden ser de varias formas a la que el doctor en particular reafirma en fijarse en las repeticiones o sinónimos de la idea o alrededor

de ésta para confirmar el núcleo de la idea en cuestión, es decir emplea una forma léxico-semántica. Por otra parte, también declara que utiliza la extracción de una frase principal y necesaria para quitarle el sentido del contexto, por lo que también esta otra forma semántica de identificar el tópico o idea principal del texto.

Para un mejor entendimiento, nuestra interpretación esquemática recogiendo toda la información proporcionada por el médico especialista en el proceso de generación de un resumen de manera manual ha sido plasmado en la siguiente ilustración.

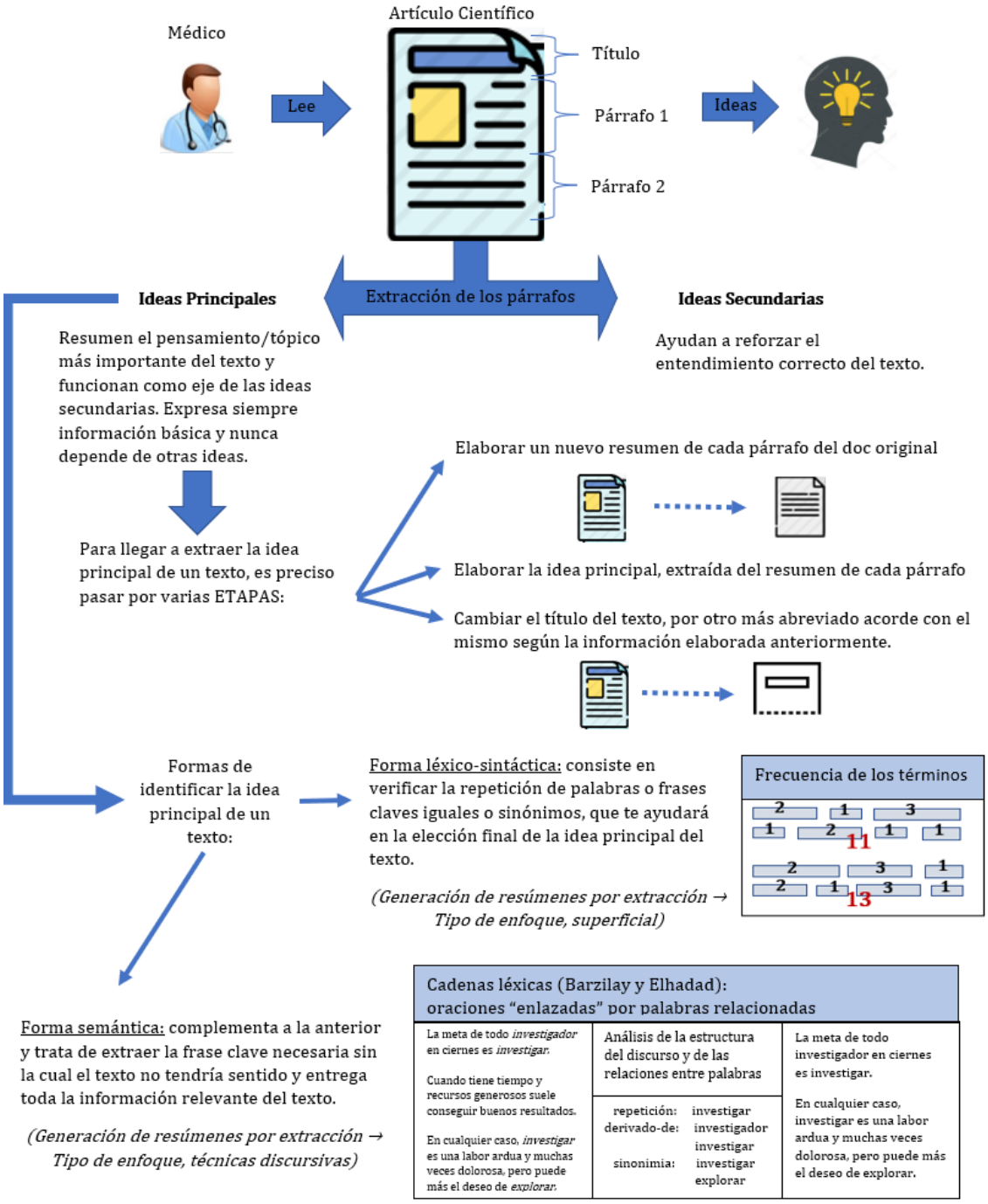


Ilustración 2. Extracción esquemática de un artículo científico como resumen por un médico realizado manualmente

4.2 Diseño y desarrollo de la herramienta Appenecum

En esta sección nos basaremos en la explicación del doctor especialista en medicina como referencia inicial a la hora de diseñar y desarrollar nuestra herramienta Appenecum. El método contiene una serie de características que están más enfocadas a las técnicas superficiales y que han sido implementadas inicialmente por bloques secuenciales diferentes con distintas tareas cada uno, hasta su versión definitiva mediante funciones en un solo notebook.

4.2.1 Diseño del método

La herramienta que hemos desarrollado, llamada Appenecum, es capaz de identificar, analizar e interpretar el tema de un texto con el objetivo de generar un resumen utilizando el método extractivo. Durante la implementación de nuestro método, hemos contado con el apoyo de la herramienta TextDigester. En momentos complicados en los que el progreso en código se nos complicaba, TextDigester nos ha servido como referencia para ciertas funciones facilitando el avance para nuestro método, como por ejemplo en la función *resumen_palabras*. Nuestra herramienta tiene como propósito brindar información a los lectores, en particular al personal sanitario, especialmente a los médicos, con el fin de optimizar su tiempo de lectura y búsqueda de información relevante.

El medio por el que se desenvuelve nuestra herramienta es a través de textos. Considero que como en todas las páginas webs y bases de datos analizadas tienen archivos PDF para poder descargarlas y dado que los usuarios van a ser médicos, podrán descargar dichos archivos en PDF por lo que vamos a transformarlos en TXT mediante herramientas y librerías para manipular los artículos científicos introducidos y así poder generar el resumen más simple.

Otro aspecto a tener en cuenta es el idioma con el que vamos a desarrollar nuestra herramienta, que en nuestro caso es monolingüe debido a que la entrada y la salida tiene que ser de un mismo idioma permitiendo esto mismo en un sólo idioma, concretamente está creada para el castellano. Si la entrada de un artículo es en español, la salida de dicho artículo es en español.

La entrada del sistema de resúmenes del ámbito biomédico relacionados con el covid va a ser monodocumento, prestaremos nuestra atención en sólo textos científicos relacionados con el covid. Hemos precisado manejar la reiteración de información de varios artículos relacionados con el covid por separado para minimizar considerablemente el tiempo de lectura, sobre todo si el usuario pretende obtener un volumen alto de información en el menor tiempo posible. El formato de archivo que vamos a emplear de entrada es en PDF que mediante una serie de

librerías modificaremos a TXT para poder trabajar con texto plano que es mucho más sencillo y cómodo, una vez generado el resumen volveremos a pasar en PDF para la posible descarga del archivo.

Nuestro método es sobre todo de carácter extractivo, es decir, extraemos de un artículo original recapitulando las oraciones más representativas del documento. Para generar dicho resumen necesitamos determinar la importancia de las oraciones y para ello establecer un criterio de selección que describiremos en el desarrollo del método, en el punto de a continuación 4.2.2 Implementación del método.

La salida de nuestros resúmenes constituye entre un 20 y un 30 por ciento aproximadamente de los extractos de los artículos originales que se quieren resumir. En el punto 5.3 resultados de la evaluación mediante la métrica Rouge podemos visualizar con más detalle las características de medición para nuestra herramienta. Con el porcentaje de compresión del resumen podemos calcular el porcentaje extractivo mediante la siguiente relación:

$$\frac{nº\ de\ palabras\ contenidas\ en\ el\ abstract}{nº\ de\ palabas\ totales\ del\ documento\ original} \cdot 100 = nº\ de\ porcentaje\ extractivo\ del\ resumen$$

Recapitulando la información anterior y para un mejor entendimiento visual mostramos la siguiente tabla resumen con las características principales de nuestra herramienta Appenecum:

Nombre	Descripción	Medio	Idioma	Entrada	Tipo de enfoque	Salida	Finalidad	Rendimiento
Appenecum	Herramienta capaz de identificar el tópico, analizarlo e interpretarlo para posteriormente generar el resumen	Texto	Monolingüe (español)	Monodocumento	Superficial	Extractos	Informar	Mediante la herramienta Rouge medimos los porcentajes de eficacia: entre un 20% y un 30% aproximadamente

Tabla 2. Resumen de las características principales de nuestra herramienta

4.2.2 Implementación del método

El lenguaje de programación escogido y utilizado para nuestro método es Python [26] por las siguientes características que nos resulta más útiles y adaptables para la elaboración de nuestra herramienta [27]:

- La sintaxis que utiliza nos resulta más sencilla al tratarse de pseudocódigo, es decir con poco código permite mucho.
- Su utilidad no está ligado a un sector determinado, sino que vale para todo.
- Es uno de los lenguajes que tiene mayor cantidad de librerías y admite muchos recursos construidos por la enorme comunidad de programadores.
- Es un lenguaje multiplataforma, es decir el mismo código escrito funciona para cualquier sistema operativo (macOS, Linux, Windows, etc).
- Es un lenguaje interpretado, no compilado, esto quiere decir que el código no se ejecuta estrictamente instrucción por instrucción, sino que pasa por un proceso de compilación. Además, utiliza tipado dinámico, esto quiere decir que una variable puede adoptar distintos tipos de valores.

El entorno de trabajo para desarrollar nuestra herramienta Appenecum utilizado con Python es Jupyter Notebook [28] por la interfaz sencilla, la simpleza a la hora de ejecutar y modificar los bloques de código en local sin necesidad de compilar todo el código completo y la adaptación e integración que tiene con el lenguaje Python. Es una aplicación cliente-servidor de código abierto que permite combinar elementos como texto, video, audio e imágenes. Al ejecutarse el código mediante el navegador pudiendo admitir diversos lenguajes, la comunicación se realiza a través de *Kernels* (núcleos) [29] de cálculo, asimismo cuando ejecutamos Jupyter Notebook en nuestro ordenador local aparecen como primera página *Jupyter Dashboard,* en la cual se pueden visualizar los archivos disponibles donde hayamos ejecutado nuestro programa. También nos decantamos por esta interfaz web por el llamado "cuaderno computacional" que está organizado por celdas y cada una de las celdas pueden contener varias formas, especialmente de tipo *code* (código) o de *markdown* (notas o comentarios), con esto conseguimos tener bien estructurado y separado por bloques todo nuestro código.

La herramienta Appenecum en un principio ha sido implementada por diferentes celdas o bloques, llamados notebooks que nos permiten trabajar de forma similar a los cuadernos de laboratorio tradicionales y nos ofrecen la posibilidad de llevar un seguimiento estructurado de nuestro trabajo ya sea con código Python, tablas, anotaciones o comandos bash (conjunto de parámetros para la configuración y administración del sistema). Posteriormente se unificó todos los bloques del código en un mismo notebook mediante funciones, sufriendo unas pequeñas modificaciones y adaptaciones hasta conformar el código definitivo como resultado final.

Para la implementación del método antes que nada hemos realizado una arquitectura para demostrar en qué consiste nuestra herramienta Appenecum, donde mostraremos de forma general el sistema de generación monodocumento.

- Extracción de artículos científicos en la plataforma ScienceDirect: descargamos varios documentos en PDF de los cuales escogeremos uno de ellos como entrada una vez preparada, es decir habiéndolo pasado por una serie de filtros dejándola listo y transformando el artículo científico a TXT en un mismo paso para así poder manipularlo.

- Selección de pasajes: apoyándonos en la explicación del doctor especialista en medicina del punto anterior 4.1 pensamiento en la forma de proceder un médico, analizaremos de forma superficial los pasajes más importantes, usando las heurísticas más utilizadas como la frecuencia de los términos médicos del documento y la posición de las oraciones en el mismo por secciones.

- Representación de los pasajes: estudio minucioso para la interpretación y comprensión de los pasajes seleccionados, y la muestra de los mismos.

- Obtención de similaridad y diferencias: detectamos los pasajes comunes y de los distintos artículos.

- Eliminación de redundancias: mediante una serie de librerías y algoritmos creados por nosotros reducimos el texto de repeticiones, palabras no relevantes, saltos de línea, etc.

- Generación de resumen: detección de información relevante con técnicas de PLN.

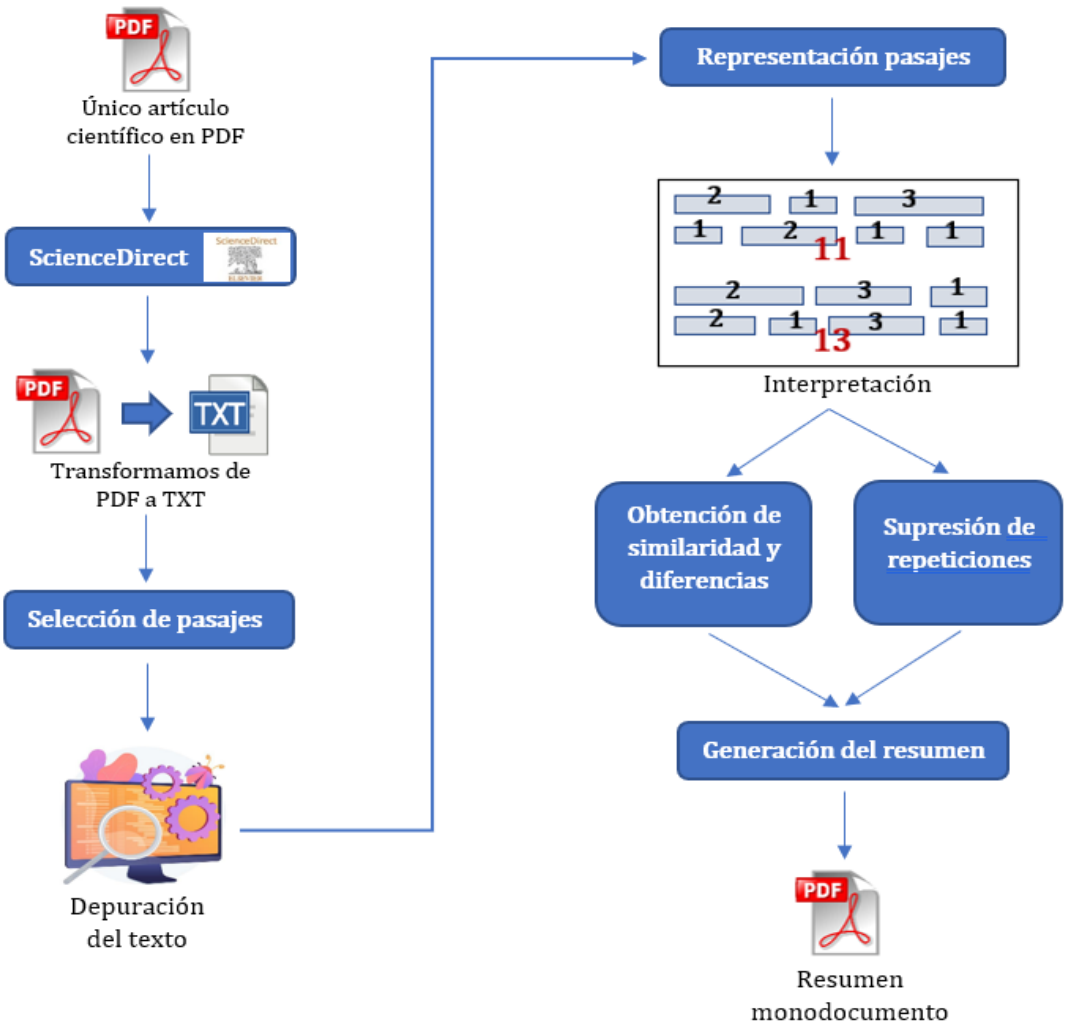


Ilustración 3. Representación arquitectónica de la herramienta Appenecum

Esta representación arquitectónica está basada en 5 bloques ordenados numéricamente destacando a grandes rasgos la forma de proceder que ha conformado nuestro método.

- Primero: transformar un único artículo científico en PDF a TXT dejándolo lo más adecuado posible para poder trabajar mejor.
- Segundo: separar las oraciones compuestas del texto original en sólo palabras confeccionando nuestro método para visualizar las palabras que más se repitan y seleccionarlas para puntuarlas.
- Tercero: obtener un diccionario que contenga frases puntuadas manteniendo el orden textual original.
- Cuarto: originar el resumen definitivo ordenando nuestro diccionario de puntuación de frases.
- Quinto: según los parámetros recibidos imprimimos el resumen y transformamos esa salida en PDF generando un nuevo archivo en este mismo formato.

Una vez conocidos el lenguaje de programación empleado para nuestro método, como es Python, el entorno de desarrollo con el que hemos trabajado durante todo este trabajo es Jupyther Notebook y el método representativo como arquitectura de nuestra herramienta Appenecum, ahora podemos abordar la descripción del resultado definitivo en código. En este punto explicaremos con mayor detalle cada una de las funciones y sus finalidades, así como también ilustraremos los fragmentos de todo el código para su mejor entendimiento.

