**EXTRACCIÓN AUTOMÁTICA DE REQUISITOS DE SOFTWARE A PARTIR DE INFORMACIÓN TEXTUAL**

**Primer Autor (nombres y apellidos)1, Segundo Autor (nombres y apellidos)2**

1Filiación, dirección, 2Filiación, dirección

1e-mail:

**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. En el trabajo se presenta un método para la extracción automática de requisitos de software, a partir de información textual no estructurada. El método propuesto se enfoca en el análisis sintáctico basado en patrones léxicos – sintácticos. 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.

**PALABRAS CLAVES:** Captura de requisitos, extracción automática de requisitos, procesamiento de lenguaje natural.

**AUTOMATIC EXTRACTION OF SOFTWARE REQUIREMENTS FROM TEXTUAL INFORMATION**

**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. In the work, a method for the automatic extraction of software requirements, from unstructured textual information, is presented. The proposed method focuses on syntactic analysis based on lexical-syntactic patterns. 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.

**KEY WORDS:** Requirements capture, automatic requirements extraction, natural language processing.

1. **INTRODUCCIÓN**

La Ingeniería de Requisitos (RE) constituye una de las fases más importantes en el desarrollo de proyectos software, ya que su ciclo de desarrollo está basado fundamentalmente en los requisitos. La ocurrencia de errores en los requisitos puede provocar numerosas consecuencias, tales como: retardo en la terminación del proyecto, incremento de los costos, no satisfacción de los solicitantes, entre otros [1].

Entre las fases que se llevan en la Ingeniería de Requisitos, la captura de requisitos es la fase más importante y crítica en el proceso, debido al alto consumo de tiempo que se requiere para su ejecución y el impacto negativo que pueden tener sus resultados en el producto final [2] [3], y porque la mayor parte de los sistemas fallan debido a errores en este proceso de captura [4]. Para lograr un producto software de calidad los requisitos deben satisfacer varias características y cumplir ciertos criterios. Además, un requisito debería ser completo, correcto, realizable, necesario, priorizable, no ambiguo y verificable [5]. La delimitación del alcance del proyecto constituye otro de los elementos críticos de la captura requisitos [6], por lo que, con el objetivo de reducir los problemas en este proceso se han definido numerosas técnicas y herramientas.

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 [7]. 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 del lenguaje natural [8]. 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 [9].

Se han llevado a cabo muchas investigaciones sobre la automatización de diferentes tareas de RE. 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.

Este trabajo se centra en la automatización de la captura de requisitos apoyándose en herramientas y técnicas de procesamiento de lenguaje natural.

La estructura del resto del trabajo se organiza de la siguiente forma: Sección 2 Procesamiento del lenguaje natural aplicado a la Ingeniería de Requisitos. Sección 3 Extracción automática de requisitos candidatos. Sección 4 Resultados y discusión.

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 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 NLP y RE.

Tales como la identificación de defectos de calidad y ambigüedad, clasificación y agrupación de grandes colecciones de requisitos, extracción de abstracciones clave, generación de modelos y trazabilidad entre los requisitos de lenguaje natural (NL) [10].

Hasta hace poco, muchas de estas aplicaciones de la NLP se han limitado al mundo académico debido a la inaccesibilidad de las herramientas de la NLP y la pronunciada curva de aprendizaje. Afortunadamente, los avances en el aprendizaje profundo y la disponibilidad de grandes corpus de NL han reducido significativamente las barreras de entrada al uso de NLP. Esto crea oportunidades sin precedentes para aplicar técnicas de NLP a la práctica de RE y ayudar a analizar automáticamente los documentos relacionados con los requisitos [10].

Centrándose específicamente en la extracción automática de requisitos de software:

La solución reportada por [11] describe: Cargar el diseño del documento y analizarlo en oraciones. Dentro de las oraciones, también convertir las oraciones en palabras individuales y recuperar las etiquetas de parte del discurso para cada palabra. Se probaron 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. Se verificó la ambigüedad semántica, si existe, se construye el árbol estructural de la oración para identificar el sujeto de la oración anterior: se desarrolló y definió una gramática y se analiza la oración en un árbol de acuerdo con esta gramática; el sujeto se identifica como el sintagma nominal principal en el árbol.

