UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

"IDENTIFICACIÓN DE REQUISITOS FUNCIONALES MEDIANTE LA GENERACIÓN AUTOMÁTICA DE RESÚMENES A PARTIR DE LISTAS DE REQUISITOS PRELIMINARES"

TESIS

PARA OBTENER EL TÍTULO DE

MAESTRA EN INGENIERÍA EN SOFTWARE

PRESENTA

L. I. CELIFLORA DÍAZ JIMÉNEZ

DIRECTORA DE TESIS

DRA. CARLA LENINCA PACHECO AGÜERO

CODIRECTOR DE TESIS

DR. CHRISTIAN EDUARDO MILLÁN HERNÁNDEZ

HUAJUAPAN DE LEÓN, OAX. MARZO 2026

Dedicatoria

A mi madre y a mis hermanos, por su dedicación y apoyo incondicional, lo que me permitió llegar hasta aquí.

A la memoria de mi padre, que fue mi mayor inspiración para poder culminar mis estudios académicos.

"Le he pedido al cielo que te diga que lo he logrado, ya que él puede verte cada día y yo solo en mis sueños".

Agradecimientos

A la Dra. Carla Leninca Pacheco Agüero, mi directora de tesis, por el valioso tiempo que me brindó a lo largo de este proceso. Agradezco todo el apoyo que me ha brindado, su disposición para orientarme y por compartir conmigo su amplio conocimiento.

Al Dr. Christian Eduardo Millán Hernández, mi codirector de tesis, por su apoyo durante el desarrollo de este trabajo de tesis, el cual ha sido esencial para alcanzar este logro.

A las y los integrantes del Comité revisor de este trabajo de tesis, por el tiempo y el esfuerzo invertidos en la revisión y evaluación de este trabajo de investigación.

A todas y todos los profesores, compañeros y amigas que conocí a lo largo del curso de esta maestría, por el valor de sus conocimientos y experiencias, y por la ayuda y la amistad que me brindaron en cada etapa.

A mi mamá Catalina Jiménez Gallardo, por su constante aliento durante mi formación académica.

A mis hermanos Wille, Jacobo, Rigo, Vero y Wene que a pesar de las distancias siempre han estado ahí. Su amistad y aliento han sido mi motivación para no rendirme.

A Luis, por su dedicación a lo largo de este proceso, su confianza en mí, su paciencia y cariño han sido una fuente de gran fortaleza para superar los desafíos y seguir adelante.

Finalmente, a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI), antes Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), por brindarme el apoyo económico durante todo el periodo que comprendió la realización de mis estudios de maestría.

Índice

Lista de tablas

Lista de figuras

Resumen

La Ingeniería de *Software*, pese a contar con más de seis décadas de desarrollo, continúa enfrentando problemas que afectan la culminación exitosa de sus proyectos. Entre los más relevantes destacan los retrasos en las entregas, las desviaciones presupuestales y, especialmente, la incapacidad de satisfacer las necesidades de los *stakeholders*. Esta última situación constituye una causa central por la cual los productos generados no cumplen con los objetivos para los que fueron diseñados. Una razón fundamental de este problema radica en la idea equivocada de que comprender, identificar y documentar las necesidades y expectativas de los *stakeholders* (i.e., requisitos preliminares[1]), es una tarea sencilla dentro de la Ingeniería de Requisitos, cuando en realidad representa un proceso complejo y crucial para el éxito del proyecto. En este sentido, la elicitación de requisitos adquiere una importancia decisiva, ya que la calidad con la que se realiza determina la correcta definición del *software* a desarrollar. Si los requisitos no son precisos, completos y estructurados adecuadamente, aumenta el riesgo de que el producto final no satisfaga las expectativas previstas, afectando su utilidad. Sin embargo, este proceso depende en gran medida del uso del lenguaje natural, el cual introduce desafíos como ambigüedad, vaguedad e inconsistencias que pueden conducir a malentendidos entre los *stakeholders* y el equipo de desarrollo.

Es por lo que, la presente tesis propone utilizar técnicas de Procesamiento de Lenguaje Natural (NLP, por sus siglas en inglés), disciplina que combina Inteligencia Artificial (IA) y lingüística computacional, para identificar y seleccionar requisitos funcionales preliminares a partir de resúmenes generados automáticamente a partir de listas de requisitos preliminares, con lo que se busca mejorar la precisión de este proceso.

[1]Un requisito preliminar incluye, por ejemplo, las necesidades de los *stakeholders*, los conceptos del sistema, las expectativas de los usuarios y el entorno del sistema. Los requisitos preliminares son descritos usualmente mediante expresiones de NL (Hayes et al. 2008). Además, un requisito preliminar es un "requisito" antes de su transformación en un documento de especificación, ya que los requisitos son especificaciones técnicas formales en el espacio de soluciones (Ravichandar et al., 2007).

1. Introducción

Desde los años 60's se han identificado diversos problemas relacionados con el desarrollo de *software*, como, por ejemplo: la entrega de productos deficientes e incompletos, plazos de entrega incumplidos, presupuestos fuera del alcance, falta de confiabilidad y/o los altos costos de la modificación y el mantenimiento (Brooks y Bullet, 1987; Colburn et al., 2008; Giró et al., 2016). A estos problemas, se les conoció en aquel entonces como "Crisis del *Software*", situación que condujo al surgimiento de la Ingeniería de *Software* (IS) como una opción para resolverlos.

De acuerdo con Sommerville (2010), Pantaleo y Rinaudo (2015), Delgado Olivera y Díaz Alonso (2021) el término IS se mencionó por primera vez en 1969 en la Conferencia de la Organización del Tratado del Atlántico Norte (OTAN) donde se establecieron las bases de esta disciplina. En dicha conferencia se abordaron problemas que en la actualidad siguen presentes en la industria, tales como el desarrollo de *software* que no satisface las necesidades de los *stakeholders*[2], el uso de prácticas deficientes de desarrollo, la mala gestión de los requisitos, la desviación del cronograma –que retrasa la terminación del proyecto de *software*–, y los altos costos de desarrollo (Hughes et al., 2017; Nizam, 2022). Así mismo, el *Chaos Report*[3] del *Standish Group Inc*. Afirma que en el año 2020 sólo el 31% de los proyectos relacionados con el desarrollo de *software* fueron exitosos, el 50% presentó problemas y desafíos que conllevaron a retrasos y el 19% fue un fracaso, lo que indica que todavía queda mucho por hacer (Standish Group, 2020).

En este sentido, múltiples autores consideraron que la evolución de la IS podía contrarrestar los efectos negativos de estos problemas, puesto que se trata de una disciplina que se enfoca en el desarrollo sistemático y ordenado del *software* mediante el establecimiento de procesos que van desde la elicitación de los requisitos hasta el mantenimiento de este; sin embargo, no ha sido así, ya que actualmente se siguen presentando los problemas que le dieron origen (IEEE Standards Committee, 1990; Sommerville y Velázquez, 2011; Pressman y Maxim, 2019).

---

[2] Un *stakeholder* o parte interesada, es cualquier persona, organización o documento que debe ser tomado en cuenta para el desarrollo de un *software*: usuarios finales, ingenieros(as) de *software*, gerentes comerciales, personal de soporte, etc. (Pressman y Maxim, 2019).

[3] El *Chaos Report* o Reporte del Caos, es uno de los informes más consultados durante la realización de investigaciones que analizan el comportamiento de la IS moderna. En este sentido, el *Standish Group* recoge información cada cuatro años en la que resume las causas de éxito y fracaso de los proyectos de Tecnologías de Información en la industria de *software* de los Estados Unidos de Norteamérica (60%), Europa (25%), y el resto del mundo (15%) (Standish Group, 2020).

Para Salazar et al. (2011), Sommerville y Velázquez (2011), Pressman y Maxim (2019), y Briano (2023), la IS define modelos o ciclos de vida que a menudo establecen cinco procesos para cumplir con esta tarea:

- Definición de requisitos: Proceso en el que se debe entender y comprender el dominio del problema para así, definir, especificar y priorizar las funcionalidades y restricciones del *software*.
- Diseño: Proceso que define la estructura del *software* y de los datos, así como los modelos e interfaces entre los diferentes componentes de éste. Las actividades dentro de este proceso varían dependiendo del tipo de *software* a desarrollar.
- Implementación: Proceso donde se codifica el *software* para obtener un programa ejecutable.
- Pruebas: Proceso que comprueba que el *software* satisfaga las expectativas de los *stakeholders* y que se cumplen los requisitos establecidos en la especificación. Además, se prueba el *software* para detectar fallos y errores.
- Mantenimiento: Es el proceso donde se realizan cambios al *software* una vez que se ha entregado al usuario. Estos cambios a menudo provienen de mejoras y/o reparación de problemas en el diseño o codificación.

Durante mucho tiempo la IS se enfocó en crear lenguajes de programación, sistemas expertos, entornos de programación, enfoques incrementales, y así como en la creación rápida de prototipos, a pesar de que uno de sus principales desafíos es lograr que los profesionales del *software* desarrollen productos que cumplan correctamente con las necesidades de los *stakeholders* (Giró et al., 2016; Broy, 2018). Crear un producto de *software* representa un desafío atractivo para la mayoría de los desarrolladores, pero, desafortunadamente, muchos de éstos prefieren comenzar a programar antes de tener una idea clara de lo que el usuario realmente necesita. Es decir, no se detienen a comprender las necesidades, deseos y limitaciones de los *stakeholders*, lo que a menudo conduce al desarrollo de un producto fallido debido a la elicitación de requisitos incorrectos (Pressman y Maxim, 2019).

En este contexto surge la Ingeniería de Requisitos (IR), como una disciplina que trata de establecer lo que se pretende desarrollar y que, de acuerdo con Paetsch et al. (2003), Westfall (2005), Sommerville y Velázquez, (2011) y, Bennaceur et al. (2019) se puede definir como el área de la IS que identifica y define el comportamiento de un *software*, analiza y describe las especificaciones detalladas de lo que éste debe hacer mediante la documentación y gestión de los requisitos a lo largo del ciclo de vida de desarrollo con el fin de garantizar que éstos sean correctos, consistentes y completos. Sommerville (2019) y Guelfi (2022) mencionan que la IR es crucial para el desarrollo de *software*, independientemente del ciclo de vida, puesto que es donde se lleva a cabo el establecimiento de los requisitos de los *stakeholders*, aspecto que determina el éxito o fracaso de un proyecto. Para Sawyer y Kotonya (2001), Bourque y Fairley (2014) y Barrientos Núñez y Carballo Muñoz (2021) la IR se compone de los siguientes procesos:

- Elicitación: Es donde mediante la aplicación de técnicas se obtienen los deseos y necesidades de los *stakeholders* sobre el *software* que se pretende desarrollar.
- Análisis: Con la información obtenida en el proceso anterior, se escriben los requisitos y se identifican posibles problemas entre ellos, asimismo se hace una clasificación de éstos para establecer prioridades de acuerdo con el objetivo del proyecto.
- Especificación: Se puntualizan de forma detallada los requisitos del *software* en un documento estándar.

- Validación y verificación: Se validan los requisitos para garantizar que describen adecuadamente lo que el *software* debe realizar, además se asegura de que los requisitos cumplan con las características de calidad del estándar ISO/IEC/IEEE 29148 (2018) (i.e., precisos, consistentes, completos, singulares, factibles, trazables, y verificables).
- Gestión: Mejora la administración de los requisitos y asegura que el *software* evolucione de acuerdo con las necesidades de los *stakeholders*.

Con base en lo anterior, se puede apreciar que la IR es un proceso importante dentro de la IS, puesto que asegura que el desarrollo de *software* cuente con requisitos bien definidos, lo que de acuerdo con Sommerville (2019) evitará errores que conduzcan a problemas en su diseño o implementación.

Ahora bien, dentro de la IR, la elicitación de requisitos es el proceso que se enfoca en entender y descubrir los deseos y necesidades de los *stakeholders*, antes de que éstas sean convertidas en requisitos. De acuerdo con Zowghi y Coulin, (2005) y Sharma y Pandey (2014), el proceso de elicitación de requisitos involucra actividades tales como la identificación de los *stakeholders*, la colaboración, negociación y comunicación con los *stakeholders*, el uso de múltiples técnicas para la obtención de sus deseos y necesidades (e.g., cuestionarios, encuestas, lluvia de ideas, entrevistas estructuradas y no estructuradas) y, su posterior refinamiento.

En este sentido, el proceso de elicitación de requisitos es dónde se deben determinar las necesidades de los *stakeholders*, aspecto crítico en el desarrollo de *software* ya que, de no hacerse correctamente, éste no cumplirá con el objetivo para el cuál fue creado (Christel y Kang, 1992; Wong y Mauricio, 2019). Por lo que, como primer paso, se deben identificar correcta y completamente todos los *stakeholders* del proyecto, establecer una comunicación activa y constante con todos ello(a)s, y utilizar técnicas de elicitación que permitan obtener sus deseos y necesidades (Coughlan et al., 2003; Dar et al., 2018; Wong y Mauricio, 2019).

De acuerdo con Holste y Fields (2010), Suppiah y Singh Sandhu (2011), Kumari y Pillai (2013), Sharma y Pandey (2014), Dar et al., (2018) y Dar et al., (2022) algunos de los problemas que pueden presentarse durante la elicitación de los requisitos son los siguientes:

- Ambigüedad en los requisitos, dado que muy frecuentemente éstos se escriben en Lenguaje Natural (NL, por sus siglas en inglés) se puede originar que las palabras tengan diferentes significados dependiendo de su contexto social.
- Mala definición de los requisitos, lo que ocasiona que el proyecto no cumpla con el objetivo para el cual fue creado, falle y por consiguiente se eleve el presupuesto inicial.
- Volatilidad de los requisitos (i.e., que éstos tienden a cambiar continuamente con el tiempo).
- Malentendidos entre los *stakeholders*, lo que ocasiona que se obtengan requisitos inconsistentes.
- Diferencias de idioma, lo que conlleva a que se generen múltiples interpretaciones de una sola idea y, por ende, que se dificulte expresar su significado en una oración.
- Conocimiento tácito, es aquel que se encuentra en un nivel inconsciente. Es personal ya que está enraizado en la experiencia y los valores de un individuo, específico al contexto y difícil de formalizar y comunicar (e.g., habilidades, experiencias, procesos no documentados, instintos, entre otros).

Como puede verse, el proceso de elicitación de requisitos impacta en todo el desarrollo de *software*, ya que el obtener requisitos incorrectos, incompletos y ambiguos afectará la calidad, el costo y el tiempo de entrega (Christel y Kang, 1992; Wong y Mauricio, 2019). Por lo tanto, la elicitación de requisitos es un proceso crítico donde se debe invertir tiempo y esfuerzo considerable para realizarlo de forma correcta, puesto que esto garantizará una base sólida para los siguientes procesos de la IR. Aunado a lo anterior, se generará un impacto positivo en el tiempo de desarrollo del proyecto, reduciendo así los costos, mejorando la satisfacción del cliente y, por ende, aumentando la calidad del *software* (Dar, 2020; Rodríguez-Silva, 2022).

Desde inicios de década de los 80's se han intentado desarrollar herramientas y métodos de NLP, para detectar ambigüedades en los requisitos, y generar modelos y herramientas que permitan clasificarlos (Zhao et al., 2021). En este sentido, el NLP es la capacidad de una máquina para comprender y trabajar con el lenguaje humano. Para lograr esta comunicación es necesario desarrollar modelos y técnicas computacionales con el objetivo de comprender y entender el lenguaje humano (Moreira et al., 2021; Zhao et al., 2021). En general, el NLP se está convirtiendo en un complemento importante para la IS, específicamente en el contexto de la IR, puesto que a menudo los requisitos se redactan en NL con diferentes grados de formalidad, que van desde una declaración o historias de usuario hasta lenguajes no estructurados (Dalpiaz et al., 2022).

Para abordar los problemas que se presentan en la elicitación de requisitos se han llevado a cabo diversas investigaciones sobre el uso de técnicas de NLP, con el objetivo de detectar y evitar requisitos ambiguos, mal escritos e incompletos. A continuación, se describen algunas de éstas:

- Ferrari y Esuli (2019), por ejemplo, propusieron un método basado en NLP para detectar ambigüedades lingüísticas durante la elicitación de requisitos. Debido a que los *stakeholders* poseen diferentes grados y habilidades para comunicarse, es común que durante la elicitación se presenten ambigüedades debido a que los *stakeholders* poseen diferentes dominios de conocimiento, lo que puede generar frustración y desconfianza durante la elicitación de los requisitos que, como consecuencia, terminan influyendo negativamente en las etapas posteriores del desarrollo del *software*. El método propuesto permite clasificar términos ambiguos entre los diferentes dominios y clasificar su grado de ambigüedad, para lo cual se utilizan modelos de lenguaje específico de dominio (i.e., se crea un dominio por cada *stakeholder*). En la investigación se evaluaron siete escenarios hipotéticos de elicitación de requisitos con cinco modelos de dominio. Las clasificaciones obtenidas bajo este enfoque alcanzaron un 88% máximo de Tau de Kendall[4]; sin embargo, en términos de desempeño el enfoque no tuvo éxito. Aunado a esto, se determinó que las principales imprecisiones se dieron entre el conocimiento del dominio y la ambigüedad del NL. A partir de los resultados obtenidos, los investigadores afirmaron que el enfoque propuesto era preciso en situaciones donde el número de dominios era limitado, pero no cuando existen muchos dominios involucrados.

- En la investigación realizada por Deshpande et al. (2019), los autores proponen un método para extraer dependencias entre requisitos, utilizando NLP y Aprendizaje Débilmente Supervisado (WSL, por sus siglas en inglés) en dos etapas, puesto que dichas dependencias afectan directamente la toma de decisiones en ciclo de desarrollo de *software*, ya que pueden generar errores, problemas cognitivos, además de que son computacionalmente complejas

[4] Tau de Kendall es una medida de correlación de rangos que mide la fuerza de la relación entre dos variables de nivel ordinal (Dehling, 2017).

debido que la mayoría de los requisitos están documentados en NL. En la primera etapa se identificaron el tipo de dependencia (i.e., independientes y dependientes), y en la segunda, etapa se analizó el tipo de dependencia para realizar una evaluación con el conjunto de datos PURE utilizando tres algoritmos de Aprendizaje Automático (ML, por sus siglas en inglés): bosques aleatorios, Máquinas de Vectores de Soporte (SVM, por sus siglas en inglés) y clasificador Bayesiano, aplicando validación cruzada. Inicialmente, los datos originales obtuvieron una precisión de 0.7 y una medida-F de 0.7. Posteriormente, tras aplicar WSL, se observó una mejora en el desempeño, alcanzando una precisión de 0.81 y una medida-F de 0.87. Los resultados demostraron que aplicando el WSL se pueden generar pseudo anotaciones más precisas, mejorando así la toma de decisiones.

- Kengphanphanit y Muenchaisri (2020) propusieron elicitar automáticamente nuevos requisitos de *software*, a partir de aplicaciones (*Apps*) en fase de mantenimiento, tomando como base los comentarios de los usuarios. Las plataformas que utilizaron fueron *App Store* y *Play Store,* así como las redes sociales *Twitter* (ahora *X*) y *Facebook*. El enfoque presentado en este trabajo consta de cinco pasos: primero se recolectan y obtienen comentarios en *Twitter* y *Facebook*; después se limpian los textos, eliminando palabras sin sentido; posteriormente, se clasifican los comentarios en requisitos o no requisitos utilizando un clasificador bayesiano entrenado previamente; luego se aplica un proceso para eliminar texto duplicado, y por último se visualizan los resultados en *Power BI* para que estos puedan ser visualizados por lo(a)s ingeniero(a)s y responsables del producto. En este trabajo se analizaron 600 comentarios, de los cuales 193 fueron clasificados como requisitos y 407 como no requisitos. En cuanto al desempeño del clasificador Bayesiano, este alcanzó una precisión de 51.72, una recuperación de 81.08 y una medida-F de 63.15, indicando que recupera un número considerable de falsos positivos, es decir, predice un comentario como requisito, pero al final este no es un requisito. Finalmente, los autores destacan que mediante *web scraping* se puede obtener retroalimentación en tiempo real; sin embargo, la precisión del modelo refleja la necesidad de mejorar el balance de datos y utilizar clasificadores más avanzados.

- Hey et al. (2020) en su trabajo de investigación presentaron un Clasificador de Requisito Funcional (FR, por sus siglas en inglés) y Requisito No Funcional (NFR, por sus siglas en inglés) utilizando BERT (NoRBERT), una versión mejorada de Representaciones Bidireccionales de Codificadores a partir de Transformadores (BERT, por sus siglas en inglés) que es un modelo de NLP basado en transferencia por aprendizaje. La propuesta que hacen los autores consistió en ajustar BERT (*fine-tuning*) para clasificar requisitos según su función, datos y comportamiento. Para este trabajo se utilizó el conjunto de datos: PROMISE NFR. Los resultados demuestran que NoRBERT obtuvo una puntuación promedio del 87% en cuanto a la medida-F, superando a enfoques de clasificación de subclases en NFR. Además, alcanzó una puntuación del 92% cuando se utilizó el conjunto de datos re-etiquetado. Estos resultados indican que NoRBERT mejora la clasificación de requisitos, especialmente en el contexto de FR y NFR.

- En la investigación de Lafi et al. (2021) se propuso un método automático para eficientar la elicitación de los requisitos representativos a partir de respuestas abiertas de múltiples *stakeholders* usando técnicas de NLP, debido a que este proceso suele complicarse cuando existe una lista extensa de *stakeholders* que cuentan con sus propias necesidades. La propuesta consta de 4 fases principales: 1) la recolección de requisitos; 2) el preprocesamiento de

respuestas utilizando la *tokenización*[5], eliminación de palabras vacías y la *lematización*[6]; 3) selección de requisitos representativos mediante la generación de una lista de palabras en la que se presentan todas las palabras distintas y se crea una matriz para calcular el valor de puntuación por cada respuesta; 4) validación donde se comparan los resultados obtenidos con resultados de codificadores humanos. Esta investigación propuso tres mejoras puntuales al proceso de elicitación: 1) disminuir la carga de trabajo del(la) ingeniero(a) de requisitos al elicitar los requisitos; 2) reducción del tiempo utilizado en la elicitación; y 3) destacar la importancia del uso de técnicas de minería de datos en la IR. Para la experimentación, los autores generaron sus propios conjuntos de datos debido a la falta de repositorios de requisitos, utilizando preguntas sobre un sistema de registro universitario para obtener las respuestas de lo(a)s estudiantes. Al realizar la validación, el enfoque seleccionó las respuestas que mejor representaban a los *stakeholders*, obteniendo un 80% (exactitud) al utilizar la métrica de similitud de coseno; sin embargo, respecto al tiempo de ejecución, este aumentó con el tamaño de los datos de entrada. Mientras que los codificadores humanos verificaron que las respuestas seleccionadas por la propuesta fueran las correctas. De esta manera se demostró que este enfoque redujo tanto el esfuerzo humano como el tiempo dedicado a la elicitación de los requisitos, además de que se obtuvo un nivel adecuado de exactitud.

- La investigación de Calle Gallego y Zapata Jaramillo (2023) sigue el uso de las técnicas del NLP para la IR (NLP4RE, por sus siglas en inglés) para presentar un Modelo basado en Preguntas y Respuestas para la Elicitación de Requisitos (QUARE, por sus siglas en inglés). Considerando que las propuestas existentes basadas en NLP únicamente se centran en dominios específicos y aún no alcanzan la comprensión de diversos estilos de redacción de los requisitos, QUARE incorporó una meta ontología para facilitar la generación de preguntas durante la elicitación de los requisitos con el fin de obtener una estructura inicial de cualquier dominio de *software*. Además, dicho modelo incluyó un sistema de extracción de relaciones y reconocimiento de entidades nombradas que se centra en la elicitación de requisitos para permitir que los analistas usen múltiples estilos de redacción de los requisitos. Este enfoque se puso a prueba sobre un conjunto de documentos de requisitos reales de distintos dominios de *software* utilizando el conjunto de datos PURE como base para la validación y los resultados preliminares indicaron que el modelo era una propuesta innovadora, puesto que superó a algunas propuestas, como el enfoque basado en consultas, con un promedio de 0.78. Por último, se demostró que QUARE puede mejorar el proceso de elicitación de requisitos.

Como se puede apreciar en los trabajos descritos anteriormente, aunque se han propuesto diferentes herramientas, técnicas y métodos para resolver algunos problemas de la IR respecto al uso inherente del NL, existe un área de oportunidad dado que la mayoría de las soluciones propuestas no tienen en cuenta técnicas avanzadas del NLP, lo que demuestra que no se ha aprovechado todo el potencial que está área puede proporcionar (Zhao et al. 2021). Esta tesis de investigación pretende

[5] La *"tokenización"* consiste en dividir un documento de texto, primero en párrafos y luego en elementos más pequeños denominados *tokens*, para posteriormente aplicar herramientas del NLP como el etiquetador de categorías gramaticales o un analizador morfológico (Orquín, 2009, Talele y Phalnikar, 2021).

[6] La *"lematización"* es una tarea del NLP que sirve para agrupar palabras derivadas que comparten una raíz, es decir, que están formadas por la misma palabra primitiva, de manera que se pueda reducir el número de palabras a clasificar (Akhmetov et al., 2020; Luque García, 2023).

aprovechar las capacidades del NLP para identificar y clasificar FR a partir de resúmenes generados de manera automática a partir de listas de requisitos preliminares de *software*.

1.1. Delimitaciones de la tesis

Este trabajo de investigación cubrirá los siguientes puntos:

- La solución propuesta se enfocará únicamente en la etapa de elicitación de requisitos.
- En este trabajo de investigación solo se identificarán requisitos funcionales.
- El método propuesto en esta investigación para generar resúmenes utilizará las listas de requisitos preliminares, sin descartar que éstas pudiesen tener ambigüedades, inconsistencias o falta de singularidad.
- La evaluación empírica se realizará mediante el desarrollo de un caso de estudio en una pequeña organización desarrolladora de *software*.

1.2. Hipótesis

De acuerdo con la problemática presentada anteriormente, se define la siguiente la hipótesis de investigación para llevar a cabo este trabajo de tesis:

"A través de la generación automática de resúmenes se mejorará la identificación de requisitos funcionales en la elicitación de requisitos".

1.3. Objetivos del trabajo

Considerando lo anteriormente expuesto, este proyecto de tesis se desarrollará en el marco del siguiente objetivo general y los objetivos específicos definidos posteriormente:

1.3.1. Objetivo general

"Identificar requisitos funcionales a partir de resúmenes generados de manera automática desde listas de requisitos preliminares de *software*".

1.3.2. Objetivos específicos

Para lograr el objetivo general se realizarán los siguientes objetivos específicos:

1.- Realizar una búsqueda en la literatura pertinente de NLP sobre los métodos existentes para la generación automática de resúmenes y su uso en la identificación de requisitos funcionales en la elicitación de requisitos.

2.- Proponer un método para la generación de resúmenes con NLP para detectar palabras claves en la elicitación de requisitos funcionales.

3.- Implementar una herramienta de *software* para validar el método propuesto.

4.- Evaluar empíricamente la herramienta propuesta mediante un caso de estudio.

1.4. Aproximación a la solución

Tomando en cuenta que el resultado final de la etapa de elicitación de requisitos es la lista de deseos y necesidades de los *stakeholders*, se propone un método que parte de dicha lista para generar resúmenes automáticos empleando técnicas del NLP. Este enfoque permite sintetizar la información esencial contenida en los requisitos preliminares, facilitando así la identificación y extracción de los FRs.

La utilización de NLP en este contexto tiene el objetivo el extraer, de manera eficiente y precisa, los elementos clave que definen el comportamiento o funcionamiento esperado del *software* por parte de los usuarios y demás *stakeholders*. De este modo, el proceso no solo reducirá la carga de información a analizar, sino que también contribuirá a una mejor estructuración de los requisitos, lo que resulta fundamental en las fases posteriores del desarrollo del *software*.

De acuerdo con Bijal et al. (2017) el resumen de un texto se genera a través del proceso de extraer de forma coherente y precisa la información esencial del mismo, reduciendo su longitud, pero conservando la información importante del documento. Un resumen facilita, al usuario, encontrar información sintetizada e importante en una gran cantidad de datos. Para Gudivada (2018), Batra et al. (2020) y Widyassari et al. (2022) la generación de resúmenes de textos es una tarea del NLP que se puede clasificar en dos enfoques: extractivo y abstractivo. El primer enfoque elige palabras, oraciones y párrafos del documento original de acuerdo con su importancia en el texto y posteriormente concatena estas partes para formar el resumen. Mientras que el segundo enfoque trata de replicar la forma en que las personas generan un resumen a partir de un documento, es decir primero se comprende el texto y luego se genera el resumen con nuevas palabras u oraciones, preservando la esencia del texto original, pero expresado de manera única dependiendo de la perspectiva de la persona.

El-Kassas et al. (2021) indican que el resumen automático de un texto consta principalmente de las siguientes tareas:

1. Preprocesamiento: Se genera una representación estructurada del texto original utilizando segmentación de oraciones, *"tokenización"* de palabras, eliminación de palabras vacías, Etiquetado de Parte del Discurso (POS, por sus siglas en inglés) y la derivación.
2. Procesamiento: Utiliza uno de los dos enfoques para la generación de resúmenes de texto (extractivo o abstractivo) y técnicas de análisis para sintetizar documentos (e.g modelos de gráficos de texto).
3. Posprocesamiento: Resuelve problemas que se pueden encontrar al generar oraciones abreviadas, tales como la resolución de anáforas[7] y el reordenamiento de oraciones seleccionadas antes de generar el resumen.

Una vez generado el resumen, este debe ser evaluado. En este sentido, Awasthi et al. (2021) mencionan que dicha evaluación puede ser de forma automática o manual (hecha por una persona), siendo así, la evaluación automática una opción más factible, puesto que es más rápida. Sin embargo, es importante mencionar que, aunque la evaluación sea automática, se necesita del esfuerzo humano para comparar el resumen generado por un modelo (El-Kassas et al., 2021). Para Steinberger y Ježek (2009) y Awasthi et al. (2021) los métodos de evaluación se clasifican en dos categorías:

[7] La resolución de anáforas es un proceso donde se identifica la referencia de un término que se repite en un texto, así mismo determina la relación entre diferentes partes del texto mejorando la coherencia y claridad del mismo (de los Llanos Carrión Varela, 2014).

- Evaluación extrínseca: Que se centra directamente en la utilidad del resumen, es decir se valora el resumen en función de su capacidad de realizar otras tareas como clasificar información o responder preguntas con éxito.
- Evaluación intrínseca: Se basa directamente en el análisis del resumen generado, evalúa la diferencia entre un resumen generado automáticamente y uno creado de forma manual, con base a la calidad del texto, la selección y el contenido. La métrica más utilizada con frecuencia en este tipo de evaluación es la puntuación ROUGE.

Una vez que el resumen ha sido generado a partir de la lista de los requisitos preliminares, es necesario identificar y extraer los FR. De acuerdo con Dias Canedo y Cordeiro Mendes (2020) la clasificación de textos permite organizar documentos en categorías basadas en propiedades y atributos de cada uno de ellos, lo que hace posible agrupar documentos que contengan información similar. Ahora bien, centrando la atención en la Clasificación de Requisitos de *Software* (SRC, por sus siglas en inglés), los requisitos de *software* se pueden clasificar en dos categorías: FR que describen el comportamiento o funcionamiento del *software* y los NFR que representan las restricciones del *software* y las características de calidad (Zubcoff et al. 2019; Rahimi et al., 2021).

Para llevar a cabo la clasificación de los requisitos de *software*, en este trabajo de tesis se implementarán las siguientes fases de acuerdo con Dias Canedo y Cordeiro Mendes (2020) y Quba et al. (2021):

- Normalización. Es el paso inicial para la identificación de los requisitos, ya que es donde se "limpian" los datos textuales, a través de la *tokenización*, el cambio de mayúsculas y minúsculas, así como la expansión de abreviaturas (Zhang et al. 2019). Una vez que se ha concluido con la "limpieza" de los datos, estos deben ser vectorizados (Quba et al., 2021).
- Vectorización. De acuerdo con Yang et al. (2022) y Krzeszewska et al. (2022) es un proceso del NLP donde los textos se convierten en representaciones numéricas (vectores), lo que permite extraer sus características mediante algoritmos de ML. Algunas de las técnicas utilizadas para la vectorización en el NLP es la Bolsa de Palabras (BoW, por sus siglas en inglés), la Frecuencia de Términos-Frecuencia de Oraciones Inversa (TF-IDF por sus siglas en inglés) y, los vectores de palabras promediadas (Dias Canedo y Cordeiro Mendes, 2020).
- Clasificación. Como mencionan Talele y Phalnikar (2021), se lleva a cabo utilizando algoritmos de aprendizaje supervisados, no supervisados, o semisupervisados. Para Van Engelen y Hoos (2020) el aprendizaje supervisado utiliza un conjunto de datos compuesto por una entrada (x) y un valor de salida correspondiente (y) y predicen el valor de salida para las entradas que no han sido previamente observadas, estos algoritmos incluyen una solución deseada llamada "etiqueta". En el aprendizaje no supervisado los datos de entrenamiento no están etiquetados, por lo que los algoritmos deben aprender a clasificar los datos sin asistencia previa. Por último, el aprendizaje semisupervisado intenta mejorar el rendimiento del algoritmo supervisado y no supervisado, utilizando datos parcialmente etiquetados. Algunos de los algoritmos de aprendizaje son el Vecino Más Cercano (k-NN, por sus siglas en inglés), SVM y la Regresión Logística (LR, por sus siglas en inglés).

Es importante mencionar que, a partir de la propuesta antes descrita se implementará una herramienta de *software* para ser aplicada en un caso de estudio que se llevará a cabo en una

microempresa de desarrollo de *software* y así poder evaluar el número de FRs identificados y clasificados.

El desarrollo de esta investigación se llevará a cabo bajo un enfoque mixto: cualitativo y cuantitativo. El enfoque cualitativo permitirá describir la situación sobre un evento en particular, y el enfoque cuantitativo permitirá explicar fenómenos mediante la recopilación de datos numéricos que se analizan utilizando métodos matemáticos, este enfoque permitirá cuantificar la variación de una determinada situación (Kandel, 2020).

1.5. Estructura de la tesis

En el capítulo 2 se presenta el marco teórico y el estado del arte sobre la etapa de elicitación de requisitos, los problemas que se presentan en ella y la aplicación del NLP para resolverlos. Así mismo, se presentan los trabajos relacionados con el tema de esta tesis con el objetivo de analizarlos y compararlos para poder sustentar una propuesta de metodología.

El Capítulo 3 se describe la propuesta de solución para el desarrollo de esta tesis, como la generación de resúmenes extractivos, la identificación de los FR; y la construcción de la herramienta *software* propuesta.

El Capítulo 4 se llevará a cabo la validación de la propuesta a través de un caso de estudio en un entorno controlado, mostrando los resultados encontrados.

El Capítulo 5 presentarán las principales conclusiones derivadas de la investigación, conclusiones personales, y una propuesta de trabajo futuro.

En la sección de Anexos se presentará cualquier información adicional requerida para esta tesis (e.g., lista de acrónimos).

Por último, la sección de Bibliografía presenta las referencias citadas en el presente trabajo de investigación.