Empezamos con las descripciones de las librerías que hemos tenido que importar por la necesaria utilidad en nuestra herramienta:

Librerías	Descripción	Acceso	Referencia
Fitz	Permite transformar los artículos relacionados con el covid en formato pdf para convertirlos a txt. Se adapta bien accediendo a archivos con extensiones pdf, xps, oxps, cbz, fb2 o epub y formatos de imagen como png,jpg, etc. En los documentos PDF, particularmente tiene muchas utilidades como la optimización del texto, la creación de subdocumentos e incluso extracción de imágenes entre otras.	Gratuita y de código abierto	Forma parte del módulo PyMuPDF. https://pymupdf.readthedocs.io/en/latest/module.html
Spacy	Utilizada para el PLN en Python. Diseñada generalmente para la utilización en producción y apoyo en la creación de aplicaciones que procesan y contienen grandes volúmenes de texto. Es muy útil para la segmentación de oraciones o clasificaciones de texto siendo muy fácil su implementación.	Gratuita y de código abierto.	Tiene soporte para más de 66 idiomas incluido el español. https://spacy.io/
Datetime	Suele utilizarse para la presentación de fechas de diversas formas, en nuestro caso la aplicamos para añadir al resumen la fecha actual.	Viene incorporada en Python por lo que es gratuita y de código abierto	https://docs.python.org/es/3/library/datetime.html
ReportLab	Sirve para la creación de PDF desde Python. Es una librería muy extensa con muchas funcionalidades para incluir en el PDF que queremos obtener. En nuestro método hemos escogido varios módulos de esta librería para conseguir márgenes centrados del texto, tipo de letra o el nombre del archivo en PDF, entre otras opciones.	Gratuita y de código abierto.	https://recursospython.com/guias-y-manuales/crear-documentos-pdf-en-python-con-reportlab/

Tabla 3. Librerías utilizadas para la herramienta Appenecum

```
#IMPORTACIONES
import fitz
import spacy
from spacy.lang.es.stop_words import STOP_WORDS

from datetime import date

from reportlab.platypus import SimpleDocTemplate, Paragraph
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.lib.units import inch
from reportlab.lib.pagesizes import letter
```

Ilustración 4. Librerías importadas para la herramienta Appenecum

A continuación, en la siguiente fase hemos utilizado parámetros para definir nuestras funciones que nos permiten recibir valores cuando las llamamos. Lo más destacado de esta fase es la intención de recopilar un diccionario de terminología especializada del dominio biomédico.

Los parámetros a tener en cuenta y utilizados en nuestro código son los siguientes que vamos a mostrar en la ilustración 5. Asignamos el valor añadido para las palabras de la lista de palabras médicas con extrapunto_med = 0,01 este valor es por defecto según nuestro criterio creando nuestra lista propia de palabras médicas relacionadas con el covid-19, debido a que no encontramos ninguna librería que se adapte correctamente para lo que buscamos, con el nombre de palabras_med_extrapunto. El algoritmo implementado hace uso de la lista con palabras médicas buscadas de forma manual en diccionarios médicos y, a continuación, se mostrarán algunas de las palabras que se han considerado más relevantes para realzar el filtrado de los documentos:

Análisis	Virus	Covid	Sars
Epi	Virólogo	OMS	Sars-cov-2
Pandémico	Epidemia	Pandemia	Enfermedad
Infectar	Cuarentena	Mascarilla	Cepa

Tabla 4. Listado de algunas palabras médicas implementadas en herramienta Appenecum

A diferencia de palabras_extrapunto que es una lista de palabras que emplea el usuario donde le añadimos el mismo valor como extrapunto = 0,01 por flexibilidad en el código ya que son parámetros que se pueden cambiar manualmente, recibiendo ese valor en forma de porcentaje bajo nuestro criterio.

Observamos que en los extractos de un resumen original suelen oscilar entre un 20 y 30% prestando atención en otros trabajos como Compendium [30]. No obstante, realizamos una prueba con la intención de considerar cuanto es la longitud que tiene uno de los corpus de Kaggle [31] siendo una de las plataformas de las mayores comunidades de datos científicos del mundo donde ofrecen herramientas y recursos a cualquier usuario. De esta forma podemos conocer cuántas palabras contienen los abstracts de los documentos del dataset. En nuestra prueba seleccionamos 4

documentos diferentes y lo calculamos mediante la fórmula vista en el punto 4.2.1 Diseños del método a modo de ejemplo.

$$\frac{nº\ de\ palabras\ contenidas\ en\ el\ abstract}{nº\ de\ palabas\ totales\ del\ documento\ original} \cdot 100 = nº\ de\ porcentaje\ extractivo\ del\ resumen$$

En nuestro método utilizamos la variable *resumen_value* para obtener un 25% del texto original, donde hemos agregado restricciones de 100 caracteres como mínimo hasta 700 para considerar que es una frase. Añadimos *palabras_excluidas* para descartar las frases que contengan palabras http como pueden ser *keywords,* y también podemos eliminar palabras del texto como marcas de agua que aparezcan o pie de páginas con la variable *palabras_eliminar:*

Para la preparación en código del método, el uso de contenedores ha resultado ser de gran ayuda. El concepto de contenedor tiene una analogía con los contenedores físicos. De acuerdo con la librería de Python [32], el concepto de un contenedor es sencillamente cualquier objeto que contiene un numero arbitrario de otros elementos.

Los contenedores globales usados durante todo nuestro método son los siguientes:

- Sort_resumen: es una lista de palabras con el valor de apariencia ordenada de compilación descendiente.
- dict_puntuación: es un diccionario donde key:value, es decir la clave es valor y palabra es la puntuación obtenida.
- frases: es una lista donde se almacenan frases.
- dict_puntuacion_frases: es un diccionario donde key:value, es decir la clave es valor y palabra es la puntuación obtenida.
- sort_dict_puntuacion_frases: es una lista que representa el diccionario dict_puntuacion_frases ordenado de forma descendiente.

```
#PARAMETROS
extrapunto_med = 0.01
palabras_med_extrapunto = ["afectar", "análisis", "asepsia", "asintomático",
                           "contagiar","contagio", "cuarentena", "cuidar", "
                           "empírico", "endemia","endémico", "enfermedad", "
                           "fómite", "hipocondría", "hisopo", "infectar", "i
                           "mascarilla", "medicar", "médico", "morbilidad",
                           "prevalencia", "remitir", "resiliencia", "salud",
                           "apanicar", "covid", "cuarentenar", "cubrebocas",
                           "encuarentar", "EPI", "infodemia", "nasobuco", "c
                           "tamizaje", "virucida", "Coronavirus", "SARS-CoV-
                           "PCR", "reactivo", "esputo", "OMS", "CDC"]

extrapunto = 0.01
palabras_extrapunto = ["Keywords"]

resumen_value = 0.25 # 0.25 = 25% de texto original

frase_min_len = 100
frase_max_len = 700
palabras_excluidas =  ["http"]
palabras_elinimar = ["Journal Pre-proof"]

#CONTENEDORES GLOBALES
sort_resumen = []
dict_puntuacion = {}
frases = []
dict_puntuacion_frases = {}
sort_dict_puntuacion_frases = []
```

Ilustración 5. Parámetros y contenedores globales de la herramienta Appenecum

Primer bloque:

La función inicial situada en la ilustración 6, está diseñada para cualquier fichero que introduzca el usuario, una vez recibido en nuestro método lo transformamos de PDF a TXT mediante la librería **Fitz** está basado en el módulo PyMuPDF, el cual ya viene incorporado en el mismo módulo y nos permite leer, modificar y crear documentos en PDF de manera relativamente sencilla. Se ajusta bastante bien en la conversión a TXT, incluso se comporta de manera correcta con las imágenes y tablas transformando en texto muchos de los elementos que compone el artículo.

Si el usuario, no tiene ficheros en PDF, no importa porque gracias a las sentencias *try* y *except* el código no termina con error. Lo que recibe esta función son ficheros en forma de lista que contiene los nombres de los ficheros, y no devuelve nada, pero genera un fichero TXT para cada PDF recibido.

El funcionamiento interno de esta función es el siguiente, primero recorremos la lista de ficheros, probamos ejecutar el código mirando si sale algún error con *try* que dentro de este bloque abrimos el documento en PDF y el documento TXT que es donde vamos a guardar el contenido del PDF. Recorremos el contenido del documento PDF página por página decodificando el texto de cada página, y del texto obtenido escribimos dentro del fichero TXT cerrando el mismo documento en formato TXT. Mientras que con el bloque *except* si encontramos un error pasamos a dar otra vuelta del bucle con continue.

```
44 def pdf_to_txt(ficheros):
45     for fichero in ficheros:
46         try:
47             documento = fitz.open(fichero + ".pdf")
48             salida = open(fichero+".txt","wb")
49             for pagina in documento:
50                 texto = pagina.get_text().encode("utf")
51                 salida.write(texto)
52             salida.close()
53         except:
54             continue
```

Ilustración 6. Función de conversión de PDF a TXT de la herramienta Appenecum

Segundo bloque:

Con la función de la ilustración 7 de a continuación obtenemos el texto de todos los documentos recibidos. Recibe los ficheros, que es una lista con los nombres de los ficheros y devuelve *text*, una variable de texto que contiene el cuerpo de todos los ficheros recibidos por la función.

El funcionamiento interno de esta función es que pasamos como parámetro de la función los ficheros, declaramos una variable de texto y recorremos la lista de ficheros por lo que abrimos el documento TXT desde el cual sacamos el contenido, obtenemos el contenido y lo concatenamos a la variable *text* cerrando el documento

en TXT. Por último, devolvemos la variable *text* porque de esta manera podemos guardar su valor en otra variable y puede ser utilizada en otras funciones.

```
53 def get_text(ficheros):
54     text = ""
55     for fichero in ficheros:
56         file = open(fichero + ".txt", "r",encoding="utf8")#ISO-8859-1
57         text += file.read()
58         file.close()
59     return text
```

Ilustración 7. Función de obtención de textos de documentos recibidos de herramienta Appenecum

La siguiente función de la ilustración 8 tiene como objetivo principal recoger el texto obtenido de todos los ficheros y eliminar las palabras que el usuario ha indicado en la lista de *palabras_eliminar*. Lo que recibe es la variable *text* que contiene el resultado de la función *get_text,* es decir el texto de los ficheros y lo que devuelve es *text,* pero depurado.

El funcionamiento interno de esta función es que recorremos la lista de *palabras_eliminar* y dentro de este bucle, iteramos con otro bucle *while* para que a la vez se ejecuta el primer bucle *for* con el de *while* también vaya corriendo mientras que la palabra está en el texto, entonces cuando detectamos que la palabra está en el texto la eliminamos reemplazándolo por un campo vacío con dobles comillas, por último devolvemos el texto depurado.

```
61 def eliminar_palabras(text):
62     for palabra in palabras_elinimar:
63         while(palabra in text):
64             text = text.replace(palabra,"")
65     return text
```

Ilustración 8. Función de eliminación de palabras indicadas por el usuario de herramienta Appenecum

La próxima función de la ilustración 9 consiste en grandes rasgos procesar el texto depurándolo y dividiéndolo en palabras sueltas, que posteriormente contaremos cuantas veces aparecen cada palabra en el texto, para que realice una puntuación de cada una de esas palabras sueltas, teniendo en cuenta las veces que se repita la palabra y si se encuentra en alguna de las listas de parámetros iniciales vistas al principio de este punto, como la lista de *palabras_med_extrapunto* o la lista de *palabras_extrapunto* obteniendo una mayor puntuación.

Esta función recibe: *text,* que contiene el resultado de la *funcion get_text*, es decir el texto de todos los ficheros y devuelve en primer lugar *sort_resumen* que es la lista de palabras con el valor de apariencia ordenada de manera descendiente y en segundo lugar *dict_puntuacion* que es un diccionario donde *key=value,* y *palabra=puntuación obtenida.*

El funcionamiento interno de esta función es el siguiente:

1. Procesamos las listas para quitar las mayúsculas recorriendo la lista palabra por palabra y sustituyendo la palabra actual con su valor en minúsculas.

2. Excluimos palabras de la librería spacy.lang.es.stop_words donde utilizaremos la lista de palabras no deseadas de este módulo, completaremos la lista con palabras no deseadas, iteramos la lista de palabras no deseadas nuestra y completamos la lista de palabras no deseadas de la librería.

3. Eliminamos del texto los signos de puntuación, los saltos de líneas y espacios dobles y triples.

4. Cambiamos todo el texto en minúsculas, lo separamos en palabras sueltas para obtener una lista donde cada palabra del texto es un elemento y así conseguir el número de palabras con el fin de conocer el denominador común.

5. Creamos una nueva lista donde cada palabra aparece solo una vez, sin que se repitan. Para ello, recorremos la lista de todas las palabras del texto y añadimos la condición de que si la palabra aun no está en la lista de las originales y no está en la lista de palabras excluidas e incluso su longitud es mayor que 3 caracteres, incorporamos la palabra a la lista de palabras originales.

6. Declaramos un diccionario llamado resumen en el que introduciremos el contenido de nuestra sinopsis donde key = value, y palabra = número de apariencia y lo ordenamos el diccionario con el comando sorted.

7. Por último, obtenemos nuestro diccionario de puntuación recorriendo el diccionario que hemos ordenado en el paso anterior y le añadimos la condición de que si el valor es mayor que 1 o la clave está en palabras_extrapunto obtenemos la puntuación y con la misma clave lo guardamos en dict_puntuación. Dentro de esta condición introducimos otra condición que si la clave esta en palabras_extrapunto le sumamos puntos a su valor, además de que si la clave está en palabras_med_extrapunto también le sumamos puntos a su valor. Para finalizar devolvemos todos los datos obtenidos.

```
70  def resumen_palabras(text):
71
72      #Procesamos las lstas para quitar los mayusculas
73      for i in range(len(palabras_med_extrapunto)):
74          palabras_med_extrapunto[i] = palabras_med_extrapunto[i].lower()
75
76      for i in range(len(palabras_extrapunto)):
77          palabras_extrapunto[i] = palabras_extrapunto[i].lower()
78
79
80      #Palabras a excluir
81      stopwords = list(STOP_WORDS)
82      aditional_stopwords = ["©","y"] # completar la lista con palabras no deseadas
83      for word in aditional_stopwords:
84          stopwords.append(word)
85
86      resumen = {}
87      dict_puntuacion = {}
88
89      replace_list = [",",".",")","(",":","/",";","“","”","[","]"]
90      for i in range(len(replace_list)):
91          while(replace_list[i] in text):
92              text = text.replace(replace_list[i],"")
93
94      replace_list = ["\n","   "," "]
95      for i in range(len(replace_list)):
96          while(replace_list[i] in text):
97              text = text.replace(replace_list[i]," ")
98
99
100     text = text.lower()
101     # print(text)
102     palabras = text.split()
103
104     numero_de_palabras = len(palabras);
105     # print("Numero de palabras: " + str(numero_de_palabras))
106
107     palabras_originales = []
108     for palabra in palabras:
109         if(palabra not in palabras_originales and palabra not in stopwords and len(palabra)>3):
110             palabras_originales.append(palabra)
111
112     resumen = {}
113     # print(palabras_originales)
114     for palabra in palabras_originales:
115         num = palabras.count(palabra)
116         resumen[palabra] = num
117
118
119     # Con el comando sorted estamos ordenando el diccionario -> Func sorted recibe 3 parámetros
120     sort_resumen = sorted(resumen.items(), key=lambda x: x[1], reverse=True)
121
122     # Obtenemos nuestro diccionario de puntuacion
123     for element in sort_resumen:
124         if element[1] > 1 or element[0] in palabras_extrapunto:
125             dict_puntuacion[element[0]] = round(element[1] / numero_de_palabras, 4)
126             if element[0] in palabras_extrapunto:
127                 dict_puntuacion[element[0]] = dict_puntuacion[element[0]] + extrapunto
128             if element[0] in palabras_med_extrapunto:
129                 #print("palabra extrapunto_med " + str(element[0]))
130                 dict_puntuacion[element[0]] = dict_puntuacion[element[0]] + extrapunto_med
131
132     return(sort_resumen,dict_puntuacion)
```

1
2
3
4
5
6
7

Ilustración 9. Función resumen de palabras de la herramienta Appenecum

Tercer bloque:

Esta función que le sigue la podemos visualizar en la ilustración 10, que de forma general realiza un procedimiento para obtener una lista de frases que componen el texto. Lo que recibe es la variable *text* que contiene el resultado de la función *get_text,* es decir el texto de todos los ficheros y devuelve una lista donde cada elemento es una frase del texto.

El funcionamiento interno de esta función es eliminar los saltos de líneas y espacios dobles y triples, después recorremos la lista signo por signo mientras que el signo está en el texto y reemplazamos el signo por un espacio. Acto seguido separamos el texto con el comando *text_split* introduciendo un punto seguido de un espacio como parámetro con el fin de obtener una lista de frases. Luego creamos una lista nueva que rellenaremos sólo con frases que cumplan las condiciones impuestas de a continuación. Para ello primeramente vamos a iterar frase por frase y le añadimos la condición de que si la longitud de la frase es mayor que el de la frase mínima y a su vez es menor que la longitud máxima de la frase (referencia en la ilustración 5. Declaración de parámetros y contenedores globales), declaramos una variable que nos ayude a manejar el flujo del código, y dentro de esta misma condición recorremos las palabras excluidas, palabra por palabra incorporando dentro de este bucle la condición de que si encontramos una palabra excluida en la frase se modifica la variable de control porque si no la variable de control no se modifica, es decir permanece igual conteniendo el valor con el cual la hemos declarado, añadiendo la frase a la lista. Por último, devolvemos la lista de las frases donde hemos rellenado estas condiciones.

```
135 def resumen_frases(text):
136 
137     replace_list = ["\n","   ","  "]
138     for i in range(len(replace_list)):
139         while(replace_list[i] in text):
140             text = text.replace(replace_list[i]," ")
141 
142     frases = text.split(". ")
143 
144     frases2 = []
145     for i in range(len(frases)):
146         if(len(frases[i]) > frase_min_len and len(frases[i]) < frase_max_len):
147             excluir = 0
148             for j in range(len(palabras_excluidas)):
149                 if(palabras_excluidas[j].lower() in frases[i].lower()):
150                     excluir = 1
151             if(excluir == 0):
152                 frases2.append(frases[i])
153     frases = frases2
154     return frases
```

Ilustración 10. Función resumen de frases de la herramienta Appenecum

La función de a continuación se encuentra plasmada en la ilustración 11, que realiza la puntuación obtenida en la ejecución de la función *resumen_frases(text).* No recibe nada porque el resultado de *resumen_frases(text),* esta almacenado en una variable global y lo que devuelve al ejecutarse esta función es por un lado

dict_puntuacion_frases, es decir un diccionario cuya *clave:valor* y *frase:putuación,* por otro lado *sort_dict_puntuacion_frases* es una lista que contiene datos de *dict_puntuacion_frases,* pero en forma de tuplas ordenadas descendentes.

