REVISTA CUBANA
DE TRANSFORMACIÓN DIGITAL

RNPS 2487 • ISSN 2708-3411
Vol. 4 • Nro. 1 • enero-marzo 2023 • e203

ARTÍCULO ORIGINAL

Método de extracción automática de requisitos de *software* a partir de información textual no estructurada

Method of Automatic Extraction of Software Requirements from Non-Structured Textual Information

Amanda Hernández Carreras
ahernandezc@ceis.cujae.edu.cu • https://orcid.org/0000-0002-6776-9434

Alfredo Simón Cuevas
asimon@ceis.cujae.edu.cu • https://orcid.org/0000-0001-9648-6209

UNIVERSIDAD TECNOLÓGICA DE LA HABANA "JOSÉ ANTONIO ECHEVERRÍA", CUJAE, CUBA

Anaisa Hernández González
anaisa@ceis.cujae.edu.cu • https://orcid.org/00000-0003-1169-301X

UNIVERSIDAD DE GRANADA, ESPAÑA

Recibido: 2023-01-19 • Aceptado: 2023-02-26

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 es un área de investigación y desarrollo que busca aplicar técnicas, herramientas y recursos de procesamiento del lenguaje natural (PLN) al proceso de ingeniería de requisitos (RE), para colaborar con los analistas humanos en la realización de 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 apoyado en patrones léxico-sintácticos, análisis de dependencias y un enfoque basado en la combinación de ambas técnicas de educción. Las métricas de precisión, cobertura y Medida-F fueron computadas, comparando el requisito que se obtuvo con

el elaborado manualmente por el experto. En esta comparación se empleó la distancia Levenshtein, usando como umbral de aceptación el 60 %. Los resultados demuestran relevancia en el valor de la precisión, por parte de la técnica de extracción basada en patrones, así como en la cobertura y Medida-F, para la solución que integra ambas técnicas de extracción de información.

Palabras clave: captura de requisitos, extracción automática de requisitos, procesamiento de lenguaje natural.

ABSTRACT

Obtaining requirements is one of the most important and critical phases in software development, due to the influence of its results on the success of the 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 investigations focused on its automation. Natural Language Processing for Requirements Engineering (PLNRE) is an area of research and development that seeks to apply Natural Language Processing (PLN) 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, was presented. The proposed method focuses on syntactic analysis based on lexical-syntactic patterns, on dependency analysis and an approach based on the combination of both eduction techniques. The Precision, Coverage and Measure-F metrics were computed by comparing the requirement that was obtained, with the one elaborated manually by the expert. In this comparison, the Levenshtein distance was used, using 60% as the acceptance threshold. The results obtained demonstrate a relevance in the value of precision by the pattern-based extraction technique, as well as in the coverage and F-measure for the solution that integrates both information extraction techniques.

Keywords: *requirements capture, automatic requirements extraction, natural language processing.*

INTRODUCCIÓN

La Ingeniería de Requisitos (IR) constituye una de las etapas más importantes en el desarrollo de proyectos de *software*, ya que su ciclo de desarrollo está basado fundamentalmente en

cómo se capturan, diseñan, implementan, prueban y despliegan los requisitos. La ocurrencia de errores en los requisitos especificados puede provocar numerosas consecuencias, como: retardo en la terminación del proyecto, incremento de los costos e insatisfacción de los solicitantes, entre otros (Shadab, 2014).

Entre las actividades 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 (Hendrik, 2013) (Abbasi *et al.,* 2015), y porque la mayor parte de los sistemas fallan debido a errores en este proceso de captura (Garg, Agarwal, & Khan, 2015). 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 (Alonso, 2016). La delimitación del alcance del proyecto constituye otro de los elementos críticos de la captura de requisitos (Bourque, Dupuis, Abran, Moore, & Tripp, 2014), por lo que con el objetivo de reducir los problemas 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, por rejemplo, la interferencia, la interdependencia y la incoherencia (Lamsweerde, Darimont, & Letier, 1998). Además, los requisitos generalmente se especifican en lenguaje natural, lo que aumenta la complejidad de la ingeniería de requisitos, debido a la ambigüedad inherente, la incompletitud y la inexactitud del lenguaje natural (Denger, Berry, & Kamsties, 2003). Estos factores hacen que las tareas de IR sean desafiantes, lentas y propensas a errores, principalmente para proyectos grandes, ya que es necesario procesar, analizar y comprender grandes volúmenes de requisitos (Vlas & Robinson, 2011).

Se han llevado a cabo muchas investigaciones sobre la automatización de diferentes tareas de RE (Dalpiaz, Ferrari, Franch, & Palomares, 2018). Los enfoques propuestos generalmente comienzan aplicando un conjunto de pasos de Procesamiento del Lenguaje Natural (PLN), que extraen información y características lingüísticas de los textos de requisitos y construyen varias representaciones basadas en PLN. Este trabajo se centra en la automatización de la extracción de requisitos, apoyándose en herramientas y técnicas de procesamiento de lenguaje natural.

En Dalpiaz, Ferrari, Franch, & Palomares (2018) se analizan varias publicaciones sobre la relación entre las tareas de PLN y IR, y se encontraron trabajos que se centran en 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).

En Hussain, Kosseim, & Ormandjieva (2008), la metodología destinada a mejorar la detección de los requisitos no funcionales (RNF) en los documentos de requisitos, usa el Stanford Parser para derivar morfológicamente las palabras y extraer cinco características sintácticas de cada una de las instancias de entrenamiento (oraciones) del corpus.

En Rolland & Salinesi (2009), el procedimiento que se desarrolla es el enfoque L'Ecritoire (Rolland, Souveyet & Ben-Achour, 1998), con una relación 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 son los que producen un 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 Murugesh & Jaya (2015), una tarea importante para lograr este objetivo es construir una ontología consistente 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 de este.