En [12] la metodología destinada es a mejorar la detección de los requisitos no funcionales (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 [13] el procedimiento que 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 [14] 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 [15], 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).

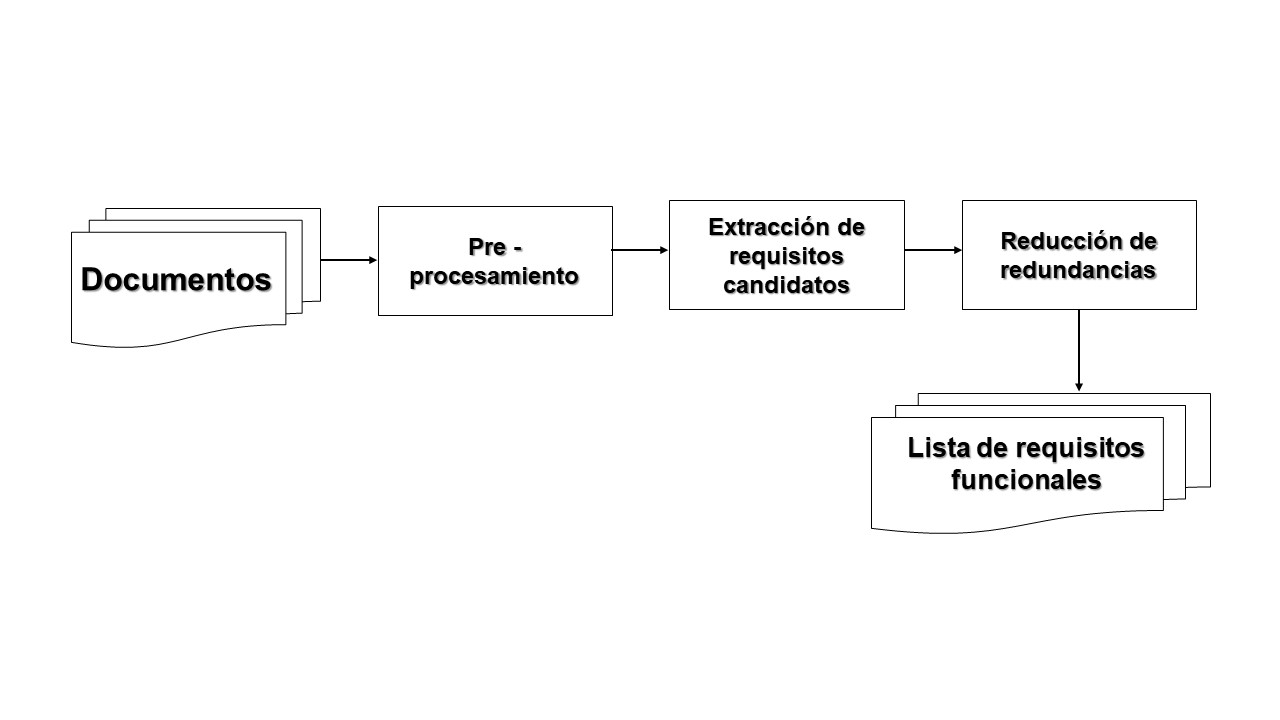
Este trabajo [16] explora cómo la cantidad y el tipo de conocimiento afectan la calidad de obtención de requisitos en dos simulaciones consecutivas. Se calcula con las salidas del automatismo con el patrón oro introducido en la última palabra. La recuperación puede verse como una medida de completitud, comparando el número de requisitos identificados con el número total de requisitos existentes en un documento.

El documento [17] propone un método para obtener los requisitos del usuario en la 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.

Como conclusión de este análisis se identificaron varias limitaciones, tales como, la no existencia de disminución de redundancias entre las frases extraídas, solo se procesan textos en idioma inglés.

1. **EXTRACCIÓN AUTOMÁTICA DE REQUISITOS CANDIDATOS**

En esta sección se describe la solución propuesta en el trabajo. Las principales etapas del proceso de desarrollo se describen en la Fig. 1. 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 macheo con un conjunto de 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 para ser grupadas en conjuntos según el 90% de similitud entre ellas. Finalmente se exporta un archivo con los resultados obtenidos.