El funcionamiento interno de esta función es recorrer frase por frase, iniciar la variable a la cual vamos a sumar los puntos que obtiene cada frase y añadimos una variable auxiliar para utilizar cuando tengamos que imprimir por pantalla como es *palabras="".* Dentro de este bucle generamos otro para recorrer el diccionario *palabra:puntuación* añadiendo la condición de que si una palabra está en la frase entonces contamos cuantas veces aparece, la puntuación según le marquemos con los parámetros, utilizamos la variable auxiliar *palabras* para prints y almacenamos la frase como clave con el valor igual a la puntuación obtenida. Por último, ordenamos el diccionario de *puntuación_frases* de manera descendiente obteniendo una lista de elementos de diccionario en forma de tuplas ordenadas, devolviendo los datos obtenidos.

```
159 def puntuacion_frases():
160     for frase in frases:
161         puntos = 0;
162         palabras = ""
163 
164         for key in dict_puntuacion:
165             if key in frase.lower():
166                 multiplicador = frase.lower().count(key)
167                 puntos = round(puntos + dict_puntuacion[key] * multiplicador,3)
168                 palabras = palabras + str(key) + "(" + str(multiplicador) +")" + ":" + str(dict_puntuacion[key])
169                 + ", " dict_puntuacion_frases[frase] = puntos
170     #print("Puntuacion final: " + str(puntos))
171     #print("Palabras encontradas: " + palabras)
172     #print()
173 
174     sort_dict_puntuacion_frases = sorted(dict_puntuacion_frases.items(), key=lambda x: x[1], reverse=True)
175     return dict_puntuacion_frases,sort_dict_puntuacion_frases
```

Ilustración 11. Función puntuación de frases de la herramienta Appenecum

Esta nueva función representada en la ilustración 12 es auxiliar de *dict_puntuacion_frases,* y realiza la depuración del diccionario de frases puntuadas, eliminando las frases de menor puntuación. De esta manera podemos mantener el orden original del texto porque *dict_puntuacion_frases* que contiene las frases ordenadas según su apariencia en el texto. Lo que recibe es de la función *dict_punt_frases* y lo que devuelve es dict_puntuacion_frases2, es decir un nuevo diccionario depurado precedido de *dict_puntuacion_frases.*

El funcionamiento interno de esta función es guardar en *dict_puntuacion_frases2* todo el contenido de *dict_puntuacion_frases,* obtener la menor puntuación de frase contenida en este diccionario, después recorremos todas las frases yendo frase por frase y añadimos la condición de que el valor es más pequeño o igual al de esta frase eliminamos esta frase, salimos del segundo bucle y devolvemos el diccionario.

```
def puntuacion_frases_orden_original(dict_puntuacion_frases):
    dict_puntuacion_frases2 = dict_puntuacion_frases
    nuevo_tamaño = int(len(dict_puntuacion_frases2)*resumen_value)
    print("Tamaño original: " + str(len(dict_puntuacion_frases)))
    print("Tamaño nuevo: " + str(nuevo_tamaño))
    elementos_a_eliminar = len(dict_puntuacion_frases2) - nuevo_tamaño

    for i in range(elementos_a_eliminar):
        min_value = min(list(dict_puntuacion_frases2.values()))
        for key, value in dict_puntuacion_frases2.items():
            if value == min_value:
                del dict_puntuacion_frases2[key]
                break

    return dict_puntuacion_frases2
```

Ilustración 12. Función puntuación de frases en el orden original de la herramienta Appenecum

Cuarto bloque:

La próxima función la podemos ver reflejada en la ilustración 13, según los parámetros recibidos imprimimos el resumen. Lo que recibe primero es el parámetro *formats* que es la cadena que contiene los formatos en los cuales el usuario quiere obtener su resumen, en segundo lugar, el orden, en caso de que sea afirmativo, el resumen siga el orden original del texto, en caso negativo el resumen imprima en primer lugar las frases más puntuadas. Tercero, *resumen_name* es el nombre que vamos a poner al resumen.

El funcionamiento interno de esta función está dividido en tres partes, es decir por los tres parámetros por lo que en primer lugar vamos a obtener la fecha de hoy y declarar la variable que va a contener el cuerpo del resumen.

En segundo lugar, según el parámetro que recibamos mantenemos el orden original o no, es decir añadimos una primer condición para mantener el orden llamando a la función que nos devuelve el diccionario con el orden original, recorremos el diccionario obtenido, clave por clave y le sumamos la clave del diccionario al cuerpo, por otro lado, también recortamos el diccionario para que tenga volumen según los parámetros (*resumen_value*) eliminando las frases menos puntuadas, también recorremos el diccionario obtenido clave por clave y le sumamos la clave del diccionario al cuerpo textual. Si el diccionario se encuentra vacío, entonces imprimimos un mensaje de error, adjuntamos el mensaje al resumen y declaramos la variable donde vamos a almacenar las palabras claves. Como queremos solo 5 palabras con mayor apariencia porque así lo decidimos, añadimos la condición que, si estamos con la última palabra, es decir en la quinta palabra cuando haya una coma le sumamos un punto y concatenamos esta quinta palabra, también las palabras de posición del 1 al 4.

En tercer lugar, creamos una variable que va a contener los nombres de los ficheros originales que iteraremos fichero por fichero y el nombre del fichero lo vamos a concatenar a la variable. Declaramos una variable para el texto completo resumen, al texto del resumen lo concatenamos con la fecha de creación actual, el nombre del archivo, las palabras claves y el cuerpo textual.

Cabe destacar que según el dato que contiene la variable formats inicialmente imprimimos el resumen, es decir tanto si es en TXT como en PDF se imprimen en ambos formatos el resumen.

```
def imprimir(formats,orden,resumen_name):

    today = date.today()
    cuerpo = ""

    if(orden == "si"):
        d = list(puntuacion_frases_orden_original(dict_puntuacion_frases))
        for i in range(len(d)):
            cuerpo += d[i] + "." + "\n"
    else:
        d = sort_dict_puntuacion_frases[:int((len(sort_dict_puntuacion_frases)*resumen_value))]
        for i in range(len(d)):
            cuerpo += d[i][0] + "." + "\n"

    if(len(d)==0):
        print("Longitud del resumen es 0. Probablemente hay demaciadas palabras excluidas")
        cuerpo += "Longitud del resumen es 0. Probablemente hay demaciadas palabras excluidas"

    palabras_clave = ""
    for i in range(5):
        if(i == 4):
            palabras_clave = palabras_clave + sort_resumen[i][0] + "."
        else:
            palabras_clave = palabras_clave + sort_resumen[i][0] + ", "

    resumen_name = "resumen_" + resumen_name

    text = ""
    text += "Creado: " + str(today) + "\n\n"
    text += "Resumen autogenerado del articulo: " + resumen_name + "\n\n"
    text += "Palabras claves: " + palabras_clave + "\n\n"
    text += cuerpo

    print(text)

    if("txt" in formats):
        f = open(resumen_name + ".txt","w+",encoding="utf8")
        f.write(text)
        f.close()
```

Ilustración 13. Función de imprimir de la herramienta Appenecum

Quinto bloque:

Esta última función reflejada en la ilustración 14, sirve para crear un fichero PDF con un contenido y un nombre dado que es recibido en los parámetros. Recibe por un lado el contenido del texto y el nombre del archivo en PDF y lo que devuelve es un fichero en formato PDF.

El funcionamiento interno de esta función lo podemos dividir también en 2 partes. En primer lugar, le aplicamos formato al texto obteniendo una hoja de estilo donde seleccionamos el nombre y estilo de la fuente, el tamaño y color de las letras y palabras, etc. y creamos una lista vacía dentro de la variable *story.* En segundo lugar, grabamos el nombre del archivo en PDF después de generarlo con el parámetro doc de modo que nos muestra la plantilla del documento simple con una serie de atributos los cuales van a configurar el archivo, como los márgenes o tamaño de las

páginas. Para finalizar esta función hemos dividido en párrafos el texto del PDF añadiéndoles en la lista y de esta manera construimos el texto estructurado.

```
243 def text_to_pdf(text_content, filename):
244     styles = getSampleStyleSheet()
245     styleN = styles['Normal']
246     styleH = styles['Heading1']
247     story = []
248 
249     pdf_name = filename+'.pdf'
250     doc = SimpleDocTemplate(
251         pdf_name,
252         pagesize=letter,
253         bottomMargin=.4 * inch,
254         topMargin=.6 * inch,
255         rightMargin=.8 * inch,
256         leftMargin=.8 * inch)
257 
258     parrafos = text_content.split("\n")
259     for parrafo in parrafos:
260         P = Paragraph(parrafo, styleN)
261         story.append(P)
262     doc.build(
263         story,
264     )
```

Ilustración 14. Función de transformación a PDF de la herramienta Appenecum

Originalmente, se priorizó el desarrollo de las funcionalidades del programa frente al desarrollo gráfico. Debido a ese motivo, se quiere reflejar que el diseño de la algoritmia se ha ido modificando durante el trancurso de este trabajo.

En primera instancia, los documentos se cargaban en la aplicación como valores de una variable, no se tenía sistema de autenticación, las funciones no estaban encapsuladas y no se tenía gestión de errores. Posteriormente, una vez el código era plenamente funcional, se refactorizaron aquellas partes que eran hábiles de ser mejoradas.

En este epígrafe se puede observar la evolución del código desde su fase inicial hasta la ahora aquí presentada:

```
302 #main
303 # pdf to txt
304 ficheros = ["Articulo8"]
305 resumen_name = "ArticuloPDF"
306 pdf_to_txt(ficheros)
307 text = get_text(ficheros)
308 text = eliminar_palabras(text)
309 
310 sort_resumen,dict_puntuacion = resumen_palabras(text)
311 frases = resumen_frases(text)
312 dict_puntuacion_frases,sort_dict_puntuacion_frases = puntuacion_frases()
313 imprimir("txt, pdf","si",resumen_name)
```

Ilustración 15. Inicio del método main de la herramienta Appenecum

```
405 def main():
406     first_window()
407 
408 if __name__ == '__main__':
409     main()
```

```
def load_pdf():
    global document
    document = filedialog.askopenfilename(filetypes = [('Text files', '.pdf')])
    pdf_info.set(os.path.basename(document))
```

Ilustración 16. Final del método main de la herramienta Appenecum

4.2.3 Implementación de la interfaz gráfica

En esta sección se describen los apartados más relevantes en la creación de una interfaz, a pesar de que nuestra herramienta contiene dos, se han empleado el mismo código para ambas cambiando valores para definir y configurar las interfaces por separado. También se destacan algunos aspectos como abrir el explorador de archivos en windows, los mensajes de error y la mostración de la salida exclusivamente en PDF con la aplicación predeterminada que utilice el usuario con su ordenador, ya que son importantes para complementar y perfeccionar sobre todo la segunda interfaz, que es donde está incorporado nuestro método, confeccionando en su conjunto la herramienta Appenecum.

Creación de una interfaz:

La interfaz gráfica en Tkinter [33] se inicializa creando un root que será el objeto tkinter sobre el que se escribirán el resto de widgets [34] que se mostrarán por pantalla. Siempre que root exista, el programa se ejecutará, para ello, al final del código deberá se existir una sentencia root.mainloop(). En la sección 5.2 demostrativa volvemos a nombrar la librería Tkinter pero con un enfoque más visual.

La orden anterior, es un bucle infinito que mantiene root activo.

El flujo de la aplicación se optó por un modelo secuencial ya que se decidió primar por un control integral del contenido que se le iba a mostrar al usuario en cada momento frente al modelo de libre navegación.

Posteriormente, una vez definido el flujo que consistirá nuestro método y el root, se procede a establecer la configuración inicial que queremos que tenga la pantalla como:

- Dimensiones.
- Restricción de reescalado.
- Título de la aplicación.
- Posición absoluta o relativa
- Etc..

En nuestra aplicación, ese proceso se ha llevado a cabo de la siguiente manera:

```
def first_window():
    root = Tk()
    myFrame = Frame(root)
    root.resizable(True, False)
    root.title('Héctor Valdés Ferrer')
    ancho_ventana, alto_ventana = 450, 200
    x_ventana = root.winfo_screenwidth() // 2 - ancho_ventana // 2
    y_ventana = root.winfo_screenheight() // 2 - alto_ventana // 2
    posicion = posicion = str(ancho_ventana) + "x" + str(alto_ventana) + "+" + str(x_ventana) + "+" + str(y_ventana)
    root.geometry(posicion)
    myFrame.pack()
```

Ilustración 17. Creación de una interfaz de la herramienta Appenecum

Creación de los campos de texto y campos editables:

La primera pantalla que nos aparece al iniciar la aplicación es el login. En esta ventana se ha hecho uso de diferentes propiedades que ofrece la librería.

Las propiedades que se han usado se denominan widgets que son componentes de nuestras interfaces y que podemos personalizar.

- Label: crea una etiqueta de tipo texto. Es el texto que se muestra por pantalla. Al tratarse de texto plano, es decir que no interactúa dentro de la interfaz podemos configurarlo mediante .grid indicando la posición.

- StringVar: define el tipo de dato que se espera recibir. Contiene 2 estados, un campo de bloqueo con "redonly", es decir de no escritura y otro en el podemos introducir información y rescatar esa misma información con "normal". Tienen un funcionamiento parecido a los labels en cuanto al posicionamiento con el añadido de que al usuario le hacemos partícipe interactuando con la interfaz.

- Entry: empaqueta diferentes parámetros y genera un campo que adquiere dicha configuración. En relación con los widgets anteriores, se usa para aceptar las cadenas de texto del recuadro en una sola línea del usuario, utilizando el widget text si se quiere mostrar varias líneas de texto que se pueden modificar o con widget label para manifestar una o varias líneas de texto que el usuario no puede modificar. Que a su vez tiene la opción de poder recuperar el texto actual de su entrada estableciendo relación en una instancia de la clase StringVar

```
main_greetings = Label(myFrame,
            text = 'Bienvenido a la herramienta Appenecum',
            width = 42,
            font = ('Courier', 12)).grid(row = 0, column = 0, padx = 0, pady = 10, sticky = "w", columnspan = 2)

log_text = Label(myFrame,
                text = 'Usuario',
                width = 9,
                anchor = "e",
                font = ('Arial', 12)).grid(row = 1, column = 0, padx = 0, pady = 10, sticky = "e")

user_log = StringVar()
user_login = Entry(myFrame, textvariable = user_log, width = 30)
user_login.grid(row = 1, column = 1, padx = 5, pady = 12)
user_login.config(state = 'normal')

psw_text = Label(myFrame,
                text = 'Contraseña',
                width = 9,
                anchor = "e",
                font = ('Arial', 12)).grid(row = 2, column = 0, padx = 0, pady = 10, sticky = "e")

user_psw = StringVar()
user_pswin = Entry(myFrame, textvariable = user_psw, width = 30)
user_pswin.grid(row = 2, column = 1, padx = 5, pady = 12)
user_pswin.config(state = 'normal', show = '*')
```

Ilustración 18. Creación de campos textos y editables de la herramienta Appenecum

Creación de un botón:

- Button: llamada a una función onclick, se utilizan para iniciar procesos, en este ejemplos en concreto, se usa para pasar a la segunda interfaz. En la elaboración del conjunto de interfaces existen varios botones dan comienzo a otras funciones ejecutando otros procesos.

```
go_in = Button(myFrame,
               text = 'Entrar',
               command = lambda:go_second_window(),
               width = 25).grid(row = 3, column = 0, padx = 0, pady = 8)
```

Ilustración 19. Creación de un botón de la herramienta Appenecum

Mensajes de información, alertas y error:

Cuando el usuario se identifica se llaman a las funciones que comprueba su identidad y en caso de error se genera una excepción de tipo *messagebox* que cancela el proceso en curso y devuelve un mensaje de error:

```
messagebox.showinfo(message = "Usuario incorrecto", title = "Proceso cancelado"), delete_elements()
```

Ilustración 20. Creación de un mensaje erróneo de la herramienta Appenecum

Una vez se ha ingresado al sistema, se crea una segunda ventana finalizando el proceso de root.mainloop() de la primera interfaz, con lo que conseguimos una gestión de ventanas y memoria eficiente.

Apertura del explorador de archivos de windows:

El proceso de carga de archivos se realiza mediante un *askopenfilename,* el cual, permite llamar el explorador de archivos. El diccionario contenido en la lista, otorga la posibilidad de definir el formato de archivos queremos buscar, ocultando todos los que sean diferentes.

```
askopenfilename(filetypes = [('Text files', '.pdf')])
```

Ilustración 21. Apertura del explorador de archivos en local de la herramienta Appenecum

Visualizar con la aplicación predeterminada en PDF:

Por último, la forma de mostrar la salida generada es mediante la llamada al webbrowser.

```
webbrowser.open_new('Exit.pdf')
```

Ilustración 22. Visualización de la salida en formato PDF con la aplicación predeterminada

5 EXPERIMENTACIÓN Y EVALUACIÓN

En un principio por comodidad, estructura y sobre todo para una mejor visualización en las salidas de ejecución, se ha implementado el código por bloques o notebooks diferentes, que posteriormente hemos unido a través de funciones en un mismo notebook o hoja de cálculo como hemos observado en el punto anterior. Cada bloque constaba de una tarea propia que no aparecen en los otros bloques y actúan de forma secuencial, es decir detrás de uno otro hasta conseguir nuestra salida un resumen como resultado final a partir del texto original que es la finalidad de este TFG.

5.1 Funcionamiento de Appenecum mediante un ejemplo práctico

Los documentos escogidos para nuestro método son de la plataforma ScienceDirect (explicado en el apartado 1.1 en la definición del problema), aplicamos un filtro de búsqueda para obtener los artículos relacionados con el covid más recientes que nos interesan, es decir de los últimos publicados en este año 2022 y seleccionamos cuatro artículos de origen español en distintos hospitales.