En el documento descrito en Shah, Patel, & Jinwala (2016), se propone un enfoque semiautomático llamado RNF-Specifier, cuyo objetivo es generar especificaciones precisas, a partir de requisitos informales, incluidos los RNF. El enfoque consta de cinco módulos, a saber, preprocesamiento, resolución de ambigüedades, formación de ontologías, generación de diagramas UML y clasificación de RNF. Inicialmente, el ingeniero de requisitos recopila el conocimiento del dominio de los usuarios por medio de varios enfoques de comunicación: cuestionarios, entrevistas, lista de chequeo, 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 (o sea, diagramas de caso de uso, de clase, de secuencia).

En Meth, Maedche, & Einoeder (2013) se explora cómo la cantidad y el tipo de conocimiento afectan la calidad de obtención de requisitos en dos simulaciones consecutivas. 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 (Lili, 2010) 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 preprocesamiento 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.

El método de extracción propuesto por Pablo & Marcos (2012) consta de tres pasos organizados como una cadena de comandos en una canalización:

1. Análisis de dependencias: cada frase del texto de entrada se analiza mediante el analizador basado en dependencias DepPattern (Otero & González, 2011).
2. Cláusulas constituyentes: para cada oración analizada se descubren las cláusulas verbales que contiene, y luego para cada cláusula se identifican los participantes verbales, incluidas sus funciones: sujeto, objeto directo, atributo y complementos preposicionales.

3. Reglas de extracción: se aplica un conjunto de reglas sobre los componentes de la cláusula para extraer los triples objetivos.

Como conclusión de este análisis se identificaron varias limitaciones, como la inexistencia de disminución de redundancias entre las frases extraídas, el que no existe procesamiento para el idioma español y el poco uso de la técnica de extracción de información basada en el análisis de dependencias.

METODOLOGÍA

En esta investigación fue necesario definir el procedimiento que se debe seguir para describir el método de extracción automática de requisitos de *software* que se desarrolla (figura 1). El proceso inicia cuando el usuario carga un archivo en formato texto, no estructurado, ya sea una entrevista, una descripción de proceso, etc.; seguidamente el texto es preprocesado, con el objetivo de limpiar el texto de ruido y estandarizar para que pueda ser procesado computacionalmente. Luego se procede, a partir de ese texto preprocesado, a la extracción de los requisitos candidatos por dos vías, análisis sintáctico basado en patrones léxico-sintácticos y en el análisis de dependencias, mostrando así una solución final afirmada en la combinación de ambas técnicas de extracción de información. Estos requisitos extraídos son filtrados, haciendo uso de una estrategia de reducción de redundancias, que se enfoca en eliminar aquellas frases que sean exactamente iguales y aquellas que estén contenidas unas dentro de otras, seleccionando la frase más completa. Este conjunto de frases, luego de haber sido filtrado, se agrupa formando clústeres, a partir de la similitud semántica que exista por cada dos pares de frases. Estos clústeres son exportados al usuario en un archivo txt, concluyendo así el proceso de extracción automático de requisitos de *software*.

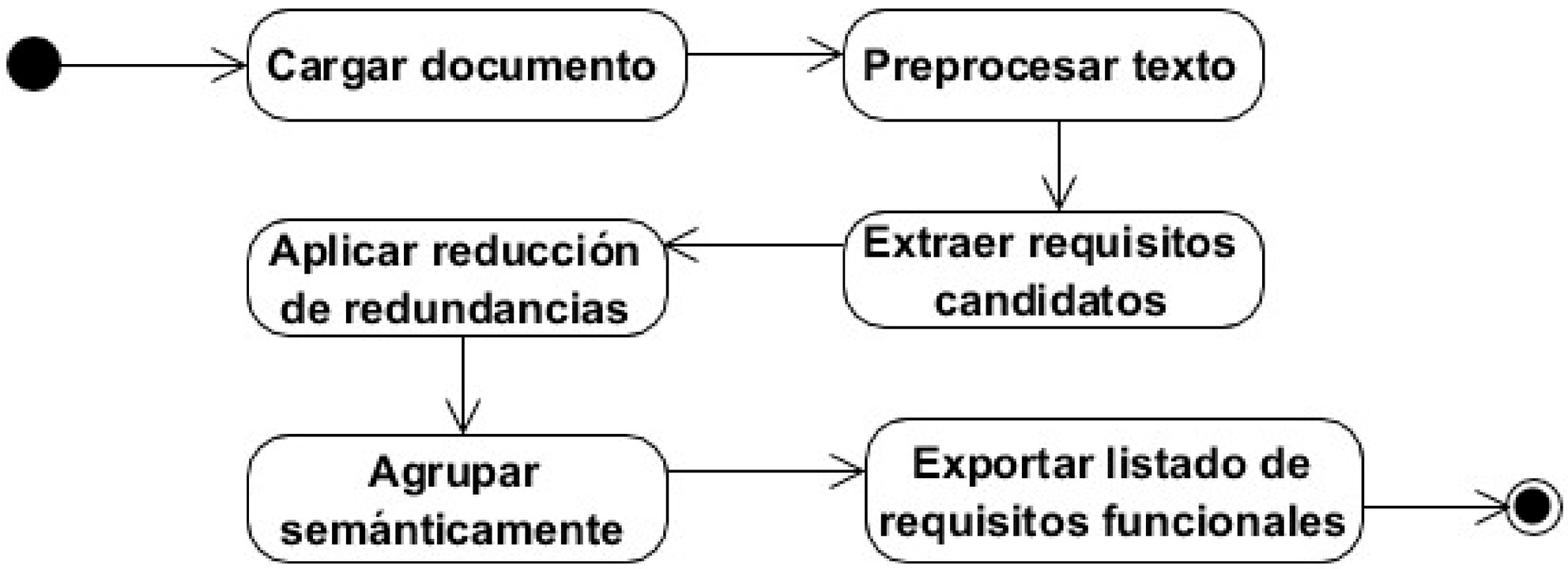

Fig. 1 Flujo de trabajo de la solución propuesta (Fuente: elaboración propia).

