Universidad de las Ciencias Informáticas

Facultad 1

Título: Clasificación automática de opiniones con técnicas de aprendizaje profundo para la elicitación de requisitos de software

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Marcos Antonio Batista Zaldívar

Tutor: P. Aux Ing. Vladimir Milián Núñez

Co-tutor: Michel Pedrera Suen

La Habana, 2021

ÍNDICE

RESUMEN

La elicitación de requisitos constituye una etapa vital del proceso de desarrollo de software y contribuye en gran medida al éxito o fracaso del mismo. El proceso convencional de elicitación es lento y propenso a cometer errores; además de ser incompleto. Con el fin de mejorar la calidad y eficiencia del mismo se ha propuesto el empleo de opiniones generadas dinámicamente por usuarios del producto como potencial fuente de propuestas para requisitos de software. Con el auge del internet, las redes sociales y otros sitios en línea, donde se mueven grandes volúmenes de información en pequeños lapsos de tiempo, constituyen una fuente invaluable de datos dinámicos antropogénicos para el proceso de elicitación de requisitos. El análisis adecuado de esta información requiere el empleo de técnicas de procesamiento de lenguaje natural y aprendizaje automático con el fin de garantizar la automatización y agilidad del procesamiento de los datos. El empleo de estas tecnologías podría mejorar considerablemente la elicitación de requisitos sobre datos dinámicos antropogénicos.

Palabras Clave

elicitación de requisitos, aprendizaje automático, procesamiento de lenguaje natural, clasificación automática de opiniones

ABSTRACT

Requirements elicitation is a vital stage in the software development process and contributes greatly to its success or failure. The conventional elicitation process is slow and prone to mistakes; besides being incomplete. In order to improve its quality and efficiency, the use of opinions generated dynamically by users of the product has been proposed as a potential source of software requirements proposals. With the rise of the internet, social networks and other online sites, where large volumes of information move in small periods of time, constitute an invaluable source of dynamic human-generated data for the requirements elicitation process. Proper analysis of this information requires the usage of natural language processing and machine learning techniques in order to ensure the automation and agility of data processing. The use of these technologies could significantly improve the process of requirements elicitation over dynamic human-generated data.

Key Words

requirement elicitation, machine learning, natural language processing, automatic opinion classification

INTRODUCCIÓN

La ingeniería de requisitos es una **fase determinante** en el desarrollo de software y es un factor clave en el éxito o fracaso de un proyecto. En esta etapa se establecen las especificaciones funcionales y no funcionales que el cliente y los futuros usuarios esperan del software a desarrollar. La ingeniería de software se divide en cuatro etapas: elicitación, análisis, especificación y administración. De estas la primera constituye la piedra angular del proceso, y es en la que se centrará todo el trabajo.

Convencionalmente es el analista de sistemas, o ingeniero de requisitos, el encargado de realizar el proceso de elicitación. Las técnicas utilizadas para el completamiento de esta etapa dependen enteramente del experto a cargo, de la empresa donde labore y de las condiciones particulares del proyecto mismo; por esta razón un análisis exhaustivo en esta área puede abarcar cientos de técnicas diferentes que pueden ser y han sido empleadas en la elicitación de requisitos. [ZOWG05] Conocer todas las técnicas y propuestas no es, sin embargo, escencial para la investigación corriente.

La elicitación manual de requisitos debe enfrentar tres problemas fundamentales, según Christel y Kang en [CHRI92]: problemas de alcance, donde los requisitos manejan demasiada o muy poca información; problemas de comprensión entre las partes involucradas y los desarrolladores; y problemas de volatilidad debido a la naturaleza cambiante de los requisitos. La existencia de tantas y tan variadas fuentes de error explica por qué esta etapa del proceso de desarrollo suele consumir tanto tiempo: pues una equivocación en esta puede repercutir en las etapas posteriores retrasando todo el proyecto o generando grandes pérdidas y consumo de recursos.

Según Zowghi et al. en [ZOWG05], en la **elicitación de requisitos** tradicional los requisitos son escogidos de un **dominio de conocimiento** obtenido del cliente, principalmente a través de métodos para la recolección de datos de calidad (ej. Entrevistas, talleres, discusiones grupales, revisión de documentos). Sin embargo, la creciente digitalización de la sociedad y las compañías ha creado una basta fuente de datos de naturaleza heterogénea y dinámica comúnmente conocida como **Big Data**, que se origina en las redes. Toda esta información se debe considerar como un recurso valioso a la hora de establecer los requisitos, en adición al dominio de conocimiento.

Estudios como el de Zhara S. H. Abad et al. [ABAD17] han comprobado que tomar en cuenta las opiniones de los usuarios puede tener gran influencia en los requisitos funcionales y no funcionales descubiertos. En su investigación varios equipos debían realizar el proceso de elicitación de requisitos antes y después de tomar en cuenta las opiniones de los usuarios. Todos los equipos reportaron modificaciones en sus requisitos no funcionales y cerca de la mitad (46 %) también reportó cambios en los requisitos funcionales. Esto refuerza la importancia de incluir fuentes no tradicionales de información para el proceso de elicitación.

Analizar tanto las fuentes de datos tradicionales como las nuevas fuentes (generalmente públicas y gratuitas), constituyen una forma confiable de mejorar la calidad de los sistemas y facilitar el desarrollo de software. No obstante las técnicas tradicionales son poco escalables y adaptables, así como grandes consumidores de tiempo. Por esto es necesario un enfoque orientado a los datos que permita un proceso continuo y automático de ingeniería de requisitos sobre un mar de información en constante crecimiento.

Existen numerosos intentos de crear herramientas para la elicitación automática de requisitos desde datos estáticos: con crecimiento lento y rara actualización. Sin embargo se ha otorgado mucho menos foco a los estudios sobre la elicitación de requisitos sobre fuentes de datos dinámicas. [LIM21] De los trabajos que caen en esta última categoría se puede apreciar como ha habido una tendencia al incremento durante los últimos años en el número de publicaciones que estudian alguna forma de elicitación automática de requisitos. [Anexo 1] Lim y Finkelstein en [LIM21] recogieron 68 estudios que, en términos generales, han alcanzado muy buenos resultados y realizado avances en la elicitación automática de requisitos sobre datos dinámicos. La mayoría de los estudios se centran en el procesamiento de texto en Inglés y no se ha investigado si la replicación de esas técnicas dará resultados similares en sistemas basados en el idioma Español.

La idea principal del enfoque orientado a datos es aplicar técnicas de aprendizaje automático y procesamiento de lenguaje natural a las opiniones emitidas por los usuarios en las redes con el fin de extraer de ellos requisitos de software. La propia naturaleza del lenguaje natural, en que el contexto de una palabra es importante para su significado; así como las cantidades cada vez mayores de opiniones que se realizan a diario; hace que los métodos convencionales para analizarar dicha información no puedan mejorar los resultados obtenidos. No obstante las técnicas de aprendizaje automático han demostrado ser

eficientes al procesar grandes volúmenes de información. Además, el desarrollo de nuevas formas de representación de texto, basadas en estas técnicas, han demostrado ser eficientes a la hora de manejar el problema del contexto, a niveles aceptables. Esta situación, a priori, parece ser una buena solución a las limitaciones de las propuestas existentes, sobre todo, a la hora de clasificar las opiniones para extraer requisitos.

Dada esta información se propone el siguiente **problema científico** como centro de la investigación corriente: ¿Cómo mejorar el proceso de elicitación de requisitos de requisitos de software basado sobre datos dinámicos?

Para dar solución a este problema se investigará en el marco del siguiente **objeto de estudio**:

- Elicitación de Requisitos basada en datos antropogénicos

Centrándo el estudio sobre el siguiente **campo de acción**:

- La elicitación automática de requisitos de software a partir de datos dinámicos antropogénicos

Con el fin de dar solución al problema planteado se propondrán los siguientes objetivos a seguir como pautas para la investigación corriente:

Objetivo general: Evaluar varias técnicas de aprendizaje automático para mejorar el proceso de elicitación de requisitos de software a partir de datos dinámicos antropogénicos.

Objetivos específicos:

- Analizar los principales conceptos, contribuciones y tecnologías relacionadas con el aprendizaje automático en la elicitación de requisitos sobre datos dinámicos.
- Realizar una propuesta de diferentes métodos de aprendizaje automático para mejorar el proceso de elicitación de requisitos sobre datos dinámicos antropogénicos.
- Validar la propuesta de solución mediante la aplicación experimentos y pruebas de software.

Del análisis efectuado se desprende la siguiente **hipótesis**: El desarrollo de un método que aproveche las técnicas de aprendizaje automático para procesar datos dinámicos antropogéncos y automatizar

parcial o totalmente el proceso de elicitación de requisitos de software contribuirá al mejoramiento de dicho proceso en la producción de software.

De la hipótesis anterior se pueden extrapolar la **variable independiente**: método de procesamiento de datos dinámicos antropogénicos y clasificación de requisitos; y las **variables dependientes**: precisión y regresión, cuya operacionalización será definida y explicada en el capítulo 3.

Se empleará la metodología KDD (*Knowledge Discovery on Databases*) como método científico de investigación para dar cumplimiento a los objetivos propuestos. A dicha metodología se añadirán métodos de investigación para conformar el siguiente esquema:

(KDD, paso **1**) Desarrollar una comprensión del dominio de la aplicación: Prepara el escenario para comprender qué se debe hacer con las múltiples decisiones (sobre transformación, algoritmos, representación, etc.). En esta etapa se debe comprender y definir el entorno en el que se llevará a cabo el proceso de descubrimiento de conocimientos (incluido el conocimiento previo relevante).

- **Analítico-Sintético**: para analizar independientemente las principales fuentes bibliográficas dentro del área de estudio tratada; lo que permite profundizar en cada uno de los textos disponibles. Aquí se identifican los elementos primordiales para dar solución al problema.

- **Histórico-Lógico**: para estudiar la trayectoria y desarrollo de las herramientas, conceptos y técnicas relacionadas con la elicitación de requisitos basada en el análisis de datos dinámicos. Aquí se identifican los aportes realizados en este campo, así como las tendencias actuales.

