****

Autor: Amanda Hernández Carreras ([ahernandezrr@ceis.cujae.edu.cu](mailto:ahernandezrr@ceis.cujae.edu.cu))

Tutores: Dr. C. Alfredo Simón Cuervas. ([asimon@ceis.cujae.edu.cu](mailto:asimon@ceis.cujae.edu.cu))

Dr. C. Anaisa Hernández González ([anaisa@ceis.cujae.edu.cu](mailto:anaisa@ceis.cujae.edu.cu))

*Extracción automática de requisitos software a partir de información textual no estructurada.*

Trabajo de Diploma

***Universidad Tecnológica de La Habana***

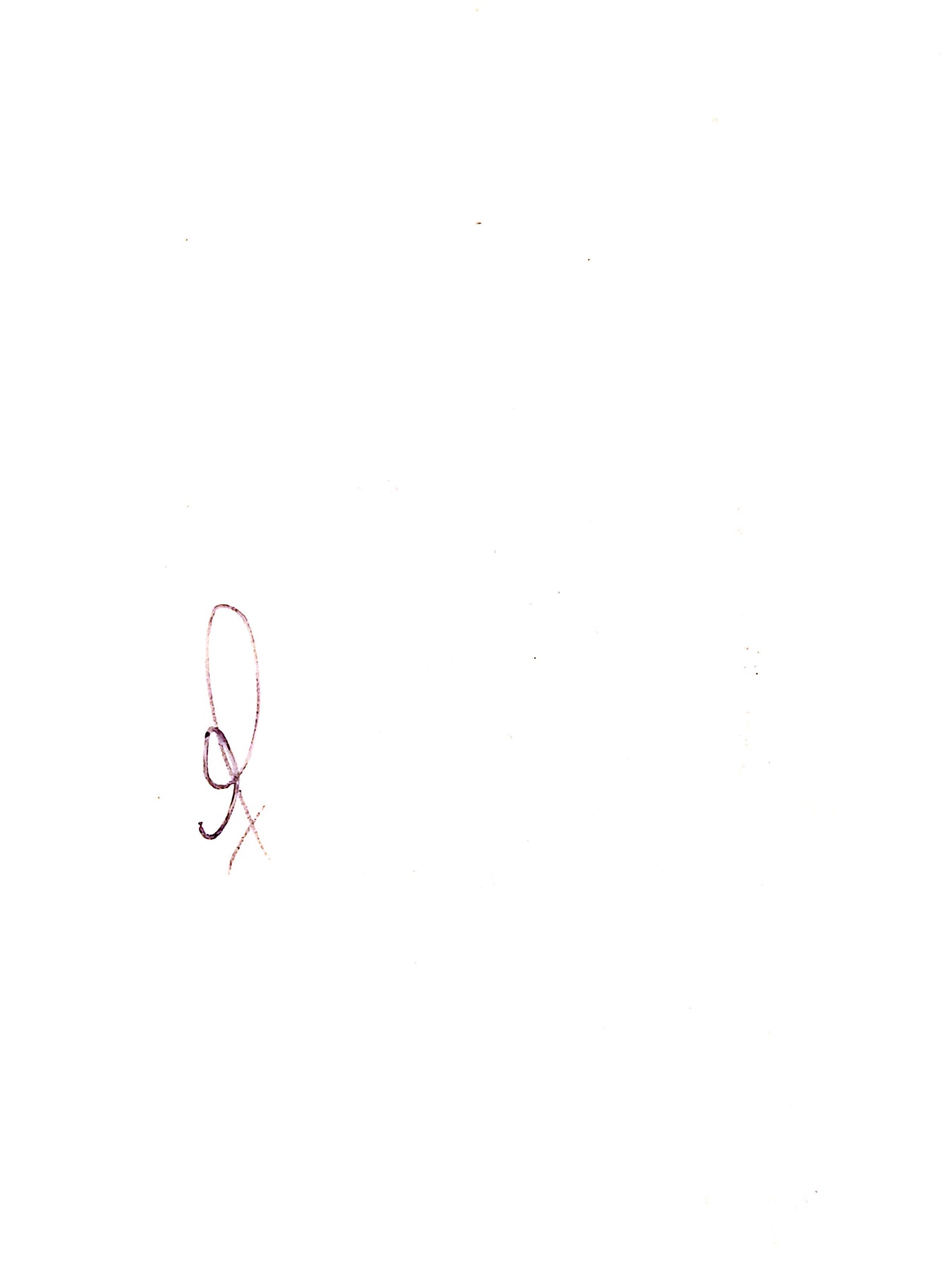
***“José Antonio Echeverría”***

Facultad de Ingeniería Informática

La Habana, Cuba 2022

# Declaración de autoría

Yo, Amanda Hernández Carreras, declaro que soy la única autora de esta tesis titulada “Extracción automática de requisitos de software a partir de información textual no estructurada” y autorizo a la Empresa CALISOFT y a la Facultad de Ingeniería Informática de la Universidad Tecnológica de La Habana José Antonio Echeverría, CUJAE, a que hagan el uso que estimen pertinente con este trabajo.

Para que así conste firmo a los 9 días del mes de diciembre del 2022.

# Opinión de tutores

# Agradecimientos

Hace 5 años atrás veía este momento muy lejos, y cada día me preguntaba cómo sería el instante en el que tenga que agradecerle a cada una de las personas que han formado parte de este trayecto, hoy, 5 años después me encuentro en la misma incertidumbre, pero bueno, comencemos.

La palabra “GRACIAS” se queda corta en comparación con todo lo que tengo que agradecerles a ustedes, a ustedes que confiaron y apostaron por mi desde de la pancita y hoy ya ven a la cuquita hacerse ingeniera y cumplir su mayor meta, a ustedes por siempre darme el aliento de seguir adelante en cada momento de flaqueza, a ustedes por haber estado luchando contantemente sin rendirse por hacer de mi formación la mejor, a ustedes mis guerreros, mi mah y mi cachorro, MIL GRACIAS, este logro es de ustedes.

Muchas gracias a mis tutores por el apoyo y la compresión durante toda la etapa de formación, al profe Alfredo por soportar, además, mis malcriadeces. Gracias por convertirse en algo más que mis tutores, se ganaron un lugarcito en mi corazón.

A mi Tati por darme un apoyo incondicional, mira que hace casi 10 años eras tú quien redactaba estas líneas y me hiciste llorar al verte triunfar y convertirte en toda una profesional, hoy te toca ver a tu tatica hacerse grande y seguir tus pasos, te amo, gracias por todo. A mis tíos y mi hermanito que desde la distancia fueron una pieza clave durante toda la etapa universitaria, a pesar de la lejanía fueron muy importantes en mi formación.

“Gracias a toda la familia en general por el gran apoyo” …

Y bueno la universidad no hubiese sido la misma sin esos amigos que pasaron de noches constantes de estudios, insomnios a base de café, de fiestas y luego exámenes, a formar parte de mi familia, mi familia universitaria. Comienzo por una persona que me prohíbe terminantemente que le agradezca, pero no te voy a hacer caso, para variar, porque te has convertido en mi mejor amiga, mi confidente, mi hermana de otra mami, gracias por ser mi verbo to be, gracias por los desvelos escuchándome llorar por alguna decepción amorosa o por algún problema por mínimo que fuera y por sobre todo ser un gran apoyo en los momentos en los que me superaba la situación, gracias a la otra mitad del monguiequipo, a mi puerquis y gracias a tus papis por adoptarme en la familia desde el 1er día.

Gracias a mi bichita por estar en las verdes y en las maduras, gracias por escucharme pelear, llorar, reír y siempre quedarte, gracias por ser mi mejor amiga y compartir conmigo tu vida. Gracias a ese viejito peleón medio rojizo que se altera y pelea por todo y con todos pero que tiene un corazón inmenso que no le cabe en el pecho: mi rojito, gracias por todas esas noches de desvelo. A la VaneBu bipolar de la familia gracias por compartir momentos maravillosos a mi lado y estar presente en todo momento. A Papotico y Jonás por guardarme un lugarcito en sus corazones y ser mis hermanitos de otra mami. Mil gracias a las familias de mis amigos por quererme y hacerme formar parte de ellos. ¡Los quiero a todos, MIL GRACIAS!!!!

Muchas gracias a todas las personas que se incorporaron a mi vida en la última iteración de este proceso, Adriana, David, Javier, Yamel mi súper oponente, también los llevo en mi corazón…

Y, por último, pero no menos importante gracias a la CUJAE y a ese excelente claustro de profesores que fueron partícipe de toda mi formación como profesional y sobre todo como persona.

**Llevo un pedacito de cada uno en mí. ¡GRACIAS!!!!!!!**

**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 presentó 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, en 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 obtenidos demuestran una 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.

**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 (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, 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.

Índice

[**Introducción** 1](#_Toc121143484)

[**Capítulo 1. Fundamentos teóricos** 5](#_Toc121143485)

[1.1 Ingeniería de requisitos 5](#_Toc121143486)

[1.1.1 Estándares y formalización de requisitos de software 6](#_Toc121143487)

[1.1.2 Gestión de requisitos con soporte computacional 9](#_Toc121143488)

[1.1.3 Rol del lenguaje natural en los requisitos de software 10](#_Toc121143489)

[1.1.3.1 Construcción de un requisito en lenguaje natural. 11](#_Toc121143490)

[1.2 Procesamiento de lenguaje natural 12](#_Toc121143491)

[1.2.1 Sistemas de extracción de información 16](#_Toc121143492)

[1.2.2 Evaluación de sistemas de extracción de información 20](#_Toc121143493)

[1.3 Extracción automática de requisitos de software 22](#_Toc121143494)

[1.3 Análisis de tecnologías de implementación 27](#_Toc121143495)

[1.3.1 Python 27](#_Toc121143496)

[1.3.2 PyCharm (IDE) 28](#_Toc121143497)

[1.3.3 SpaCy 29](#_Toc121143498)

[**Conclusiones parciales** 32](#_Toc121143499)

[**Capítulo 2. Extracción automática de requisitos de software** 33](#_Toc121143500)

[2.1 Método de extracción automática de requisitos 33](#_Toc121143501)

[2.1.1 Pre – procesamiento 34](#_Toc121143502)

[2.1.2 Extracción de requisitos candidatos 35](#_Toc121143503)

[2.1.2.1 Análisis sintáctico basado en patrones léxicos – sintácticos 36](#_Toc121143504)

[2.1.2.2 Análisis sintáctico basado análisis de dependencias 38](#_Toc121143505)

[2.1.3 Reducción de redundancias 39](#_Toc121143506)

[2.1.4 Agrupamiento 40](#_Toc121143507)

[2.2 Desarrollo de la solución de extracción de requisitos de software 42](#_Toc121143508)

[2.2.1 Modelo del dominio 42](#_Toc121143509)

[2.2.2 Diagrama de casos de uso y descripción de casos de uso 43](#_Toc121143510)

[2.2.2.1 Actores del sistema 44](#_Toc121143511)

[2.2.2.2 Especificación de alto nivel de caso de uso 44](#_Toc121143512)

[2.2.3 Arquitectura del modelo 49](#_Toc121143513)

[2.2.4 Estilos y patrones de arquitectura 50](#_Toc121143514)

[**Conclusiones parciales** 53](#_Toc121143515)

[**Capítulo 3. Evaluación de la solución propuesta** 54](#_Toc121143516)

[3.1 Marco de evaluación 54](#_Toc121143517)

[3.1.1 Métricas de análisis de resultados 54](#_Toc121143518)

[3.1.2 Descripción de la colección de pruebas 56](#_Toc121143519)

[3.2 Resultados y discusión 57](#_Toc121143520)

[**Conclusiones parciales**: 61](#_Toc121143521)

[**Conclusiones Generales** 62](#_Toc121143522)

[**Recomendaciones** 63](#_Toc121143523)

[**Referencias Bibliográficas** 64](#_Toc121143524)

# Índice de Tablas

[Tabla 1 Resumen de revisión bibliográfica 26](#_Toc120813844)

[Tabla 2 Diferencias entre IDE Python 28](#_Toc120813845)

[Tabla 3 Comparación entre herramientas NLP 30](#_Toc120813846)

[Tabla 4 Patrones léxicos - sintácticos 37](#_Toc120813847)

[Tabla 5 Ejemplo de extracción usando análisis de dependencias 39](#_Toc120813848)

[Tabla 6 Actores del sistema y una descripción 44](#_Toc120813849)

[Tabla 7 Descripción del caso de uso Extraer requisitos de software. 45](#_Toc120813850)

[Tabla 8 Descripción del caso de uso Pre-Procesar texto. 45](#_Toc120813851)

[Tabla 9 Descripción del caso de uso Extraer requisitos candidatos 46](#_Toc120813852)

[Tabla 10 Descripción del caso de uso Aplicar estrategia de reducción de redundancias. 46](#_Toc120813853)

