Universidad Tecnológica de La Habana "José Antonio Echeverría"

"Extracción automática de requisitos de software a partir de información textual no estructurada"

Autor: Amanda Hernández Carreras

Grupo: 41

Tutor(es): Dra.Cs Anaisa Hernández González

Dr. Cs Alfredo Simón Cuevas

Práctica Profesional 2

Carrera: Ing. Informática

Fecha: Julio, 2022

La Habana, Cuba

Resumen

La obtención de requisitos es una de las fases más importantes y críticas en el desarrollo de software, debido a la influencia de sus resultados en el éxito de los proyectos. El análisis documental constituye una de las técnicas más utilizadas en este proceso. La ejecución manual de este análisis se ha caracterizado por el alto consumo de tiempo y la frecuente aparición de errores, motivando el desarrollo de investigaciones enfocadas en su automatización. El procesamiento del lenguaje natural para la ingeniería de requisitos (NLP4RE) es un área de investigación y desarrollo que busca aplicar técnicas, herramientas y recursos de procesamiento del lenguaje natural (NLP) al proceso de ingeniería de requisitos (RE), para ayudar a los analistas humanos a llevar a cabo diversas tareas lingüísticas. El presente trabajo tiene como objetivo desarrollar un sistema para la extracción automática de requisitos de software, a partir de información textual no estructurada. Como resultado se obtuvieron sentencias de requisitos de software que fueron evaluadas por métricas de calidad, dichos resultados no fueron muy altos, resaltando el valor de la cobertura que, en comparación a la precisión, fue relevante.

Abstract

Requirements elicitation is one of the most important and critical phases in software development, due to the influence of its results on the success of projects. Documentary analysis is one of the most used techniques in this process. The manual execution of this analysis has been characterized by the high consumption of time and the frequent appearance of errors, motivating the development of research focused on its automation. Natural Language Processing for Requirements Engineering (NLP4RE) is an area of research and development that seeks to apply natural language processing (NLP) techniques, tools, and resources to the requirements engineering (RE) process, to help human analysts to carry out various linguistic tasks. The objective of this work is to develop a system for the automatic extraction of software requirements, from unstructured textual information. As a result, software requirements sentences were obtained that were evaluated by quality metrics, these results were not very high, highlighting the value of coverage that, compared to precision, was relevant.

Índice

Índice de Tablas

Índice de Ilustraciones

Introducción

La industria del software presenta índices de éxito en los proyectos relativamente bajos si se tienen en cuenta estudios e investigaciones realizados por Standish Group's en los reportes Chaos, los cuales analizan el comportamiento de proyectos desde el año 1994. Según estos estudios aproximadamente el 50% de los expertos encuestados considera que el comportamiento de fallo en los proyectos es idéntico al de hace 10 años, el 48% considera que existen más fallos en los proyectos hoy en día y solamente el 2% considera que ha existido una mejoría en este aspecto. Los datos estadísticos reflejan que en el año 2018 solo el 16.2% de los proyectos de software fueron completados cumpliendo con el tiempo, el presupuesto y con al menos la mitad de los requisitos planteados inicialmente. Estos datos son incluso peores teniendo en cuenta los proyectos llevados a cabo por grandes empresas donde estos datos pueden llegar al 9% [1, 2].

Las principales causas de fallo de un proyecto han sido descritas en muchos estudios entre las principales se encuentran:

- Incompleta o errónea declaración de requisitos.
- Requisitos cambiantes.
- Expectativas poco realistas.
- Poca claridad en los objetivos y metas.
- Incompetencia de la tecnología.
- Metas de tiempo poco realistas.
- Calidad del producto.
- Poca claridad de los procesos de negocio.
- Satisfacción del cliente.
- Poca comunicación entre clientes, desarrolladores y usuarios finales.
- Políticas de los stakeholders (partes interesadas).

Como se observa en los estudios el porcentaje de fallos en este proceso sigue siendo considerablemente alto y el costo de corregir un error a partir de un mal proceso de IR (Ingeniería de requisitos) puede suponer hasta 5 veces superior, en

comparación a otros errores, aspectos que motivan y fundamentan la necesidad de continuar con el estudio de esta rama con el objetivo de disminuir los porcientos de fallos esta temprana etapa del proyecto. [3, 4].

Los requisitos provienen de varias partes interesadas que tienen diferentes necesidades, funciones y responsabilidades y, como tales, son propensas a que se produzcan conflictos, como la interferencia, la interdependencia y la incoherencia [5]. Además, los requisitos generalmente se especifican en lenguajes naturales, lo que aumenta la complejidad de la ingeniería de requisitos debido a la ambigüedad inherente, la incompletitud y la inexactitud de los lenguajes naturales [6]. Estos factores hacen que las tareas de RE sean desafiantes, lentas y propensas a errores, principalmente para proyectos grandes, ya que es necesario procesar, analizar y comprender grandes volúmenes de requisitos [7].

Se han llevado a cabo muchas investigaciones sobre la automatización de diferentes tareas de ER. Los enfoques propuestos generalmente comienzan aplicando un conjunto de pasos de Procesamiento del Lenguaje Natural (NLP) que extraen información y características lingüísticas de los textos de requisitos y construyen varias representaciones basadas en NLP.

Inspirándose en la estrecha relación entre el lenguaje natural (NL) y los requisitos, desde principios de la década de 1980, los investigadores han intentado desarrollar herramientas y métodos de procesamiento del lenguaje natural (NLP) para procesar los textos de requisitos [8]. Sin embargo, durante casi tres décadas, esta investigación se vio obstaculizada por tecnologías de PNL inadecuadas, que no funcionaron lo suficientemente bien en ese momento para respaldar aplicaciones fuera de la investigación de la PNL [9]. La situación ha ido cambiando radicalmente desde finales de años 2000, enormes mejoras y avances en las tecnologías de PNL, en particular la amplia disponibilidad de herramientas y recursos de PNL [10], hizo posible que los investigadores exploraran una variedad de herramientas y recursos de PNL habilitados herramientas y métodos para tareas de ER [11] [12]. Desde entonces, la investigación en PN para ER, o NLP4RE para abreviar, ha crecido en un área de investigación activa y completa [11], que atrae a investigadores de la

comunidad de ER en general. Hoy en día, la investigación en NLP4RE está prosperando y tiene un taller anual dedicado, llamado NLP4RE [12]. Ahora existe un gran potencial para desarrollar herramientas NLP4RE efectivas que puedan servir al mundo real práctica de RE. Esto es particularmente necesario, ya que las tareas de ER todavía se realizan manualmente en la práctica industrial, como lo demuestran los datos de la encuesta NaPiRE que muestran que solo el 16% de las empresas utilizan técnicas automatizadas para el análisis de requisitos [13]. Pero la mayoría de estos trabajos son enfocados al procesamiento de texto en idioma inglés.

Por lo tanto, nuestra **situación problemática** es: Elicitación de requisitos, trabajo mayoritariamente manual que tiene sus bases en la experiencia y capacidad de especialistas en llevar a cabo las diferentes técnicas, metodologías y métodos para el análisis de la información y su transformación en requisitos de software.

Como **objeto de estudio** para dar solución a este problema se tiene: Información textual no estructurada en idioma español.

Como **campo de acción** se tiene: Procesamiento de lenguaje Natural aplicado a la Ingeniería de Requisitos.

Se define como **objetivo general** Desarrollar un sistema para la extracción automática de requisitos de software, a partir de información textual no estructurada.

Objetivos Específicos

1. Caracterizar trabajos reportados en la literatura.
    - Caracterizar trabajos reportados en la literatura afines con la captura de requisitos de software a partir de información textual no estructurada.
2. Implementar sistema para la extracción automática de requisitos de software.
    - Extender implementación de los patrones léxicos-sintácticos.
    - Concebir mejoras al proceso de reducción de redundancias de requisitos capturados.

- Implementar mejoras concebidas al proceso de reducción de redundancias.

3. Evaluar la nueva versión de solución de capturas de requisitos.
    - Evaluar la nueva versión de solución de capturas de requisitos a partir de textos identificando debilidades y definiendo soluciones.

El resto del documento está organizado de la siguiente manera: Capítulo 1 Análisis del estado del arte. Capítulo 2 Descripción la solución propuesta. Capítulo 3 Evaluación de la solución propuesta.

Capítulo 1. Fundamentos teóricos

1.1 Ingeniería de requisitos

Los requerimientos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. Al proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se le llama ingeniería de requisitos (IR). [14]

Por su parte [15] plantea que un requisito es una condición o capacidad necesitada por un usuario para resolver un problema o lograr un objetivo; una condición o capacidad que debe ser cumplida o procesada por un sistema o componente del sistema para satisfacer un contrato, estándar, especificación u otro documento formalmente impuesto; una representación documentada de una condición o capacidad como en las dos ideas previas expuestas.

Según [15] la Ingeniería de Requisitos es un enfoque disciplinado y sistemático a la especificación y gestión de requerimientos con las siguientes metas: conocer los requerimientos relevantes, logrando un conceso sobre estos entre los interesados, documentarlos de acuerdo a estándares establecidos, y gestionarlos sistemáticamente; comprender y documentar los deseos y necesidades de los interesados, tenerlos a ellos especificando y gestionando los requerimientos minimizará el riesgo de entregar un sistema que no cumpla con sus deseos o necesidades.

Otra definición, planteada por [16] es que el espectro amplio de tareas y técnicas que llevan a entender los requerimientos se denomina ingeniería de requerimientos. Desde la perspectiva del proceso del software, la ingeniería de requerimientos es una de las acciones importantes de la ingeniería de software que comienza durante la actividad de comunicación y continúa en la de modelado. Debe adaptarse a las necesidades del proceso, del proyecto, del producto y de las personas que hacen el trabajo.

1.1.1 Estándares y formalización de requisitos de software

Para la documentación de requisitos de software se puede estructurar en tres grupos fundamentales:

1. Utilizando lenguaje natural: Esta es la variante más comúnmente utilizada en la práctica, teniendo la ventaja de que los stakeholders no deberán aprender nuevas notaciones, y que el lenguaje puede utilizarse de múltiples propósitos, desde estructuras formales hasta descripciones detalladas. En cambio, los requisitos en lenguaje natural pueden ser ambiguos, y perder perspectiva si no se redactan correctamente.
2. Modelos conceptuales: A diferencia del lenguaje natural los diferentes modelos conceptuales no pueden utilizarse de forma universal. Al documentar requisitos mediante modelos se deben utilizar lenguajes de modelado específicos, que deben ser utilizados de forma correcta para poder alcanzar el objetivo deseado. Entre los modelos más utilizados se encuentran:
    - Diagrama de clases
    - Diagramas de casos de uso
    - Diagrama de actividades
    - Diagrama de estados
3. Documentos híbridos: Actualmente la variante que más fuerza va tomando. La utilización combinada de los dos puntos anteriores permite elaborar un documento enfocado a la audiencia objetivo. Brindando diferentes perspectivas del sistema. La utilización combinada del lenguaje natural y modelos conceptuales arrastra las desventajas de ambas opciones, pero se complementa por la combinación de las fortalezas y su versatilidad.