(KDD, paso **2**) Seleccionar y crear el conjunto de datos sobre los que se efectuará el descubrimiento: Determina los datos que se utilizarán para el descubrimiento de conocimientos. Esto incluye averiguar qué datos están disponibles, adicionar los datos necesarios y luego integrar todos los datos para el descubrimiento de conocimientos en un único conjunto de datos que incluiya todos los atributos que se considerarán para el proceso.

(KDD, paso **3**) Preprocesamiento y limpieza: para mejorar la confiabilidad de los datos. Esta etapa incluye la depuración de los datos mediante el manejo de valores perdidos y la eliminación de ruido o valores atípicos.

(KDD, paso **4**) Transformación de datos: para preparar y desarrollar la generación de datos más aptos para la minería de datos. Los métodos aquí incluyen la reducción de dimensiones (como la selección y extracción de carácterísticas y el muestreo de registros) y la transformación de atributos (como la discretización de atributos numéricos y la transformación funcional).

(KDD, paso **5**) Elección de la tarea de minería de datos adecuada: para decidir qué tipo de minería de datos usar (clasificación, regresión, agrupamiento, etc.)

(KDD, paso **6**) Elección del algoritmo de minería de datos: para seleccionar el método específico que se utilizará para buscar patrones.

(KDD, paso **7**) Emplear el algoritmo de minería de datos: para realizar varias iteraciones empleando el algoritmo seleccionado, realizando los ajustes pertinentes, hasta obtener un resultado satisfactorio.

(KDD, paso **8**) Evaluación: Para evaluar e interpretar los patrones minados (reglas, confiabilidad, etc.), con respecto a las metas definidas. Aquí se consideran los pasos de preprocesamiento con respecto a su efecto en los resultados del algoritmo de minería de datos (por ejemplo, se pueden agregar carácterísticas en el paso 4 y repetir todo elproceso desde allí). En este paso se documenta el conocimiento descubierto para su uso posterior.

(KDD, paso **9**) Uso del conocimiento descubierto: para incorporar el conocimiento en otro sistema para acciones futuras. El conocimiento se vuelve activo en el sentido de que podemos realizar cambios en el sistema y medir los efectos. Este es un paso con perspectivas futuras y cuya implementación no viene recogida en el estudio.

El resto de la investigación tendrá la siguiente estructura:

CAPÍTULO 1. Fundamentación teórica: se abordan los elementos que conforman el marco teórico y que componen el objeto de estudio de la investigación. Se tratan los problemas que afectan la correcta identificación de requisitos de software en datos dinámicos. Se presentan los principales métodos y técnicas para el análisis de opiniones usando técnicas de procesamiento de lenguaje natural y aprendizaje automático.

CAPÍTULO 2. Método para la extracción de requisitos de software: se fundamenta y describe el método propuesto para la identificación de requisitos de software a partir del análisis de datos dinámicos. Se describen las herramientas y tecnologías propuestas para la implementación de la propuesta de solución.

CAPÍTULO 3. Validación y análisis de resultados: se describen los resultados de la validación del método propuesto a partir de la realización de experimentos. Se describen las configuraciones experimentales y se presenta un análisis de los resultados obtenidos.

Finalmente se presentan las **Conclusiones**, se emiten las **Recomendaciones** derivadas de la investigación, se listan las **Referencias Bibliográficas** consultadas y los anexos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se expondrán los conocimientos adquiridos durante la consulta de la bibliografía. En la sección 1.1 se explicarán los pormenores de la ingeniería de requisitos, las técnicas convencionales empleadas en la misma y sus limitaciones.

En la sección 1.2 se extiende el concepto de ingeniería de requisitos a la plataforma informática, usando datos dinámicamete generados por usuarios en línea.

En la sección 1.3 se abarcan los diferentes trabajos que han tratado con anterioridad la elicitación automática de requisitos sobre datos dinámicos, y se exploran sus aportes en el campo y sus limitaciones.

La sección 1.4 consiste de una breve disertación sobre el procesamiento de lenguaje natural y su importancia en el procesamiento de datos antropogénicos.

Finalmente la sección 1.5 tratará los pormenores del uso del aprendizaje automático en la elicitación automática de requisitos, sus principales métodos y herramientas.

1.1 Ingeniería de Requisitos

El término **Requisito** será utilizado de acuerdo a la definición dada por la IEEE en [IEEE90]: (Un requisito es) **(1)** una condición o capacidad necesitada por un usuario para resolver un problema o lograr un objetivo; **(2)** una condición o capacidad que debe ser cumplida o poseída por un sistema o componente del sistema para satisfacer un contrato, estándar, especificación u otros documentos impuestos formalmente; **(3)** una representación documentada de una condición o capacidad como en **(1)** o **(2)**. Y serán distinguidos en requisitos funcionales y no funcionales, siendo estos últimos clasificados en requisitos de rendimiento / fiabilidad, interfaces y restricciones de diseño.

De acuerdo a la definición otorgada por Gea et al. an [GEA11], la ingeniería de requisitos es el enfoque disciplinado y sistemático para obtener, especificar, analizar, comprometer, validar y gestionar los requisitos al tiempo que se consideran las necesidades y objetivos orientados al negocio del usuario, técnicos y la economía de las partes. Abarca todo el ciclo de vida del proyecto, a menudo involucrando equipos distribuidos y cadenas de suministro.

1.1.1 Elicitación de requisitos de software

La elicitación de requisitos de software es un proceso iterativo que, simplificadamente, puede resumirse como el aprendizaje y comprensión de las necesidades de los usuarios y patrocinadores de un proyecto; que apunta a la comunicación concisa de estas necesidades a los desarrolladores de sistemas. [ZOWG05] Descubrir correctamente los requisitos de software es una parte vital, sin embargo sumamente compleja, en el proceso de desarrollo; y constituye los cimientos del mismo. Christel y Kang [CHRI92], basándose en un estudio de Rzepka, en 1989, descomponen el proceso en cinco etapas carácterísticas:

1. Identificar las partes relevantes que serán fuentes de requisitos. (Ej. La fuente puede ser un usuario final, un sistema de interfaz o factores ambientales)

2. Reunir la “lista de deseos” de cada parte relevante. Es probable que esta lista de deseos contenga ambigüedades, inconsistencias, requisitos inviables y requisitos no comprobables; además de estar probablemente incompleta.

3. Documentar y perfeccionar la "lista de deseos" para cada parte relevante. La lista de deseos incluye todas las actividades y datos importantes, y durante esta etapa debe analizarse repetidamente hasta que sea autoconsistente. La lista suele ser de alto nivel, específica para el dominio del problema relevante y expresada en términos específicos del usuario.

4. Enfrentar las listas de deseos con las diferentes partes relevantes, que serán denominadas puntos de vista, resolviendo así los conflictos entre los puntos de vista. La comprobación de la coherencia es una parte importante de este proceso. En esta etapa también se comprueba la viabilidad de las listas de deseos u objetivos.

5. Determinar los requisitos no funcionales, como problemas de rendimiento y confiabilidad, e indicarlos en el documento de requisitos.

Estas etapas no siguen un estándar en su práctica; y la forma en que se llevan a cabo puede estar fuertemente influenciada por factores ambientales del proyecto de software: como el presupuesto y los cronogramas establecidos. El producto resultante de una iteración de este proceso es un conjunto de requisitos que constituye una posible representación de los especificaciones de los putos de vista

involucrados. Posteriormente, en el proceso de ingeniería de requisitos, debe determinarse si coinciden exactamente con las pretenciones iniciales del cliente y usuario, para ser validadas o refinadas/reelectas en una nueva iteración.

1.1.2 Criterios de calidad para los requisitos. Obstáculos

El objetivo de la elicitación de requisitos es la obtención de requisitos con buena calidad. De acuerdo a la IEEE en [IEEE83]; para que una selección de requisitos sea de buena calidad, debe ser:

- **Inequívoca**: No deben existir ambigüedades. Cada especificación debe tener una única interpretación. Cada carácterística del producto final debe ser referenciada usando un único término. En los casos donde la interpretación de un término dependa del contexto o pueda ser comprendida de diferentes formas por diferentes intérpretes, este debe definirse con precisión.

- **Completa**: La selección de requisitos debe abarcar todas las funcionalidades y especificaciones no funcionales relevantes. Debe contener todas las respuestas a todos los tipos posibles de entrada de datos y en todas las situaciones posibles, dentro del marco conocido del proyecto. Debe contener un completo etiquetado; referencias a todas las figuras, tablas y diagramas; y definiciones de todos los términos y unidades de medida.

- **Verificable**: Debe existir para cada requisito un proceso concreto de costo efectivo finito mediante el cual una persona o máquina pueda verificar que el requisito hs sido satisfecho por el software desarrollado.

- **Consistente**: No debe haber contradicciones entre los requisitos. Significa que dos requisitos que describan el mismo objeto o proceso lógico no deben establecer términologías o parámetros contradictorios.

- **Modificable**: Su estructura y estilo debe facilitar su edición. Debe estar organizado coherentemente y ser simple de usar. Debe contener una tabla de contenidos, índices y referencias cruzadas. Debe evitar la redundancia.

- **Rastreable**: El origen de cada requisito debe ser claro y estar correctamente referenciado. Cada requisitodebe poseer un identificador único.

- **Utilizable durante la fase de operaciones y mantenimiento**: Está muy relacionado con su rastreabilidad y modificabilidad. Debe especificar las previsiones especiales que aplican a componentes individuales de la selección de requesitos, como su criticidad, relación con necesidades temporales y origen.

Christel y Kang [CHRI92] extienden el concepto añadiendo una nueva dimención:

- **Necesaria**: Cada requisito solo representa información pertinente al desarrollo de la solución.

En [CHRI92] plantean que la elicitación de requisitos debe enfrentar tres problemas fundamentales: problemas de alcance, donde los requisitos manejan demasiada o muy poca información; problemas de comprensión entre las partes involucradas y los desarrolladores; y problemas de volatilidad debido a la naturaleza cambiante de los requisitos.