[Tabla 11 Descripción del caso de uso Agrupar semánticamente. 47](#_Toc120813854)

[Tabla 12 Descripción del caso de uso Exportar listado de requisitos funcionales. 47](#_Toc120813855)

[Tabla 13 Descripción de casos de estudio. 54](#_Toc120813856)

[Tabla 14 Resultados de la evaluación del método en colección de prueba. 55](#_Toc120813857)

# Índice de Ecuaciones

[Ecuación 1 Ecuación para calcular Precisión 51](#_Toc120807266)

[Ecuación 2 Ecuación para calcular Cobertura 51](#_Toc120807267)

[Ecuación 3 Ecuación para calcular Medida-F 51](#_Toc120807268)

# Índice de Ilustraciones

[Ilustración 1 Proceso de construcción de un requisito 11](#_Toc120813858)

[Ilustración 2 Estructura de un requisito en lenguaje natural 12](#_Toc120813859)

[Ilustración 3 Flujo de trabajo de la solución propuesta 34](#_Toc120813860)

[Ilustración 4 Diagrama de actividades del pre - procesamiento 35](#_Toc120813861)

[Ilustración 5 Diagrama de actividades del módulo Extracción de requisitos candidatos 36](#_Toc120813862)

[Ilustración 6 Árbol de dependencias generado por el analizador sintáctico SpaCy 39](#_Toc120813863)

[Ilustración 7 Diagrama de actividades de módulo Reducción de redundancias 41](#_Toc120813864)

[Ilustración 8 Modelo del dominio 43](#_Toc120813865)

[Ilustración 9 Diagrama de caso de uso 44](#_Toc120813866)

[Ilustración 10 Estructura en capas del modelo. 48](#_Toc120813867)

[Ilustración 11 Estructura modular del sistema. Arquitectura basada en componentes. 50](#_Toc120813868)

[Ilustración 12 Comportamiento de las métricas Precisión, Cobertura y Medida-F con el uso de Patrones léxicos-sintácticos 56](#_Toc120813869)

[Ilustración 13 Comportamiento de las métricas Precisión, Cobertura y Medida-F con el uso de análisis de dependencias 57](#_Toc120813870)

[Ilustración 14 Comportamiento de las métricas Precisión, Cobertura y Medida-F en el enfoque híbrido 57](#_Toc120813871)

[Ilustración 15 Análisis del comportamiento promedio de las métricas 58](#_Toc120813872)

# 

# Introducción

La industria del software presenta índices de éxito relativamente bajos en los proyectos, 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 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 [1]:

* 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).

En el proceso de captura de requisitos de software se ha demostrado que el porcentaje de fallos sigue siendo considerablemente alto y el costo de corregir un error a partir de un mal proceso de ingeniería de requisitos (RE) 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 este porcentaje de fallos en 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 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.

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 NLP para procesar los textos de requisitos [8]. Sin embargo, durante casi tres décadas, esta investigación se vio obstaculizada por tecnologías de NLP inadecuadas, que no funcionaron lo suficientemente bien en ese momento para respaldar aplicaciones fuera de la investigación de la NLP [9]. La situación ha ido cambiando radicalmente desde finales del año 2000. Enormes mejoras y avances en las tecnologías de NLP, en particular la amplia disponibilidad de herramientas y recursos de NLP [10], hicieron posible que los investigadores exploraran una variedad de herramientas y recursos de NLP, habilitando herramientas y métodos para tareas de ER [11] [12]. Desde entonces, la investigación en NLP 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 RE 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 para la práctica de RE. Esto es particularmente necesario, ya que las tareas de RE todavía se realizan manualmente en la práctica industrial, como lo demuestran los datos de la encuesta de “Naming the Pain in Requirements Engineering”(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: La elicitación de requisitos es un 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 estándares y modelos de formalización de requisitos reportados en la literatura.

* Estudiar documentación sobre estándares y modelos de formalización de requisitos de software reportados en la literatura.
* Caracterizar estándares y modelos de formalización de requisitos, en cuanto a: elementos de información que intervienen, especificaciones, artefactos, elementos comunes, entre otros elementos.

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.
* Identificar y caracterizar fuentes de información que será suministrada al modelo para la generación de requisitos.
* Determinar especificaciones de salidas del modelo a diseñar.
* Identificar recursos (bases de conocimiento u otros) que pueden aportar información/conocimiento al modelo

1. Implementar sistema para la extracción automática de requisitos de software.

* Implementar método basado en el análisis sintáctico a partir de patrones léxico – sintácticos.
* Extender implementación de los patrones léxicos-sintácticos.
* Implementar solución basa en el análisis de dependencias.
* Concebir mejoras al proceso de reducción de redundancias de requisitos capturados.
* Implementar mejoras concebidas al proceso de reducción de redundancias.

1. Evaluar la nueva versión de solución de capturas de requisitos.

* Construir colecciones de prueba para la experimentación y medición de resultados.
* Ejecutar una fase de ajuste del marco de evaluación diseñado.
* Refinar modelo e implementación según resultados preliminares de la evaluación realizada.
* Ejecutar marco de evaluación diseñado.
* 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 Fundamentos teóricos. Capítulo 2 Extracción automática de requisitos de software. Capítulo 3 Evaluación de la solución propuesta.

# Capítulo 1. Fundamentos teóricos

En este capítulo se presentan las bases teóricas del trabajo, los aspectos necesarios para lograr la comprensión de conceptos que permitan el cumplimiento de los objetivos trazados. Se abordan los conceptos relativos a la Ingeniería de requisitos (RE) y el Procesamiento del lenguaje natural (NLP), así como la aplicación de técnicas, recursos y herramientas de NLP a la RE.

# 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 (RE) [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.

## Estándares y formalización de requisitos de software

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

1. Documentos híbridos: Actualmente es 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 definen 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

1. **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)

1. **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.

1. **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.
2. **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…”.
3. 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.

## 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, en [20] se propone un método para identificar automáticamente los requisitos de seguridad, combinando el análisis lingüístico con técnicas de aprendizaje automático.

En [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. En [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.

En [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.

En [24], se aplica el aprendizaje automático para predecir el rendimiento de ciertas configuraciones del sistema que puedan afectar a los requerimientos. En [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. En [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 RE mediante las técnicas de clasificación. En [20], proponen un método para detectar ambigüedad en los documentos de especificación de requisitos, mediante el uso de redes bayesianas. En [27], proponen el uso de clasificadores bayesianos para clasificar los riesgos en los requerimientos. En [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. En [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 [30].

## 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, 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, se puede 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].

### Construcción de un requisito en lenguaje natural.

Para la construcción de un requisito en lenguaje natural se pueden plantear 5 pasos fundamentales como se muestra en la ilustración 1, con el objetivo de garantizar la calidad de los requisitos mediante el uso de una estructura definida y glosarios de términos [16].

Ilustración 1 Proceso de construcción de un requisito

Se obtiene la estructura de la ilustración 2 como base para la construcción del requisito en lenguaje natural:

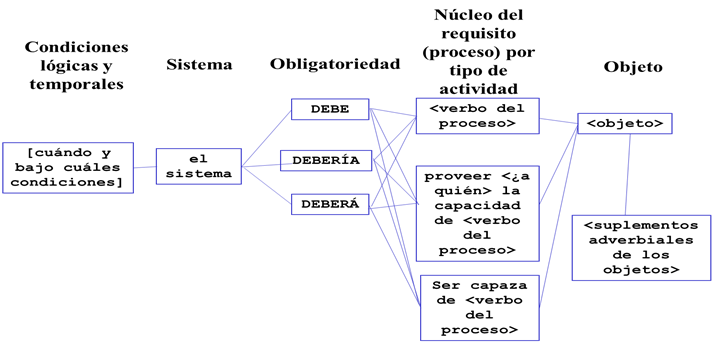


Ilustración 2 Estructura de un requisito en lenguaje natural

# 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. Tradicionalmente, existen tres niveles principales de procesamiento en un enfoque basado en NLP [36]:

* 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(Part-of-speech): 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)

Tipos de tecnología NLP:

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

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

* Stanford CoreNLP7[[1]](#footnote-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 NLP 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]](#footnote-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 conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de NLP 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]](#footnote-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.

## Sistemas de extracción de información

Gran parte del conocimiento del mundo está registrado en texto de lenguaje natural, pero hacer un uso efectivo de este en esta forma plantea un gran desafío. La extracción de información convierte este conocimiento en una forma estructurada adecuada para la manipulación informática, abriendo muchas posibilidades para su uso [37].

A menudo nos referimos al texto como datos no estructurados. Sin embargo, en realidad, el texto tiene mucha estructura, solo que la mayor parte no es explícita, lo que dificulta la búsqueda o el análisis de la información dentro del texto. El objetivo de la extracción de información (IE) es hacer explícita la estructura semántica del texto para que podamos hacer uso de ella. Más precisamente, IE es el proceso de analizar texto e identificar menciones de entidades definidas semánticamente y relaciones dentro de él. Estas relaciones pueden luego registrarse en una base de datos para buscar relaciones particulares o para inferir información adicional de los hechos declarados explícitamente [37].

Por otro lado, el término NLP se refiere a los métodos para interpretar los datos, es decir, hablados o escritos por humanos. Para procesar los lenguajes humanos usando NLP, varias tareas como la traducción automática, el sistema de preguntas y respuestas, la recuperación de información, la extracción de información y la comprensión del lenguaje natural se consideran tareas de alto nivel. El proceso de IE es una de las tareas importantes en el análisis de datos, KDD y minería de datos [38] que extrae información estructurada de los datos no estructurados. IE se define como "extraer instancias de categorías predefinidas de datos no estructurados, construyendo una representación estructurada y sin ambigüedades de las entidades y las relaciones entre ellas" [39]. Una de las intenciones de IE es poblar las bases de conocimientos para organizar y acceder a información útil. Toma una colección de documentos como entrada y genera diferentes representaciones de información relevante que satisfacen diferentes criterios. Las técnicas de IE analizan eficientemente el texto en forma libre al extraer la información más valiosa y relevante en un formato estructurado. Por lo tanto, el objetivo final de las técnicas de IE es identificar los hechos más destacados del texto para enriquecer las bases de datos o bases de conocimiento.

El reconocimiento de entidades con nombre es una de las tareas importantes de los sistemas IE utilizados para extraer entidades descriptivas. Ayuda a identificar las entidades genéricas o independientes del dominio, como la ubicación, las personas y la organización, y las entidades específicas del dominio, como la enfermedad, el fármaco, la sustancia química, las proteínas, etc. En este proceso, las entidades se identifican y se clasifican semánticamente en entidades pre-caracterizadas, clases [40]. Los sistemas NER tradicionales usaban métodos basados ​​en reglas (RBM), métodos basados ​​en el aprendizaje (LBM) o enfoques híbridos [41]. IE, junto con NLP, juega un papel importante en el modelado del lenguaje y el IE contextual mediante el análisis morfológico, sintáctico, fonético y semántico de los idiomas. Los lenguajes morfológicos ricos como el ruso y el inglés facilitan el proceso de IE. IE es difícil para los idiomas morfológicamente pobres porque estos idiomas necesitan un esfuerzo adicional para que las reglas morfológicas extraigan el sustantivo debido a la falta de disponibilidad de un diccionario completo [42].

La extracción de relaciones es una subtarea de IE que extrae relaciones sustanciales entre entidades. Las entidades y relaciones se utilizan para anotar correctamente los datos mediante el análisis de las propiedades semánticas y contextuales de los datos. Los enfoques supervisados ​​utilizan técnicas basadas en funciones y basadas en kernel para la extracción de relaciones. DIPRE, Snowball, KnowItAll son algunos ejemplos de extracción de relaciones semisupervisados [43]​​. Se han introducido varios enfoques supervisados, débilmente supervisados ​​y auto-supervisados ​​para extraer relaciones de uno a uno y de muchos a muchos entre entidades.

Los etiquetadores de parte del discurso (POS) se utilizan muy comúnmente en los sistemas de extracción de información [44]. Se pueden usar para ayudar en la coincidencia de patrones/plantillas o como un atributo de entrada para modelos de aprendizaje automático.