Para determinar que estructuras de requisitos utilizar, los estereotipos adecuados en cada situación existen diferentes estándares desde los métodos tradicionales

hasta las más actuales metodologías ágiles de desarrollo. Entre los estándares más utilizados se encuentran:

1. ***Rational Unified Process (RUP)*** [17]: Este proceso es usualmente utilizado al desarrollar bajo el paradigma orientado a objeto. En este se utilizan modelos para describir el negocio y la solución mediante el uso de diferentes artefactos como: Reglas del negocio, casos de uso, combinando el lenguaje natural con modelos conceptuales. La especificación de requisitos normalmente es redactada utilizando la estructura *software requirements specification (SRS),* la cual está muy relacionada al estándar ISO/IEC/IEEE 29148:2011.

   Entre los modelos que propone RUP se encuentran:

   - Modelos de casos de uso
   - Modelo de análisis
   - Modelo de diseño
   - Modelo de despliegue
   - Modelo de implementación
   - Modelo de prueba

2. **Estándar ISO/IEC/IEEE 29148:2011** [18]: Contiene normas establecidas para la documentación de requisitos de software. El estándar sugiere dividir el documento de especificación de requisitos en 5 capítulos utilizando el lenguaje natural como variante a utilizar.

   - Información introductoria
   - Referencias
   - Requisitos del sistema (funcionales, rendimiento, interfaces)
   - Medidas para la verificación
   - Apéndice (Se aclaran los elementos que se asumen, identifican las dependencias)

3. **Modelo V**: Define diferentes estructuras a partir de quien es el creador:

   - Especificación de requisitos del cliente: Elaborado por el cliente, incluye servicios, restricciones, explicación del proceso.

Generalmente se escribe para responder el **qué** se hace y **para qué** se hace.

- Especificación de requisitos del sistema: Se basa en la especificación del cliente, pero tiene detalles del diseño e implementación que elabora el contratista. Se elabora a partir del proceso y restricciones establecidas por el cliente.

4. **CMMi** [19]: Este estándar plantea como propósito del desarrollo de requisitos de software, producir y analizar los requerimientos de cliente, de producto y de componente del producto.
5. **Estándar IEEE 830**: El estándar IEEE 830-1998 está enfocado en recomendaciones prácticas para la especificación de requerimientos, fue desarrollado por la IEEE y la IEEE-SA (Standards Association), indica la estructura y organización de toda la información que debe incluirse en un buen documento de especificación de requerimientos de software. Una de las características más distinguibles del estándar IEEE 830 es que los requisitos deben ser escritos de la siguiente manera: “El sistema debe…”.
6. Metodologías ágiles e historias de usuario: Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones acerca de las necesidades de los clientes. Son **descripciones cortas y simples** de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario.

1.1.2 Gestión de requisitos con soporte computacional

Ingenieros e investigadores de la ciencia de datos utilizan el aprendizaje automático para obtener nuevas perspectivas y soluciones de diversos desafíos, que estaban hasta este momento fuera de su alcance, aprovechando la infraestructura de hardware y disponibilidad de grandes volúmenes de datos. Algunas de las actividades que están siendo respaldadas por las tecnologías de aprendizaje automático son las relacionadas al desarrollo y gestión de requisitos.

En cuanto a las actividades de educción de requisitos, Sharma y otros [20], proponen un método para identificar automáticamente los requisitos de seguridad, combinando el análisis lingüístico con técnicas de aprendizaje automático.

Bagheri y otros [21], proponen un framework de apoyo a la toma de decisiones, que emplea procesamiento de lenguaje natural y una ontología para procesar especificaciones de requisitos a modo de facilitar y brindar soporte a ingenieros y gerentes de proyectos.

Otras propuestas abordan los problemas de priorización de requisitos. McZara y otros [22], presentan un método, denominado SNIPR, para ordenar y seleccionar los requerimientos de acuerdo a las prioridades de los interesados, mediante el empleo de técnicas de procesamiento del lenguaje natural.

Duan y otros [23], proponen el uso de técnicas de minería de datos y aprendizaje automático para priorizar los requisitos de acuerdo con los intereses de las partes interesadas, los objetivos empresariales y las preocupaciones transversales, tales como los requisitos de seguridad y rendimiento. La eficacia del enfoque se evalúa a través de un caso de estudio basado en un conjunto de requisitos extraídos de un proyecto de código abierto denominado SugarCRM.

Guo [24], aplica el aprendizaje automático para predecir el rendimiento de ciertas configuraciones del sistema que puedan afectar a los requerimientos. Fitzgerald y otros [25], proponen una herramienta capaz de predecir los requerimientos que pueden fallar, dada la ambigüedad y escasez de información en la especificación de los requisitos. Sagrado y Aguila [26], proponen el uso de las redes Bayesianas para predecir cuales son las especificaciones de requisitos que deben ser revisadas.

Otras propuestas resuelven los desafíos de la IR mediante las técnicas de clasificación. Sharma y otros [20], proponen un método para detectar ambigüedad en los documentos de especificación de requisitos, mediante el uso de redes bayesianas. Águila y Sagrado [27], proponen el uso de clasificadores bayesianos para clasificar los riesgos en los requerimientos. Kurtanovic y Maalej [28], proponen

clasificar automáticamente los requisitos de software en funcionales y no funcionales, con técnicas de aprendizaje automático supervisado. Paralelamente, se evalúa la precisión con la que se puede identificar varios tipos de requerimientos no funcionales, en particular los requisitos de usabilidad, seguridad, operación y rendimiento. Dekhtyar y Fong [29], consideran la aplicación de aprendizaje automático guiado por Tensorflow usando Word2vec para resolver el problema de clasificación de requerimientos.

Las técnicas de aprendizaje automático también son aplicadas para dar solución a los problemas de trazabilidad en [30].

1.1.3 Rol del lenguaje natural en los requisitos de software

El importante papel del lenguaje natural (NL) en la ingeniería de requisitos (RE) se ha establecido hace mucho tiempo [31] [32]. En una encuesta publicada en 1981, destinada a proporcionar una visión general de las técnicas para expresando requisitos y especificaciones, Abbott y Moorhead afirmaron que “el mejor lenguaje para los requisitos es el lenguaje natural [33].” Si bien es difícil demostrar que NL es en realidad la mejor opción, la evidencia empírica a lo largo de los años ha demostrado que es al menos la notación más común para expresar requisitos en la práctica industrial. La encuesta en línea de 151 empresas de software a principios de la década de 2000 realizada por Mich et al. [34] concluyó que en el 95% por ciento de los casos los requisitos de los documentos se expresaron en alguna forma de NL. Este predominio de NL fue confirmado por un reciente encuesta de Kassab et al. [35], que involucró a 250 practicantes. La mayoría de los participantes (61%) en esa encuesta indicaron que NL se usaba normalmente en sus empresas para describir y especificar requisitos del software y del sistema. Por lo tanto, con base en la evidencia empírica pasada y actual, podemos suponer con seguridad que NL continuará sirviendo como la lengua franca para los requisitos en el futuro también. Inspirándose en la estrecha relación entre el NL y los requisitos, desde principios de

la década de 1980, los investigadores han intentado desarrollar herramientas y métodos de procesamiento del lenguaje natural (NLP) para procesar los textos de requisitos. [32]

1.1.3.1 Construcción de un requisito en lenguaje natural.

Para la construcción de un requisito en lenguaje natural se pueden plantear 5 pasos fundamentales. Con el objetivo de garantizar la calidad de los requisitos mediante el uso de una estructura definida y glosarios de términos.

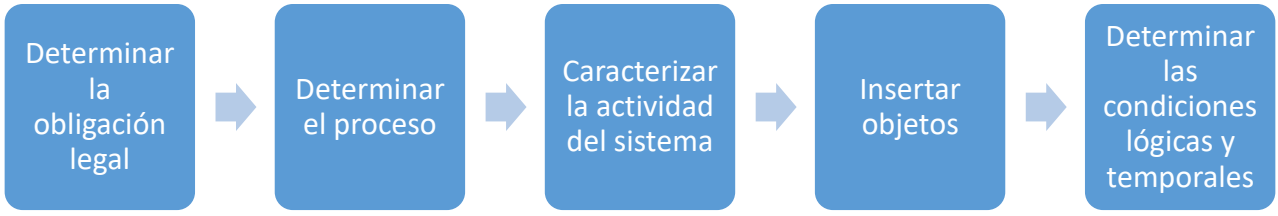

Ilustración 1 Proceso de construcción de un requisito

Obteniendo la siguiente estructura como base para la construcción del requisito en lenguaje natural:

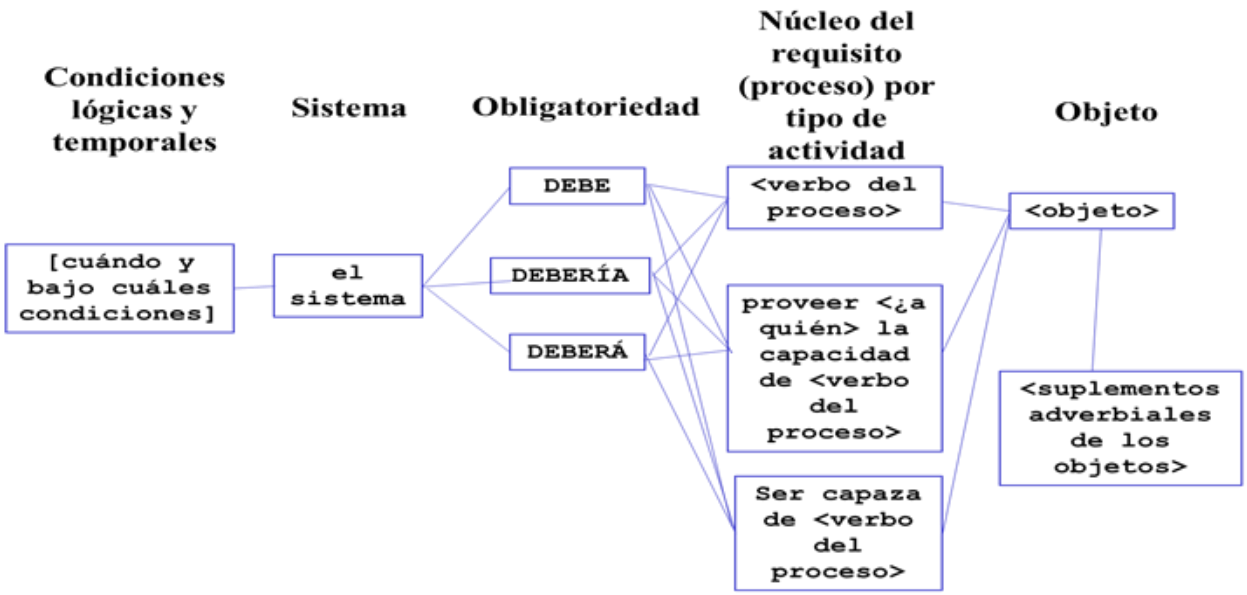

Ilustración 2 Estructura de un requisito en lenguaje natural

1.1.3.2 Procesamiento de lenguaje natural

El procesamiento del lenguaje natural es una de las principales disciplinas de la inteligencia artificial. Su objetivo es permitir que los programas informáticos "entiendan" y procesen textos en lenguaje natural para lograr algunos objetivos específicos [36]. Tradicionalmente, existen tres niveles principales de procesamiento en un enfoque basado en PNL [37]:

- Nivel léxico y morfológico: El nivel morfológico se enfoca en analizar palabras en sus morfemas como prefijos, sufijos, y palabras base. Incluye tareas comunes, como tokenización (proceso de dividir un texto en una lista de tokens) y lematización (proceso de encontrar la forma de diccionario, o el lema de cada palabra).
- Nivel sintáctico: El nivel sintáctico se enfoca en analizar la estructura gramatical de las oraciones. Este nivel generalmente incluye etiquetado de parte del discurso (etiquetado de POS: proceso de etiquetar cada token en una oración con su parte correspondiente de una etiqueta de voz basada en su contexto sintáctico), fragmentación (proceso de detección de constituyentes sintácticos como frases nominales y frases verbales en una oración), análisis de dependencia (proceso de analizar la estructura sintáctica de la oración, descubriendo las palabras relacionadas gramaticalmente, así como el tipo de relación entre ellas) y reconocimiento de entidad nombrada (localizar y clasificar las entidades nombradas mencionadas en la oración, en categorías predefinidas, como nombres de personas, organizaciones, ubicaciones)
- Nivel semántico: El nivel semántico se enfoca en comprender el significado del texto. El objetivo principal del procesamiento semántico es convertir automáticamente una oración del lenguaje natural en una representación formal de su significado, por ejemplo, la representación basada en ontologías (modelo de datos que representa un conjunto de conceptos dentro de un dominio y las relaciones entre esos conceptos), el modelo de espacio vectorial (modelo de representación básico que representa el texto como una matriz de término por documento), la representación basada en modelos de

temas: (modelado estadístico utilizado para descubrir los temas latentes o abstractos que ocurren en un conjunto de textos) y las técnicas avanzadas de incrustación (método eficiente para aprender representaciones vectoriales de palabras de alta calidad a partir de grandes cantidades de datos de texto no estructurados)

Diferenciamos entre tres tipos de tecnología PNL:

Técnica PNL: Una técnica de PNL es un método, enfoque, proceso o procedimiento práctico para realizar una tarea de PNL en particular, como tokenización, segmentación (chunking), etiquetado POS, análisis sintáctico superficial y de dependencias, reconocimiento de entidades, desambiguación, análisis semántico.

Herramienta PNL: Una herramienta de PNL es un sistema de software o una biblioteca de software que admite una o más técnicas de PNL, como:

- Stanford CoreNLP7[1]: ¡CoreNLP es su ventanilla única para el procesamiento del lenguaje natural en Java!, CoreNLP permite a los usuarios derivar anotaciones lingüísticas para el texto, incluidos los límites de oraciones y tokens, partes del discurso, entidades nombradas, valores numéricos y de tiempo, análisis de dependencias y constituyentes, correferencia, sentimiento, atribuciones de citas y relaciones. CoreNLP actualmente admite 6 idiomas: árabe, chino, inglés, francés, alemán y español. La pieza central de CoreNLP es la tubería. Las canalizaciones toman texto sin formato, ejecutan una serie de anotadores de PNL en el texto y producen un conjunto final de anotaciones. Las canalizaciones producen CoreDocuments, objetos de datos que contienen toda la información de anotaciones, accesibles con una API simple y serializables en un búfer de protocolo de Google.
- NLTK[2]: NLTK es una plataforma líder para crear programas Python que funcionen con datos de lenguaje humano. Proporciona interfaces fáciles de usar para más de 50 corpus y recursos léxicos como WordNet, junto con un

[1] https://stanfordnlp.github.io/CoreNLP/
[2] https://www.nltk.org/

conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de nivel industrial, y un foro de discusión activo. NLTK ha sido llamado "una herramienta maravillosa para enseñar y trabajar en lingüística computacional usando Python" y "una biblioteca increíble para jugar con el lenguaje natural". El procesamiento del lenguaje natural con Python proporciona una introducción práctica a la programación para el procesamiento del lenguaje. Escrito por los creadores de NLTK, guía al lector a través de los fundamentos para escribir programas Python, trabajar con corpus, categorizar texto, analizar la estructura lingüística y más. La versión en línea del libro se ha actualizado para Python 3 y NLTK 3.

- Pattern.es: es una biblioteca para Python que contiene herramientas para la conjugación de verbos, la singularización o la pluralización de sustantivos, la división de chunks y permite realizar el etiquetado POS para español. En la versión para el idioma inglés adiciona una interfaz para WordNet. Su instalación y uso son muy sencillos. Para el etiquetado gramatical utiliza Penn TreeBank Tags. Penn TreeBank está basado en el corpus Brown, pionero en etiquetado POS para inglés. A pesar de haberse creado para el inglés, Pattern lo usa también para el español. Cada etiqueta está formada con dos o tres caracteres que indican la función que cumple cada palabra en una oración.
- SpaCy: SpaCy es una biblioteca para el procesamiento avanzado de lenguaje natural en Python. Incluye modelos estadísticos pre-entrenados y vectores de palabras, admite tokenización para más de 45 idiomas. Provee etiquetado, análisis y reconocimiento de entidades nombradas y una fácil integración de aprendizaje profundo. Es muy sencilla de utilizar, basta con incorporar desde un programa Python el modelo para el español. Devuelve un etiquetado POS completo, no sólo indica la función de la palabra en la oración, sino otros datos tales como tiempo verbal, persona, número, modo, género, entre otros. La tokenizacion y el etiquetado gramatical se basan en el corpus OntoNotes57 que sigue la sintaxis de Penn TreeBank.

- Freeling[3]: Freeling es una librería de código abierto para el análisis de texto (tokenización, análisis morfológico, detección de entidades nombradas, etiquetado POS, etc.) para una variedad de idiomas. Desde su primera versión, tiene soporte para el español. Fue desarrollado por el Centro de Tecnologías y Aplicaciones del Lenguaje y el Habla (TALP) de la Universidad Politécnica de Catalunya. Se puede utilizar y ampliar los recursos lingüísticos por defecto (diccionarios, lexicones, gramáticas, etc.) adaptándolos a dominios específicos. Su código fue escrito en C++ bajo una arquitectura cliente – servidor. Existen diversas APIs que permiten utilizar Freeling en otros lenguajes, incluido Python. Su instalación no es sencilla, sobre todo en entornos Windows. Sin embargo, subsanado este inconveniente, al importar la API desde el programa Python se puede invocar a todas las funciones implementadas. El analizador morfológico de Freeling realiza el etiquetado POS usando las etiquetas EAGLES.

Recurso PNL: Un recurso de PNL es un recurso de datos lingüísticos para respaldar técnicas o herramientas de PNL, que puede ser un léxico del lenguaje (es decir, un diccionario) o un corpus (es decir, una colección de textos). Los léxicos existentes incluyen WordNet10 y FrameNet11, mientras que los ejemplos de corpus incluyen British National Corpus12 y Brown Corpus13.

1.2 Soluciones reportadas en la literatura para la extracción automática de requisitos

1.2.1 Procesamiento de lenguaje natural aplicado a la ingeniería de requisitos

El Procesamiento del lenguaje natural (NLP) se usa ampliamente para respaldar la automatización de diferentes tareas de ingeniería de requisitos (RE). La mayoría de los enfoques propuestos comienzan con varios pasos de NLP que analizan las

[3] https://nlp.lsi.upc.edu/freeling/node/1

declaraciones de requisitos, extraen su información lingüística y las convierten en representaciones fáciles de procesar, como listas de características o representaciones vectoriales basadas en incrustaciones. Estas representaciones basadas en NLP generalmente se usan en una etapa posterior como entradas para técnicas de aprendizaje automático o métodos basados en reglas. Por lo tanto, las representaciones de los requisitos juegan un papel importante en la determinación de la precisión de los diferentes enfoques.

Se han publicado muchas revisiones sobre la relación entre las tareas de PNL y RE [38] [39] [40] [33, 34, 35]

Algunas de estas revisiones brindan una imagen amplia de las actividades y tecnologías de PNL utilizadas en el dominio RE [41] [42] [38]. Dermeval et al. [41] realizó una revisión sistemática de la literatura para identificar los principales estudios sobre el uso de ontologías en el dominio RE. Este estudio consideró 77 estudios publicados entre enero de 2007 y octubre de 2013. Nazir et al. [42] investigó las aplicaciones de la PNL en el contexto de la ER. Incluyó 27 estudios publicados entre 2002 y 2016. Zhao et al. [38] presentó una descripción general completa de las aplicaciones de la PNL en la investigación de ER centrándose en el estado de la literatura, el estado de la investigación empírica, el enfoque de la investigación, el estado de la práctica y las tecnologías de PNL utilizadas. Este estudio revisó 404 estudios primarios relevantes informados entre 1983 y abril de 2019.

Otras revisiones se centraron en problemas específicos de ER. Bozyigit et al. [43] proporcionó una revisión de 44 estudios primarios relacionados con la transformación automática de requisitos de software en modelos conceptuales. Abarca obras publicadas entre 1996 y 2020.