2. Marco teórico

Este capítulo abordará conceptos y fundamentos teóricos que darán sustento a la solución del problema abordado en este trabajo de investigación.

2.1. El proceso de la elicitación de requisitos en la Ingeniería de Requisitos

La elicitación de requisitos es la primera etapa dentro del proceso de la IR, ya que es aquí donde se obtienen y recopilan los deseos, necesidades y expectativas de los *stakeholders*. En este punto, a toda esta información se le conoce como "requisitos preliminares", los mismos que en la etapa de análisis se transformarán continuamente hasta convertirse en requisitos (Gunda, 2008; Görer et al., 2024). Se llaman requisitos preliminares dado que son la base sobre la cual se definen los requisitos del *software* (i.e., delinean las condiciones básicas o restricciones que deben cumplirse antes de que el equipo de desarrollo comience a conceptualizar el *software* como tal en función de las necesidades y deseos de los *stakeholders*) (Andry et al., 2020). Un requisito preliminar incluye necesidades y expectativas de los *stakeholders*, conceptos del medioambiente y del sistema (e.g., "capaz de especificar que la función del lenguaje revise que el deletreo sea correcto", "correr bajo Linux", o "no requiere conversión de datos"). En este sentido, es crucial elicitar "buenos" requisitos preliminares para que, de esta forma, se asegure un diseño y desarrollo del *software* basados en requisitos correctos (Femmer et al., 2016; Medeiros et al., 2018; Ali et al., 2019).

De acuerdo con el estándar ISO/IEC/IEEE 29148 (2018) un "buen" requisito preliminar debe poseer las siguientes características:

1. Necesario. Debe definir una capacidad, característica, restricción y/o factor de calidad esencial del *software* y si se elimina, existirá una deficiencia que no podrá ser suplida por otras características del producto o proceso.
2. No ambiguo. Debe escribirse de tal manera que sólo exista una interpretación de este. Su redacción debe ser simple y fácil de entender.
3. Consistente. Es decir, debe estar libre de conflictos con otros requisitos.
4. Completo. Debe describir de una forma medible y suficiente, la capacidad y las características que satisfagan las necesidades de los *stakeholders*.
5. Singular. La redacción del requisito debe incluir únicamente una sentencia o enunciado, es decir no se pueden utilizar conjunciones.
6. Factible. Debe ser técnicamente alcanzable, i.e., no requerir avances tecnológicos importantes y ajustarse a las restricciones del sistema (e.g., costo, cronograma, técnico, legal, regulatorio) con un riesgo aceptable.

7. Trazable. Debe poderse rastrear desde la lista de deseos y necesidades al documento de especificación, al diseño y a otros artefactos incluyendo aquellos posteriores a la implementación. También debe ser posible el rastreo de su relación con otros requisitos y con su fuente u origen (*stakeholder*).
8. Verificable. Debe demostrarse que el sistema satisface el requisito especificado. La verificabilidad mejora cuando el requisito es medible.
9. Libre implementación. Debe describir lo que necesita el *software*, evitando imponer restricciones innecesarias del diseño arquitectónico. Esto le permitirá ser independiente de la implementación centrándose en lo que se requiere y no en cómo se debe hacer.

Dado que la elicitación de requisitos se realiza en una etapa temprana del desarrollo de *software*, los *stakeholders* pueden no estar seguros de lo que quieren o necesitan, o bien, no diferenciar entre estos dos aspectos, esta falta de claridad hace que éste se vuelva un proceso complejo. Como puede verse, el factor humano es indispensable en esta etapa y al ser el NL usado por excelencia para comunicarse, los requisitos preliminares elicitados presentan una variedad de problemas como ambigüedades, inconsistencias, falta de integridad, etc. (Dar, 2020; Attanayaka et al., 2022).

Por lo anterior, puede decirse que la elicitación de requisitos es una de las etapas críticas de la IR, ya que es aquí donde se obtienen los requisitos; punto de partida necesario y determinante para el éxito o fracaso de los proyectos de *software* (Dar, 2020; Ferrari et al., 2021; Chaudhary et al., 2020). Es importante mencionar que llevar a cabo una "mala" elicitación de requisitos afecta no solo esta etapa, si no a todas y cada una de las etapas posteriores de la IR, y al desarrollo de *software* en general, ocasionando entre otras cosas: atrasos en las fechas de entrega, incrementos en el esfuerzo requerido, retrabajo, altos costos de mantenimiento e incluso puede provocar el fracaso de un proyecto al no cumplir con el objetivo del mismo (Wong et al., 2017; Dar, 2020; Attanayaka et al., 2022). Sobre este punto, Hamid et al. (2019) y Lam et al., (2023), analizan en sus investigaciones los factores que influyen en el fracaso de los proyectos de *software* y mencionan que algunos de estos son: la falta de participación e involucramiento de los *stakeholders*; una compresión poco clara del proyecto lo que conlleva a una falta de requisitos y a cambios constantes en estos y en el alcance del proyecto; impactando así, de forma negativa en el costo y tiempo de desarrollo. Así mismo mencionan que es importante involucrar a los *stakeholders* desde las etapas iniciales del proyecto para así, asegurar que la calidad del producto final cumpla con las necesidades y expectativas de éstos.

Hussain et al. (2016) mencionan que los fracasos en los proyectos de *software* pueden rastrearse, principalmente, a unos requisitos mal redactados, con falta de claridad, inadecuados, volátiles, deficientes, o mal diseñados. Así mismo, los autores hacen hincapié en que una comprensión adecuada de los requisitos, así como la participación de los *stakeholders* son las principales determinantes del éxito de un proyecto, ya que permitirán que los requisitos queden claramente definidos (Kotowaroo y Sungkur, 2022).

En el mismo tenor, para Schmidt (2013), Sampada et al. (2020), y Agredo-Delgado et al. (2023) el principal factor de éxito de un proyecto de *software* es la comprensión y entendimiento de los requisitos de *software* por parte de los *stakeholders,* aspecto crucial para que la gestión de los requisitos se lleve a cabo de una manera efectiva.

Ahora bien, ¿en qué consiste la elicitación de requisitos? De acuerdo con Mauger et al. (2010), Laplante y Kassab (2022) y Attanayaka et al. (2022) este proceso implica varias fases fundamentales:

- Comprensión del dominio del problema,

- Identificación de los *stakeholders*,
- Análisis de los *stakeholders*, donde se agrupan según sus intereses y niveles de participación,
- Selección de las técnicas de elicitación, y finalmente,
- Realización de la elicitación de requisitos.

Por otro lado, Bourque y Fairley (2014), Sandhu y Weistroffer (2018), Dar et al. (2018), y Pacheco et al. (2023) agrupan esas fases mencionadas en el párrafo anterior, en tres actividades principales:

- Identificar a los *stakeholders* de forma individual o en clases,
- Elicitar los requisitos de los *stakeholders* identificados a través del uso de técnicas de elicitación, e
- Integrar, refinar y organizar la información obtenida en el punto anterior, para determinar las funcionalidades y limitaciones del *software*.

Como se puede ver en lo expuesto anteriormente, es fundamental dedicar el tiempo y esfuerzo necesarios a la etapa de elicitación de requisitos para que de esta forma se asegure que los requisitos preliminares cumplen tanto las necesidades de los *stakeholders* como los objetivos del proyecto. Si embargo para poder obtener la lista de los requisitos preliminares es indispensable la selección de los *stakeholders*, así como el uso correcto de las técnicas de elicitación que permitan identificar, analizar y recopilar las necesidades de los *stakeholders* (Dar, 2020).

A continuación, se mencionan qué son las técnicas de elicitación, así como su clasificación e importancia dentro de la elicitación de requisitos.

2.1.1. Uso de las técnicas de elicitación de requisitos

La IR es fundamental en el proceso de desarrollo de *software* ya que establece las bases de lo que se va a desarrollar. En este contexto, las técnicas de elicitación desempeñan un papel importante para que el(la) ingeniero(a) de requisitos comprenda el dominio del problema y esto le permita obtener los requisitos preliminares necesarios para el desarrollo del *software* (Menezes, 2023). Como señalan Puspitaningrum y Sintiya (2022), el conocer y emplear adecuadamente las técnicas de elicitación, permite optimizar la calidad del *software* al asegurar que los requisitos recopilados reflejan las necesidades y expectativas de los *stakeholders*. Sin embargo, la selección y aplicación de éstas no son tareas triviales, ya que la selección de técnicas inadecuadas dificulta la obtención de unos "buenos" requisitos preliminares. Aunado a lo anterior, es importante que el(la) ingeniero(a) tenga la experiencia y conocimientos adecuados que le permitan seleccionar correctamente las técnicas de elicitación.

Es crucial entender que no existe una técnica mejor que otra, como mencionan Besrour et al., (2014) y Puspitaningrum y Sintiya (2022), el seleccionar una técnica de elicitación adecuada dependerá de la experiencia y conocimiento del(a) ingeniero(a) de requisitos, así como de las características del proyecto de *software* a desarrollar (e.g. tamaño, complejidad, tipo, o contexto) y de la personalidad, conocimiento, habilidades y limitaciones de los *stakeholders*.

Haciendo un consenso de las Revisiones Sistemáticas de Literatura (RSL) sobre técnicas de elicitación de requisitos, realizadas por Tiwari y Rathore (2017), Pacheco et al. (2018), Okesola et al. (2019), Canché y Pino (2021), y Alkhomsan et al. (2024), éstas se pueden agrupar de la siguiente manera:

- Técnicas tradicionales: Se enfocan en recopilar información de manera directa a través de interacciones con los *stakeholders*. Algunos ejemplos son:

  - Entrevistas: Permiten lograr una comprensión global sobre qué hacen los *stakeholders*, cómo interactuarán con el nuevo *software* y las dificultades que enfrentan con los sistemas actuales. Existen dos tipos de entrevistas: cerradas, donde los participantes responden a un conjunto de preguntas preestablecidas; y abiertas, donde no hay una estructura de entrevista predefinida.
  - Cuestionarios y encuestas: Los cuestionarios permiten elicitar requisitos de diferentes *stakeholders* y, su efectividad depende del formato y preparación de las preguntas. Son útiles para obtener requisitos precisos y consistentes. Las encuestas sirven para elicitar requisitos de un gran número de *stakeholder*s incluso cuando están dispersos geográficamente, no obstante, restringen una exploración más detallada de sus perspectivas.
- Técnicas colaborativas: Enfatizan la interacción y la participación de los *stakeholders* durante el proceso de elicitación. Entre estas se pueden encontrar las siguientes:
  - *Focus Group*: Implica reunir un grupo de *stakeholders* que tienen intereses comunes para discutir sus inquietudes sobre las características que tendrá el *software* a desarrollar.
  - Lluvia de ideas: Consiste en llevar a cabo sesiones donde los *stakeholders* generan ideas, que posteriormente se convertirán en requisitos preliminares, con el objetivo de definir las bases del proyecto.
  - *Workshops*: Son reuniones realizadas con los *stakeholders* para resolver y obtener los requisitos, esta técnica ayuda a priorizar y gestionar los requisitos al ser dinámicas e interactivas. También permiten resolver conflictos entre los *stakeholders* llegando a un consenso.
- Técnicas contextuales y observacionales: Se basan, como su nombre lo indica, en la observación del contexto donde se usará el *software*. Lo que permite descubrir requisitos que los *stakeholders* no pueden verbalizar fácilmente. Algunos ejemplos son:
  - Observación etnográfica: Consiste en interactuar con los *stakeholders*, tomar notas sobre las actividades y procesos que éstos realizan en su entorno de trabajo, obteniendo así, necesidades que los *stakeholders* no pueden comunicar verbalmente. También permite descubrir requisitos implícitos del *software* que reflejen las formas actuales en que trabajan los *stakeholders*, en lugar de los procesos formales definidos por la organización.
  - Creación de prototipos: Esta técnica proporciona una representación visual de cómo será el producto final, lo que facilita tanto la comprensión como la comunicación de los requisitos y valida ideas de diseño.
- Técnicas cognitivas: Permiten descubrir requisitos que se basan en el conocimiento tácito y la experiencia de los *stakeholders* a través del uso de técnicas basadas en análisis de tareas para así, estudiar a los *stakeholders* de acuerdo con su perspectiva.
  - *Card sorting*: En esta técnica se proporciona un conjunto de tarjetas a los *stakeholders* para que las llenen con información que creen importante acerca de la funcionalidad del producto de *software* a desarrollar.
  - *Laddering:* Es una forma de entrevista estructurada donde se formula un conjunto limitado de preguntas a los *stakeholders*. El conjunto de preguntas está ordenado en

orden jerárquico tomando en cuenta los valores personales (e.g. creencias, actitudes y comportamientos de una persona). El éxito de esta técnica radica en que los *stakeholders* tengan conocimiento sobre el dominio del problema.

- *Repertory Grid:* Esta técnica permite a los *stakeholders* expresar su experiencia desde su propia percepción del mundo (i.e. de acuerdo con sus propios constructos personales) e ir llenando una matriz donde deben quedar registrados los requisitos. Es una técnica útil para distinguir los diferentes dominios de información.

- Técnicas basadas en modelos: Como su nombre lo indica, se utilizan modelos o representaciones para elicitar los requisitos de manera efectiva. Algunos ejemplos son:
  - Modelos: Con el objetivo de conceptualizar los requisitos se usan diagramas de flujo, de estado y UML. Esta técnica permite al *stakeholder* a visualizar el proceso del *software* a desarrollar, ayuda a los(as) ingenieros(as) de requisitos a comunicarse de una forma clara con los *stakeholders*.
  - Casos de uso: Especifican la interacción entre el *software* y sus usuarios (actores), así mismo detallan la comunicación y el comportamiento del nuevo *software* con otros sistemas o bases de datos. Los casos de uso se documentan mediante un diagrama de alto nivel y una descripción textual, representando así los FRs del *software*.
  - Escenarios: Describen el estado del *software* mediante un flujo de eventos, es decir presentan la situación inicial del *software* e indican las actividades que pueden ocurrir hasta su finalización, detallando la descripción de cada requisito. Cada escenario incluye detalles sobre el entorno, las acciones de los usuarios y las respuestas esperadas del *software*.

La selección de las técnicas de elicitación se basa en el conocimiento del dominio del problema por parte del(a) ingeniero(a) de requisitos, así como en las habilidades, experiencia, forma de comunicarse con los *stakeholders*, redactar requisitos precisos, analizar documentos existentes, adaptación al entorno del proyecto, entre otros factores. Es importante mencionar que la correcta aplicación de las técnicas de elicitación permite que el(a) ingeniero(a) de requisitos obtenga información relevante, completa y correcta de los *stakeholders* (i.e. requisitos preliminares) (Henriksson y Zdravkovic, 2022). De acuerdo con Ferreira Martins et al., (2019) y Henriksson y Zdravkovic (2022), es fundamental la comprensión del dominio del problema por parte del(a) ingeniero(a) de requisitos para así, poder hacer una correcta selección de las técnicas de elicitación, ya que de otro modo se complicará la obtención de la lista de los requisitos preliminares (Sampada et al., 2020). Ahora bien, una vez que los requisitos han sido elicitados, estos deben escribirse de forma clara (evitando utilizar términos ambiguos), con información completa y consistente.

Como puede verse, la elicitación de requisitos requiere de un gran esfuerzo y atención por parte del(a) ingeniero(a) de requisitos, ya que en su ejecución pueden presentarse diversos problemas tales como: una mala selección de la técnica de elicitación, problemas de comunicación o la falta de participación de los *stakeholders*; provocando que el tiempo y presupuesto del proyecto aumente (Attanayaka et al., 2022). Es importante mencionar que usando una sola técnica de elicitación es imposible extraer el 100% de los requisitos preliminares, lo que conlleva un alto riesgo de que el proyecto no satisfaga las necesidades de los *stakeholders* y, por ende, termine siendo un fracaso (Wiegers y Beatty, 2013; Akram et al.,2024). Por lo tanto, es necesario que el(a) ingeniero(a) de requisitos emplee más de una técnica de elicitación tomando en cuenta las características de los *stakeholders* y del proyecto a desarrollar, para tratar así, de obtener la mayor información posible.

Sin embargo, como se ha expuesto a lo largo de este punto, en todo el proceso de la IR se utiliza el NL, tanto para elicitar los requisitos como para escribirlos y puede que, a pesar, de haber utilizado las técnicas adecuadas, se obtengan "malos" requisitos debido a los problemas característicos del NL como: la ambigüedad, la inconsistencia o la incompletitud de los requisitos preliminares. A continuación, se abordará específicamente la aplicación del NLP en el proceso de elicitación de requisitos.

2.1.2. El NLP en la elicitación de requisitos

Durante mucho tiempo, el NL ha jugado un papel importante en la IR y en el desarrollo de *software* en general, ya que es, a menudo, la elección por antonomasia para representar requisitos. Sin embargo, tiene un alto grado de propensión innata a la ambigüedad, inconsistencia, redundancia, etc. (Umber et al., 2011; Haque et al., 2019; Ezzini et al., 2021). A pesar de estas desventajas, el NL es la herramienta por excelencia del(a) ingenieros(as) de requisitos para comunicarse con los *stakeholders* y así obtener sus requisitos preliminares, debido a que es un modelado informal que tiene una sintaxis y una semántica intuitiva que puede ser entendida y utilizada por cualquier persona (Ambriola y Gervasi, 1997; Pandey et al., 2013; Kang et al., 2020).

En la RSL de Umar y Lano (2024) sobre los avances en la automatización de la IR, se menciona que el 94% de las herramientas desarrolladas con este propósito, están basadas en tecnologías de NLP. Por ejemplo, para el caso específico de la elicitación de requisitos, se hace mediante la extracción de entradas léxicas relevantes a partir de una gran cantidad de datos textuales obtenidos al usar las técnicas de elicitación (e.g., entrevistas, *workshops*, análisis de protocolos, etnografía, etc.). Algunas tecnologías y técnicas de NLP incluyen aquellas basadas en reglas, POS y Patrones Léxico-Sintácticos (LSP, por sus siglas en inglés) y técnicas de análisis basadas en actos de habla, entre otras.

Es importante mencionar que todas las técnicas para elicitar requisitos preliminares generan un gran porcentaje de notas o documentos escritos en NL, ya que por lo general, los *stakeholders* no conocen lenguajes formales (e.g., lenguajes basados en modelos: Z, VDM, B, Alloy, JML, Event-B; lenguajes basados en estados finitos: gráficos de estado, SDL, Máquinas de estados; redes de Petri, lógicas temporales, etc.) por lo que el(a) ingeniero(a) de requisitos tiene que utilizar únicamente el NL para asegurar la comunicación con y entre ellos (Ambriola y Gervasi, 1997, Ibrahim y Ahmad, 2010; More y Phalnikar, 2012; y Šenkýr y Kroha, 2019). Kamsties (2005) indica que el 79% de los documentos de requisitos están escritos en NL, el 16% están escritos en NL estructurado y sólo el 5% están escritos en un lenguaje formal, aunque con frecuencia se utilizan diagramas y otras representaciones semiformales para complementar las especificaciones de requisitos informales.

Es en este tenor, que en la década de los 80´s surge el uso del NLP en la IR como una alternativa para mejorar el proceso de elicitación de requisitos (Zhao et al., 2022). De acuerdo con Ambriola y Gervasi (1997); Ibrahim y Ahmad (2010); Pandey et al. (2013); y Šenkýr y Kroha (2019) los requisitos expresados en NL a menudo son ambiguos, incompletos e inconsistentes debido a que la interpretación y comprensión de todo lo expresado y descrito en NL tiene el potencial de verse influenciado por factores geográficos, psicológicos y sociológicos del(a) ingeniero(a) de requisitos (More y Phalnikar, 2012).

El NLP se aplica en la elicitación de requisitos para revisar una gran variedad de documentos que se encuentran escritos en NL, con el objetivo de comprender el dominio del problema (e.g., transcripciones de entrevistas, documentos legales, o procedimientos operativos) (Zhao et al., 2022). También puede mejorar notablemente la elicitación y especificación de requisitos al facilitar el procesamiento de los documentos generados en estas etapas del *software* extrayendo información útil, organizándola y analizándola de manera efectiva con el objetivo de agilizar el manejo de documentos

extensos (Calle y Zapata, 2022; Sampada et al., 2020). A continuación, se abordarán los problemas relacionados con el NL que se pueden presentar durante la elicitación de requisitos.

2.2. Problemas que se presentan en la elicitación de requisitos

La comprensión de los requisitos preliminares representa un desafío significativo para un(a) ingeniero(a) de requisitos, ya que el elicitar información clave de los *stakeholders* podría parecer una tarea trivial al principio. Sin embargo, en la mayoría de las ocasiones el conocimiento de los *stakeholders* suele ser tácito, lo que complica la transferencia de dicha información y la convierte en una tarea difícil. De la misma forma, los *stakeholders* a menudo no saben cómo expresar lo que realmente quieren o necesitan, debido a que el conocimiento varía de persona en persona y está sujeto a múltiples interpretaciones, lo que ocasiona malentendidos o ambigüedades en los requisitos preliminares obtenidos, impactando así directamente en el *software* que se pretende desarrollar (Lamsweerde 2009; Pressman y Maxim, 2019; Popoola et al., 2024).

Como se expuso en el punto anterior, en la etapa de elicitación, los requisitos preliminares frecuentemente son documentados en NL, ya que los *stakeholders* expresan sus necesidades en sus propios términos y conocimientos. Sin embargo, el uso del este lenguaje ocasiona que los requisitos puedan ser ambiguos, inciertos, incompletos, inconsistentes o incoherentes (Maatuk y Abdelnabi, 2021; Lafi et al., 2021).

Según Sandhu y Sikka (2015), el empleo del NL en la elicitación de requisitos presenta principalmente problemas de inconsistencia, ambigüedad e incompletitud. No obstante, existen otros retos relevantes, como la presencia de numerosos *stakeholders,* así como su diversidad y, el manejo de grandes volúmenes de documentos y datos.

2.2.1. La ambigüedad en los requisitos preliminares

La existencia de ambigüedades e inconsistencias en los requisitos preliminares representa desafíos significativos para el(la) ingeniero(a) de requisitos (Ashfaq y Bajwa, 2021). Dado que estos requisitos provienen de diversos *stakeholders* y suelen estar redactados en NL, pueden conducir al uso de términos cuyo significado varía según el contexto y el entorno. La ausencia de conocimiento sobre el dominio del *software* a desarrollar puede provocar ambigüedad en la elicitación de los requisitos preliminares (Ashfaq y Bajwa, 2021). De acuerdo con Jusoh (2018), la ambigüedad en los requisitos, generada por el uso del NL se refiere a la posibilidad de que una oración sea interpretada por el (la) ingeniero(a) de requisitos o los *stakeholders* en más de un sentido. Esto puede derivar en que la documentación producida no resulte lo suficientemente clara, conduciendo así al desarrollo de un *software* diferente al esperado por los *stakeholders*. Tal situación afecta negativamente el potencial de éxito del proyecto (Ribeiro y Berry, 2020; Ezzini et al., 2021).

Dado que la ambigüedad es un concepto complejo de explicar en la IR, se han propuesto varias clasificaciones y taxonomías que se exponen a continuación (Massey et al., 2014; Bano, 2015; Maurya et al., 2015; Sabriye y Zainon, 2017; Ferrari y Esuli, 2019):

- La ambigüedad léxica ocurre cuando una palabra tiene múltiples significados válidos.
- La ambigüedad sintáctica o ambigüedad estructural se presenta cuando una secuencia de palabras tiene múltiples análisis gramaticales válidos, es decir una oración se puede analizar de más de una manera, lo que da como resultado más de una estructura gramatical donde cada una proporciona significados diferentes.

- La ambigüedad semántica puede presentarse cuando la oración tiene varias interpretaciones dentro de su contexto sin contener ambigüedad léxica, estructural y sintáctica, es decir una oración tiene más de una interpretación basada enteramente en el contexto que lo rodea.
- La ambigüedad pragmática se ocupa de la relación entre las interpretaciones de una oración y su contexto. Depende del contexto del requisito preliminar, incluido el conocimiento del lector de este, por lo que dos lectores con antecedentes diferentes pueden interpretar un único requisito preliminar de dos maneras diferentes.
- La ambigüedad de los errores de lenguaje que es una construcción gramaticalmente incorrecta que; sin embargo, se entiende y posiblemente de diferentes maneras debido al error.

Kamsties (2005) menciona que la ambigüedad es típica de la elicitación de requisitos dado que los requisitos preliminares regularmente son informales, incompletos y representan opiniones. Sin embargo, también menciona que algunas causas de la ambigüedad son las siguientes:

- La falta de exhaustividad en la redacción del requisito preliminar: el(a) ingeniero(a) de requisitos no abunda en el al análisis de la información contenida en los requisitos preliminares, por lo que si están incompletos generarán ambigüedad sobre lo que se quiere expresar (i.e., entre más completos sean éstos, menos ambiguos serán).
- La falta de acuerdo: Los puntos de vista individuales de los *stakeholders* muchas veces suelen ser contradictorios, es decir, que las expectativas y objetivos de los *stakeholders* pueden ser divergentes, lo que conduce a interpretaciones diferentes y, por ende, genera ambigüedad.
- La representación: Los requisitos informales son inevitablemente ambiguos, mientras que los requisitos formales lo son significativamente menos, pero aún dejan cierto margen a la ambigüedad. Este tipo de ambigüedad es causada por debilidades del NL, en particular, su incapacidad para expresar conceptos técnicos.

Como se ha evidenciado, la utilización del NL en la redacción de requisitos preliminares puede introducir ambigüedades, incompletitud o interpretaciones erróneas por parte de los *stakeholders*, lo que afecta la coherencia y dificulta la correcta clasificación posterior en FRs o NFRs (Alzayed y Al-Hunaiyyan, 2021; Umar y Lano, 2024).

2.2.2. La inconsistencia en los requisitos preliminares

Se presenta cuando uno o más de los requisitos preliminares incluyen descripciones contradictorias sobre el comportamiento esperado del *software* a desarrollar, complicando tanto su diseño como su implementación, repercutiendo así, negativamente en todo el proceso de desarrollo. En particular, si una inconsistencia no es identificada oportunamente, puede derivar en productos de *software* incorrectos, poco confiables e incluso generar sobrecostos presupuestarios (Gervasi y Zowghi, 2005; Popoola et al., 2024). Estas inconsistencias suelen estar asociadas a la existencia de múltiples perspectivas entre los *stakeholders* respecto a una misma necesidad o expectativa. Asimismo, es posible que las inconsistencias surjan cuando se incorporan nuevos requisitos o se eliminan existentes sin considerar aquellos relacionados, o bien debido a una falta de conocimiento del dominio por parte del(la) ingeniero(a) de requisitos. Adicionalmente, el conocimiento implícito o tácito —denominado presuposición— que no queda reflejado en la documentación también contribuye a que ciertos aspectos sean omitidos en la lista de requisitos obtenidos. Cabe destacar que tales presuposiciones inciden de manera relevante tanto en la inconsistencia como en la incompletitud de los requisitos (Sharma y Biswas, 2012).

Los conflictos en los requisitos preliminares surgen debido a diferencias en los roles, responsabilidades y perspectivas de los *stakeholders*, así como a objetivos, limitaciones y prioridades organizacionales contradictorios. Los conflictos pueden manifestarse en la escritura de FRs y NFRs contradictorios, así como en restricciones contrapuestas. Unos requisitos preliminares contradictorios pueden tener varias consecuencias adversas para los proyectos de *software*, incluidos los retrasos en la toma de decisiones, una calidad comprometida o bien, la insatisfacción de los *stakeholders* (Behutiye et al., 2022; Popoola et al., 2024).

Debido a que el resultado de la elicitación de requisitos es una colección de notas proveniente de cada técnica de elicitación utilizada o bien, un documento informal que describe los requisitos preliminares, y dado que ambos se encuentran desestructurados y sin procesar, pueden contener irrelevancias, inconsistencias y omisiones (Pandey et al., 2013).

2.2.3. La incompletitud en los requisitos preliminares

Otro desafío relacionado con el uso del NL en la elicitación de requisitos es la incompletitud de los requisitos preliminares, ya que estos suelen estar incompletos debido a que los *stakeholders* no pudieron expresar sus necesidades o bien, a que el(a) ingeniero(a) de requisitos no comprendió ni procesó bien la información obtenida o porque dio por hecho algunos pormenores del requisito y no los mencionó durante la elicitación, o bien, olvidó detalles desde el principio que se agregaron posteriormente al documento de requisitos sin verificar su impacto en los demás requisitos preliminares o, porque simplemente no se incluyó toda la información obtenida, lo que conduce inevitablemente a un desarrollo de *software* incompleto e incorrecto (Sharma y Biswas, 2012; Burnay et al., 2015; Šenkýr y Kroha, 2019).

Para Eckhardt et al. (2016), Tariq et al. (2019), Zahid et al. (2020), y Šenkýr y Kroha (2023) la incompletitud es unos de los problemas más frecuentes en la IR y una de las causas más reiteradas del fracaso de un proyecto de *software*, ya que, si los requisitos no se establecen correctamente, entonces todo el proceso de desarrollo estará incorrecto, incluyendo cronogramas y presupuestos.

2.2.4. Otros problemas de los requisitos preliminares

Según Zahid et al. (2020) y Alzayed y Al-Hunaiyyan (2021), durante la elicitación de requisitos pueden surgir otras dificultades relacionadas con los *stakeholders*, tales como: falta de disponibilidad de información relevante, comprensión insuficiente del dominio, expectativas poco realistas respecto al *software* por construir, barreras comunicativas o idiomáticas, así como un escaso compromiso. Estas situaciones suelen derivar en requisitos ambiguos, inconsistentes o incompletos, lo que incrementa los costos y retrasa los plazos de desarrollo del *software*, factores que pueden comprometer el éxito del proyecto (Siahaan et al., 2023).

Como ha podido observarse, por lo general la elicitación de requisitos suele ser un proceso realizado manualmente por el(la) ingeniero(a) de requisitos, lo que puede hacerlo tedioso y susceptible a errores humanos (Hassan y Le, 2020). Por lo que, para garantizar la obtención de requisitos preliminares precisos, completos y consistentes, no solo es fundamental que el(la) ingeniero(a) de requisitos posea un conocimiento profundo del dominio del problema, así como una sólida experiencia metodológica, sino que también se utilicen herramientas de la IA como el NLP.

2.3. Aplicación del Procesamiento de Lenguaje Natural en la IR

El NLP, es un área de la IA que combina la lingüística con la y su objetivo es que las computadoras comprendan las oraciones escritas en NL y las traduzcan a lenguaje máquina (i.e., ceros

y unos) para interpretarlas, analizarlas, y manipularlas. Esto permite extraer información relevante (dependiendo del contexto) mediante el uso de herramientas, métodos y algoritmos (Kang et al., 2020; Khurana et al., 2023).

Un enfoque común de la clasificación las tareas y aplicaciones que resuelve el NLP son: la Comprensión del Lenguaje Natural (NLU, por sus siglas en inglés) y la Generación del Lenguaje Natural (NLG, por sus siglas en inglés). El NLU se encarga de comprender el NL utilizado por los seres humanos para extraer conceptos, entidades, y palabras clave de un texto, y el NLG es un proceso que sirve para generar frases, oraciones y párrafos significativos a partir de textos en NL.

De acuerdo con Khurana et al. (2023) el uso de NLP se ha vuelto popular para representar y analizar computacionalmente el NL. Algunas de las aplicaciones del NLU son las siguientes:

- **Categorización de texto**: Consiste en clasificar textos en diferentes categorías predefinidas, por ejemplo, asignar etiquetas a correos electrónicos, artículos de noticias, o publicaciones en redes sociales.
- **Detección de correos *spam***: Identifica y filtra correos electrónicos no deseados o maliciosos.
- **Extracción de información**: trata de identificar y extraer información específica de un texto, como nombres de personas, fechas, eventos, etc.
- **Comprensión de textos médicos**: Procesa y entiende datos textuales relacionados con la atención médica, como informes médicos y descripciones de síntomas.

Mientras que para el NLG se encuentran las siguientes aplicaciones:

- **Traducción automática**: Consiste en traducir texto de un idioma a otro.
- **Generación de resúmenes**: Involucra la creación de resúmenes concisos a partir de textos largos.
- **Sistemas de diálogo**: Se refiere a la generación de respuestas en conversaciones, como en *chatbots* y asistentes virtuales.

El NLP presenta numerosas aplicaciones dentro de la IS, especialmente en el ámbito de la IR, donde ha sido utilizado durante más de cuatro décadas, remontándose sus orígenes a principios de los años ochenta. El NLP en la IR se emplea fundamentalmente en los procesos de elicitación y especificación de requisitos, permitiendo extraer características lingüísticas de los documentos para la generación de resúmenes y la clasificación de FRs y NFRs, entre otros usos (Ferrari et al., 2021; Sonbol et al., 2022; Zhao et al., 2022).

2.3.1. Tareas del NLP en la IR

La estrecha vinculación entre el NLP y la IR ha motivado a la comunidad investigadora a analizar la incorporación de técnicas y herramientas del NLP en el manejo de textos de requisitos (Zhao et al., 2021). Para Frattini et al. (2024), el NLP comprende múltiples tareas orientadas a optimizar y automatizar las fases de la IR, las cuales pueden ser categorizadas según las actividades desarrolladas. Por ejemplo:

- Generación: Es la creación de un nuevo texto relevante para los requisitos a partir de un texto del que se extraen sus características importantes, por ejemplo, un resumen.
- Detección: Permite ubicar problemas lingüísticos en un texto de requisitos, por ejemplo, en la siguiente oración "*El sistema debe actualizar el inventario cada vez que un producto es*

vendido" Se detecta un error gramatical ya que el verbo debe estar acorde al sustantivo "*producto*"; sin embargo, en este caso se identifica "*es vendida*".

- Extracción: Identifica, en el texto, abstracciones y conceptos clave del dominio de conocimiento, por ejemplo, de la siguiente oración "*El sistema deberá permitir la exportación de reportes de inventario en formato CSV*", se puede extraer, por ejemplo, el sustantivo "*sistema*".
- Clasificación: Clasifica el texto de requisitos en diferentes clases o categorías en función del propósito de la clasificación, i.e., con base en su funcionalidad, para facilitar su distribución y reutilización, en función de su calidad, para identificar NFRs que puedan estar mezclados con los funcionales y viceversa. Por ejemplo, los correos electrónicos se pueden clasificar en categorías como "*Trabajo*", "*Personal*" y "*Spam*".
- Transformación: Traduce el texto a una especificación semánticamente similar pero sintácticamente diferente. En esta tarea se realiza el parafraseo de un texto, i.e., se expresa un texto en otras palabras, pero manteniendo el mismo significado.
- Seguimiento y relación: Establece una relación entre los requisitos y otros artefactos del *software*. Permite respaldar las actividades de un proceso manual y demostrar su coherencia, como por ejemplo identificar las relaciones entre requisitos.
- Búsqueda y recuperación: Permite la búsqueda y recuperación de información relevante de los requisitos en *corpus*[8] existentes. Se pueden reutilizar corpus de requisitos para satisfacer las necesidades de un nuevo *stakeholder* o para respaldar el alcance de dominio para un nuevo producto con características específicas basadas en las descripciones del *software*.