Wordnet es una gran base de datos léxica de sustantivos, verbos, adverbios y adjetivos desarrollada en la Universidad de Princeton [45]. Las palabras se agrupan en conjuntos de sinónimos, synsets, que se pueden utilizar de manera efectiva en los sistemas IE. Wordnet tiene actualmente 155.287 palabras/frases y 117.659 conjuntos de sintaxis

La mayoría de los enfoques sobre IE se pueden agrupar en cualquier categoría de tecnologías basadas en reglas, es decir, basadas en conocimientos o basadas en aprendizaje automático (ML). Los principales marcos de anotación para los sistemas IE modernos son Apache UIMA y Gate [46]. Ambos marcos cuentan con un lenguaje de reglas integral, UIMA Ruta1 (anotación de texto basada en reglas) y JAPE2 respectivamente, que también se utilizan en el mundo académico. Han surgido muchos otros lenguajes, que agregan características valiosas y conceptos novedosos que abordan las deficiencias de los lenguajes ya existentes. Hasta ahora, se ha realizado poca investigación sistémica para preparar el ecosistema de reglas para la nueva era de los métodos de aprendizaje automático. Mostrar los poderes versátiles de las reglas mediante la creación de sistemas de extracción de información más prominentes utilizando enfoques basados en reglas, agrupando el esfuerzo de investigación en una plataforma común y creando herramientas reutilizables e independientes del sistema, podría revitalizar el interés de la investigación en el prometedor campo de la extracción de información basada en reglas [47].

El uso cada vez mayor de los sistemas IE para tareas como la búsqueda bibliográfica y la búsqueda de información para las decisiones de contratación exige altos niveles de precisión. Los siguientes son ejemplos de investigaciones de IE que podrían contribuir al objetivo de satisfacer estas necesidades.

La investigación en la Universidad Carnegie Mellon, que explora las estructuras del modelo oculto de Markov para los modelos IE, está aplicando técnicas estadísticas de aprendizaje automático que han sido útiles en otras áreas de la Inteligencia Artificial (IA) [48]. Se centran específicamente en cómo aprender la estructura del modelo a partir de datos etiquetados y no etiquetados. Exploran estrategias para aprender la estructura del modelo automáticamente a partir de los datos. Han demostrado que los datos etiquetados a distancia (un método para el etiquetado automático al relajar reglas estrictas para las relaciones) se pueden usar para establecer parámetros del modelo para mejorar la precisión de la extracción de información. Un ejemplo de demostración ha sido la tarea de extraer campos importantes de los encabezados de trabajos de investigación en informática.

La investigación en la Universidad de Massachusetts Amherst analiza los enfoques de campo aleatorio condicional (CRF) para la tarea de extraer varios campos comunes de los encabezados y citas de trabajos de investigación [49]. Si bien la teoría de los CRF se ha ido desarrollando, aún quedan desafíos para la investigación aplicada cuando se enfrenta a los tipos de datos dinámicos a gran escala en el entorno práctico actual. La investigación de CRF explora varios factores, incluidas las variaciones en los priores L1 gaussianos, exponenciales e hiperbólicos para mejorar la regularización. Están logrando un nuevo rendimiento de vanguardia en comparación con los mejores resultados anteriores de SVM y HMM.

El Siena College Institute for Artificial Intelligence (SCIAI) tiene varios proyectos de IE que buscan usos potenciales de la computación adaptativa. El trabajo de SCIAI sobre IE, a partir de datos médicos comenzó con el concurso TREC de registros médicos de 2012 [50], que tenía como tarea procesar listas de consultas seleccionadas aleatoriamente contra un gran corpus de registros médicos. El objetivo era simular búsquedas de pacientes que pudieran cumplir con los criterios para participar en ensayos clínicos particulares, y el conjunto de datos tenía un promedio de 15 informes de cada una de aproximadamente 100 000 visitas de pacientes a un hospital.

La investigación y el desarrollo de aplicaciones en el campo de la extracción de información deben basarse en los mejores avances en investigación y tecnología de IA, incluido el trabajo en computación adaptativa. Los imperativos del mundo actual centrado en la web, con cantidades explosivas y fuentes ubicuas de datos, y los impactos sociales de la interactividad web presentan oportunidades sin precedentes para crear conocimiento que tendrá un gran impacto en nuestras vidas. Las nuevas áreas emergentes de investigación y desarrollo, como la Web Semántica, la Ciencia Web y Big Data, presentan oportunidades interesantes en las que la computación neuronal puede desempeñar un papel importante.

## Evaluación de sistemas de extracción de información

Además de las métricas de evaluación estándar de los sistemas IE, como la precisión (Precisión = Número de elementos encontrados/número de elementos correctos; donde los elementos pueden ser entidades, relaciones, eventos, etc.) y el Recall (Recall = Número de elementos encontrados/número total de elementos que deberían haberse encontrado; donde los elementos pueden ser entidades, relaciones, eventos, etc), las técnicas también se pueden juzgar por medio de la implementación. Si bien el aprendizaje automático estadístico se usa ampliamente como una tecnología de caja negra con respecto a la transparencia de la trazabilidad de las decisiones, los enfoques basados en reglas siguen un enfoque principalmente declarativo que conduce a modelos altamente transparentes y expresivos [47].

Para evaluar los sistemas de extracción de información, se deben crear documentos anotados manualmente. Para los sistemas de extracción de información de dominio específico, los documentos anotados deben provenir del dominio de destino. Por ejemplo, para evaluar la extracción de nombres de genes y proteínas, se utilizan documentos biomédicos como los resúmenes de PubMed. Pero si el propósito es evaluar técnicas generales de extracción de información, se pueden usar conjuntos de datos de referencia estándar. Los conjuntos de datos de evaluación comúnmente utilizados para el reconocimiento de entidades nombradas incluyen los de MUC [51], CoNLL-2003 [52] y ACE[[4]](#footnote-4). Para la extracción de relaciones, se suelen utilizar conjuntos de datos ACE.

Las métricas de evaluación típicas para la extracción de información son *Precision*, *Recall* y *F-1 scores*. La precisión mide el porcentaje de acierto entre las instancias positivas identificadas. El recall mide el porcentaje de instancias correctas que se pueden identificar entre todas las instancias positivas. F-1 es la media geométrica de precisión y recall.

Para el reconocimiento de entidades nombradas, estrictamente hablando, una entidad nombrada correctamente identificada debe satisfacer dos criterios, a saber, límite de entidad correcto y tipo de entidad correcto. La mayoría de las evaluaciones se basan en la coincidencia exacta de los límites de la entidad. Sin embargo, no vale la pena que en algunos casos también se deba dar crédito a coincidencias parciales, cuando el objetivo es solo decir si una entidad se menciona en un documento o en una oración [53].

Para la extracción de relaciones, existen dos niveles de extracción, nivel de corpus y nivel de mención. Mientras que la evaluación a nivel de mención requiere instancias de mención de relación anotadas, la evaluación a nivel de corpus requiere solo pares de entidades verdaderamente relacionadas, que pueden ser más fáciles de obtener o anotar que las menciones de relación.

Actualmente, los métodos de reconocimiento de entidades nombradas de última generación pueden lograr alrededor del 90 % de las scores F-1 cuando se entrenan y prueban en el mismo dominio [52]. En general, se observa que las entidades de personas son más fáciles de extraer, seguidas de las ubicaciones y luego las organizaciones. Es importante tener en cuenta que cuando hay un cambio de dominio, el rendimiento del reconocimiento de entidades nombradas puede caer sustancialmente. Se han realizado varios estudios que abordan el problema de la adaptación del dominio para el reconocimiento de entidades nombradas, (por ejemplo: [54], [55] )

Para la extracción de relaciones, el rendimiento de última generación es inferior al del reconocimiento de entidades nombradas. En el conjunto de datos de referencia de ACE 2004, por ejemplo, la mejor puntuación de F-1 es de alrededor del 77 % para los siete tipos principales de relaciones [56].

# Extracción automática de requisitos de software

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) [57].

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 [57].

Centrándose específicamente en la extracción automática de requisitos de software:

La solución reportada por [58] 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 analizó la oración en un árbol de acuerdo con esta gramática; el sujeto se identificó como el sintagma nominal principal en el árbol.

En [59] 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 [60] el procedimiento que se desarrolla es el enfoque L'Ecritoire, 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 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 [61] 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 [62], 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 [63] 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 [64] 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.

El método de extracción propuesto por [65] 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.(2) Constituyentes de la cláusula, 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 objetivo.

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

En la Tabla 1 se muestra un resumen de este análisis bibliográfico:

Tabla 1 Resumen de revisión bibliográfica

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Trabajos relacionados*** | ***Información a extraer*** | ***Idioma a procesar*** | ***Herramientas PLN*** | ***Técnica de extracción de requisitos*** | | ***Reducción de redundancias*** | ***Métricas de calidad*** |
| [58] | Requisito funcional | inglés | NLTK | análisis de dependencias |  | similitud de coseno |  |
| [59] | Requisito no funcional | inglés | Stanford |  |  | \_\_ |  |
| [60] | Requisito funcional | inglés | NLTK |  | patrones sintácticos | \_\_ |  |
| [61] | Requisito funcional | inglés | \_\_\_ |  | patrones sintácticos | \_\_ |  |
| [62] | Requisito no funcional | inglés | Stanford |  | patrones sintácticos | \_\_ |  |
| [63] | Requisito funcional | inglés | Stanford |  | patrones sintácticos | similitud de coseno |  |
| [64] | Requisito funcional | inglés | Stanford |  | patrones sintácticos | \_\_ |  |
| [65] | Requisito funcional | inglés | \_\_\_ | análisis de dependencias |  | \_\_ |  |
| [48] | Extracción de relaciones | inglés |  |  |  |  | Precisión, Recall, scoreF1 |
| [51] | Entidades nombradas | inglés |  |  |  |  | Precisión, Recall, scoreF1 |
| [52] | Entidades nombradas | inglés |  |  |  |  | Precisión, Recall, scoreF1 |
| [43] | Extracción de relaciones | inglés |  |  |  |  | Precisión, Recall, scoreF1 |

# 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 PyCharm 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.

## Python

**Python[[5]](#footnote-5)** 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 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.

## PyCharm (IDE)

**PyCharm[[6]](#footnote-6)**  es un [entorno de desarrollo integrado](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) utilizado en [la programación de computadoras](https://en.wikipedia.org/wiki/Computer_programming) , específicamente para el lenguaje de programación [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) . Está desarrollado por la empresa [checa](https://en.wikipedia.org/wiki/Czech_Republic)[JetBrains](https://en.wikipedia.org/wiki/JetBrains) (anteriormente conocida como IntelliJ). Proporciona análisis de código, un depurador gráfico, un probador de unidades integrado, integración con [sistemas de control](https://en.wikipedia.org/wiki/Revision_control) de versiones (VCS) y es compatible con el desarrollo web con [Django](https://en.wikipedia.org/wiki/Django_(web_framework)) , así como con [la ciencia de datos](https://en.wikipedia.org/wiki/Data_science) con [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)).

PyCharm es [multiplataforma](https://en.wikipedia.org/wiki/Cross-platform) , con versiones de [Windows](https://en.wikipedia.org/wiki/Windows) , [macOS](https://en.wikipedia.org/wiki/MacOS) y [Linux](https://en.wikipedia.org/wiki/Linux) . La edición comunitaria se publica bajo la [licencia Apache](https://en.wikipedia.org/wiki/Apache_License), y también hay una versión educativa, así como una edición profesional con características adicionales (publicada bajo una [licencia propietaria](https://en.wikipedia.org/wiki/Proprietary_software)[financiada por suscripción](https://en.wikipedia.org/wiki/Subscription_business_model) ).

***PyCharm vs Jupyter***

**Jupyter[[7]](#footnote-7)**

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 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.

**Diferencias entre Jupyter y PyCharm[[8]](#footnote-8):**

En la tabla 2 se muestran las diferencias que existen entre PyCharm y Jupyter.

Tabla 2 Diferencias entre IDE Python

|  |  |
| --- | --- |
| **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. |
| 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 |

## SpaCy

**SpaCy[[9]](#footnote-9)** es una librería de software para  [procesamiento de lenguajes naturales](https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales) desarrollado por Matt Honnibal y programado en lenguaje [Python](https://es.wikipedia.org/wiki/Python). Fue lanzado en febrero de 2015 estando su desarrollo activo y siendo utilizado en distintos entornos. Es software libre con [Licencia MIT](https://es.wikipedia.org/wiki/Licencia_MIT) su repositorio se encuentra disponible en [Github](https://es.wikipedia.org/wiki/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.

La Universidad Católica de Salta realizó una comparación de herramientas de procesamiento de textos en español para Python y explica lo siguiente [66]:

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 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.

Tabla 3 Comparación entre herramientas NLP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |

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.

# Conclusiones parciales

Al terminar el capítulo de estudio teórico 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 NLP como parte del proceso de RE, en este estudio se incluyeron solo aquellas a las cual se tuvo acceso por ser Open Source.
* Como conclusión del análisis de la revisión bibliográfica se identificaron varias limitaciones, tales como, la no existencia de disminución de redundancias entre las frases extraídas, no existe procesamiento para el idioma español y poco uso de la técnica de extracción de información basada en el análisis de dependencias.
* El lenguaje de programación Python fue seleccionado para implementar la 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.

# 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.

# Método de extracción automática de requisitos

En esta se sección describe la solución propuesta en el trabajo. Las principales etapas del proceso de desarrollo se describen en la Ilustración 3.