1.2 Ingeniería de Requisitos sobre datos dinámicos

Según Zowghi et al. en [ZOWG05] existen varias técnicas convencionales para incluir las opiniones de los usuarios en el proceso de elicitación de requisitos. Algunos de los métodos convencionales incluye: la creación de **cuestionarios**, que son empleados en las etapas tempranas de la elicitación de requisitos y pueden estar constituidos por preguntas abiertas y/o cerradas (Para que sean efectivos, los términos, conceptos y límites del dominio han de estar bien establecidos y ser bien comprendidos por los participantes y el diseñador del cuestionario). Otro popular método manual es la etnografía, o el estudio de las personas en su entorno natural; en este caso el analista participa activa o pasivamente en las actividades normales de los usuarios durante un período prolongado de tiempo mientras recopila información sobre las operaciones que se realizan.

El concepto **Información** resulta complejo de definir dada su dependencia del contexto. Sus acepciones más aceptadas son: la información como conocimiento archivado (tradicionalmente registrada en forma de libros, pinturas, fotografías; aunque la llegada de la era digital ha portado prácticamente toda la información al formato electrónico); la información como percepción del entorno (que refiere a los datos

que recopilan los seres vivos, principalmente los humanos, utilizando los sentidos; y que luego procesan y transforman en conocimiento); información como recurso (la información se transmite de un emisor a un receptor y cada una de las partes involucradas hace uso de la misma en el proceso de toma de decisiones). [MADD00] En lo que concierne al documento corriente, **información** hace referencia a datos en formato digital, principalmente texto en lenguaje natural y/o documentos generados por la empresa y que promisoramente puedan ser empleados en el proceso de elicitación de requisitos.

1.2.1 Fuentes de datos dinámicos

En la comunidad de investigadores no existe un concenso general en cuanto a la definición exacta de **Big Data**. En lo que concierne a este documento el término será definido según la definición que dan Firmani et al. en [FIRM16]: conjuntos de datos estructurados o no estructurados que son imposibles de almacenar y/o procesar empleando herramientas convencionales de software (ej. bases de datos relacionales), independientemente de la potencia de cómputo o del almacenamiento físico disponible. Se caracteriza por las palabras claves: volumen (tamaño de los datos), velocidad (tasa de aprovisionamiento de datos y tiempo dentro del cual es necesario actuar sobre ellos) y variedad (heterogeneidad de la adquisición de datos, representación e interpretación semántica). Existen tres fuentes fácilmente diferenciables de Big Data: **Información antropogénica** (el término hará referencia a toda la información de origen humano, concentrando la atención del documento en la información en formato digital como publicaciones en redes sociales, blogs, comentarios, mensajes de texto, etc.); **Información generada por el proceso** (es información generada por el negocio; ej. Datos de transacciones, registros de clientes, pagos y servicios, etc.); **información generada por máquinas** (refiere a los diversos datos generados por sensores y maquinaria especializada usados para medir eventos y situaciones en el mundo físico).

1.2.2 Tipos de datos dinámicos

En el trabajo realizado por Lim et al. [LIM21] se concluyó que existen al menos siete tipos principales de datos dinámicos que han sido explotados con anterioridad para la elicitacón automática de requisitos de software. Ordenados desde el más comunmente utilizado hasta el menos frecuente estos son: reseñas en línea (revisiones de un producto o servicio determinado que las personas que lo han comprado elaboran y muestran públicamente en línea), microblogs (son normalmente publicados en redes sociales; la información puede venir en diferentes formatos de contenido, como textos cortos, audios, videos e

imágenes; y están diseñados para interacciones conversacionales rápidas entre usuarios), discusiones / foros en línea (son sitios de discusión en línea donde las personas pueden publicar mensajes para intercambiar conocimientos), repositorios de software (son plataformas para compartir paquetes de software o códigos fuente, que contienen principalmente tres elementos: un tronco, ramas y etiquetas; aquí se incluyen los sistemas de seguimiento de problemas, que son informes detallados de errores o quejas escritos en formato de texto libre), descripciones de producción de software / aplicaciones, lecturas de sensores (salidas eléctricas de dispositivos que detectan y responden a las entradas de un fenómeno físico, lo que da como resultado una gran cantidad de datos de transmisión), datos de uso de interacciones entre el sistema y el usuario (son datos en tiempo de ejecución recopilados cuando los usuarios interactúan con un sistema determinado), y listas de correo (Los mensajes de correo electrónico enviados por suscriptores específicos son compartidos por todos en una lista de distribución).

1.2.3 Alcance de la elicitación automática

Entre las herramientas analizadas en [LIM21] los resultados finales de los métodos empledos suelen escalar entre tres niveles de completamiento: identificación y clasificación de la información relacionada con los requisitos, identificación de carácterísticas candidatas relacionadas con los requisitos y, directamente, obtención de requisitos. Cada una de los niveles es considerablemente más complejo que el anterior y requiere de disímiles pasos extras para llegar de uno a otro.

Los niveles de completamiento afectan directamente el grado de automatización del artefacto pues inciden en qué parte del proceso requiere interacción humana; ya sea solo como supervisor o para completar la parte no automatizada del proceso.

1.3 Estado del arte

Han existido contados proyectos, con una tendencia incremental desde el 2012, que han investigado las posibilidades de la automatización de requisitos de software mediante el análisis de datos dinámicos, generalmente antropogénicos. Esta seccción revisará los avances y problemas principales de los estudios realizados hasta la fecha.

1.3.1 Pasos comunes

Dado que los datos de origen humano generalmente se expresan en lenguaje natural, el Procesamiento de Lenguaje Natural (PLN) se usa comúnmente para analizar este tipo de datos. Los estudios que utilizaron información de origen humano iniciaron el proceso de obtención de requisitos mediante el preprocesamiento de los datos sin procesar mediante técnicas de PLN. El preprocesamiento de datos generalmente implica eliminar el ruido (por ejemplo, etiquetas HTML, signos de puntuación) para retener solo datos de texto. Otra actividad crítica de preparación de datos es la tokenización, que significa dividir el texto en oraciones y tokens (palabras, signos de puntuación y dígitos), respectivamente.

1.3.2 carácterísticas de los otros proyectos

Un estudio reciente [LIM21] acerca del estado del arte de la elicitación automática de requisitos sobre datos dinámicos, tras una profunda búsqueda, reunió 68 investigaciones independientes (entre 2012 y 2019, con una excepción en 2009) que reunían los criterios pertinentes (trabajaban sobre datos dinámicos, no trabajaban con requisitos previamente elicitados, automatizaban el proceso hasta uno de los tres niveles de completamiento previamente mencionados y presentaban suficiente evaluación).

La investigación dio como resultado que el 93 % de los estudios se basaron en datos dinámicos antropogénicos. Del total de los estudios, más de la mitad (53 %) utilizó revisiones en línea como la fuente principal para su investigación. El resto de las fuentes empleadas ordenadas descendientemente según su popularidad fueron: micro-blogs (18 %), discusiones en línea (12 %), repositorios de software (10 %) y descripcciones del producto (7 %).

1.3.3 Herramientas e investigaciones existentes

(2012) StakeRare : Using social networks and collaborative Filtering for Large-Scale Requirements Elicitation

La principal contribución del proyecto [LIM12] fue la creción del método StakeRare; el cual permite la obtención de requisitos en grandes proyectos de software. Es una de las primeras aplicaciones de las redes sociales y el filtrado colaborativo para identificar y priorizar las partes interesadas y sus requerimientos. Este trabajo fue pionero en tres formas significativas de evaluación: la comparación con

los métodos de elicitación existentes utilizados en el proyecto, la comparación con la verdad básica construida a partir del conocimiento posterior al proyecto y el uso de medidas estadísticas estándar de la literatura de recuperación de información.

(2015) Towards Automatic Requirements Elicitation from Feedback Comments: Extracting Requirements Topics Using LDA

En el estudio [TAKA15] los autores utilizaron reseñas de la lista de usuarios de *Apache Commons* y de la *App Store*. Sin embargo, esos dos tipos de conjuntos de datos se utilizaron de forma independiente, sin integrarse para evaluar el proceso de obtención propuesto. Utilizaron modelado de tópicos mediante el algoritmo Latent Dirichlet Allocation (LDA) sobre reseñas y listas de correo para demostrar que era posible la clasificación de revisiones de los usuarios en comentarios que incluyen requisitos de software y aquellos que no. Como conclusión se determinó que era posible, aunque era menester aplicar ciertas restricciones entre las que debe considerarse la determinación del tamaño del tema y la modificación de la lista de palabras vacías (*stop words*).

(2017) Learn more, pay less! lessons learned from applying the wizard-of-oz technique for exploring mobile app requirements

El documento [ABAD17] se realizó para estudiar la aplicación de la técnica Wizard-of-Oz (WOz) en la elicitación de requisitos de aplicaciones móviles; así como comparar la capacidad de este método con el análisis de rereseñas de usuarios. Con este fin se realizaron dos estudios: un estudio de campo en 13 equipos de desarrollo de aplicaciones móviles y un análisis en retrospectiva sobre reseñas de aplicaciones móviles (70 592 reseñas de 40 aplicaciones disponibles en Google Play). De ambos estudios se concluyó que, si bien el análisis de reseñas de aplicaciones y WOz se pueden utilizar para capturar tipos específicos de requisitos; un proceso integrado de obtención de requisitos que incluya ambos métodos reduciría la brecha de comunicación entre usuarios y desarrolladores en las primeras etapas del proceso de desarrollo y mitigaría el riesgo de cambios en los requisitos durante las etapas posteriores.

(2018) Speech-acts based analysis for requirements discovery from online discussions

El artículo [MORA19] presenta una técnica para el análisis de discusiones en línea que involucra a usuarios de aplicaciones de software que pueden publicar sus mensajes en foros de usuarios, listas de

correo, wikis, grupos de noticias y blogs. Ttiene como objetivo ayudar a los desarrolladores a extraer información relevante para los requisitos de software y las tareas de mantenimiento.

Este método se basa en una técnica lingüística denominada análisis basado en actos de discurso, que fue introducida en trabajos anteriores de los mismo autores. La técnica fue revisado y mejorada mediante la ejecución de un conjunto de experimentos en dos conjuntos de datos diferentes: tomados del proyecto Apache OpenOffice, que contiene 161 120 comentarios textuales; y un conjunto de datos que contiene 575 comentarios y mensajes proporcionados por los usuarios de una aplicación de software en el dominio de gestión de la energía doméstica.