**Fig. 1** Flujo de trabajo de la solución propuesta.

**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. 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 PLN, 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.

**Extracción de requisitos candidatos**

El proceso de extracción de requisitos candidatos se lleva a cabo a partir del uso de patrones léxico-sintácticos. Los patrones léxicos – sintácticos son la secuencia de caracteres que se forman a partir de la clasificación gramatical de cada token. 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 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 secuencia de la etiqueta gramatical de cada tokens fueron formados los patrones. Las etiquetas[[1]](#footnote-1) que forman el conjunto de patrones son las usada por el propio analizador sintáctico del módulo para el idioma español de SpaCy.

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 funcionales.

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

**Tabla 1** Patrones léxicos – 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 |
| 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 |

**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[[2]](#footnote-2) : La distancia de [Levenshtein](https://es.wikipedia.org/wiki/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, además de ser capaz de comparar cadenas de diferente tamaño.

Métrica Similitud de coseno[[3]](#footnote-3): En [el análisis de datos](https://en.wikipedia.org/wiki/Data_analysis) , la similitud del coseno es una [medida de similitud](https://en.wikipedia.org/wiki/Measure_of_similarity) entre dos secuencias de números. Para definirlo, las sucesiones se ven como vectores en un [espacio de producto interior](https://en.wikipedia.org/wiki/Inner_product_space), y la similitud de coseno se define como el [coseno](https://en.wikipedia.org/wiki/Cosine) del ángulo entre ellos, es decir, el [producto escalar](https://en.wikipedia.org/wiki/Dot_product) 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 subgrupos y se devuelve una lista general con cada uno estos subgrupos.

1. **RESULTADOS Y DISCUSIÓN**

El método propuesto fue desarrollado en el lenguaje de programación Python[[4]](#footnote-4), 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 entorno de desarrollo integrado (IDE) PyCharm[[5]](#footnote-5) por su gran rendimiento y flexibilidad, lo que permite un desarrollo ágil y reutilizable. La herramienta de procesamiento de lenguaje natural SpaCy[[6]](#footnote-6) se utiliza por su fácil acceso y todo el potencial que posee.

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 clasificación de textos llamadas Precisión (P), Cobertura (C) y Medida-F (F).

**Tabla 2** Descripción de casos de estudio

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de estudio** | **Características** | | |
| **Palabras** | **Requisitos primarios** | **Dominio** |
| Caso de estudio 1 | 614 | 17 | seguridad |
| Caso de estudio 2 | 629 | 17 | seguridad |
| Caso de estudio 3 | 503 | 18 | decoración |
| Caso de estudio 4 | 614 | 17 | construcción |
| Caso de estudio 5 | 583 | 18 | telecomunicaciones |
| Caso de estudio 6 | 593 | 18 | administración |
| Caso de estudio 7 | 1074 | 54 | docencia |

**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:

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:

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:

Los resultados obtenidos en los experimentos realizados con la colección de casos de estudio para satisfacer los objetivos de evaluación planteados se muestran a continuación.

**Tabla 3** Resultados

|  |  |  |  |
| --- | --- | --- | --- |
| **Casos de estudio** | **Resultados** | | |
| **Precisión** | **Cobertura** | **Medida F** |
| Caso de estudio 1 | 15.78 | 26.08 | 19.67 |
| Caso de estudio 2 | 18.18 | 40.0 | 25.0 |
| Caso de estudio 3 | 17.14 | 33.33 | 22.64 |
| Caso de estudio 4 | 5.71 | 10.0 | 7.27 |
| Caso de estudio 5 | 5.40 | 9.52 | 6.89 |
| Caso de estudio 6 | 10.81 | 21.05 | 14.28 |
| Caso de estudio 7 | 8.23 | 12.5 | 9.92 |
| Promedio | 11.60 | 21.78 | 15.09 |

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**

Se llevó a cabo la extracción automática de requisitos, a partir de un método que se enfoca en el análisis sintáctico del texto, basado en patrones léxicos sintácticos.