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 pre-procesado 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 pre-procesado, a la extracción de los requisitos candidatos por dos vías, análisis sintáctico basado en patrones léxicos-sintácticos y basada en el análisis de dependencias, mostrando así, una solución final basada 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 q sean exactamente iguales y aquellas que estén contenidas unas dentro de otras, seleccionado la frase más completa. Este conjunto de frases luego de haber sido filtrado se agrupa, formando clústers a partir de la similitud semántica que exista por cada dos pares de frases. Estos clústers son exportados al usuario en un archivo txt, concluyendo así, el proceso de extracción automático de requisitos de software.

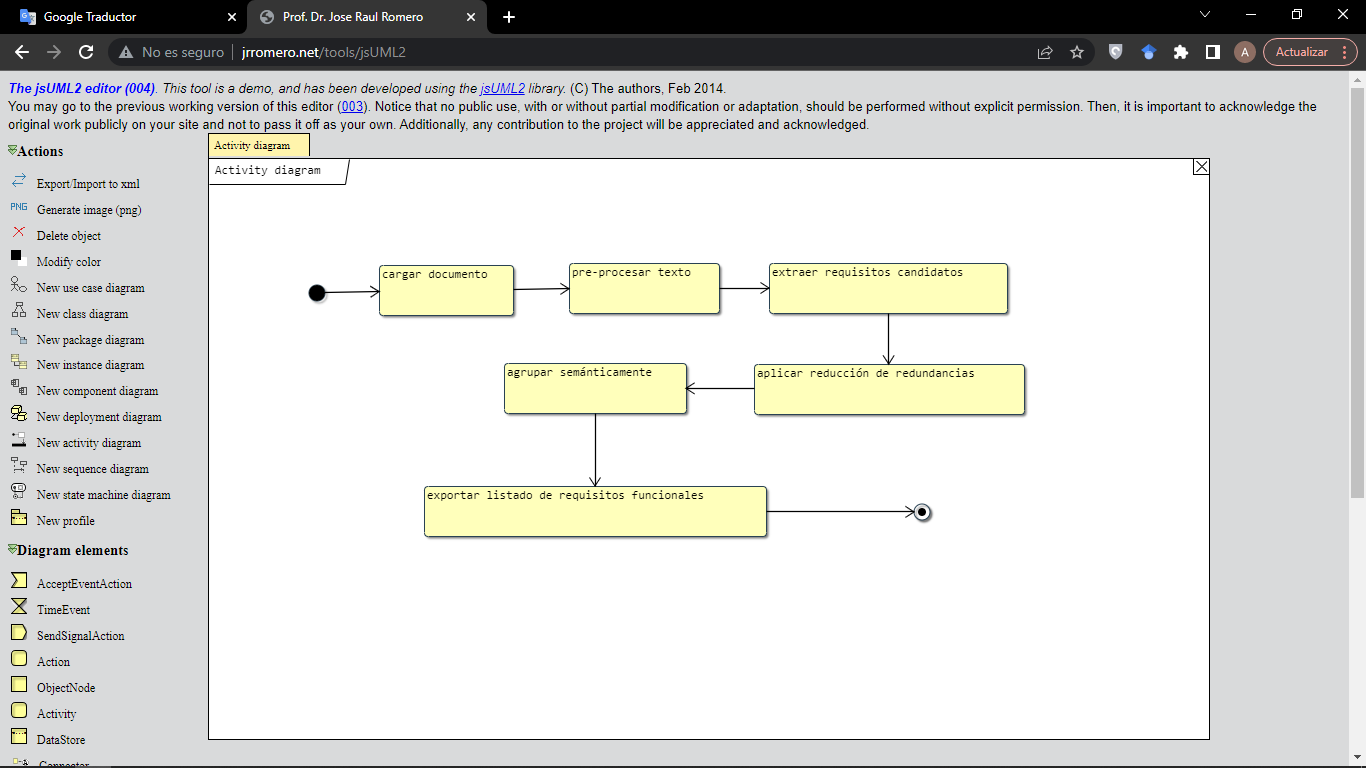


Ilustración 3 Flujo de trabajo de la solución propuesta

## 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 y prepara 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 Ilustración 4 se muestra un diagrama de actividades del pre – procesamiento, que inicia cuando el usuario carga un archivo en formato texto.

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 plano en palabras.
* 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.

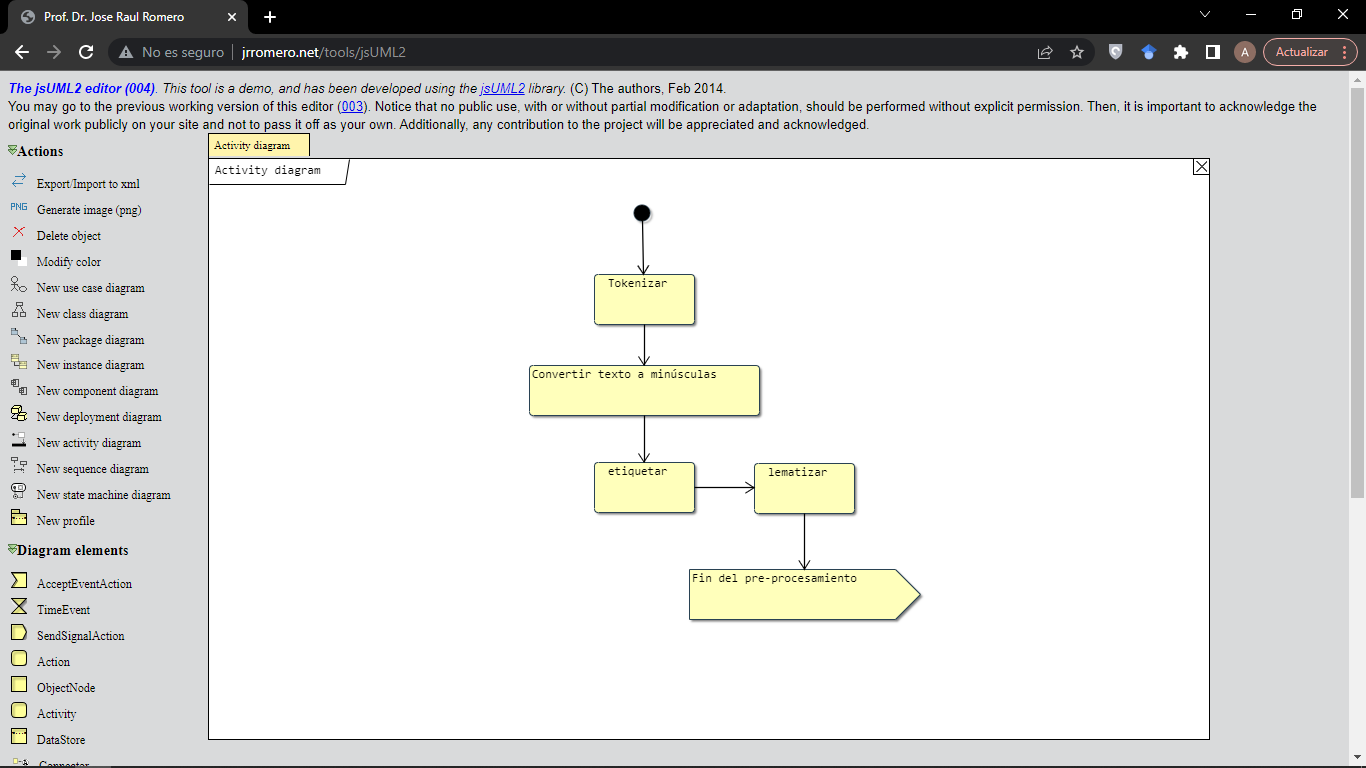


Ilustración 4 Diagrama de actividades del pre - procesamiento

## 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 basada en la concatenación de ambas técnicas de extracción de información. El flujo de este proceso se muestra en la ilustración 5.

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 los sintagmas formados por cada una de las unidades léxicas que aparecen en la oración.

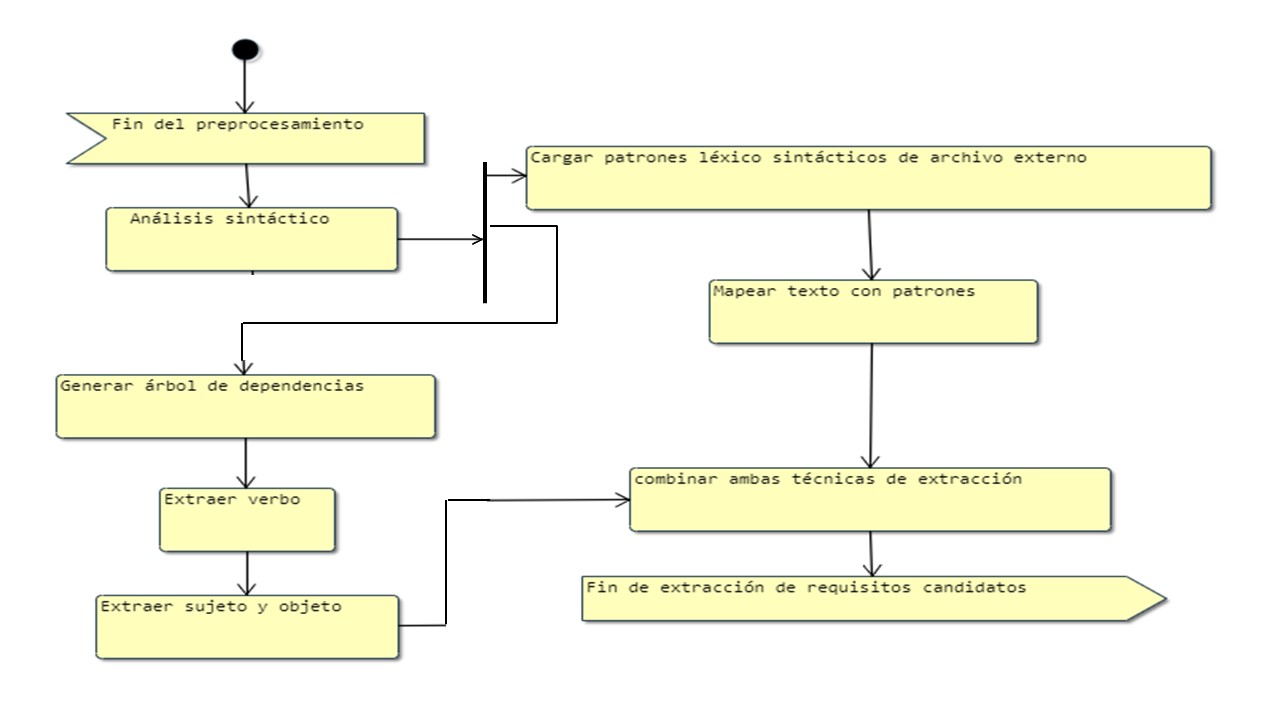


Ilustración 5 Diagrama de actividades del módulo Extracción de requisitos candidatos

### Análisis sintáctico basado en patrones léxicos – 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, aquella secuencia de etiquetas gramaticales que identifican la clasificación gramatical de cada uno de los tokens que conforman el sintagma a extraer.

Para extraer las frases que dan lugar a los requisitos candidatos, se definieron previamente un conjunto de patrones léxicos – 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 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 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 requisitos candidatos.

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

Tabla 4 Patrones léxicos - sintácticos

|  |  |
| --- | --- |
| ***Patrón léxico – sintáctico*** | ***Ejemplo de frase*** |
| 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 sintáctico basado 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 [67]. 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 Ilustración 7, 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 la misma, se pueden establecer distintas relaciones de dependencia, donde la raíz de todas ellas 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 [68], reconocimiento de la implicación textual [69], detección de conceptos negados en una frase [70], etc.