Por otro lado, varias revisiones han limitado su trabajo a plantillas específicas para los requisitos [39] [40]. Amna et al. [40] revisaron estudios que investigan o desarrollan soluciones para problemas relacionados con la ambigüedad en las historias de usuario. El estudio abarcó 36 investigaciones publicadas entre 2001 y 2020. Del mismo modo, Raharjana et al. [39] presentó una revisión sistemática de

la literatura para la investigación relacionada con el papel de la PNL en la especificación de historias de usuario. Este trabajo encontró 30 estudios primarios entre enero de 2009 y diciembre de 2020.

1.2.2 Análisis de trabajos relacionados

La solución reportada por [44] describe : Cargamos el diseño del documento y lo analizamos en oraciones. Dentro de las oraciones, también convertimos las oraciones en palabras individuales y recuperamos las etiquetas de parte del discurso para cada palabra. Probamos dos enfoques para extraer requisitos de software, (1) extraer oraciones que contienen verbos en su forma base y (2) extraer frases que contienen palabras clave que se utilizan a menudo para indicar requisitos. Verificamos la ambigüedad semántica, si existe, construimos el árbol estructural de la oración para identificar el sujeto de la oración anterior: desarrollamos y definimos una gramática y analizamos la oración en un árbol de acuerdo con esta gramática; el sujeto se identifica como el sintagma nominal principal en el árbol.

En [45] la metodología destinada es a mejorar la detección de NFR en los documentos de requisitos. Usa el Stanford Parser (equipado con el etiquetador POS de Brill y un lematizador morfológico) para derivar morfológicamente las palabras y extraer cinco características sintácticas de cada una de las instancias de entrenamiento (oraciones) del corpus.

En [46] el procedimiento q se desarrolla es el enfoque L'Ecritoire en el que esta relación es bidireccional: así como los objetivos pueden ayudar en el descubrimiento de escenarios, los escenarios pueden ayudar en el descubrimiento objetivo. La solución total está en dos partes, se crean escenarios textuales que es el que produce objetivo. La correspondencia entre un patrón semántico y el modelo de escenario define la relación entre la forma textual de un escenario y su forma conceptual.

En [47] una tarea importante para lograr este objetivo es construir una ontología que consista en un conjunto de conceptos, es decir, entidades, atributos y relaciones basadas en el dominio de aplicación de interés. La ontología construida aquí representa el conocimiento del dominio y los requisitos son el subconjunto especializado del mismo.

En este documento [48], se propone un enfoque semiautomático llamado NFR-Specifier, cuyo objetivo es generar especificaciones precisas a partir de requisitos informales, incluidos los NFR. El enfoque consta de cinco módulos, a saber. Pre-procesamiento, resolución de ambigüedades, formación de ontologías SRS, generación de diagramas UML y clasificación de NFR. Inicialmente, el ingeniero de requisitos recopila el conocimiento del dominio de los usuarios por medio de varios enfoques de comunicación, a saber, cuestionarios, entrevistas, checklist, prototipado, reuniones, entre otros. Una vez finalizada la fase de comunicación, el ingeniero de requisitos representa la información recopilada por medio de archivos de texto, documentos, gráficos o modelos UML (es decir, caso de uso, clase, diagrama de secuencia).

Este trabajo [49] explora cómo la cantidad y el tipo de conocimiento afectan la calidad de obtención de requisitos en dos simulaciones consecutivas. Se calcula con las salidas del automatismo con el patrón oro introducido en la última palabra. La recuperación puede verse como una medida de completitud, comparando el número de requisitos identificados con el número total de requisitos existentes en un documento.

El documento [50] propone un método para obtener los requisitos del usuario enla industria de maquinaria basado en la regla de asociación de texto. El primer paso es el pre-procesamiento de datos de los requisitos del usuario. El modelo de espacio vectorial se utiliza para describir los requisitos del usuario. En segundo lugar, se utiliza una teoría mejorada de la regla de asociación gris para calcular el grado de correlación entre las palabras características y los nombres propios de la industria de la maquinaria. Luego se construye la matriz de candidatos a nombres propios

seleccionando una palabra de mayor grado de correlación. Finalmente, el requerimiento del usuario se obtiene utilizando la matriz ponderada.

1.2.2.1 Tabla resumen de revisión bibliográfica

Trabajos relacionados	***Tipo de requisito***	***Idioma a procesar***	***Herramientas PLN***	***Análisis sintáctico***			***Reducción de redundancias***		
				Análisis de dependencias	***Patrones sintácticos***	**Otros**	**Similitud de coseno**	**Algoritmo de agrupamiento**	**Otros**
[44]	funcional	ingles	NLTK	X			X		
[45]	no funcional	ingles	Stanford			clasificación			
[46]	funcional	ingles	NLTK		X				
[47]	funcional	ingles			X				
[48]	no funcional	ingles	Stanford		X				desambiguación
[49]	funcional	ingles	Stanford		X		X		
[50]	funcional	ingles	Stanford		X				

Tabla 1 Resumen de trabajos relacionado

1.3 Métodos de evaluación

Las tareas de evaluación de la calidad se ocupan de detectar defectos en las especificaciones de los requisitos del software

1.3.1 Métodos de evaluación reportados en la literatura

En el artículo [44] plantean como método para evaluar sus resultados:

Las oraciones extraídas, que se han identificado como requisitos probables, se envían a un archivo. Se crean vectores TF-IDF para cada una de las frases extraídas y cada uno de los requisitos en DOORS. Luego se calcula la similitud del coseno

para cada posible par de vectores de oraciones y vectores de requisitos de DOORS. Si la similitud del coseno para cualquier par dado supera un umbral, aquí 0,95 en una escala de 0 a 1, se dice que este requisito es una coincidencia con una oración extraída. Se usa el vectorizador TF-IDF y la métrica de similitud de coseno por pares de scikitlearn (paquete de código abierto para herramientas de aprendizaje automático en Python).

1.3.2 Métricas de análisis de resultados

Las métricas seleccionadas para la evaluación de la propuesta son precisión, cobertura y medida-F. La mayoría de los métodos reportados en el estado del arte hacen uso de estas métricas para evaluar sus resultados, esto permite comparar los resultados del método propuesto con los reportados por otras soluciones reportadas.

1.4 Análisis de tecnologías de implementación

La implementación de la solución se desarrolla en el lenguaje de programación Python el cual es un lenguaje de programación multiparadigma que soporta varios paradigmas de programación como orientación a objetos, estructurada, programación imperativa y, en menor medida programación funcional. Se utiliza el marco de trabajo Django por su gran rendimiento y flexibilidad, lo que permite un desarrollo ágil y reutilizable. La herramienta de procesamiento de lenguaje natural SpaCy se utiliza por su fácil acceso y todo el potencial que posee.

1.3.1 Python

Python[4] es un lenguaje de programación interpretado, orientado a objetos de alto nivel y con semántica dinámica. Su sintaxis hace énfasis en la legibilidad del código, lo que facilita su depuración y, por tanto, favorece la productividad. Ofrece la potencia y la flexibilidad de los lenguajes compilados con una curva de aprendizaje suave. Aunque Python fue creado como lenguaje de programación de uso general, cuenta con una serie de librerías y entornos de desarrollo para cada una de las fases del proceso de Data Science. Esto, sumado a su potencia, su carácter open

[4] https://www.python.org/

source y su facilidad de aprendizaje le ha llevado a tomar la delantera a otros lenguajes propios de la analítica de datos por medio de Machine Learning como pueden ser SAS (software comercial líder hasta el momento). Python fue creado por Guido Van Rossum en 1991 y, como curiosidad, debe su nombre a la gran afición de su creador por las películas del grupo Monty Python. Además de librerías de herramientas científicas, numéricas, de herramientas de análisis y estructuras de datos, o de algoritmos de Machine Learning como NumPy, SciPy, Matplotlib, Pandas o PyBrain, Python ofrece entornos interactivos de programación orientados al Data Science.

1.3.2 PyCharm (IDE)

PyCharm[5] es un entorno de desarrollo integrado (IDE) utilizado en la programación de computadoras , específicamente para el lenguaje de programación Python . Está desarrollado por la empresa checa JetBrains (anteriormente conocida como IntelliJ). Proporciona análisis de código, un depurador gráfico, un probador de unidades integrado, integración con sistemas de control de versiones (VCS) y es compatible con el desarrollo web con Django , así como con la ciencia de datos con Anaconda.

PyCharm es multiplataforma , con versiones de Windows , macOS y Linux . La edición comunitaria se publica bajo la licencia Apache, y también hay una versión educativa, así como una edición profesional con características adicionales (publicada bajo una licencia propietaria financiada por suscripción).

1.3.2.1 Comparación con otros IDE de Python

PyCharm vs Jupyter

Jupyter[6]

Jupyter notebook es un IDE de código abierto que se utiliza para crear documentos Jupyter que se pueden crear y compartir con códigos activos. Además, es un

[5] https://www.jetbrains.com/help/pycharm/quick-start-guide.html
[6] https://jupyter.org/

entorno computacional interactivo basado en la web. El cuaderno Jupyter puede admitir varios lenguajes que son populares en la ciencia de datos, como Python, Julia, Scala, R, etc.

PyCharm

PyCharm es un IDE desarrollado por JetBrains y creado específicamente para Python. Tiene varias características como análisis de código, probador de unidades integrado, depurador Python integrado, soporte para frameworks web, etc. Pycharm es particularmente útil en machine learning porque admite bibliotecas como Pandas, Matplotlib, Scikit-Learn, NumPy, etc.

Diferencias entre Jupyter y PyCharm[7]:

Jupyter	PyCharm
El portátil Jupyter es una plataforma informática interactiva basada en web.	Pycharm es un editor de código inteligente.
El cuaderno combina código en vivo, ecuaciones, texto narrativo, visualizaciones, paneles interactivos y otros medios.	El editor proporciona soporte de primera clase para Python, JavaScript, CoffeeScript, TypeScript, CSS, lenguaje de plantillas popular y más. ¡Aproveche la finalización de código consciente del idioma, la detección de errores y las correcciones de código sobre la marcha!
Se puede clasificar como una herramienta en los "Cuadernos de ciencia de datos"	PyCharm se agrupa en “Entorno de desarrollo integrado(IDE)”.
Proporciona ejecución de código en línea mediante bloques.	Proporciona finalización automática inteligente.
Puede tener un tema y es compatible con kernel y látex.	Es una poderosa refactorización, integración virtual e integración Git

[7] https://es.acervolima.com/diferencia-entre-jupyter-y-pychar

Tabla 2 Diferencias entre IDE Python

1.3.3 SpaCy

SpaCy[8] es una librería de software para procesamiento de lenguajes naturales desarrollado por Matt Honnibal y programado en lenguaje Python. Fue lanzado en febrero de 2015 estando su desarrollo activo y siendo utilizado en distintos entornos. Es software libre con Licencia MIT su repositorio se encuentra disponible en Github.