Como se aprecia, las tareas del NLP en el ámbito de la IR son variadas. No obstante, este trabajo se enfocará únicamente en dos: la generación de resúmenes y la clasificación de texto. Estas actividades resultan fundamentales para la IR, ya que la generación de resúmenes contribuye a la comprensión y síntesis de requisitos clave a partir de documentos extensos, mientras que la clasificación de texto posibilita la organización y priorización de los requisitos según su relevancia y categoría, lo cual optimiza el proceso de desarrollo de *software*.

2.3.1.1. Generación de resumen de texto

Para Jha et al. (2022), Lee (2023) y Zhang et al. (2023) un resumen de texto, en el contexto del NLP, es un nuevo texto que se crea a partir de uno o más escritos, que contiene información crucial del texto original y no mide más de la mitad de la longitud de este. Es decir, un resumen reduce el tiempo para leer todo el documento y el espacio necesario para almacenar esa cantidad de datos, además agiliza la búsqueda y la obtención de una mayor cantidad de información sobre un tema, optimizando así el proceso de selección de información (Patel et al., 2017; Abualigah et al., 2019; Prudhvi et al., 2020).

En la elicitación de requisitos se genera una cantidad considerable de información por lo que sintetizar los textos para identificar y formular requisitos es esencial debido a la presencia significativa de información irrelevante. Resumir información facilita al(a) ingeniero(a) de requisitos el análisis y

[8] Un *corpus* de texto es un conjunto de textos que pueden estar constituidos por textos de diferentes dominios, por ejemplo: noticias, artículos científicos, textos literarios, de redes sociales, recetas de cocina, entre otros. El objetivo principal de un *corpus* de texto es aprovechar dichos textos para un análisis lingüístico y estadístico, además de utilizarlos como datos para crear herramientas de NLP (Sarkar, 2016; Matias, 2020).

procesamiento de la información de manera precisa (Spijkman et al., 2023). No obstante, la generación de resúmenes representa una tarea compleja que demanda tiempo, ya que generalmente las computadoras no logran resultados óptimos por su limitada comprensión del NL. Por ello, se ha buscado mejorar los resultados empleando dos enfoques principales para la generación automática de resúmenes: el extractivo y el abstractivo, los cuales se describen a continuación (Mridha, 2021).

Enfoques para la generación de resumen automático

Como se mencionó al cerrar el punto anterior, en el NLP la generación automática de un resumen se puede clasificar con base a al enfoque y este puede ser: abstractivo y extractivo (Awasthi et al., 2021; Prudhvi et al., 2021).

- El enfoque abstractivo es aquel donde el resumen se genera formando frases y oraciones propias a partir del texto original, obteniendo así un resumen coherente. En este enfoque se parafrasean los datos relevantes para que encajen en la semántica del texto creando así, un resumen que contenga todos los puntos nodales del documento fuente, pero con oraciones diferentes a las originales (Awasthi et al., 2021; Zhang et al., 2023). Los resúmenes abstractivos son más complejos en comparación con los extractivos, ya que aún se está muy lejos de alcanzar el nivel de la mente humana en la generación de resúmenes, aunque con los avances del uso de Redes Neuronales Artificiales (ANN, por sus siglas en inglés) para la traducción automática neuronal y los modelos de secuenciación como por ejemplo, el modelo pre entrenado BERT, o la generación automática de datos de entrenamiento, para aprovechar recursos como los resúmenes existentes han mejorado su calidad (Awasthi et al., 2021).
- El enfoque extractivo las oraciones se seleccionan directamente del documento fuente para formar, posteriormente el resumen. Es decir, se escogen las oraciones más importantes del documento original, con base al dominio de conocimiento, y luego se concatenan para generar así el resumen (Awasthi et al., 2021; Zhang et al., 2023). Este enfoque es más rápido y sencillo de implantar que el abstractivo, ya que contiene oraciones y terminologías idénticas a las del texto original. De acuerdo con Widyassari (2022) el resumen extractivo es un campo de investigación extenso que está alcanzando un estado de madurez ya que los resúmenes extractivos tienden a ofrecer resultados satisfactorios y eficientes a diferencia de los abstractivos.

Spijkman et al. (2023) indican que persisten oportunidades para la implementación del enfoque extractivo en la IR, considerando su simplicidad y eficiencia en la generación de resúmenes. Dicho enfoque resulta más adecuado en comparación con el abstractivo, ya que produce información pertinente mediante la selección e incorporación directa de fragmentos relevantes del texto original en el resumen, sin alteraciones, lo que implica una menor demanda de recursos computacionales (Yadav et al., 2021; Pati y Rautray, 2024). Por lo tanto, esta tesis centrará su atención exclusivamente en el método extractivo de resumen.

2.3.1.2. Clasificación de texto (Sarkar, 2016)

La clasificación de textos o documentos consiste en asignarlos a una o más categorías de las previamente definidas en el punto 2.3.1. Dado que los documentos son textuales y pueden variar desde una sola oración hasta un párrafo completo, la clasificación de un texto se puede enfocar ya sea en el contenido de este: es decir, se dan prioridades o ponderaciones a temas específicos que permitan

determinar la clase del documento; o bien, en las solicitudes de los *stakeholders*: donde estas están dirigidas a audiencias específicas.

Para la clasificación de texto es fundamental seguir los siguientes pasos de forma ordenada:

- **Preprocesamiento y extracción de texto:** Se limpia el texto, eliminando caracteres especiales, números o cualquier otra cosa que no sea relevante. Además, se identifican y seleccionan las palabras claves del texto que ayudarán al modelo a realizar la clasificación de texto.
- **Entrenamiento del modelo:** Se selecciona el tipo aprendizaje para el modelo, ya sea el ML supervisado o no supervisado.
- **Predicción y evaluación del modelo**: En este paso se mide el desempeño del modelo con respecto a las métricas de precisión, de recuerdo y la medida-F.
- **Implementación del modelo:** Se aplican todos los puntos mencionados antes, para posteriormente integrarlo en un entorno real.

2.3.2. El NLP en la elicitación de requisitos

Con el aumento en la demanda de *software* de mayor complejidad, se incrementa también la necesidad de automatizar la elicitación de requisitos mediante el uso del NLP, dado que estos requisitos suelen estar redactados en NL. El emplear herramientas, métodos y recursos de NLP puede mejorar la calidad de los requisitos preliminares obtenidos y, de este modo, facilitar a los(as) ingenieros(as) de requisitos la realización de tareas sobre requisitos textuales orientadas a identificar y perfeccionar aspectos lingüísticos problemáticos (Arellano et al., 2015; Ronanki et al., 2023).

La elicitación de requisitos tiene como núcleo o parte central la participación de los *stakeholders*, por lo tanto, el NL sigue siendo la forma natural de obtener, documentar y validar los requisitos. Sin embargo, conforme el alcance del dominio del *software* aumenta, los(as) ingenieros(as) de requisitos tienen que procesar una mayor cantidad de documentos escritos en NL. En la mayoría de las ocasiones esto resulta en retrasos y errores en la caracterización de dicho dominio ya que los requisitos provienen de múltiples *stakeholders* con diferentes necesidades, roles y responsabilidades, por lo que este proceso está sujeto a inconsistencias, ambigüedad, incompletitud e inexactitud propios del NL (Sonbol et al., 2022). Así mismo, la falta de comprensión de los requisitos, por parte tanto del(a) ingeniero(a) de requisitos como de los demás *stakeholders*, aumenta el riesgo del fracaso, así como el tiempo y costos del proyecto.

En este ámbito el NLP se vuelve clave, ya que permite acelerar el proceso de la elicitación de los requisitos preliminares y mitigar el impacto de la ambigüedad, las inconsistencias, incompletitud, inexactitud y las malas interpretaciones provenientes de los documentos de requisitos basados en NL (Calle Gallego y Zapata Jaramillo, 2023).Además facilita el procesamiento de texto, brindando así un soporte en las actividades de la IR, principalmente en proyectos grandes, donde se requiere un tiempo mayor para procesar, analizar y comprender grandes volúmenes de documentos de requisitos. El NLP trata de evitar que el(a) ingeniero(a) de requisitos genere documentos de requisitos con los problemas mencionados en el párrafo anterior, ya que éstos no se quedan en la fase de requisitos, si no que se acarrearan en todas y cada una de las etapas posteriores del desarrollo de *software* (Sonbol et al., 2022).

En la elicitación de requisitos, el NLP se emplea principalmente en tareas de extracción y clasificación de texto, lo cual facilita la identificación y obtención de información relevante a partir de un corpus textual. Esto permite al(la) ingeniero(a) de requisitos realizar procesos automatizados de comprensión, extracción y análisis (Frattini et al., 2024). La generación automática de resúmenes es

una aplicación del NLP que produce versiones abreviadas y coherentes de grandes volúmenes de datos textuales, con el propósito de hacerlos más gestionables sin perder su significado esencial (Prudhvi et al., 2021; Raza et al., 2023). Además, una definición temprana de los requisitos contribuye a la adecuada selección y configuración tanto de *hardware* como de *software*. Cabe señalar que la extracción manual de requisitos y su posterior clasificación demanda considerable tiempo y atención por parte del personal especializado; por ello, en los últimos años, se han desarrollado métodos automáticos de clasificación en el ámbito de la IR (Kaur y Kaur, 2024).

2.4. Trabajos relacionados

Esta sección ofrece una revisión detallada de la literatura relacionada con el tema de la tesis: la aplicación del NLP en la elicitación de requisitos de *software*. A continuación, se identifican y analizan estudios relevantes que exploran aspectos fundamentales de esta línea de investigación.

2.4.1. Requirements-Collector

2.4.1.1. Objetivo

Panichella y Ruiz (2020) presentan *Requirements-Collector*, una herramienta diseñada para automatizar la especificación de requisitos y el análisis de los comentarios de los *stakeholders* en las sesiones de elicitación.

2.4.1.2. Descripción

En esta investigación se diseñó una herramienta de recopilación de requisitos basada en el concepto de una sala de ingeniería de requisitos, donde los *stakeholders* discuten los requisitos que se van especificando de forma automática en las historias de usuario, así como las opiniones, que se clasifican automáticamente como reseñas para su análisis posterior. *Requirements-Collector* está compuesto por componentes que utilizan estrategias de Aprendizaje Profundo (DL, por sus siglas en inglés) (no supervisado) y ML (supervisado) que permiten la clasificación automatizada, de los requisitos discutidos en las reuniones, en forma de grabaciones de audio y de reseñas de usuarios (comentarios textuales). En el entrenamiento de esta herramienta se utilizaron datos de texto provenientes de trabajos anteriores de los autores, relacionados con el análisis de requisitos de las transcripciones y revisiones y comentarios de usuarios. La herramienta clasifica los requisitos obtenidos en las sesiones de elicitación en FRs y NFRs.

2.4.1.3. Resultados

R*equirements-Collector* se aplicó en más de diez modelos de ML supervisado y en un método de DL para evaluar su precisión. Los resultados mostraron que, para la clasificación de transcripciones es más conveniente utilizar las estrategias de DL, dado que en las pruebas realizadas se obtuvo una medida-F promedio del 33% con estos modelos, y un 5 % con el modelo de ML. En cuanto a la clasificación de los comentarios de los usuarios, el modelo ML SMO[9] demostró un mejor rendimiento con un porcentaje del 77 %.

[9] La Optimización Mínima Secuencial (SMO, por sus siglas en inglés) es un algoritmo de SVM, permite encontrar parámetros óptimos para una SVM que maximice el margen entre las clases de datos, el algoritmo SMO es fácil y simple de implementar, se adapta de manera eficiente a problemas grandes (Sun et al., 2010).

2.4.2. Identificación automatizada de requisitos

2.4.2.1. Objetivo

Hassan y Le propusieron en 2020 un modelo para la identificación automática de requisitos en documentos relativos a contratos de construcción, mediante técnicas de NLP con el objetivo de reducir el tiempo de lectura del(a) ingeniero(a) de requisitos.

2.4.2.2. Descripción

El modelo desarrollado por los autores facilita la identificación de requisitos en contratos de construcción, analizando el texto que describe obligaciones y necesidades, mientras que las instrucciones y declaraciones de apoyo, como definiciones y encabezados, se consideran no requisitos. Para el entrenamiento y validación del modelo, se construyó un conjunto de datos etiquetados mediante la clasificación manual de los requisitos, logrando extraer 1,787 declaraciones de siete contratos diferentes. Posteriormente, el proceso de clasificación textual se llevó a cabo empleando dos metodologías: una basada en reglas y otra fundamentada en técnicas de ML.

En el modelo basado en reglas, se generaron 15 reglas manualmente para optimizar el rendimiento de este. Se incluyeron un conjunto de reglas “IF-THEN”, donde “IF” muestra una condición mientras que la salida de la parte posterior “THEN” muestra una conclusión del resultado que se deriva de la evaluación de la condición IF. En la clasificación basada en ML primero se realizó la representación de texto, en donde las características importantes del texto se convirtieron en un vector numérico (una característica importante hace referencia a una palabra de una declaración textual). Para la representación se utilizó BoW y el modelo Word2vec usados en ANN para generar un vector multidimensional que representa la semántica de cada palabra única en todo el *corpus*.

Posteriormente se implementaron cuatro algoritmos supervisados para desarrollar los modelos de identificación de requisitos: NB, SVM, LR y Perceptrón Multicapa (MLP, por sus siglas en inglés) para lo cual se dividió aleatoriamente el conjunto de datos: 70% para el entrenamiento y 30% para las pruebas (ver Figura 2.1).

2.4.2.3. Resultados

Los autores realizaron un estudio experimental para validar la efectividad del modelo de detección de requisitos comparándolo contra el desempeño humano. Para la clasificación basada en reglas se evaluaron un total de 15 reglas, la primera regla desarrollada utilizó la palabra clave "*shall*" siendo esta la que más contribuyó a la clasificación basada en reglas, donde se obtuvo una precisión del 84.7%, posteriormente se agregaron dos reglas más para mejorar el rendimiento en un 4.1%, al final las ultimas 12 reglas mejoraron colectivamente el rendimiento en un 1.8% alcanzando una precisión total del 90.6%.

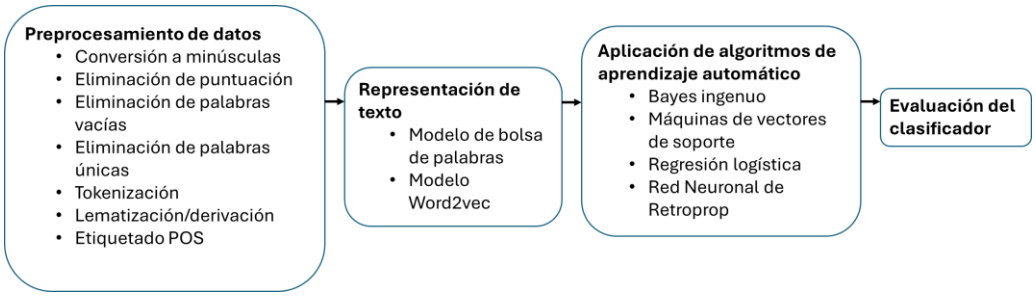

Figura 2.1. Metodología de clasificación de textos basada en el ML (traducida de Hassan y Le, 2023).

Mientras que la clasificación basada en ML, el modelo SVM demostró superioridad frente a los otros modelos en términos de exactitud, precisión, recuperación y puntuación F1. En el estudio experimental del modelo se obtuvo un valor de recuperación del 98.15% y en términos de tiempo, utilizó 1.88 segundos para extraer las declaraciones de requisitos de un documento de 9 páginas. Mientras que el método manual requirió entre 15 a 30 minutos para la misma tarea, pero obtuvo mejores en cuanto a exactitud y precisión.

2.4.3. Descubrimiento de temas no supervisado en los comentarios de usuarios

2.4.3.1. Objetivo

Stanik et al., (2021) en su trabajo de investigación identifican temas o comentarios similares que se pueden identificar como requisitos a partir de comentarios o *tweets* semánticamente similares en reseñas de aplicaciones de *App Store* y *Google Play Store*.

2.4.3.2. Descripción

En este trabajo, los autores utilizaron un enfoque automatizado como SBERT (variante de BERT, pero optimizada para generar incrustaciones) sobre un conjunto de *tweets* como entrada, para posteriormente realizar su preprocesamiento y *tokenización* y, así extraer la representación de texto de cada uno de estos. Además, crearon una representación vectorial con SBERT y a través de las representaciones de texto se agruparon los *tweets* semánticamente similares en grupos, donde cada grupo representa un tema.

Esta investigación se llevó a cabo utilizando dos fases principales: la representación de texto y la identificación de *tweets* relacionados. Para la primera fase se reprocesaron todos los *tweets* con el objetivo de eliminar información redundante y luego se calculó su representación vectorial tomando en cuenta su similitud y su agrupación. SBERT se entrenó con dos *corpus:* el de Inferencia en Lenguaje Natural (NLI, por sus siglas en inglés) y el *corpus* Similitud Textual Semántica (STS, por sus siglas en inglés) ya que estos conjuntos de datos permiten que el modelo genere incrustaciones que se aproximen, en términos de similitud de coseno, cuando dos frases son semánticamente similares. Para la fase de identificación de *tweets* relacionados, se aplicó primeramente la técnica Aproximación y Proyección Uniformes de Múltiples para la Reducción de Dimensiones (UMAP[10], por sus siglas en inglés) de reducción de dimensionalidad en las incrustaciones y posteriormente, el algoritmo Agrupación Espacial Jerárquica Basada en la Densidad de Aplicaciones con Ruido (HDBSCAN[11], por sus siglas en inglés) que determina automáticamente el número óptimo de grupos en función de los datos.

2.4.3.3. Resultados

Para evaluar el modelo se realizó una evaluación empírica con una empresa de telecomunicaciones cuyo principal canal de retroalimentación es *Twitter* (ahora X). Con el enfoque aplicado se rastrearon 138.639 *tweets* en inglés y se obtuvieron 425 temas que cubre el 23% del conjunto de datos. Se realizaron dos evaluaciones que coincidieron con el modelo temático en el 95%

[10] La UMAP es una técnica para reducir dimensiones que se usa comúnmente para visualizar datos de alta dimensión en un espacio de menor dimensión.

[11] HDBSCAN es un algoritmo para agrupar e identificar clústeres de formas y tamaños variables en un conjunto de datos.

de los casos, posteriormente se asignaron documentos a sus respectivos temas, donde coincidieron con el modelo de temas en el 98.4% de las veces en 2,000 tareas de anotación. Finalmente, los anotadores codificaron entre ellos 800 anotaciones de texto libre para evaluar si ambos tenían la misma compresión de temas alcanzando una evaluación del 91%. Finalmente, los autores integraron el enfoque de SBERT a una herramienta de código abierto para analizar continuamente *tweets* y así, poder ayudar a los profesionales a identificar informes de problemas similares, solicitudes de características y tendencias de los temas de los comentarios.

2.4.4. RE-BERT

2.4.4.1. Objetivo

De Araújo y Marcacini (2021) propusieron un modelo para identificar automáticamente los requisitos de *software* a partir de las reseñas y comentarios realizadas en diferentes aplicaciones, mismas que suelen estar escritas en un lenguaje informal.

2.4.4.2. Descripción

Los autores presentan el modelo RE-BERT basado en lingüísticas dependientes del contexto lo que hace que cubra las lagunas existentes en enfoques basados en reglas como la eficiencia, precisión, generalización y la dependencia de fuentes externas. Este modelo utiliza una estrategia de capacitación entre dominios, lo que le permite entrenarse con datos etiquetados preexistentes de diferentes aplicaciones. RE-BERT es capaz de identificar palabras semánticamente relacionadas con los requisitos del *software* mediante el uso de un modelo de lenguaje neuronal previamente entrenado para generar representaciones textuales semánticas con incrustaciones de palabras contextuales.

2.4.4.3. Resultados

Los autores realizaron una validación cruzada con múltiples dominios de aplicaciones de *eBay*, *Evernote*, *Facebook*, *Netflix*, *Photo editor*, *Spotify*, *Twitter* y *WhatsApp*, para evaluar estadísticamente el modelo propuesto donde RE-BERT obtuvo resultados prometedores al identificar correctamente los *tokens* de los requisitos de *software*. Así mismo, los resultados de la medición de evaluación F1 oscilaron en 80% y el 560%.

También se comparó el rendimiento de RE-BERT contra otros tres modelos: la extracción de requisitos de *software* que se basan principalmente en reglas lingüísticas desarrollados por ReUS[12], SAFE[13] y GuMa[14], obteniendo un rendimiento superior a ellos en 8 conjuntos de datos de reseñas de aplicaciones al extraer automáticamente los requisitos de *software*. Los autores afirman que incluso en escenarios más pesimistas, RE-BERT es competitivo en comparación con los otros modelos basados en reglas para extraer requisitos de *software*.

[12] Se refiere a reglas lingüísticas compuestas por patrones de clases gramaticales y relaciones de dependencia semántica (De Araújo y Marcacini, 2021).

[13] Utiliza la extracción de características basadas en patrones lingüísticos: 18 patrones POS y 5 patrones de frase (De Araújo y Marcacini, 2021).

[14] Utiliza un algoritmo de búsqueda de colocaciones proporcionado por el conjunto de herramientas NLTK para realizar la extracción de requisitos (De Araújo y Marcacini, 2021).

2.4.5. REConSum

2.4.5.1. Objetivo

Spijkman et al., (2023) desarrollaron REConSum, un prototipo de NLP que permite ayudar a los profesionales e investigadores a procesar conversaciones obtenidas en la elicitación de requisitos, resumiendo transcripciones de audio y extrayendo información relevante para el dominio de conocimiento.

2.4.5.2. Descripción

Los autores desarrollaron una herramienta denominada Resumen de Conversaciones de Elicitación de Requisitos (*REConSum*, por sus siglas en inglés), para resumir las transcripciones y extraer información relevante para conformar los requisitos, mediante una estructura de Preguntas y Respuestas (Q&A, por sus siglas en inglés). *REConSum* conserva únicamente preguntas relevantes y sus respectivas respuestas a través de un resumen extractivo, lo que ayuda al(a) ingeniero(a) de requisitos a analizar una versión más corta de la transcripción original. En la Figura 2.2 se puede ver la descripción general de *REConSum*.

Se considera como entrada la transcripción de una entrevista para determinar si en ella está presente una pregunta, utilizando el etiquetado de partes de la oración y/o la clasificación de actos de diálogo. Posteriormente *REConSum* determina si las preguntas son relevantes, evaluando si contiene términos específicos de un dominio.

Posteriormente determina si en la transcripción del entrevistador se incluyen términos específicos de un dominio a través de calcular la TF–IDF, para lo cual primero se calcula el IDF de la transcripción y después se calcula la Frecuencia de Términos (TF, por sus siglas en inglés) de un *corpus* de *Wikipedia*, para encontrar requisitos relevantes. Finalmente, *REConSum* conserva únicamente los turnos del entrevistador que contienen preguntas y palabras con una puntuación alta de TF–IDF.

2.4.5.3. Resultados

Para medir la efectividad de *REConSum,* se evaluó su capacidad de filtrado mediante métricas de recuperación de información estándar: precisión, recuperación, puntuación F1 y exactitud. Los autores demostraron la capacidad de su propuesta para extraer preguntas de las conversaciones de manera eficaz, con aproximadamente 63 preguntas etiquetadas como relevantes. Además, *REConSum*, destacó la relevancia de las respuestas que siguen a las preguntas, lo que permitió indicar la importancia de extraer información relevante para los requisitos a partir de las conversaciones.

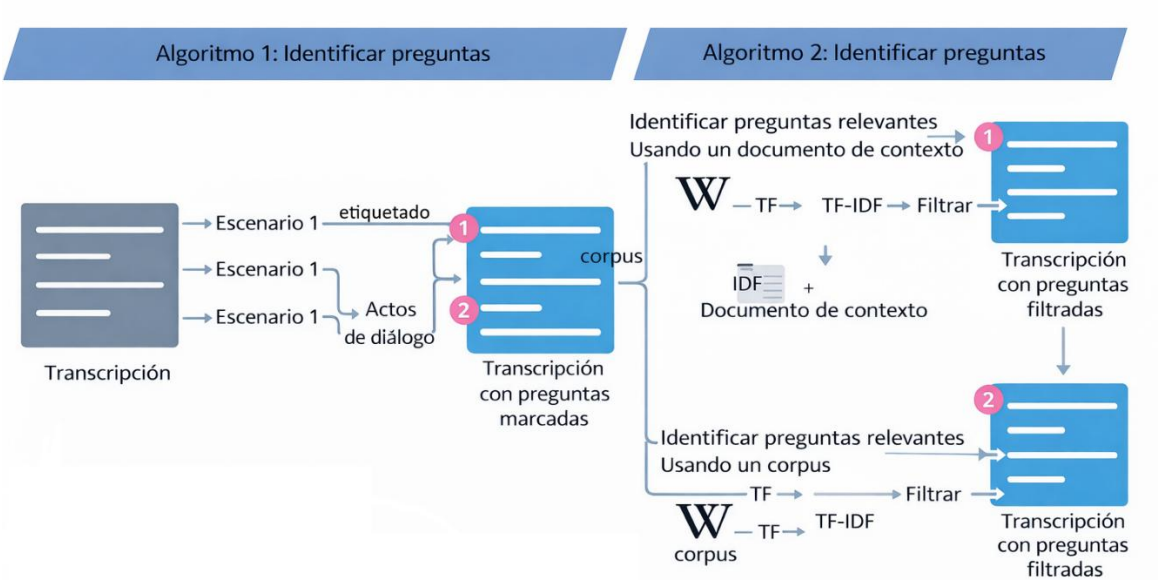

Figura 2.2. Descripción general del flujo de proceso de *REConSum* (traducida de Spijkman et al., 2023)

2.4.6. *Chatbot* para la elicitación de información contextual

2.4.6.1. Objetivo

Wolfinger et al., (2022) presentan una herramienta de *chatbot* para ayudar a los equipos de desarrollo de *software* a elicitar información contextual a partir de los comentarios de usuarios.

2.4.6.2. Descripción

En este trabajo se presenta una herramienta que permite elicitar información contextual para los comentarios mediante el uso de *chatbot*.

A través de un caso de estudio los autores identificaron las principales características que ofrece el *bot*[15].

- Categorización del planteamiento del problema: El *chatbot* identifica deferentes tipos de problemas predefinidos como caídas o fallos de funcionamiento.
- Extracción de información contextual: Mediante la conversación de los usuarios con el *chatbot* se puedo extraer información basada en las siguientes taxonomías: *hardware* (proveedor, nombre del modelo, plataforma, red), *software* (nombre del navegador, sistema operativo, versión), contenido (tipo de contenido de vídeo, ID del contenido de vídeo, calidad del contenido) y conocimiento descriptivo (problema detallado, comportamiento esperado, pasos para reproducirlo).

[15] Un *bot* proviene de la palabra de robot, es una aplicación de *software* que permite interactuar con los usuarios de una forma autónoma. Los *bots*, incluidos los *chatbots* y los *bots* de voz, tienen la capacidad de conversar con los usuarios a través del texto o la voz, y funcionan como un puente entre las personas y los servicios en diversos ámbitos como el servicio al cliente (Adamopoulou y Moussiades, 2020; Shihab et al., 2022)

- Elicitación de información faltante: Para obtener la información faltante se utilizó un formulario de Rasa[16], que pide a los usuarios información en un bucle continúo permitiendo crear una base de conocimientos independiente que incluye una lista de información contextual disponible, Lo que permitió al chat*bot* mejorar la conversación.
- Soporte a la elicitación: Si el usuario necesitaba ayuda el *chatbot* proporcionaría una frase de ayuda adecuada, o si el usuario manifestaba confusión o dudas sobre la información que se le pedía, el *chatbot* le proporcionaría una explicación adicional.
- Notificaciones de incidencias: El *chatbot* permitió a los equipos de desarrollo a obtener las opiniones de los usuarios y almacenarlo en una base de datos.

2.4.6.3. Resultados

Para el entrenamiento del *chatbot* se utilizó el acceso para desarrolladores de *Google Play* de la aplicación del caso de estudio para descargar todos los comentarios de los usuarios de los 30 días anteriores. Además, se utilizó un Desarrollo Basado en Conversación (CDD, por sus siglas en inglés) en conjunto con NLU, obteniendo un total de 1200 frases de ejemplo. Los resultados de la validación mostraron que el *chatbot* podía detectar con precisión 96.8% de la información contextual, mientras que con una encuesta tradicional únicamente se encontraba el 31.3%.

2.4.7. *ChatGPT* para ayudar en los procesos de elicitación de requisitos

2.4.7.1. Objetivo

Ronanki et al., (2023) presentan en su trabajo de investigación una exploración del potencial de *ChatGPT* (versión del 9 de febrero de 2023) en los procesos de elicitación de requisitos, mediante la formulación de seis preguntas para obtener los requisitos.

2.4.7.2. Descripción

En este trabajo de investigación, los autores emplearon un proceso de dos fases para evaluar la calidad de los requisitos elicitados por el *ChatGPT*.

La primera fase consistió en recopilar datos sintéticos, que en este caso fueron requisitos para desarrollar sistemas de IA, para lo cual se formuló un cuestionario con 6 preguntas para el *ChatGPT*, que permitiera la elicitación de requisitos; sin embargo, antes de aplicarlo se proporcionó un contexto de la información que se quería obtener, con el objetivo de garantizar de que se estuvieran elicitando los requisitos correctos.

La segunda fase que consistió en realizar entrevistas a 5 *stakeholders* con conocimientos en el área de IR para sistemas de IA/ML, para conocer los requisitos para desarrollar un sistema de IA confiable. A estos expertos se les hicieron las mismas 6 preguntas que se le dieron al *ChatGPT*.

Se obtuvo un total de 36 respuestas, 30 de los expertos y 6 de *ChatGPT*. Estas 36 respuestas fueron analizadas por evaluadores expertos, y para evitar posibles sesgos, no se les reveló que había 6 respuestas generadas por el *ChatGPT*.

[16] Rasa es una herramienta que se utiliza para el desarrollo de *chatbots*, particularmente en varios dominios de aplicación, como la educación, las tareas de mantenimiento y los protocolos de interacción (Ferrando et al., 2023)

2.4.7.3. Resultados

Los evaluadores proporcionaron una puntuación para cada respuesta, en una escala de 0 a 10 para cada atributo de calidad (abstracción, atomicidad, coherencia, corrección, falta de ambigüedad, comprensibilidad y viabilidad), siendo 0 la puntuación más baja y la 10 la puntuación más alta.

Los resultados obtenidos indicaron que, en los requisitos generados por el *ChatGPT,* la corrección, comprensibilidad y la consistencia de los requisitos, fueron las cualidades más destacadas, mientras que la ambigüedad y la viabilidad fueron la menos notorias. Por lo que pudieron concluir que el elicitar requisitos no ambiguos e inviables fue una clara deficiencia identificada en el uso de *ChatGPT*. Sin embargo, en la mayoría de los casos, los puntajes de evaluación de la calidad de requisitos elicitados por el *ChatGPT* superaron los puntajes obtenidos en los requisitos elicitados de los expertos, por lo que los evaluadores consideraron aceptables los requisitos generados por el *ChatGPT* en competencia directa con requisitos formulados por los expertos.

2.4.8. Consideraciones finales sobre los trabajos relacionados

Como se ha expuesto en este capítulo, la elicitación de requisitos preliminares constituye una fase fundamental en el proceso de desarrollo de *software*. Diversos autores han propuesto herramientas orientadas a automatizar esta etapa, centrándose en el análisis y clasificación de información derivada de comentarios realizados por los *stakeholders*, ya sea durante las sesiones de elicitación o almacenados en plataformas como *Google Apps* (e.g., Panichella y Ruiz (2020); Hassan y Le (2020); Stanik et al. (2021); De Araújo y Marcacini (2021); Wolfinger et al. (2022); Ronanki et al. (2023)). Adicionalmente, se han desarrollado investigaciones dirigidas a resumir automáticamente las conversaciones obtenidas de los *stakeholders* a lo largo de la elicitación, permitiendo sintetizar y extraer información relevante (e.g., Spijkman et al. (2023)).

Sin embargo, continúa existiendo una brecha considerable en lo referente a la extracción de FRs preliminares a partir de resúmenes generados de manera automática basados en los documentos proporcionados por los *stakeholders*. Esto representa una posibilidad para aportar innovación al área de la IR, particularmente en la etapa de elicitación. Por consiguiente, esta tesis propone el desarrollo de una herramienta computacional capaz de generar resúmenes automáticos a partir de documentos de requisitos preliminares, así como la identificación y extracción de los FRs preliminares pertinentes, con el objetivo de optimizar el tiempo y esfuerzo dedicados a estas tareas.

3. Desarrollo de la solución

Como se ha mencionado en los capítulos anteriores, la elicitación de requisitos es un proceso crítico en el desarrollo de *software*, requiere una comunicación continua entre el(la) ingeniero(a) de requisitos y los *stakeholders*. Sin embargo, lo(a)s ingeniero(a)s de requisitos pueden enfrentar desafíos relacionados con percepciones y significados subjetivos que los *stakeholders* puedan asociar a determinadas palabras u oraciones, dado que el NL, principal medio de comunicación, es susceptible a diversas interpretaciones (Pressman y Maxim, 2019; Memon y Xiaoling, abril de 2019).

Uno de estos desafíos es la obtención del conocimiento tácito de los *stakeholders* ya que se basa en las acciones y experiencias de cada individuo dependiendo del momento y entorno particular (Jafar et al., 2023). La complejidad en la obtención del conocimiento tácito aumenta cuando hay varios *stakeholders* involucrados, ya que las opiniones de todo(a)s ello(a)s tienen que ser analizadas a detalle para comprender sus puntos de vista, expectativas y prioridades. El análisis y comprensión, por parte del o la ingeniera (o) de requisitos, de las diferentes perspectivas de los *stakeholders* es crucial para evitar que algún requisito importante se omita o se malinterprete en el proceso de desarrollo del *software* (Pressman y Maxim, 2019; Anwar, et al., 2022). Cuando el(la) ingeniero(a) de requisitos no logra expresar y documentar los requisitos de manera adecuada, los demás miembros del equipo de desarrollo pueden tener dificultades para implementarlos, lo que ocasiona, a menudo, que se puedan producir graves fallos en el *software*. Esta problemática se agrava cuando lo(a)s desarrolladore(a)s de *software* no tiene conocimientos sólidos sobre el dominio del problema a resolver, por lo que surge la fluctuación entre el conocimiento explícito y el tácito (Jafar et al., 2023).

El ciclo de vida del desarrollo de *software* inicia con la elicitación de requisitos, etapa en la que se definen los requerimientos preliminares que conformarán la Especificación de Requisitos del *Software*. Dado que las necesidades y expectativas de los *stakeholders* se transmiten al(la) ingeniero(a) de requisitos tanto de manera explícita como tácita, es esencial establecer requisitos claros y precisos para asegurar la viabilidad y éxito del proyecto (Popoola et al., 2024). En relación con los FRs, debido a que estos especifican la funcionalidad del *software*, resulta imprescindible expresarlos sin ambigüedades ni inconsistencias; de lo contrario, podría comprometerse la comprensión e implementación por parte del equipo de desarrollo, generando posibles reprocesos, defectos o retrasos en la entrega del proyecto (Jafar et al., 2023).

En este capítulo se presentan los principales componentes que conforman el desarrollo de la propuesta presentada en esta tesis:

- Generación de resúmenes extractivos a partir de textos de requisitos preliminares, obtenidos mediante técnicas de elicitación como entrevistas y reuniones, así como su respectiva evaluación;
- La identificación de los FRs preliminares y,

- El diseño y construcción de la herramienta *software* destinada a la validación de dichos elementos.