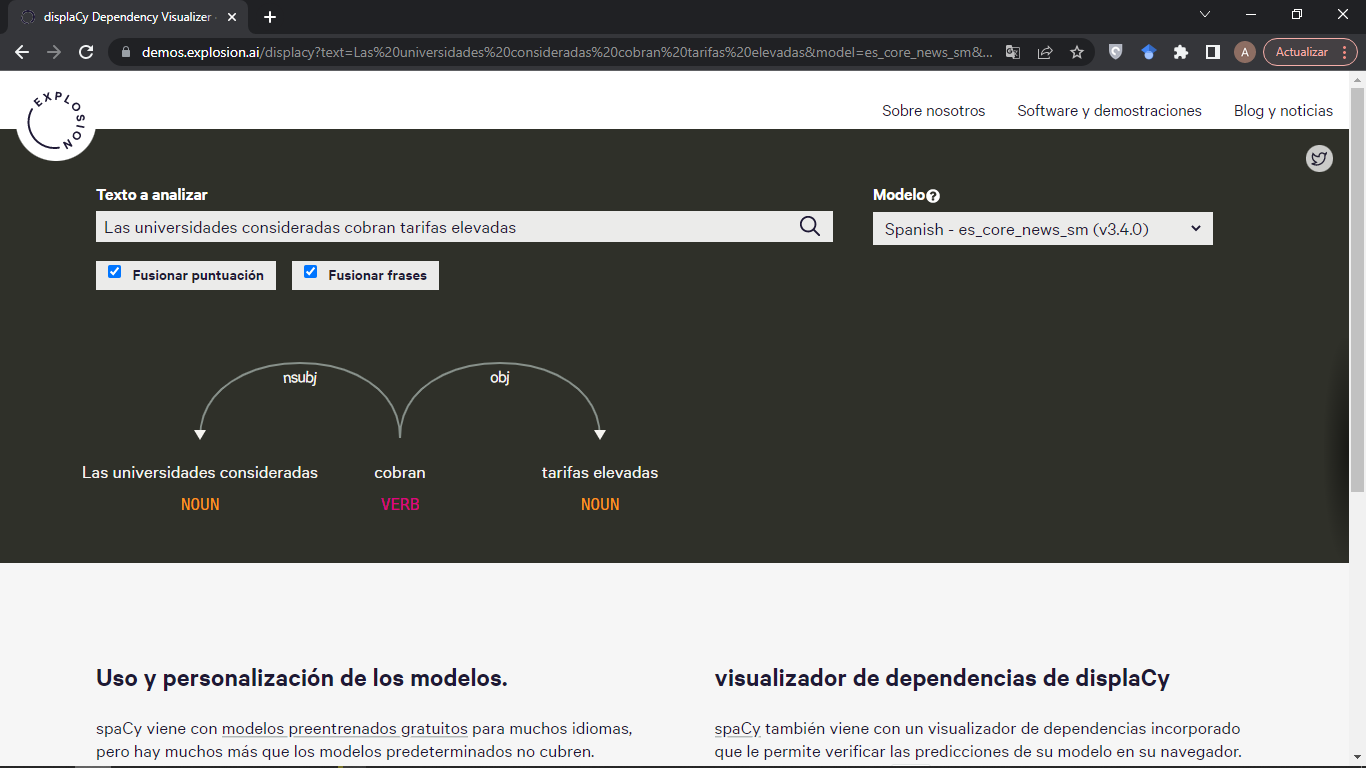


Ilustración 6 Árbol de dependencias generado por el analizador sintáctico SpaCy

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 [65], se estará utilizando un patrón para extraer todos 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 dicha estructura.

Reutilizando el ejemplo anteriormente mencionado la frase generada por este método de extracción quedaría de la siguiente forma:

Tabla 5 Ejemplo de extracción usando análisis de dependencias

|  |  |
| --- | --- |
| **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 ilustración 7, con el objetivo de mejorar la calidad de los requisitos extraídos. Durante la primera de prueba se observó en los requisitos extraídos que algunos 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 2 criterios:

* Reducir a una, aquellas frases que hagan referencia a un mismo requisito seleccionando de ellas la que abarque la mayor cantidad información y descartando el resto, criterio que se corresponde al ejemplo explicado anteriormente.
* 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.

## 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 ilustración 7, con el objetivo de mejorar la visualización de los resultados para una mayor compresión de los mismos por el analista. Esta encarga de agrupar en diferentes clústers aquellas frases que tengan cierta similitud semántica. Para ello se usó la métrica de similitud semántica de WordNet Similarity, Wu-Palmer y el algoritmo de agrupamiento Hierarchical Agglomerative. Para la evaluación de la calidad del agrupamiento se usó la métrica Sihlouette.

* 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 ellas 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 [71], wup [72] 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 [73].
* 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 [74].
* El valor de Sihlouette[[10]](#footnote-10) 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 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.

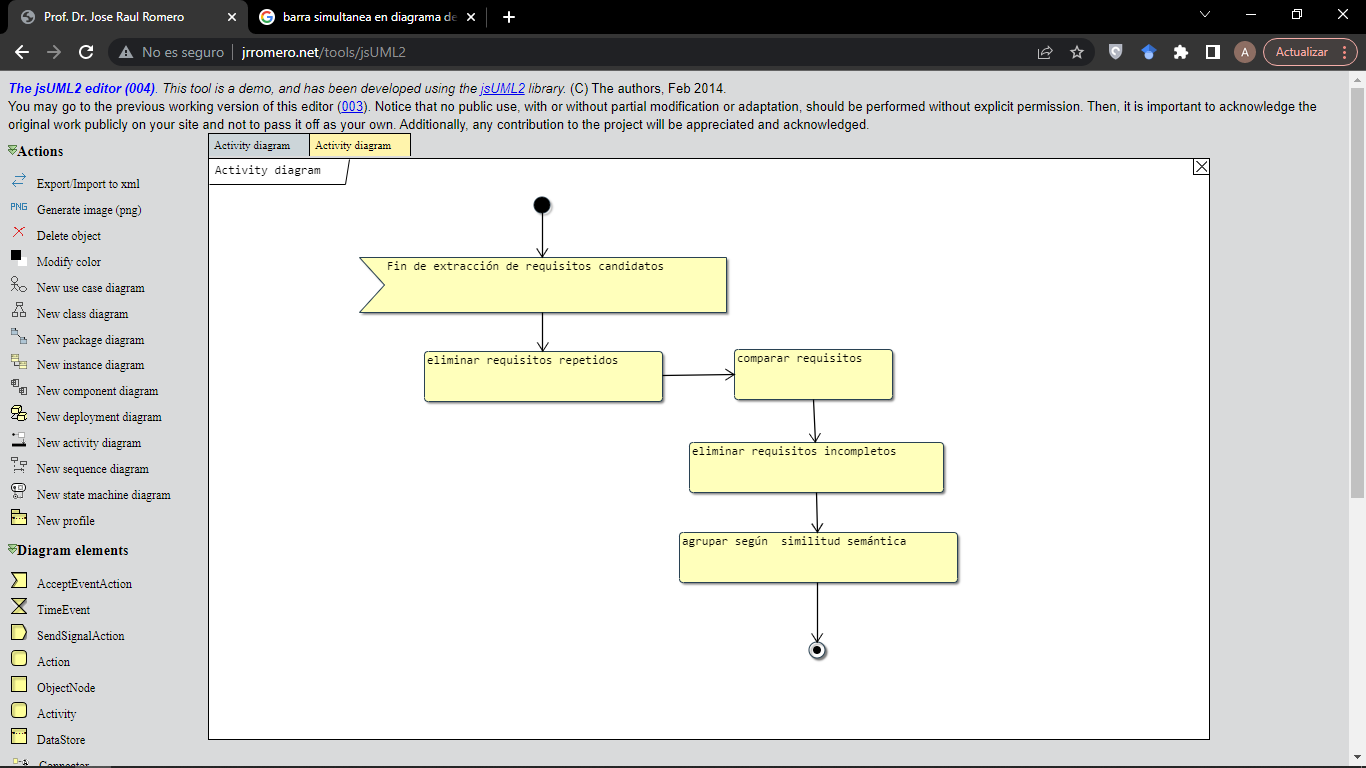


Ilustración 7 Diagrama de actividades de los módulos Reducción de redundancias y Agrupamiento

# Desarrollo de la solución de extracción de requisitos de software

## Modelo del dominio

El entorno de la solución está enmarcado en diferentes conceptos que se presentan a continuación como modelo del dominio.

Un **modelo de dominio[[11]](#footnote-11)** describe los tipos de dominio que admite una organización y sus restricciones, como se muestra en la ilustración 8.

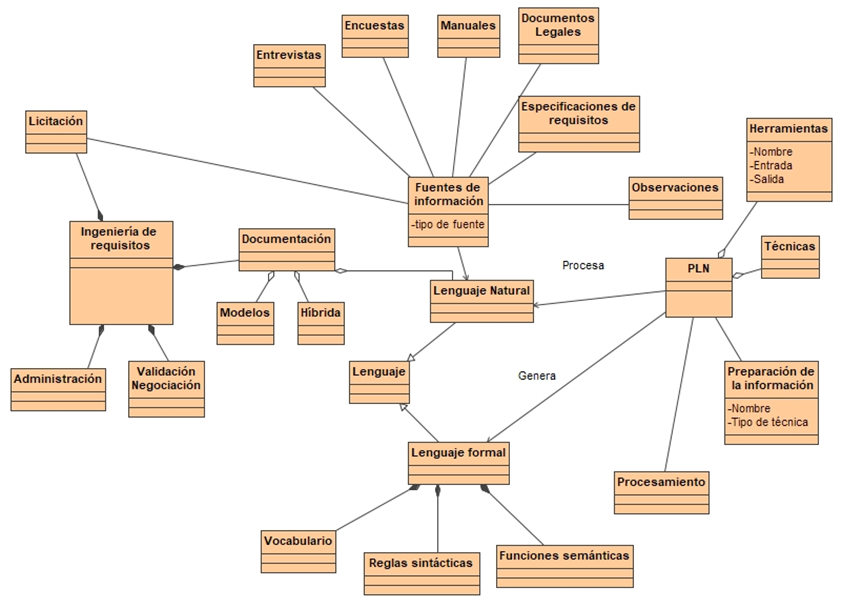


Ilustración 8 Modelo del dominio

## Diagrama de casos de uso y descripción de casos de uso

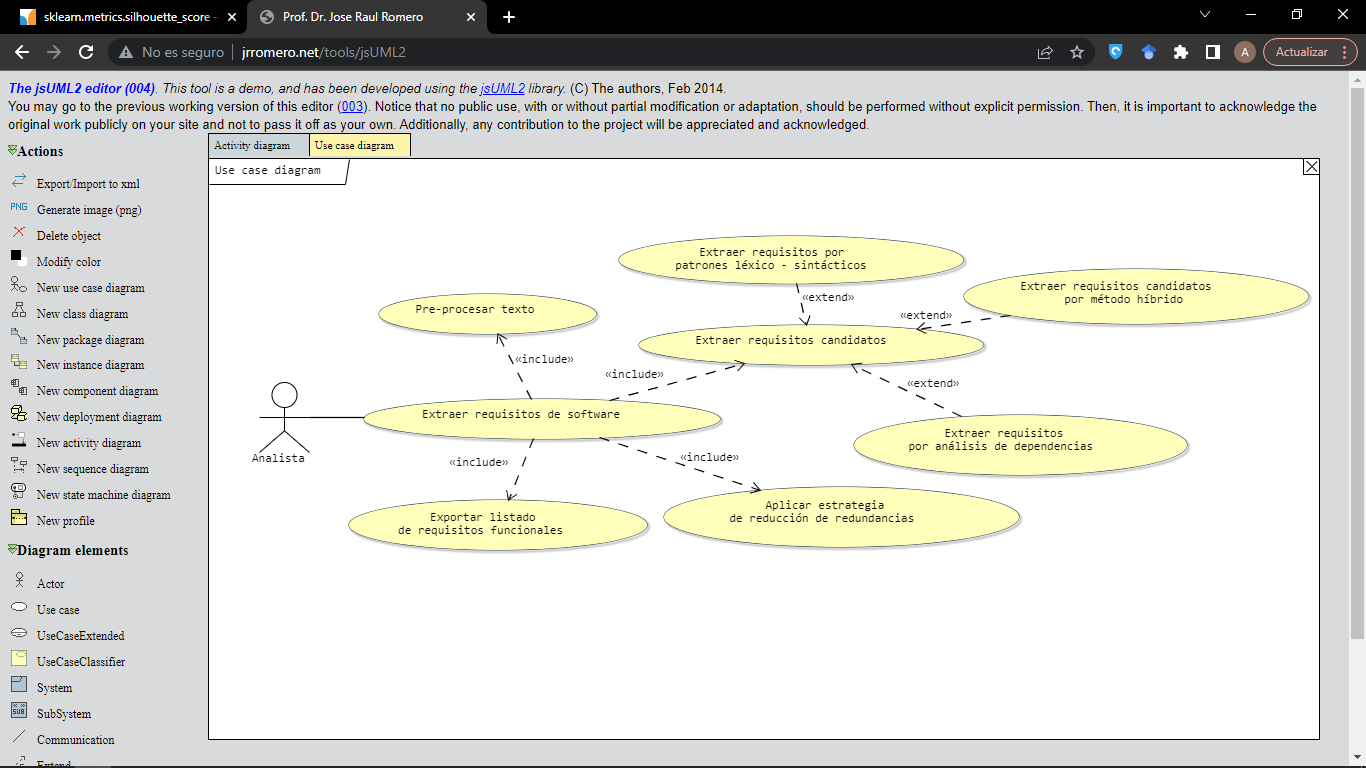
El modelado de casos de uso se utiliza ampliamente para apoyar la adquisición de requerimientos. Un caso de uso puede tomarse como un simple escenario que describa lo que espera el usuario de un sistema. Cada caso de uso representa una tarea discreta que implica interacción externa con un sistema. Un caso de uso es una unidad coherente de funcionalidad expresada como una transacción entre los actores y el sistema [75]. 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 9 se muestra el diagrama de Casos de Uso de la solución propuesta.