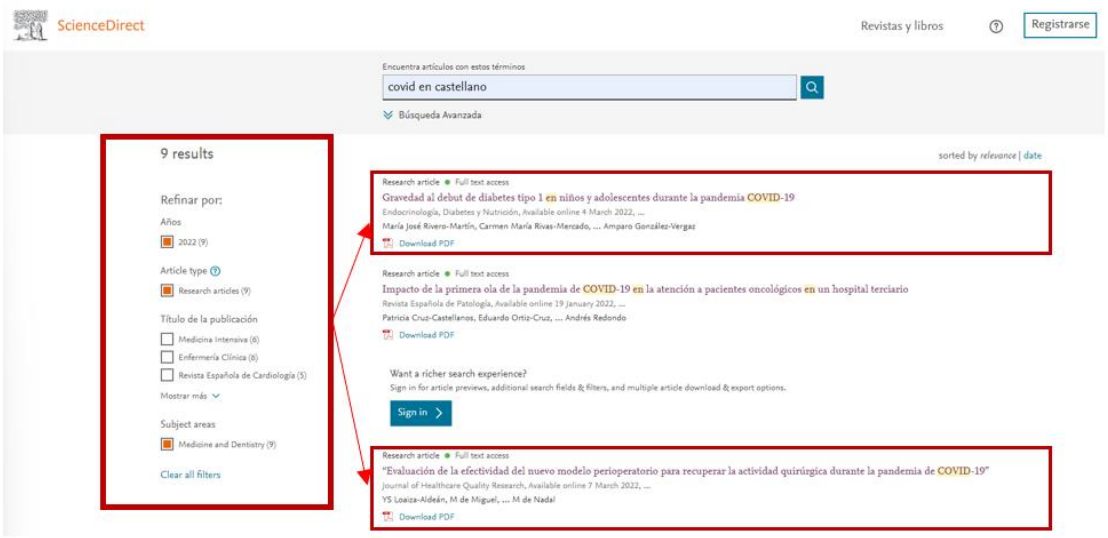


Ilustración 23. Selección de artículos médico-científicos con la plataforma ScienceDirect

A medida que se han ido integrando los bloques en funciones, se ha ido testando el código con sus librerías pertinentes y comprobando las salidas por pantalla, por lo que también han sufrido modificaciones y cambios a medida que se iban implementando hasta alcanzar su versión definitiva. El artículo científico número 10 es el que hemos evaluado como ejemplo en las tareas a realizar dentro de cada bloque, siendo el mismo resultado para cualquiera de los artículos descargados de nuestra batería de documentos en PDF.

BLOQUE 1:

En este primer bloque consideramos en un principio que podíamos obtener de un artículo científico original en PDF transformarlo en un TXT adecuado, es decir recibir como entrada un PDF y conseguir como salida un TXT.

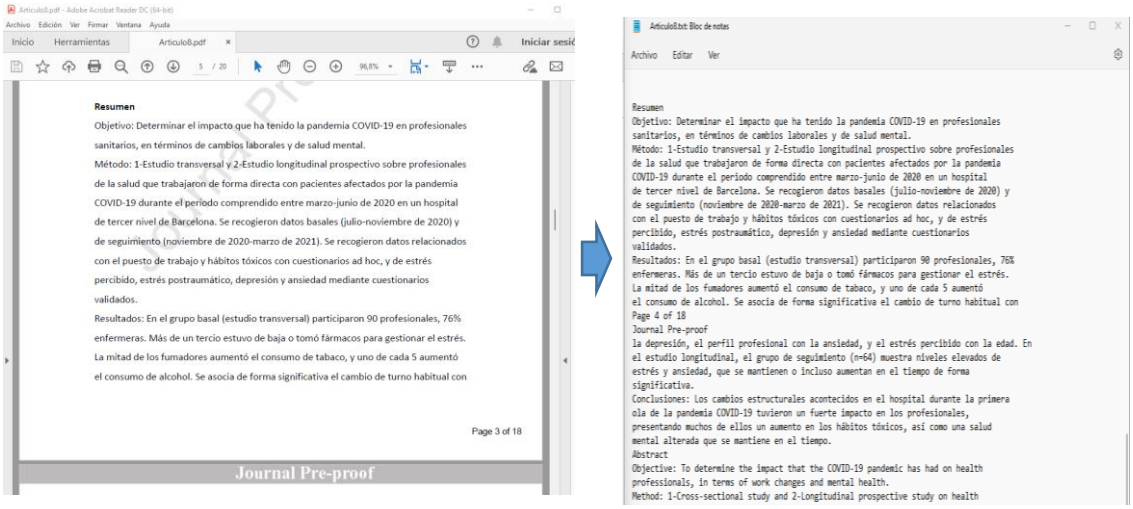


Ilustración 24. Transformación del artículo 10 original en PDF a TXT de la herramienta Appenecum

Para esta conversión la utilidad de la librería Fritz importada e implementada en nuestro método como se ha podido observar en la ilustración 6 ha resultado satisfactoria apreciándose en la ilustración 17 para poder empezar a construir nuestro código y conformar el siguiente bloque.

BLOQUE 2:

Este bloque tiene como objetivo principal extraer palabras de un artículo científico para poder puntuarlas, es decir clasificar de una batería de palabras independientes para procesarlas del texto original eliminando palabras que carecen de valor en nuestro sistema de puntuación como son preposiciones, determinantes o caracteres específicos como arroba, interrogaciones, exclamaciones, etc. Para esta valoración, incorporamos la librería spacy, ya comentada en la tabla 3 de la sección 4.2.3 y pudimos comprobar lo gran efectiva que es para el PLN y para la utilidad que necesitamos de segmentar las oraciones y captar palabras destacadas del texto. Nuestra evaluación es positiva ya que funciona correctamente con palabras en español, además también utiliza esta librería la herramienta Textdigester para el procesamiento del texto.

En este bloque se han evaluado las siguientes tareas como un sistema de filtrado para una mejor visualización:

- Procesar el texto en formato lista para eliminar las mayúsculas y cambiar todas las palabras a minúsculas.
- Eliminación manual de caracteres como son las comas, dos puntos, paréntesis, arrobas, etc. Donde se pueden añadir de manera manual.

- Crear para almacenar listas de palabras y un diccionario de puntuación sin que se repitan las palabras del texto original. De esta manera podemos contabilizar y añadir las palabras a un diccionario que constará parte de nuestro resumen final, permitiendo ser ordenado además de incluir dentro nuestro sistema de puntuación y así evaluar las palabras.

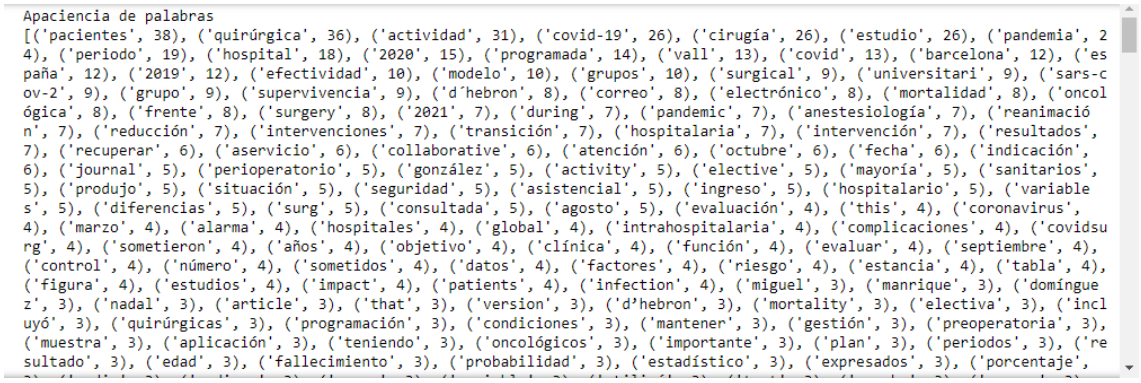


Ilustración 25. Segmentación y puntuación del artículo 10 de la herramienta Appenecum

BLOQUE 3:

En este bloque, evaluamos primero las frases de todo el texto por separado midiendo sus longitudes en caracteres, es decir estimamos según nuestro criterio que una frase debe contener entre 100 y 700 caracteres.

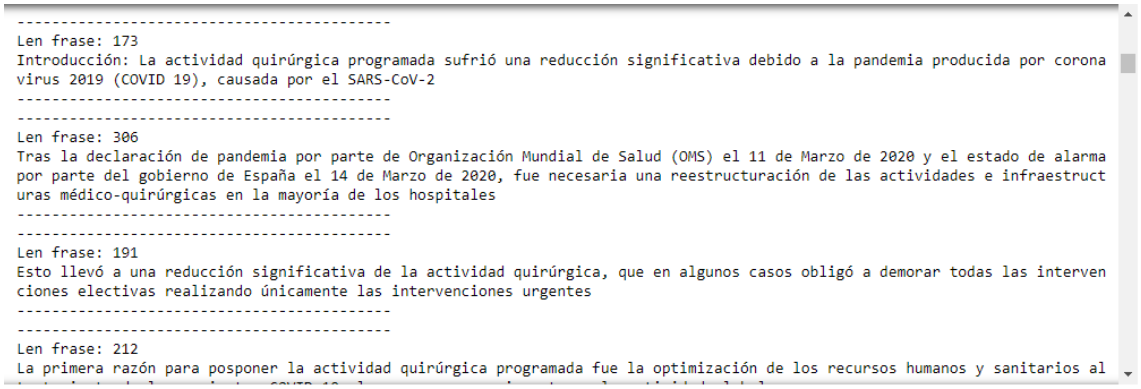


Ilustración 26. Longitud y separación por frases del artículo 10 de la herramienta Appenecum

En segundo lugar, un poco más complejo, evaluamos la funcionalidad de un diccionario denominado puntuación de frases donde almacenar estas frases manteniendo el orden textual, que a su vez lo comparamos con el diccionario de puntación de palabras vistas en la ilustración 25 del bloque 2 para que, si una palabra está en la frase, cuenta las veces que aparece según los parámetros que hemos establecido y almacene las frases con el valor igual a la puntuación obtenida quedando de la siguiente forma:

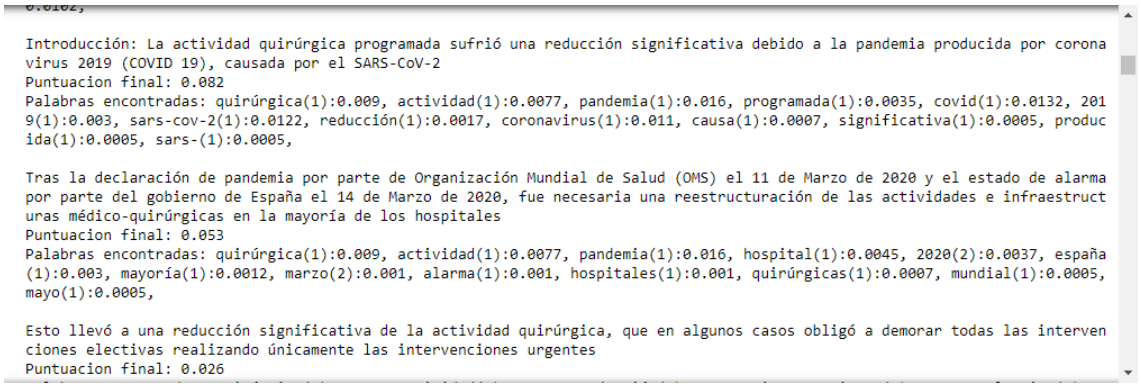


Ilustración 27. Puntuación y frecuencia de repeticiones por frases del artículo 10 de la herramienta Appenecum

BLOQUE 4:

Los resultados evaluados en este bloque se centran en el resumen como salida del método. Unificamos las frases con su respectivo puntaje y las ordenamos tal y como aparecen en el texto original.

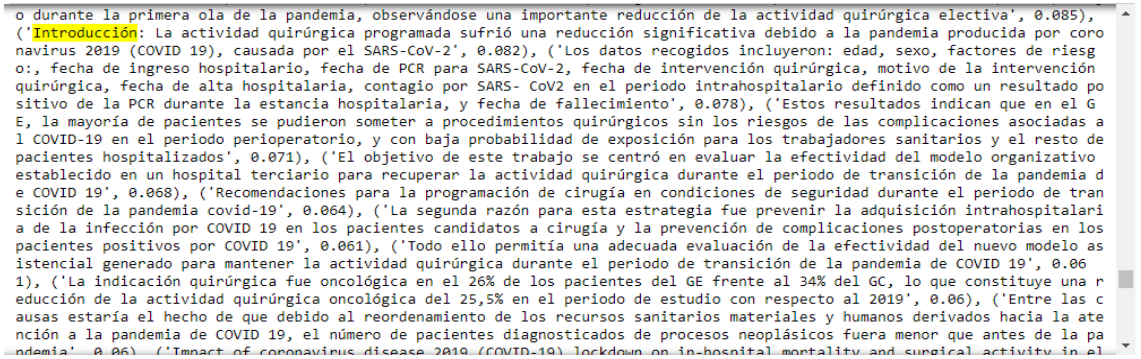


Ilustración 28. Unificación y puntuación de las frases del artículo 10 de la herramienta Appenecum

Generamos el resumen definitivo seleccionando las frases con mayor puntuación, es decir, con las de mayor número de repeticiones respecto a nuestro diccionario procesado y imprimimos con otra función de imprimir, generando un archivo en la carpeta del explorador de Windows obteniendo una salida en formato TXT. Esto fue una de las modificaciones que se hicieron por las creaciones posteriores de las interfaces hasta llegar a la versión final de Appenecum. Comparándola con el artículo original en PDF hemos obtenido un resultado como el que se muestra a continuación:

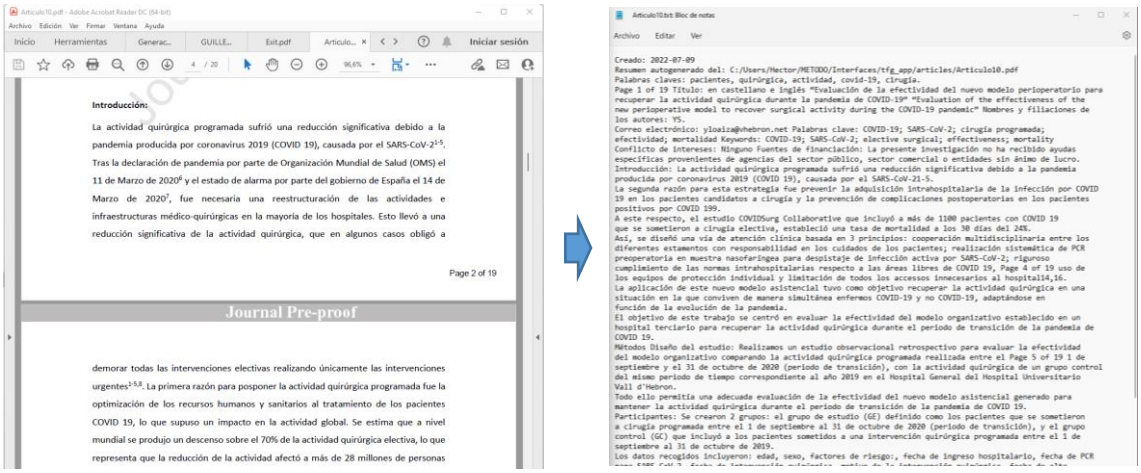


Ilustración 29. Comparativa del artículo 10 original en PDF con la salida en TXT de la herramienta Appenecum

BLOQUE 5:

Una vez impreso el resumen en TXT y transformar esa salida en PDF generando un nuevo archivo constituye la evaluación de este bloque. Este último bloque se le aplica la librería fitz como en el bloque 1 pero a la inversa, es decir, una vez que tenemos el texto en formato txt ya manipulado, se le añade la función conversora a PDF para darle formato e incorporarle el nombre de salida al nuevo pdf transformado.