Características principales:

- Tokenización no destructiva.
- Compatibilidad con tokenización alfa para más de 65 idiomas.
- Soporte integrado para componentes de canalización entrenables, como reconocimiento de entidades nombradas, etiquetado de parte de la voz, análisis de dependencias, clasificación de texto, vinculación de entidades, entre otros.
- Modelos estadísticos para 17 idiomas.
- Aprendizaje multitarea con transformadores previamente entrenados como BERT.

1.3.3.1 Comparación con otras herramientas de NLP

La Universidad Católica de Salta realizo una comparación de herramientas de procesamiento de textos en español para Python y explica lo siguiente [51]:

De cada herramienta se tuvieron en cuenta algunos aspectos como la facilidad en su instalación, sobre todo para una persona con conocimientos básicos o nulos en programación; la necesidad de utilizar una interfaz de comunicación; la presencia de funciones específicas para segmentación y tokenización; el tipo de etiquetado POS que realizan y la implementación de lematizador para el lenguaje español. A

[8] https://spacy.io/

modo de resumen se presenta un cuadro comparativo de algunas funciones básicas de procesamiento de texto y las herramientas mencionadas que fueron objeto de evaluación.

Herramienta	Código	Segmentación y tokenización	Etiquetado POS	Lematización
NLTK	Nativo Python	Si	Eagles	No
Freeling	API	Si	Eagles	Si
Pattern.es	Nativo Python	Si	Penn TreeBank	Si
SpaCy	Nativo Python	Si	Penn TreeBank	Si
Stanford NLP	API	Si	Eagles	Si

Tabla 3 Comparación entre herramientas NLP

En la segunda columna de la tabla 3 se muestra la relación de las herramientas con el lenguaje Python. NLTK, Pattern.es y SpaCy se pueden instalar muy fácilmente de los repositorios de Python a través de la línea de comando. No ocurre lo mismo con Stanford NLP y Freeling. Ambos paquetes no son nativos de Python; fueron escritos en Java y C++, respectivamente. La instalación de Stanford es más sencilla que la de Freeling, aunque ambos necesitan sendos servidores que se estén ejecutando para funcionar. Sin embargo, si no se requiere de todo el potencial de ambas herramientas, existe otra opción para su uso: Stanford se puede utilizar a través de NLTK y Freeling provee un ejecutable que realiza la segmentación, la tokenización, el etiquetado POS y la lematización recibiendo un archivo como entrada. Todas las herramientas poseen funciones para segmentación y tokenización. NLTK, Freeling y Stanford NLP realizan el etiquetado POS con etiquetas Eagles mientras que Pattern.es y SpaCy lo hacen con etiquetas de Penn TreeBank. Salvo NLTK, las demás herramientas, implementan un lematizador para el idioma español. El lematizador disponible en NLTK, WordNet, está disponible para el idioma inglés.

NLTK requiere que se hagan una serie de operaciones sobre el texto de segmentación, tokenizacion, etiquetado y Chunking antes de hacer el

reconocimiento de entidades nombradas como tal, esto más control al usuario sobre el proceso. Por otro lado, en SpaCy se permite trabajar sobre un texto a nivel de entidad directamente, lo que simplifica mucho el proceso.

Capítulo 2. Extracción automática de requisitos de software

En el presente capítulo se documenta toda la propuesta de solución. Se exponen los elementos que forman parte de la solución. Se presentan los diagramas en Lenguaje Unificado de Modelado (UML, Unified Modeling Language) necesarios para explicar el flujo de las actividades que forman parte de la arquitectura de la solución propuesta, como los casos de uso y modelo del dominio. Se muestra para un mejor entendimiento de la solución las principales etapas de desarrollo en un diagrama de flujo.

2.1 Solución teórica

2.1.1 Descripción general

En esta se sección describe la solución propuesta en el trabajo. Las principales etapas del proceso de desarrollo se describen en la Ilustración 3. En primer lugar, el texto a analizar es pre-procesado, se hace una limpieza del texto, eliminando signos de puntuación y convirtiéndolo en su totalidad a letras minúsculas. A continuación, se lleva acabo el análisis sintáctico, a partir del etiquetado de cada tokens del texto según su clasificación sintáctica, se realiza un mapeo con patrones basados en reglas definidos previamente, extrayendo así frases que dan lugar a sentencias de requisitos de software. Tomando como base las frases extraídas en la fase anterior se realiza un refinamiento del volumen de sentencias extraídas, reduciendo a una, aquellas que sean completamente iguales. Luego se agrupan aquellas que tengan un 90% de similitud entre ellas. Finalmente se exporta un archivo txt donde se muestran los resultados obtenidos.

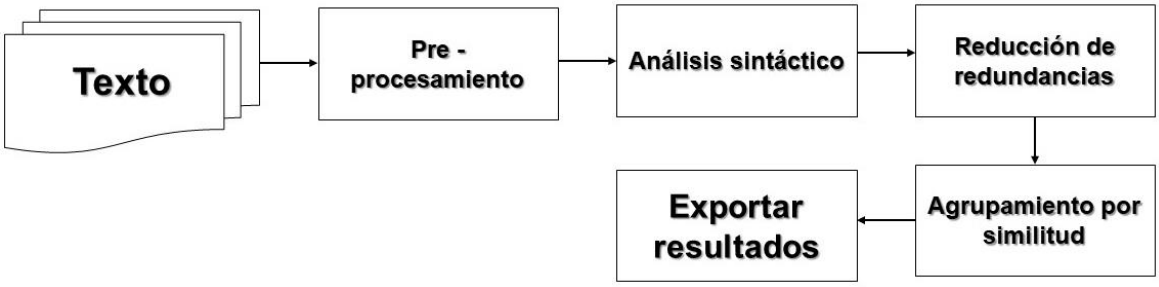

Ilustración 3 Flujo de trabajo de la solución propuesta

2.1.2 Pre – procesamiento

Como se planteaba en la subsección anterior la solución contará fundamentalmente con un componente para la preparación o pre-procesamiento de la información haciendo uso de la biblioteca de Procesamiento de Lenguaje Natural para español, SpaCy.

El componente de pre-procesamiento se encargará de limpiar el texto de ruido y estandarizar para que sea fácil de procesar. Algunos de estos pasos incluyen quitar las puntuaciones, transformar el texto a minúscula, o eliminar palabras frecuentes usadas en el lenguaje como preposiciones, artículos, entre otros, que por sí mismas no aportan mucha información, además de usar la raíz de las palabras. En la Ilustración 4 se muestra un diagrama de actividades del pre - procesamiento.

Dentro de las principales actividades dentro del módulo de pre-procesamiento se encuentra:

- Tokenización: esta actividad consiste en dividir el texto sin procesar en pequeños trozos. La tokenización dividirá el texto en bruto en palabras u oraciones llamadas tokens. Estos tokens permitirán comprender el contexto o desarrollar el modelo para la PNL, ayudando a interpretar el significado del texto al analizar la secuencia de las palabras.
- Tras la tokenización, se procede al stopwords. Se trata de palabras muy habituales en cualquier idioma que, sin embargo, aportan poco valor tales como artículos, signos de puntuación, palabras de enlace, etc. Por ello, es interesante identificarlas y «excluirlas». Es una forma de «limpiar» el texto.
- Etiquetado PoS (Part-of-speech) tagging: Se encarga de clasificar las partes de las oraciones en verbo, sustantivo, adjetivo, preposición entre otras.
- Lematización: es el proceso mediante el cual las palabras de un texto que pertenecen a un mismo paradigma flexivo o derivativo son llevadas a una forma normal que representa a toda la clase. En este caso solo se les realiza a los verbos conjugados.

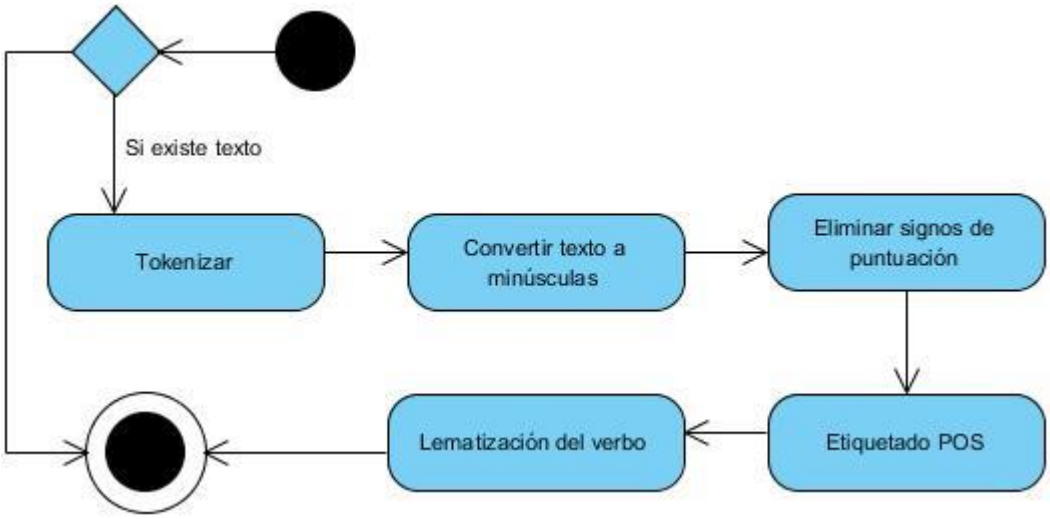

Ilustración 4 Diagrama de actividades del pre - procesamiento

2.1.3 Análisis sintáctico

El análisis sintáctico tiene como función etiquetar cada uno de los componentes sintácticos que aparecen en la oración y analizar cómo las palabras se combinan para formar construcciones gramaticalmente correctas. El resultado de este proceso consiste en generar la estructura correspondiente a las categorías sintácticas formadas por cada una de las unidades léxicas que aparecen en la oración.

2.1.3.1 Análisis sintáctico basado en patrones léxicos – sintácticos

Un patrón es una descripción de la forma que pueden tomar los lexemas de un token. En el caso de una palabra clave como token, el patrón es sólo la secuencia de caracteres que forman la palabra clave. Para los identificadores y algunos otros tokens, el patrón es una estructura más compleja que se relaciona mediante muchas cadenas.