Preprocesamiento

Como se planteó en la subsección anterior, la solución contará fundamentalmente con un componente para la preparación o preprocesamiento de la información, haciendo uso de la biblioteca de Procesamiento de Lenguaje Natural para español, SpaCy (Altinok, 2021). El componente de preprocesamiento se encargará de limpiar y preparar los datos de texto, para que sea fácil de procesar. Algunos de estos pasos incluyen quitar las puntuaciones, transformar el texto a minúscula y segmentar en oraciones. En la figura 2 se muestra un diagrama de actividades del preprocesamiento, que inicia cuando el usuario carga un archivo en formato texto. Entre las principales actividades dentro del módulo de preprocesamiento están:

- Tokenización: esta actividad consiste en dividir el texto sin procesar en pequeños trozos. La tokenización dividirá el texto plano en palabras.
- Etiquetado PoS (*Part of speech*): se encarga de clasificar las partes de las oraciones en verbo, sustantivo, adjetivo, preposición, entre otras.
- Lematización: 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.

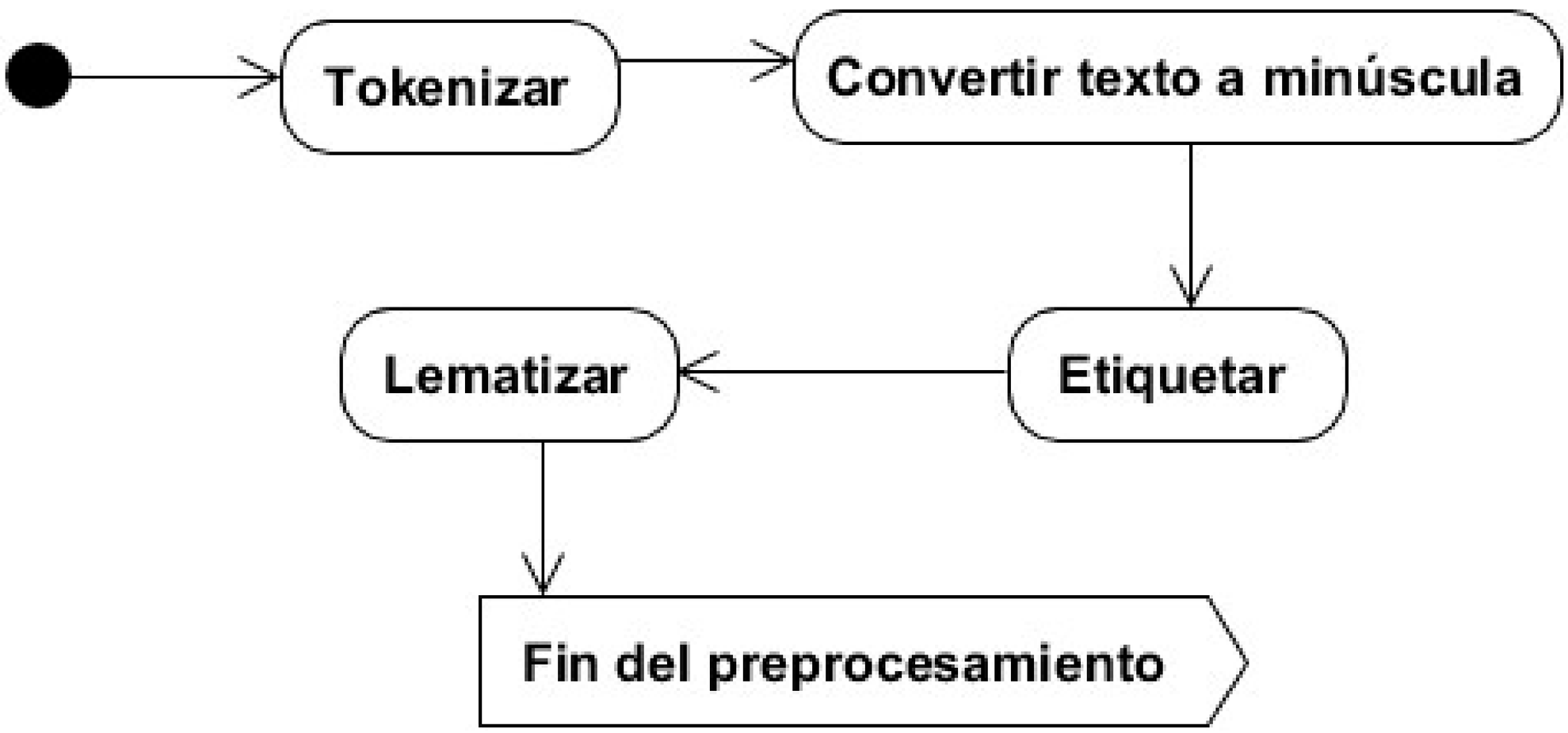

Fig. 2 Diagrama de actividades del preprocesamiento (Fuente: elaboración propia).

Extracción de requisitos candidatos

La extracción de requisitos candidatos se enfoca en el análisis sintáctico basado en patrones léxico-sintácticos y en el análisis de dependencias, además de una solución final afirmada en la concatenación de ambas técnicas de extracción de información. El flujo de este proceso se muestra en la figura 3.

El análisis sintáctico tiene como función etiquetar a 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 los sintagmas formados por cada una de las unidades léxicas que aparecen en la oración.