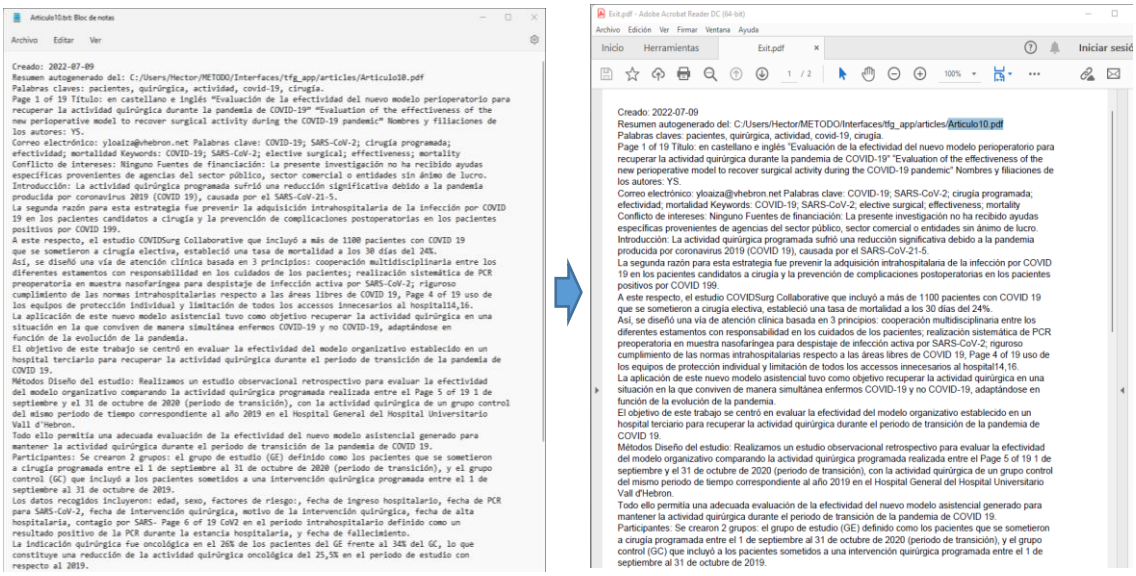


Ilustración 30. Transformación aplicada del artículo 10 en TXT de nuestro método a PDF con la herramienta Appenecum

Recapitulando las partes más relevantes de los bloques evaluadas, podemos simplificar el flujo de actuación de la siguiente manera:

1	Descargamos el articulo original en PDF de la plataforma ScienceDirect	2	Convertimos el artículo de PDF a TXT con la librería Fitz
4	Volvemos a convertir el artículo de TXT a PDF con la misma librería Fitz	3	Le introducimos el método de nuestra herramienta Appenecum

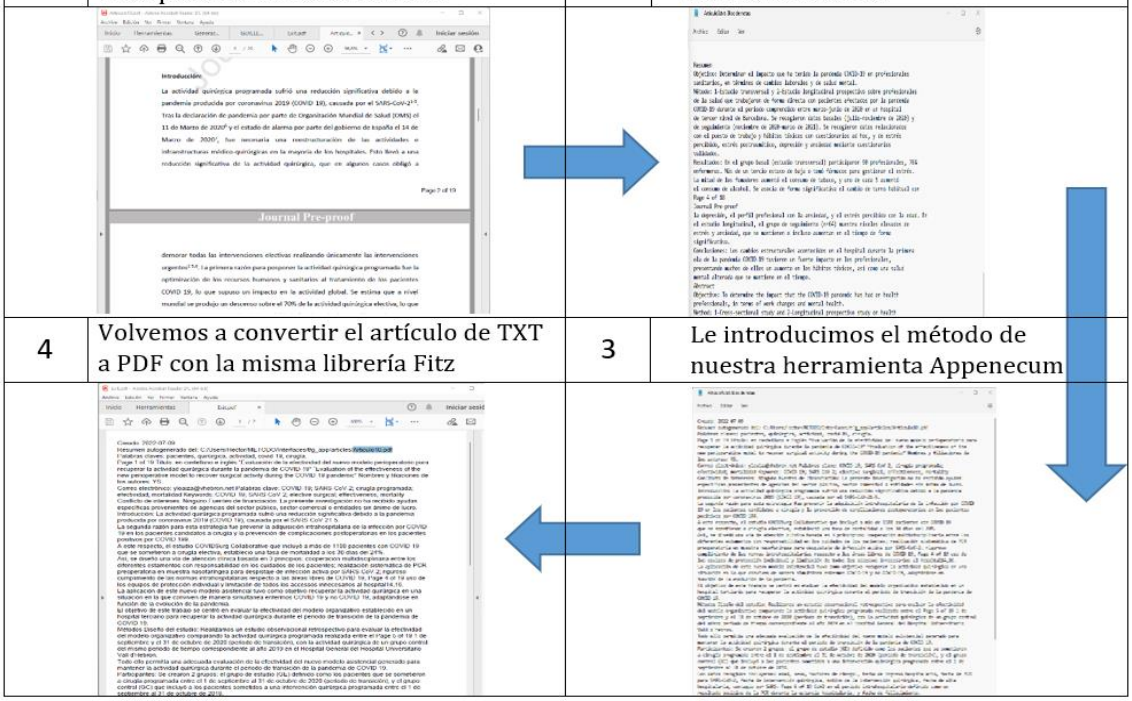


Tabla 5. Flujo de evaluación en la generación de un resumen del artículo 10 con la herramienta Appenecum

5.2 Demostración

En este apartado vamos a desarrollar brevemente y visualmente como está formada la interfaz gráfica, asimismo está orientada para que el usuario final pueda utilizar la aplicación en un entorno real. Durante la creación de las distintas pantallas que se le muestran al usuario se ha buscado en todo momento que su diseño fuese lo más usable e intuitivo posible.

La librería que se ha empleado ha sido Tkinter como hemos mencionado en la sección 4.2.3 Implementación de la interfaz gráfica. Es un paquete que viene integrado por defecto en Python para el kit de herramientas GUI que se encuentra disponible para la mayoría de los sistemas Unix y Windows mantenido por ActiveState [35].

El motivo principal por el que se ha decidido usar tkinter frente a otras alternativas es debido a varios motivos en los que se destacan:

1. Documentación externa y gran cantidad de recursos disponibles
2. Curva de iniciación favorable
3. Paquete integrado en Python
4. Generación de ejecutables altamente eficientes

En primer lugar, se plantearon diferentes diseños gráficos para modelar como deberían de estar conformadas y se dibujaron bocetos con lápiz y papel los elementos compuestos en las dos interfaces. El concepto es conseguir que la primera sirviera como acceso a la segunda que es donde se genera nuestro resumen.

Para la configuración de la primera pantalla se tuvo en cuenta cual sería la relación del aspecto más idóneo para mostrar el contenido teniendo en cuenta que solo se le iba a mostrar un formulario de acceso a la pantalla principal.

Tras establecer la configuración deseada el resultado es el siguiente:

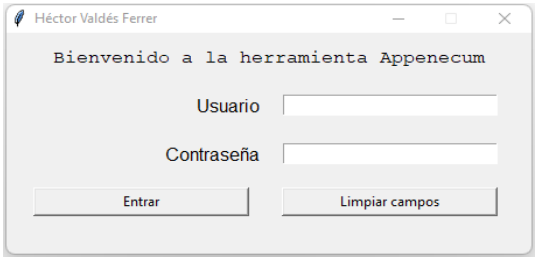


Ilustración 31. Formulario de login en la primera interfaz de la herramienta Appenecum

El formulario de acceso implementa ciertas características que dan soporte al usuario en caso de que fuese necesario. Si durante el proceso de autenticación el usuario o la contraseña se introdujesen erróneamente la aplicación devolvería una alerta haciéndole saber al usuario cual de los dos campos ha sido el incorrecto.

Ilustración 32. Mensajes de error al introducir un usuario y una contraseña incorrecto en la primera interfaz de la herramienta Appenecum

En caso de que el proceso de autenticación sea satisfactorio se construye una segunda ventana que da acceso a las funcionalidades del programa.

Los elementos importantes a destacar en este punto serían:

1. La generación dinámica de botones durante el flujo de ejecución
2. La carga sincronía de los datos accediendo al menú de navegación del SO.
3. Interrupción del proceso y mensajes de error.

En menor medida:

1. Visualización de datos: hora de inicio de la sesión y nombre del archivo que se está cargando. A traves de la librería datetime [36] que nos facilita la manipulación de las fechas y horas.

La ventana principal presenta el siguiente aspecto:

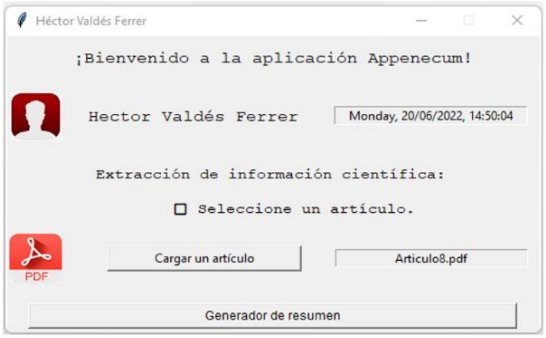


Ilustración 33. Interfaz principal de la herramienta Appenecum

La gestión de errores de la ventana principal consiste en tener cuenta diversos aspectos como:

1. Que exista un artículo cargado sobre la aplicación para poder proceder a la extracción de características; en caso de que el supuesto anterior no ocurriese al usuario se le devolvería el siguiente mensaje de error:

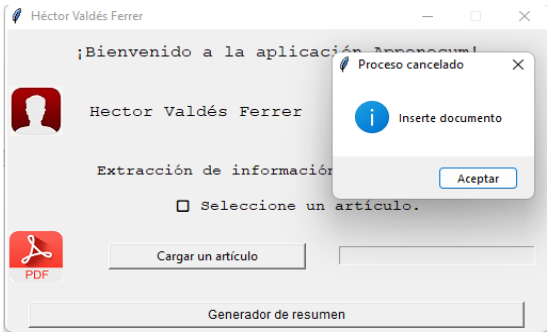


Ilustración 34. Mensaje de error por no cargar ningún artículo en la interfaz principal de la herramienta Appenecum

2. Otro de los aspectos que se han querido reforzar ha sido la ayuda a la búsqueda del documento deseado; para ello, mediante un interfaz amigable como es la del explorador de Windows restringimos la visualización de todos aquellos elementos que no tengan formato .PDF.

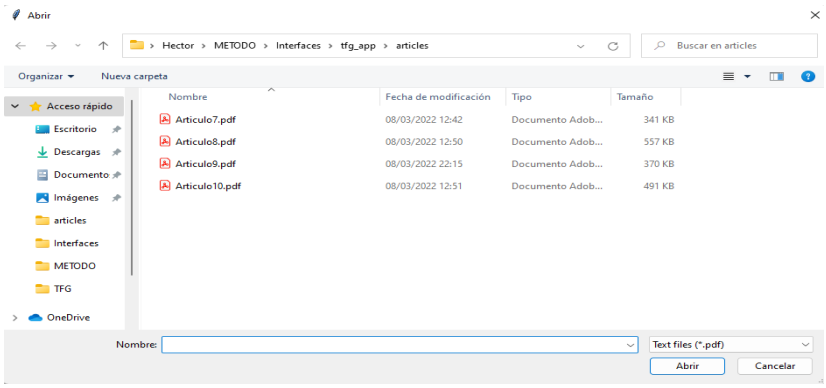


Ilustración 35. Búsqueda exclusiva de archivos con formato PDF del explorador de archivos de windows en la interfaz principal de la herramienta Appenecum

Si el flujo ha sido el correcto, el procedimiento para cargar el artículo en la aplicación es pulsar sobre el botón "Cargar un artículo" > "Seleccionar el artículo deseado mediante el explorador de archivos" > "Visualizar que el documento se ha cargado correctamente".

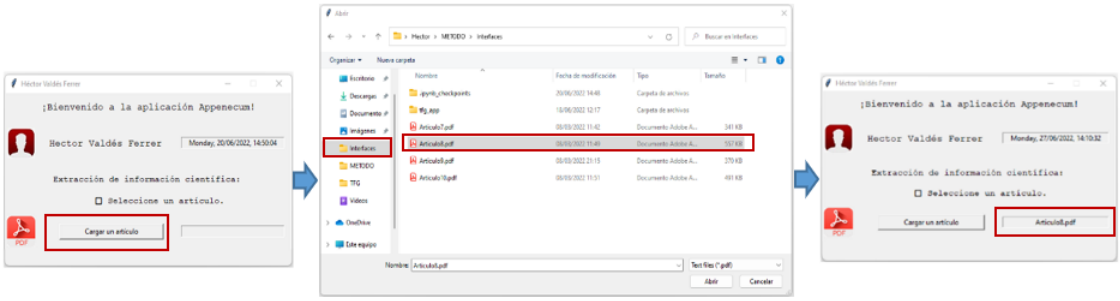


Ilustración 36. Funcionamiento del botón cargar artículo en la interfaz principal de la herramienta Appenecum

El documento que se crea cuando el usuario pulsa sobre el botón "Generador de resumen" se abre con la aplicación predeterminada del sistema para visualizar ese tipo de archivos.

En el caso de ejemplo que se muestra, la aplicación por defecto es Adobe Acrobat, sin embargo, si no existiese ningún software en el ordenador sobre el que se está ejecutando la aplicación, la salida del programa se vería en el navegador por defecto.

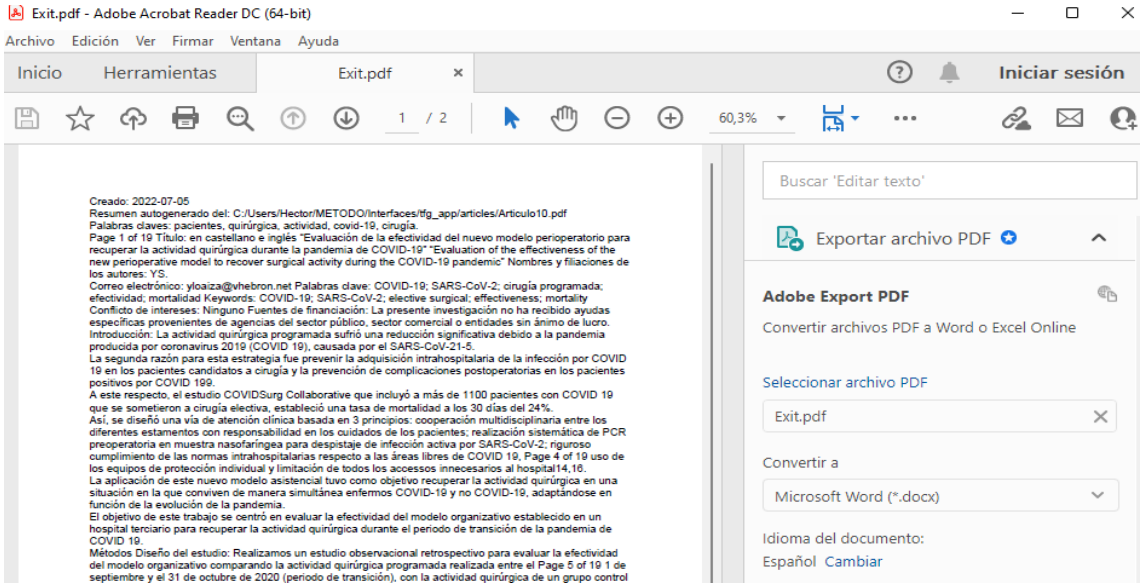


Ilustración 37. Resumen generado del artículo 8 científico por la herramienta Appenecum

Realizando una comparativa visual entre el artículo generado por la herramienta Appenecum de la ilustración 30 y el original descargado de la plataforma ScienceDirect de la ilustración 31, ambos en PDF observamos una reducción considerable del número de páginas del original que consta de 18 páginas al resumen generado por nuestra herramienta siendo 2. Además, mantiene una correlación en la estructura organizativa igual que en el original, resumiendo de forma secuencial los puntos tratados del artículo en cuestión.

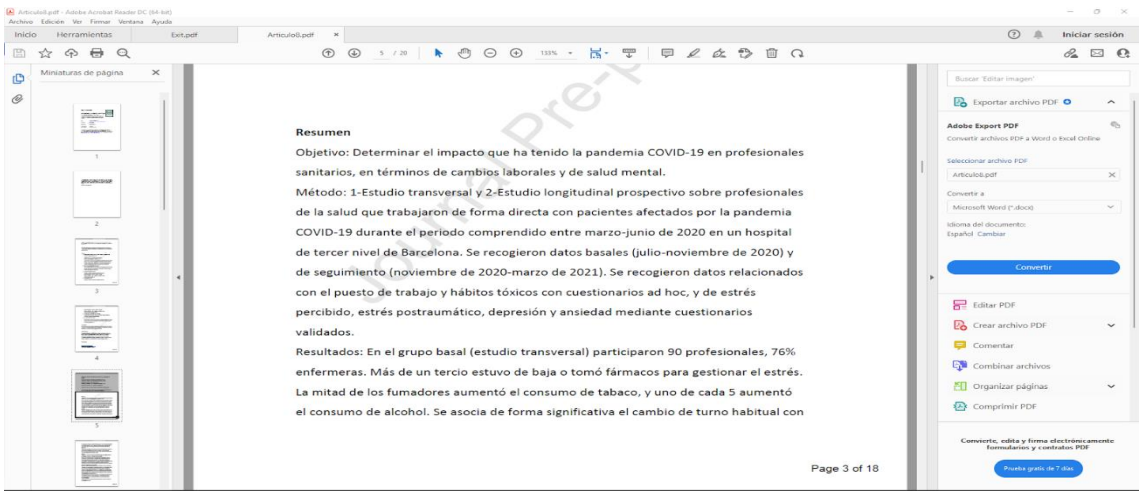


Ilustración 38. Artículo 8 científico original, descargado directamente de la plataforma ScienceDirect

Recogiendo la información de este punto y haciendo un resumen gráfico con los pasos a seguir que el usuario debe seguir para que la herramienta funcione correctamente son 3 sencillos pasos:

1. Introduces los campos de usuario y contraseña correctamente, y pulsas entrar.
2. Cargas un artículo científico en PDF para que funcione en el botón.
3. Pinchas sobre el botón de generador de resumen para obtener el resumen.

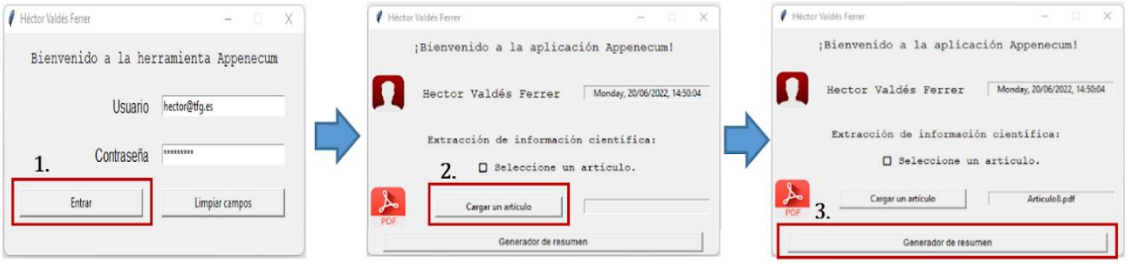


Ilustración 39. Resumen en la forma de proceder por el usuario en la interfaz principal de la herramienta Appenecum

5.3 Resultados de la evaluación

A lo largo de los últimos años las métricas han servido para medir la eficacia de los modelos de generación de resúmenes siendo los más utilizados Rogue y Bleu. En nuestro trabajo sólo hemos utilizado la métrica Rouge que es la encargada de medir cual es la precisión y el recall de un sistema [37]. Es una métrica ampliamente utilizada para el resumen de texto que a su vez se compone de varias variantes métricas, pero la idea básica detrás de todas ellas es asignar una sola puntuación numérica a un resumen que nos dice qué tan bueno es en comparación con uno o más resúmenes de referencia. En definitiva, Rouge compara un resumen generado con uno o más resúmenes de referencia. Cabe destacar que actualmente no existe forma alguna que sea segura al 100% de eficacia para la medición en la generación de resúmenes.