Para extraer las frases que dan lugar a las sentencias de requisitos, se definieron previamente un conjunto de patrones léxicos – sintácticos que fueron generados a partir de un proceso estadístico realizado a 40 tesis del curso 2020-2021 de la facultad de Ingeniería Informática de la CUJAE, donde fueron tomados cada uno de

los requisitos funcionales de los casos de uso, dando lugar a un total de 555 requisitos funcionales analizados. Estos requisitos fueron procesados por el analizador sintáctico de la biblioteca de SpaCy y a partir de la etiqueta gramatical de cada tokens fueron formados los patrones.

Luego de realizado el pre-procesamiento del texto, partiendo de la clasificación gramatical de cada tokens se realiza un mapeo con los patrones predefinidos y se generan las frases que dan lugar a las sentencias de requisitos.

En la tabla se muestran cada uno de estos patrones léxico – sintácticos.

Patrón léxico – sintáctico	***Ejemplo de frase***
VERB NOUN	analizar muestras, recetar medicamentos
VER NOUN ADJ	analizar muestras bilógicas
VERB DET NOUN	establecer un diagnóstico, entregar las muestras
VERB NOUN ADP NOUN	incluir detectores de humo
VERB NOUN ADP DET NOUN	recoger datos sobre su estructura
VERB NOUN ADP NOUN ADJ	obtener reportes en tiempo real
VERB NOUN ADP NOUN ADP NOUN	exportar base de datos del día
VERB NOUN ADJ ADP NOUN	gestionar tratamientos asociados al paciente
VERB NOUN VERB DET NOUN ADJ	reproducir voz alertando el billete reconocido
VERB ADJ PRON VERB ADJ DET NOUN ADJ	ejecutar nodo que tiene implementado el algoritmo RSKkNN
VERB ADJ PRON VERB DET NOUN AUX	ejecutar nodo que crea el árbol IUR-tree
VERB CCONJ VERB DET NOUN ADP NOUN	salvar y restaurar la base de datos
VERB ADV ADP NOUN ADP DET NOUN ADJ	organizar No. de lista de los estudiantes matriculados
VERB NOUN CCONJ NOUN	enviar reclamación o protesta
VERB DET NOUN ADJ	explicar la propuesta diseñada
VERB DET NOUN CCONJ DET NOUN	visualizar las delegaciones y sus atletas

VERB ADP NOUN	apelar a sanción
VERB NOUN ADJ ADP NOUN ADJ	cifrar código binario con contraseña modulada
VERB NOUN ADJ ADJ ADP DET NOUN ADP NOUN	obtener código binario original en el buffer de memoria
VERB NOUN ADP NOUN ADP NOUN	enviar correo de petición de acceso

Tabla 4 patrones léxico – sintácticos

2.1.4 Estrategia de reducción de redundancias

Como estrategia de reducción de redundancias se definió la siguiente:

- Reducir a una, aquellas frases que sean completamente iguales, almacenando las restantes en otro archivo independiente a la solución, debido a que el objetivo de la solución propuesta es asistir el trabajo del analista – diseñador del software, se decidió no eliminar ninguna frase generada y que sea el especialista encargado quien decida la relevancia de la misma.
- Agrupar aquellas frases que tengan un 90% de similitud sintáctica entre ellas. Se utilizan dos métricas de similitud para agrupar:
  - ✓ Métrica Levenshtein : La distancia de *Levenshtein* es quizá la más común y simple de las métricas de edición basadas en caracteres, por lo que generalmente (y quizá también por la dificultad de pronunciación) se le conoce simplemente como “distancia de edición”. Esta métrica es, en cierto sentido, una forma de cuantificar la diferencia entre dos cadenas de texto, sin embargo, aunque por su descripción parezca sencilla, la forma de calcularla no es trivial como en el caso de la distancia de *Hamming,* además, a diferencia de ésta última, la distancia de *Levenshtein* tradicional es capaz de comparar cadenas de diferente tamaño.

- ✓ Métrica Similitud de coseno: En el análisis de datos , la similitud del coseno es una medida de similitud entre dos secuencias de números. Para definirlo, las sucesiones se ven como vectores en un espacio de producto interior , y la similitud de coseno se define como el coseno del ángulo entre ellos, es decir, el producto escalar de los vectores dividido por el producto de sus longitudes. De ello se deduce que la similitud del coseno no depende de las magnitudes de los vectores, sino solo de su ángulo. La semejanza del coseno siempre pertenece al intervalo [1, -1]. La similitud del coseno se usa particularmente en el espacio positivo, donde el resultado está claramente delimitado en [0, 1].

❖ Al calcular la similitud entre las frases se generan sublistas con cada uno de los grupos y se devuelve una lista general con cada uno de los grupos.

2.2 Desarrollo de la solución de extracción de requisitos de software

2.2.1 Diagrama de casos de uso y descripción de casos de uso

Un Modelo de Casos de Uso o Modelo de Vista de Casos de Uso constituye un modelo donde se perciben las funcionalidades del sistema por usuarios externos llamados actores. Un caso de uso es una unidad coherente de funcionalidad expresada como una transacción entre los actores y el sistema. El propósito de esta vista es listar los actores y los casos de usos y mostrar que actores participan en cada caso de uso. En la Ilustración 5 se muestra el diagrama de Casos de Uso del Sistema que se diseñó como parte de la propuesta.

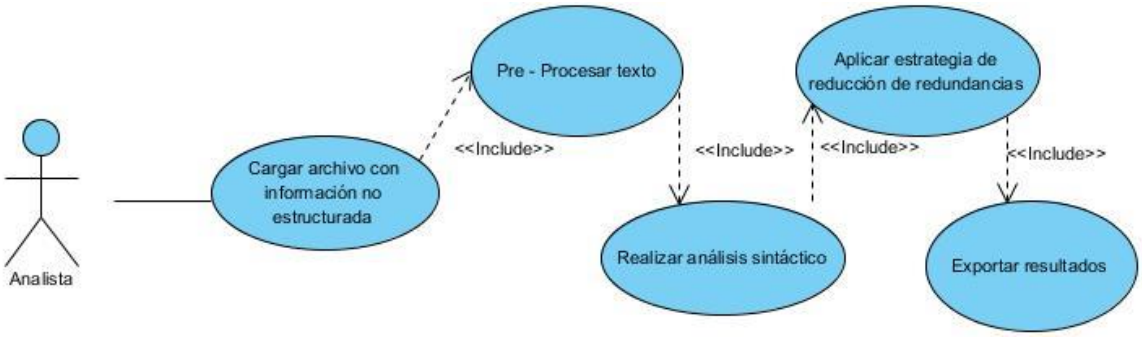

Ilustración 5 Diagrama de caso de uso

2.2.1.1 Actores del sistema

En la Tabla se muestra los actores del sistema y una descripción.

Actores del sistema	Descripción
Analista	Es el actor que interactúa directamente con el sistema. Es el encargado de iniciar todos los procesos del sistema. Utiliza el sistema para generar los requisitos de software.

Tabla 5 Actores del sistema y una descripción

2.2.1.2 Especificación de alto nivel de caso de uso

La especificación de cada caso de uso se muestra en las Tablas 5, 6, 7, y 8

Tabla 6 Descripción del caso de uso Pre-procesar texto.

Caso de uso	Pre – procesar texto
Actores	Analista
Propósito	Pre procesar el texto para realizar análisis sintáctico.
Descripción	Para el pre-procesamiento del texto se buscaron herramientas que permitieran realizar las técnicas de tokenización de

	las palabras, se llevaron las palabras a minúscula, se eliminaron las palabras que no aportan información significativa del texto y se eliminaron signos de puntuación.
Responsabilidad	Pre procesar el texto para realizar análisis sintáctico.
Precondiciones	Existen un fichero en formato texto
Poscondiciones	Se obtiene texto preprocesado

Tabla 7 Descripción del caso de uso Realizar análisis sintáctico

Caso de uso	Realizar análisis sintáctico
Actores	Analista
Propósito	Generar frases a partir del análisis sintáctico
Descripción	Para la generación de las frases se definieron un conjunto de patrones donde de mapean cada uno con la clasificación gramatical de cada token en el texto.
Responsabilidad	Generar sentencias de requisitos
Precondiciones	Texto previamente procesado
Poscondiciones	Se obtienen frases

Tabla 8 Descripción de caso de uso Aplicar estrategia de reducción de redundancias

Caso de uso	Aplicar estrategia de reducción de redundancias
Actores	Analista
Propósito	Reducir el volumen de frases generadas

Descripción	Para la reducción de redundancias se definió como estrategia • Reducir a una, aquellas frases que fueran completamente iguales, separando el resto en otro archivo. • Aplicar métricas de similitud Levenshtein y Coseno creando grupos de frases según el 90% de similitud
Responsabilidad	Reducir la redundancias de frases
Precondiciones	Conjunto de frases generadas
Poscondiciones	Se obtienen los requisitos agrupados por similitud

Tabla 9 Descripción de caso de uso Exportar resultados.

Caso de uso	Exportar resultados
Actores	Analista
Propósito	Mostrar los resultados al analista
Descripción	Generar archivo txt con grupos de requisitos de software
Responsabilidad	Devolver resultados
Precondiciones	Conjunto de frases generadas y no redundantes
Poscondiciones	Se obtienen txt con los requisitos.

2.2.2 Especificación de requisitos

Requisito R1	Cargar datos a analizar

Rol	Público		
Descripción textual	Permite la carga de documentos a procesar		
Datos de entrada			
Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Documento	no	Formatos variados	Se debe notificar la carga errónea del documento y el motivo
Precondiciones			
Respuesta esperada	Se identifican y etiquetan entidades		

Requisito R2	Pre procesamiento de la información		
Rol			
Descripción textual	El caso de uso inicia una vez realizada la carga de los documentos, procediendo a la limpieza y preparación de la información a procesar. Comenzando con la Tokenización del texto		
Datos de entrada			
Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Texto en lenguaje natural			
Precondiciones	Se debe haber cargado al menos un documento		
Respuesta esperada	Se separa el texto en Tokens		

Requisito R3	Etiquetar términos
Rol	
Descripción textual	Permite identificar términos en el texto y clasificarlos
Datos de entrada	

Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Texto en lenguaje natural			
Precondiciones	Se verifica el fin del paso anterior		
Respuesta esperada	Se etiqueta cada token según su función sintáctica		

Requisito R4	Análisis sintáctico
Rol	
Descripción textual	Analizar la gramática de las palabras en una oración, y su posición en la oración, y en cierta medida la relación con otras palabras.
Datos de entrada	

Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Texto preprocesado			
Precondiciones			
Respuesta esperada	Devuelve texto y etiquetado gramatical		

Requisito R5	Identificar frases
Rol	
Descripción textual	A partir de reglas sintácticas predefinidas se construyen frases que cumplan con estas reglas.
Datos de entrada	

Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Texto preprocesado			

Precondiciones	Deben existir reglas sintácticas
Respuesta esperada	Se generan frases a partir de las reglas prestablecidas.