Se descubrió que existe potencialmente una asociación entre las partes del discurso (por ejemplo, informativos, receptivos, solicitantes, etc.) y categorías de problemas (por ejemplo, de optimización y otros). Por lo tanto, se pueden implicar algunos supuestos de combinaciones de actos de habla basados en el tipo de tema discutido por las partes interesadas.

Los resultados experimentales proporcionaron evidencias de que ciertos tipos de actos del discurso se utilizan más cuando se informa una solicitud de mejora en lugar de otro tipo de solicitudes. Por ejemplo, los actos del discurso: Adjunción y Línea de Código fueron usados en el 90% de los mensajes que expresan “otro tipo de solicitud”, mientras que en las solicitudes de Mejora se usa el acto Requestivo en el 80% de los casos.

Se usaron los actos del discursoy análisis de sentimiento como parámetros para entrenar tres algoritmos de aprendizaje automático (Random Forest, J48 y SMO) y clasificaron los comentarios en Mejora y Otras categorías, mejorando la precisión y la recuperación obtenidas en estudios previos.

(2019, August). Your opinions let us know: mining social network sites to evolve software product lines

El documento [ALI19] presenta un proceso que respalda la rápida evolución de los productos SPL al obtener requisitos de sitios populares de redes sociales. Para obtener los requisitos de redes sociales, aplicaron clasificación, modelado de temas y análisis de sentimientos. Para la clasificación, seleccionaron clasificadores SVM, RF y RN multinomiales. Seleccionaron tres características como rasgos de clasificación para los clasificadores: Estos incluyen bolsa de palabras, extracción de sentimientos y

eliminación de palabras vacías (stop words). Observaron que todos los clasificadores producen resultados casi similares.

Además, presentaron una estrategia de diseño de arquitectura adaptativa para reflejar los nuevos requisitos en el sistema rápidamente. Es muy útil para responder rápidamente las necesidades de los usuarios y del mercado, y para mantener actualizada la arquitectura de referencia.

(2019, September). Classifying multilingual user feedback using traditional machine learning and deep learning

El trabajo [STAN19] tuvo como objetivo comparar la efectividad de los métodos tradicionales de Machine Learningcontra los métodos de Deep Learning, en el campo de la elicitación automática de requisitos; más específicamente, en la extracción de reportes de problemas, solicitudes e información irrelevante de la retroalimentación de los usuarios. Para ello empleó tres juegos de datos: reseñas de aplicaciones, *tweets* en inglés y *tweets* en italiano; contando con 6000, 10000 y 15000 casos para cada uno de ellos respectivamente. Como resultado relevante pudo determinar que los algoritmos de Machine Learning clásicos presentaban un rendimiento similar a los de Deep Learning en el parámetro F (que es la media armónica entre la precisión y la recuperación según está definida en [MAAL16]); Machine Learning tiene una recuperación ligeramente mejor, en cambio, los algoritmos de Deep Learning presentaban una pequeña mejoría en la precisión.

1.4 Procesamiento de Lenguaje Natural

El análisis de texto puede entenderse como la técnica mediante la cual se obtiene **información útil** de un texto. Existen varias métodos para lograr este propósito; pero la investigación presente se centrará en las técnicas de **Procesamiento de Lenguaje Natural** (PLN), **Lingüística Computacional** (LC) y en el uso de herramientas numéricas para obtener la información. Primeramente, algunas definiciones:

Lenguaje natural: es el lenguaje de representación del conocimiento más antiguo y exitoso. Se utiliza para la comunicación, la negociación y la lógica; y la mayor parte de las tareas cognitivas humanas lo involucran. [SANT06]

Información útil: el epíteto *útil* es difícil de definir por su carácter relativo al contexto. La utilidad o vanidad de la información es inherente al propósito de la misma. Se comprende como *información útil* aquella parte de los datos que es relevante según sus expectativas de uso. Separar y clasificar la información útil descartando todo lo demás es la tarea principal del análisis de texto.

Procesamiento de Lenguaje Natural (PLN): hace referencia al uso de una computadora para procesar el lenguaje natural.

Lingüística Computacional (LC): es el estudio de la lingüística desde una perspectiva computacional. Se manifiesta mediante el uso de computadoras y algoritmos para realizar tareas lingüísticas como etiquetar de un texto la parte del discurso (verbo, sustantivo, etc.) que representa, en lugar de realizar esta tarea manualmente.

Según Elizabeth D. Liddy en [LIDD01] existen siete niveles en el procesamiento del lenguaje natural. De ellos tres son relevantes en la elicitación automática de requisitos: los niveles correspondientes al léxico, la sintaxis y la semántica.

1.4.1 Nivel Léxico / POS-tagging

Según [SRIN18] las palabras constituyen la unidad más básica en que puede ser dividida una sentencia. Cada palabra constituye un átomo que posee ciertas caracerísticas especiales para describir, por ejemplo, su rol en el discurso, tiempo verbal, concordancia de género y número, etc. Las etiquetas que pueden asignarse a una palabra dependen enteramente del grupo al que pertenecen (no tiene sentido hablar del género de un verbo, o del tiempo verbal de un adjetivo, o del grado de un sustantivo). Esta clasificación primaria de las palabras permite dividirlas en **grupos** según su **rol en el discurso** (sustantivo, adjetivo, verbo, adverbio, pronombre, preposición, conjunción, interjección) y es conocida en inglés como *Part-Of-Speach tagging*, que se traduce como **etiquetado por parte del discurso** y que será referido en adelante como **POS-tagging**.

1.4.2 Nivel Sintáctico / Análisis de dependencias

La estructura de una oración puede verse como el resultado de una serie de elecciones sintácticas realizadas al generarla, seleccionadas de un conjunto limitado. De acuerdo a [WINO72], este conjunto limitado lo conforman las reglas gramaticales propias del lenguaje objeto de análisis.

Las palabras no están organizadas de forma arbitraria dentro de la oración, en cambio, están jerárquicamente relacionadas unas con otras de acuerdo a su correspondiente etiqueta como parte del discurso (o POS-tagging). Si bien algunos lenguajes, entre ellos el español, son más flexibles respecto al uso del hipérbaton; las relaciones de dependencia entre los diferentes palabras está siempre presente. La función del analizador sintáctico es generar una forma de representación estructurada (grafo) de las relaciones internas entre las partes del enunciado. **[LIDD01]** En este nivel donde se realiza el POS-tagging de aquellas palabras donde su correspondiente parte del discurso depende del contexto (la palabra “cocina” puede hacer funciones de verbo o sustantivo según su contexto: El cheff cocina en la cocina).

1.4.3 Nivel Semántico

Identificadas las dependencias entre las diferentes palabras de una sentencia y sus correspondientes clasificaciones; el siguiente paso es determinar los posibles “significados” de esta.

En este nivel ocurre la desambiguación semántica de palabras con múltiples sentidos cuyo significado no pudo ser resuelto en los niveles anteriores (si el analizador sintáctico solucionaba palabras con multiples opciones de POS-tagging, el analizador semántico resuelve palabras polisémicas que tienen un solo etiquetado posible. (la palabra “razón” puede ser empleada como sinónimo de fracción, como sinónimo de conciencia, como sinónimo de propósito o como sinónimo de acierto). Se puede implementar una amplia gama de métodos para lograr la desambiguación: algunos requieren información estadística sobre la frecuencia con la que cada sentido ocurre en un corpus particular de interés para el entrenamiento del modelo estadístico, mientras que otros toman en consideración únicamente el contexto local. **[LIDD01]**

1.5 Aprendizaje Automático

El aprendizaje automático es un muy amplio campo de investigación y desarrollo. Se puede definir como el subcampo de la Inteligencia Artificial que otorga a las computadoras la capacidad de aprender a

dar solución a un problema sin ser programadas explícitamente para ello, es decir, sin requerir que el programador indique las reglas que deben seguir para lograr su tarea. [TORR18]

Típicamente, los algoritmos de Aprendizaje Automático, o Machine Learning (ML), utilizan datos históricos sobre un fenómeno para generar un modelo que permita predecir el comportamieto de las nuevas variables. El modelo se obtiene durante la fase de aprendizaje, el cual puede ser no supervisado o supervisado dependiendo de si los datos de entrenamiento están previamente clasificados o de si es necesario encontrar un patrón que los carácterice, respectivamente

1.5.1 Clasificación usando Aprendizaje Supervisado

El problema de clasificación ocurre cuando un conjunto de *n* objetos O_i debe ser analizados para obtener una función $F(O_x)$ que permita identificar la clase C_j dentro de un conjunto de *m* clases a que pertenece un objeto O_x (ajeno o no al set de datos de entrenamiento) basándose en sus atributos.

En el contexto del procesamiento de lenguaje natural para la elicitación de requisitos de software, la clasificacón del texto generado por el usuario según su temática, intencionalidad o simplemente su utilidad para los ingenieros, constituyó el paso final de la mayoría de los estudios realizados hasta la fecha con esta temática. Si bien unos pocos trabajos probaron el empleo de clasificadores basados en reglas, la gran mayoría utilizó algoritmos de aprendizaje automático; siendo los más populres: Naïve Bayes, Naïve Bayes multinomial, Máquina Vectorial de Soporte y Regresión Logística; con algunos trabajos utilizando Decision tree y Random forest. Muchos de los trabajos analizados en [LIM21] emplearon muchas de las variantes para realizar comparaciones de rendimiento entre una y otra. La mayoría de estos algoritmos utilizó como forma de representación de los datos Bolsas de Palabras (Bag of Words), Siendo TF-IDF (Term Frequency–Inverse Document Frequency) el siguiente más popular.

1.5.2 Agrupamiento mediante Aprendizaje no Supervisado

Aunque similar al aprendizaje supervisado en sus condiciones iniciales, en este caso se desconocen las clases en que deben ser clasificados. El objetivo del agrupamiento es determinar n conjuntos disjuntos Cj donde los objetos Oi serán asignados según sus atributos.