3.1. Generación de resúmenes Extractivo de Texto

La generación de resúmenes extractivos de texto consiste en seleccionar oraciones destacadas de un texto, llamado original, para formar una versión abreviada concisa y coherente, reutilizando partes del texto original (i.e., oraciones). Cada oración combina información relevante, dependiendo del dominio, por lo que en el resumen deben reordenarse conservando las reglas gramaticales (Abdel-Salam y Rafea, 2022).

De acuerdo con El-Kassas et al. (2021), Yadav et al. (2021), Abdel-Salam y Rafea (2022), Abo-Bakr y Mohamed (2023) la generación de un resumen automático extractivo de un texto se divide principalmente en tres fases: preprocesamiento, procesamiento algorítmico y posprocesamiento. A continuación, se describen cada una de estas fases.

3.1.1. Preprocesamiento

El Preprocesamiento de Texto (TP, por sus siglas en inglés) implica estandarizar textos de entrada (archivos) y establecer delimitaciones entre las secuencias de palabras. Para Alami Merrouni et al. (2023) la forma en la que se ejecuta el TP puede influir significativamente en la precisión del resultado final. De acuerdo con Abdel-Salam y Rafea (2022), Abo-Bakr y Mohamed (2023) y Alami Merrouni et al. (2023) algunas técnicas del preprocesamiento de texto son las siguientes:

- División o segmentación de oraciones: Implica en dividir el texto de entrada en oraciones individuales. En una colección de textos se segmenta el conjunto de oraciones.
- *Tokenización*: Es el proceso de dividir un flujo de texto en palabras, oraciones, símbolos u otros elementos significativos. Por ejemplo, si se tiene el siguiente texto "*el sistema deberá enviar notificaciones por correo electrónico cuando se complete una tarea*", una vez realizada la *tokenización* se obtiene lo siguiente: "*el*", "*sistema*", "*deberá*", "*enviar*", "*notificaciones*", "*por*", "*correo*", "*electrónico*", "*cuando*", "*se*", "*complete*", "*una*", "*tarea*".
- Identificación y eliminación de palabras vacías: Se denominan palabras vacías aquellas que no tienen un significado relevante en el texto, tomando en cuenta el dominio de conocimiento. Estas palabras deben eliminarse de las oraciones, ya que no tienen ningún efecto al momento de extraer las palabras significativas. Por ejemplo, de la siguiente oración "*El sistema deberá permitir a los usuarios crear textos en línea*" se obtiene la nueva oración "*sistema deberá permitir usuarios crear textos línea*".
- Lematización o derivación de palabras: El proceso consiste en reducir las palabras a su raíz o forma base (e.g., *perro, perros, perra, perrito, perrote*, etc. se reducirían a su raíz *perr*), este proceso "trunca" los afijos, prefijos y postfijos de las palabras.

3.1.2. Procesamiento algorítmico del texto

Esta fase consiste en aplicar técnicas de NLP para la generación automática del resumen. Como primer paso se debe realizar la extracción de las características de las oraciones que aparecen en el texto fuente (i.e., verbo, adjetivo, preposición, adverbio, etc.). De acuerdo con Mridha (2021) las técnicas que se utilizan para esta tarea, con el fin de calcular la puntuación de una oración y determinar su relevancia en la generación del resumen, son las siguientes:

- **TF:** Se utiliza para determinar la importancia de los términos en un solo texto, aunque comúnmente se usa para representar el valor de una palabra.
- **TF-IDF:** Permite medir los términos comunes en el texto, y en conjunto con la TF ayuda a capturar la importancia contextual y la relevancia de los términos dentro de un texto.
- **Similitud entre oraciones:** Consulta la similitud entre las oraciones del texto y extrae las características con base al título, la posición de la oración, términos temáticos asociados con oraciones específicas del dominio y datos numéricos.

Cuando se han extraído las características de las oraciones se realiza la representación de éstas como vectores (i.e., se les asigna un valor con base en su relevancia e importancia dentro del texto), para así encontrar una oración "elemental". La representación de oraciones consiste en convertir las palabras en números (0 y 1) para que la computadora pueda comprender y decodificar patrones dentro del texto, dependiendo del idioma (El-Kassas et al., 2021; Mridha, 2021). De acuerdo con Ledeneva y García Hernández (2013), y Mridha (2021), algunas de las técnicas para la representación de texto son las siguientes:

- **N-gramas*:*** Un n-grama[17] es una colección de palabras o caracteres con N componentes, utilizando unigramas, bigramas, trigramas, cuádruples y otros n-gramas que componen un conjunto de N-gramas de texto y se pueden representar como vectores de términos.
- **BoW:** Implica representar los datos del texto utilizando la frecuencia con que las palabras aparecen en un texto, sin tener en cuenta el orden o la estructura de estas.
- **TF-IDF:** La TF mide la frecuencia con la que aparece una palabra en una oración, mientras que el IDF mide la importancia de la palabra en el texto.
- **Incrustación de palabras:** Representan palabras como vectores de números, donde a cada palabra se asigna a un vector N-dimensional de valores absolutos.

Por último, para generar automáticamente el resumen extractivo, a partir de un texto preprocesado, se puede aplicar un enfoque algorítmico o de modelos, como los que se describen a continuación.

3.1.2.1. Algoritmos para la generación automática de resúmenes extractivos

De acuerdo con Abdel-Salam y Rafea (2022), y Giarelis et al., (2023) existen diversos algoritmos para la generación de resúmenes extractivos como *Luhn*, *LSA*, *Edmundson*, *TextRank*, *LexRank*, *PositionRank*, y *TopicRank*, mismos que son descritos, brevemente, a continuación.

3.1.2.1.1. Algoritmo de Luhn

Luhn es un algoritmo heurístico para resumir texto que se centra en identificar y extraer palabras que tengan una frecuencia alta de aparición, evaluando así su importancia - ya que es probable que las palabras con una frecuencia alta estén asociadas con el tema principal del texto- a excepción de las palabras vacías, que son aquellas no aportan información y por eso deben ser eliminadas (Sharma et al. 2021; Lee, 2023). El algoritmo de *Luhn* se basa en la percepción de que algunas palabras son descriptivas del contenido de un texto, y, por ende, las oraciones que contienen esas palabras

[17] Un N-grama es una representación de un texto que utiliza una secuencia de N palabras o N caracteres (n-grama de caracteres), donde N puede ser cualquier número. Así, se puede tener 1 grama (unigrama), 2 gramas (bigrama), 3 gramas (trigrama), etc. (Zhao et al. 2022).

descriptivas son las que deben extraerse para formar el resumen, puesto que contienen la información más significativa del texto (Dutta et al., 2019). Para identificar las palabras descriptivas se utilizan umbrales de alta y baja frecuencia, determinados empíricamente (i.e., los umbrales de alta frecuencia filtran palabras que aparecen con frecuencia en todo el texto, mientras que los de baja frecuencia filtran las palabras que aparecen pocas veces). Este algoritmo se utiliza a menudo en la generación de resúmenes de texto junto con técnicas de vectorización de características, así como esquemas de ponderación TF-IDF para las tareas de clasificación de texto y de textos (Van Zachary, 2023).

3.1.2.1.2. Algoritmo de Análisis Semántico Latente (LSA)

El Análisis Semántico Latente (*LSA*, por sus siglas en inglés) se basa en un algoritmo gramatical que analiza el texto y construye una estructura sintáctica del mismo, seleccionando y reordenando su subestructura, generando un resumen extractivo a partir de una identificación de oraciones semánticamente importantes (Lee, 2023). El *LSA* es un algoritmo que extrae y representa el significado textual de las palabras combinando técnicas algebraicas y estadísticas que permiten calcular la similitud entre palabras, oraciones y textos (Mandal y Singh, 2020; Gupta y Patel, 2021). El *LSA* compara y extrae información sobre palabras que se usan juntas, así como de palabras comunes que se han identificado en diferentes oraciones, ya que si existe un número alto de éstas significa que las oraciones están relacionadas semánticamente - (Dutta et al., 2019). Este algoritmo utiliza esquemas de ponderación TF-IDF para analizar el texto y la Descomposición en Valores Singulares (SVD, por sus siglas en inglés) para descomponer matricialmente los términos de este, así como para descubrir las interrelaciones entre oraciones y palabras (Gupta y Patel, 2021). El texto de entrada u original se convierte inicialmente en una matriz donde las filas representan una palabra y las columnas una oración, y el valor de la celda representa la importancia de la palabra en el texto, donde al aplicar la SVD se seleccionan las oraciones que conformaran el resumen (Dutta et al., 2019).

3.1.2.1.3. Algoritmo de Edmundson

Edmundson se considera una mejora del algoritmo de *Luhn*, pero depende del lenguaje utilizado en el texto, ya que hace uso de varios componentes como, por ejemplo, las palabras pragmáticas (i.e., palabras clave o *key words*), palabras del título y encabezado, así como la ubicación de la oración dentro del texto; lo que permite identificar “palabras adicionales” - palabras que el usuario considera importantes (positivamente relevantes)-, “palabras estigma”- palabras de poca importancia o incluso de importancia negativa (negativamente relevantes)-; y “palabras vacías” o “palabras nulas” (i.e., palabras irrelevantes) (Verma et al., 2019; Manojkumar et al. 2023).

Este algoritmo propone el uso de una combinación ponderada de las características de las palabras (e.g., posición, frecuencia, si son clave, estructura del texto, etc.) para extraer las oraciones con mayor relevancia dentro de un texto - asignándoles un peso que refleje su importancia relativa en este-, y así transmitir en sus resúmenes la idea del texto original (Freire et al. 2024). El cálculo del peso de las oraciones depende del tipo de texto utilizado (e.g., artículos, noticias, entre otros.), ya que dependiendo del contexto del texto es como se calcula la importancia de las oraciones (Verma et al., 2019).

3.1.2.1.4. Algoritmo TextRank

Este algoritmo genera un resumen de un texto mediante la clasificación de sus oraciones, esto se realiza a través de la teoría de grafos. El texto para analizar se representa como un grafo ponderado donde las oraciones representan a los nodos y las relaciones que hay entre ellas a las aristas. Por lo tanto, una conexión entre dos oraciones indica que tienen una similitud. Una vez que el grafo ha sido

creado, se aplica el algoritmo *PageRank*[18] para clasificar cada oración en función de sus conexiones con las demás. Finalmente se seleccionan las oraciones mejores clasificadas para generar así el resumen de texto (Giarelis et al., 2023).

3.1.2.1.5. Algoritmo LexRank

Este algoritmo utiliza el *PageRank,* por lo que también está basado en grafos. Sin embargo, su principal diferencia es que cada oración del texto se representa como un vector en función de la TF-IDF, y la relación o similitud entre estos vectores, se lleva a cabo utilizando la similitud del coseno. Posteriormente se crea una matriz de similitud donde se muestra qué tan parecidas son las oraciones entre sí (Giarelis et al., 2023).

3.1.2.1.6. Algoritmo TopicRank

Este algoritmo utiliza técnicas de modelado de temas, es decir agrupa las oraciones de un texto, mediante la similitud de los temas que representan, y extrae las oraciones más importantes de cada grupo para así generar el resumen. Para realizar la extracción de oraciones, primero se preprocesa el texto realizando una segmentación de las oraciones, y posteriormente se lleva a cabo la *tokenización* de las palabras, así como el etiquetado de las categorías gramaticales para agruparlas en temas, y como último paso se clasifican según su importancia dentro del texto de acuerdo con su tema. El resumen se genera seleccionando y extrayendo las oraciones candidatas más importantes de acuerdo con el tema principal del texto (Bougouin et al.2013; Giarelis et al., 2023).

3.1.2.1.7. Algoritmo PositionRank

Este algoritmo considera la distribución de la posición de las palabras dentro de un texto, así como sus frecuencias y las clasifica utilizando el algoritmo *PageRank* (Giarelis et al., 2023). El algoritmo *PageRank* permite ajustar el cálculo de la importancia de las palabras de acuerdo con el contexto del texto, con la finalidad de extraer las palabras claves a partir de la información de todas las posiciones de ocurrencia de una palabra (Thushara et al., 2019).

3.1.2.2. Modelos pre entrenados para generación automática de resúmenes extractivos

Es importante mencionar que el uso de modelos en la generación de resúmenes extractivos ha demostrado un alto nivel de adecuación e informatividad de los resúmenes. En este sentido, el modelo BERT es el mayormente utilizado en áreas como en motores de búsqueda o en el NLP para generar resúmenes, ya que como modelo de IA tiene mayor precisión, informatividad y la capacidad de procesar grandes volúmenes de texto de forma eficiente (Kuznietsov y Kyselov, 2024).

Los modelos de lenguaje pre entrenados son una tecnología clave para llevar a cabo las tareas del NLP, como por ejemplo la identificación de una similitud textual, la generación de preguntas a partir de un texto, o bien, la inferencia textual del NL; sin que se requieran cambios significativos en el modelado de la arquitectura de cada tarea específica (Kenton y Toutanova, 2019). Estos modelos se utilizan generalmente para mejorar el rendimiento en tareas de comprensión del NL, ya que extienden la idea de aprender representaciones contextuales mediante la incrustación de palabras en un texto.

[18] Este algoritmo clasifica las oraciones de un texto con base a su conexión con otras oraciones (Giarelis et al., 2023). *PageRank* construye el grafo que generará el resumen utilizando las oraciones como nodos, sus conexiones como aristas y la similitud entre las palabras u oraciones como bordes o límites. *PageRank* es utilizado para calcular la importancia de las páginas *web* conectadas o enlazadas entre sí (Jiang et al, 2017; Elbarougy et al., 2020).

3.1.2.2.1. Modelo BERT

BERT es una pila de codificadores con arquitectura de transformadores[19], entrenados previamente, que pueden comprender los datos textuales a través de mecanismos de "atención", lo que lo convierte en un modelo de lenguaje contextualizado. Así mismo, BERT genera un vector de salida para cada *token* con información contextual enriquecida y se entrena con un modelado de lenguaje "enmascarado" y una tarea de "predicción de la siguiente oración" (Liu y Lapata, 2019; Abdel-Salam y Rafea, 2022).

3.1.3. Posprocesamiento del texto

En este punto se reordenan las oraciones extraídas del texto original o de entrada y se reemplazan las expresiones temporales relativas como fechas reales, etc. Por ejemplo, las expresiones ("ayer", "mañana", "la próxima semana") son reemplazadas por fechas absolutas, con el objetivo de mejorar la claridad y precisión del texto.

- Texto: "La reunión será mañana."
- Posprocesamiento: "La reunión será el 15 de junio de 2024."

Así mismo, se mejora la coherencia y legibilidad, así como la secuencia lógica del resumen generado, permitiendo que este sea comprensible para el lector. También puede incluir otros ajustes, tales como la corrección gramatical, la eliminación de redundancias y la mejora de la fluidez del texto (El-Kassas et al., 2021; Abo-Bakr y Mohamed, 2023).

Después del posprocesamiento, se realiza la evaluación del resumen para asegurar que los resúmenes automáticos no solo sean técnicamente correctos, sino también útiles y relevantes para los usuarios finales (El-Kassas et al., 2021; Abo-Bakr y Mohamed, 2023).

En la siguiente sección, se mencionan estos tipos de evaluaciones:

3.1.4. Métricas de evaluación

Una vez que el resumen ha sido generado este debe ser evaluado para garantizar que no solo sea técnicamente correcto, sino también útil y relevante para los usuarios finales (El-Kassas et al., 2021; Abo-Bakr y Mohamed, 2023). De acuerdo con El-Kassas et al., (2021) y Akter et al., (2022) la métrica más utilizada para evaluar un resumen de texto es ROUGE, principalmente porque permite cuantificar la precisión de un resumen sin requerir una evaluación humana continua, lo cual facilita la comparación de diferentes modelos o técnicas.

ROUGE agrupa un conjunto de métricas que permiten comparar los resúmenes de texto generados por una computadora contra resúmenes de referencia (i.e., creados por personas). Realiza la evaluación mediante el cálculo de la superposición de coincidencias de las unidades léxicas, es decir, mide la similitud entre el resumen generado automáticamente y el de referencia. ROUGE compara cuántas palabras u oraciones coinciden en ambos resúmenes, ya sea en términos de palabras individuales, pares de palabras consecutivas o coincidencias de oraciones más largas (Tsai et al., 2021). De acuerdo con Gambhir y Gupta (2017) y Alami et al. (2021) las métricas ROUGE utilizadas para este tipo de evaluación son las siguientes:

[19] Una arquitectura de transformador o *transformer* es una red de codificador-decodificador que utiliza la "autoatención" en el lado del codificador y la "atención" en el lado del decodificador, es decir, un codificador que lee el texto de entrada y un decodificador que produce una predicción para la tarea.

- ROUGE-1: Se basa en medir el unigrama entre el resumen generado automáticamente y el de referencia. Una puntuación alta indica que el resumen automático tiene una mejor cobertura del contenido del texto original (Tsai et al., 2021).
- ROUGE-N mide las unidades de N-grama comunes entre un resumen generado automáticamente y una colección de resúmenes de referencia, donde N determina la longitud del N-grama (e.g., ROUGE-2 que evalúa la superposición de los bigramas).
- ROUGE-L calcula la métrica de Subsecuencia Común más Larga (LCS, por sus siglas en inglés). La LCS es el tamaño máximo de la subsecuencia común para dos secuencias determinadas, x e y. ROUGE-L calcula la relación entre el tamaño de la LCS de dos resúmenes y el tamaño del resumen de referencia.

Es importante mencionar que ROUGE permite evaluar cuantitativamente el número de palabras compartidas por un resumen "candidato" (i.e., generado automáticamente) y un resumen de referencia, utilizando para ello las medidas de precisión y recuperación (Akhmetov et al. (2022); Barbella y Tortora (2022); Muniraj et al., (2023)):

- La precisión determina qué tan conciso es el resumen generado, es decir que únicamente debe incluir palabras necesarias en el resumen final.
- La recuperación determina qué parte del resumen generado está en el resumen de referencia elaborado por una persona.
- Medida-F es la combinación de la recuperación y la precisión, por lo que su resultado es una puntuación equilibrada de ambas medidas.

En resumen, las métricas ROUGE (ROUGE-1, ROUGE-N y ROUGE-L) proporcionan diferentes enfoques para evaluar la similitud entre un resumen generado automáticamente y un resumen de referencia, al brindar información sobre la calidad y la relevancia del contenido generado (Takeshita et al., 2024).

3.2. Evaluación de los algoritmos TextRank, LexRank, LSA Luhn, Edmundson, TopicRank, PositionRank y el modelo BERT para la generación automática de resúmenes de texto

Para la implementación de los algoritmos de generación automática de resúmenes extractivos (*TextRank, LexRank, LSA Luhn, Edmundson, TopicRank, PositionRank y BERT*) se utilizaron las siguientes librerías gratuitas y de código abierto en el lenguaje de programación *Python* (versión 3.12.7):

- *Sumy* (versión 0.11.0) permite generar resúmenes automáticos extractivos a través de encontrar las oraciones más significativas del texto, incluye los algoritmos como *TextRank, LexRank, LSA Luhn, Edmundson,* y sus módulos de *tokenización* y *lematización*.
- *SpaCy* es una librería del NLP para el análisis de texto, ofrece funciones avanzadas como la *tokenización*, el etiquetado gramátical, el análisis sintáctico y la clasificación de texto.
- *PyTextRank* que es una implementación del *TextRank* en *Python* como una extensión de *SpaCy*.

- *Hugging Face Transformers* (versión 4.48.1) que proporciona modelos previamente entrenados para diversas tareas de ML, específicamente para la implementación del modelo BERT en *Python* (versión 1.13.1)[20].
- *NLTK* (versión 2.10.1) que es un *kit* de herramientas del NL diseñado para realizar tareas del NLP como la *tokenización*, *lematización*, etiquetado, análisis sintáctico, clasificación, razonamiento semántico y análisis de sentimientos. En este caso en particular del trabajo de tesis, fue utilizada para segmentar las oraciones.
- *Pandas* que es utilizada para la manipulación y el análisis de datos (versión 2.2.3) y,
- *NumPy* (versión1.26.4) especializada en el cálculo numérico y el análisis de datos, especialmente para grandes volúmenes de datos.
- *Python* es de código abierto y ofrece ventajas significativas para desarrollar tareas del NLP a través de la implementación de una amplia variedad de librerías como *Scikit-learn, NLTK, Pandas, Numpy,* etc. facilitando así tanto el procesamiento como el manejo de datos (Dhruv et al., 2021).

Ahora bien, una vez implementados tanto los algoritmos *Luhn, LSA, Edmundson, TextRank*[21]*, LexRank, PositionRank,* y *TopicRank* como el modelo BERT se procedió a evaluar su desempeño en la generación del resumen extractivo utilizando los casos de estudios del libro de *Requirements Engineering: From System Goals to UML Models to Software Specifications* (Lamsweerde, 2009).

A continuación, se describen estos tres casos de estudios:

Caso de estudio 1

Case Study 1: Library Management

The University of Wonderland (UWON) wants to convert its library system into a new system to ensure more effective access to state-of-the-art books, periodicals and proceedings while reducing operational costs. The current system consists of multiple unconnected library subsystems, one for each UWON department. Each department subsystem is responsible for its own library according to department-specific procedures for book acquisition, user registration, loan management, bibliographical search and access to library resources. Such services are essentially manual in most UWON libraries. They rely on card indexes maintained by library staff according to some keyword-based classification scheme. Such schemes are specific to each department. A few departments are using rudimentary file-based software written by their members.

Some of the complaints about the current system as reported by university authorities, library staff, department members or students include the following:

- *Unnecessary duplicate acquisition, by several departments, of infrequently accessed copies of books or proceedings that are relevant to more than one department.*
- *Unnecessary subscription, by several departments, to expensive journals that are relevant to more than one department.*
- *Acquisition of books or proceedings of marginal interest to the university, which could be borrowed from other universities with which UWON has an agreement.*

[20] Es importante mencionar que se utilizó la versión de *Phyton* (versión 1.13.1) ya que la versión (3.12.7) no era compatible con esta librería.

[21] Para este algoritmo se utilizaron las implementaciones respectivas de las bibliotecas *Sumy* y *pyTextRank*.

- *Subscription to journals of marginal interest to the university, which could be accessed in other universities with which UWON has an agreement.*
- *Unavailability of requested books, for a variety of reasons such as department budget restrictions, excessive borrowing by the same user, lack of enforcement of rules limiting loan periods, loss or stealing of book copies and so on.*
- *Unavailability of journal issues while they are being bound into yearly volumes.*
- *Lack of traceability to previous borrowers when books, proceedings or journal volumes are found to be damaged.*
- *Inaccuracy of card indexes, e.g. a book is stated as being available whereas it is not found at the appropriate place in the shelves.*
- *Bibliographical search restricted to library opening hours.*
- *Slow, tedious bibliographical search due to manipulation of card indexes.*
- *Inaccurate search results, due to poor classification of books, journals or proceedings within departments.*
- *Incomplete or ineffective search results, due to relevant books, journals or proceedings being indexed in other UWON department libraries, or unavailable at UWON.*

The new UWON library system should address such problems through a software-based solution integrating all department libraries. The new system should interoperate with library systems from partner universities.

It should provide interactive online facilities for book acquisition, user registration, loan management, bibliographical search and book reservation. Access to such facilities should be restricted to specific user categories, according to authorization rules specific to each facility.

The new system should take advantage of opportunities provided by new technologies. In particular, it should support subscriptions to e-journals, provide access to foreign digital libraries (under specific conditions), support e-mail communication between staff and users, enable bibliographical search from anywhere at any time, and provide a Web-based interface for book e-seller comparison, selection, and order submission.

Caso de estudio 2

Case Study 2: Train Control

Traffic at Wonderland airport (WAX) has increased drastically over the past few years. The increase in the number of companies and flights calls for the building of new terminals. The increase in the number of passengers calls for a new transportation system between all terminals, and between the main terminal and Wonderland City. The current bus transportation system has reached its limits in terms of transportation capacity and quality of service. Buses are slow and often late due to traffic jams; passengers need to stand in long queues, sometimes for an unacceptably long time, which may cause them to miss flight connections, and so on.

The government of Wonderland has decided to replace bus transportation by a train-based system. The envisaged system is aimed at increasing transportation capacity, speed and quality of service. The decision is also motivated by recent regulations for reducing greenhouse gas production.

Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency and better information to passengers.

A consortium has been set up to undertake this project. It brings together government representatives, airport authorities, Wonderland Railways and the engineering company selected to implement the project. The latter is subcontracting the software part of it to a software house.

In the new system, all terminals will be interconnected through an underground circular, one-track railway. The main terminal and city terminal will be interconnected by a two-track line (one for each direction). The main terminal also has parking tracks for inactive trains, servicing and so on. Each track is divided into track segments of a fixed size called blocks. Each terminal holds one block called a station block. Each block is equipped with an entry signal (or 'virtual gate') and multiple sensors to detect the presence of trains, identify trains and their speed and so on.

The envisioned software is expected to control the acceleration of trains, the opening of train doors, the block signals and the display of information about the current/next station on information panels inside trains. The railway company would also like to reduce operational costs. A fully automated, driverless option is envisaged as an alternative to the standard option. In this standard option, train drivers have to follow recommendations issued by the software and respond to regular stimuli issued by the software to check driver responsiveness. The driverless option is currently being discussed with the unions.

Various concerns about the new system have already emerged at this preliminary stage:

- *In order to ensure rapid transportation of passengers, trains should run fast, without unnecessary delays, and at high frequency, during rush hours at least.*
- *In order to guarantee safe transportation of passengers, the probability of accidents must fall below the threshold imposed by safety regulations. In particular, the distance between two trains following each other must always be sufficient to prevent the back train from hitting the front train in case the latter stops suddenly. The speed of a train on a particular block may never exceed the limit associated with that block. Trains may never enter a block whose entry signal is set to 'stop'. Train doors must always be kept closed while a train is moving.*
- *In order to ensure comfortable transportation, trains should accelerate/decelerate smoothly. Passengers at a station should be informed in time about trains arriving. Passengers inside a train should be informed in time that the train is departing, which companies are being served by the next stop and so on.*

Caso de estudio 3

Case Study 3: Meeting Scheduling*

With the advent of globalization, companies and organizations are increasingly distributed over multiple sites and countries. Wonderland Software Services (WSS) has identified a large potential market for meeting-scheduling software that would exploit Internet-based communication technologies. Scheduling meetings with busy people is generally a nightmare. It is hard to find a date and a place that suit everyone's constraints; meeting organizers need to pester people to get their availability; other people are unnecessarily inconvenienced by messages that do not concern them; when the meeting is scheduled some constraints have changed in the meantime; new scheduling cycles need to be repeated when no date/location is found in a reasonably short period; and so forth. As a result, meetings tend to be organized poorly and late; important people sometimes do not show up; and there is a significant, unnecessary overhead in the scheduling process.

Meetings are typically scheduled as follows. A meeting initiator informs potential participants about the need for a meeting and specifies a date range within which the meeting should take place, asking people to return their availability constraints within that time interval. Constraints are

typically expressed as two sets: an exclusion set specifying dates within the date range when the participant could not attend, and an optional preference set specifying dates within the date range on which the participant would prefer the meeting to take place (a date may refer to a full day or a period in a day). In some cases, the initiator may also ask participants who will play an active role in the meeting for specific requirements regarding the meeting room (e.g. projector, laptop, network connection, videoconferencing facilities etc.). 'Important' participants may optionally be asked to state preferences for meeting locations.

The scheduled meeting date should belong to the stated date range and to none of the exclusion sets; it should ideally belong to as many preference sets as possible. The meeting venue should ideally fit the preferences of important participants. A date conflict occurs when no date can be found outside all exclusion sets. A room conflict occurs when no room can be found, at any date outside all exclusion sets, which meets the room requirements. Conflicts can be resolved in several ways: the initiator may extend the date range, some participants may remove dates from their exclusion set, or some participants may decline the invitation to attend. A new scheduling cycle may thus be required in case of conflict.

The envisioned meeting scheduler software should reflect as closely as possible the way meetings are typically managed. It should be useable by administrative staff and provide major improvements in several respects:

- *Average participant attendance should increase thanks to the selection of meeting dates and locations that are the most convenient to potential participants.*
- *Meetings should be scheduled as quickly as possible once they are initiated.*
- *Meeting dates and locations should be notified as quickly as possible to all potential participants once they are scheduled. In all cases, there should be sufficient time between notification and the meeting date.*
- *The organizational overhead should be kept as low as possible on the initiator's side. In particular, the meeting scheduler should support all required interactions with participants, for example to communicate requests, get replies (even from participants not reacting promptly), assist in negotiation and conflict-resolution processes, and inform participants on request about the state of the scheduling process.*
- *The amount of interaction with potential participants for meeting scheduling, in number and length of messages, should be kept as small as possible.*

The new meeting scheduler must be able to handle multiple meeting requests in parallel. Meeting requests can be competing by overlapping in time or space. Concurrency must thus be managed under physical constraints; a person may not be at two different places at the same time, and a meeting room may not be allocated to more than one meeting at a time. To allow as much flexibility as possible, dynamic replanning of meetings should be supported. On the one hand, participants should be allowed to modify their exclusion set, preference set and/or preferred location until the meeting is scheduled. On the other hand, exceptional constraints should be accommodated after a meeting is scheduled, such as the need to schedule an urgent, more important meeting. The original meeting date or location may then need to be changed; sometimes it may even be cancelled. In all cases some way of replanning should be set up.

The system should be flexible enough to accommodate different data formats (e.g. date or address formats) and evolving data (e.g. the set of concerned participants may vary during the scheduling process, and the address at which a participant can be reached may change).

There are also security concerns to be taken into account, such as the following:

- *Meeting initiation should be restricted to authorized personnel.*

- *Confidentiality rules should be enforced, for instance a non-privileged participant should not be aware of constraints stated by other participants, or of other meetings to which the latter are invited.*

Rather than a single product, WSS is thinking of a product family. The customization space should cover the following variations:

- *Professional meetings, private meetings.*
- *Single-site meetings, meetings where the target site needs to be determined as well, electronic meetings.*
- *Regular meetings (e.g. for a university course), occasional meetings.*
- *Single-level meetings or multi-level meetings where the importance of a person attending a specific meeting is higher or lower with respect to other meetings.*
- *Single-level participation or multi-level participation where the importance of a meeting getting a specific attendee is higher or lower with respect to other attendees.*
- *Single-level participants or multi-level participants where some participants are hierarchically more important than others (regardless of a specific meeting) or have less flexibility in changing their constraints.*
- *Variations on what participating in a meeting means, e.g. full attendance, partial attendance, participation through delegation.*
- *Variations on what constraints are about, e.g. no preference set, unordered preferences, ordered preferences, date availability dependent on meeting location.*
- *Parameterizability on explicit conflict-resolution rules that are tunable by the client, e.g. 'best meeting dates and locations should be determined by considering participants with higher importance first', 'in case of date conflict the scheduler will propose a person of lower importance to withdraw from the meeting', 'in case of date conflict the meeting scheduler will propose a participant to withdraw from another meeting of lower importance', 'a date within some exclusion set will be considered if the corresponding participant has high flexibility'.*
- *Mono-lingual, multi-lingual communication with participants.*
- *Variations on additional features such as support for elaborating the meeting agenda or the meeting minutes.*

Para evaluar los resúmenes generados automáticamente se tuvieron que crear los resúmenes de referencia, uno por cada caso de estudio, contra los cuáles se realizó dicha evaluación, Para la elaboración de dichos resúmenes, como primer paso se enumeraron de forma progresiva cada oración del texto, como segundo paso se seleccionaron aquellas oraciones que se consideraron importantes para cada caso de estudio (con base al dominio del problema) y, finalmente estas oraciones fueron concatenadas para formar así el resumen de referencia. Esta actividad fue realizada por estudiantes de la maestría en Ingeniería de *Software* del tercer semestre de la Universidad Tecnológica de la Mixteca, los cuales fueron validados por una profesora-investigadora con más de 20 años de experiencia en área de IR.

Para el primer caso de estudio (***Library Management***) se identificaron 26 oraciones de las cuales se seleccionaron 6 al ser consideradas relevantes. Para el segundo caso de estudio (***Train Control***) se identificaron 32 oraciones de donde se seleccionaron 9 oraciones relevantes. Y para el tercer caso de estudio (***Meeting Scheduling***) se identificaron 47 oraciones de las cuales 27 fueron seleccionadas para generar el nuevo resumen de referencia. Una vez ya con los resúmenes de referencia generados se procedió a aplicarles las métricas ROUGE descritas en la Sección 3.1.4, obteniendo los siguientes resultados:

Como puede verse en la Tabla 1, el algoritmo LSA obtuvo el mejor promedio en cuanto a la precisión, es decir, generó el resumen conteniendo un mayor porcentaje de las oraciones relevantes y, que coincide en su mayoría con el resumen de referencia. En la Tabla 2, en cuanto a la recuperación, que se refiere a la capacidad de los algoritmos y al modelo de identificar los elementos relevantes del texto, el mejor algoritmo fue el *TextRank* con un promedio de 0.72. Finalmente, en la Tabla 3, donde se evalúo la medida-F - que captura la información relevante del resumen generado -, tomando en cuenta el porcentaje de texto original que fue incluido, el mejor algoritmo fue nuevamente el LSA con un promedio de 0.69.

Tabla 1. Precisión de los algoritmos y modelo.

	Caso de estudio 1			Caso de estudio 2			Caso de estudio 3			Promedio			
Algoritmos y modelo	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Caso1**	**Caso2**	**Caso3**	**Total**
TopicRank	0.80	0.71	0.49	0.79	0.64	0.33	0.80	0.65	0.38	0.67	0.58	0.61	0.62
PositionRank	0.70	0.51	0.36	0.69	0.51	0.29	0.80	0.60	0.40	0.52	0.50	0.60	0.54
TextRank	0.70	0.51	0.36	0.69	0.51	0.34	0.83	0.68	0.30	0.52	0.51	0.60	0.55
TextRank Summy	0.68	0.58	0.59	0.61	0.50	0.52	0.75	0.64	0.65	0.62	0.54	0.68	0.61
LexRank	0.73	0.53	0.58	0.69	0.50	0.54	0.82	0.68	0.70	0.62	0.57	0.73	0.64
LSA	**0.78**	**0.69**	**0.68**	**0.83**	**0.69**	**0.72**	**0.84**	**0.75**	**0.76**	**0.72**	**0.75**	**0.78**	**0.75**
Luhn	0.67	0.54	0.59	0.72	0.54	0.59	0.78	0.69	0.70	0.60	0.62	0.72	0.65
Edmundson	0.70	0.51	0.36	0.69	0.51	0.34	0.83	0.68	0.30	0.52	0.51	0.60	0.57
BERT	0.60	0.59	0.65	0.48	0.29	0.33	0.7	0.66	0.68	0.61	0.37	0.68	0.55

Tabla 2. Recuperación de los algoritmos y modelo.