Ilustración 9 Diagrama de caso de uso

### Actores del sistema

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

Tabla 6 Actor del sistema y su 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. |

### Especificación de alto nivel de caso de uso

La especificación de cada caso de uso se muestra en la Tabla 7, 8, 9, 10,11,12, 13, 14 y 15.

Tabla 7 Descripción del caso de uso Extraer requisitos de software.

|  |  |
| --- | --- |
| **Caso de uso** | Extraer requisitos de software |
| **Actores** | Analista |
| **Propósito** | Generar las sentencias que dan lugar a requisitos funcionales de software. |
| **Descripción** | Para la extracción de los requisitos el analista debe, inicialmente, cargar el documento a procesar. |
| **Responsabilidad** | Cargar el archivo con el texto a procesar. |
| **Precondiciones** | Archivo en formato de texto |
| **Poscondiciones** | Se obtiene listado de requisitos funcionales |

Tabla 8 Descripción del caso de uso Pre-Procesar texto.

|  |  |
| --- | --- |
| **Caso de uso** | Pre-procesar texto |
| **Actores** | Analista |
| **Propósito** | Procesar el contenido de forma tal que se ejecute una limpieza y normalización que posibilite el manejo de la información por la computadora. |
| **Descripción** | 1. Convertir en minúsculas 2. Eliminar signos de puntuación 3. Etiquetar cada tokens según clasificación gramatical |
| **Responsabilidad** | El sistema debe pre-procesar el contenido textual de forma tal que sea eliminado el ruido o información que no aporta |
| **Precondiciones** | Se debe recibir la información a procesar |
| **Poscondiciones** | Debe devolver el texto normalizado |

Tabla 9 Descripción del caso de uso Extraer requisitos candidatos

|  |  |
| --- | --- |
| **Caso de uso** | Extraer requisitos de candidatos |
| **Actores** | Analista |
| **Propósito** | Extraer los requisitos candidatos. |
| **Descripción** | Para la extracción de los requisitos candidatos se empelan dos técnicas de extracción de información:   1. Análisis sintáctico basado en patrones léxicos – sintácticos 2. Análisis de dependencias   Finalmente se integran ambas técnicas |
| **Responsabilidad** | El sistema debe extraer los requisitos candidatos aplicando las técnicas de extracción de información. |
| **Precondiciones** | Texto normalizado |
| **Poscondiciones** | Se obtiene listado de requisitos candidatos |

Tabla 10 Descripción del caso de uso Aplicar estrategia de reducción de redundancias.

|  |  |
| --- | --- |
| **Caso de uso** | Aplicar estrategia de reducción redundancias |
| **Actores** | Analista |
| **Propósito** | Reducir los requisitos candidatos para mejorar la calidad del método de extracción. |
| **Descripción** | Se creó una estrategia para la reducción de as redundancias basada en 2 criterios. |
| **Responsabilidad** | Filtrar listado de requisitos candidatos. |
| **Precondiciones** | Listado de requisitos candidatos. |
| **Poscondiciones** | Se obtiene listado de requisitos funcionales reducido. |

Tabla 11 Descripción del caso de uso Agrupar semánticamente.

|  |  |
| --- | --- |
| **Caso de uso** | Agrupar semánticamente |
| **Actores** | Analista |
| **Propósito** | Agrupar el listado de requisitos funcionales a partir de la similitud semántica. |
| **Descripción** | Formar clústers a partir de la similitud semántica entre requisitos. |
| **Responsabilidad** | Agrupar a partir de la semántica |
| **Precondiciones** | Listado de requisitos funcionales |
| **Poscondiciones** | Se obtienen los diferentes clústers |

Tabla 12 Descripción del caso de uso Exportar listado de requisitos funcionales.

|  |  |
| --- | --- |
| **Caso de uso** | Exportar listado de requisitos funcionales |
| **Actores** | Analista |
| **Propósito** | Mostrar al usuario el listado de requisitos funcionales que fueron extraídos. |
| **Descripción** | Exportar al usuario un archivo txt con los clústers formados a partir de la etapa de agrupamiento. |
| **Responsabilidad** | Mostrar al usuario resultados |
| **Precondiciones** | Listado de requisitos funcionales agrupados en clústers |
| **Poscondiciones** | Se obtienen archivo txt con requisitos funcionales |

Tabla 13 Descripción del caso de uso Extraer requisitos candidatos por método híbrido.

|  |  |
| --- | --- |
| **Caso de uso** | Extraer requisitos candidatos por método híbrido. |
| **Actores** | Analista |
| **Propósito** | Extraer requisitos candidatos a partir de la combinación de las dos técnicas |
| **Descripción** | A partir de la integración de las técnicas de extracción de información implementadas anteriormente extraer requisitos candidatos. |
| **Responsabilidad** | Extraer requisitos candidatos |
| **Precondiciones** | Texto pre-procesado |
| **Poscondiciones** | Se obtienen requisitos candidatos |

Tabla 14 Descripción del caso de uso Extraer requisitos por análisis de dependencias.

|  |  |
| --- | --- |
| **Caso de uso** | Extraer requisitos por análisis de dependencias |
| **Actores** | Analista |
| **Propósito** | Extraer requisitos haciendo uso de la técnica de extracción de información basada en análisis de dependencias. |
| **Descripción** | A partir del árbol de dependencias generado por el analizador sintáctico de la biblioteca SpaCy se extraen los requisitos candidatos a partir de la estructura VERB + nsubj + obj |
| **Responsabilidad** | Extraer requisitos candidatos |
| **Precondiciones** | Texto pre-procesado |
| **Poscondiciones** | Se obtienen requisitos candidatos |

Tabla 15 Descripción del caso de uso Extraer requisitos por patrones léxico-sintácticos.

|  |  |
| --- | --- |
| **Caso de uso** | Extraer requisitos por patrones léxicos – sintácticos |
| **Actores** | Analista |
| **Propósito** | Extraer requisitos haciendo uso de la técnica de extracción de información basada en patrones léxicos – sintácticos |
| **Descripción** | Extraer los requisitos candidatos haciendo un mapeo en el texto con los patrones léxico – sintácticos que se definieron en el estudio estadístico. |
| **Responsabilidad** | Extraer requisitos candidatos |
| **Precondiciones** | Texto pre-procesado |
| **Poscondiciones** | Se obtienen requisitos candidatos |

## Arquitectura del modelo

El estilo arquitectónico Llamada-Retorno también se encuentra presente a través del patrón N-Capas en la solución. Este enfoque en capas soporta el desarrollo incremental de sistemas. Conforme se desarrolla una capa, algunos de los servicios proporcionados por esta capa deben quedar a disposición de los usuarios. La arquitectura también es cambiable y portátil. En tanto su interfaz no varíe, una capa puede sustituirse por otra equivalente. Más aún, cuando las interfaces de capa cambian o se agregan nuevas facilidades a una capa, sólo resulta afectada la capa adyacente. A medida que los sistemas en capas localizan dependencias de máquina en capas más internas, se facilita el ofrecimiento de implementaciones multiplataforma de un sistema de aplicación. Sólo las capas más internas dependientes de la máquina deben reimplantarse para considerar las facilidades de un sistema operativo o base de datos diferentes [76].

Este enfoque se caracteriza por brindar una vista del sistema completo con los componentes divididos en 4 capas. La primera es la capa específica, que muestra los paquetes del sistema, que no son reutilizados. La siguiente es la capa general, quien muestra los elementos que son reutilizadosyforman parte también del sistema. La tercera capa es empleada para mostrar los paquetes, que no son propios del sistema que se está desarrollando, sino de terceros, son brindados por el entorno de desarrollo, componentes descargados de Internet, o Sistemas Legados, esta es la capa intermedia. Finalmente está la capa software del sistema, donde se recogen todos los protocolos, componentes y subsistemas que son provistos por el propio sistema operativo.

Teniendo en cuenta la descripción de las capas anteriormente expuesta, se pudiera decir que el sistema no presenta elementos en las capas específica y software de sistema, pues no tiene elementos que no puedan ser reutilizados por su especificidad o protocolos provistos por el sistema operativo. Sin embargo, en la capa general se pueden localizar los paquetes referentes a la carga de archivos, al pre-procesamiento, extracción de requisitos candidatos, reducción de redundancias, agrupamiento y exportar resultados. Como fue definido con anterioridad, el sistema reutiliza componentes brindados por terceros entre los que se encuentran los paquetes de SpaCy, WordNet y Scikit-lern los que se ubican en la capa intermedia.

A continuación, se presenta una ilustración que se corresponde con el diagrama de arquitectura en capas con enfoque basado en la reutilización que describe el modelado de la solución.

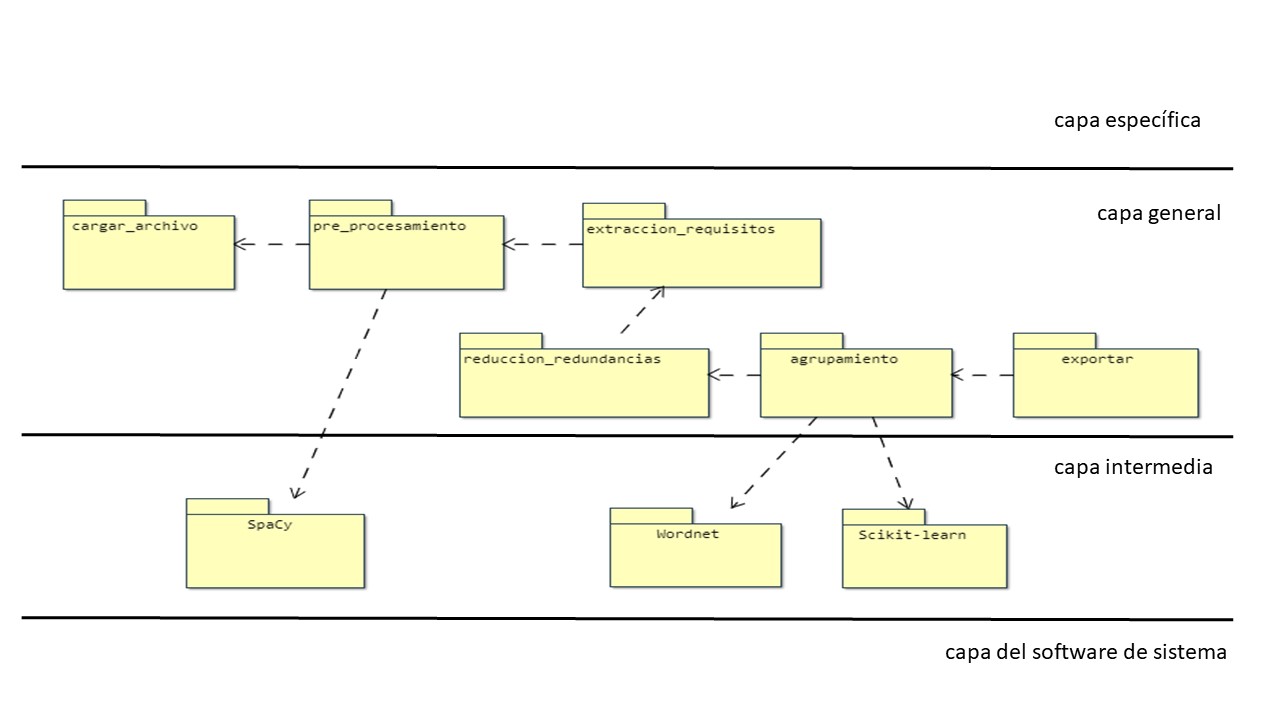


Ilustración 10 Estructura en capas del modelo.

## Estilos y patrones de arquitectura

**Arquitectura basada en componentes**:

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado [76].

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\_archivo: 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, fragmentación)
* extraccion\_requisitos: paquete donde se implementa las funciones de extracción de requisitos por análisis sintáctico basado en patrones léxicos-sintácticos y por análisis de dependencias.
* reducción\_redundancias: paquete donde se implementan la funcionalidad encargada de filtrar las frases extraídas en el componente anterior, basada en una estrategia de reducción de redundancias.
* agrupamiento: paquete donde se implementa la funcionalidad de agrupar los requisitos extraídos en clústers a partir de su relación semántica.
* exportar: paquete donde se exportan en archivo .txt el listado de requisitos funcionales.