En el contexto del procesamiento de lenguaje natural para la elicitación de requisitos de software, el estudio [LIM21] reveló que han sido empleados los algoritmos k-means, y Latent Dirichlet Allocation y otros algoritmos de Modelado de Tópicos como BTM (Biterm Topic Model) para agrupar los requisitos en rangos de prioridad; para agrupar las opiniones de los usuarios según su temática.

1.6 Concluciones Parciales

- La ingeniería de requisitos es una etapa vital de la ingeniería de software; sin embargo los métodos convencionales son ineficacez para capturar la totalidad de los requisitos; especialmente si no involucran las opiniones de los usuarios.
- La automatización de todo o parte del proceso de cara a los usuarios finales agiliza y mejora la calidad del proceso de elicitación
- Ya que generalmente se procesan opiniones de usuarios el proceso suele comenzar con un procesamiento de lenguaje natural y continua con clasificaciones y/o agrupamientos usando herramientas de Machine Learning.

CAPÍTULO 2: MÉTODO PROPUESTO

En este capítulo se presentará el método propuesto para la elicitación de requisitos usando datos dinámicos de acuerdo al siguiente esquema:

En la sección 2.1, como indica su nombre, se formulará el problema como una secuencia de pasos concretos a seguir para darle solución.

Luego, en 2.2: se explica el método de solución desglosado por sus diferentes etapas; describiendo los pormenores de cada una de las mismas.

Finalmente, en la sección 2.3: se exponen las herramientas de software utilizadas en la propuesta de solución, proveyéndose una breve explicación de cada una.

2.1 Formulación del problema

En la sección anterior se esclareció que los datos antropogénicos; especialmente las reseñas emitidas por los usuarios son la fuente de datos dinámicos predilecta para la elicitación automática de requisitos; además de ser la más abundante y potencialmente útil. Partiendo de esta premisa y en el contexto de una aplicación **APP** desarrollada o en fase beta cuya distribución o promoción está a cargo del sitio **SITE**, donde sus usuarios difunden libremente sus opiniones sobre la misma; debe ejecutarse el siguiente ciclo de acción:

Del sitio se extrae el conjunto **REVIEWS** de las opiniones realizadas en un lapso de tiempo predeterminado con límite superior en el presente. Para cada opinión $\mathbf{REVIEW_i}$ del conjunto **REVIEWS** debe efectuarse un proceso de eliminación de ruido (metadatos incluidos por **SITE** que no aportan contexto al contenido de la opinión, cláusulas HTML) y el preprocesamiento de los datos para su clasificación; con lo que se obtiene el nuevo conjunto **REVIEWS***.

Las opiniones del conjunto **REVIEWS*** serán filtradas de acuerdo a su utilidad para el proceso de elicitación de requisitos, formando el subconjunto **USEFUL-REVIEWS** de opiniones útiles preprocesadas. Una opninión será filtrada como no útil si el análisis de tópicos revela que: no está relacionada con la

aplicación en cuestión, se limita a establecer el grado de satisfacción del usuario sin abordar ningún aspecto específico de la aplicación, contiene información contradictoria o insuficiente.

Las opiniones del conjunto **USEFUL-REVIEWS** serán clasificadas según el tipo de requisito que potencialmente será extraído de las mismas en: *bug*, solicitud para añadir carácterística, problemas de interfaz, problemas de compatibilidad, solicitud de mejoramiento de una carácterística / queja sobre el funcionamiento de una carácterística, otras.

2.1.1 Ejemplo hipotético

Reddit (APP) es una importante red social con millones de usuarios. En *Google Store* (SITE) con más de dos millones de reseñas como puede observarse en la captura de pantalla:

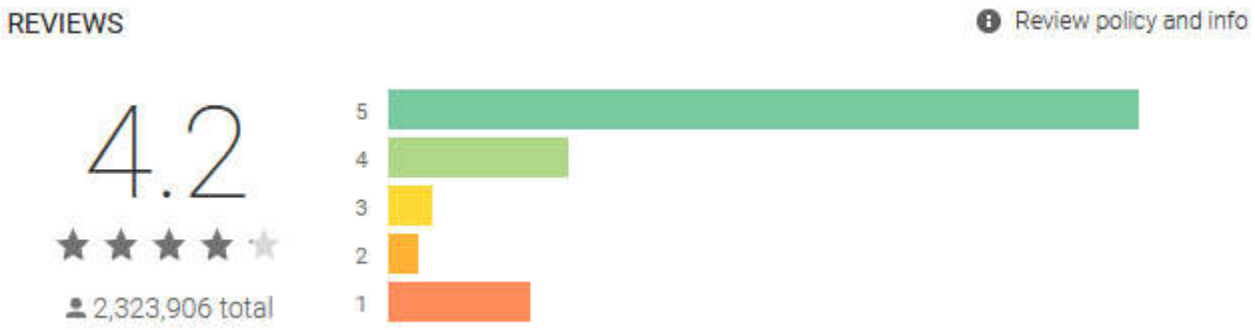

Figura 2.1

Tomando el conjunto de las opiniones emitidas por los usuarios en los últimos días se puede obtener la siguiente emulación del proceso deseado:

Tabla 2.1: REVIEWS (emulación del proceso)

ID	Opinión
R_1	La aplicación se ha vuelto muy lenta recientemente. Ya no puedo desplazarme hacia los comentarios anteriores que pasé; la aplicación solo muestra el ícono de recarga y luego se congela cuando intento desplazarme hacia arriba. A veces, más de dos videos comienzan a reproducirse si están muy juntos en el feed, es muy molesto tener que buscar y pausar todas las publicaciones con videos en reproducción.

R_2	Reddit solía funcionar bien, ahora no puedo cargar nada. Nada, ni la pantalla de inicio, los mensajes, las notificaciones, nada. Estoy seguro de que es una mala actualización o algo así, espero que se solucione pronto. De lo contrario, le daría 5 estrellas porque me encantó la aplicación.
R_3	Actualmente, la aplicación solo se puede usar cuando estoy usando WiFi, había estado funcionando bien hasta hace aproximadamente 2 semanas. Y no, no es mi teléfono, ya que puedo ver Netflix y YouTube mientras uso datos móviles. Intenté desinstalar y reinstalar, pero eso no solucionó mi problema. Arreglaré mi calificación una vez que se solucione, ya que realmente me gusta usar Reddit.
R_4	La última versión que te pone en un modo de video extraño cuando miras un video, es realmente molesta. A veces, *volver* sale de la aplicación, otras veces, vuelve a mi feed (el comportamiento deseado sería permanecer en mi feed todo el tiempo)

Tras filtrar y clasificar las opiniones anteriores obtenemos la siguiente distribución:

Tabla 2.2: Resultados (emulación del proceso)

Opiniones no útiles	Opiniones útiles (USEFUL-REVIEWS)
R_2: Los problemas generales que afectan el funcionamiento de toda la aplicación para un usuario específico solo son analizables si se conoce el contexto particular del usuario. R_3: Al igual que R_2, no brinda contexto ni datos concretos sobre la afectación.	R_1: Reporta un posible *bug* R_4: Sugiere el mejoramiento de una carácterística

2.2 Propuesta de solución

El presente trabajo dejará las opciones abiertas para la selección del método de extracción de opiniones, desde que el mismo no influye en el rendimiento del resto del proceso, y se centrará en las tres fases posteriores para dar solución al problema planteado.

Fase 1: pre-procesamiento de texto: en esta fase ocurren las tareas de depuración y transformación de los datos para eliminar la información innecesria, rellenar los huecos, y, en terminos generales, preparar los datos para la siguiente fase.

Fase 2: filtrado: en esta fase se eliminan las opiniones que no aporten nada al proceso de elicitación de requisitos.

Fase 3: agrupamiento: en esta fase se catalogan las diferentes opiniones en clases no predeterminadas según la temática que aludan.

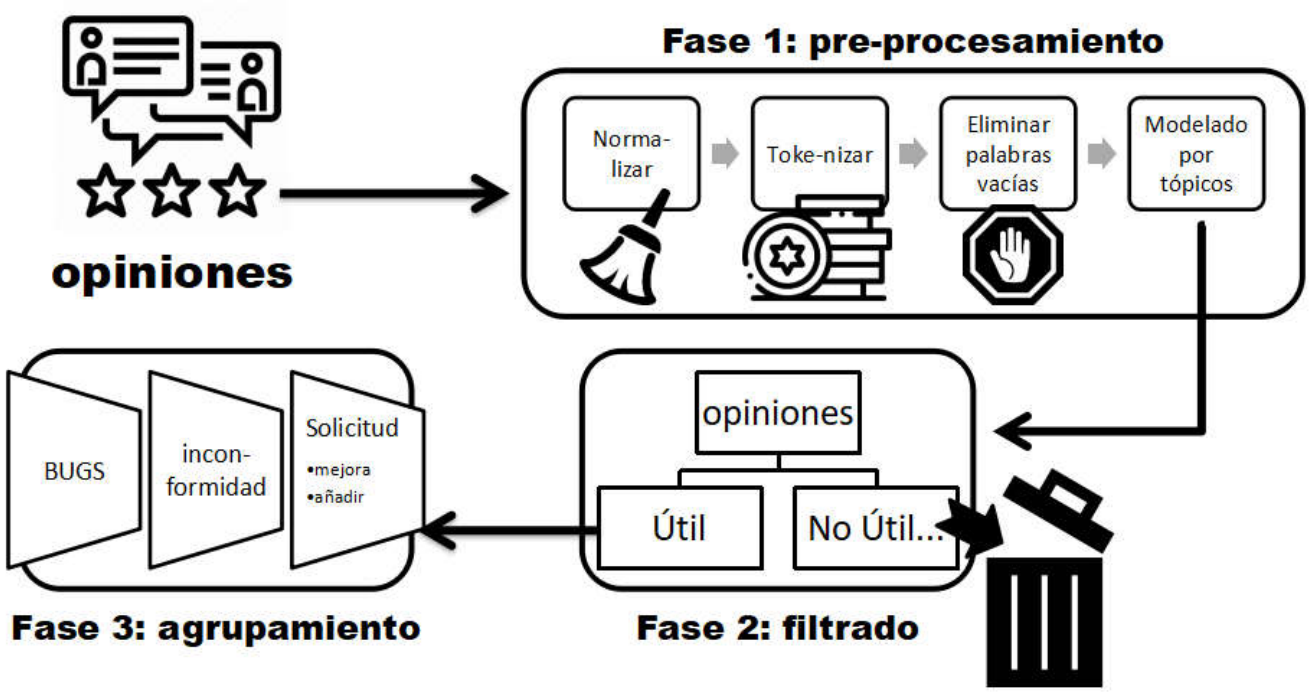

Figura 2.2

2.2.1 Pre-procesamiento

La etapa de preprocesamiento como suele serle el primer eslabón de toda cadena, es de vital importancia para el resto del proceso pues es donde se garantiza que los datos sean manejables para las instancias posteriores del mismo.