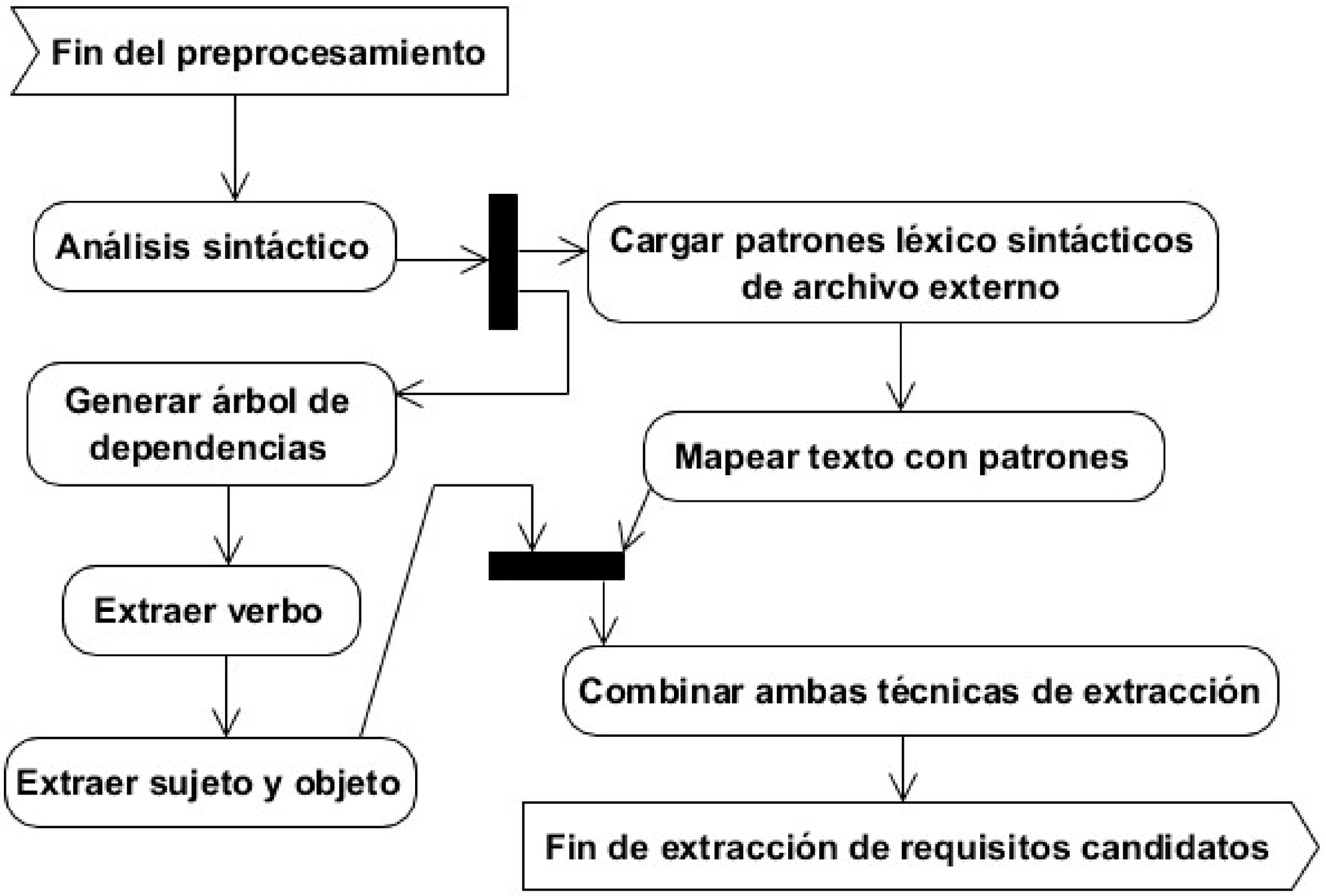

Fig. 3 Diagrama de actividades del módulo Extracción de requisitos candidatos (Fuente: elaboración propia).

Análisis sintáctico basado en patrones léxico-sintácticos

Un patrón es una descripción de la forma que pueden tomar los lexemas de un token. En este caso se define como patrón léxico-sintáctico, a aquella secuencia de etiquetas gramaticales que identifican la clasificación gramatical de cada uno de los tokens que conforman el sintagma que se va a extraer.

Para extraer las frases que dan lugar a los requisitos candidatos, se definieron previamente un conjunto de patrones léxico-sintácticos, que fueron concebidos a partir de un proceso estadístico realizado a 40 tesis del curso 2020-2021, de la Facultad de Ingeniería Informática, de la Universidad Tecnológica de La Habana José Antonio Echeverría (CUJAE), donde fueron tomados cada uno de los requisitos funcionales de los diagramas de 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. Finalizado el preprocesamiento 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 requisitos candidatos.

En la tabla 1 se muestra cada uno de estos patrones léxico-sintácticos, los cuales expresan la estructura de una frase, por ejemplo, «VERB NOUN» representa la frase «VERBO + SUSTANTIVO», tal como se aprecia en el ejemplo: «…analizar muestras…»

Tabla 1. Patrones léxico-sintáctico (Fuente: elaboración propia)

Patrón léxico-sintáctico	Ejemplos de frases
VERB NOUN	«…analizar muestras…»
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…»

Análisis de dependencias

La idea fundamental de dependencia está basada en que la estructura sintáctica de una frase consiste en relaciones binarias asimétricas entre las palabras de esa frase (Kübler, McDonald, & Nivre, 2009). Por tanto, han de establecerse criterios para definir qué relaciones de dependencia existen, para distinguir de qué forma están relacionadas dos palabras en una frase y si esas relaciones están etiquetadas o no. Por tanto, a partir de las relaciones asimétricas y los diferentes criterios, dada una frase en cualquier lenguaje, se puede establecer un árbol sintáctico de dependencias etiquetado, como el del ejemplo que se muestra en la figura 4. Este árbol fue generado por el analizador sintáctico de la biblioteca SpaCy ejecutado en Jupyter. En este ejemplo se observa el análisis de dependencias de la frase en español: «Las universidades consideradas cobran tarifas elevadas». Se puede razonar que dada esa frase y conociendo cierta información sintáctica de cada uno de los tokens que aparecen en esta, se pueden establecer distintas relaciones de dependencia, donde la raíz de todas es la acción principal de la frase, dígase el verbo. Con la información contenida en un árbol de dependencias se pueden realizar múltiples tareas, como simplificación de textos (Caseli, *et al.,* 2009), reconocimiento de la implicación textual (Herrera, Peñas, & Verdejo, 2005), detección de conceptos negados en una frase (Ballesteros, Martín, & Agudo, 2010), etcétera.