**REFERENCIAS**

|  |  |
| --- | --- |
| [1] | S. Khan, A. B. Dulloo y M. Verma, «Systematic Review of Requirement Elicitation Techniques,» *International Journal of Information and Computation Technology,* vol. 4, nº 2 , pp. 133-138 , 2014. |
| [2] | H. Metha, M. Brhelb y A. Maedcheab, «The state of the art in automated requirements elicitation.,» *Information and Software Technology,* vol. 55, pp. 1695-1709, 2013. |
| [3] | M. A. Abbasi, J. Jabeen, Y. Hafeez, D.-e.-B. Batool y N. Fareen, «Assessment of Requirement Elicitation Tools and Techniques by Various Parameters,» *Software Engineering,* vol. 3, nº 2, pp. 7-11, 2015. |
| [4] | N. Garg, P. Agarwal y S. Khan, «Recent advancements in requirement elicitation and prioritization techniques,» de *2015 International Conference on Advances in Computer Engineering and Applications*, Ghaziabad, India, 2015. |
| [5] | A. T. Lazo y J. G. G. Botero, «Especificación de requisitos de software: Una mirada desde la revisión teórica de antecedentes,» *Entre ciencia e ingeniería,* vol. 10, nº 19, pp. 249-261, 2016. |
| [6] | P. Bourque, R. Dupuis, A. Abran, J. Moore y L. Tripp, «Guide to the Software Engineering - Body of Knowledge,» *IEEE Software,* vol. 16, nº 6, pp. 35-44, 2014. |
| [7] | A. Lamsweerde, R. Darimont y E. Letier, «Managing conflicts in goal-driven requirements engineering,» *IEEE Transactions on Software Engineering,* vol. 24, nº 11, p. 908–926, 1998. |
| [8] | C. Denger, D. Berry y E. Kamsties, «Higher quality requirements specifications through natural language patterns,» de *Proceedings 2003 Symposium on Security and Privacy*, Herzlia, Israel, 2003. |
| [9] | R. Vlas y W. N. Robinson, «A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects,» de *2011 44th Hawaii International Conference on System Sciences*, Kauai, HI, USA, 2011. |
| [10] | F. Dalpiaz, A. Ferrari, X. Franch y C. Palomares, «Natural Language Processing for Requirements Engineering,» *IEEE Software,* vol. 35, pp. 115 - 119, 2018. |
| [11] | W. Shira y B. Paul, «A Fully Automated Approach to Requirement Extraction from Design Documents,» de *2021 IEEE Aerospace Conference* , Big Sky, MT, USA, 2021. |
| [12] | I. Hussain, L. Kosseim y O. Ormandjieva, «Using Linguistic Knowledge to Classify Non-functional Requirements in SRS documents,» *Lecture Notes in Computer Science,* vol. 5039, pp. 287-298, 2008. |
| [13] | C. Rolland y C. Salinesi, «Supporting Requirements Elicitation through Goal/Scenario Coupling,» *Lecture Notes in Computer Science,* vol. 5600, p. 398–416, 2009. |
| [14] | S. &. J. Murugesh, «Construction of Ontology for Software Requirements Elicitation. J. Agric,» *Scien,ce Eng,* 2015. |
| [15] | U. S. Shah, S. J. Patel y D. C. Jinwala, «Specification of Non-Functional Requirements: A Hybrid Approach,» 2016. |
| [16] | H. Meth, A. Maedche y M. Einoeder, «Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation,» de *Lecture Notes in Computer Science* , 2013. |
| [17] | L. Dong, X. Zhang, N. Ye y X. Wan, «Research on User Requirements Elicitation Using Text Association Rule,» de *2010 International Symposium on Intelligence Information Processing and Trusted Computing*, Huanggang, China, 2010. |

1. https://datapeaker.com/big-data/regresion-lineal-introduccion-a-la-regresion-lineal-para-la-ciencia-de-datos/ [↑](#footnote-ref-1)
2. https://soldai.com/blog/inteligencia-artificial/inteligencia-artificial-inteligencia-artificial/metricas-operaciones-edicion/ [↑](#footnote-ref-2)
3. https://www.grapheverywhere.com/algoritmo-de-similitud-de-coseno/ [↑](#footnote-ref-3)
4. https://www.python.org/ [↑](#footnote-ref-4)
5. https://www.jetbrains.com/help/pycharm/quick-start-guide.html [↑](#footnote-ref-5)
6. https://spacy.io/ [↑](#footnote-ref-6)