Un posible paso inicial podría ser la segmentación de las opiniones en oraciones. Cada segmento será añadido al conjunto como una opinión independiente. Aunque esta acción aumente considerablemente el número de opiniones a procesar; puede estar justificada por los beneficios que aporta al proceso: no es inusual que una misma reseña contenga varios reportes, o bien información que no es útil al proceso mezclada con la información útil; generalmente distribuidos entre varias oraciones. Al dividir las opiniones en oraciones se atomizan las opiniones mixtas reduciendo las posibilidades de pasar por alto algunos requisitos potenciales cuando varios converjen en una misma reseña y se reducen las posibilidades de clasificar erróneamente opiniones útiles como no útiles. En contrapocisión, existe la posibilidad de diluir algún requisito potencial de larga explicación haciéndolo imposible de identificar.

A continuación el flujo de este paso debe seguir la siguiente estructura:

Normalización: Los datos deben ser presentados de forma consistente, para esto algunas convenciones importantes son la conversión de todo el texto a minúsculas, la corrección de errores ortográficos, y la expansión de contracciones (en inglés son comunes las contracciones como *can not* **=>** *can't* y *I am* **=>** *I'm*) y abreviaturas.

Analisis léxico: Este paso, usualmente denominado "*tokenización*", consiste en transformar la oración en un n-vector donde cada uno de sus componentes será una de las palabras o símbolos de la misma (*token*) tras haber sido debidamente etiquetadas y lenmatizadas durante el proceso de POS-tagging tal fue explicado en el capítulo anterior.

Eliminación de ruido: En este paso se eliminan todos los carácteres no alfanuméricos tales como signos de puntuación, carácteres especiales, emojis y etiquetas html.

Supresión de palabras vacías: La listas de palabras vacías (*stop words*) consiste de palabras que no aportan significado al enunciado; por ejemplo: artículos, prepocisiones, adverbios, pronombres, conjunciones, entre otras. Python brinda soporte en varios idomas, incluido el Español, para instalar la lista de palabras vacías a través del Intalador de Paquetes de Python (pip). [https://pypi.org/project/stop-words/]

2.2.2 Transformación

Los algoritmos de Machine Learning no están diseñados para trabajar con palabras como dato de entrada; tras la limpieza de los datos el siguiente paso consiste en transformar los datos a una forma de representación manejable por los algoritmos de Machine Learning. El modelo de representación más popular es el modelo de Frecuencia del Término o "**Bolsa de Palabras**"; cuya escencia es convertir documentos de texto en vectores de modo que cada documento (en este caso, reseña) se convierta en un vector que represente la frecuencia de todas las palabras distintas que están presentes en el espacio vectorial de ese documento (reseña) específico. Usando Bolsa de Palabras, una reseña $\mathbf{REVIEW^*_j}$ es expresada en un N-vector $\mathbf{VECTOR_j} = (\mathbf{V_{1,j}} \ldots \mathbf{V_{i,j}} \ldots \mathbf{V_{N,j}})$, en el cual $\mathbf{V_{i,j}}$ denota el peso de la i-ésima característica calculado por la frecuencia del término "i" en la reseña $\mathbf{REVIEW^*_j}$, y N denota el número de términos en el diccionario.

Como alternativa a Bolsa de Palabras es posible utilizar la Frecuencia del Término - Frecuencia Inversa del Documento, más conocido por sus siglas en inglés **TF-IDF**. TF-IDF es un modelo que combina dos métricas: (1) El valor sin procesar de la frecuencia de un término en un documento (reseña) en particular; y (2) la inversa de la frecuencia del documento para cada término, que se calcula dividiendo el número total de documentos (reseñas) en el corpus de interés (el conjunto **REVIEWS** de todas las reseñas) por la frecuencia del documento para cada término y luego aplicando una escala logarítmica en el resultado. La Frecuencia Inversa del Documento se puede representar matemáticamente mediante la siguiente fórmula:

$$IDF_i = \log (\text{total\_de\_requerimientos} / \text{total\_de\_requerimientos\_con\_el\_término\_i})$$

Combinando las dos métricas, el vector de características TF-IDF se puede definir matemáticamente como:

$$\text{TF-IDF}(\text{term}_{i,j}) = TF_{i,j} \times IDF_i$$

donde TF representa la frecuencia del término e IDF la inversa de la frecuencia del documento, para el término i y la reseña j.

2.2.3 Filtrado

En esta etapa se eliminan aquellas opiniones que no posean potencial para convertirse en requisitos de software, que según estadísticas, representan el 70 % de la totalidad de los datos generados por usuarios. Para clasificar una reseña como información útil o inútil debe responder a la siguiente propuesta de clasificación brindada por el estudio de Chen et al. [CHEN14]:

Tabla 2.3: Clasificación de reseñas según su utilidad

Utilidad	Criterio	Ejemplo
Sí	Error que produce un resultado incorrecto o no esperado	A veces, más de dos videos comienzan a reproducirse si están muy juntos en el feed
	Error que degrada el rendimiento de la aplicación	Ya no puedo desplazarme hacia los comentarios anteriores que pasé; la aplicación solo muestra el ícono de recarga y luego se congela cuando intento desplazarme hacia arriba.
	Pedido de adición o mejora de una funcionalidad	A veces, *volver* sale de la aplicación, otras veces, vuelve a mi *feed* (el comportamiento deseado sería permanecer en mi *feed* todo el tiempo)
	Pedido para eliminar anuncios, publicidad y/o notificaciones	Demasiados anuncios, es imposible navegar.
	Pedidos para eliminar restricciones y/o permisos	Esta aplicación pide demasiados permisos que no debería.
No	Expresiones emocionales	Le daría 5 estrellas porque me encantó la aplicación.
	Descripción de aplicaciones, funcionalidades, acciones, etc.	Actualmente, la aplicación solo se puede usar cuando estoy usando WiFi, había estado funcionando bien hasta hace aproximadamente 2 semanas.

	Expresiones generalizadas o no claras de fallas o pedidos	Reddit solía funcionar bien, ahora no puedo cargar nada.
	Preguntas y pedido de información	¿Cómo consigo monedas?

Para dar solución a este problema es necesario entrenar un clasificador binario, de entre los algoritmos de Machine Learning. La selección de un algoritmo para la clasificación de opiniones según su utilidad requiere de un proceso previo de comparación; donde de varios algoritmos candidatos que son testeados según las variables precisión y recuperación (que se refiere al tiempo de ejecución del algoritmo).

De los estudios realizados se pueden destacar como candidatos los algritmos de **Máquina Vectorial de Soporte** o SVM por sus siglas en inglés; y el método de **Regresión Logística Binaria**. Además de otros algoritmos que se mencionarán a continuación.

Las **Máquinas Vectoriales de Soporte** son una clase particularmente poderosa y flexible de algoritmos supervisados para clasificación y regresión. SVM es un modelo de aprendizaje automático potente y versátil, capaz de realizar clasificación lineal o no lineal, regresión e incluso detección de valores atípicos. El algoritmo realiza la clasificación creando un hiperplano lineal de margen máximo que separa **dos clases**. Este margen hace que haya pocas posibilidades de separar los datos de la muestra y, por lo tanto, hay pocas posibilidades de clasificar erróneamente nuevas instancias.

La **Regresión Logística** se usa comúnmente para estimar la probabilidad de que una instancia pertenezca a una clase específica. Es un tipo de regresión en el que se utiliza una variable independiente para pronosticar la variable dependiente. Se llama regresión logística binaria cuando la variable dependiente tiene dos clasificaciones.

Naive Bayes Multinomial es una variante de Naive Bayes que se utiliza principalmente en el procesamiento de texto. Nive Bayes es un algoritmo de aprendizaje automático para la clasificación, basado en el teorema de Bayes que da la probabilidad de ocurrencia de un evento dado. Naive Bayes

es un clasificador probabilístico, lo que significa que, dada una entrada, predice la probabilidad de que la entrada se clasifique para todas las clases.

Árbol de Decisiones Binarias: es una técnica de aprendizaje supervisado que tiene una variable objetivo predefinida y se usa con mayor frecuencia en problemas de clasificación. El proceso de entrenamiento se asemeja a un diagrama de flujo, con cada nodo interno (no hoja) representando una prueba sobre un determinado atributo, cada rama es el resultado de esa prueba y cada nodo hoja (terminal) contiene una etiqueta de clase.

2.2.4 Agrupamiento

El proceso de agrupamiento consiste en dividir el conjunto de opiniones útiles en grupos de forma tal que cada uno de los elementos dentro de un grupo **GROUP** compartan más similitudes internamente entre ellos que con los elementos externos al grupo.Villarroel et al. en [VILL16] propone tres clases disjuntas:

Sugerencias de funcionalidades: propone, pide o sugiere la mejora o incorporación de una nueva funcionalidad al sistema.

Informes de errores: está informando un error en la aplicación.

Otro: su contenido no pertenece a ninguna de las categorías anteriores. (Generalmente está asociado a requisitos no funcionales).

Distintivamente, el proceso de agrupamiento no divide el conjunto en un conjunto de clases predeterminado; sino que las clases serán determinadas durante la ejecución del algoritmo. Las clases serán luego nombradas de acuerdo a sus características comunes. Algunos de los algoritmos que más se utilizan para esta tarea son: **K-Means clustering** y **Density-Based Spatial Clustering of Applications with Noise**.