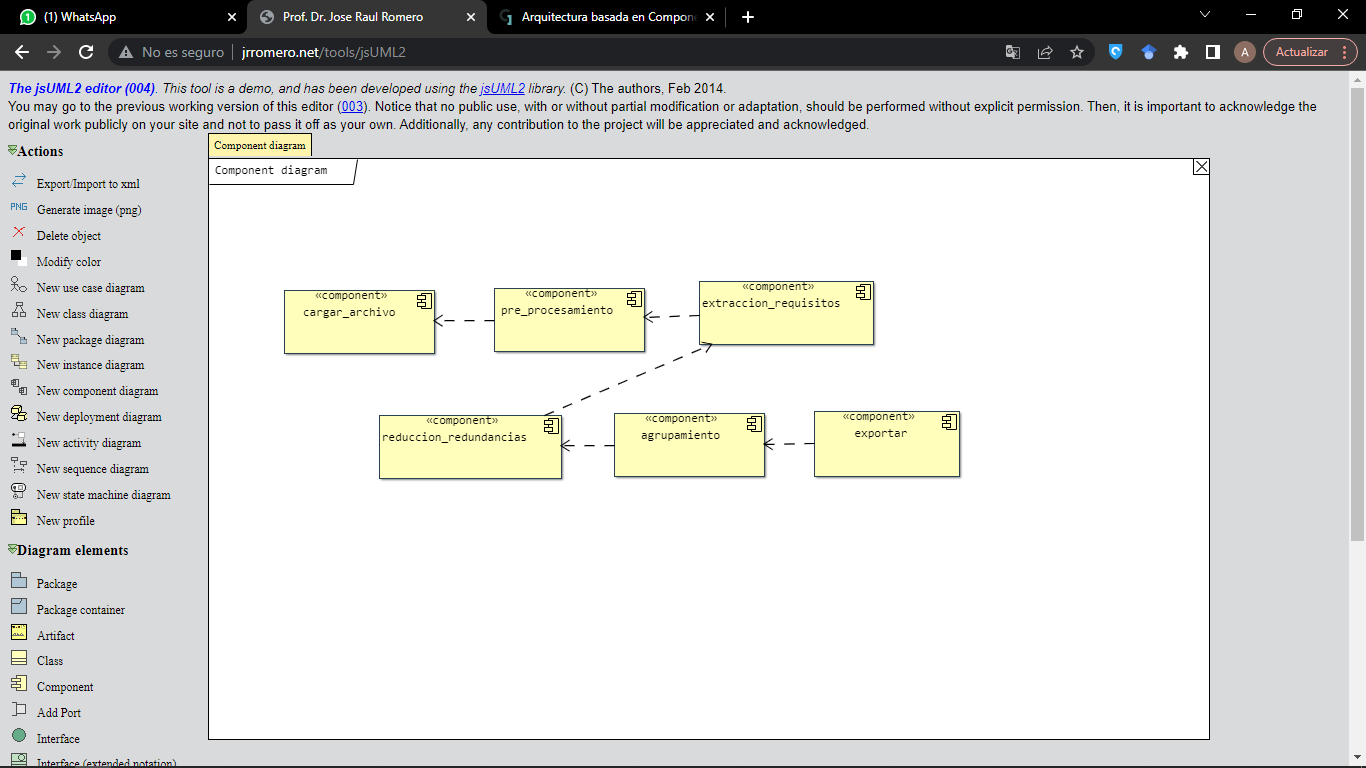


Ilustración 11 Estructura modular del sistema. Arquitectura basada en componentes.

**Patrón Filtros y Tuberías**:

La solución se modela siguiendo el patrón filtro y tubería del estilo arquitectónico Flujo de datos de entrada. Éste es un modelo de la organización en tiempo de operación de un sistema, donde las transformaciones funcionales procesan sus entradas y producen salidas. Los datos fluyen de uno a otro y se transforman conforme se desplazan a través de la secuencia. Cada paso de procesamiento se implementa como un transformador. Los datos de entrada fluyen por medio de dichos transformadores hasta que se convierten en salida [76].

Se concibe el tratamiento de los datos como una serie de transformaciones sobre diferentes etapas en el procesamiento de los datos de entrada, como se muestra en la *Ilustración 3.*

# Conclusiones parciales

Como conclusiones del capítulo se puede afirmar que:

* Fue desarrollado un método para la extracción automática de requisitos de software.
* Las técnicas de extracción de información usadas en el método desarrollado fueron análisis sintáctico basado en patrones léxicos sintácticos y análisis sintáctico basado en análisis de dependencias.
* La métrica de similitud semántica de WordNet, Wu-Palmer fue la seleccionada para formar los diferentes clústers en la etapa de agrupamiento.
* El algoritmo de agrupamiento Hierarchical Agglomerative fue el utilizado en la etapa de agrupamiento.
* La métrica Sihlouette fue la utilizada para medir la calidad de agrupamiento.

# 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).

# 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%.

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

* Analizar los resultados obtenidos 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

Las métricas seleccionadas para la evaluación de la propuesta son precisión, cobertura y medida-F. Estas métricas fueron generadas como parte de la investigación.

**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:

Ecuación 1 Ecuación para calcular Precisión

**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:

Ecuación 2 Ecuación para calcular Cobertura

**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:

Ecuación 3 Ecuación para calcular Medida-F

Métrica Levenshtein : 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 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.

## Descripción de la colección de pruebas

La solución propuesta fue evaluada tomando como referencias 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. 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.

Tabla 13 Descripción de casos de estudio.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***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 | 1074 | 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 | 2342 | 25 | turismo |
| Agencias de viajes | 96 | 2224 | 31 | turismo |
| Sistema de control para una bomba de insulina | 18 | 376 | 6 | salud |

# Resultados y discusión

En esta sección se sintetizan y analizan los resultados obtenidos en los experimentos realizados con la colección de casos de pruebas para satisfacer los objetivos de evaluación planteados. Las métricas de Precisión y Cobertura son computadas comparando el requisito que se obtiene extrayendo información del texto, con el elaborado manualmente por el experto. En esa comparación se empleó la distancia Levenshtein, usando como umbral de aceptación el 60%. Los resultados obtenidos en los experimentos realizados con la colección de prueba se muestran en la siguiente Tabla.

Tabla 14 Resultados de la evaluación del método en colección de prueba.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ***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** | **55.95** |
| **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 arroja el análisis basado en patrones léxicos sintácticos.
* El valor más elevado de *Cobertura* lo arroja en método que combina ambas técnicas de extracción de información.
* La medida-F también se aprecia su mayor valor en el enfoque híbrido.

A continuación, se presentan los gráficos 13, 14, 15 y 16 que complementan el análisis del comportamiento de los resultados:

Ilustración 12 Comportamiento de las métricas Precisión, Cobertura y Medida-F con el uso de Patrones léxicos-sintácticos

Ilustración 13 Comportamiento de las métricas Precisión, Cobertura y Medida-F con el uso de análisis de dependencias

Ilustración 14 Comportamiento de las métricas Precisión, Cobertura y Medida-F en el enfoque híbrido

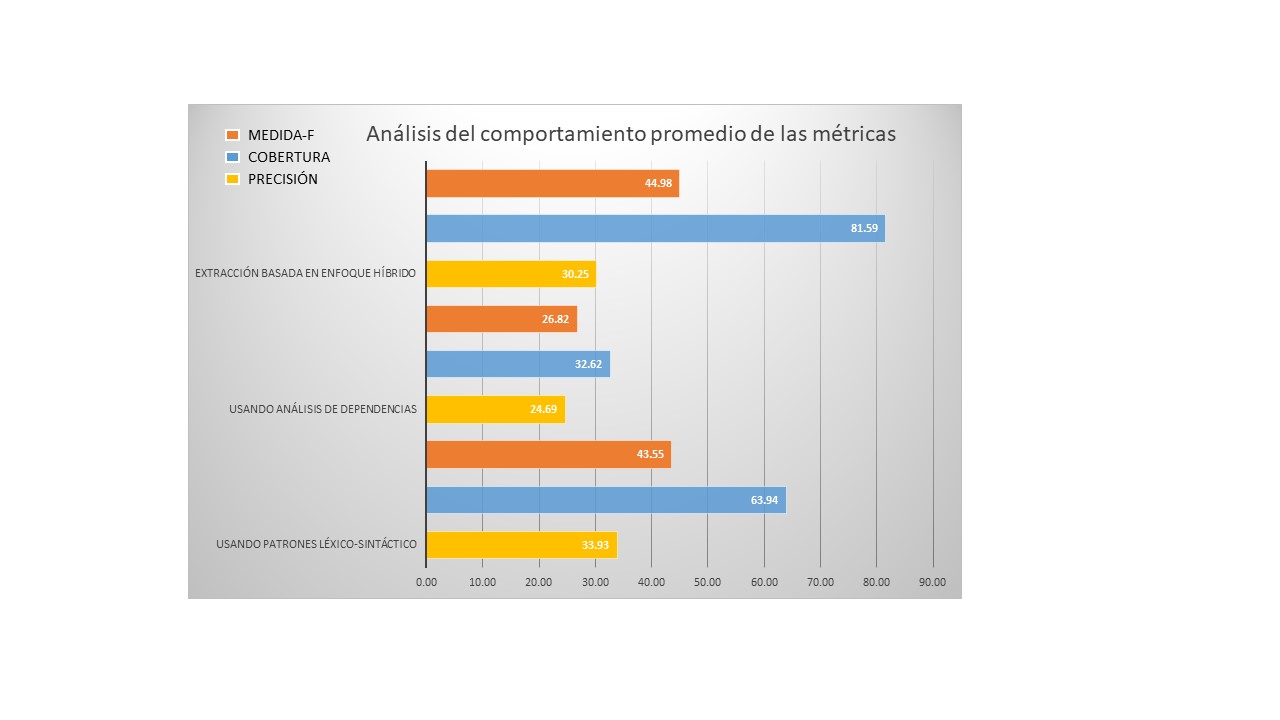


Ilustración 15 Análisis del comportamiento promedio de las métricas

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****,* esto 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 parciales:

A continuación, se exponen las conclusiones del capítulo:

* La colección de casos de prueba que se definió contó con 14 documentos de diferentes dominios, entre entrevistas y descripciones de procesos.
* La métrica de similitud sintáctica Levenshtein fue la seleccionada en la comparación entre los requisitos extraídos manualmente y los requisitos extraídos de forma automática.
* Fueron computadas las métricas ***Precisión, Cobertura*** y ***Medida-F***.
* En general la técnica de extracción de información basada en patrones léxico -sintácticos fue la técnica que mayores resultados aportó al proceso de extracción de requisitos funcionales.
* 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.***
* Las métricas de ***Cobertura*** y ***Medida-F*** reportan mayores resultados con el método que integra ambas técnicas de extracción de información.
* La técnica de extracción de información basada en análisis de dependencias no aporto ninguna relevancia entre las métricas en comparación con la técnica basada en patrones y el método que integra amabas técnicas.

# Conclusiones Generales

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

* Fue desarrollado un método para la extracción automática de requisitos de software.
* El diseño de solución que se propuso 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.
* Los resultados preliminares luego de ser realizada una evaluación a la solución con la colección de pruebas confeccionada por el experto, que reúne entrevistas y descripciones de procesos, y el conjunto requisitos funcionales extraídos manualmente, 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.
* Un mayor valor de presión se observa con el uso de la técnica de patrones, sin embargo, el mayor valor de cobertura y medida-F lo arroja la solución final que combina ambas técnicas de extracción.
* La técnica de extracción de información basada en análisis de dependencias no aporto ninguna relevancia entre las métricas en comparación con la técnica basada en patrones y el método que integra amabas técnicas.

# Recomendaciones

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

* Profundizar el estudio de extracción de requisitos de software con la técnica de extracción de información basada en análisis de dependencias.
* Extender la solución a la extracción de requisitos de calidad y requisitos de restricción.