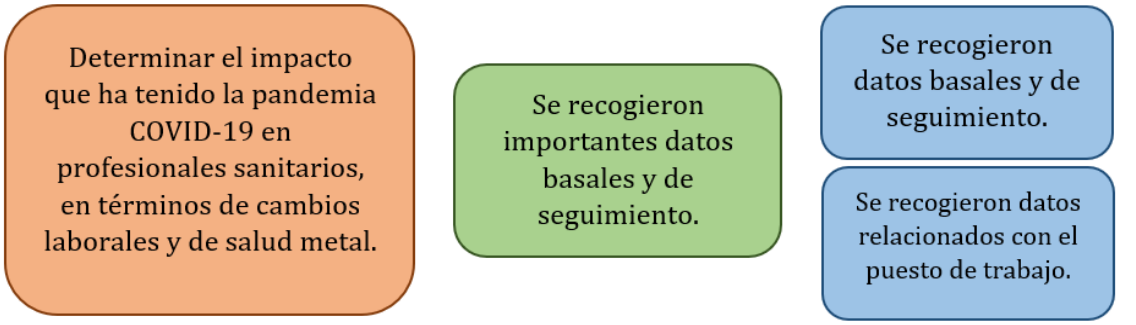
Los conceptos para medir la calidad de un resumen son la precisión y el recall que son métricas complementarias debido las cuales posibilitan eludir resúmenes demasiados largos o cortos. La precisión es la encargada de medir la cantidad de palabras destacadas en el interior del resumen generado en relación del resumen que esperamos. En otras palabras, cuál es el porcentaje de palabras del resumen generado que se hallan dentro del resumen esperado. La repetición sucesiva de la misma palabra nos proporcionará porcentajes elevados de precisión, por esta razón entre otras podemos llegar a tener dificultades en nuestra medición y es aquí cuando aparece en escena el recall que es la encargada de la medición en la cantidad de palabras que esperamos del resumen encontradas en el interior del resumen generado. Aunque con esto también podemos tener un problema principal ya que, si tenemos un resumen muy largo, con multitud de palabras distintas ampliará las probabilidades de repetir nuestro resumen generando las palabras del resumen esperado. Estas medidas se complementan y se utilizan juntas para evitar los problemas más comunes, aunque no corrigen todos.

Las tres métricas variantes más comunes de la métrica Rouge que hemos utilizado han sido:

- Rouge-1: calcula la precisión y el recall como hemos explicado en el párrafo anterior, partiendo de los unigramas del resumen generado y del resumen esperado.
- Rouge-2: esta variante es parecida a ROUGE-1, pero en vez de utilizar unigramas usa bigramas.
- Rouge-L: esta variante difiere mucho de las dos anteriores. Lo que realiza es una búsqueda de la cadena común más larga que se encuentra contenida dentro de ambos resúmenes.

Existen más tipos de métricas, pero Rouge-1 y Rouge-2 pueden agruparse en la denominada Rouge-N, que incluye todos los tipos de n-gramas.

Para un mejor entendimiento vamos a visualizar con un ejemplo de una frase de uno de los artículos científicos escogidos, concretamente el artículo 8, todo lo expuesto anteriormente.

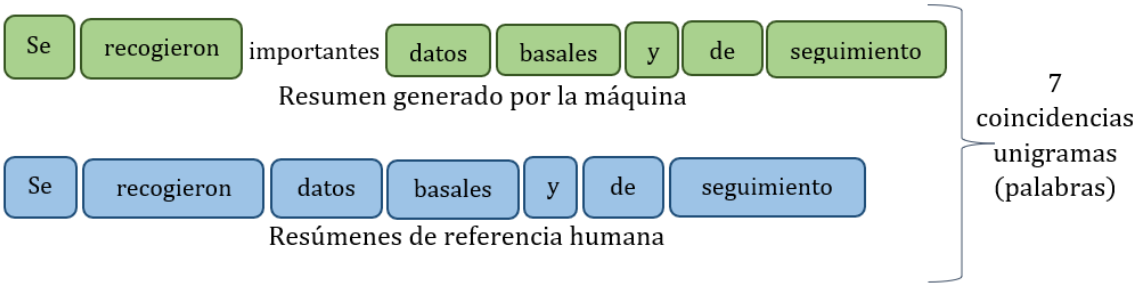


Frase del artículo_8 científico | Resumen generado por la máquina | Resúmenes de referencia humana

ROUGE compara un resumen generado con uno o más resúmenes de referencia.

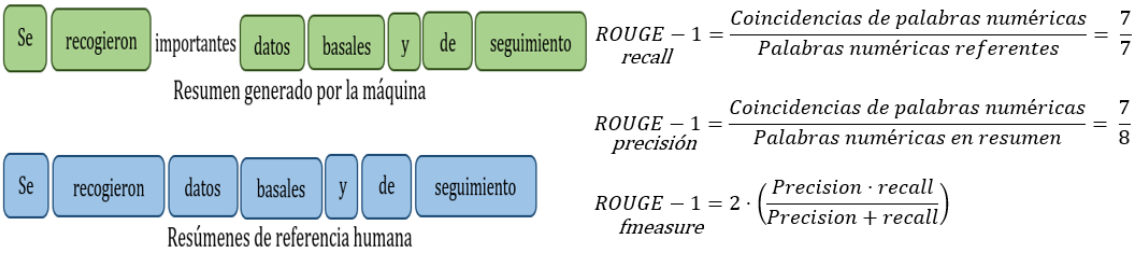
Tenemos una oración de un de los artículos científicos seleccionados anteriormente que ha sido resumido. Si comparamos el resumen generado con algunos resúmenes humanos de referencia, podemos ver que el modelo es bastante bueno y solo difiere en una o dos palabras. Para poder medir la calidad de un resumen de forma automática, el enfoque que toma rouge es comparar los n-gramas del resumen generado con los n-gramas de las referencias. Un n-grama es sólo una forma elegante de decir un trozo de n palabras, así que podemos comenzar con los unigramas que corresponden a las palabras individuales en una oración.

En este mismo ejemplo, podemos observar que seis de las palabras en el resumen generado también se encuentran en uno de los resúmenes de referencia.



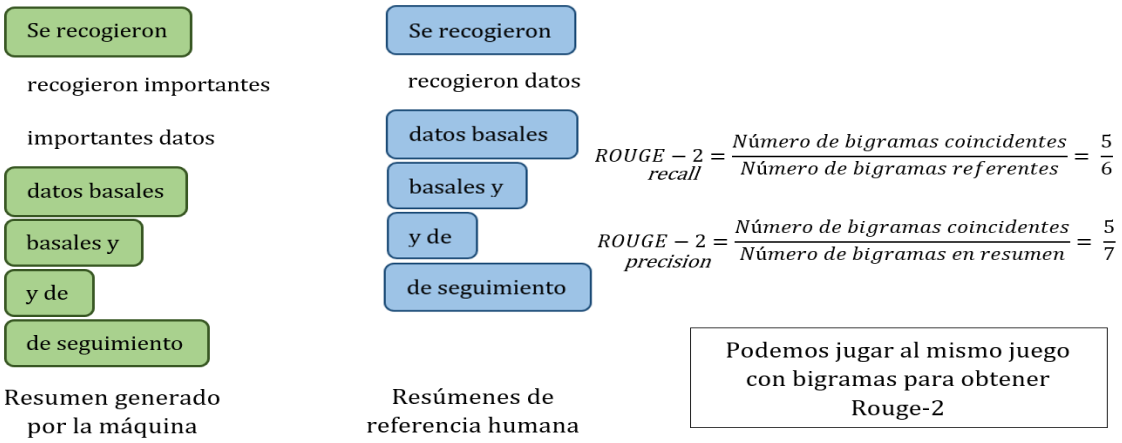
Rouge-N compara los n-gramas de la generación con los n-gramas de las referencias

La métrica rouge que compara los unigramas se llama Rouge-1. Ahora que hemos encontrado coincidencias, una forma de asignar una puntuación al resumen es calcular el recall de los unigramas, esto significa que solo contamos el número de palabras coincidentes en el resumen generado y de referencia, normalizamos el recuento dividiendo por el número de palabras en la referencia. Con este ejemplo se va a entender mejor. Encontramos siete palabras coincidentes y nuestra referencia tiene siete palabras, por lo que nuestro recall de unigrama es perfecto. Lo que significa que todas las palabras en el resumen de referencia se han producido en el generado.

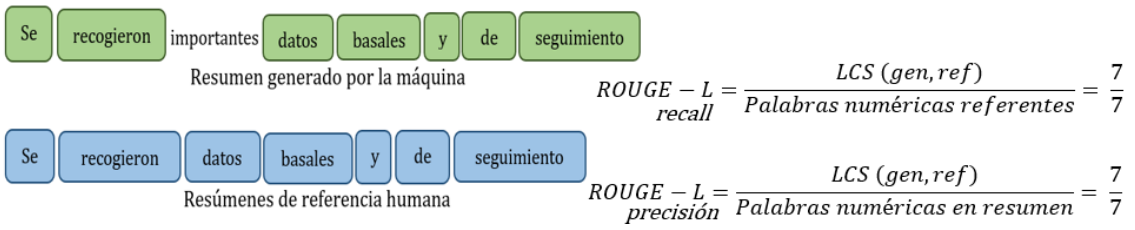


También podemos calcular la precisión, que en el contexto rouge mide qué parte del resumen generado fue relevante en la práctica. Por lo general, se calcula tanto la precisión como la recuperación y luego se informa el puntaje F-measure. El valor de F-measure se emplea para determinar la precisión y la cobertura como medida.

Ahora se puede cambiar la granularidad de la comparación comparando bi-gramas en lugar de unigramas. Con bi-gramas dividimos la oración en pares de palabras consecutivas y luego contamos cuántos pares en el resumen generado están presentes en el de referencia esto nos da precisión Rouge-2 y recuerda cuál podemos ver, es más bajo que los puntajes de la ruta 1 de antes. Ahora, si los resúmenes son largos, los puntajes de rouge 2 generalmente serán pequeños porque hay menos bi-gramas para igualar y esto también es válido para la síntesis abstracta, por lo que se suele informar de las puntuaciones de Rouge-1 y Rouge-2.



La última métrica variante que discutiremos es la Rouge-L, que no compara los n-gramas, sino que trata cada resumen como una secuencia de palabras y luego busca la subsecuencia común más larga o "LCS". Entendemos por subsecuencia como una secuencia que aparece en el mismo orden correlativo, pero no necesariamente contiguo, así como se ve en este ejemplo de a continuación.



A la hora de calcular las puntuaciones de las distintas métricas Rouge en los conjuntos de datos es muy sencillo, sólo hay que utilizar la función de carga0 métrica y proporcionar los resúmenes del modelo con las referencias. Calculamos las puntuaciones de Rouge en los conjuntos de datos con la función *load_metric().*

```
# UNA ORACIÓN DEL ART.8 COMO TEST DE COMPROBACIÓN PARA UN EJEMPLO
from datasets import load_metric
rouge = load_metric("rouge")
predictions = ["Se recogieron importantes datos basales y de seguimiento"]
references = ["Se recogieron datos basales y de seguimiento"]
rouge.compute(predictions=predictions,references=references)
```

```
{'rouge1': AggregateScore(low=Score(precision=0.875, recall=1.0, fmeasure=0.9333333333333333), mid=Score(precision=0.875, recall=1.0, fmeasure=0.9333333333333333), high=Score(precision=0.875, recall=1.0, fmeasure=0.9333333333333333)),
 'rouge2': AggregateScore(low=Score(precision=0.7142857142857143, recall=0.8333333333333334, fmeasure=0.7692307692307692), mid=Score(precision=0.7142857142857143, recall=0.8333333333333334, fmeasure=0.7692307692307692), high=Score(precision=0.7142857142857143, recall=0.8333333333333334, fmeasure=0.7692307692307692)),
 'rougeL': AggregateScore(low=Score(precision=0.875, recall=1.0, fmeasure=0.9333333333333333), mid=Score(precision=0.875, recall=1.0, fmeasure=0.9333333333333333), high=Score(precision=0.875, recall=1.0, fmeasure=0.9333333333333333)),
```

Ilustración 40. Valores de la métrica Rouge en una frase del artículo 8 de la herramienta Appenecum

La salida del cálculo contiene mucha información, lo primero que podemos ver es que los intervalos de confianza de cada puntuación se proporcionan en los campos bajo, medio y alto, lo cual es muy útil si queremos saber la dispersión de las puntaciones de ruta al comparar dos o más modelos.

Una vez visto todos los conceptos de Rouge nuestra evaluación de la herramienta Appenecum con la métrica Rouge consta de 3 fases:

Fase 1: Valorar el tamaño del resumen original. El resumen está manualmente extraído del artículo. Lo dividimos en frases y el resumen del artículo original consta de 9 frases.

Fase 2: traemos el método original y hacemos un resumen automático. Un criterio obligatorio es que el resumen generado debe contener exactamente el mismo número de frases contenidas en el abstract del documento original que el resumen generado por nosotros que en este caso consta un total de 9. Por lo que también modificamos el método para que el resumen no adjunte cabeceras, referencias o similares. Para hacerlo de una forma más simple, modificamos el método *puntuacion_frases_oredn_original(),* donde manualmente modificamos el parámetro *nuevo_tamaño,* igualando su valor a 9. Para terminar, modificamos el método *imprimir()* para que no imprima el resumen en un fichero, sino que lo devuelva.

Fase 3: aplicamos la métrica Rouge del resumen original, es decir del abstract de cada artículo científico con respecto a nuestro resumen generado por nuestra herramienta Appenecum. Los resultados de las métricas han sido los siguientes valores:

- Artículo 7:

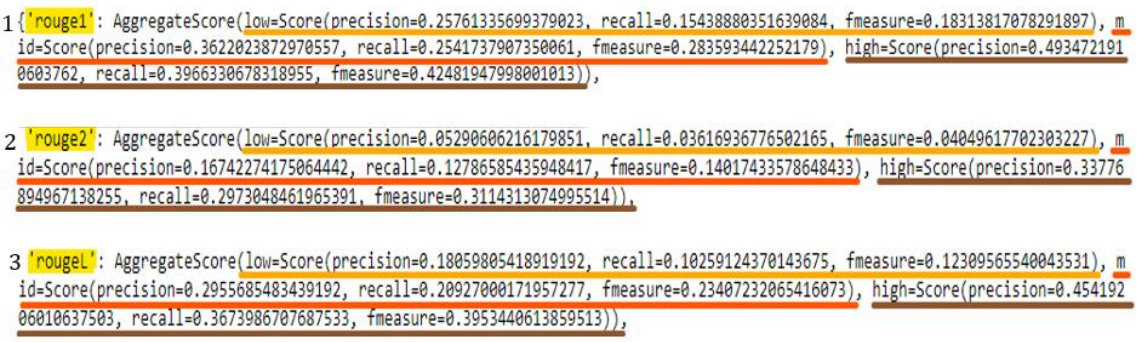


Ilustración 41. Valores de la métrica Rouge para el artículo 7 de la herramienta Appenecum

- Artículo 8:

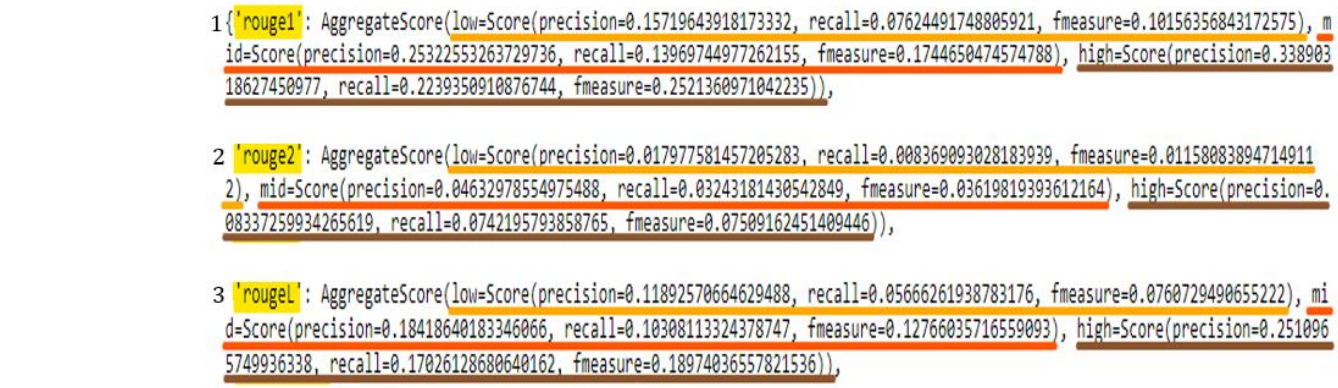


Ilustración 42. Valores de la métrica Rouge para el artículo 8 de la herramienta Appenecum

- Artículo 9:

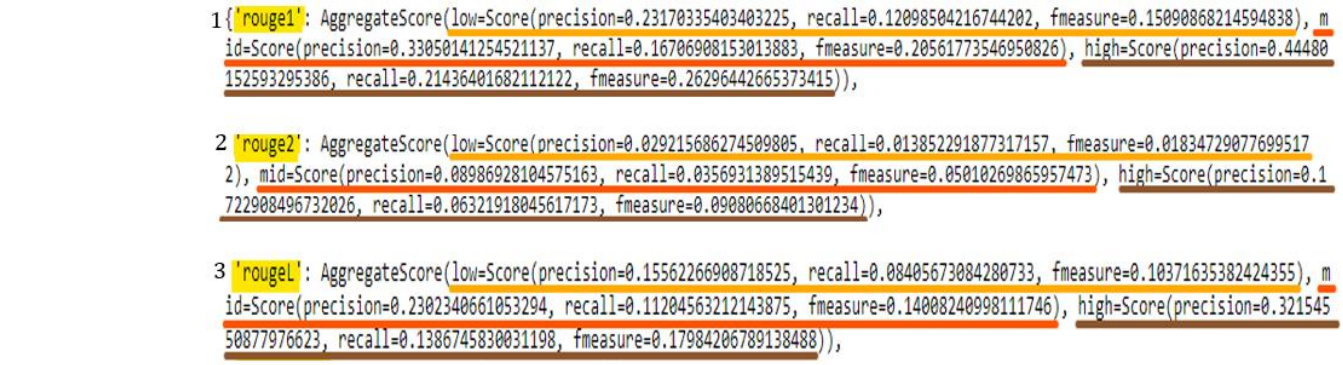


Ilustración 43. Valores de la métrica Rouge para el artículo 9 de la herramienta Appenecum

Recapitulamos todos los valores obtenidos en las siguientes tablas comparativas para una visualización y entendimiento más correcto utilizando la media de los campos bajo (en color naranja), medio (en color rojo) y alto (en color marrón) en tanto por ciento de la frase del artículo científico número 8 mencionada como ejemplo explicativo, además de tres resúmenes generados por nuestra herramienta Appenecum respecto a los abstract de los documentos originales con todos los valores obtenidos con la herramienta Rouge y sus diversas métricas obteniendo los siguientes resultados:

ROUGE-1	Precisión	Recall	F-measure
Ejemplo frase del artículo 8	87,50	100	93,30
Artículo 7	37,11	26,83	29,71
Artículo 8	24,97	14,66	17,60
Artículo 9	33,56	16,74	20,65

Tabla 6. Resultados de la métrica Rouge-1 del ejemplo frase del artículo 8 y varios artículos científicos completos con la herramienta Appenecum

ROUGE-2	Precisión	Recall	F-measure
Ejemplo frase del artículo 8	71,42	83,33	76,92
Artículo 7	18,60	15,37	16,40
Artículo 8	13,44	3,83	4,09
Artículo 9	10,71	3,75	8,27

Tabla 7. Resultados de la métrica Rouge-2 del ejemplo frase del artículo 8 y varios artículos científicos completos con la herramienta Appenecum

ROUGE-L	Precisión	Recall	F-measure
Ejemplo frase del artículo 8	87,50	100	93,33
Artículo 7	31,03	22,63	25,08
Artículo 8	18,47	10,94	13,11
Artículo 9	23,58	11,16	14,12

Tabla 8. Resultados de la métrica Rouge-L del ejemplo frase del artículo 8 y varios artículos científicos completos con la herramienta Appenecum

Podemos observar que tienen los porcentajes más elevado el ejemplo de la frase del artículo científico número 8, debido a que consta de un total de 7 palabras la oración mientras que en los artículos científicos completos contiene más de 4.000 palabras ya que constan de 20 páginas cada uno por lo que la dispersión es muy grande y la eficacia del rendimiento se reduce considerablemente por el tamaño de la longitud.