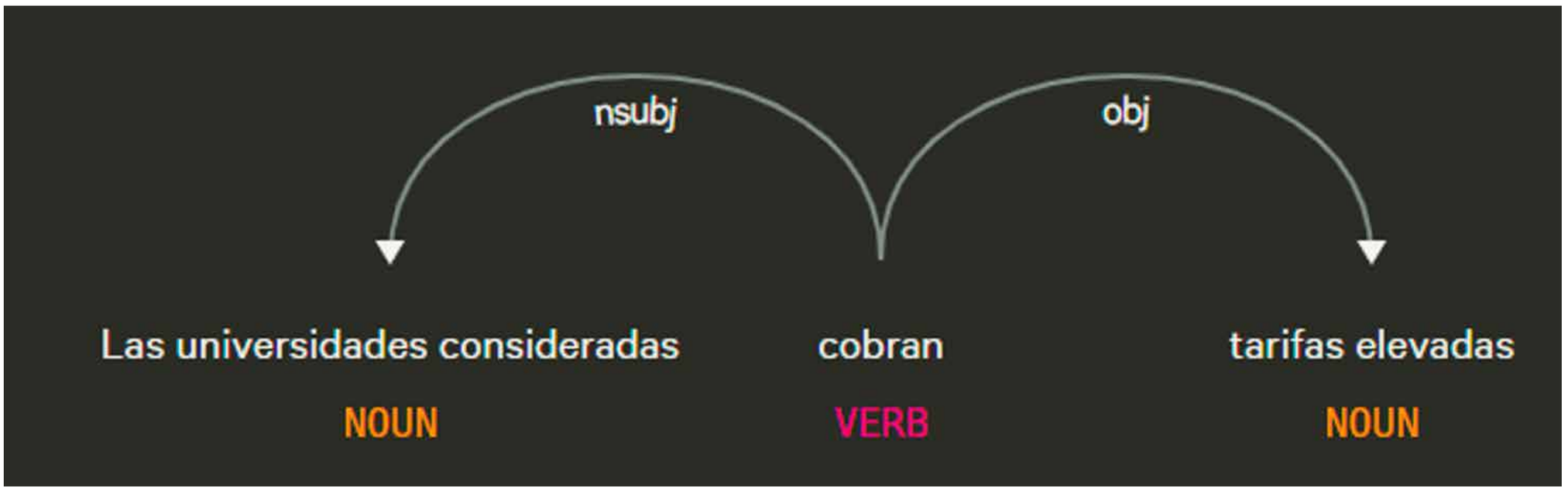

Fig. 4 Árbol de dependencias generado por el analizador sintáctico SpaCy (Fuente: elaboración propia).

En el trabajo se usará esta técnica de extracción de información como otra vía para la generación de los requisitos candidatos. Basándose en lo planteado por Pablo & Marcos (2012), se estará utilizando un patrón para extraer todas las frases con una misma estructura, la combinación del verbo en conjunto con el sujeto de la oración y el objeto al que se realiza la acción. El patrón se forma con las etiquetas de dependencias de la biblioteca SpaCy. Se recorre todo el árbol de dependencias generado por la biblioteca y se extraen las frases cumpliendo con esa estructura.

Reutilizando el ejemplo mencionado antes, la frase generada por este método de extracción quedaría como se muestra en la tabla 2.

Tabla 2. Ejemplo de extracción usando análisis de dependencias (Fuente: elaboración propia)

Patrón	Frase
VERB + nsubj + obj	cobran las universidades consideradas tarifas elevadas

Reducción de redundancias

Se incorpora al flujo de la solución una etapa de reducción de redundancias, como se muestra en la figura 5, con el objetivo de mejorar la calidad de los requisitos extraídos. Durante la primera prueba se observó que algunos requisitos extraídos aparecían con la siguiente forma: [‘incluir inyecciones’, ‘incluir inyecciones regulares’, ‘incluir inyecciones regulares de insulina’]. Como se observa en el ejemplo, el primer requisito está incompleto al ser comparado con el segundo, y a su vez, ambos están incompletos al ser comparados con el tercero. Es decir, desde una vista general de los resultados, los requisitos: [‘incluir inyecciones’, ‘incluir inyecciones regulares’] no son relevantes y pudieran ser eliminados para garantizar la reducción de reiteraciones en los requisitos extraídos, por lo que se estableció una estrategia de reducción de redundancias que satisfaga esta condición.

Esta estrategia de reducción de redundancias se basó en dos criterios:

1. Reducir a una, aquellas frases que hagan referencia a un mismo requisito, seleccionando de ellas la que abarque la mayor cantidad de información y descartando el resto, criterio que se corresponde con el ejemplo explicado anteriormente. La justificación de esta decisión está basada en el hecho de que hay patrones que son un subconjunto de otros.

2. 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 esta.

Agrupamiento

La fase de agrupamiento se incorpora a la solución de extracción de requisitos de *software*, luego que el conjunto de requisitos haya pasado por una estrategia de reducción de redundancias, como se muestra en la figura 5, con el objetivo de mejorar la visualización de los resultados para una mayor compresión de estos por el analista. Esta fase se encarga de agrupar en diferentes clústeres aquellas frases que tengan cierta similitud semántica. Para ello se usó la métrica de similitud semántica de WordNet Similarity (Miller *et al.,* 1990), Wu-Palmer y el algoritmo de agrupamiento Hierarchical Agglomerative. Para la evaluación de la calidad del agrupamiento se usó la métrica Sihlouette (Rousseeuw, 1987). Estas métricas se emplean debido a:

- WordNet Similarity: es un paquete de *software* disponible gratuitamente, que hace posible medir la similitud semántica y la relación entre un par de conceptos (o *synsets*). Proporciona seis medidas de similitud y tres medidas de relación, todas basadas en la base de datos léxica WordNet. Estas medidas se implementan como módulos de Perl que toman como entrada dos conceptos y devuelven un valor numérico que representa el grado en que son similares o relacionados. Tres de estas medidas de similitud se basan en longitudes de camino entre un par de conceptos: lch (Leacock & Chodorow, 1998), wup (Wu & Palmer, 1994) y path. lch encuentra el camino más corto entre dos conceptos y escala ese valor por la longitud máxima del camino que se encuentra en la jerarquía en la que ocurren. Wup encuentra la profundidad del subsumidor menos común (LCS) de los conceptos y luego la escala por la suma de las profundidades de los conceptos individuales. La profundidad de un concepto es simplemente su distancia al nodo raíz. La ruta de medida es una línea de base que es igual a la inversa de la ruta más corta entre dos conceptos (Pedersen, Patwardhan, & Michelizzi).
- El agrupamiento aglomerativo jerárquico es una técnica importante y bien establecida en el aprendizaje automático no supervisado. Los esquemas de agrupamiento aglomerativo comienzan con la partición del conjunto de datos en nodos únicos y fusionan paso a paso el par actual de nodos más cercanos entre sí en un nuevo nodo, hasta que queda un nodo final que comprende todo el conjunto de datos (Mullner, 2011).
- El valor de Sihlouette[1] (silueta) es una medida de cuán similar es un objeto a su propio grupo (cohesión) en comparación con otros grupos (separación). La silueta varía de –1 a +1, donde un valor alto indica que el objeto está bien emparejado con su propio grupo y