K-Means clustering: es un algoritmo simple de aprendizaje automático sin supervisión que se utiliza para resolver problemas de agrupación. Sigue un procedimiento sencillo de clasificar un conjunto de datos dado en un número k predefinido de grupos. Luego, los grupos se colocan como puntos y todas las

observaciones o puntos de datos se asocian con el grupo más cercano, se calculan, se ajustan y luego el proceso comienza de nuevo utilizando los nuevos ajustes hasta que se alcanza el resultado deseado.

Density-Based Spatial Clustering of Applications with Noise: más conocido por sus siglas DBSCAN, es un algoritmo de agrupación basado en la densidad, que se puede utilizar para identificar agrupaciones de cualquier forma en un conjunto de datos que contiene ruido y valores atípicos. Los clústeres son regiones densas en el espacio de datos, separadas por regiones de menor densidad de puntos. El algoritmo DBSCAN se basa en esta noción intuitiva de "clústeres" y "ruido" para agrupar los datos de forma eficiente.

2.3 Herramientas de desarrollo

2.3.1 Lenguaje de programación Python 3.7

Python [PYTH17] es un lenguaje de programación interpretado cuya principal filosofía es que sea legible por cualquier persona con conocimientos básicos de programación. Entre las ventajas de utilizarlo pueden contarse:

- Es totalmente gratuito.
- Está respaldado por una enorme comunidad por lo que continuamente se están desarrollando nuevas librerías, aplicaciones y una documentación muy actualizada. (Este punto es el que más influyó en la decisión de utilizarlo como lenguaje de programación principal; pues algunas de las librerías más importantes y mundialmente utilizadas de Machine Learning y Procesamiento de Lenguaje Natural están programadas en Python)
- Es un lenguaje multiparadigma, o sea, combina propiedades de diferentes paradigmas de programación, lo que permite ser muy flexible y fácil de aprender independientemente de los conocimientos del interesado.
- Sus aplicaciones no se limitan a un área en concreto y abarca campos aparentemente tan dispares como el diseño de aplicaciones web o la inteligencia artificial, entre muchos otros.
- Python es apto para todas las plataformas y es portable.

2.3.2 spaCy

spaCy [ALTI21] es una librería avanzada, gratuita, de código libre, para el Procesamiento de Lenguaje Natural (NLP) en Python. Está diseñado específicamente para su uso en producción con el propósito de brindar soporte para crear aplicaciones que procesan y “comprenden” grandes volúmenes de texto. Se puede utilizar en la creación de sistemas de extracción de información o de comprensión del lenguaje natural; o bien para el preprocesamiento de texto para el aprendizaje profundo.

Una de las ventajas de spaCy es que está diseñado para “hacer el trabajo” sin dar rodeos. Envés de saturar al usuario con muchas herramientas para cada uno de los campos del procesamiento de texto limita sus opciones a solo una para cada una de las tareas, seleccionando por el usuarionla herramienta más óptima. Sacrificando en variedad ganan mucho en eficiencia y simpleza de uso.

2.3.3 Natural Language Proccesing Toolkit (NLTK)

Natural Language Proccesing Toolkit [PERE14] es una biblioteca de Procesamiento de Lenguaje Natural que utiliza el lenguaje de programación Python. NLTK es software libre, de código abierto. Como herramienta tiene gran soporte de su inmensa comunidad de usuarios; y es una de las herramientas de Procesamiento de Lenguaje Natural de mayor aceptación en el ámbito científico.

Entre las ventajes que brinda NLTk podemos mencionar que 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.

2.3.4 Numpy

NumPy [OLIP06] es una librería de Python de código libre que introduce los vectores y las matrices en Python; y posee numerosas funciones para trabajar con dichos vectores y matrices. Y esto es fundamental porque todos los algoritmos de Machine Learning utilizan vectores y matrices como datos de entrada.

Entre las ventajas de Numpy se pueden mencionar que:

- Está escrito en C, lo que le proporciona de una velocidad muy alta cuando se trabaja con grandes conjuntos de datos.
- Incluye funciones para operaciones matemáticas, de lógica, de ordenación, estadísticas, de entrada y salida, para la lectura y escritura sobre ficheros, entre otras.

2.3.5 Pandas

Pandas [MCKI11] es una librería de software libre escrita para el lenguaje de programación Python con el propósito de asistir tareas de manipulación y el análisis de datos. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series de tiempo. Pandas se utiliza principalmente para el aprendizaje automático en forma de DataFrames. Permite importar datos de varios formatos de archivo como csv, excel, etc. Y proporciona varias operaciones de manipulación de datos como agrupar, unir, fusionar, fundir, concatenar, así como funciones de limpieza de datos como rellenar, reemplazar o imputar valores nulos.

2.3.6 Scikit-Learn

La librería scikit-learn [KRAM16], también llamada sklearn, es un conjunto de rutinas escritas en Python para hacer análisis predictivo, que incluyen clasificadores, algoritmos de agrupamiento, etc. Está basada en NumPy, SciPy y matplotlib. Además, incorpora varias funciones para preprocesar los datos. Como las anteriores, es una librería gratuita y de código abierto.

2.4 Conclusiones Parciales

- El proceso descrito cuenta con tres etapas fundamentales: preprocesamiento, filtrado y clasificación.
- Obtiene como datos de entrada un cojunto opiniones de usuarios no clasificadas o estructuradas y devuelve un conunto menor de opiniones potencialmente explotables para la elicitación de requisitos, divididas en 4 grupos: sugerencias de funcionalidad, errores, problemas de compativilidad o eficiencia y otros.
- Se usará como lenguage de programación Python 3.7 y se aprovecharán sus múltiples librerías para el procesamiento de lenguaje natural, procesamiento de datos y machine learning.

CAPÍTULO 3: VALIDACIÓN Y ANÁLISIS DE RESULTADOS

El proceso consta de tres etapas diferentes, en cada una de las cuales se deberá utilizar un algoritmo diferente para su completamiento. Los algoritmos que serán evaluados según las tres faces son:

Transformación: Bolsa de Palabras, TF-IDF

Filtrado: Máquina Vectorial de Soporte, Regresión Logística Binaria, Bernoulli Naive Balles, Árbol de Decisiones.

Agrupamiento: K-Means clustering, Density-Based Spatial Clustering of Applications with Noise

A lo largo de este capítulo se aplicarán pruebas de eficiencia a los diferentes algoritmos para descubrir cuál presenta un mejor rendimiento en el presente caso de estudio.

3.1 Descripción de los datos de prueba

El conjunto de datos empleado para las pruebas es el mismo empleado por Milián en [MILI21]. Cuenta con 12 000 muestras etiquetadas como información útil y no útil. Las muestras consisten en reseñas de usuarios sobre diferentes aplicaciones de software (Facebook, Tap Fish, Temple Run 2, Swift Key). Cada uno de los sitios comentados cuenta con 3 000 reseñas.

La proporción entre opiniones útiles y no útiles queda recogida en la siguiente tabla:

Tabla 3.1: Distribución de los datos de prueba

	Facebook	**Tap Fish**	**Temple Run 2**	**Swift Key**	**Total**
Útil	1 662 (55.4 %)	739 (24.6 %)	933 (31.1 %)	879 (29.3 %)	4 213 (35.1 %)
No Útil	1 338 (44.6 %)	2 261 (75.4 %)	2 067 (68.9 %)	2 121 (60.7 %)	7 787 (64.9 %)
Total	3 000	3 000	3 000	3 000	12 000

Se puede apreciar como los datos reflejan los resultados de estudios anteriores: las opiniones útiles para el proceso de elicitación constituyen solamente cerca del 30 % (35 % en este caso) de las opiniones emitidas.

3.2 Evaluación de los algoritmos de aprendizaje automático

La algoritmos utilizados serán evaluando usando dos metodologías distintas. Durante el proceso de filtrado, donde se utlizarán **algoritmos de aprendizaje supervisado**, se empleará el método de validación cruzada de k-pliegues; mientras que en la etapa de clasificación, donde se emplearán **algoritmos de aprendizaje no supervisado** para el agrupamiento, se emplearán índices de validación internos que serán definidos en la sección correspondiente.

3.2.1 Validación Cruzada

En la fase de filtrado se entrenaron varios algoritmos de aprendizaje automático para completar la tarea de clasificación; y se empleó el método de validación cruzada de k-pliegues para comparar los algoritmos.

La validación cruzada es una técnica estadística para evaluar modelos de aprendizaje automático mediante el entrenamiento de varios modelos de aprendizaje automático usando subconjuntos con la mayoría de los datos de entrada disponibles y evaluándolos en el subconjunto complementario.

Como nunca hay suficientes datos para entrenar los modelos, eliminar una parte para su validación afectará en gran medida la capacidad de los mismos e incluso planteará problemas como un ajuste inadecuado. Al reducir los datos de entrenamiento se corre el riesgo de perder patrones / tendencias importantes en el conjunto de datos, lo que a su vez aumenta el error inducido por el sesgo. En estas circunstancias la validación cruzada de k-pliegues es un método exactamente adecuado que proporciona una gran cantidad de datos para entrenar el modelo y también deja una gran cantidad de datos para la validación.

En este método, todos los datos se dividen en k subconjuntos. Para cada iteración, se toma un subconjunto como conjunto de validación y los subconjuntos restantes como conjuntos de entrenamiento. El número de iteraciones será igual al número de subconjuntos.