Unificando aún más los valores obtenidos de cada tabla por las diferentes métricas variantes que componen Rouge, añadimos la tabla 7 para alcanzar mediante el cálculo de la media para cada criterio de medida precisión, recall y f-mesure obteniendo una estimación en conjunto como resultados finales.

MEDIA DE ROUGE	Precisión	Recall	F-measure
Artículo 7	29,19	21,81	23,95
Artículo 8	18,81	10,10	11,94
Artículo 9	22,77	10,67	14,25
TOTAL	**23,59**	**14,20**	**16,71**

Tabla 9. Media de todas las variantes métricas Rouge

Con la medición de las métricas Rouge podemos decir que el rendimiento de nuestra herramienta oscila entre un 15% y un 25% debido a que la media del conjunto de características de nuestro método en 3 artículos científicos obtiene los resultados de la tabla 7.

6 DISCUSIÓN Y POSIBLES LÍNEAS DE MEJORAS DE NUESTRA HERRAMIENTA APPENECUM

Este punto se subdivide en dos puntos, por un lado, la discusión de los errores que hemos ido encontrando en la realización de nuestro método, donde a su vez hemos dividido este punto en dos partes para destacar los problemas principales resaltando dos problemas considerando los más importantes que derivan a unos secundarios que expondremos en el punto de a continuación. En el otro punto, describimos algunas mejoras que se pueden realizar después de analizar distintos errores con el fin de corregir y facilitar a la hora de crear métodos para la generación de resúmenes.

6.1 Análisis de errores

Problemas principales:

En este apartado hemos querido destacar dos problemas principales que a su vez generan más dificultades para el funcionamiento del método. Partimos de dos problemas generales que para poder corregirlos hemos tenido que subsanar una serie de inconvenientes que se iban propagando a cada paso dado. Poco a poco hemos tratado errores particulares hasta alcanzar nuestro objetivo para cada tarea que nos íbamos proponiendo.

El primer problema principal consiste en la conversión del fichero en PDF a un fichero TXT para el procesamiento llevado a cabo posteriormente. Una vez realizada la conversión con la librería Fitz hemos observado que la salida obtenida no es la más idónea, es decir no se adecua exactamente a lo que buscamos para conseguir una salida limpia. Por una salida adecuada entendemos aquella que no tiene artefactos. Por ejemplo, si un fichero en formato PDF tiene una marca de agua impresa en sitios aleatorios por todo el documento, en su versión de conversión futura tendrá lugar a una cadena de texto y posteriormente a una posible confusión de una frase ya que se repetirá varias veces durante todo el texto. Como solución a la adaptación de esta librería con el documento biomédico y para que la salida fuera lo más limpia posible ha sido la maquetación de forma manual, es decir primero observamos los errores secundarios corregidos en la entrada y en segundo lugar lo adaptamos en código.

El segundo problema principal es la estructura del texto en cada documento porque no es común para todos los ficheros sino todo lo contrario. Hemos podido comprobar que el aspecto físico en la redacción de los artículos científicos no sigue unas normas universales establecidas impuestas por las editoriales ni revistas, sino que cada autor o autores elaboran los artículos por cuenta propia y personal. Algunos documentos tienen textos en dos columnas, otros ponen los signos de

puntuación en los sitios donde en otros documentos no los ponen, etc. Todo esto complica bastante la segmentación del texto a la hora de poder dividirlo en frases, así que hemos realizado una búsqueda recapitulando artículos que aparentemente fuesen lo más parecido posible.

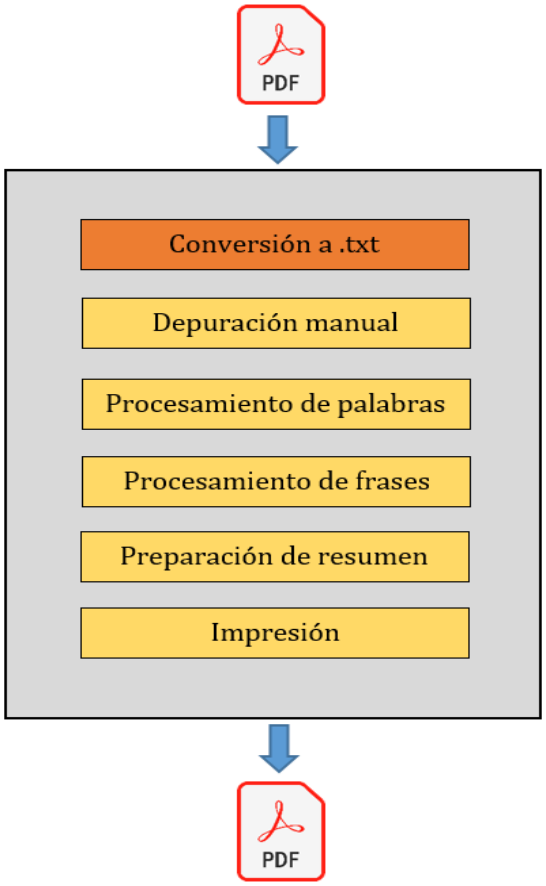


Ilustración 44. Resumen gráfico de problemas principales de la herramienta Appenecum

Problemas secundarios:

A medida que realizamos comprobaciones, nos encontramos con una serie de inconvenientes que obstaculizan la progresión del funcionamiento de nuestra herramienta Appenecum. Algunas de las correcciones más significativas que se emplearon para los objetivos de las tareas se llevasen a cabo han sido:

1. Números de páginas y elementos repetidos.

```
Page 2 of 18
Journal Pre-proof
```

Ilustración 45. Problemas secundarios con números de páginas y elementos repetidos en la elaboración de la herramienta Appenecum

Algunas particularidades de ciertos artículos que se repiten, si son regulares, es decir que se repiten sucesivas veces a lo largo del texto como "Journal Pre-proof" pueden ser excluidas, pero si son irregularidades como "Page 2 of 18" es bastante

complicado elaborar un algoritmo para eliminarlos. Además, que los regulares acaban en un fichero.

2. Símbolos especiales y acentos

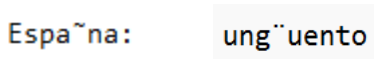
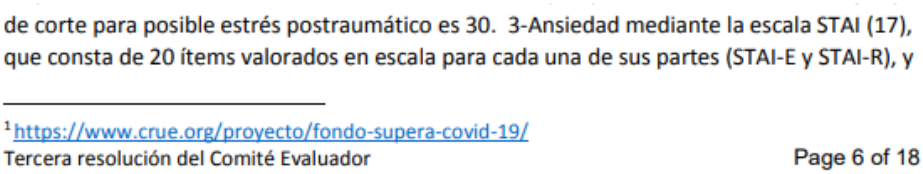


Ilustración 46. Problemas secundarios con símbolos especiales y acentos en la elaboración de la herramienta Appenecum

Problemas con la decodificación de algunos PDF. En general el formato PDF no está pensado para una conversión de retroceso, sino todo lo contrario, lo usamos para convertir varios documentos a este formato y así poder transportarlos de forma cómoda. En nuestro caso, en una conversión de PDF a TXT se producen problemas con algunos caracteres especiales. Como solución se pueden elaborar algoritmos que corrigen algunos de estos problemas, pero no en todas.

3. Elementos que son texto, pero no parte del cuerpo, los llamados footers.


de corte para posible estrés postraumático es 30. 3-Ansiedad mediante la escala STAI (17), que consta de 20 ítems valorados en escala para cada una de sus partes (STAI-E y STAI-R), y

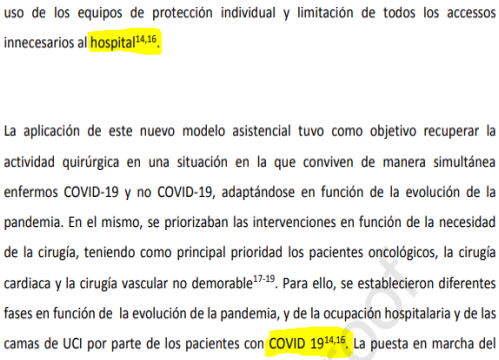
[1] https://www.crue.org/proyecto/fondo-supera-covid-19/

Tercera resolución del Comité Evaluador Page 6 of 18

Ilustración 47. Problemas secundarios con los pies de páginas en la elaboración de la herramienta Appenecum

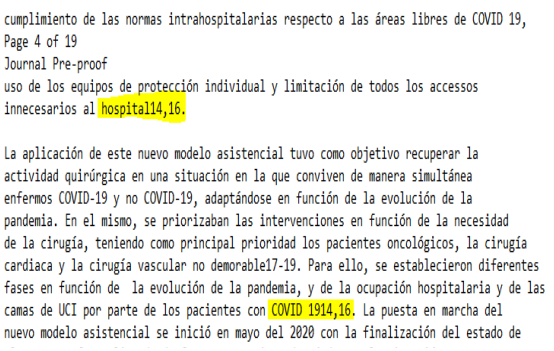
El mismo problema nos encontramos con los footers, o elementos que no forman parte del texto original pero que aparecen cerca de él. La librería conversor no puede interpretarlo como podemos visualizar los humanos a simple vista y lo reconoce como parte del cuerpo incluyéndolo dentro.

4. Referencias


uso de los equipos de protección individual y limitación de todos los accessos innecesarios al hospital[14,16].

La aplicación de este nuevo modelo asistencial tuvo como objetivo recuperar la actividad quirúrgica en una situación en la que conviven de manera simultánea enfermos COVID-19 y no COVID-19, adaptándose en función de la evolución de la pandemia. En el mismo, se priorizaban las intervenciones en función de la necesidad de la cirugía, teniendo como principal prioridad los pacientes oncológicos, la cirugía cardiaca y la cirugía vascular no demorable[17-19]. Para ello, se establecieron diferentes fases en función de la evolución de la pandemia, y de la ocupación hospitalaria y de las camas de UCI por parte de los pacientes con COVID 19[14,16]. La puesta en marcha del

Fragmento original del artículo científico 8


cumplimiento de las normas intrahospitalarias respecto a las áreas libres de COVID 19,
Page 4 of 19
Journal Pre-proof
uso de los equipos de protección individual y limitación de todos los accessos innecesarios al hospital14,16.

La aplicación de este nuevo modelo asistencial tuvo como objetivo recuperar la actividad quirúrgica en una situación en la que conviven de manera simultánea enfermos COVID-19 y no COVID-19, adaptándose en función de la evolución de la pandemia. En el mismo, se priorizaban las intervenciones en función de la necesidad de la cirugía, teniendo como principal prioridad los pacientes oncológicos, la cirugía cardiaca y la cirugía vascular no demorable17-19. Para ello, se establecieron diferentes fases en función de la evolución de la pandemia, y de la ocupación hospitalaria y de las camas de UCI por parte de los pacientes con COVID 1914,16. La puesta en marcha del nuevo modelo asistencial se inició en mayo del 2020 con la finalización del estado de

Fragmento generado por Appenecum del artículo científico 8

Ilustración 48. Problemas secundarios con las referencias en la elaboración de la herramienta Appenecum

Para un mejor entendimiento podemos observar en las Ilustración 49 como en algunos utilizados hemos encontrado referencias que no están ubicadas con corchetes [ ] sino escritas como exponenciales. En el plazo de tiempo establecido para la realización de este proyecto no hemos encontrado una forma de solucionar este tipo de problema, por lo tanto, estas cosas ha sido necesario depurarlo manualmente.

5. Signos de puntuación o su ausencia.

   Principalmente este problema nos preocupa sólo cuando tenemos que separar el texto en frases. En un texto narrativo las frases están separadas una de otra por un punto. El problema es que existen elementos como títulos, subtítulos, footers, header y otros que no son parte del cuerpo.

6. Imágenes o tablas.

   El PDF es un formato poco amigable para manipular datos debido a que la transformación de los datos ya sean imágenes o bien tablas con información de contenido no suele ser del todo correcto en la mayoría de las veces, es decir las librerías no convierten exactamente igual que el archivo original, incluso puede llegar a ser una tarea bastante compleja y larga de programar si disponen de muchas observaciones a tener en cuenta.

7. Ordenación de nuestro diccionario.

   En la primera versión del método para ordenar nuestro diccionario importamos la librería OrderedDict de Python, pero posteriormente nos dimos cuenta de que se podía ordenar mejor con el comando sorted.

6.2 Mejoras en favor de los resúmenes

Una vez realizada nuestra herramienta y durante la confección de esta, han ido surgiendo varias mejoras que se podrían aplicar en un futuro con el fin de perfeccionar el método y aumentar los porcentajes de rendimiento para ello hemos conformado una lista mostrando aspectos que puedan ser reforzados a nuestro método.

- La creación de una librería universal que transforme correctamente, de PDF a TXT sería idóneo para obtener una entrada estándar porque facilita mucho el trabajo a la hora de depurar el texto.

- A partir de las técnicas extractivas para líneas futuras podrían mejorarse añadiendo técnicas abstractivas que reescriban las frases del documento original, de manera que completen la frase con similitudes.

- El número de unidades léxicas de las lenguas naturales, así como a la constante incursión de palabras nuevas o nuevas acepciones de palabras existentes.

- Considerar la utilización del método propuesto en base a resúmenes de múltiples documentos, mejorando el algoritmo para que permita tener como entrada varios artículos PDF que no sólo pertenezcan al mismo tema como es el covid sino que también hablen del mismo tema porque al tratarse de un enfoque extractivo selecciona las partes relevantes de cada texto, de lo contrario perdería coherencia y faltaría congruencia en el resumen generado.

- Es interesante distinguir el idioma en el que se encuentra escrito el artículo o documento, porque en nuestro método la selección de palabras cambia si es en español o si es en inglés por lo que la ponderación de las frases también, modificando el resultado de las métricas. La unificación de distintas lenguas en una misma o diferentes librerías donde ya contengan el mayor número de palabras relacionadas en nuestro caso con el ámbito biomédico incluyendo todas las nuevas palabras relacionadas con el covid puede ser una mejora sugerente sobre todo a la hora de crear cualquier algoritmo.

- Otro problema para resolver es el tratamiento de tablas y gráficos a la hora de construir el resumen, aunque la librería Fritz convierte de manera aceptable, se denota la falta de cohesión con el texto e incluso nos muestra la información desordenada y dispersa en determinadas ocasiones. Al ser elementos que tienen presencia constantemente en los textos científicos y, normalmente comprenden información relevante, se deberían tener en cuenta en el resumen.

7 CONCLUSIONES Y TRABAJOS FUTUROS

En esta última sección se presenta el razonamiento del autor una vez que se han alcanzado los objetivos establecidos y se ha documentado de la manera más precisa y factible posible todo el proceso llevado a cabo para la creación de la herramienta Appenecum. En esta etapa, se analiza el punto de vista del autor y se reflexiona sobre los resultados obtenidos, destacando los logros y los desafíos superados en el camino hacia la materialización de Appenecum. Esta sección ofrece una visión integral y reflexiva de todo el proceso de desarrollo de la herramienta. Con lo aprendido durante el tiempo de elaboración se pone de manifiesto posibles líneas de investigación y próximos trabajos venideros que sirvan como progreso y evolución de las actuales, incluida la herramienta Appenecum.

7.1 Conclusiones

Contar con un resumen es siempre una ventaja independiente del contexto que nos encontremos, ya sea un marco político, educativo o cualquier otro, se trata de una tarea de interés en todos los niveles ya que salimos todos beneficiados con la masificación de documentos que tenemos disponibles. A pesar de parecer una tarea sencilla, realizar un resumen es una actividad compleja y lo es más aún, cuando se trata de artículos científicos porque implica la habilidad y creatividad para aplicar, lo que la lingüística llama "propiedades del discurso", propuestas por Van Dijk en 2003 [38] como son los temas del discurso, los esquemas discursivos, el estilo, los recursos retóricos, los actos del habla y múltiples dimensiones interaccionales del discurso ya sea la distribución de turnos, división en secuencias, etc. Cada día notamos más la gran necesidad de procesar datos de artículos científicos.