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

mal emparejado con los grupos vecinos. Si la mayoría de los objetos tienen un valor alto, la configuración de agrupación en clústeres es adecuada. Si muchos puntos tienen un valor bajo o negativo, es posible que la configuración de agrupación tenga demasiados o muy pocos clústeres.

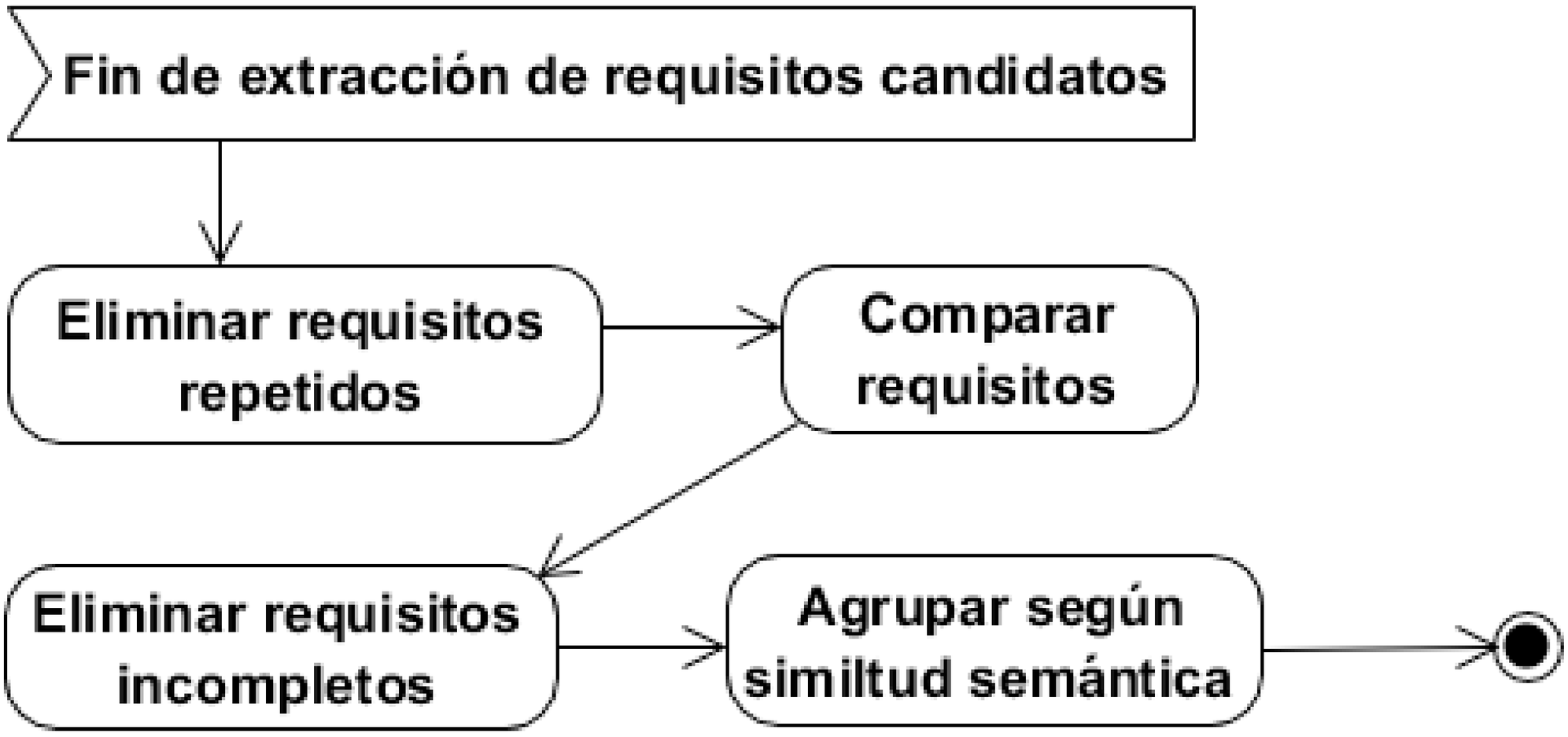

Fig. 5 Diagrama de actividades de los módulos Reducción de redundancias y Agrupamiento (Fuente: elaboración propia).

RESULTADOS Y DISCUSIÓN

En el estudio experimental de la solución propuesta se evaluaron los requisitos extraídos de manera automática, de un grupo de casos de estudio 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 estos, empleando las métricas habituales en la clasificación de texto, llamadas precisión (P), cobertura (C) y Medida-F (F).

Marco de evaluación

El proceso consiste en comparar sintácticamente los requisitos extraídos de forma manual con los requisitos extraídos automáticamente. Para ello se usó la métrica de similitud semántica Levenshtein, con un umbral de similitud de 60 %. Esta métrica evalúa la distancia entre dos cadenas y se usa cuando se quiere identificar que una cadena es o no la misma que otra. En este caso, estamos comparando cadenas obtenidas automáticamente contra las obtenidas por un experto.

Para llevar a cabo la evaluación se fijaron los siguientes objetivos:

- Analizar los resultados al aplicar cada una de las técnicas de extracción de información en cada uno de los casos de prueba.