Requisito R6		Lematización	
Rol			
Descripción textual		Los verbos conjugados identificados se reducen a su forma canónica para simplificar el análisis estadístico	
Datos de entrada			
Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Frases generadas			
Precondiciones	Se verifica el fin del paso anterior		
Respuesta esperada	Se representa cada verbo identificado en la frase, en su forma canónica		

Requisito R7		Reducir redundancias	
Rol			
Descripción textual		A partir del análisis de similitud se reducen aquellas freses redundantes.	
Datos de entrada			
Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Frases generadas			
Precondiciones	Se deben haber identificado al menos dos frases		
Respuesta esperada	Se reduce el número de frases		

Requisito R8	Agrupar por similitud

Rol			
Descripción textual		Se verifica el nivel de similitud entre las frases y se agrupan por el 90% de similitud	
Datos de entrada			
Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Frases generadas			
Precondiciones	Se deben haber identificado frases		
Respuesta esperada	Se agrupan por similitud las frases		

Requisito R9		Determinar especificidad en el dominio	
Rol			
Descripción textual		Se determina la concordancia de las frases con el dominio de la problemática que se está analizando.	
Datos de entrada			
Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Frases generadas			
Precondiciones	Se deben haber identificado los aspectos anteriores		
Respuesta esperada	Se genera una representación formal del requisito a partir del procesamiento del texto y se verifica el análisis completo del texto		

Requisito R10	Exportar resultados
Rol	
Descripción textual	Una vez cumplido los pasos anteriores se permite la exportación de los resultados.
Datos de entrada	

Campos de entrada	**Admite Vacío**	**Tipo de Dato y su validación**	**Respuesta del sistema cuando el campo es nulo o se introducen valores inválidos**
Frases generadas			
Precondiciones	Se deben haber identificado los aspectos anteriores		
Respuesta esperada	Se exporta en formato txt los requisitos identificados		

2.2.3 Arquitectura del modelo

La implementación de la solución se lleva a cabo con el lenguaje de programación Python, con la versión 3.9.0, y PyCharm como repositorio de los paquetes y librerías. Python tiene muchas librerías para el desarrollo de diversos tipos de aplicaciones, en este trabajo en particular se emplean la librería SpaCy famosa en el empleo de análisis de fuentes textuales de información, con las que se pueden realizar tareas propias de la minería de texto como el pre - procesamiento de los datos, el agrupamiento, entre otras.

El modelo se divide en 5 paquetes principales: cargar_varios_archivos, pre_procesamiento, reducción_redundancias, exportar_resultados y SpaCy. La estructura de estos paquetes y la interacción entre ellos se representa en la Ilustración 6, usando un diseño multicapas.

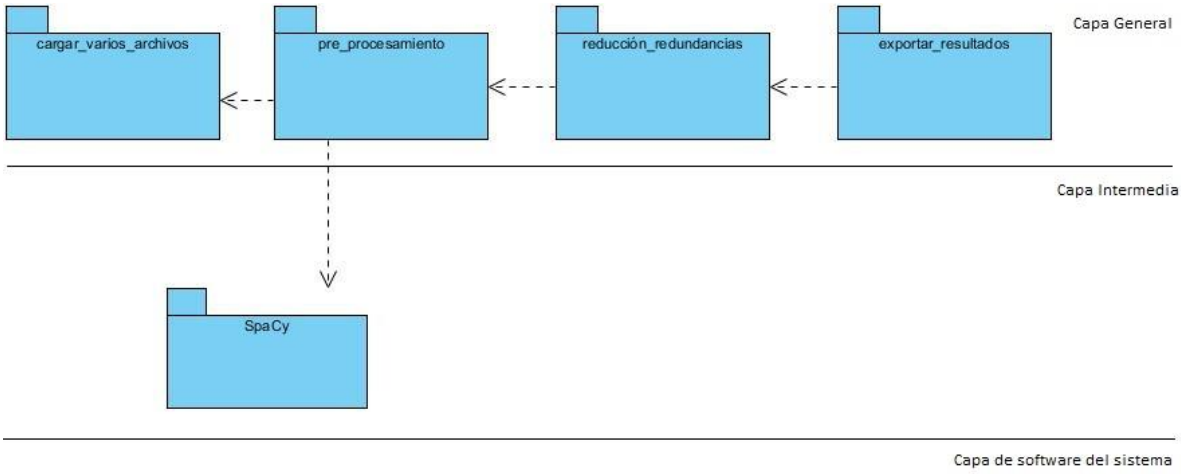

Ilustración 6 Estructura en capas del modelo.

2.2.4 Estilos y patrones de arquitectura

Arquitectura basada en componentes:

Representa una división del sistema en módulos más pequeños o subsistemas, que son unidades de software los cuales contienen un conjunto de funciones que tienen alguna relación entre ellas, realizan las acciones del sistema y que se implementan en archivos físicos.

Descripción detallada de los paquetes:

cargar_varios_archivos: paquete donde se implementa la funcionalidad de cargar el archivo a procesar.

pre_procesamiento: paquete donde se implementan las diferentes funcionalidades encargadas de la preparación del texto a procesar (tokenizacion, etiquetado, análisis sintáctico).

reducción_redundancias: paquete donde se implementan la funcionalidad encargada de eliminar freses repetidas (eliminar_redundancias) y la funcionalidad de agrupar dichas frases por un 90% de similitud entre ellas (agrupar_por_similitud)

exportar_resultados: paquete donde se exportan en archivo .txt las sentencias de requisitos generadas.

extraccion_requisitos: modulo general donde se invocan cada uno de los paquetes y sus respectivas funciones.

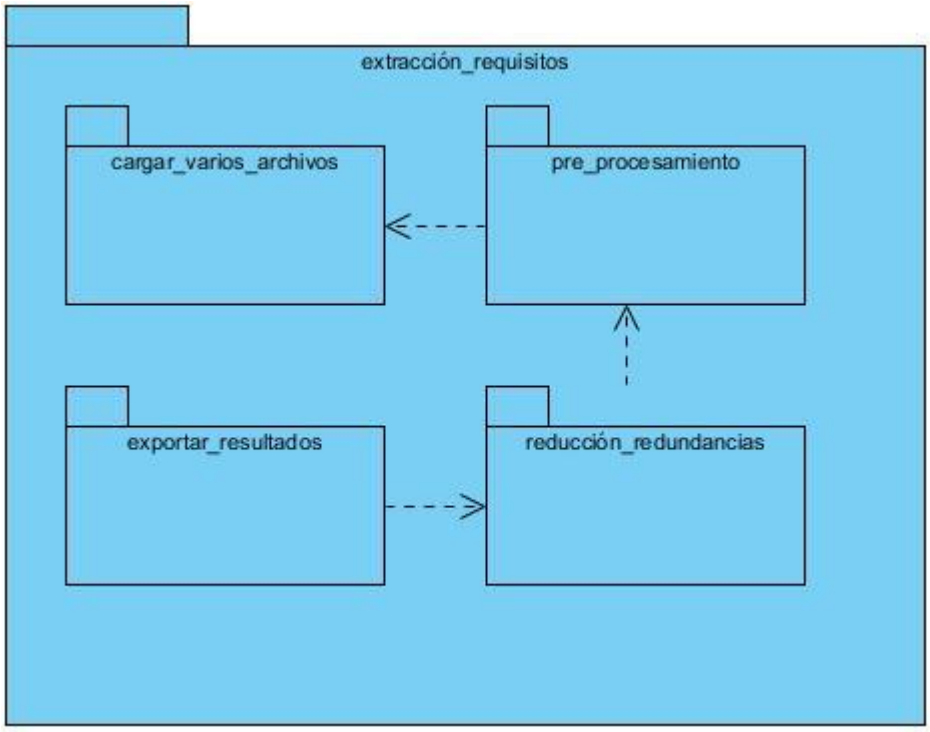

Ilustración 7 Estructura modular del sistema. Arquitectura basada en componentes.

Capítulo 3. Evaluación de la solución propuesta

En el presente capítulo se presenta el estudio experimental de la solución propuesta. Se evalúan los requisitos extraídos de manera automática de un grupo de caso de estudios definidos, comparándolos con los requisitos extraídos de forma manual, por un especialista en la materia, de dichos casos de estudios. Se llevó a cabo la evaluación de los mismos empleando las métricas habituales en la clasificación de texto, llamadas Precisión (P), Cobertura (C) y Medida-F (F).

3.1 Marco de evaluación

La solución propuesta fue evaluada tomando como referencias la extracción manual de requisitos de software de un conjunto de casos de estudio que se definieron previamente por el equipo de trabajo. Estos requisitos fueron comparados sintácticamente con los requisitos extraídos de forma automática, obteniendo una tercera lista de requisitos. Posteriormente se evalúan los resultados usando las métricas de evaluación recisión, cobertura y medida-F.

3.2 Métricas de evaluación

Las métricas seleccionadas para la evaluación de la propuesta son precisión, cobertura y medida-F. La mayoría de los métodos reportados en el estado del arte hacen uso de estas métricas para evaluar sus resultados, esto permite comparar los resultados del método propuesto con los reportados por otras soluciones reportadas.

Precisión (P) permite evaluar con que precisión los requisitos extraídas se pueden tomar realmente como los adecuados. La precisión brinda la proporción de requisitos funcionales extraídos correctamente (*requisitos_extraidos_correctos*) del total de los requisitos extraídos, y se calcula como se muestra en la fórmula:

$$P = \frac{requisitos\_extraidos\_correctos}{requisitos\_extraidos} * 100$$

Cobertura (C) permite evaluar la medida en la que se cubren los requisitos extraídos automáticamente en comparación con los requisitos identificados manualmente (*requisitos_correctos*) y se calcula como se muestra en la fórmula:

$$C = \frac{requisitos\_extraidos\_correctos}{requisitos\_correctos} * 100$$

Medida-F (F) permite otorgarle una evaluación general a la propuesta a partir de las dos métricas definidas anteriormente. Un mayor valor de Medida-F significa un valor razonablemente mayor de la Precisión y la Cobertura, dado que se corresponde con la media harmónica de estas dos, y se calcula como se muestra en la fórmula:

$$F = 2 * \frac{P * C}{P + C}$$

3.3 Resultados y discusión

En esta sección se sintetizan y analizan los resultados obtenidos en los experimentos realizados con el dataset de casos de estudio para satisfacer los objetivos de evaluación planteados. Se muestran resultados aplicando la función de similitud sintáctica coseno y el agrupamiento por similitud.