# 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,» Universidad Nacional de San Agustín de Arequipa, Perú, 2019. |
| [3] | B. W. Boehm, «Software Engineering Economics,» *IEEE Transactions on Software Engineering,* vol. 10, pp. 4 - 21, 1984. |
| [4] | B. Boehm, «Verifying and Validating Software Requirements and Design Specifications,» *IEEE,* vol. 1, nº 1, p. 75–88, 1984. |
| [5] | 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. |
| [6] | 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. |
| [7] | R. Vlas y W. N. Robinson, «A Rule-Based Natural Language Technique for Requirements Discovery and Classification in Open-Source Software Development Projects,» de *44th Hawaii International Conference on System Sciences*, Kauai, HI, USA, 2011. |
| [8] | K. Ryan, «The role of natural language in requirements engineering,» de *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, CA, USA, 1993. |
| [9] | J. Hirschberg y C. D. Manning, «Advances in natural language processing,» *Science,* vol. 349, nº 6245, p. 261–266, 2015. |
| [10] | G. Goth, «Deep or shallow, NLP is breaking out,» *Communications of the ACM,* vol. 19, nº 3, p. 13–16, 2016. |
| [11] | A. Ferrari, F. Dell’Orletta, A. Esuli, V. Gervasi y S. Gnesi, «Natural language requirements processing: A 4D vision.,» *IEEE,* p. 28–35, 2017. |
| [12] | F. Dalpiaz, A. Ferrari, X. Franch y C. Palomares, «Natural Language Processing for Requirements Engineering: The Best Is Yet to Come,» *IEEE Software,* vol. 35, nº 5, p. 115–119, 2018. |
| [13] | D. M. Fernández, «Naming the pain in requirements engineering,» *Empirical Software Engineering,* vol. 22, p. 2298–2338, 2017. |
| [14] | I. Sommerville, Ingeniería de Software, Ciudad de México: Pearson Education, 2011. |
| [15] | C. Rupp y K. Pohl, Requirements Engineering Fundamentals, California, 2017. |
| [16] | R. S. Pressman, Ingeniería del Software. Un enfoque Práctico. Séptima Edición, Mexico, 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] | S. Richa, B. Jaspreet y B. K.K., «Automated identification of business rules in requirements documents,» de *IEEE International Advance Computing Conference (IACC)*, Gurgaon, India, 2014. |
| [21] | B. Ebrahim, E. Faezeh y G. Dragan, «Decision support for the software product line domain engineering lifecycle,» *Automated Software Engineering,* vol. 16, pp. 335-377, 2012. |
| [22] | M. Jason, S. Shahryar, H. Thomas y E. Timothy, «Software requirements prioritization and selection using linguistic tools and constraint solvers—a controlled experiment,,» *Empirical Software Engineering,* vol. 20, p. 1721–1761, 2015. |
| [23] | C. Duan, P. Laurent, J. Cleland-Huang y C. Kwiatkowski, «Towards automated requirements prioritization and triage,» *Requirements Engineering,* vol. 14, pp. 73-89, 2009. |
| [24] | J. Guo y D. Yang, «Data-efficient performance learning for configurable systems,» *Empirical Software Engineering,* vol. 23, p. 1826–1867, 2018. |
| [25] | C. Fitzgerald, E. Letier y A. Finkelstein, «Early failure prediction in feature request management systems: an extended study,» *Requirements Engineering,* vol. 17, p. 117–132, 2012. |
| [26] | J. d. Sagrado y I. d. Águila, «Stability prediction of the software requirements specification,» *Software Quality Journal,* vol. 26, pp. 585-605, 2018. |
| [27] | I. M. d. Águila y J. d. Sagrado, «Requirement risk level forecast using Bayesian networks classifiers,» *International Journal of Software Engineering and Knowledge Engineering,* vol. 21, nº 02, pp. 167-190, 2011. |
| [28] | Z. Kurtanović y W. Maalej, «Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning,» de *IEEE 25th International Requirements Engineering Conference (RE)*, Lisbon, Portugal, 2017. |
| [29] | A. Dekhtyar y V. Fong, «RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow,» de *IEEE 25th International Requirements Engineering Conference (RE)*, Lisbon, Portugal, 2017. |
| [30] | L. Zeheng y C. Mingrui, «Tracing requirements in software design,» de *International Conference on Software and System Process*, 2017. |
| [31] | C. Rolland y C. Proix, «A natural language approach for requirements engineering,» *In Advanced Information Systems Engineering,* vol. 593, p. 257–277, 2005. |
| [32] | K. Ryan, «The role of natural language in requirements engineering,» de *Proceedings of the IEEE International Symposium on Requirements Engineering*, San Diego, CA, USA, 1993. |
| [33] | R. J. A. a. D. K. Moorhead, «Software requirements and specifications: A survey of needs and languages,» *Journal of Systems and Software,* vol. 2, nº 4, pp. 297-316, 1981. |
| [34] | M. Luisa, F. Mariangela y N. I. Pierluigi, «Market research for requirements analysis using linguistic tools,» *Requirements Engineering,* vol. 9, p. 40–56, 2004. |
| [35] | M. Kassab, C. Neill y P. Laplante, «State of practice in requirements engineering: contemporary data,» *Innovations in Systems and Software Engineering,* vol. 10, p. 235–241, 2014. |
| [36] | J. H. Martín y D. Jurafski, «Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition,» *Prentice Hall,* 2000. |
| [37] | R. Grishman, «Information Extraction,» *IEEE Intelligent Systems,* vol. 30, nº 5, pp. 8-15, 2015. |
| [38] | R. Lomotey y R. Deters, «Real-Time Effective Framework for Unstructured Data Mining,» de *12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, Melbourne, VIC, Australia, 2013. |
| [39] | D. Nadeau y S. Sekine, «A survey of named entity recognition and classification,» *Lingvisticæ Investigationes,* vol. 30, nº 1, pp. 3-26, 2007. |
| [40] | M. Marrero, J. Urbano, S. Sánchez-Cuadrado, J. Morato y J. M. Gómez-Berbís, «Named Entity Recognition: Fallacies, challenges and opportunities,» *Computer Standards & Interfaces,* vol. 35, nº 5, pp. 482- 489, 2013. |
| [41] | Z. S. Abdallah, M. Carman y G. Haffari, «Multi-domain evaluation framework for named entity recognition tools,» *Computer Speech & Language,* vol. 43, pp. 34-55, 2017. |
| [42] | S. S. Sazali, N. A. Rahman y Z. A. Bakar, «Information extraction: Evaluating named entity recognition from classical Malay documents,» de *Third International Conference on Information Retrieval and Knowledge Management (CAMP)*, Malacca, Malaysia, 2016. |
| [43] | N. Konstantinova, «Review of Relation Extraction Methods: What Is New Out There?,» *Communications in Computer and Information Science ,* vol. 436, pp. 15-28, 2014. |
| [44] | K. Toutanova, D. Klein, C. D. Manning y Y. Singer, «Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network,» *Proceedings of HLT-NAACL,* pp. 173-180, 2003. |
| [45] | J. A. Miller, «WordNet: a lexical database for English,» *Comunications of the ACM,* vol. 38, nº 11, pp. 33-41, 1995. |
| [46] | G. Wilcock, «The Evolution of Text Annotation Frameworks,» *Handbook of Linguistic Annotation,* pp. 193-207, 2017. |
| [47] | B. Waltl, G. Bonczek y F. Matthes, «RULE-BASEDINFORMATIONEXTRACTION: ADVANTAGES,LIMITATIONS,ANDPERSPECTIVES,» 2018. |
| [48] | S. Cassidya y J. Harringtonab, «Multi-level annotation in the Emu speech database management system,» *Speech Communication,* vol. 33, nº 1-2, pp. 61-77, 2001. |
| [49] | F. Peng y A. McCallum, «Accurate Information Extraction from Research Papers,» *Inf Process Manag,* vol. 42, nº 4, pp. 963-979, 2006. |
| [50] | L. Medsker y S. Small, «The Siena College Medical Information Retrieval System (MIRS),» 2012. |
| [51] | R. Grishman y B. Sundheim, «Message Understanding Conference - 6: A brief history,» de *Proceedings of the 16th International Conference on Computational Linguistics.*, 1996. |
| [52] | F. D. M. Erik Tjong Kim Sang, «Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition,» de *In Proceedings of the 7th Conference on Natural Language Learning*, 2003. |
| [53] | R. T.-H. Tsai, S.-H. Wu, W. C. Chou, Y. C. Lin, D. él, J. Hsiang y Ting-Yi, «Various criteria in the evaluation of biomedical named entity recognition,» *BMC Bioinformatics,* vol. 7, nº 92, 2006. |
| [54] | J. Jiang y C. Zhai, «Exploiting Domain Structure for Named Entity Recognition,» de *Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL*, New York, 2006. |
| [55] | O. Etzioni, M. Banko, S. Soderland y D. S. Weld, «Open information extraction from the web,» *Communications of the ACM,* vol. 51, nº 12, pp. 68-74, 2008. |
| [56] | L. Qian, G. Zhou, F. Kong, Q. Zhu y P. Qian, «Exploiting constituent dependencies for tree kernelbased semantic relation extraction,» de *In Proceedings of the 22nd International Conference on Computational Linguistics*, 2008. |
| [57] | F. Dalpiaz, A. Ferrari, X. Franch y C. Palomares, «Natural Language Processing for Requirements Engineering,» *IEEE Software,* vol. 35, pp. 115-119, 2018. |
| [58] | W. Shira y B. Paul, «A Fully Automated Approach to Requirement Extraction from Design Documents,» de *IEEE Aerospace Conference (50100)*, Big Sky, MT, USA, 2021. |
| [59] | I. Hussain, L. Kosseim y O. Ormandjieva, «Using Linguistic Knowledge to Classify Nonfunctional Requirements in SRS documents,» *Lecture Notes in Computer Science,* vol. 5039, pp. 287-298, 2008. |
| [60] | C. Rolland y C. Salinesi, «Supporting Requirements Elicitation through Goal/Scenario Coupling,» *Conceptual Modeling: Foundations and Applications,* vol. 5600, p. 398–416, 2009. |
| [61] | S. Murugesh y A. Jaya, «Construction of Ontology for Software Requirements Elicitation.,» *Indian Journal of Science and Technology,* vol. 8, nº 29, 2015. |
| [62] | U. Shah, S. Patel y D. Jinwala, «Specification of non-functional requirements: A hybrid approach,» de *In 22nd International Working Conference on Requirements Engineering.* , Gothenburg, Sweden, 2016. |
| [63] | H. Meth, A. Maedche y M. Einoeder, «Is Knowledge Power? The Role of Knowledge in Automated Requirements Elicitation,» de *International Conference on Advanced Information Systems Engineering*, 2013. |
| [64] | Lili, «Research on User Requirements Elicitation Using Text Association Rule,» de *In International Symposium on Intelligence Information Processing and Trusted Computing.*, Huanggang, China, 2010. |
| [65] | G. Pablo y G. Marcos, «Dependency-Based Open Information Extraction,» de *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, Avignon, France, 2012. |
| [66] | A. C. M. A. Lorena Talamé, «Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python,» de *ASAI, Simposio Argentino de Inteligencia Artificial*, 2019. |
| [67] | S. Kübler, R. McDonald y J. Nivre, «Dependency Parsing. Synthesis Lectures on Human Language Technologies,» *Morgan & Claypool Publishers.,* p. 127, 2009. |
| [68] | H. Caseli, T. Pereira, L. Specia, T. Pardo, C. Gasperin y S. Aluisio, «Building a Brazilian Portuguese parallel corpus of original and simplified texts,» de *Center of Computational Linguistics*, Brasil, 2009. |
| [69] | J. Herrera, A. Peñas y F. Verdejo, «Textual Entailment Recognition Based on Dependency Analysis and WordNet,» *Part of the Lecture Notes in Computer Science book series,* vol. 3944, p. 231–239, 2005. |
| [70] | M. Ballesteros, R. Martín y B. D. Agudo, «JadaWeb: A CBR System for Cooking Recipes,» de *En Proceedings of Workshop on Computer Cooking Contest (ICCBR 2010)*, Italy, 2010. |
| [71] | C. Leacock y M. Chodorow, Combining local context and WordNet similarity for word sense identiﬁcation, In C. Fellbaum, editor, WordNet: An electronic lexical database, 1998. |
| [72] | Wu y Palmer, « Verb semantics and lexical selection,» de *In 32nd Annual Meeting of the Association for Computational Linguistics*, Mexico, 1994. |
| [73] | T. Pedersen, S. Patwardhan y J. Michelizzi, «WordNet::Similarity - Measuring the Relatedness of Concepts». |
| [74] | D. Mullner, «Modern hierarchical, agglomerative clustering algorithms,» de *stat.ML*, 2011. |
| [75] | I. Sommerville, Ingenieria de Software, Ciudad de Mexico: Pearson Education, 2011. |
| [76] | I. Sommerville, Ingeniería de Software, Ciudad de México: Pearson Education, 2011. |
| [77] | V. Teller, « Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition,» 2000. |

1. https://stanfordnlp.github.io/CoreNLP/ [↑](#footnote-ref-1)
2. https://www.nltk.org/ [↑](#footnote-ref-2)
3. https://nlp.lsi.upc.edu/freeling/node/1 [↑](#footnote-ref-3)
4. http://www.itl. nist.gov/iad/mig/tests/ace/ [↑](#footnote-ref-4)
5. https://www.python.org/ [↑](#footnote-ref-5)
6. https://www.jetbrains.com/help/pycharm/quick-start-guide.html [↑](#footnote-ref-6)
7. https://jupyter.org/ [↑](#footnote-ref-7)
8. https://es.acervolima.com/diferencia-entre-jupyter-y-pychar [↑](#footnote-ref-8)
9. https://spacy.io/ [↑](#footnote-ref-9)
10. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette\_score.html [↑](#footnote-ref-10)
11. https://www.ibm.com/docs/es/ida/9.1.2?topic=types-domain-models [↑](#footnote-ref-11)