- Evaluar el comportamiento de las diferentes técnicas de extracción de información con cada métrica.
- Evaluar el comportamiento promedio de los resultados para determinar cuál es la mejor de las técnicas para la extracción de requisitos de *software*.

Métricas de análisis de resultados

En la literatura revisada no se identificaron variantes para medir los resultados de este tipo de métodos, por tanto, como parte de este trabajo, se definieron un conjunto de métricas, tomando como base las comúnmente usadas en soluciones de extracción de información: precisión (P), cobertura (C) y Medida-F (F). Las métricas son computadas a partir de la comparación entre los requisitos que se extraen de forma automática, con los requisitos definidos para cada texto del conjunto de prueba.

- 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:

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

Tomando en consideración que los requisitos incluidos en la colección de prueba fueron elaborados manualmente y puede que no se expresen exactamente como aparece en el texto, se consideró utilizar un enfoque optimista en el proceso de comparación entre los requisitos. En concreto, se empleó la distancia Levenshtein para realizar la comparación entre los dos requisitos, usando como umbral de aceptación el 60 %.

Descripción de la colección de pruebas

La solución propuesta fue evaluada, tomando como referencia la extracción manual de requisitos de *software* de un conjunto de casos de pruebas, que divergen en dominios y abordan

diferentes temáticas, que se definieron previamente por el equipo de trabajo (tabla 3). Estos requisitos fueron comparados sintácticamente con los extraídos de forma automática, y se obtuvo una tercera lista de requisitos. Posteriormente se evalúan los resultados y se usan las métricas de evaluación recisión, cobertura y Medida-F.

Tabla 3. Descripción de casos de estudio (Fuente: elaboración propia)

Casos de prueba	Características			
	oraciones	palabras	requisitos	dominio
Sistema de seguridad vivienda	34	614	17	seguridad
Sistema de seguridad evento deportivo	34	629	17	seguridad
Decoración de interiores	22	503	18	decoración
Inversiones en telefonía fija	21	583	18	comunicación
Distribución de combustible	26	593	18	administración
Gestión cursos de posgrado	67	1 074	54	docencia
Gestión de reservas de casas de campo	23	607	18	recreación
Préstamo de libros	16	408	16	docencia
Préstamo de video	16	405	16	docencia
Salud mental	13	302	12	salud
Atención a niños	13	296	6	salud
Reserva de habitaciones de un hotel	71	2 342	25	turismo
Agencias de viajes	96	2 224	31	turismo
Sistema de control para una bomba de insulina	18	376	6	salud

Valoración de los resultados

Las métricas de precisión y cobertura son computadas, comparando el requisito que se obtiene al extraer información del texto, con el elaborado manualmente por el experto. En esa comparación se empleó la distancia Levenshtein y se usó como umbral de aceptación el 60 %. Los resultados de los experimentos realizados con la colección de prueba se muestran en la tabla 4.

Tabla 4. Resultados de la evaluación del método en colección de prueba (Fuente: elaboración propia)

	Usando patrones léxico-sintáctico			Usando análisis de dependencias			Extracción basada en enfoque híbrido		
	P	C	F	P	C	F	P	C	F
Entrevista 1	**30,55**	55	39,28	21,42	30	24,99	27,41	**85**	**41,46**
Entrevista 2	**33,33**	82,35	47,45	31,03	52,94	39,13	32,39	**100**	**52,27**
Entrevista 3	**28,57**	55,55	**37,73**	17,64	16,66	17,14	24,52	**72,22**	36,61
Entrevista 4	26,31	50	34,48	**31,57**	30	30,76	26,31	**75**	**38,96**
Entrevista 5	**28,94**	61,11	39,28	27,27	33,33	30	27,86	**94,44**	**43,03**
Entrevista 6	35,44	51,85	42,1	**47,45**	51,85	49,55	38,84	**100**	**55,5**
Descripción de proceso 1	**41,37**	34,28	**37,5**	4,54	2,85	3,5	26,53	**37,14**	30,95
Descripción de proceso 2	**59,25**	88,88	**71,11**	31,25	27,77	29,41	51,28	**100**	70,17
Descripción de proceso 3	**67,85**	**100**	82,6	50	44,44	47,05	63,41	**100**	**88,13**
Descripción de proceso 4	**17,39**	30,76	**22,22**	15.38	15,38	15,38	15,15	**38,46**	21,73
Descripción de proceso 5	16,66	26,66	20,51	23,07	20	21,42	**18,18**	**40**	**25**
Entrevista 7	23,95	92	38,01	21,12	60	31,24	**24,34**	**100**	**41,8**
Entrevista 8	**45,34**	**100**	**66,66**	17,7	54,83	26,77	31,01	**100**	51,85
Descripción de proceso 6	**20**	66,66	30,76	6,25	16,66	9,09	16,21	**100**	**31,81**
Promedio	**33,93**	**63,94**	**43,55**	**24,69**	**32,62**	**26,82**	**30,25**	**81,59**	**44,98**

De manera general:

- La mayor **precisión** la muestra el análisis basado en patrones léxicos sintácticos.
- El valor más elevado de **cobertura** lo expone el método que combina ambas técnicas de extracción de información.
- El mayor valor de la **Medida-F** también se aprecia en el enfoque híbrido.

Al analizar el comportamiento promedio de los resultados es posible notar que la técnica de extracción de información basada en patrones léxicos sintácticos, reporta el mejor comportamiento en cuanto a la medida **precisión**, lo que significa que esta técnica tuvo una mayor exactitud en el proceso de extracción de requisitos funcionales.
Sin embargo, las métricas de **cobertura** y **Medida-F** reportan mayores resultados con el método que integra ambas técnicas de extracción de información. Esto implica que, teniendo en cuenta la medida precisión, el método que combina ambas técnicas abarcó mayor cantidad de requisitos funcionales en el proceso de extracción.