	Caso de estudio 1			Caso de estudio 2			Caso de estudio 3			Promedio			
Algoritmos y modelo	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Caso1**	**Caso2**	**Caso3**	**Total**
TopicRank	0.85	0.76	0.52	0.73	0.59	0.30	0.69	0.56	0.33	0.71	0.54	0.53	0.59
PositionRank	0.62	0.45	0.32	0.60	0.45	0.26	0.54	0.41	0.27	0.47	0.44	0.41	0.44
PyTextRank	0.62	0.45	0.32	0.60	0.45	0.30	0.67	0.55	0.24	0.47	0.45	0.49	0.47
TextRank Summy	**0.80**	**0.68**	**0.70**	**0.73**	**0.59**	**0.62**	**0.88**	**0.75**	**0.77**	**0.72**	**0.64**	**0.80**	**0.72**
LexRank	0.63	0.46	0.50	0.60	0.43	0.47	0.81	0.67	0.69	0.53	0.50	0.72	0.58
LSA	0.75	0.66	0.66	0.68	0.56	0.58	0.71	0.62	0.63	0.69	0.60	0.65	0.65
Luhn	0.73	0.59	0.64	0.68	0.52	0.56	0.84	0.74	0.76	0.65	0.59	0.78	0.67
Edmundson	0.70	0.51	0.36	0.69	0.51	0.34	0.83	0.68	0.30	0.52	0.51	0.60	0.57
BERT	0.62	0.46	0.52	0.48	0.29	0.33	0.81	0.77	0.79	0.53	0.37	0.79	0.56

Tomando en cuenta el desempeño más alto en las tres medidas: precisión, recuperación y medida-F, se seleccionó el algoritmo LSA para generar los resúmenes automáticos. Sin embargo, este algoritmo tuvo que ser modificado, sobre todo en la forma de obtención del TF-IDF, para que al generar el resumen tomara en cuenta lo que establece el estándar ISO/IEC/IEEE 29148 (2018) -específicamente en el punto 5.2.4 correspondiente al *"requirements constructor"*-, donde se hace mención que las oraciones que denotan un FR deben contener la palabra en inglés *"shall"*, ya que así se especifica que éstos son disposiciones vinculantes obligatorias.

Tabla 3. Medida-F de los algoritmos y modelo.

	Caso de estudio 1			Caso de estudio 2			Caso de estudio 3			Promedio			
Algoritmos y modelo	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Caso1**	**Caso2**	**Caso3**	**Total**
TopicRank	0.83	0.73	0.51	0.73	0.59	0.30	0.74	0.61	0.35	0.69	0.54	0.57	0.60
PositionRank	0.66	0.48	0.34	0.64	0.48	0.27	0.64	0.48	0.33	0.49	0.47	0.48	0.48
PyTextRank	0.66	0.48	0.34	0.64	0.48	0.31	0.74	0.61	0.27	0.49	0.48	0.54	0.50
TextRank Summy	0.73	0.62	0.64	0.67	0.54	0.57	0.81	0.69	0.71	0.67	0.59	0.73	0.66
LexRank	0.68	0.50	0.54	0.64	0.46	0.50	0.82	0.67	0.69	0.57	0.53	0.73	0.61
LSA	**0.76**	**0.67**	**0.67**	**0.75**	**0.61**	**0.64**	**0.77**	**0.68**	**0.69**	**0.70**	**0.59**	**0.71**	**0.69**
Luhn	0.70	0.57	0.61	0.70	0.53	0.57	0.81	0.72	0.73	0.63	0.60	0.75	0.66
Edmundson	0.70	0.51	0.36	0.69	0.51	0.34	0.83	0.68	0.30	0.52	0.51	0.60	0.57
BERT	0.53	0.28	0.39	0.48	0.29	0.33	0.81	0.77	0.79	0.40	0.37	0.79	0.52

3.2.1. Algoritmo de LSA Modificado

Con el objetivo de encontrar en un texto de deseos y necesidades de los *stakeholders* el mayor número de oraciones que representen un FR preliminar, se decidió expandir el término *"shall"* utilizando para ello el diccionario de inglés *WordReference*. Encontrando como sinónimos las palabras "*need*", "*has*", "*want*", "*have*", "*should*", "*require*", "*demand*" y, "*must*", todas estas palabras fueron consideradas como palabras clave.

El ajuste al TF se realizó con la finalidad de identificar únicamente aquellas oraciones, dentro del texto de requisitos preliminares, que incluyeran las palabras clave mencionadas en el párrafo anterior, sin importar su frecuencia de aparición, dado que en un texto de requisitos preliminares se pueden encontrar palabras clave con menor frecuencia que otras, pero que si representan un requisito. Dado de que en el IDF cuanto más aparece una palabra en las oraciones que conforman un texto, esta se vuelve menos importante, se tuvo que realizar un ajuste para que cuando se encontraran las palabras clave dentro de una oración, esta se seleccionara como importante, sin importar su frecuencia de aparición en el texto.

Ahora bien, dado que el espacio de búsqueda se transformó en uno de menor dimensión al aplicar el SVD, el tamaño del texto también se redujo ya que se seleccionó solo el texto importante y representativo, por lo que se realizó nuevamente la búsqueda de las palabras claves para seleccionar solo aquellas oraciones que las incluyeran, y así finalmente generar el resumen concatenándolas. A continuación, se muestra el resumen generado automáticamente para cada caso de estudio.

Caso de estudio 1

Case Study 1: Library Management

The University of Wonderland (UWON) ***wants*** *to convert its library system into a new system to ensure more effective access to state-of-the-art books, periodicals and proceedings while reducing operational costs. The new UWON library system* ***should*** *address such problems through a software-based solution integrating all department libraries. The new system* ***should*** *interoperate with library systems from partner universities. It* ***should*** *provide interactive online facilities for book acquisition, user registration, loan management, bibliographical search and book reservation. Access to such facilities* ***should*** *be restricted to specific user categories, according to authorization rules specific to each facility. The new system* ***should*** *take advantage of opportunities provided by new technologies.*

*In particular, it **should** support subscriptions to e-journals, provide access to foreign digital libraries (under specific conditions), support e-mail communication between staff and users, enable bibliographical search from anywhere at any time, and provide a Web-based interface for book e-seller comparison, selection, and order submission.*

Caso de estudio 2

Case Study 2: Train Control

*Buses are slow and often late due to traffic jams; passengers **need** to stand in long queues, sometimes for an unacceptably long time, which may cause them to miss flight connections, and so on. In this standard option, train drivers **have to** follow recommendations issued by the software and respond to regular stimuli issued by the software to check driver responsiveness. Various concerns about the new system have already emerged at this preliminary stage: In order to ensure rapid transportation of passengers, trains **should** run fast, without unnecessary delays, and at high frequency, during rush hours at least. In order to guarantee safe transportation of passengers, the probability of accidents **must** fall below the threshold imposed by safety regulations. In particular, the distance between two trains following each other **must** always be sufficient to prevent the back train from hitting the front train in case the latter stops suddenly. Train doors **must** always be kept closed while a train is moving. In order to ensure comfortable transportation, trains **should** accelerate/decelerate smoothly. Passengers at a station **should** be informed in time about trains arriving. Passengers inside a train **should** be informed in time that the train is departing, which companies are being served by the next stop and so on. It brings together government representatives, airport authorities, Wonderland Railways and the engineering company selected to implement the project.*

Caso de estudio 3

Case Study 3: Meeting Scheduling*

*It is hard to find a date and a place that suit everyone's constraints; meeting organizers **need** to pester people to get their availability; other people are unnecessarily inconvenienced by messages that do not concern them; when the meeting is scheduled some constraints have changed in the meantime; new scheduling cycles **need** to be repeated when no date/location is found in a reasonably short period; and so forth. A meeting initiator informs potential participants about the **need** for a meeting and specifies a date range within which the meeting **should** take place, asking people to return their availability constraints within that time interval. In some cases, the initiator may also ask participants who will play an active role in the meeting for specific requirements regarding the meeting room (e.g. The scheduled meeting date **should** belong to the stated date range and to none of the exclusion sets; it **should** ideally belong to as many preference sets as possible. The meeting venue **should** ideally fit the preferences of important participants. A room conflict occurs when no room can be found, at any date outside all exclusion sets, which meets the room requirements. A new scheduling cycle may thus be **required** in case of conflict. The envisioned meeting scheduler software **should** reflect as closely as possible the way meetings are typically managed. It **should** be useable by administrative staff and provide major improvements in several respects: Average participant attendance **should** increase thanks to the selection of meeting dates and locations that are the most convenient to potential participants. Meetings **should** be scheduled as quickly as possible once they are initiated. Meeting dates and locations **should** be notified as quickly as possible to all potential participants once they are scheduled. In all cases, there **should** be sufficient time between notification and the meeting date. The organizational overhead **should** be kept as low as possible on the initiator's*

*side. In particular, the meeting scheduler **should** support all **required** interactions with participants, for example to communicate requests, get replies (even from participants not reacting promptly), assist in negotiation and conflict-resolution processes, and inform participants on request about the state of the scheduling process. The amount of interaction with potential participants for meeting scheduling, in number and length of messages, **should** be kept as small as possible. The new meeting scheduler must be able to handle multiple meeting requests in parallel. Concurrency must thus be managed under physical constraints; a person may not be at two different places at the same time, and a meeting room may not be allocated to more than one meeting at a time. To allow as much flexibility as possible, dynamic replanning of meetings **should** be supported. On the one hand, participants **should** be allowed to modify their exclusion set, preference set and/or preferred location until the meeting is scheduled. On the other hand, exceptional constraints **should** be accommodated after a meeting is scheduled, such as the **need** to schedule an urgent, more important meeting. The original meeting date or location may then **need** to be changed; sometimes it may even be cancelled. In all cases some way of replanning **should** be set up. The system **should** be flexible enough to accommodate different data formats (e.g. the set of concerned participants may vary during the scheduling process, and the address at which a participant can be reached may change). There are also security concerns to be taken into account, such as the following: Meeting initiation **should** be restricted to authorized personnel. Confidentiality rules **should** be enforced, for instance a non-privileged participant **should** not be aware of constraints stated by other participants, or of other meetings to which the latter are invited. Rather than a single product, WSS is thinking of a product family. The customization space **should** cover the following variations: Professional meetings, private meetings. Single-site meetings, meetings where the target site **needs** to be determined as well, electronic meetings. 'best meeting dates and locations **should** be determined by considering participants with higher importance first', 'in case of date conflict the scheduler will propose a person of lower importance to withdraw from the meeting', 'in case of date conflict the meeting scheduler will propose a participant to withdraw from another meeting of lower importance', 'a date within some exclusion set will be considered if the corresponding participant has high flexibility'. As a result, meetings tend to be organized poorly and late; important people sometimes do not show up; and there is a significant, unnecessary overhead in the scheduling process. Meetings are typically scheduled as follows. for a university course), occasional meetings. Single-level meetings or multi-level meetings where the importance of a person attending a specific meeting is higher or lower with respect to other meetings. Single-level participation or multi-level participation where the importance of a meeting getting a specific attendee is higher or lower with respect to other attendees.*

Nuevamente a estos resúmenes generados automáticamente por el LSA modificado se les aplicó las métricas de ROUGE (ver Sección 3.1.4), para evaluarlos, obteniendo los siguientes mostrados en las Tablas 4, 5 y 6.

Donde se puede apreciar en las que el LSA modificado obtuvo mejores resultados que en las Tablas 1, 2, y 3, ya que se demostró una mayor eficiencia tanto en la recuperación de las palabras claves que representan un requisito preliminar, como en la precisión. Por lo tanto, el LSA modificado fue seleccionado como la mejor opción para el desarrollo de la herramienta de *software* propuesta en este trabajo de tesis.

Tabla 4. Precisión del algoritmo LSA.

	Caso de estudio 1			Caso de estudio 2			Caso de estudio 3			Promedio			
Algoritmo	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Caso1**	**Caso2**	**Caso3**	**Total**
Lsamod	1.00	1.00	1.00	1.00	1.00	1.00	0.99	0.98	0.98	1.00	1.00	0.99	0.99

Tabla 5. Recuperación del algoritmo LSA.

	Caso de estudio 1			Caso de estudio 2			Caso de estudio 3			Promedio			
Algoritmo	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Caso1**	**Caso2**	**Caso3**	**Total**
Lsamod	1.00	1.00	1.00	1.00	1.00	1.00	0.85	0.84	0.84	1.00	1.00	0.85	0.95

Tabla 6. Medida-F del algoritmo LSA.

	Caso de estudio 1			Caso de estudio 2			Caso de estudio 3			Promedio			
Algoritmo	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Rouge-1**	**Rouge-2**	**Rouge-L**	**Caso1**	**Caso2**	**Caso3**	**Total**
Lsamod	1.00	1.00	1.00	1.00	1.00	1.00	0.92	0.91	0.91	1.00	1.00	0.91	0.97

3.3. Identificación de requisitos funcionales

La identificación de FRs a partir de un documento de requisitos preliminares puede realizarse de forma manual; sin embargo, requiere de un tiempo, esfuerzo y presupuesto grandes. Es por lo que, de acuerdo con Shreda y Hanani (2021), y Kaur y Kaur (2023), se están utilizando tanto el NLP como los algoritmos de clasificación binaria para la realización automática de esta tarea.

Sin embargo, antes de proceder a realizar la clasificación de los requisitos preliminares, fue necesario revisar su estructura para que el formato de estos éstos cumpliera con la característica de calidad (singularidad) indicado en el estándar ISO/IEC/IEEE 29148 (2018). Para poder llevar a cabo esta revisión se utilizaron Modelos de Lenguaje Largos (LLM, por sus siglas en inglés) ya que estos permiten realizar tareas tales como la generación y simplificación de textos, razonamiento de una cadena de pensamiento (*"Chain of Thought"* (CoT)), así como el seguimiento de instrucciones y aprendizajes del contexto (Qin et al., 2024; Mahapatra y Garain, 2024).

3.3.1. Uso de los LLMs para la segmentación de oraciones de requisitos preliminares de acuerdo con el estándar ISO/IEC/IEEE 29148 (2018)

A partir del resumen generado de manera automática (ver Sección 3.2.1), se procedió a la extracción de los FRs preliminares. Para esto fue necesario como primera tarea, segmentar el texto en oraciones simples para así estas cumpliesen con la característica de calidad de la singularidad mencionada en el apartado 5.2.5 del estándar ISO/IEC/IEEE 29148 (2018) que a la letra menciona que un requisito no debe incluir conjunciones y solo debe describir una sola necesidad. Por lo que se procedió a "cortar" las oraciones cuando fuesen detectados conectores como la "y" o la "o", así como las comas (","), puntos y coma (";"), dos puntos y seguido (":") y los puntos y seguido.

Para realizar esta segmentación sin perder la coherencia de las oraciones fue necesario utilizar un LLM, ya que este tipo de lenguajes permite procesar el NL de forma avanzada al ser capaces de identificar la estructura gramatical de las oraciones y segmentarlas de manera automática según las reglas definidas (en este caso las mencionadas en el párrafo anterior).

En este trabajo de tesis se seleccionó el *LM Studio* (versión 0.3.9) - al ser una aplicación de escritorio que permite a los usuarios ejecutar de forma local diferentes LLM´s y desarrollar aplicaciones con IA personalizadas. En *LM Studio* se instalaron los modelos de lenguaje *Llama* (versión 3.2) de META IA, *DeepSeek* (versión 1.0) de la compañía del mismo nombre y *Gemma* (versión 3) con el objetivo de entrenarlos para llevar a cabo la segmentación del texto. Sin embargo, para realizar esta tarea, primero fue necesario definir un *prompt*[22] que permitiera generar una lista de oraciones simples (definidas en el primer párrafo de esta subsección) para que así cada oración representara un solo requisito preliminar (característica de singularidad) pero manteniendo su coherencia y significancia dentro del texto original.

Posteriormente se configuró el parámetro de "temperatura" de cada uno de los LLM´s (*META*, *DeepSeek*, y *Gemma*) que regula la aleatoriedad de un resultado en la generación de texto (Peeperkorn et al., 2024). Nakaishi et al. (2024) y Peeperkorn et al., (2024) mencionan que cuando un LLM utiliza bajas temperaturas (i.e., en un rango de 0.0 a 0.3) este genera estructuras claras, en temperaturas medias (i.e., entre 0.5 – 0.8) genera conversaciones naturales, y en temperaturas altas (i.e., >= 1), los resultados suelen ser impredecibles. En este caso, los valores utilizados fueron de 0.3, 0.5, y 0.8, donde 0.3 representa la selección de las palabras con mayor probabilidad de aparecer en el texto, 0.5 las respuestas pueden variar y 0.8 la respuesta más creativa (i.e., el LLM agrega detalles innecesarios a la respuesta o hasta puede cambiar el enfoque de la respuesta).

El LLM de *Gemma* mostró los mejores resultados con una temperatura de 0.3; sin embargo, en ciertas situaciones las oraciones perdían su contexto original o se dividían de manera inadecuada (i.e., no contextualizaba correctamente una oración respecto al texto proporcionado). Así mismo, cuando se le enviaba dos veces el mismo *prompt*, los resultados variaban, afectando así la escritura adecuada de los requisitos preliminares. En la Tabla 7 se muestra un ejemplo de los resultados obtenidos usando como entrada la siguiente oración ya segmentada: *"Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency and better information to passenger."*

Para solucionar estos comportamientos, se ajustó el *prompt* añadiéndole las instrucciones de que tenía que contextualizar la oración de acuerdo con el dominio del texto y en entradas repetidas, generar siempre el mismo resultado. Finalmente, se realizó el etiquetado manual de los requisitos preliminares (i.e., oraciones generadas por *Gemma*) en funcionales, no funcionales, de negocio y por revisar (aquellos mal redactados de origen y que por ende no satisfacían la estructura del estándar ISO/IEC/IEEE 29148 (2018). Esta actividad fue realizada por la autora y supervisada por una profesora-investigadora de la Universidad Tecnológica de la Mixteca con más de 20 años de experiencia en la IR. Este conjunto de requisitos preliminares ya etiquetados sirvió como parte del conjunto de entrenamiento para los algoritmos de clasificación binaria que fueron utilizados para extraer los FRs de un texto de requisitos preliminares.

3.3.2. Algoritmos de clasificación binaria para la identificación de requisitos funcionales

Para la identificación de FRs preliminares se utilizaron algoritmos de clasificación binaria mismos que se entrenaron con muestras de datos pre-etiquetados llamados datos de entrenamiento. Con estos datos de entrenamiento el algoritmo de clasificación binaria aprende a predecir a qué tipo de clase pertenecen los datos de clases muestra llamados datos de prueba (Sarkar, 2016; Mirjalili y Raschka, 2020).

[22] Conjunto de instrucciones que van a seguir los LLM´s para encontrar el resultado deseado (Schulhoff et al., 2024).

Tabla 7. Resultado preliminar de una oración.

LLM	Temperatura 0.3	Temperatura 0.5	Temperatura 0.8
Gemma	***Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality.*** ***Software-controlled movement of trains will allow for better frequency.*** ***Software-controlled movement of trains will allow for better information to passengers.***	*Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency, and better information to passengers.*	*Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality.* *Higher frequency and better information to passengers.*
DeepSeek	*Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency and better information to passengers.* *The use of software to control train movements could enhance reliability by tracking each train's progress more effectively.*	*Software-controlled movement will improve punctuality.* *This system also enhances the frequency of train services.* *Passengers benefit from better information availability.*	*Software-controlled movements may ensure trains adhere more closely to schedules by efficiently managing complex routing and traffic management, thus improving punctuality.* *Such systems could potentially increase the frequency of train services.* *Passengers would receive real-time updates, including expected arrival times and journey durations, potentially via predictive analytics and dynamic rerouting, enhancing their ability to plan trips effectively.*
Llama	*Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency and better information to passengers.*	*Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality, higher frequency, and better information to passengers.*	*Preliminary investigations suggest that software-controlled movement of trains will allow for better punctuality.* *Higher frequency and better information to passengers.*

Binkhonain y Zhao (2019), Kaur y Kaur (2024) en sus RSL, recopilaron estudios sobre el uso de la IA en la clasificación de FRs y NFRs, y encontraron que pueden utilizarse algoritmos de ML, algoritmos de DL y algoritmos de aprendizaje por transferencia. En estas dos revisiones, los autores evaluaron clasificadores binarios y multiclases para requisitos de *software* como las SVM, el Bayes Ingenuo (NB, por sus siglas en inglés), la LR, aproximaciones de DL para clasificación como las Redes Neuronales Convolucionales (CNN, por sus siglas en inglés), así como el modelo de lenguaje preentrenado BERT.

A continuación, se describen los algoritmos de clasificación binaria previamente mencionados (Shreda y Hanani, 2021; Talele y Phalnikar, 2021, Enero; Kaur y Kaur, 2024):

- SVM: se utilizan para la clasificación, regresión y detección de valores atípicos dentro de un conjunto de datos. Este algoritmo toma un conjunto de datos de entrenamiento y construye un hiperplano y cuanto más grandes sean los márgenes de este, mejor será la

separación de características, lo que conllevará a menores errores de generalización del clasificador.

- NB: es un clasificador probabilístico que se basa en el teorema de Bayes. Evalúa de forma independiente la probabilidad de que una oración pertenezca a una clase definida por cada característica de entrada (i.e., una oración), así como de las etiquetas de clase y genera la salida combinando las probabilidades de estos dos parámetros.
- LR: es un clasificador de ML supervisado basado en la probabilidad. El clasificador toma el valor de entrada (i.e., una oración) y lo multiplica por el valor de una ponderación. La LR aprende la característica única de la oración (e.g., la presencia o frecuencia de una palabra) más adecuada para diferenciar las distintas clases (e.g., una oración que describe un FR pertenece a una clase, mientras que otra oración que expresa un NFR pertenece a otra clase).
- CNN: es una red neuronal profunda que consta de una serie de capas convolucionales que permiten extraer las características de las aplicaciones de NLP. El filtro de convolución se desliza sobre un espacio vectorial para calcular el producto escalar entre el valor del filtro y el vector de palabras. La capa de máxima agrupación se utiliza para obtener la característica (i.e., la secuencia de valores numéricos que representan patrones del texto) más importante de la CNN, es decir, la que obtenga el valor máximo. Al final, estas características se envían a la capa completamente conectada para su clasificación.
- BERT: forma parte de los modelos de aprendizaje por transferencia, ya que utiliza una arquitectura llamada transformador, que permite analizar las palabras dependientes de acuerdo con su contexto, en lugar de analizar las palabras de manera independiente. El modelo BERT tiene dos variantes: el modelo base y el modelo largo, ambos varían en su capacidad de modelado es decir en la cantidad de parámetros recibidos.

3.3.3. Implementación de algoritmos de clasificación binaria para la identificación de requisitos funcionales

Para la implementación de los algoritmos de clasificación binaria o también llamados clasificadores binarios (SVM, BN, LR, CNN) y del modelo de aprendizaje por transferencia (BERT) se utilizó el lenguaje de programación *Python* (versión 3.12.7) en conjunto con sus librerías gratuitas como *Pandas* (versión 2.2.3), *NumPy* (versión 1.26.4) y *Hugging Face Transformers* (versión 4.52.4), además de las siguientes librerías de código abierto:

- *Scikit-learn* (versión 1.6.0) que es una biblioteca para ML en *Python*.
- *TensorFlow* (versión 2.19.0) desarrollada por Google, facilita la creación de modelos de ML, así como la construcción y entrenamiento de ANN, la detección y desciframiento de patrones y correlaciones en un texto.
- *PyTorch* (versión 2.6.0) biblioteca de ML utilizada para aplicaciones de visión artificial y NLP.

Dado que la clasificación binaria es una tarea de aprendizaje supervisado, esta se llevó a cabo utilizando las siguientes etapas:

3.3.3.1. Recolección de los datos de entrenamiento

Se realizó la búsqueda de repositorios de FRs para poder entrenar los clasificadores binarios en la identificación de FRs preliminares, para esto se utilizaron dos conjuntos:

- PROMISE [23].
- Conjunto datos de los casos de estudio[24].

La combinación de estos dos conjuntos de datos permitió entrenar a los clasificadores y evaluar su capacidad de generalización[25]. Los datos de entrenamiento se utilizan para enseñarle al clasificador binario a reconocer patrones y hacer predicciones, mientras que los datos de prueba sirven para evaluar su desempeño

3.3.3.2. Preprocesamiento de los datos de entrenamiento

Como mencionan Shreda y Hanani (2021), antes de entrenar a los clasificadores binarios es necesario someter los datos de entrenamiento a un preprocesamiento (i.e., *tokenización*, limpieza de datos y normalización) para asegurar su calidad.

Con la finalidad de probar qué clasificador binario permite identificar con mejor precisión, recuperación, medida-F, y exactitud los FRs, se realizaron 6 experimentos. En los experimentos del 1 al 4 se aplicó el preprocesamiento anteriormente mencionado; sin embargo, para los experimentos 5 y 6 llevados a cabo con el modelo BERT se observó que con la conversión de las mayúsculas en minúsculas junto con la expansión de oraciones (preprocesamiento mínimo) éste tuvo un mejor desempeño.

3.3.3.3. División del conjunto de datos de entrenamiento

Para los 7 experimentos mencionados anteriormente, se aplicaron diferentes proporciones para la división de los datos de entrenamiento.

- En el experimento 1 se utilizó el 100% del conjunto de datos de PROMISE como de entrenamiento mientras que el 100% de los datos de los casos de estudio se usó como el conjunto de prueba.
- En los experimentos 2, 4, 5, 6 y 7 se utilizaron:
    - El 80% del conjunto de datos de PROMISE y el 50% del conjunto de los datos de los casos de estudio como datos de entrenamiento.
    - Los restantes 20% del conjunto de datos de PROMISE y el 50% los datos de los de los casos de estudios como datos de prueba.
- En el experimento 3 se utilizó la siguiente proporción de datos:
    - Se combinó tanto el 100% del conjunto de datos de PROMISE como del conjunto de los datos de los casos de estudios, una vez formado el nuevo conjunto de datos, éste se particionó en 80% para entrenamiento y 20% para prueba.

Estas particiones de datos (*folds*) permitieron observar el comportamiento de los clasificadores binarios tanto en dominios conocidos (datos de entrenamiento), así como para dominios completamente desconocidos (datos de prueba).

---

[23] PROMISE es un repositorio de requisitos de *software* que consta de un total de 625 requisitos, de los cuales 255 son FRs y 370 NFRs (Luo et al., 2022).

[24] Conjunto resultante en la Sección 3.3.1. que contiene un total de 65 FRs y 53 NFRs.

[25] La generalización en clasificadores se refiere a su capacidad de clasificar correctamente en nuevos datos en función de su entrenamiento (Borra et al., 2019).

3.3.3.4. Entrenamiento de los clasificadores binarios

Como primer paso se aplicó el preprocesamiento a los dos conjuntos de datos (PROMISE y casos de estudio) en su totalidad, posteriormente se realizó su división en datos de "entrenamiento" y datos de "prueba" para "entrenar" a los clasificadores SVM, LR, NB, CNN y al modelo BERT. Para el entrenamiento de los clasificadores binarios fue necesario registrar, para cada uno de ellos, sus medidas de rendimiento. Esto permitió observar su comportamiento para detectar posibles sobreajustes (*overfitting*[26]), tal como ocurrió con la CNN donde se obtuvieron buenos resultados en el entrenamiento, pero no así en las pruebas.

Para el experimento 1 se entrenaron los clasificadores binarios SVM, LR y NB, primero realizando la vectorización mediante la TF-IDF y después aplicando una validación cruzada (*cross-validation*[27]) para lo cual el conjunto PROMISE fue dividido en cinco particiones. Para cada partición se ejecutó el experimento utilizando cuatro particiones como datos de entrenamiento (i.e., el 80% de los datos) y la restante como datos de validación (i.e., el 20% de los datos). Una vez concluida la validación cruzada, por cada clasificador binario ya entrenado, se realizó la prueba con los datos de los casos de estudio.

En el experimento 2 se entrenaron, al igual que en el experimento 1, los clasificadores binarios SVM, LR y NB, utilizando la TF-IDF para la vectorización del texto. Para el entrenamiento, cada clasificador binario usó un conjunto combinado formado por el 80 % de los datos de PROMISE y el 50 % de los datos de los casos de estudio. El objetivo fue identificar patrones que les permitieran a los clasificadores binarios diferenciar entre FRs y NFRs. Una vez entrenados, los clasificadores binarios fueron evaluados en los conjuntos de prueba formados por los datos restantes (20 % PROMISE y 50 % casos de estudio).

En el experimento 3 se entrenó la CNN en donde se utilizó *Word2Vec*[28] para la vectorización del texto. Para el entrenamiento se utilizó el 80 % de los datos de PROMISE y el 80 % de los datos de los casos de estudio, y se aplicó una validación cruzada con 5 iteraciones. En cada iteración, la CNN fue entrenada con 50 épocas[29]. Posteriormente, fue evaluada con el conjunto de prueba (el 20% restante de ambos conjuntos).

En el experimento 4, al igual que en el experimento 3, se entrenó la CNN utilizando *Word2Vec* para la vectorización del texto. Este entrenamiento se llevó a cabo utilizando un conjunto conformado por el 80 % de los datos de PROMISE y el 50 % de los datos de los casos de estudio y además se entrenó con 50 épocas. Posteriormente, fue evaluada con el conjunto de prueba formado por el 20% restante de los datos de PROMISE y 50% restantes de los casos de estudio.

---

[26] El sobreajuste u *overfitting* se produce cuando el clasificador no puede generalizar y este se ajusta demasiado al conjunto de datos de entrenamiento, lo que conduce a un alto rendimiento en el entrenamiento, pero da como resultado deficiente en las pruebas (Sliusarenko y Pohurska, 2024).

[27] Permite evaluar el rendimiento de un modelo predictivo mediante la partición de un conjunto de datos principal en un conjunto de datos de prueba y entrenamiento; esto ayuda a comprender cómo se desempeñará un modelo sobre diferentes subconjuntos de datos. El conjunto de datos original se divide en *k* subconjuntos denominados como *folds*, el modelo se entrena *k v*eces, utilizando un subconjunto diferente en cada entrenamiento y el resto como el conjunto de prueba (Arthi et al., 2024).

[28] *Word2Ve*c es un modelo entrenado con un enfoque de NLP que incluye representaciones vectoriales de palabras (Adewumi et al., 2020)

[29] La época (*epoch* en inglés) es un hiperparámetro de una CNN que desempeña un papel fundamental en el proceso de entrenamiento de un modelo. En cada época el modelo aprende un poco más, ajustándose para mejorar sus predicciones (Afaq y Rao, 2020).

En el experimento 5 dado que se entrenó el modelo de BERT y que este tiene su propia función de vectorización no fue necesario utilizar *Word2Vec*. Para su entrenamiento se utilizó el conjunto conformado por el 80 % de los datos de PROMISE y el 50 % de los datos de los casos de estudio y la validación con el conjunto de prueba se utilizó el 20 % de los datos de PROMISE y el 50 % de los datos de los casos de estudio.

En el experimento 6 se entrenó el modelo de BERT; sin embargo, se cambió el preprocesamiento de los conjuntos de datos. Como se mencionó anteriormente, únicamente se aplicó la conversión de mayúsculas en minúsculas junto con la expansión de oraciones. Para el entrenamiento se utilizó el conjunto conformado por el 80 % de los datos de PROMISE y el 50 % de los datos de los casos de estudio y la validación con el conjunto de prueba conformado por el 20 % de los datos de PROMISE y el 50 % de los datos de los casos de estudio.

Finalmente, en el experimento 7 se entrenó también el modelo BERT con el mismo porcentaje tanto de los datos de PROMISE como de los datos de los casos de estudio que en el experimento 5 (i.e., 80% y 50%, respectivamente). Pero la diferencia radicó en que en este experimento se realizó la búsqueda de los valores óptimos para los hiperparámetros[30]: *learning_rate*[31] y *weight_decay*[32]. Para lo cual se utilizaron los valores de 1e-5, 2e-5 y 3e-5 para el primer hiperparámetro y de 0, 1e-4, 1e-3, 1e-2, 5e-2 y 1e-1 para el segundo. Así mismo, se utilizaron 4 épocas para el entrenamiento, una vez encontrados los mejores valores para los hiperparámetros (*learning_rate* de 3e-5 y un *weight_decay* de 1e-2) dentro del conjunto de valores definidos en el entrenamiento, se realizó la validación con el conjunto de prueba (20 % PROMISE y 50 % casos de estudio).

3.3.3.5. Evaluación de los algoritmos para la identificación de requisitos funcionales

En esta sección se describen los 7 experimentos llevados a cabo con los clasificadores de SVM, LR, NB, CNN y BERT para evaluar su rendimiento y su capacidad de generalización con diferentes particiones de datos. Las medidas consideradas para estas evaluaciones fueron las siguientes: medida-F, recuperación, precisión y exactitud. Cada experimento se describe a través de su objetivo, así como la partición de datos y el clasificador binario utilizado.

Experimento 1

- Objetivo: Evaluar la capacidad de generalización de los clasificadores binarios SVM, LR, y NB.
- Partición de datos: Se utilizó el 100% del conjunto de datos de PROMISE para el entrenamiento y el 100% del conjunto de datos de los casos de estudio para las pruebas.
- Clasificador utilizado: SVM, LR y NB en los cuales se aplicó la validación cruzada con un *k-fold* [33] de 5.

[30] Los hiperparámetros en el ML son variables que se establecen antes de que comience el proceso de entrenamiento y regulan comportamiento del algoritmo (Agrawal, 2024, marzo).

[31] El *learning_rate* es un hiperparámetro que controla el tamaño del paso en cada iteración, desempeña un papel importante en la convergencia del modelo (Ding et al., 2023, octubre).

[32] El *weight_decay* en BERT se refiere a una técnica de regularización que penaliza los pesos grandes durante el entrenamiento lo que ayuda a prevenir el sobreajuste (Kobayashi et al., 2024).

[33] La validación cruzada *K-fold* es una técnica de remuestreo utilizada para evaluar modelos de ML, especialmente cuando se dispone de un conjunto de datos limitado. Divide el conjunto de datos en *k* subconjuntos (pliegues), y entrena y prueba el modelo en cada pliegue, utilizando *k-1* pliegues para entrenamiento y el pliegue restante para prueba, repitiendo el proceso *k* veces.

En las Tablas 8, 9, 10 y 11 se presentan los resultados tanto de la validación cruzada como de la prueba. En la validación cruzada se puede observar que los tres clasificadores (SVM, LR y NB) se desempeñan bien, alcanzando puntajes superiores a 0.83.

Tabla 8. Medida-F de los clasificadores binarios.

Clasificadores binarios	Promedio Validación cruzada	Prueba
SVM	**0.90**	**0.58**
LR	0.8452	0.59
NB	0.8843	0.50

Tabla 9. Recuperación de los clasificadores binarios tradicionales.

Clasificadores binarios	Promedio Validación cruzada	Prueba
SVM	**0.90**	**0.57**
LR	0.8340	0.45
NB	0.8812	0.51

Tabla 10. Precisión de los clasificadores binarios tradicionales.

Clasificadores binarios	Promedio Validación cruzada	Prueba
SVM	**0.91**	**0.59**
LR	0.8779	0.51
NB	0.8906	0.50

Tabla 11. Exactitud de los clasificadores binarios tradicionales.