Tabla 10 Resultados con Caso de estudio 1

Característica	**Resultados**		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	14.89	30.43	20.0
Similitud coseno >= 0.90 Agrupamiento por similitud	15.78	26.08	19.67

Tabla 11 Resultados con Caso de estudio 2

Característica	Resultados		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	17.24	50.0	25.64
Similitud coseno >= 0.90 Agrupamiento por similitud	18.18	40.0	25.0

Tabla 12 Resultados con Caso de estudio 3

Característica	Resultados		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	22.22	55.55	31.74
Similitud coseno >= 0.90 Agrupamiento por similitud	17.14	33.33	22.64

Tabla 13 Resultados con Caso de estudio 4

Característica	Resultados		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	4.65	10.0	6.34
Similitud coseno >= 0.90 Agrupamiento por similitud	5.71	10.0	7.27

Tabla 14 Resultados caso de estudio 5

Característica	Resultados		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	4.0	9.52	5.63
Similitud coseno >= 0.90 Agrupamiento por similitud	5.40	9.52	6.89

Tabla 15 Resultados caso de estudio 6

Característica	Resultados		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	14.00	36.84	20.28
Similitud coseno >= 0.90 Agrupamiento por similitud	10.81	21.05	14.28

Tabla 16 Resultados caso de estudio 7

Característica	Resultados		
	P	**C**	**F**
Similitud Levenshtein >= 0.90 Agrupamiento por similitud	8.16	14.28	10.38
Similitud coseno >= 0.90 Agrupamiento por similitud	8.23	12.5	9.92

Los resultados obtenidos muestran, en la mayoría de casos, valores relativamente bajos, resaltando el valor de la cobertura que se muestra alto en comparación con la precisión.

Conclusiones

Con la culminación del presente trabajo y el análisis de los resultados obtenidos se puede concluir que:

- El lenguaje natural constituye la principal fuente de información para la ingeniaría de requisitos, pero sus características dificultan el proceso ocasionando elevados valores en las estadísticas de fallos de los proyectos.
- Existen numerosos estándares y formalizaciones de requisitos, pero el lenguaje natural y la utilización de modelos de forma independientes o combinadas en una estructura son las más comunes.
- En el mercado se encuentran presentes varias herramientas de PLN como parte del proceso de IR, en este estudio se incluyeron solo aquellas a las cual se tuvo acceso por ser Open Source.
- Se justificó el uso del lenguaje de programación Python para implementar nuestra propuesta de solución, así como de las librerías de SpaCy y IDE PyCharm que es la plataforma líder para crear programas Python que funcionen con datos de lenguaje humano.
- Se demostró que la extracción de requisitos basada en patrones léxico sintáctico posee una alta cobertura muy bajo los valores de precisión.

Recomendaciones

Para darle continuidad a la investigación realizada se proponen las siguientes recomendaciones:

- Aplicar análisis de dependencias entre tokens a nivel de oración.

Referencias Bibliográficas

[1] S. Group, «Chaos Report,» 2018.

[2] E. S. Calisaya, *CONSTRUCCIÓN DE UNA HERRAMIENTA PARA EL ANÁLISIS DE REQUISITOS DE SOFTWARE DESCRITOS EN LENGUAJE NATURAL,* AREQUIPA – PERÚ: UNIVERSIDAD NACIONAL DE SAN AGUSTÍN DE AREQUIPA, 2019.

[3] B. Boehm, Software Engineering Economics, Prentice Hall: Englewood Cliffs, 1981.

[4] B. Boehm, «Verifying and Validating Software Requirements and Design Specifications,» *IEEE,* vol. 1, nº 1, p. 75–88, 1984.

[5] R. D. E. L. A. Van Lamsweerde, « Managing conflicts in goal-driven requirements engineering,» *IEEE transactions on Software engineering ,* p. 908–926, 1998.

[6] D. M. B. E. K. C. Denger, «Higher quality requirements specifications through natural language patterns,» *Proceedings 2003 Symposium on Security and Privacy,* p. 80–90, 2003.

[7] W. N. R. R. Vlas, «A rule-based natural language technique for requirements discovery and classification in open-source software development projects,» p. 1–10, 2011.

[8] K. Ryan, «The role of natural language in requirements engineering,» *In Proceedings of the IEEE International Symposium on Requirements Engineering,* p. 240–242, 1993.

[9] J. H. a. C. D. Manning, «Advances in natural language processing,» p. 261–266, 2015.

[10] G. Goth, «Deep or shallow, NLP is breaking out,» p. 13–16, 2016.

[11] F. D. A. E. V. G. a. S. G. A. Ferrari, «Natural language requirements processing: A 4D vision.,» p. 28–35, 2017.

[12] A. F. X. F. a. C. P. F. Dalpiaz, «Natural language processing for requirements engineering: The best is yet to come,» p. 115–119, 2018.

[13] D. M. Fernández, «Naming the pain in requirements engineering. Empir. Softw. Eng,» p. 2298–2338, 2017.

[14] I. Sommerville, Ingeniería de Software, Ciudad de México: Pearson Education, 2011.

[15] K. Pohl y C. Rupp, Requirements Engineering Fundamentals;, California, 2017.

[16] R. S. Pressman, Ingeniería del Software. Un enfoque Práctico. Séptima Edición, D. F., México, 2010.

[17] P. Kruchten, The Rational Unified Process: An Introduction, Addison-Wesley, 2001.

[18] International Organization for Standardization, *[ISO/IEC/IEEE 29148:2011] Systems and software engineering Life cycle processes – Requirements engineering,* Geneva, 2011.

[19] M. K. S. S. Mary Beth Chrissis, CMMI for Development, Guidelines for Process Integration and Product Improvement, Boston, MA: Addison-Wesley, Carnegie Mellon Software Engineering Institute (SEI) , 2011.

[20] R. Sharma, J. Bhatia y K. K. Biswas, Automated identification of business rules in requirements documents, 2014.

[21] E. Bagheri, F. Ensan y D. Gasevic, Decision support for the software product line domain engineering lifecycle, 2012.

[22] J. McZara, J. Sarkani, T. Holzer y T. Eveleigh, Software requirements prioritization and selection using linguistic tools and constraint solvers—a controlled experiment,, 2015.

[23] C. Duan, P. Laurent, J. Cleland-Huang y C. Kwiatkowski, Towards automated requirements prioritization and triage, 2009.

[24] J. Guo et al, Data-efficient performance learning for configurable systems, 2019.

[25] C. Fitzgerald, E. Letier y A. Finkelstein, Early failure prediction in feature request management systems: an extended study, 2012.

[26] J. del Sagrado y I. M. del Águila, Stability prediction of the software requirements specification, 2017.

[27] I. M. d. Águila y J. D. Sagrado, Requirement risk level forecast using Bayesian networks classifiers, 2011.

[28] Z. Kurtanović y W. Maalej, Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning, 2017.

[29] A. Dekhtyar y V. Fong, RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow, 2017.

[30] Z. Li, M. Chen, L. Huang, V. Ng y R. Geng, Tracing requirements in software design, 2017.

[31] C. R. a. C. Proix, «A natural language approach for requirements engineering,» *In Advanced Information Systems Engineering,* pp. Springer, 257–277, 1992.

[32] K. Ryan, « The role of natural language in requirements engineering,» *Proceedings of the IEEE International Symposium on Requirements Engineering,* pp. IEEE, 240–242. , 1993.

[33] R. J. A. a. D. K. Moorhead, «Software requirements and specifications: A survey of needs and languages,» 1981.

[34] F. M. a. N. I. P. L. Mich, « Market research for requirements analysis using linguistic tools,» p. 40–56, 2004.

[35] C. M.Kassab, «State of practice in requirements engineering: contemporary data,» *Innovat. Syst. Softw. Eng,* p. 235–241, 2014.

[36] D. Jurafsky, « Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition,» 2000.

[37] V. Teller, « Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition,» 2000.

[38] W. A. A. F. K. J. L. M. A. A. E.-V. C. R. T. B.-N. L. Zhao, « Natural language processing for requirements engineering: A systematic mapping study,» *ACM Computing Surveys* .

[39] D. S. C. F. I. K. Raharjana, «User stories and natural language processing: A systematic literature review,» 2021.

[40] G. P. A. R. Amna, «Ambiguity in user stories: A systematic literature review,» *Information and Software Technology,* 2022.

[41] J. V. I. I. B. J. C. S. I. P. B. D. Dermeval, «A systematic review on the use of ontologies in requirements engineering,» *Brazilian Symposium on Software Engineering,* 2014 .

[42] W. H. B. M. W. A. M. A. K. K. F. Nazir, «The applications of natural language processing (nlp) for software requirement engineering-a systematic literature review,» *International conference on information science and applications,* 2017.

[43] ¨. A. D. K. F. Bozyi˘git, «Linking software requirements and conceptual models: A systematic literature review,» *Engineering Science and Technology, an International Journal ,* 2021.

[44] P. B. ShiraWein, «A Fully Automated Approachto Requirement Extraction from Design Documents».

[45] I. K. L. &. O. Hussain, «Using Linguistic Knowledge to Classify Nonfunctional Requirements in SRS documents,» *Lecture Notes in Computer Science ,* pp. 287-298, 2008.

[46] C. &. S. Rolland, «Supporting Requirements Elicitation through Goal/Scenario Coupling,» *Springer,* 2009.

[47] S. &. J. Murugesh, «Construction of Ontology for Software Requirements Elicitation. J. Agric,» *Scien,ce Eng,* 2015.

[48] U. P. S. &. J. Shah, «Specification of non-functional requirements: A hybrid approach,» *In 22nd International Working Conference on Requirements Engineering. Gothenburg, Sweden,* 2016.

[49] H. M. A. &. E. Meth, «Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation,» *Lecture Notes in Computer Science,* 2013.

[50] Lili, «Research on User Requirements Elicitation Using Text Association Rule,» *In International Symposium on Intelligence Information Processing and Trusted Computing. Huanggang, China,* 2010.

[51] A. C. M. A. Lorena Talamé, «Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python,» *ASAI, Simposio Argentino de Inteligencia Artificial.*

[52] Cynoteck, «Cynoteck,» 2022. [En línea]. Available: https://cynoteck.com/es/blog-post/flask-vs-django/. [Último acceso: 2022].