Este proceso conlleva mucho tiempo a los investigadores y especialistas sanitarios ya que requieren extraer datos y conceptos complejos para actualizar la información o para realizar sus propias investigaciones. Frente a este problema surge la práctica de las tecnologías del lenguaje humano la inteligencia artificial en el área de la inteligencia artificial para resumir textos. Destacando la clasificación más característica de dos enfoques para realizar un resumen, el enfoque extractivo visto en la sección 3.2 donde busca encontrar una frase o un conjunto de frases del texto que capturen su esencia como partes representativas del mismo. El otro enfoque es el abstractivo, también visto en la misma sección que el extractivo, con la diferencia que el abstractivo genera nuevas frases para formar parte o la mayor parte del resumen, esto quiere decir la implicación de una interpretación del texto original y un tipo de representación del mismo texto que permita formar un resumen, lo que lo hace más complejo que el método extractivo. En la misma sección 3.2 se menciona el enfoque abstractivo que trabaja mejor en resúmenes automáticos multi documento, pero según lo aprendido durante la elaboración de este trabajo con el enfoque extractivo en los resúmenes mono documentos ha sido la mejor elección. El enfoque escogido para nuestra herramienta se caracteriza por la simplicidad y capacidad de conservar la idea original del autor sin necesidad de hacer uso de una interpretación del texto que formará parte del resumen. Por estas razones, se prefirió el resumen automático extractivo mono documento.

Con el transcurso del tiempo dedicado en la elaboración de este trabajo, hemos podido alcanzar los objetivos marcados y poco a poco conseguir el objetivo principal de generar un resumen a partir de un documento en PDF como se puede observar en la sección 4. En el cuerpo del trabajo de este documento. Cabe destacar que nuestra herramienta no es perfecta. El corpus como prueba de entrada han sido cuatro artículos científicos de la plataforma ScienceDirect en PDF donde la herramienta Appenecum selecciona las oraciones como unidades textuales, y se realizó considerando la relevancia dentro del campo de la investigación sanitaria concretamente del covid, además se tuvo muy en cuenta la extracción de cada de los artículos con un formato de texto bien estructurado, que nos facilitó esa gran labor organizativa de la ya mencionada plataforma ScienceDirect. Por el contrario, la

identificación de los diferentes fragmentos que conforman el texto resultaría altamente compleja, por lo que degradaría considerablemente los resultados del proceso de generación del resumen. Es importante destacar que cada artículo científico fue sometido a un adecuado proceso de preprocesamiento antes de su análisis, es decir se le aplicaron al texto una serie de filtros para limpiar y preparar el texto, que descomponemos en oraciones y después en palabras para ordenarlas, definir el tamaño del resumen a generar (generalmente se suele expresar como un porcentaje del tamaño del documento en cuestión) y escoger las más puntuadas. Las métricas Rouge utilizadas en este trabajo para medir el rendimiento de nuestra herramienta consiste en utilizar el abstract del artículo científico original y hacer una comparativa con el resumen generado por nuestra herramienta, prestando mucha atención en que tengan la misma longitud de frases, es decir ambos resúmenes deben tener el mismo número de oraciones para que Rouge y sus distintas métricas funcionen correctamente. Como se ha podido observar en la sección 5.3 que alcanza entre un 20 y 30% rendimiento en la extracción del documento original dando unos resultados efectivos.

7.2 Trabajos futuros y líneas de investigaciones futuras

Las herramientas del PLN pueden resultar ser un motor de recuperación económica. Después de la pandemia nos hemos visto en la obligación de reinventarnos para compensar el déficit económico por el que estamos atravesando y por ello, sabiendo de las capacidades que tienen estas herramientas para motivar y emprender nuevas acciones de cambio ya sean empresas privada o públicas.

Una de las líneas a mejorar que se consideran más prioritarias es la generalización de la entrada al sistema, es decir, diseñar un algoritmo mediante inteligencia artificial capaz de detectar las palabras clave del texto que se le introduzca a la aplicación para que la eficiencia del programa no sea dependiente de la temática del artículo.

La salida de nuestro resumen pondera la relevancia de las oraciones del documento y ofrece como resultado un subconjunto formado por las más representativas. No obstante, queda por definir un aspecto subjetivo por el resumen estando relacionado según el criterio del lector. Oraciones que pueden ser significativas para una persona, no pueden serlo de la misma forma para otra. Es aquí donde surge una posible línea de investigación de como aprender este criterio mencionado. Un estudio realizado en la universidad nacional de la plata por Julieta Pilar [39], demuestra que con un modelo predictivo denominado combinador lineal posee ciertas ventajas midiendo distintas alternativas creadas a partir de este combinador lineal y seleccionando distintas métricas como atributos de entrada, obteniendo unos resultados alentadores y confirmando lo factible que resulta esta herramienta porque predice las oraciones de manera similar a como lo haría un lector, sin embargo hay pocos estudios que estén relacionados y queda mucho aún por descubrir.

Buscando la unidad de texto del documento que se busca ponderar. Se pueden elegir secciones completas, párrafos, oraciones o palabras del texto original. En base a esta elección es que se podrá determinar un listado ordenado de unidades según cuán significativas sean para formar parte del resumen. En nuestro caso han sido las oraciones y de esas oraciones las palabras ponderadas, en cambio otra línea de investigación a corto plazo podría ponderarse secciones y de esas secciones párrafos. Dicho de esta forma parece un proceso sencillo. Sin embargo, este proceso cambia totalmente todo lo expuesto durante este trabajo.

En el campo de la inteligencia artificial y concretamente en el PLN la generación de resúmenes de forma automática un artículo científico extenso de más de 30 páginas poder resumirlo en poco más que una sola frase podría ayudar a la hora de seleccionar y leer artículos mucho más rápido.

La medición entre las métricas Rouge utilizadas y el tipo de documento o temática del corpus de los artículos empleados disponen de una estructura organizativa bien establecida por la plataforma ScienDirect ya que son todos artículos científicos, pero esto cambia si se tratase de un cuento narrativo o infantil debido a que su composición estructural es totalmente distinta. Para las líneas de investigación futuras la creación de un algoritmo que permita seleccionar cualquier texto de cualquier tipo de texto puede llegar a ser muy relevante.

La modificación de un algoritmo que genere resúmenes adaptados al usuario podría ser interesante, pero para poder llevarse a cabo se precisa preparar un modelo del usuario, interpretado como una representación de sus intereses y preferencias. Para cualquier ámbito que no sea biomédico o científico, esta posibilidad puede resultar potencial, porque posibilita con un mismo documento que se puedan generar diferentes resúmenes dependiendo del perfil del usuario.

La veracidad de garantizar los nuevos resúmenes generados automáticamente por extracción debería estar acreditados por las mismas fuentes fiables en las que se introduce el original. Los artículos científicos que se publican en revistas importantes garantizan la veracidad y fiabilidad de la información extraídos ya que no se hacen comprobaciones adicionales al respecto, dando por correctos los datos que son publicados en artículos, confiando en la revisión de las editoriales. En el caso de información que provienen de otros medios, como pueden ser foros, blogs o redes sociales la veracidad no está garantizada y es necesario que permitiesen esta veracidad de la que hablamos o por lo menos que verificasen la información para obtener cierta credibilidad. Esta línea de investigación tendría un nivel elevado debido a que se trata de una línea de investigación novedosa ya que no he podido encontrar nada en relación y, por consiguiente, hay mucho desconocimiento en este campo.

8 REFERENCIAS

[1] Ministerio de Sanidad (2020). Información sobre el coronovirus: https://www.sanidad.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/ciudadania.htm (visitado por última vez 02/07/2023).

[2] Ministerio de Sanidad, servicios sociales e igualdad (2012). Sistema nacional de salud (SNS): https://www.sanidad.gob.es/organizacion/sns/docs/sns2012/SNS012_Espanol.pdf (visitado por última vez 02/07/2023).

[3] Revista TIA. Néstor Camilo Beltrán y Edda Camila Rodriguez Mojica (2021), Procesamiento del lenguaje natural (PLN) -GPT-3, y su aplicación en la Ingeniería de Software. Tecnol.Investig. AcademiaTIA,ISSN: 2344-8288,8(1), pp.18-37. Bogotá-Colombia. https://revistas.udistrital.edu.co/index.php/tia/article/view/17323/17210 (visitado por última vez 02/07/2023).

[4] Introducción a MEDLINE y a las búsquedas bibliográficas (II). D. Barroso Espadero, G. Orejón de Luna, M. Fernández Rodríguez (2004). Guía de uso de PubMed. http://repositori.uji.es/xmlui/bitstream/handle/10234/188626/BarrosoD_IntroduccionaMEDLINEyalasbusquedasbibliograficasIIGuiadeusodePubMed.pdf?sequence=1&isAllowed=y (visitado por última vez 02/07/2023).

[5] ELSEVIER (1997). Plataforma ScienceDirect: https://www.elsevier.com/solutions/sciencedirect (visitado por última vez 02/07/2023).

[6] Biblioteca Nacional de Medicina de EEUU (1998). Directora de NLM: Patricia Flatley Brennan. MedlinePlus: https://medlineplus.gov/spanish/acercade/ (visitado por última vez 02/07/2023).

[7] GitHub, plataforma en línea que sirve como repositorio de control de versiones basado en Git. Francesco Ronzano (2017). Herramienta TextDigester: https://github.com/fra82/textdigester (visitado por última vez 02/07/2023).

[8] Instituto de ingeniería del conocimiento (iic). Antonio Moreno (2018) Definición de PLN: https://www.iic.uam.es/author/antonio/ (visitado por última vez 02/07/2023).

[9] Revista de Filología de la Universidad de La Laguna, Nº24, págs. 77-90. Alfonso Corbacho Sánchez (2006). Textos, tipos de texto y textos especializados: https://dialnet.unirioja.es/servlet/articulo?codigo=2100070 (visitado por última vez 02/07/2023).

[10] Real Academia Española. Edición del Tricentenario (2022). Resumen y resumir. https://dle.rae.es/resumir#WFYe5Ne. (visitado por última vez 02/07/2023).

[11] Real Academia Española. Edición 22 (2001). Definición de texto: https://www.rae.es/drae2001/texto
(visitado por última vez 30/06/2023).

[12] Plataforma ResearchGate. Werlich, E. (1975). Typologie der Texte. Munich: https://www.researchgate.net/profile/Esperanza-Acin/publication/267376957_Curso_La_tipologia_textual_Cuatro_creditos/links/56eacc9e08ae2a58dc49a181/Curso-La-tipologia-textual-Cuatro-creditos.pdf
(visitado por última vez 02/07/2022).

[13] Association for Computational Linguistics. Hovy, E. y Chin-Yew Lin (1998). Automated Text Summarisation and the Summarist System. https://aclanthology.org/X98-1026.pdf (visitado por última vez 02/07/2023).

[14] Repositorio Institucional de la UA. Procesamiento del Lenguaje Natural, revista nº47, pp 107-115. Elena Lloret y Manuel Palomar (2011). Tipos de resúmenes: https://rua.ua.es/dspace/bitstream/10045/18518/1/PLN_47_11.pdf
(visitado por última vez 02/07/2023).

[15] Repositorio Institucional de la UCM. Laura Plaza (2008). Generación automática de resúmenes con apoyo en ontologías aplicada al dominio biomédico. https://eprints.ucm.es/id/eprint/10076/1/TesisMasterLauraPlazaMorales.pdf
(visitado por última vez 03/05/2023).

[16] Book Advances in Automatic Text Summarization. Barzilay y Elhadad (1997-1999). Técnicas discursivas: https://books.google.es/books?hl=en&lr=&id=YtUZQaKDmzEC&oi=fnd&pg=PA111&dq=Barzilay,+R.,+Elhadad,+M.:+Using+Lexical+Chains+for+Text+Summarization.+In:+Mani,+I.,+Maybury,+M.T.+(eds.)+Advances+in+Automatic+Text+Summarization,+pp.+111-121.+MIT+Press,+Cambridge+(1999).&ots=ZqvAwtE29H&sig=3QKiVvuYV8vxNmmcsKNuo3CdRns&redir_esc=y#v=onepage&q&f=false
(visitado por última vez 02/07/2023).

[17] Plataforma ResearchGate. Mann y Thompson (1988). Estructura retórica: https://www.researchgate.net/publication/236672863_Estructura_retorica_y_estructura_generica_del_Abstract_del_articulo_de_investigacion_cientifica/link/00463518d7f3189c4d000000/download
(visitado por última vez 02/07/2023).

[18] En Proceedings of Human Language Technology Conference and Empirical Methods in Natural Language Processing (HLT/EMNLP). Otterbacher, J., Erkan, G. y Radev, D. (2005). Técnicas basadas en grafos: Using random walks for question-focused sentence retrieval: https://aclanthology.org/H05-1115.pdf (visitado por última vez 02/07/2023).

[19] Biblioteca digital ACM. Pedersen y Chen (1995). Generación de resúmenes abstractivos: https://dl.acm.org/doi/pdf/10.1145/215206.215333
(visitado por última vez 02/07/2023).

[20] Penn Engineering University of Pennsylvania. Informática y ciencias de la información (Departamento de la escuale de ingeniería y ciencias aplicadas). Kupic, Pedersen y Chen, (1995). Técnicas de generación de lenguaje: https://www.cis.upenn.edu/~nenkova/Courses/cis430/trainableSummarizer.pdf (visitado por última vez 03/07/2023).

[21] Repositorio institucional de la UA. Procesamiento del Lenguaje Natural, núm.35, pp.245-252. Fernando Martínez y Miguel Angel García (2005). Propuesta de un modelo basado en reglas y aprendizaje automático: https://rua.ua.es/dspace/bitstream/10045/1321/1/PLN_35_30.pdf (Visitado por última vez 30/06/2023)

[22] Repositorio institucional de la UPV. Jesús Tomás y Francisco Casacuberta, (2003). Traducción automática de textos entre lenguas similares utilizando métodos estadísticos: http://personales.upv.es/~jtomas/articulos/tesis.pdf (Visitado por última vez 30/06/2023)

[23] Repositorio Zaguan de documentos digitales de la Universidad de Zaragoza. Miguel Lahoz, Rafel Del Hoyo y Nicolás Medrano, (2019). Estudio de nuevos modelos de Deep Learning para el nálisis y comprensión de grandes cantidades de datos: https://zaguan.unizar.es/record/87391/files/TAZ-TFG-2019-3115.pdf?version=1 (visitado por última vez 03/07/2023).

[24] Repositorio en línea ACL Anthology. Tipos de evaluación Mani, I. (2001). Summarization Evaluation: An Overview. En Proceedings of the NAACL 2001 Workshop on Automatic Summarization. https://aclanthology.org/J02-4001.pdf (visitado por última vez 03/07/2023).

[25] Repositorio Institucional de la UNM. J. M. Adam (1992). Textos expositivos, pp. 21-24: https://lecturayescrituraunrn.files.wordpress.com/2013/08/unidad-3-compl-adam.pdf
(visitado por última vez 03/07/2023).

[26] Manual de Python (2001). Python Software Foundation Version 2: https://docs.python.org/3/tutorial/index.html
(visitado por última vez 03/07/2023).

[27] Libro Python 3. Sebastien Chazallet, (2016). Los fundamentos del lenguaje, 2ªedición: https://books.google.es/books?hl=es&lr=&id=KRYyvKmZvpwC&oi=fnd&pg=PA5&dq=S%C3%A9bastien+chazallet+en+2016+python&ots=UG8bGsibO3&sig=x5e3Bico_G5NjhyGqo5nIUcWDRA#v=onepage&q=S%C3%A9bastien%20chazallet%20en%202016%20python&f=false (visitado por última vez 05/07/22).

[28] GitHub, plataforma en línea que cuenta con un repositorio de control de versiones basado en Git. Autores: comunidad de Git (2015). Documentación de jupyter notebook: https://docs.jupyter.org/en/latest/ (visitado por última vez 03/07/2023).

[29] Microsoft. Fernando Pérez (2001). Definición de Kernels: https://docs.microsoft.com/es-es/azure/hdinsight/spark/apache-spark-jupyter-notebook-kernels (visitado por última vez 03/07/2023).

[30] Repositorio Institucional de la UA. Procesamiento del Lenguaje Natural, revista nº47, pp 107-115. Elena Lloret y Manuel Palomar (2011). COMPENDIUM_Una_herramienta_de_generacion_de_resmenes_modular (visitado por última 03/07/2023).

[31] Plataforma en línea Kaggle. Propietario: Google (2017). Recursos de datos científicos: https://www.kaggle.com/ (visitado por última 03/07/2023).

[32] Manual de Python (2001). Python Software Foundation Version 2. Librería de paython definición de contenedor: https://docs.python.org/dev/library/collections.abc.html#module-collections.abc (visitado por última vez 03/07/2023).

[33] Biblioteca de procesamiento de imágenes en Python. Documentación oficial de Pillow. Jeffrey A. Clark (2010). Módulo ImageTk de la librería Pillow: https://pillow.readthedocs.io/en/stable/reference/ImageTk.html (visitado por última 03/07/2023).

[34] Manual de Python (2001). Python Software Foundation Version 2. Widgets: https://docs.python.org/es/3/library/tkinter.ttk.html#widget (visitado por última vez 05/07/2023).

[35] Plataforma en la nube ActiveState. David Ascher y Greg Stein (1997). Administración y distribución de lenguajes de programación: https://www.activestate.com/ (visitado por última vez 03/07/2023).

[36] Manual de Python (2001). Python Software Foundation Version 2. Librería datetime: https://docs.python.org/es/3/library/datetime.html (visitado por última 25/05/2023).

[37] Repositorio en línea ACL. Lin, Chin-Yew, (2004). Rouge: Un paquete para la evaluación automática de resúmenes: https://www.aclweb.org/anthology/W04-1013.pdf (visitado por última vez 03/07/2023).

[38] Repositorio en línea del modelo de lenguaje. Teun A. Van Dijk, (2003). Propiedades del discurso: http://www.discursos.org/Art/La%20multidisciplinariedad.pdf (visitado por última vez 03/07/2023).

[39] Repositorio de la UNLP.Julieta Pilar Corvi, (2019). Un análisis comparativo de técnicas de puntuación: http://sedici.unlp.edu.ar/bitstream/handle/10915/81471/Tesis.pdf-PDFA.pdf?sequence=1&isAllowed=y (visitado por última vez 03/07/2023).