Clasificadores binarios	Promedio Validación cruzada	Prueba
SVM	**0.91**	**0.63**
LR	0.8576	0.61
NB	0.8896	0.55

El clasificador SVM presenta un desempeño ligeramente superior en todas las métricas, seguido por el NB, particularmente en la precisión. No obstante, durante la prueba ambos clasificadores muestran una reducción drástica en su rendimiento, lo que evidencia una baja capacidad para identificar FRs preliminares cuando enfrentan casos totalmente desconocidos

Experimento 2

- Objetivo: Observar el rendimiento de los clasificadores SVM, LR y NB al incluir los datos de los casos de estudio en su entrenamiento.
- Partición de datos: Se utilizó el 80% del conjunto de los datos de PROMISE. El 50% para el entrenamiento, y el resto de los datos para la prueba (20% del conjunto de PROMISE y el 50% de los casos de estudio).
- Clasificador utilizado: SVM, LR y NB.

Durante el entrenamiento, los clasificadores LR y el NB alcanzaron puntajes >0.90 para la medida-F, recuperación, precisión, y exactitud, mientras que el SVM obtuvo una recuperación del 0.97 (ver Tabla 12). Para el conjunto de prueba (PROMISE), se presentó una ligera una disminución respecto a los valores obtenidos en el entrenamiento; mientras que, para los datos de los casos de

estudio, los valores bajaron en general. Sin embargo, como se aprecia en las Tablas 13 a la 15, el SVM obtuvo los mejores resultados: medida-F 0.75, recuperación 0.77, precisión 0.74 y exactitud 0.78.

Tabla 12. Recuperación de los clasificadores binarios tradicionales.

Clasificadores binarios	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
SVM	**0.97**	**0.86**	**0.77**
RL	0.87	0.75	0.62
NB	0.92	0.84	0.69

Tabla 13. Medida-F de los clasificadores binarios tradicionales.

Clasificadores binarios	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
SVM	**0.97**	**0.88**	**0.75**
RL	0.92	0.83	0.68
NB	0.94	0.86	0.68

Tabla 14. Precisión de los clasificadores binarios tradicionales.

Clasificadores binarios	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
SVM	0.97	0.90	0.74
RL	**0.97**	**0.93**	**0.76**
NB	0.95	0.88	0.67

Tabla 15. Exactitud de los clasificadores binarios.

Clasificadores binarios	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
SVM	**0.97**	**0.90**	**0.78**
RL	0.94	0.87	0.75
NB	0.95	0.89	0.71

Experimento 3

- Objetivo: Evaluar el rendimiento de la CNN.
- Partición de datos: Se combinó el 80% de PROMISE y el 80% de los casos de estudio para llevar a cabo el entrenamiento a través de la validación cruzada. El 20% restante de ambos conjuntos se utilizó como conjunto de prueba.
- Clasificador utilizado: CNN.

Para este experimento se aplicó la validación cruzada con 5 particiones y la red fue entrenada con 50 épocas. En la Figura 3.1, se puede observar el comportamiento de la medida-F durante el entrenamiento, donde los valores alcanzan puntajes cercanos al 0.90 después de 25 épocas; sin embargo, en las curvas de validación el rendimiento es inferior ya que los puntajes rondan entre el 0.6 y 0.75. Para la precisión, durante el entrenamiento los resultados están por encima de los 0.90, sin embargo, para la validación los resultados están en un rango de 0.50 a 0.75 (ver Figura 3.2).

Así mismo para la recuperación, durante el entrenamiento la curva aumenta rápidamente y se estabiliza por encima de los 0.90 en todas las particiones, mientras que en la validación es más inestable con fluctuaciones que van entre el 0.6 y el 0.85 (ver Figura 3.3). Mientras que la exactitud sigue una curva ascendente estabilizándose por encima de los 0.95, en cambio, para la validación los

resultados bajan notablemente, presentando bastante ruido, oscilando los resultados entre 0.60 y 0.75 en todas las particiones (ver Figura 3.4).

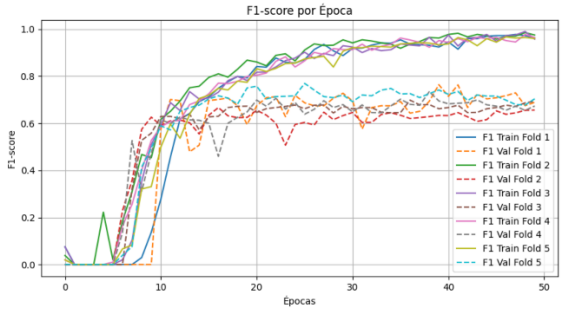

Figura 3.1. Medida- F de la validación cruzada

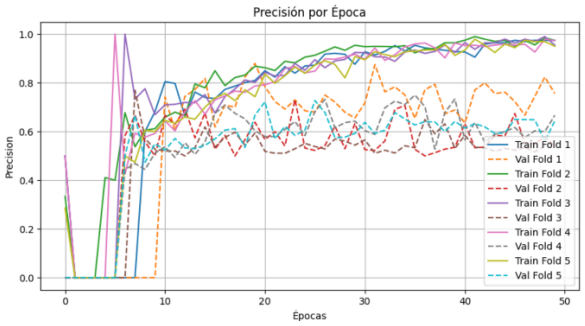

Figura 3.2. Precisión de la validación cruzada

Estos resultados sugieren que hay un sobreajuste en la CNN ya que el clasificador aprende bien los datos de entrenamiento, sin embargo, no logra mantener un buen rendimiento cuando predice nuevos datos (el conjunto de prueba), es decir, no está generalizando de forma correcta.

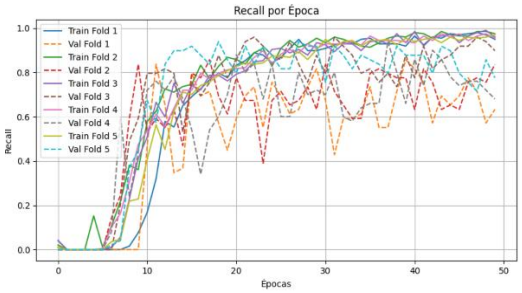

Figura 3.3. Recuperación de la validación cruzada.

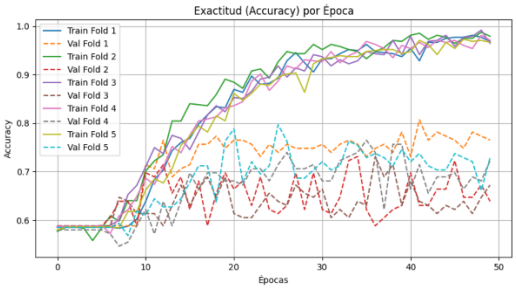

Figura 3.4. Exactitud de la validación cruzada.

En la Tabla 16 se presenta el rendimiento promedio de la CNN considerando los valores obtenidos en la última época de entrenamiento en cada una de las particiones. Se puede observar que durante el entrenamiento se obtienen medidas altas (i.e., 0.96), signo inequívoco de un sobreajuste dado que en la validación cruzada y en la prueba no generaliza correctamente, ya que en la recuperación obtiene un valor de 0.76 y 0.75, respectivamente.

Tabla 16. Rendimiento del clasificador - CNN.

	Validación cruzada *k - 5*		
Clasificador CNN	Entrenamiento	Validación	Prueba
Medida-F	0.96	0.68	0.70
Recuperación	0.96	0.76	0.75
Precisión	0.96	0.63	0.66
Exactitud	0.96	0.70	0.73

Nota: Donde *k* representa el número de particiones de los datos utilizadas.

Experimento 4

- Objetivo: Evaluar el rendimiento de la CNN entrenada con 50 épocas.

- Partición de datos: El 80% de los datos de PROMISE más el 50% de los datos de los casos de estudio para el entrenamiento; el resto (i.e., 20% y 50%, respectivamente) para pruebas.
- Clasificador utilizado: CNN.

Durante el entrenamiento se obtuvo un rendimiento cuasi perfecto, ya que tanto en la recuperación y como en la exactitud se obtuvieron puntajes de 1.00 mientras que en la medida-F y la precisión se obtuvo un puntaje de 0.99 (ver Tabla 17). Este resultado permitió identificar un sobreajuste en la red, dado que en los conjuntos de pruebas los resultados disminuyeron drásticamente, para el caso de PROMISE, donde la medida-F disminuye a 0.79, la precisión a 0.60 y la exactitud a 0.72, mientras que la recuperación, sigue siendo relativamente alta (0.94). Sin embargo, para la prueba con los casos de estudio, los valores de los resultados disminuyen aún más, indicando que la CNN tiene un comportamiento deficiente al clasificar correctamente los FRs preliminares.

Tabla 17. Rendimiento del clasificador - CNN.

Clasificador CNN	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
Medida-F	0.99	0.79	0.60
Recuperación	1.00	0.94	0.88
Precisión	0.99	0.60	0.45
Exactitud	1.00	0.72	0.47

Experimento 5

- Objetivo: Evaluar la capacidad de clasificación del modelo BERT.
- Datos: Para el entrenamiento se utilizó el 80% de los datos de PROMISE junto con el 50% de los datos de los casos de estudio. Para la prueba se utilizó el 20% y el 50% restante de los datos, respectivamente.
- Clasificador utilizado: BERT.

Durante el entrenamiento, BERT obtuvo los siguientes resultados: para la medida-F 0.97, para la recuperación 0.96, para la precisión 0.98 y, para la exactitud 0.97 (ver Tabla 18). Así mismo al realizar las pruebas con el conjunto de datos con PROMISE, BERT mantuvo el mismo rendimiento, reflejando su capacidad de generalización. Sin embargo, con el conjunto de datos de los casos de estudio, los resultados bajaron ligeramente (indicando un resultado equilibrado), lo cual sugiere que BERT puede adaptarse a distintos dominios, incluso cuanto hay diferencias entre los conjuntos de datos.

Tabla 18. Rendimiento del clasificador - BERT.

Clasificador BERT	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
Medida-F	0.97	0.92	0.81
Recuperación	0.96	0.90	0.77
Precisión	0.98	0.94	0.80
Exactitud	0.97	0.92	0.79

Experimento 6

- Objetivo: Evaluar el efecto de reducir el preprocesamiento en el rendimiento de BERT.

- Partición de datos: Se utilizó el 80% de los datos de PROMISE más el 50% de los datos de los casos de estudio para el entrenamiento y para la prueba, el 20% y el 50% respectivamente.
- Clasificador utilizado: BERT.

Con la finalidad de mejorar los resultados de BERT, se cambió el preprocesamiento de los conjuntos de datos, aplicando únicamente la conversión de mayúsculas a minúsculas y la expansión de palabras. Mediante esta experimentación se identificó que BERT trabaja mejor con un preprocesamiento simple, ya que logró mantener una medida-F de 0.98, la recuperación de 0.98, la precisión de 0.99 y la exactitud de 0.99 para el entrenamiento (ver Tabla 19). Para la prueba con los 20% de datos de PROMISE, obtuvo una medida-F de 0.89, una recuperación de 0.88, una precisión de 0.90 y una exactitud de 0.91. Mientras que, para el 50% de los datos de los casos de estudio, obtuvo una medida-F de 0.84, recuperación de 0.81, precisión de 0.88 y exactitud de 0.86, observando una mejora en comparación con todas las experimentaciones anteriores.

Tabla 19. Rendimiento del clasificador - BERT.

Clasificador BERT	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
Medida-F	0.98	0.89	0.84
Recuperación	0.98	0.88	0.81
Precisión	0.99	0.90	0.88
Exactitud	0.99	0.91	0.86

Estos resultados sugieren que con un preprocesamiento simple se facilitó una representación más clara de los FRs preliminares, mejorando la generalización de BERT con los datos de los casos de estudio, ya que, en todas las experimentaciones anteriores, los clasificadores SVM, LR, NB y CNN no lograban obtener puntajes promedios mayores al 0.80.

Experimento 7

- Objetivo: Optimizar BERT mediante una búsqueda de los valores óptimos de su hiperparámetros para mejorar su rendimiento.
- Partición de datos: Se utilizó el 80% de los datos de PROMISE más el 50% de los datos de los casos de estudios para el entrenamiento y para las pruebas los datos restantes (i.e., 20% y 50%, respectivamente).
- Clasificador utilizado: BERT ajustando los siguientes hiperparámetros:
    - *Learning_rate* con valores de 1e-5, 2e-5 y 3e-5, ya que de acuerdo con Devlin et al., (2019) esta tasa se define entre el 2e-5 al 5e-5, especialmente cuando se aplica un ajuste fino de las tareas de clasificación, y los valores definidos son cercanos a estos números.
    - *Weight_decay* con valores de 0, 1e-4, 1e-3, 1e-2, 5e-2 y 1e-1, ya que como menciona Liu et al., (2019) los valores más efectivos para este hiperparámetro oscilan entre 1e-2 y 1e-1. Para este experimento se definieron valores más amplios con la finalidad de realizar una búsqueda más extensa.

Estos resultados fueron visualizados mediante mapas de calor (i.e., es una representación visual de los datos donde los valores individuales de los de hiperparámetros se indican mediante colores: los más oscuros representan los mejores y los más claros los más bajos), lo que permitió identificar los valores óptimos para dichos hiperparámetros. En las Figuras de la 3.5 a la 3.8 se presentan los cuatro

mapas de calor correspondientes a la medida-F, a la precisión, a la exactitud y a la recuperación, respectivamente para el conjunto de datos de prueba de PROMISE. En la Figura 3.5 se observa que el valor máximo de la medida-F se alcanzó con un *learning_rate* de 3e-5 y un *weight_wecay* de 1e-3 y 1e-2. En la Figura 3.6 se presenta la precisión, donde el valor máximo obtenido es de 0.9466 con un *learning_rate* de 3e-5 y un *weight_wecay* de 1e-3 y 1e-2. Mientras que para la exactitud se logra el mejor valor con un *learning_rate* de 3e-5 y un *weight_wecay* de 1e-3 y 1e-2 (ver Figura 3.7). Para la recuperación se observa que con un *learning_rate* de 3e-5 y un *weight_wecay* de 1e-2 se obtiene el valor máximo de esta medida (ver Figura 3.8).

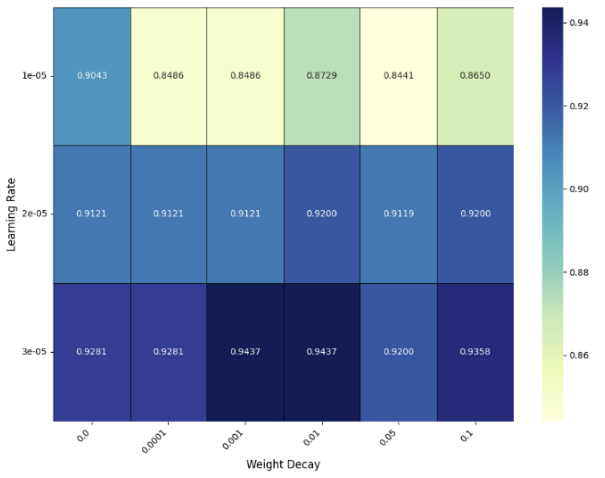

Figura 3.5. Medida- F de prueba del conjunto de datos PROMISE (20%).

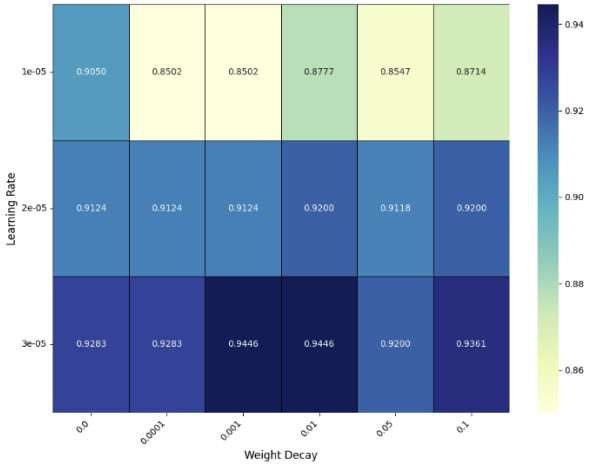

Figura 3.6. Precisión de prueba del conjunto de datos PROMISE (20%).

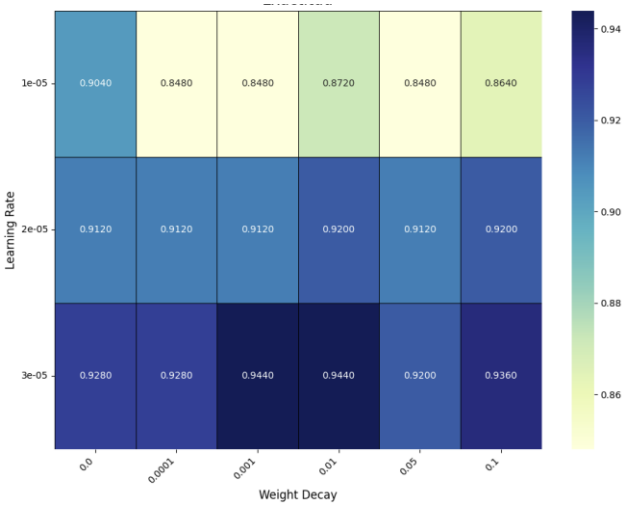

Figura 3.7. Exactitud de prueba del conjunto de datos PROMISE (20%).

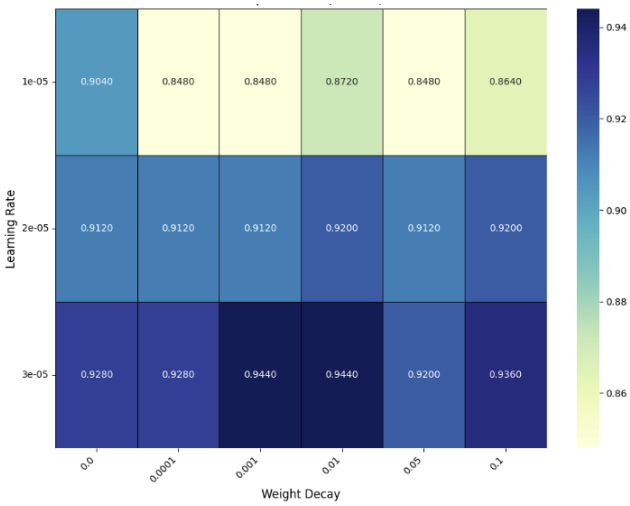

Figura 3.8. Recuperación de prueba del conjunto de datos PROMISE (20%).

Asimismo, se presentan los resultados de la evaluación sobre el conjunto de datos de prueba correspondientes a los casos de estudio. En las Figuras 3.9 al 3.12 se presentan los resultados de la medida-F, precisión, exactitud y recuperación, respectivamente, mediante mapas de calor. Se puede observar que, para todas las métricas, BERT alcanzó su valor máximo con un *learning_rate* de 3e-5 y un *weight_decay* de 0, 1e-4 y 1e-2.

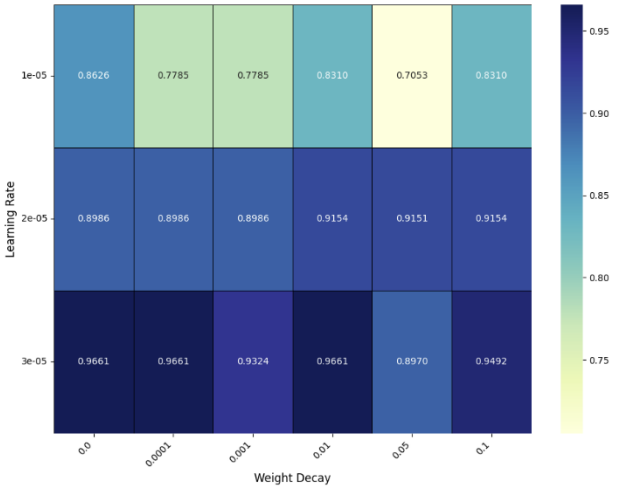

Figura 3.9. Medida- F de prueba del conjunto de datos de los casos de estudio (50%).

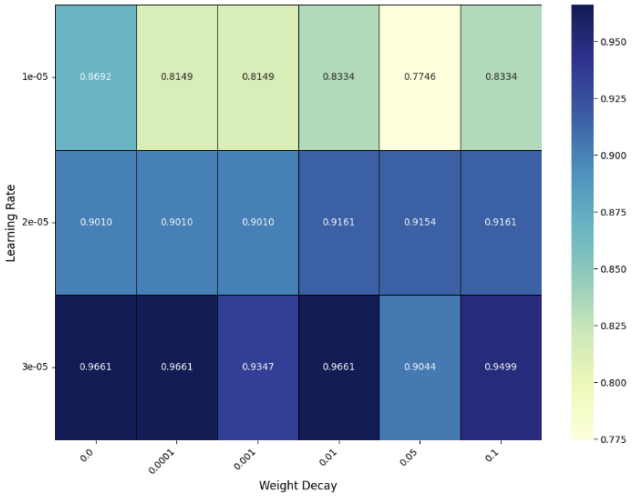

Figura 3.10. Precisión de prueba del conjunto de datos de los casos de estudio (50%).

Finalmente, en la Tabla 20 se presenta el resultado del modelo BERT con un *learning_rate* de 3e-5 y un *weight_decay* de 1e-2, en esta se puede observar que con el conjunto de datos de entrenamiento (80% de los datos de PROMISE y el 50% de conjunto de los casos de estudios) BERT obtuvo los valores máximos de 0.98 tanto para la medida-F como para la recuperación y, de 0.99 para la precisión y la exactitud, lo que evidenció su capacidad para clasificar correctamente los FRs preliminares.

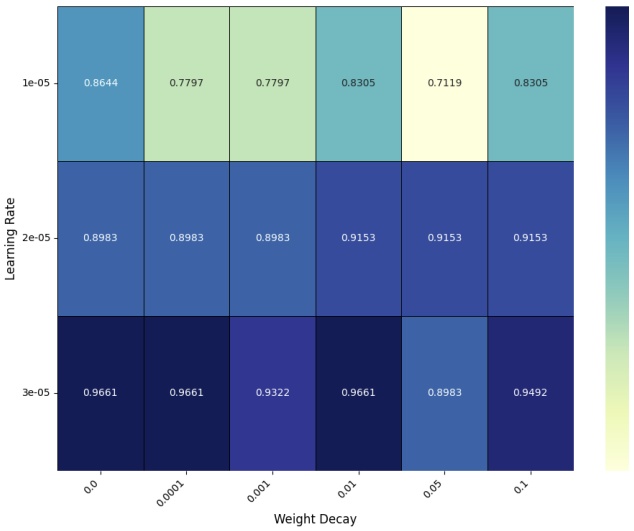

Figura 3.11. Exactitud de prueba del conjunto de datos de los casos de estudio (50%).

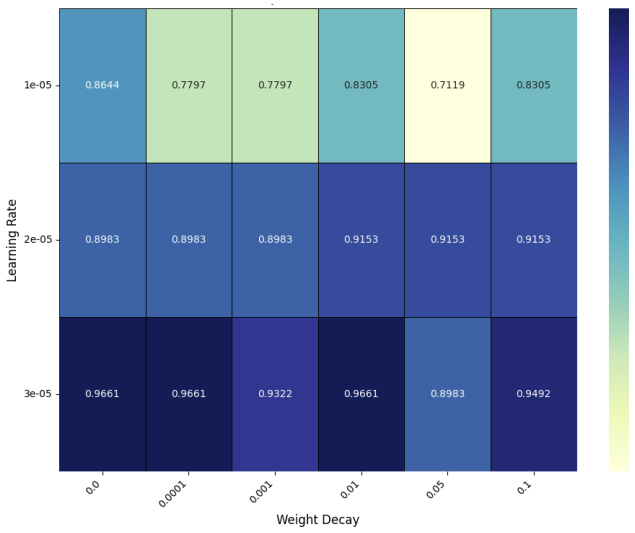

Figura 3.12. Recuperación de prueba del conjunto de datos de los casos de estudio (50%).

Respecto al conjunto de prueba de PROMISE, BERT mantuvo su habilidad para clasificar correctamente los requisitos preliminares obtuvo los valores máximos de 0.93 para la medida-F, 0.90 para la recuperación, 0.96 para la precisión y 0.94 para exactitud, mientras que, para el conjunto de los datos de los casos de estudio, los valores fueron superiores 0.96 tanto para la medida-F como para la recuperación y la precisión y de 0.97 para la exactitud, lo cual demostró nuevamente su capacidad para adaptarse a nuevos dominios y estructuras textuales.

Tabla 20. Recuperación de los clasificadores binarios.

Clasificador BERT	Entrenamiento	Prueba – Conjunto de datos PROMISE (20%)	Prueba - Conjunto de datos Casos de estudio (50%)
Medida-F	0.98	0.93	0.96
Recuperación	0.98	0.90	0.96
Precisión	0.99	0.96	0.96
Exactitud	0.99	0.94	0.97

3.3.3.6. Selección del clasificador final

A pesar de que los clasificadores tradicionales SVM, LR y NB mostraron resultados aceptables durante la validación cruzada; al ser evaluados sobre el conjunto de datos de los casos de estudio (conjunto de prueba), su desempeño disminuyó, indicando una capacidad limitada de generalización.

Por otro lado, en la CNN se observaron sobreajustes durante la fase de entrenamiento, ya que en esta se alcanzaron valores cercanos al 1.00, mientras que en la fase de prueba su rendimiento bajó significativamente, lo que evidenció que no logra predecir adecuadamente frente a datos desconocidos.

BERT tanto en el entrenamiento como en las pruebas presentó un balance entre las medidas, lo que lo convirtió en la opción a seleccionar, en este trabajo de tesis, para llevar a cabo la tarea de la clasificación de requisitos preliminares, dada su capacidad de generalizar entre distintos dominios, confirmando así su robustez como un clasificador basado en NL.

3.4. Implementación de la herramienta de *software* para generar resúmenes extractivos automáticos e identificar requisitos preliminares funcionales

Una vez seleccionado el clasificador BERT para la identificación exclusivamente de FRs preliminares, se procedió a implementar la herramienta de *software* a través de la cual se validará la solución planteada en esta tesis. Para llevar a cabo esta tarea, se utilizó la arquitectura cliente-servidor, dado que permite realizar varias peticiones al servidor al mismo tiempo (Nyabuto, 2024). Para la implementación del *backend*[34] de la herramienta, se utilizó el lenguaje de programación *Python* (versión 3.11.11). Ahora bien, para ejecutar el LLM de *Gemma* (ver Sección 3.3.1.) sin la necesidad de depender de servidores de IA se utilizó *LM Studio* (versión 0.3.9), ya que este permite levantar una API local para integrarlo a productos de desarrollo de *software*.

Mientras que el *frontend*[35] se desarrolló con *Svelte* (versión 5.0.0) que es un *framework* que permite crear aplicaciones e interfaces de usuario en la *web* con una sintaxis simple y sencilla. A continuación, se describe el funcionamiento que incorporará la herramienta. La herramienta denominada *IdentifyReq* permite adjuntar textos de texto plano en los formatos PDF, Word (ver Figura 3.13), que son procesados para generar automáticamente su resumen y posteriormente realiza la identificación y extracción de los FRs preliminares (ver Figura 3.14).

[34] Es parte de la herramienta que responde a las solicitudes de una funcionalidad, es decir, se encarga de la lógica y procesamiento, y este se encuentra del lado del servidor (Jalolov et al., 2024).

[35] Es la parte visible de la herramienta, en el *frontend* se crean las interfaces de usuario (Jalolov et al., 2024).

Figura 3.13. Pantalla de inicio.

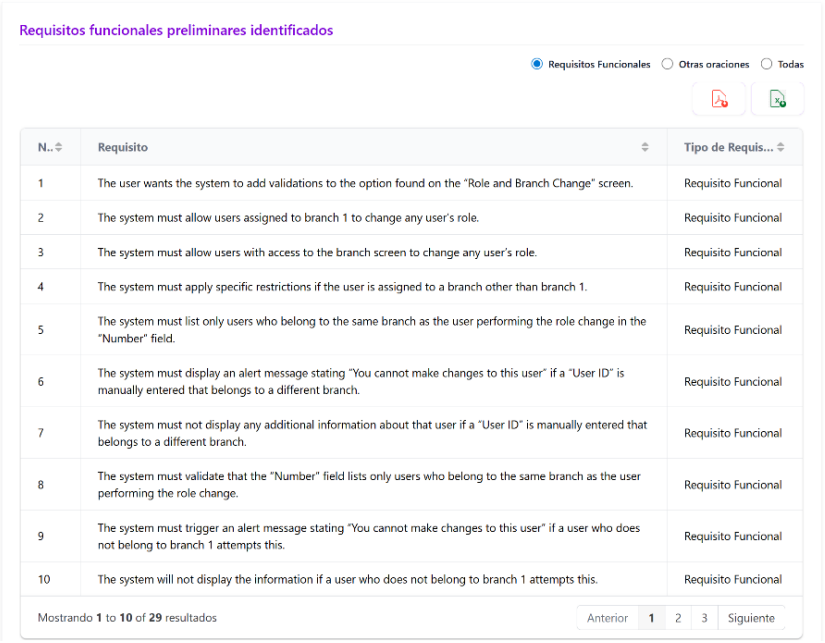

Figura 3.14. Pantalla de Resultado.

4. Desarrollo del caso de estudio

De acuerdo con Runeson y Höst (2009) y Rico (2024, abril), los estudios empíricos aplicables a la IS se centran en analizar fenómenos, artefactos y tecnologías del *software* en un contexto real. El objetivo de este tipo de estudios es que los investigadores realicen una evaluación de forma objetiva y científica, recopilando información para mejorar tanto el proceso de desarrollo como la gestión y, por ende, el producto (*software*) (Xu, 2017, junio; Zhang et al., 2018). Estos estudios empíricos se clasifican en tres tipos: experimentos controlados, casos de estudios y encuestas (*surveys*).

Ahora bien, en el contexto de esta tesis se decidió utilizar un caso de estudio, dado que este es un tipo de investigación que analiza uno o más ejemplos concretos de un fenómeno y utiliza múltiples fuentes de información, con la finalidad de comprender su comportamiento dentro de un contexto real y en un periodo de tiempo específico (Wohlin, 2021).

Wohlin et al. (2003) y Runeson et al. (2012) mencionan que un caso de estudio se puede evaluar de tres maneras:

1. Comparar los resultados obtenidos a partir de la propuesta contra una línea base.
2. Desarrollar dos proyectos en paralelo, conocidos como proyectos gemelos, seleccionando uno de estos como línea base para realizar la comparación.
3. Finalmente, implementar una nueva propuesta únicamente en ciertos componentes y comparar los resultados con aquellos que no fueron modificados

En este trabajo de tesis se seleccionó la primera estrategia, dado que el propósito es evaluar y validar la propuesta de la herramienta frente a un proyecto tomado como línea base, con la finalidad de minimizar sesgos y reforzar la validez del caso de estudio.

4.1. Antecedentes

Este caso de estudio fue llevado a cabo con datos proporcionados por una empresa ubicada en la Ciudad de México y que cuenta con aproximadamente 80 empleados. Dicha empresa cuenta con 5 áreas: comercial, fábrica de *software*, consultoría, recursos humanos, y contabilidad, así mismo, se dedica a la consultoría, desarrollo y mantenimiento de *software* bancario para algunas Sociedades Financieras Populares (SOFIPOS). Es importante mencionar que, por motivos de confidencialidad, durante el desarrollo del caso de estudio no se proporcionará el nombre de la empresa, solo se hará referencia a esta como EMP.

Cabe mencionar que EMP no hace uso de ningún estándar ni de plantillas para obtener FRs preliminares; sin embargo, proporcionó 16 documentos de requisitos preliminares que sirvieron como línea base en este caso de estudio.

4.2. Descripción del proyecto

El proyecto utilizado en este caso de estudio es un *software* bancario que cuenta con los siguientes módulos: cuentas de cliente, cajas, créditos, cuentas de ahorro, entre otros. En los documentos de requisitos preliminares proporcionados se contemplan integraciones de nuevas API, la generación de nuevas tablas y reportes, solicitud de créditos que incluyen la gestión de comisiones, así como la integración de inversiones.

Es importante mencionar que EMP realiza la elicitación de requisitos utilizando técnicas de elicitación como reuniones, entrevistas y cuestionarios. Esta tarea recae en el analista de requisitos, quien es el intermediario entre los desarrolladores y los *stakeholders*. Debe hacerse énfasis en que en EMP, los documentos de requisitos preliminares no tienen un formato especifico.

4.3. Diseño del caso de estudio

En este caso de estudio se utilizó como línea base los documentos de requisitos preliminares, que previamente obtuvieron tanto el director técnico como el analista de requisitos de EMP. Estas personas a partir de ahora se denominarán como "equipo de control", mientras que el equipo que utilizó *IdentifyReq* y que al igual que el grupo de control, estaba conformado por dos personas desempeñando los roles de director técnico y analista de requisitos, será denominado como "equipo experimental". Es importante hacer notar que el "equipo experimental" no tuvo ninguna comunicación con el "equipo de control", con la finalidad de evitar sesgos en los resultados.

Para este caso de estudio se plantearon las siguientes hipótesis:

H_{i1}: *IdentifyReq* mejorará la generación de resúmenes automáticos mediante la identificación de oraciones que representen FRs preliminares.

H_{i2}: *IdentifyReq* mejorará la extracción de FRs preliminares a partir de los resúmenes automáticos generados.

H_{i3}: *IdentifyReq* asegurará la singularidad de los FRs preliminares obtenidos.

Para analizar el desempeño de *IdentifyReq* en la aplicación del caso de estudio, se establecieron las siguientes variables de respuesta que se derivan de cada una de las hipótesis definidas para el caso de estudio:

Correspondiente a H_{i1}:

- Número de oraciones del resumen extractivo que contienen alguna de las palabras claves ("*need*", "*has*", "*want*", "*have*", "*should*", "*require*", "*demand*" y, "*must*").

Correspondiente a H_{i2}:

- Número de oraciones del resumen extractivo que tienen una estructura gramatical correcta (i.e., artículo, sustantivo, verbo y complemento).
- Número de oraciones que representan un requisito preliminar de acuerdo con el constructor del estándar ISO/IEC/IEEE 29148 (2018).

Correspondiente a H_{i3}:

- Número de FRs preliminares que cumplen con la característica de calidad de la singularidad, de acuerdo con el estándar ISO/IEC/IEEE 29148 (2018).

Ahora bien, con el objetivo de responder a estas variables, se examinaron los 16 documentos del proyecto descrito en el punto 4.1 y, se analizaron los resultados obtenidos al utilizar *IdentifyReq* (i.e., los resúmenes extractivos y la lista generada con los FRs preliminares).

4.4. Recopilación de datos

Como se mencionó anteriormente, EMP proporcionó 16 documentos categorizados como requisitos preliminares, de este conjunto inicial se seleccionaron aleatoriamente 9 documentos, los cuales fueron utilizados para el desarrollo del caso de estudio. Esta selección permitió obtener una muestra representativa de los distintos tipos de desarrollos manejados por la empresa.

A partir de los documentos seleccionados como línea base, el equipo de control elaboró manualmente los resúmenes de cada documento, asegurándose de capturar aspectos importantes de cada documento. Posteriormente, dicho grupo realizó la identificación de forma manual de los FRs preliminares de acuerdo con el constructor del estándar ISO/IEC/IEEE 29148 (2018), y finalmente, la revisión para determinar cuántos de estos requisitos preliminares cumplían con la característica de singularidad. Una investigadora[36] experta verificó si las actividades realizadas por el grupo de control se realizaron correctamente.