CONCLUSIONES

En este trabajo se propuso una solución para la extracción automática de requisitos de *software*, a partir de información textual no estructurada. El diseño de solución que se formuló empleó dos técnicas de extracción de información, dígase análisis sintáctico basado en patrones léxicos-sintácticos y análisis de dependencias, así como una solución final basada en la combinación de estas dos técnicas. Se presentaron los resultados preliminares luego de ser realizada una evaluación a la solución, con la colección de pruebas confeccionada por el experto, que reúne entrevistas o descripciones de procesos y el conjunto requisitos funcionales extraídos manualmente, que permiten corroborar la precisión y cobertura de los resultados atendiendo a las 3 métricas computadas: **precisión, cobertura** y **Medida-F.** El análisis de los datos muestra que en promedio se obtienen mejores resultados con la técnica de extracción basada en patrones léxicos-sintácticos. Se muestra un mayor valor de presión con la técnica de patrones; sin embargo, el mayor valor de cobertura lo arroja la solución híbrida.

REFERENCIAS

Alonso Toro Lazo, J. G. (2016). Especificación de requisitos de *software*: Una mirada desde la revisión teórica de antecedentes. *Entre Ciencia e Ingeniería, 10*(19): 108-115.

Altinok, D. (2021). Mastering SpaCy. Birmingham: Packt Publishing Ltd.

Ballesteros, M., Martín, R., y Agudo, B. D. (2010). JadaWeb: A CBR System for Cooking Recipes. En Proceedings of Workshop on Computer Cooking Contest (ICCBR 2010). Italy, p. 179.

Bourque, P., Dupuis, R., Abran, A., Moore, J., y Tripp, L. (2014). Guide to the *Software* Engineering - Body of Knowledge. Recuperado de: http://www. swebok. org.

Caseli, H., Pereira, T., Specia, L., Pardo, T., Gasperin, C., y Aluisio, S. (2009). Building a Brazilian Portuguese parallel corpus of original and simplified texts. *Advances in Computational Linguistics, Research in Computer Science, 41*: 59-70.

Dalpiaz, F., Ferrari, A., Franch, X., y Palomares, C. (2018). Natural Language Processing for Requirements Engineering. *IEEE Software, 35*(5): 115-119.

Denger, C., Berry, D., y Kamsties, E. (2003). Higher quality requirements specifications through natural language patterns. Proceedings 2003 Symposium on Security and Privacy, pp. 80-90. IEEE.

Gamallo, P. y González, I. (2011). A gramatical formalism based on patterns of *Part of speech* tags. *International Journal in Corpus Linguistcs, 16*(19): 45-71.

Garg, N., Agarwal, P., y Khan, S. (2015). Recent advancements in requirement elicitation and prioritization techniques. 2015 International Conference on Advances in Computer Engineering and Applications, pp. 237-240, IEEE.

Hendrik Metha, M. B. (2013). The state of the art in automated requirements elicitation. *Requisitos_correctos*(10): 1695-1709.

Herrera, J., Peñas, A., y Verdejo, F. (2005). Textual Entailment Recognition Based on Dependency Analysis and WordNet. Part of the Lecture Notes in Computer Science book series, 3944, pp. 231-239.

Hussain, I., Kosseim, L., y Ormandjieva, O. (2008). Using Linguistic Knowledge to Classify Nonfunctional Requirements in SRS documents. Lecture Notes in Computer Science, 5039, pp. 287-298.

Kübler, S., McDonald, R., y Nivre, J. (2009). Dependency Parsing. *Synthesis lectures on human language technologies, 1*(1): 1-127.

Lamsweerde, A., Darimont, R., y Letier, E. (1998). Managing conflicts in goal-driven requirements engineering. *IEEE transactions on Software engineering, 24*(11): 908-926.

Leacock, C., y Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database, 49*(2): 265-283.

Lili. (2010). Research on User Requirements Elicitation Using Text Association Rule. 2010 International Symposium on Intelligence Information Processing and Trusted Computing, pp. 357-359, IEEE

Abbasi, M. A., Jabeen, J., Hafeez, Y., Batool, D., & Fareen, N. (2015). Assessment of Requirement Elicitation Tools and Techniques by Various Parameters. *Software Engineering, 3*(2): 7-11.

Meth, H., Maedche, A., y Einoeder, M. (2013). Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation. Advanced Information Systems Engineering: 25th International Conference, CAiSE 2013, Valencia, Spain, June 17-21, 2013. Proceedings 25, pp. 578-593, Springer Berlin Heidelberg.

Miller, G., Beckwith, R. Pellbaum, C., Gross, C. y Miller, C. (1990). Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography, 3*(4): 235-244.

Mullner, D. (2011). Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378.*

Murugesh, S., y Jaya, A. (2015). Construction of Ontology for *Software* Requirements Elicitation. *Indian Journal of Science and Technology, 8*(29).

Pablo, G., y Marcos, G. (2012). Dependency-Based Open Information Extraction. Proceedings of the joint workshop on unsupervised and semi-supervised learning in NLP, pp. 10-18.

Pedersen, T., Patwardhan, S., y Michelizzi, J. (2004). WordNet::Similarity-Measuring the Relatedness of Concepts. AAAI, vol. 4, pp. 25-29.

Rolland, C., y Salinesi, C. (2009). Supporting Requirements Elicitation through Goal/Scenario Coupling. Conceptual Modeling: Foundations and Applications, 5600, pp. 398-416.

Rolland, C., Souveyet, C. y Ben-Achour, C. (1998). Guiding goal modeling using scenarios. IEEE Transation *Software* Engineering, 24, pp. 1055-1071.

Rousseeuw, P. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster análisis. *Journal of Computational and Applied Mathematics, 20*: 53-65.

Shadab Khan, A. B. (2014). Systematic Review of Requirement Elicitation Techniques. *International Journal of Information and Computation Technology.* Indian.

Shah, U., Patel, S., y Jinwala, D. (2016). Specification of non-functional requirements: A hybrid approach. 22nd International Working Conference on Requirements Engineering. Gothenburg, Sweden.

Vlas, R., y Robinson, W. N. (2011). A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source *Software* Development Projects. 2011 44th Hawaii International Conference on System Sciences, pp. 1-10, IEEE.

Wu, y Palmer. (1994). Verb semantics and lexical selection. 32nd Annual Meeting of the Association for Computational Linguistics. Mexico. *arXiv preprint cmp-lg/9406033.*