Ahora bien, respecto al grupo experimental, la medición del proceso de identificación de FRs mediante la generación automática de resúmenes a partir de listas de requisitos preliminares utilizando *IdentifyReq*, se contextualizó en las tres actividades que están involucradas. A continuación, se describe cada una de ellas:

4.4.1. Generación automática de resúmenes

Como primer paso para utilizar *IdentifyReq,* el o la ingeniero(a) de requisitos debió hacer *clic* sobre el botón “Procesar” (ver Figura 4.1), para que, se generara un resumen extractivo donde no solo se sintetizó la información del documento original, sino que además estaba conformado por oraciones que presentaban un alto potencial de ser FRs preliminares ya que incluían las palabras claves definidas en el punto 3.2.1. de este trabajo de tesis.

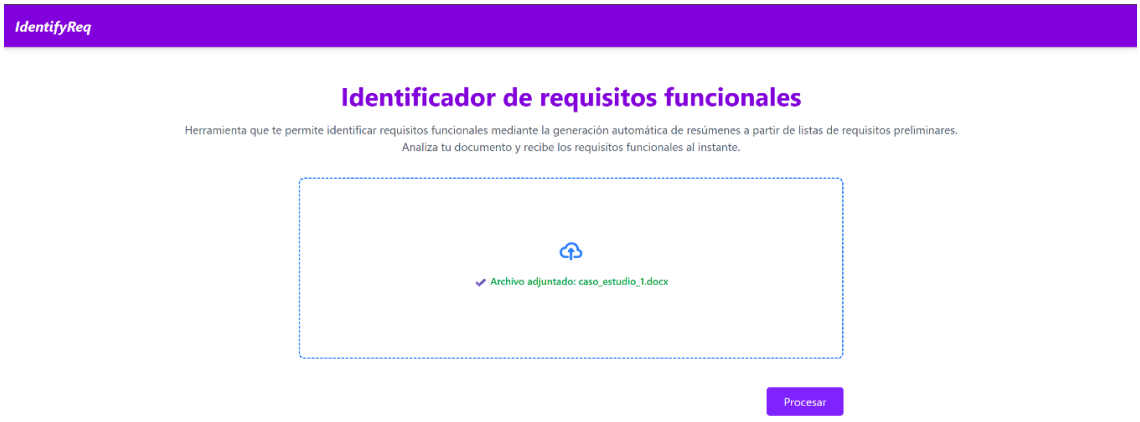

Figura 4.1. Pantalla de inicio con archivo adjuntado.

[36] La investigadora experta es una profesora con más de 20 años de experiencia y no fue parte del grupo experimental, así que no sabía que su trabajo era parte del caso de estudio.

En la Figura 4.2 se pueden ver tanto el resumen de referencia (i.e., generado por el grupo de control) como el generado por *IdentifyReq* (donde se resaltan en amarillo las palabras claves contenidas en las oraciones).

The user wants validations to be added to the option found on the "Role and Branch Change" screen. The system must consider that users who are assigned to branch 1 and have access to the branch screen will be able to change any user's role; however, if the user is assigned to a branch other than branch 1, the following restrictions will apply:

- In the "Number" field, only users who belong to the same branch as the user performing the role change must be listed.
- If a "User ID" is manually entered that belongs to a different branch, the system must display an alert message: "You cannot make changes to this user", and it must not display any additional information about that user.

The system must validate that in the "Number" field only users corresponding to the branch of the user who is performing the role change will be listed. If a user who does not belong to branch 1 attempts this, the system must trigger an alert message stating "You cannot make changes to this user", and therefore it will not display the information.
If the user performs a search by name matching, only those users who belong to the branch of the person conducting the search must be listed; if the user is a person from branch 1, all users that match the criterion must be listed without distinguishing by branch.
Role changes on the "Role and Branch Change" screen may only be made in accordance with the roles that have been previously authorized through parameterization.
Role parameterization can be done on a new screen, which will contain the following elements: Role Field", "Select All", "Roles Section" and "Save" button.
Each of the fields is described below.

- "Role Field": Allows searching for a role by ID or description, to which the roles it may modify will be assigned.
- "Select All" checkbox: Allows selecting all roles in the list.
- "Roles Section": Will display a list of all roles registered in the system.
- "Save" button: When pressing this button, the relationship between the selected role and the corresponding checked roles will be saved.

When making a role or branch change, the system must store not only the date, but also the exact time with minutes and seconds. This information must appear in the report in the "Change Date" column in the following format:

- YYYY-MM-DD HH:MM:SS
- Example: 2024-04-30 12:20:22.

In addition, the user requires adding a new field called "SAT Registered Name", which will be a free-text field with a maximum of 150 characters. The new field will be placed in the following locations depending on the person type:

- Individual
- Individual with Business Activity
- Legal Entity.

Stamping should be performed as follows: the system must place in the "Full Name" field the data captured in the "SAT Registered Name" field; in this way, when the stamping request is made, the data that will be sent is the name under which the person is registered with the SAT. Keep this in mind. When saving the customer's name in the account statement PDF, on the system screen it must appear exactly as registered in the Customer Onboarding, that is, the full name resulting from concatenating:

- For Individuals and Individuals with Business Activity, fields: First Name, Second Name, Third Name, Paternal Surname, Maternal Surname.

When the data of a "Legal Entity" type user is modified, it should be considered that if the "SAT Registered Name" field is empty, the system must allow the user to save in the tables the data corresponding to the customer's full name, that is, the full name resulting from concatenating:

- First Name, Second Name, Third Name, Paternal Surname, Maternal Surname. For Legal Entities, field: Corporate Name (Razón Social).

Now, for the "Bankingly Channels – Accounts" module, the creation of 4 APIs is required to query information related to customers' accounts.

- For the first API, a REST web service named GetAccountDetails is needed; this WS will be used to obtain detailed information on any type of account.
- For the second API, the creation of the WS Accounts_Accounts_GetAccountDetails is needed so that detailed information can be obtained for any type of account; the data indicated in the GetAccountDetails tab must be sent.
- For the third API, a REST web service named GetAccountLast5Movements is needed; this WS will be used to obtain the last 5 movements of an account. The returned information must be ordered by date in descending order. A maximum of 5 objects must be returned in that list.
- Finally, the creation of the WS Accounts_Accounts_GetAccountLast5Movements is required to obtain the last 5 movements of an account.

In addition to the previous APIs, the following APIs need to be created:

- The user needs the creation of a REST web service named GetAccountMovements; this WS will be used to obtain (paginated) movements of an account based on the indicated filters.
- The creation of the WS Accounts_Accounts_GetAccountMovements is required to obtain the (paginated) movements of an account based on the filters indicated, so the data indicated in the GetAccountMovements tab must be sent.
- The creation of a REST web service named GetTransactionVoucher is required; this WS will be used to obtain the vouchers/receipts for the movements of an account.
- The creation of the WS Accounts_Accounts_GetTransactionVoucher is required to obtain the vouchers/receipts for the movements of an account.

The user wants validations to be added to the option found on the "Role and Branch Change" screen. The system must consider that users who are assigned to branch 1 and have access to the branch screen will be able to change any user's role; however, if the user is assigned to a branch other than branch 1, the following restrictions will apply: In the "Number" field, only users who belong to the same branch as the user performing the role change must be listed. If a "User ID" is manually entered that belongs to a different branch, the system must display an alert message: "You cannot make changes to this user", and it must not display any additional information about that user. The system must validate that in the "Number" field only users corresponding to the branch of the user who is performing the role change will be listed. If a user who does not belong to branch 1 attempts this, the system must trigger an alert message stating "You cannot make changes to this user", and therefore it will not display the information. If the user performs a search by name matching, only those users who belong to the branch of the person conducting the search must be listed; if the user is a person from branch 1, all users that match the criterion must be listed without distinguishing by branch. Role changes on the "Role and Branch Change" screen may only be made in accordance with the roles that have been previously authorized through parameterization. When making a role or branch change, the system must store not only the date, but also the exact time with minutes and seconds. This information must appear in the report in the "Change Date" column in the following format: YYYY-MM-DD HH:MM:SS Example: 2024-04-30 12:20:22. In addition, the user requires adding a new field called "SAT Registered Name", which will be a free-text field with a maximum of 150 characters. Stamping should be performed as follows: the system must place in the "Full Name" field the data captured in the "SAT Registered Name" field; in this way, when the stamping request is made, the data that will be sent is the name under which the person is registered with the SAT. When saving the customer's name in the account statement PDF, on the system screen it must appear exactly as registered in the Customer Onboarding, that is, the full name resulting from concatenating: For Individuals and Individuals with Business Activity, fields: First Name, Second Name, Third Name, Paternal Surname, Maternal Surname. When the data of a "Legal Entity" type user is modified, it should be considered that if the "SAT Registered Name" field is empty, the system must allow the user to save in the tables the data corresponding to the customer's full name, that is, the full name resulting from concatenating: First Name, Second Name, Third Name, Paternal Surname, Maternal Surname. Now, for the "Bankingly Channels – Accounts" module, the creation of 4 APIs is required to query information related to customers' accounts. For the first API, a REST web service named GetAccountDetails is needed; this WS will be used to obtain detailed information on any type of account. For the second API, the creation of the WS Accounts_Accounts_GetAccountDetails is needed so that detailed information can be obtained for any type of account; the data indicated in the GetAccountDetails tab must be sent. For the third API, a REST web service named GetAccountLast5Movements is needed; this WS will be used to obtain the last 5 movements of an account.

Figura 4.2. Pantalla de inicio con archivo adjuntado.

4.4.2. Segmentación de oraciones de requisitos preliminares

Después de crear el resumen extractivo, *IdentifyReq* generó automáticamente nuevas oraciones que cumplían con la característica de singularidad del estándar ISO/IEC/IEEE 29148 (2018) (i.e., cada oración debe representar si y solo si un solo requisito, evitando el uso de conjunciones) mediante el uso de un clasificador binario. De esta forma se asegura su identificación como posibles FRs preliminares. En la Figura 4.3 se puede apreciar de lado izquierdo el resumen extractivo generado y de lado derecho (recuadro de color naranja) los requisitos preliminares singulares ya seleccionados.

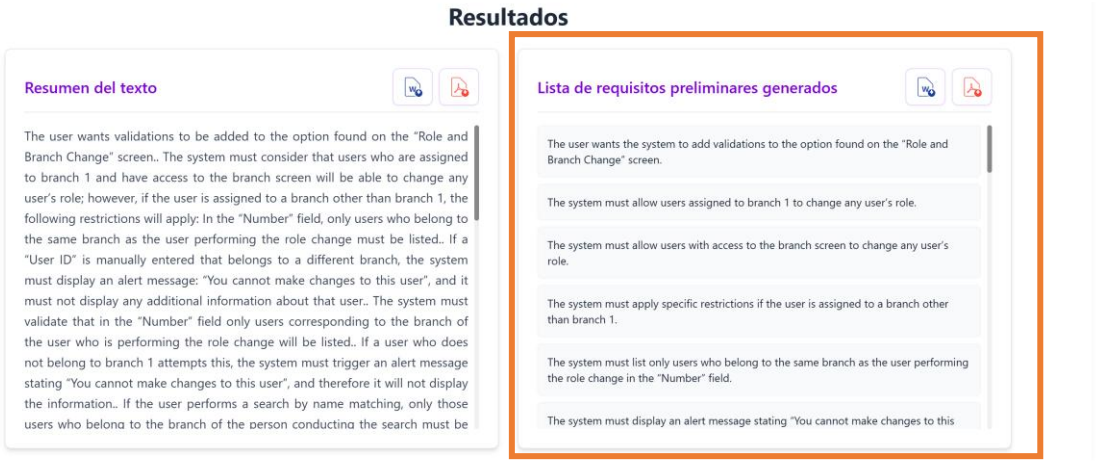

Figura 4.3. Resumen extractivo y lista de requisitos preliminares generados por *IdentifyReq*.

4.4.3. Identificación de requisitos funcionales preliminares singulares

Una vez que *IdentifyReq* generó la lista de oraciones singulares que contenían alguna de las palabras claves, siguió con la extracción de los FRs preliminares mediante un clasificador binario de BERT (procedimiento que se detalló en la Sección 3.3.3.5). De esta manera, se obtuvo una lista de FRs preliminares por cada documento (ver Figura 4.4).

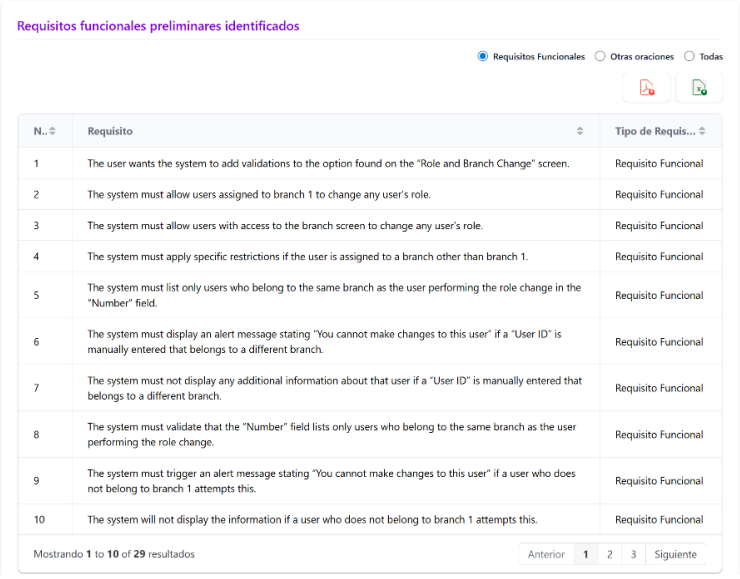

Figura 4.4. Requisitos preliminares funcionales.

4.5. Análisis de datos

Una vez que se tuvieron recopilados todos los datos obtenidos por *IdentifyReq*, se aplicó, por parte de la investigadora experta, un análisis cuantitativo para su validación. Esta acción implicó revisar los requisitos preliminares, tanto de la línea base como los obtenidos con *IdentifyReq*, para evaluar su concordancia con el constructor y la definición de singularidad de un FR, del estándar ISO/IEC/IEEE 29148 (2018). Una vez concluida esta validación, los datos analizados se procesaron y se generó el reporte de resultados.

4.6. Informe de resultados

A continuación, se presentan los resultados obtenidos al utilizar *IdentifyReq* sobre los documentos de requisitos preliminares proporcionados por EMP, donde se detallan los hallazgos encontrados en cada una de sus tres actividades. El objetivo fue contrastar los resultados generados automáticamente frente a los obtenidos de forma manual, con el propósito de validar las hipótesis planteadas (H_{i1}, H_{i2} y H_{i3}) y analizar la efectividad de la herramienta en la identificación de FRs preliminares.

4.6.1. Generación automática de resúmenes

Para esta actividad, como se mencionó anteriormente, se seleccionaron aleatoriamente 9 de los 16 documentos de requisitos proporcionados por EMP. En la Tabla 21 puede apreciarse la comparación entre el número total de oraciones que contenían los documentos originales contra los resúmenes generados tanto de forma manual como por *IdentifyReq*. Esto con el objetivo de evaluar no solo su capacidad de síntesis sino la de incluir oraciones que contuvieran las palabras clave definidas anteriormente y que, por ende, pudiesen denotar un FR preliminar. El número de oraciones en los documentos originales se encontraba en un rango de 18 y 53, mientras que en los resúmenes generados por *IdentifyReq* variaron entre 13 y 49 oraciones, y en los resúmenes manuales había entre 10 y 39 oraciones. Estos resultados reflejan que *IdentifyReq* produce resúmenes más compactos, pero con una mayor densidad informativa, al incluir principalmente oraciones con alta probabilidad de representar FRs, lo que permitió reducir el tamaño del texto, pero manteniendo las oraciones más relevantes.

Tabla 21. Número de oraciones en los resúmenes.

Documento proporcionado por EMP	Oraciones en el texto original	Oraciones en el resumen *IdentifyReq*	Oraciones en el resumen manual
1	33	**17**	17
2	18	**13**	15
3	38	**31**	26
4	36	**13**	16
5	44	**26**	24
6	39	**29**	30
7	25	**13**	10
8	53	**49**	39
9	37	**30**	33

4.6.2. Segmentación de oraciones de requisitos preliminares

Una vez generados los resúmenes: de referencia, por el grupo de control y el extractivo, por el grupo experimental utilizando *IdentifyReq*; se realizó una segmentación de las oraciones que no cumplían con la característica de singularidad para que la cumpliesen, eliminando así el uso de conjunciones que pudieran unir varias ideas en una misma oración, para que representara una idea única sin alterar su significado. Este paso permitió la posterior identificación de FRs preliminares.

En la Tabla 22 se compara el número de oraciones que contenía originalmente cada uno de los documentos proporcionados por EMP, contra el número de oraciones que se generaron después de aplicar la segmentación para el grupo experimental. Cabe mencionar que esta segmentación no fue realizada por el grupo de control, ya que EMP no contempla este proceso.

Tabla 22. Resultados después de la segmentación de oraciones.

Documento proporcionado por EMP	Número de oraciones originales	Número de oraciones segmentadas
1	33	**53**
2	18	**47**
3	38	**109**
4	36	**57**
5	44	**55**
6	39	**65**
7	25	**32**
8	53	**126**
9	37	**73**

4.6.3. Identificación de requisitos preliminares funcionales

Finalmente, *IdentifyReq* identificó aquellas oraciones que representaban FRs preliminares de acuerdo con el constructor del estándar ISO/IEC/IEEE 29148 (2018). Los resultados muestran que, para los 9 documentos de requisitos utilizados en este caso de estudio, el equipo de control identificó un total de 156 FRs, mientras que el experimental al usar *IdentifyReq* logró detectar 486 FRs (Ver tabla 23).

Así mismo se evaluó si *IdentifyReq* identificaba FRs singulares (i.e., que cada oración representara una única necesidad o función del sistema). Los resultados de la Tabla 24 muestran una diferencia notoria ya que en la revisión manual llevada a cabo por el grupo de control se obtuvieron 63 requisitos singulares y 93 no singulares; mientras que el grupo experimental utilizando *IdentifyReq* encontró 472 requisitos singulares y solo 14 no singulares. Esto demuestra que el proceso de segmentación automática mediante el LLM de Gemma fue efectivo para dividir las oraciones compuestas en oraciones simples sin perder coherencia semántica. De esta manera, la herramienta contribuyó a mejorar la calidad del conjunto de requisitos, alineándose con la característica de singularidad definida por el estándar ISO/IEC/IEEE 29148 (2018).

Tabla 23. Requisitos funcionales obtenidos.

Caso de estudio	Requisitos preliminares funcionales manuales	Requisitos preliminares funcionales *IdentifyReq*
1	10	**43**
2	13	**35**
3	25	**61**
4	8	**51**
5	21	**46**
6	26	**65**
7	7	**20**
8	24	**103**
9	22	**62**

Tabla 24. Requisitos preliminares funcionales singulares y no singulares.

Caso de estudio	No singulares manuales	Singulares manual	Total manual	No singulares *IdentifyReq*	Singulares *IdentifyReq*	Total *IdentifyReq*
1	5	5	10	2	**41**	43
2	9	4	13	2	**33**	35
3	20	5	25	0	**61**	61
4	5	3	8	6	**45**	51
5	5	16	21	3	**43**	46
6	17	9	26	0	**65**	65
7	5	2	7	0	**20**	20
8	15	9	24	1	**102**	103
9	12	10	22	0	**62**	62
Totales	**93**	**63**	**156**	**14**	**472**	**486**

En este sentido, se puede observar en los resultados obtenidos al utilizar *IdentifyReq*, un aumento significativo en la cantidad de FRs preliminares identificados al mejorar el proceso de extracción de estos. Por esta razón, se puede aceptar como valida la hipótesis establecida en la sección 1.2 del Capítulo 1 de esta tesis. Sin embargo, para confirmar con mayor certeza la hipótesis se recomienda replicar el caso de estudio con otras empresas desarrolladoras de *software*.

4.7. Conclusiones sobre los resultados de la evaluación

Para validar las hipótesis de este caso de estudio fue el de comparar los resultados obtenidos en la identificación y extracción de los FRs preliminares mediante la herramienta *IdentifyReq* y la línea base. Ahora bien, siguiendo las recomendaciones de Runeson y Höst (2009) y Runeson et al. (2012) para los estudios empíricos en IS se presentan las siguientes amenazas de validez:

- Validez interna. Evalúa si los resultados observados ocurren realmente por los factores investigados y no por variables externas no controladas. Por lo tanto, ninguno de los *stakeholders* participantes tuvo conocimiento previo acerca de la preparación del caso de estudio para evitar así, modificar sus prácticas tradicionales y, por lo tanto, ocasionar un sesgo en las mediciones. No obstante, no se puede asegurar que el entusiasmo del grupo experimental por utilizar la herramienta *IdentifyReq* no haya influido en los resultados. También se reconoce

que factores como la calidad del texto o la complejidad del dominio pueden haber influido en los resultados obtenidos.

- Validez externa. Se relaciona con el grado en que los resultados de este caso de estudio pueden aplicarse a otros contextos o proyectos de desarrollo de *software*. Dado que los resultados corresponden a un conjunto limitado de casos de estudio y las características únicas de EMP: tamaño de la empresa (i.e., con no más de 80 empleados), tamaño y complejidad del proyecto, así como el dominio de aplicación del proyecto, no es posible afirmar que los hallazgos sean representativos a todos los contextos de desarrollo de *software*. Sin embargo, los resultados pueden ser relevantes para proyectos con características similares a EMP, pero hace falta replicar este estudio con un mayor número de casos y diferentes tipos de proyectos para validar los hallazgos.
- Validez de construcción. Se refiere a que tan adecuadas son las medidas utilizadas que representan los conceptos teóricos que se investigan. En este caso de estudio, se recopilaron datos cuantitativos utilizando las variables de respuesta definidas en la Sección 4.3, garantizando así que los resultados sean los mismos, independientemente de la persona que los analice. Es importante tener en cuenta que los datos recopilados reflejaron solo el comportamiento del grupo experimental durante las actividades referentes a la generación de resúmenes extractivos e identificación de FRs preliminares singulares de acuerdo con el constructor para FRs del estándar ISO/IEC/IEEE 29148 (2018) y que, no se puede asumir que se pudiesen obtener los mismos resultados en etapas posteriores del desarrollo de *software* (e.g., diseño y codificación).

5. Conclusiones y trabajo futuro

El éxito de un proyecto de *software* depende fundamentalmente de la adecuada definición de requisitos, el cumplimiento de las funcionalidades especificadas, la gestión eficiente del presupuesto y la entrega dentro de los plazos estipulados. Sin embargo, la literatura reporta que los proyectos de *software* suelen fracasar debido a deficiencias en alguno de estos aspectos (Standish, 2020).

De acuerdo con Kumari y Pillai (2013), Sharma y Pandey (2014), y Dar et al. (2018, 2022), existen problemáticas recurrentes en el desarrollo de proyectos de *software*, especialmente en el proceso de elicitación de requisitos. Esto se atribuye al uso del NL, el cual puede resultar ambiguo o incompleto, conllevando a una definición inadecuada de los requisitos, incumplimiento de los objetivos, malentendidos entre *stakeholders* e impactos negativos en la calidad y éxito del desarrollo.

En consecuencia, el proceso de gestión de requisitos es crítico dentro del ciclo de vida de los proyectos de *software*, ya que su ejecución deficiente incrementa el riesgo de fracaso. La identificación de FRs preliminares a partir de documentos obtenidos mediante técnicas de elicitación representa una tarea compleja, dado el carácter ambiguo del lenguaje empleado.

Tal como se expuso en la sección de trabajos relacionados, solo se encuentra una propuesta previa basada en técnicas de resumen extractivo aplicadas a transcripciones de sesiones de elicitación; sin embargo, este enfoque no contempla la selección ni la identificación automática de oraciones representativas de FRs preliminares. Considerando los objetivos establecidos en el capítulo 1 y los resultados presentados en el capítulo 4, se derivan las siguientes conclusiones:

- La propuesta desarrollada en esta tesis demuestra la viabilidad de identificar oraciones potencialmente representativas de requisitos preliminares conforme al estándar ISO/IEC/IEEE 29148 (2018), utilizando el NLP.
- El uso de resúmenes extractivos contribuyó a reducir información redundante e irrelevante, centralizando oraciones pertinentes para la identificación de requisitos, lo cual fortaleció el proceso de detección de FRs preliminares.
- Los resultados del estudio de caso evidencian que el enfoque propuesto puede asistir a los ingenieros de requisitos durante la fase de identificación, facilitando la revisión de documentos de elicitación y la selección de oraciones relevantes.
- La efectividad de la herramienta está condicionada por la calidad y estructura del texto de entrada, pudiendo verse afectada en contextos con vocabulario limitado o frases altamente ambiguas.
- Se presenta la herramienta propuesta como una alternativa tecnológica viable para apoyar la identificación de requisitos, principalmente en el área de IR, seleccionando oraciones con potencial de representar requisitos en los resúmenes generados.

- Si bien el estudio de caso se realizó bajo condiciones académicas controladas, los resultados son alentadores para su futura aplicación en proyectos diversos, siempre que dispongan de documentación adecuada de elicitación.
- Asimismo, si bien la propuesta demostró utilidad en la identificación de FRs preliminares, los resultados no permiten aún generalizar su aplicabilidad a otros dominios; se recomienda, por tanto, la realización de pruebas adicionales en proyectos de diferente escala y complejidad.

Finalmente, como líneas futuras de trabajo se plantean:

- Integrar técnicas más avanzadas de NLP, como modelos basados en transformadores, para la generación automática de resúmenes, incrementando la capacidad de detección de palabras clave según el estándar ISO/IEC/IEEE 29148 (2018).
- Aplicar el enfoque propuesto en estudios de caso industriales de distinta naturaleza, tamaño y complejidad, con el objetivo de evaluar el rendimiento y riesgo asociado al uso de la herramienta.
- Incorporar métricas automáticas de calidad de requisitos, tales como trazabilidad, completitud y consistencia, para evaluar objetivamente los resultados obtenidos.
- Mejorar la herramienta mediante módulos adicionales que faciliten la administración y gestión integral de los resultados generados.

6. Bibliografía

Abdel-Salam, S., y Rafea, A. (2022). Performance study on extractive text summarization using BERT models. *Information*, *13*(2), 67.

Abo-Bakr, H., y Mohamed, S. A. (2023). Automatic multi-documents text summarization by a large-scale sparse multi-objective optimization algorithm. *Complex & Intelligent Systems*, *9*(4), 4629-4644.

Abualigah, L., Bashabsheh, M. Q., Alabool, H., y Shehab, M. (2019). Text summarization: a brief review. *Recent Advances in NLP: the case of Arabic language*, 1-15.

Adamopoulou, E., y Moussiades, L. (2020). An overview of chatbot technology. In *IFIP international conference on artificial intelligence applications and innovations* (pp. 373-383). Springer, Cham.

Adewumi, T. P., Liwicki, F., y Liwicki, M. (2020). Word2vec: Optimal hyper-parameters and their impact on nlp downstream tasks. *arXiv preprint arXiv:2003.11645*.

Afaq, S., y Rao, S. (2020). Significance of epochs on training a neural network. *Int. J. Sci. Technol. Res*, *9*(06), 485-488.

Agrawal, P. (2024, marzo). A Survey on Hyperparameter Optimization of Machine Learning Models. In *2024 2nd International Conference on Disruptive Technologies (ICDT)* (pp. 11-15). IEEE.

Agredo-Delgado, V., Ruiz, P. H., Garzón, L. E., y Collazos, C. A. (2023). A strategy for building shared understanding in requirements engineering activities. *Electronic Journal of SADIO (EJS)*, *22*(3), 41-54.

Akhmetov, I., Mussabayev, R., y Gelbukh, A. (2022). Reaching for upper bound ROUGE score of extractive summarization methods. *PeerJ C*

Akhmetov, I., Pak, A., Ualiyeva, I., y Gelbukh, A. (2020). Highly language-independent word lemmatization using a machine-learning classifier. *Computación y Sistemas*, *24*(3), 1353-1364.

Akram, F., Ahmad, T., y Sadiq, M. (2024). Recommendation systems-based software requirements elicitation process—a systematic literature review. *Journal of Engineering and Applied Science*, *71*(1), 29.

Akter, M., Bansal, N., y Karmaker, S. K. (2022). Revisiting automatic evaluation of extractive summarization task: Can we do better than ROUGE? In *Findings of the Association for Computational Linguistics: ACL 2022* (pp. 1547-1560).

Alami Merrouni, Z., Frikh, B., y Ouhbi, B. (2023). EXABSUM: a new text summarization approach for generating extractive and abstractive summaries. *Journal of Big Data*, *10*(1), 163.

Alami, N., Mallahi, M. E., Amakdouf, H., y Qjidaa, H. (2021). Hybrid method for text summarization based on statistical and semantic treatment. *Multimedia Tools and Applications*, *80*, 19567-19600.

Ali, N., Cai, H., Hamou-Lhadj, A., y Hassine, J. (2019). Exploiting Parts-of-Speech for effective automated requirements traceability. *Information and software technology*, *106*, 126-141.

Alkhomsan, M. N., Baslyman, M., y Alshayeb, M. (2024). Eliciting and modeling emotional requirements: a systematic mapping review. *PeerJ Computer Science*, *10*.

Alzayed, A., y Al-Hunaiyyan, A. (2021). A bird's eye view of natural language processing and requirements engineering. *International Journal of Advanced Computer Science and Applications*, *12*(5).

Ambriola, V., y Gervasi, V. (1997). Processing natural language requirements. In *Proceedings 12th IEEE International Conference Automated Software Engineering* (pp. 36-45). IEEE.

Andry, J. F., Tannady, H., y Nurprihatin, F. (2020, March). Eliciting requirements of order fulfilment in a company. In *IOP Conference Series: Materials Science and Engineering* (Vol. 771, No. 1, p. 012023). IOP Publishing.

Anwar, H., Khan, S. U. R., Iqbal, J., y Akhunzada, A. (2022). A tacit-knowledge-based requirements elicitation model supporting COVID-19 Context. *IEEE Access*, *10*, 24481-24508.

Arellano, A., Carney, E., y Austin, M. A. (2015). Natural language processing of textual requirements. In *The Tenth International Conference on Systems (ICONS 2015), Barcelona, Spain* (pp. 93-97).

Arthi, K., Sankaradass, V., Parveen, N., y Muralidharan, J. (2024). 7 Methods of cross-validation and bootstrapping.

Ashfaq, F., y Bajwa, I. S. (2021). Natural language ambiguity resolution by intelligent semantic annotation of software requirements. *Automated Software Engineering*, *28*(2), 13.

Attanayaka, B., Nawinna, D., Manathunga, K., y Abeygunawardhana, P. K. (2022, December). Success Factors of Requirement Elicitation in the Field of Software Engineering. In *2022 4th International Conference on Advancements in Computing (ICAC)* (pp. 240-245). IEEE.

Awasthi, I., Gupta, K., Bhogal, P. S., Anand, S. S., y Soni, P. K. (2021). Natural language processing (NLP) based text summarization-a survey. In *2021 6th International Conference on Inventive Computation Technologies (ICICT)* (pp. 1310-1317). IEEE.

Bano, M. (2015). Addressing the challenges of requirements ambiguity: A review of empirical literature. In *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)* (pp. 21-24). IEEE.

Barbella, M., y Tortora, G. (2022). Rouge metric evaluation for text summarization techniques. *Available at SSRN 4120317*.

Barrientos Núñez, I., y Carballo Muñoz, L. (2021). El uso de ontologías como apoyo a la Ingeniería de Requisitos. *Revista Cubana de Ciencias Informáticas*, *15*(1), 20-36.

Batra, P., Chaudhary, S., Bhatt, K., Varshney, S., y Verma, S. (2020). A Review: Abstractive Text Summarization Techniques using NLP. In *2020 International Conference on Advances in Computing, Communication & Materials (ICACCM)* (pp. 23-28). IEEE.

Behutiye, W., Rodríguez, P., Oivo, M., Aaramaa, S., Partanen, J., y Abhervé, A. (2022). Towards optimal quality requirement documentation in agile software development: A multiple case study. *Journal of Systems and Software*, *183*, 111112.

Bennaceur, A., Tun, T. T., Yu, Y., y Nuseibeh, B. (2019). Requirements engineering. *Handbook of software engineering*, 51-92.

Besrour, S., Ab Rahim, L. B., y Dominic, P. D. D. (2014, June). Assessment and evaluation of requirements elicitation techniques using analysis determination requirements framework. In *2014 International Conference on Computer and Information Sciences (ICCOINS)* (pp. 1-6). IEEE.

Bijal, D., Nikita, P., y Sanket, S. (2017). A review paper on text summarization for Indian languages. *IJSRD-International Journal for Scientific Research & Development*, *5*(07).

Binkhonain, M., y Zhao, L. (2019). A review of machine learning algorithms for identification and classification of non-functional requirements. *Expert Systems with Applications: X*, *1*, 100001.

Borra, F., Lagomarsino, M. C., Rotondo, P., y Gherardi, M. (2019). Generalization from correlated sets of patterns in the perceptron. *Journal of Physics A: Mathematical and Theoretical*, *52*(38), 384004.

Bougouin, A., Boudin, F., y Daille, B. (2013). Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)* (pp. 543-551).

Bourque, P., y Fairley, R. (2014). Guide to the Software Engineering Body of Knowledge: Version 3.0 (SWEBOK Guide). *IEEE Computer society*.

Briano, C. A. (2023). *Compilación de apuntes sobre conceptos fundamentales de la ingeniería de software: con una visión orientada con proyectos vinculados a las Ciencias Económicas*.Recuperado de http://bibliotecadigital.econ.uba.ar/download/libros/Briano_compilacion_apuntes.pdf

Brooks, F. P., y Bullet, N. S. (1987). Essence and accidents of software engineering. *IEEE computer*, *20*(4), 10-19.

Broy, M. (2018). Yesterday, today, and tomorrow: 50 years of software engineering. *IEEE Software*, *35*(5), 38-43.

Burnay, C., Jureta, I., y Faulkner, S. (2015, May). Towards a Model of Topic Relevance during requirements elicitation-Preliminary results. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)* (pp. 151-158). IEEE.

Calle Gallego, J. M., y Zapata Jaramillo, C. M. (2023). QUARE: towards a question-answering model for requirements elicitation. *Automated Software Engineering*, *30*(2), 25.

Calle, J., y Zapata, C. (2022). QUARE: Towards a Question-Answering Model for Requirements Elicitation.

Canché, M., y Pino, J. A. (2021). Requirements Elicitation for Collaborative Systems: A Systematic Review. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 297-304). IEEE.

Colburn, A., Hsieh, J., Kehrt, M., y Kimball, A. (2008). There is no software engineering crisis. *Available: cs. washington. edu courses cse503 0 wi crisis-con. pdf*.

Coughlan, J., Lycett, M., y Macredie, R. D. (2003). Communication issues in requirements elicitation: a content analysis of stakeholder experiences. *Information and Software Technology*, *45*(8), 525-537.

Chaudhary, M. N. A., Sabahat, N., y Toor, S. K. (2020). RES-TOOL to overcome requirements elicitation barriers in Pakistan software industry. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)* (pp. 1-6). IEEE.

Christel, M. G., y Kang, K. C. (1992). Issues in requirements elicitation.

Dalpiaz, F., Dell'Anna, D., Kopczyńska, S., y Montgomery, L. (2022). 5th Workshop on Natural Language Processing for Requirements Engineering (NLP4RE). In *CEUR Workshop Proceedings* (Vol. 3122). CEUR WS.

Dar, H. S. (2020). Reducing ambiguity in requirements elicitation via gamification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)* (pp. 440-444). IEEE.

Dar, H. S., Imtiaz, S., y Lali, M. I. (2022). Reducing Requirements Ambiguity via Gamification: Comparison with Traditional Techniques. *Computational Intelligence and Neuroscience*, *2022*.

Dar, H., Lali, M. I., Ashraf, H., Ramzan, M., Amjad, T., y Shahzad, B. (2018). A systematic study on software requirements elicitation techniques and its challenges in mobile application development. *IEEE Access*, *6*, 63859-63867.

de Araújo, A. F., y Marcacini, R. M. (2021). Re-bert: automatic extraction of software requirements from app reviews using bert language model. In *Proceedings of the 36th annual ACM symposium on applied computing* (pp. 1321-1327).

de los Llanos Carrión Varela, M. (2014). RESOLUCIÓN DE ANÁFORAS QUE REQUIEREN CONOCIMIENTO CULTURAL CON LA HERRAMIENTA FUNGRAMKB. *Revista de Lingüística y Lenguas Aplicadas (RLLA)*, *9*.

Dehling, H., Vogel, D., Wendler, M., y Wied, D. (2017). Testing for changes in Kendall's tau. *Econometric Theory*, *33*(6), 1352-1386.

Delgado Olivera, L. D. L. C., y Díaz Alonso, L. M. (2021). Modelos de Desarrollo de Software. *Revista Cubana de Ciencias Informáticas*, *15*(1), 37-51.

Deshpande, G., Arora, C., y Ruhe, G. (2019). Data-driven elicitation and optimization of dependencies between requirements. In *2019 IEEE 27th International Requirements Engineering Conference (RE)* (pp. 416-421). IEEE.

Devlin, J., Chang, M. W., Lee, K., y Toutanova, K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)* (pp. 4171-4186).

Dhruv, A. J., Patel, R., y Doshi, N. (2021). Python: the most advanced programming language for computer science applications. *Science and Technology Publications, Lda*, 292-299.

Dias Canedo, E., y Cordeiro Mendes, B. (2020). Software requirements classification using machine learning algorithms. *Entropy*, *22*(9), 1057.

Ding, J., Ren, X., y Luo, R. (2023, octubre). An adaptive learning method for solving the extreme learning rate problem of transformer. In *CCF International Conference on Natural Language Processing and Chinese Computing* (pp. 361-372). Cham: Springer Nature Switzerland.

Dutta, M., Das, A. K., Mallick, C., Sarkar, A., y Das, A. K. (2018). A graph based approach on extractive summarization. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 2* (pp. 179-187). Singapore: Springer Singapore.

Eckhardt, J., Vogelsang, A., Femmer, H., y Mager, P. (2016). Challenging incompleteness of performance requirements by sentence patterns. In *2016 IEEE 24th International Requirements Engineering Conference (RE)* (pp. 46-55). IEEE.

Elbarougy, R., Behery, G., y El Khatib, A. (2020). Extractive Arabic text summarization using modified PageRank algorithm. *Egyptian informatics journal*, *21*(2), 73-81.

El-Kassas, W. S., Salama, C. R., Rafea, A. A., y Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert systems with applications*, *165*, 113679.

Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M., y Briand, L. C. (2021, May). Using domain-specific corpora for improved handling of ambiguity in requirements. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 1485-1497). IEEE.

Femmer, H., Hauptmann, B., Eder, S., y Moser, D. (2016). Quality assurance of requirements artifacts in practice: A case study and a process proposal. In *Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings 17* (pp. 506-516). Springer International Publishing.

Ferrando, A., Gatti, A., y Mascardi, V. (2023). Rv4rasa: A Formalism-Agnostic Runtime Verification Framework For Verifying Chatbots In Rasa. In *Proceedings of the 6th International Workshop on Verification and Monitoring at Runtime Execution* (pp. 1-8).

Ferrari, A., y Esuli, A. (2019). An NLP approach for cross-domain ambiguity detection in requirements engineering. *Automated Software Engineering*, *26*(3), 559-598.

Ferrari, A., Zhao, L., y Alhoshan, W. (2021). NLP for requirements engineering: tasks, techniques, tools, and technologies. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (pp. 322-323). IEEE.

Ferreira Martins, H., Carvalho de Oliveira Junior, A., Dias Canedo, E., Dias Kosloski, R. A., Ávila Paldês, R., y Costa Oliveira, E. (2019). Design thinking: Challenges for software requirements elicitation. *Information*, *10*(12), 371.

Frattini, J., Unterkalmsteiner, M., Fucci, D., y Mendez, D. (2024). NLP4RE Tools: Classification, Overview, and Management. *arXiv preprint arXiv:2403.06685*.

Freire, D. L., de Almeida, A. M., Dias, M. D. S., Rivolli, A., Pereira, F. S., de Godoi, G. A., y de Carvalho, A. C. (2024). LegalSum: Towards Tool for Evaluation for Extractive Summarization of Brazilian Lawsuits. In *International Conference on Information Technology & Systems* (pp. 258-267). Cham: Springer Nature Switzerland.

Gambhir, M., y Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, *47*(1), 1-66.

Gervasi, V., y Zowghi, D. (2005). Reasoning about inconsistencies in natural language requirements. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *14*(3), 277-330.

Giarelis, N., Mastrokostas, C., y Karacapilidis, N. (2023). Abstractive vs. extractive summarization: An experimental review. *Applied Sciences*, *13*(13), 7620.

Giró, J. F., Disderi, J., y Zarazaga, B. (2016). Las causas de las deficiencias de la Ingeniería de Software. *Ciencia y tecnología*, (16), 69-80.

Görer, B., y Aydemir, F. B. (2024). Exploring the REIT architecture for requirements elicitation interview training with robotic and virtual tutors. *Journal of Systems and Software*, *212*, 112018.

Gudivada, V. N. (2018). Natural language core tasks and applications. In *Handbook of statistics* (Vol. 38, pp. 403-428). Elsevier.

Guelfi, N. (2022). The MESSIR Flexible Scientific Approach to Requirements Engineering. *Software*, *1*(1), 80-106.

Gunda, S. G. (2008). Requirements engineering: elicitation techniques.

Gupta, H., y Patel, M. (2021). Method of text summarization using LSA and sentence based topic modelling with Bert. In *2021 international conference on artificial intelligence and smart systems (ICAIS)* (pp. 511-517). IEEE.

Hamid, M., Ahmad, A., y Aimeur, E. (2019). Factors contributing in failures of software projects. *International Journal of Computer Science and Network Security*, *19*(5), 62-77.

Haque, M. A., Rahman, M. A., y Siddik, M. S. (2019). Non-functional requirements classification with feature extraction and machine learning: An empirical study. In *2019 1st international conference on advances in science, engineering and robotics technology (ICASERT)* (pp. 1-5). IEEE.

Hassan, F. U., y Le, T. (2020). Automated requirements identification from construction contract documents using natural language processing. *Journal of Legal Affairs and Dispute Resolution in Engineering and Construction*, *12*(2), 04520009.

Hayes, J., Antoniol, G. y Guéhéneuc, Y. (2008). Prereqir: Recovering pre-requirements via cluster analysis. In *Proceedings of the 15th Working Conference on Reverse Engineering.* (pp. 165-174). IEEE.

Henriksson, A., y Zdravkovic, J. (2022). Holistic data-driven requirements elicitation in the big data era. *Software and Systems Modeling*, *21*(4), 1389-1410.

Hey, T., Keim, J., Koziolek, A., y Tichy, W. F. (2020). Norbert: Transfer learning for requirements classification. In *2020 IEEE 28th International Requirements Engineering Conference (RE)* (pp. 169-179). IEEE.

Holste, J. S., y Fields, D. (2010). Trust and tacit knowledge sharing and use. *Journal of knowledge management*, *14*(1), 128-140.

Hughes, D. L., Rana, N. P., y Simintiras, A. C. (2017). The changing landscape of IS project failure: an examination of the key factors. *Journal of Enterprise Information Management*, *30*(1), 142-165.

Hussain, A., Mkpojiogu, E. O., y Kamal, F. M. (2016). The role of requirements in the success or failure of software projects. *International Review of Management and Marketing*, *6*(7), 306-311.

Ibrahim, M., y Ahmad, R. (2010, May). Class diagram extraction from textual requirements using natural language processing (NLP) techniques. In *2010 Second International Conference on Computer Research and Development* (pp. 200-204). IEEE.

IEEE Standards Committee. (1990). IEEE standard glossary of software engineering terminology. *IEEE Std*, *610*, 12.

ISO/IEC/IEEE 29148. (2018) Systems and software engineering- Life cycle processes – Requirements engineering. https://www.iso.org/standard/72089.html

Jafar, R., Almufareh, M. F., Ashraf, S., Khan, B., Butt, W. H., y Humayun, M. (2023). Towards Modeling Functional Requirements From Tacit Knowledge. In *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)* (pp. 708-713). IEEE.

Jalolov, T. (2024). FRONTEND AND BACKEND DEVELOPER DIFFERENCE AND ADVANTAGES. *Multidisciplinary Journal of Science and Technology*, *4*(2), 178-179.

Jha, A., Temkar, S., Hegde, P., y Singhaniya, N. (2022). Business meeting summary generation using nlp. In *ITM Web of Conferences* (Vol. 44, p. 03063). EDP Sciences.

Jiang, H., Nazar, N., Zhang, J., Zhang, T., y Ren, Z. (2017). Prst: A pagerank-based summarization technique for summarizing bug reports with duplicates. *International Journal of Software Engineering and Knowledge Engineering*, *27*(06), 869-896.

Jusoh, S. (2018). A study on NLP applications and ambiguity problems. *Journal of Theoretical & Applied Information Technology*, *96*(6).

Kamsties, E. (2005). Understanding ambiguity in requirements engineering. *Engineering and Managing Software Requirements*, 245-266.

Kandel, B. (2020). Qualitative Versus Quantitative Research. *Journal of Product Innovation Management*, *32*(5), 658.

Kang, Y., Cai, Z., Tan, C. W., Huang, Q., y Liu, H. (2020). Natural language processing (NLP) in management research: A literature review. *Journal of Management Analytics*, *7*(2), 139-172.

Kaur, K., y Kaur, P. (2023). BERT-CNN: improving BERT for requirements classification using CNN. *Procedia Computer Science*, *218*, 2604-2611.

Kaur, K., y Kaur, P. (2024). The application of AI techniques in requirements classification: a systematic mapping. *Artificial Intelligence Review*, *57*(3), 57.

Kengphanphanit, N., y Muenchaisri, P. (2020). Automatic requirements elicitation from social media (ARESM). In *Proceedings of the 2020 International Conference on Computer Communication and Information Systems* (pp. 57-62).

Kenton, J. D. M. W. C., y Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT* (Vol. 1, p. 2).

Khurana, D., Koli, A., Khatter, K., y Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, *82*(3), 3713-3744.

Kobayashi, S., Akram, Y., y Von Oswald, J. (2024). Weight decay induces low-rank attention layers. *Advances in Neural Information Processing Systems*, *37*, 4481-4510.

Kotowaroo, M. Y., y Sungkur, R. K. (2022). Success and Failure Factors Affecting Software Development Projects from IT Professionals' Perspective. In *Soft Computing for Security Applications: Proceedings of ICSCS 2022* (pp. 757-772). Singapore: Springer Nature Singapore.

Krzeszewska, U., Poniszewska-Marańda, A., y Ochelska-Mierzejewska, J. (2022). Systematic comparison of vectorization methods in classification context. *Applied Sciences*, *12*(10), 5119.

Kumari, N. S., y Pillai, A. S. (2013). A study on the software requirements elicitation issues-its causes and effects. In *Proc. 3rd World Congr. Inf. Commun. Technol. (WICT)* (pp. 245-252).

Kuznietsov, O., y Kyselov, G. (2024). An overview of current issues in automatic text summarization of natural language using artificial intelligence methods. *Technology audit and production reserves*.

Lafi, M., Hawashin, B., y AlZu'bi, S. (2021). Eliciting requirements from stakeholders' responses using natural language processing. *Computer Modeling in Engineering & Sciences*, *127*(1), 99-116.

Lam, Z., y Peña, M. R. (2023). Los Factores de Fracaso y Éxito en un proyecto de desarrollo de Software. *Revista peruana de computación y sistemas*, *5*(1), 3-12.

Lamsweerde, A. V. (2009). *Requirements engineering: from system goals to UML models to software specifications*. John Wiley & Sons, Ltd.

Laplante, P. A., y Kassab, M. (2022). *Requirements engineering for software and systems*. Auerbach Publications.

Ledeneva, Y. N., y García Hernández, R. A. (2013). *Automatic text summarization with maximal frequent sequences*. Universidad Autónoma del Estado de México.

Lee, R. S. (2023). *Natural Language Processing: A Textbook with Python Implementation*. Springer Nature.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... y Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Liu, Y., y Lapata, M. (2019). Text summarization with pretrained encoders. *arXiv preprint arXiv:1908.08345*.

Luo, X., Xue, Y., Xing, Z., y Sun, J. (2022). Prcbert: Prompt learning for requirement classification using bert-based pretrained language models. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering* (pp. 1-13).

Luque García, I. (2023). *Minería de texto mediante NLP en el sector seguros* (Bachelor's thesis, Universitat Politècnica de Catalunya).

Maatuk, M. A., y Abdelnabi, A. E. (2021, April). Generating UML use case and activity diagrams using NLP techniques and heuristics rules. In *International Conference on Data Science, E-learning and Information Systems 2021* (pp. 271-277).

Mahapatra, J., y Garain, U. (2024). Impact of model size on fine-tuned llm performance in data-to-text generation: A state-of-the-art investigation. *arXiv preprint arXiv:2407.14088*.

Mandal, S., y Singh, G. K. (2020). LSA based text summarization. *Int J Recent Technol Eng*, *9*, 150-156.

Manojkumar, V. K., Mathi, S., y Gao, X. Z. (2023). An experimental investigation on unsupervised text summarization for customer reviews. *Procedia Computer Science*, *218*, 1692-1701.

Massey, A. K., Rutledge, R. L., Antón, A. I., y Swire, P. P. (2014). Identifying and classifying ambiguity for regulatory requirements. In *2014 IEEE 22nd international requirements engineering conference (RE)* (pp. 83-92). IEEE.

Matias Mendoza, G. A., Ledeneva, Y., y García Hernández, R. A. (2020). Deteccion de ideas principales y composicion de resumenes en inglés, espanol, portugués y ruso. 60 anos de investigación.

Mauger, C., Schwartz, T., Dantan, J. Y., y Harbouche, L. (2010, December). Improving users satisfaction by using requirements engineering approaches in the conceptual phase of construction projects: The elicitation process. In *2010 IEEE International Conference on Industrial Engineering and Engineering Management* (pp. 310-314). IEEE.

Maurya, H. C., Gupta, P., y Choudhary, N. (2015). Natural language ambiguity and its effect on machine learning. *International Journal Of Modern Engineering Research*, *5*, 25-30.

Medeiros, J., Vasconcelos, A., Silva, C., y Goulão, M. (2018). Quality of software requirements specification in agile projects: A cross-case analysis of six companies. *Journal of Systems and Software*, *142*, 171-194.

Memon, K. A., y Xiaoling, X. (2019, April). Deciphering and analyzing software requirements employing the techniques of natural language processing. In *Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence* (pp. 153-156).

Menezes, T. (2023). A review to find elicitation methods for business process automation software. *Software*, *2*(2), 177-196.

Mirjalili, V., y Raschka, S. (2020). *Python machine learning*. Marcombo.

More, P., y Phalnikar, R. (2012). Generating UML diagrams from natural language specifications. *International Journal of Applied Information Systems*, *1*(8), 19-23.

Moreira, D., Cruz, I., Gonzalez, K., Quirumbay, A., Magallan, C., Guarda, T., ... y Castillo, C. (2021). Análisis del Estado Actual de Procesamiento de Lenguaje Natural. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E42), 126-136.

Mridha, M. F., Lima, A. A., Nur, K., Das, S. C., Hasan, M., y Kabir, M. M. (2021). A survey of automatic text summarization: Progress, process and challenges. *IEEE Access*, *9*, 156043-156070.

Muniraj, P., Sabarmathi, K. R., y Leelavathi, R. (2023). HNTSumm: Hybrid text summarization of transliterated news articles. *International Journal of Intelligent Networks*, *4*, 53-61.

Nakaishi, K., Nishikawa, Y., y Hukushima, K. (2024). Critical Phase Transition in Large Language Models. *arXiv preprint arXiv:2406.05335*.

Nizam, A. (2022). Software project failure process definition. *IEEE access*, *10*, 34428-34441.

Nyabuto, G. (2024). Client-server architecture, a review. *International Journal of Advanced Science and Computer Applications*, *3*(2).

Okesola, O. J., Okokpujie, K., Goddy-Worlu, R., Ogunbanwo, A., y Iheanetu, O. (2019). Qualitative comparisons of elicitation techniques in requirement engineering. *ARPN J. Eng. Appl. Sci*, *14*(2), 565-570.

Orquín, A. F. (2009). *Sistema para el pre-procesamiento de textos para el Procesamiento del Lenguaje Natural. Text Pre-processing System for Natural Language Processing* (Doctoral dissertation, Universidad de Matanzas).

Pacheco, C., Garcia, I., Calvo-Manzano, J. A., y Reyes, M. (2023). Measuring and improving software requirements elicitation in a small-sized software organization: a lightweight implementation of ISO/IEC/IEEE 15939: 2017—systems and software engineering—measurement process. *Requirements Engineering*, *28*(2), 257-281.

Pacheco, C., García, I., y Reyes, M. (2018). Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. *IET Software*, *12*(4), 365-378.

Paetsch, F., Eberlein, A., y Maurer, F. (2003). Requirements engineering and agile software development. In *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* (pp. 308-313). IEEE.

Pandey, V., Shukla, A., y Pandey, D. (2013). Requirement Engineering Issues.

Panichella, S., y Ruiz, M. (2020). Requirements-collector: Automating requirements specification from elicitation sessions and user feedback. In *2020 IEEE 28th international requirements engineering conference (RE)* (pp. 404-407). IEEE.

Pantaleo, G., y Rinaudo, L. (2015). *Ingeniería de software*. Alpha Editorial.

Patel, S. M., Dabhi, V. K., y Prajapati, H. B. (2017). Extractive Based Automatic Text Summarization. *J. Comput.*, *12*(6), 550-563.

Pati, S. P., y Rautray, R. (2024). Sentence Selection for Extractive Text Summarization using TOPSIS Approach. *Procedia Computer Science*, *235*, 1532-1538.

Peeperkorn, M., Kouwenhoven, T., Brown, D., y Jordanous, A. (2024). Is temperature the creativity parameter of large language models? *arXiv preprint arXiv:2405.00492*.

Popoola, O. A., Adama, H. E., Okeke, C. D., y Akinoso, A. E. (2024). Advancements and innovations in requirements elicitation: Developing a comprehensive conceptual model. *World Journal of Advanced Research and Reviews*, *22*(1), 1209-1220.

Pressman, R., y Maxim, B. (2019). *ISE software engineering: A practitioner's approach*. McGraw-Hill Education.

Prudhvi, K., Bharath Chowdary, A., Subba Rami Reddy, P., y Lakshmi Prasanna, P. (2020). Text summarization using natural language processing. In *Intelligent System Design: Proceedings of Intelligent System Design: INDIA 2019* (pp. 535-547). Singapore: Springer Singapore.

Puspitaningrum, A. C., y Sintiya, E. S. (2022). Teknik Elisitasi Kebutuhan Perangkat Lunak: Literatur Review. *JUSIFO (Jurnal Sistem Informasi)*, *8*(1), 35-42.

Qin, L., Chen, Q., Feng, X., Wu, Y., Zhang, Y., Li, Y., ... y Yu, P. S. (2024). Large language models meet nlp: A survey. *arXiv preprint arXiv:2405.12819*.

Quba, G. Y., Al Qaisi, H., Althunibat, A., y AlZu'bi, S. (2021). Software requirements classification using machine learning algorithm's. In *2021 International Conference on Information Technology (ICIT)* (pp. 685-690). IEEE.

Rahimi, N., Eassa, F., y Elrefaei, L. (2021). One-and two-phase software requirement classification using ensemble deep learning. *Entropy*, *23*(10), 1264.

Ravichandar, R., Arthur, J. y Pérez-Quiñones, M. (2007). Pre-requirement specification traceability: Bridging the complexity gap through capabilities. *arXiv preprint cs/0703012.*

Raza, A., Raja, H. S., y Maratib, U. (2023). Abstractive Summary Generation for the Urdu Language. *arXiv preprint arXiv:2305.16195*.

Ribeiro, C., y Berry, D. (2020). The prevalence and severity of persistent ambiguity in software requirements specifications: Is a special effort needed to find them? *Science of Computer Programming*, *195*, 102472.

Rico, S. (2024, abril). Insights Towards Better Case Study Reporting in Software Engineering. In *Proceedings of the 1st IEEE/ACM International Workshop on Methodological Issues with Empirical Studies in Software Engineering* (pp. 76-79).

Rodríguez-Silva, C. A. (2022). Guía para la mejora en el levantamiento de requerimientos de software en una pyme de logística de transporte.

Ronanki, K., Berger, C., y Horkoff, J. (2023). Investigating ChatGPT's Potential to Assist in Requirements Elicitation Processes. In *2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (pp. 354-361). IEEE.

Runeson, P., Host, M., Rainer, A., y Regnell, B. (2012). *Case study research in software engineering: Guidelines and examples*. John Wiley & Sons.

Runeson, P., y Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, *14*(2), 131-164.

Sabriye, A. O. J. A., y Zainon, W. M. N. W. (2017). A framework for detecting ambiguity in software requirement specification. In *2017 8th International Conference on Information Technology (ICIT)* (pp. 209-213). IEEE.

Salazar, O. A., Aguirre, F. A. M., y Osorio, J. A. C. (2011). Herramientas para el desarrollo rápido de aplicaciones web. *Scientia et technica*, *1*(47), 254-258.

Sampada, G. C., Sake, T. I., y Chhabra, M. (2020, November). A review on advanced techniques of requirement elicitation and specification in software development stages. In *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)* (pp. 215-220). IEEE.

Sandhu, G., y Sikka, S. (2015). State-of-art practices to detect inconsistencies and ambiguities from software requirements. In *International Conference on Computing, Communication & Automation* (pp. 812-817). IEEE.

Sandhu, R. K., y Weistroffer, H. R. (2018). A review of fundamental tasks in requirements elicitation. *EuroSymposium on Systems Analysis and Design*, 31-44.

Sarkar, D. (2016). *Text analytics with python* (Vol. 2). New York, NY, USA: Apress.

Sawyer, P., y Kotonya, G. (2001). Software requirements. *SWEBOK*, 9.

Schmidt, R. F. (2013). *Software engineering: architecture-driven software development*. Newnes.

Schulhoff, S., Ilie, M., Balepur, N., Kahadze, K., Liu, A., Si, C., ... y Resnik, P. (2024). The prompt report: a systematic survey of prompt engineering techniques. *arXiv preprint arXiv:2406.06608*.

Šenkýr, D., y Kroha, P. (2023). Quality Measurement of Functional Requirements.

Šenkýr, D., y Kroha, P. (2019). Problem of incompleteness in textual requirements specification. In *Proceedings of the 14th international conference on software technologies* (Vol. 1, pp. 323-330).

Sharma, R., y Biswas, K. K. (2012). Handling Inconsistency in Software Requirements. In *ENASE* (pp. 95-104).

Sharma, S., y Pandey, S. K. (2014). Requirements elicitation: Issues and challenges. In *2014 international conference on computing for sustainable global development (indiacom)* (pp. 151-155). IEEE.

Sharma, T., Ashri, A., Kumar, N., y Pal, S. (2021). Evaluation of Python Text Summarization Libraries. *INTERNATIONAL JOURNAL OF ENGINEERING DEVELOPMENT AND RESEARCH*, *9*(1), 159-164.

Shihab, E., Wagner, S., Gerosa, M. A., Wessel, M., y Cabot, J. (2022). The present and future of bots in software engineering. *IEEE Software*, *39*(5), 28-31.

Shreda, Q. A., y Hanani, A. A. (2021). Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. *IEEE Access*.

Siahaan, D., Raharjana, I. K., y Fatichah, C. (2023). User story extraction from natural language for requirements elicitation: Identify software-related information from online news. *Information and Software Technology*, *158*, 107195.

Sliusarenko, T., y Pohurska, M. (2024). Stop overfitting with proven techniques. *Grail of Science*, *40*, 373-375.

Sommerville, I. (2010). *Software Engineering: United States Edition*. Pearson.

Sommerville, I. (2019). *Engineering software products: An introduction to modern software engineering*. Pearson.

Sommerville, I., y Velázquez, S. F. (2011). *Ingeniería de software*. Pearson.

Sonbol, R., Rebdawi, G., y Ghneim, N. (2022). The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review. *Ieee Access*, *10*, 62811-62830.

Spijkman, T., de Bondt, X., Dalpiaz, F., y Brinkkemper, S. (2023). Summarization of Elicitation Conversations to Locate Requirements-Relevant Information. In *International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 122-139). Cham: Springer Nature Switzerland.

Standish Group. (2020). CHAOS Report 2020. Recuperado de https://www.standishgroup.com

Stanik, C., Pietz, T., y Maalej, W. (2021). Unsupervised topic discovery in user comments. In *2021 IEEE 29th International Requirements Engineering Conference (RE)* (pp. 150-161). IEEE.

Steinberger, J., y Ježek, K. (2009). Evaluation measures for text summarization. *Computing and Informatics*, *28*(2), 251-275.

Sun, Z., Ampornpunt, N., Varma, M., y Vishwanathan, S. (2010). Multiple kernel learning and the SMO algorithm. *Advances in neural information processing systems*, *23*.

Suppiah, V., y Singh Sandhu, M. (2011). Organisational culture's influence on tacit knowledge-sharing behaviour. *Journal of knowledge management*, *15*(3), 462-477.

Takeshita, S., Ponzetto, S. P., y Eckert, K. (2024). ROUGE-K: Do your summaries have keywords?. *arXiv preprint arXiv:2403.05186*.

Talele, P., y Phalnikar, R. (2021). Software requirements classification and prioritisation using machine learning. In *Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020* (pp. 257-267). Springer Singapore.

Talele, P., y Phalnikar, R. (2021, enero). Classification and prioritisation of software requirements using machine learning–a systematic review. In *2021 11th international conference on cloud computing, data science & engineering (confluence)* (pp. 912-918). IEEE.

Tariq, A., Azam, F., Anwar, M. W., Maqbool, B., y Javaid, H. A. (2019). A UML Profile for Prediction of Significant Software Requirements. In *2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 0979-0984). IEEE.

Thushara, M. G., Mownika, T., y Mangamuru, R. (2019). A comparative study on different keyword extraction algorithms. In *2019 3rd international conference on computing methodologies and communication (ICCMC)* (pp. 969-973). IEEE.

Tiwari, S., y Rathore, S. S. (2017). A methodology for the selection of requirement elicitation techniques. *arXiv preprint arXiv:1709.08481*.

Tsai, B. H., Fan, Y. C., y Leu, F. Y. (2021). Extractive summarization by rouge score regression based on bert. In *Complex, Intelligent and Software Intensive Systems: Proceedings of the 14th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS-2020)* (pp. 156-165). Springer International Publishing.

Umar, M. A., y Lano, K. (2024). Advances in automated support for requirements engineering: a systematic literature review. *Requirements Engineering*, 1-31.

Umber, A., Bajwa, I. S., y Asif Naeem, M. (2011). NL-based automated software requirements elicitation and specification. In *Advances in Computing and Communications: First International Conference, ACC 2011, Kochi, India, July 22-24, 2011. Proceedings, Part II 1* (pp. 30-39). Springer Berlin Heidelberg.

Van Engelen, J. E., y Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine learning*, *109*(2), 373-440.

Van Zachary, V., Trillo, J. C., Abalorio, C. C., Bustillo, J. C. M., Bojocan, J. T., y Elape, M. C. (2023). OCR-based hybrid image text summarizer using Luhn algorithm with FinetuneTransformer modelsfor long document. *International Journal of Emerging Technology and Advanced Engineering*, *13*(2), 47–56. https://doi.org/10.46338/ijetae0223_07

Verma, P., Pal, S., y Om, H. (2019). A comparative analysis on Hindi and English extractive text summarization. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, *18*(3), 1-39.

Westfall, L. (2005). Software requirements engineering what, why, who, when, and how. *Software Quality Professional*, *7*(4), 17.

Widyassari, A. P., Rustad, S., Shidik, G. F., Noersasongko, E., Syukur, A., y Affandy, A. (2022). Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences*, *34*(4), 1029-1046.

Wiegers, K. E., y Beatty, J. (2013). *Software requirements*. Pearson Education.

Wohlin C., Höst M., y Henningsson K. (2003). Empirical Research Methods in Software Engineering. In: Conradi R., Wang A.I. (Eds), *Empirical Methods and Studies in Software Engineering* (pp. 7-23). Springer, Berlin, Heidelberg.

Wohlin, C. (2021). Case Study Research in Software Engineering—It is a Case, and it is a Study, but is it a Case Study?. *Information and Software Technology*, *133*, 106514.

Wolfinger, R., Fotrousi, F., y Maalej, W. (2022). A chatbot for the elicitation of contextual information from user feedback. In *2022 IEEE 30th International Requirements Engineering Conference (RE)* (pp. 272-273). IEEE.

Wong, L. R., Mauricio, D. S., y Rodriguez, G. D. (2017). A systematic literature review about software requirements elicitation. *Journal of Engineering Science and Technology*, *12*(2), 296-317.

Wong, L. R., y Mauricio, D. S. (2019). Qualities that the activities of the elicitation process must meet to obtain a good requirement. *Journal of Engineering Science and Technology*, *14*(5), 2883-2912.

Xu, S. (2017, junio). Empirical research methods for software engineering: Keynote address. In *2017 IEEE 15th International Conference on Software Engineering Research, Management and Applications (SERA)* (pp. 1-1). IEEE.

Yadav, A. K., Maurya, A. K., y Yadav, R. S. (2021). Extractive Text Summarization Using Recent Approaches: A Survey. *Ingénierie des Systèmes d'Information*, *26*(1).

Yang, X., Yang, K., Cui, T., Chen, M., y He, L. (2022). A Study of Text Vectorization Method Combining Topic Model and Transfer Learning. *Processes*, *10*(2), 350.

Zahid, A. H., Liaqat, A., Farooq, M. S., y Naseer, S. (2020). Requirement elicitation issues and challenges in pakistan software industry. *VFAST Transactions on Software Engineering*, *8*(1), 28-36.

Zhang, H., Liu, X., y Zhang, J. (2023). Extractive summarization via chatgpt for faithful summary generation. *arXiv preprint arXiv:2304.04193*.

Zhang, H., Sproat, R., Ng, A. H., Stahlberg, F., Peng, X., Gorman, K., y Roark, B. (2019). Neural models of text normalization for speech applications. *Computational Linguistics*, *45*(2), 293-337.

Zhang, L., Tian, J. H., Jiang, J., Liu, Y. J., Pu, M. Y., y Yue, T. (2018). Empirical research in software engineering—a literature survey. *Journal of Computer Science and Technology*, *33*(5), 876-899.

Zhao, L., Alhoshan, W., Ferrari, A., Letsholo, K. J., Ajagbe, M. A., Chioasca, E. V., y Batista-Navarro, R. T. (2021). Natural language processing for requirements engineering: A systematic mapping study. *ACM Computing Surveys (CSUR)*, *54*(3), 1-41.

Zhao, L., Alhoshan, W., Ferrari, A., y Letsholo, K. J. (2022). Classification of natural language processing techniques for requirements engineering. *arXiv preprint arXiv:2204.04282*.

Zowghi, D., y Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. *Engineering and managing software requirements*, 19-46.

Zubcoff, J., Garrigós, I., Casteleyn, S., Mazón, J. N., Aguilar, J. A., y Gomariz-Castillo, F. (2019). Evaluating different i*-based approaches for selecting functional requirements while balancing and optimizing non-functional requirements: A controlled experiment. *Information and Software Technology*, *106*, 68-84.

7. Anexo A.- Acrónimos

ML	Aprendizaje Automático
ANN	Redes Neuronales Artificiales
API	Interfaz de Programación de Aplicaciones
BERT	Representaciones Bidireccionales de Codificadores a partir de Transformadores
BoW	Bolsa de Palabras
CDD	Desarrollo Basado en Conversación
CNN	Red Neuronal Convolucional
CoT	Cadena de pensamiento - *Chain of Thought*
DL	Aprendizaje Profundo
FR	Requisito Funcional
FRs	Requisitos Funcionales
GuMa	Algoritmo de búsqueda
HDBSCAN	Algoritmo para agrupar e identificar clústeres.
IA	Inteligencia Artificial
IR	Ingeniería de Requisitos
IS	Ingeniería de Software
k-NN	Vecino Más Cercano
LCS	Subsecuencia Común más Larga
LLM	Modelo de Lenguaje Largo
LR	Regresión Logística
LSA	Análisis Semántico Latente
LSP	Patrones Léxico-Sintácticos
MLP	Perceptrón Multicapa
NB	Bayes Ingenuo
NFR	Requisito No Funcional
NFRs	Requisitos No Funcionales
NL	Lenguaje Natural

NLG	Generación del Lenguaje Natural
NLI	Inferencia en Lenguaje Natural
NLP	Procesamiento de Lenguaje Natural
NLP4RE	Procesamiento del Lenguaje Natural para la Ingeniería de Requisitos
NLTK	Natural Language Toolkit
NLU	Comprensión del Lenguaje Natural
NoRBERT	Clasificador de requisitos funcionales y no funcionales basada en BERT
OTAN	Organización del Tratado del Atlántico Norte
POS	Etiquetado de Parte del Discurso
Q&A	Preguntas y Respuestas
QUARE	Modelo basado en Preguntas y Respuestas para la Elicitación de Requisitos
RE-BERT	Variante de BERT basado en lingüísticas dependientes del contexto
REConSum	Denominada Resumen de Conversaciones de Elicitación de Requisitos
ReUS	Reglas lingüísticas compuestas por patrones de clases gramaticales
ROUGE	Métrica de evaluación de resúmenes
ROUGE-1	Evaluación sobre unigrama
ROUGE-L	Evaluación en basada en LCS (Subsecuencia Común Más Larga
ROUGE-N	Evaluación sobre n-gramas (ej. ROUGE-2 para bigramas
RSL	Revisiones Sistemáticas de Literatura
SAFE	Extracción de características basadas en patrones lingüísticos
SBERT	Variante de BERT, optimizada para generar incrustaciones
SDL	Lenguaje de especificación y descripción; usado para estados/diagramas
SRC	Clasificación de Requisitos de Software
STS	Similitud Textual Semántica
SVD	Descomposición en Valores Singulares
SVM	Máquinas de Vectores de Soporte
TF	Frecuencia de Términos
TF-IDF	Frecuencia de Términos – Frecuencia Inversa de Documentos
TP	Preprocesamiento de Texto
UMAP	Técnica para reducir datos de altas dimensiones
UML	Lenguaje Unificado de Modelado
VDM	Metodología de especificación
WSL	Aprendizaje Débilmente Supervisado
Z	Lenguaje formal Z