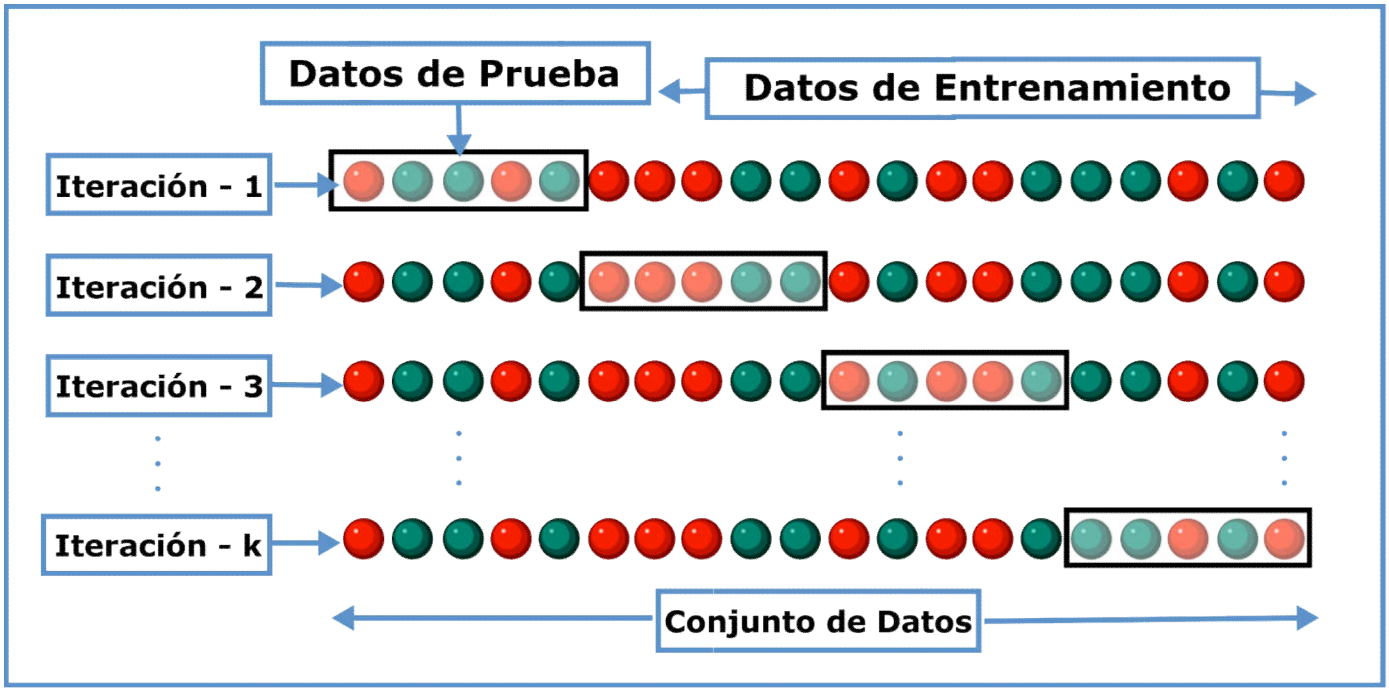

Figura 3.1

La librería Scikit-Learn de Python brinda soporte para realizar la validación cruzada de varios métodos de forma rápida y eficiente, lo que permitió agilizar este proceso.

3.2.2 Medidas de desempeño (fase de filtrado)

Para comparar los algoritmos de acuerdo a su desempeño se emplearán algunas de los criterios de comparación más utilizadas: precisión, regresión y la medida-F [DIAS20]. Para el calcular los mismos serán definidas las siguientes variables:

Tabla 3.2: Matriz de confusión (etapa de filtrado)

		Valor Predicho	
		Clase Ci	**Otra Clase**
Valor Real	**Clase Ci**	VP	FN
	Otra Clase	FP	VN

VP, FP: Total de verdaderos y falsos positivos, respectivamente

VN, FN: Total de verdaderos y falsos negativos, respectivamente

Precisión: se define como la proporción de **predicciones positivas correctas** (VP) de todos los casos clasificados como positivos. Esto significa que una precisión baja indicará un alto número de **predicciones positivas erradas** (FP).

$$\text{Precisión} = VP / (VP + FP)$$

Regresión: está definido como la razón de predicciones positivas correctas (VP) y el total de elementos de esa clase en el conjunto de datos; que se puede obtener añadiéndole el conteo de **predicciones negativas erradas** (FN)

$$\text{Regresión} = VP / (VP + FN)$$

Medida-F: se define como la media harmónica de la precisión (P) y la regresión (R). Constituye un exelente indicador que combina los dos criterios más utilizados y puede ser empleado, sino como criterio único, como medida de desempate.

$$\text{Medida-F} = 2 * P * R / (P + R)$$

Como medida extra se utilizó el coeficiente de Cohen's kappa [WAN15], que es una medida que da muy buenos resultados cuando se manipulan datos particionados en clases desbalanceadas como las opiniones de usuario (no útil - 65 % / 35 % - útil).

Cohen's kappa: se define como la razón del **valor relativo de aciertos de un clasificador a ser evaluado** (p_0) contra el **valor relativo de aciertos esperado en un clasificador aleatorio** (p_e). Convencionalmente se asume que un **coeficiente de kappa** (CK) inferior a 0.4 es malo; y es bueno con un valor superior a 0.6; siendo excepcionalmente bueno si supera 0.8.

$$CK = (p_0 - p_e) / (1 - p_e)$$

$$p_0 = (TP + TN) / (TP + TN + FP + FN)$$

$$p_e = ((TP + FN) * (TP + FP) + (FP + TN) * (FN + TN)) / (TP + TN + FP + FN)^2$$

3.2.3 Índices de validación internos (fase de clasificación)

Para evaluar y comparar la calidad de algoritmos no supervisados las métricas convencionales como la precisión se vuelven inaplicables desde que no existe un conjunto de datos previamente clasificado sobre el que trabajar; sino que serán determinadas las posibles clasificaciones, o grupos, que pueden extraerse del mismo. A las medidas de evaluación que no emplean un conjunto de datos objetivo (datos preclasificados) se les suele denominar índices de validación internos [REND11]. En la evaluación de los algoritmos de agrupamiento se emplearán algunos de ellos.

Coeficiente de silueta: Para un grupo dado, X_j (j = 1,... c), la técnica de silueta asigna a la i-ésima muestra de X_j una medida de calidad, S(i) = (i = 1,... m), conocida como ancho de silueta. Este valor es un indicador de confianza sobre la membresía de la i-ésima muestra en el grupo X_j, y se define como:

$$S(i) = (b(i) - a(i)) / Max\{ b(i), a(i) \}$$

Donde a(i) es la distancia promedio entre la i-ésima muestra y todas las muestras incluidas en X_j; b(i) es la distancia promedio mínima entre la i-ésima muestra y todas las muestras agrupadas en X_k (k = 1,... c; $k \neq j$).

Coeficiente de Calinski-Harabasz: se puede utilizar para evaluar un modelo cuando no se conocen las etiquetas de verdad del dominio y, por tanto, la validación se ha realiza la agrupación se realiza utilizando cantidades y características inherentes a la conjunto de datos. También conocido como criterio de relación de varianza es una medida de qué tan similar es un objeto a su propio grupo (cohesión) en comparación con otros grupos (separación). La cohesión se estima en función de las distancias desde los puntos de datos en un grupo a su centroide de grupo y la separación se basa en la distancia de los centroides del grupo desde el centroide global.

Índice de Dunn: es una métrica para evaluar algoritmos de agrupación en clústeres, y por ende, un esquema de evaluación interna, donde el resultado se basa en los datos agrupados en sí. Su objetivo es identificar conjuntos de clústeres que sean compactos, con pequeñas variaciones entre los miembros del clúster, y bien separados, donde las medias de los diferentes clústeres estén lo suficientemente alejadas en comparación con sus varianzas internas.

3.3 Resultados de la evaluación

Usando las facilidades que brinda la librería Scikit-Learn para la evaluación de modelos fueron comparados los algoritmos utilizados en las fases de filtrado y agrupamiento. El proceso se repitió para cada una de las formas de representación de los datos utilizadas en la fase de transformación. La información recogida será presentada a continuación.

3.3.1 Resultados de la Validación Cruzada

La Validación Cruzada se empleó para evaluar los algoritmos de aprendizaje automático supervisado de la fase de filtrado. En la siguiente tabla se muestran los resultados obtenidos:

Tabla 3.3: Resultados de la Validación Cruzada

	Máquina Vectorial de Soporte		**Regresión Logística**		**Naive Bayes Multinomial**		**Árbol de Decisión**	
	TF	**TF-IDF**	**TF**	**TF-IDF**	**TF**	**TF-IDF**	**TF**	**TF-IDF**
X_m(precisión)	0.7675	0.8114	0.8093	0.7963	0.6887	0.7561	0.6901	0.6753
S^2(precisión)	0.1122	0.0950	0.0862	0.0914	0.1296	0.1343	0.1196	0.1340
X_m(regresión)	0.5934	0.4184	0.4698	0.4162	0.6473	0.5332	0.4979	0.5444
S^2(regresión)	0.0577	0.1044	0.1163	0.0972	0.1635	0.1157	0.0310	0.0670
X_m(medida-F)	0.6675	0.5483	0.5893	0.5425	0.6504	0.6149	0.5775	0.6094
S^2(medida-F)	0.0756	0.1066	0.1081	0.0948	0.1152	0.1058	0.0535	0.0963
X_m(Cohen's kappa)	0.5193	0.4055	0.4455	0.3934	0.4702	0.4605	0.4090	0.4214
S^2(Cohen's kappa)	0.0803	0.0842	0.0871	0.0659	0.1028	0.0853	0.0688	0.1242

TF (forma de representación de los datos): Frecuencia del Término o Bolsa de Palabras.

TF-IDF (forma de representación de los datos): Term Frequency - Inverse Document Frequency.

$\mathbf{X_m}$: media estadística de la medida evaluada

$\mathbf{S^2}$: desviación típica cuadrada de la medida evaluada

3.3.2 Resultados de la evaluación mediante índices de validación internos

En la fase de clasificación se emplearon dos índices de validación internos para evaluar dos algoritmos no supervisados de agrupamiento. En la siguiente tabla se muestran los resultados obtenidos:

Tabla 3.4: Resultados de la evaluación mediante índices de validación internos

	K-means		**DBSCAN**	
	TF	**TF-IDF**	**TF**	**TF-IDF**
Coeficiente de Silueta	0.53	0.65	0.03	0.16
Coeficiente de Calinski-Harabasz	4469.02	6659.28	198.16	1230.10
Índice de Dunn	0.75	0.54	0.79	0.90
Número de clústeres	6	6	15	9
Número de opiniones no agrupadas	N/A	N/A	2247	1613

TF (forma de representación de los datos): Frecuencia del Término o Bolsa de Palabras.

TF-IDF (forma de representación de los datos): Term Frequency - Inverse Document Frequency.

3.4 Discusión de los resultados

En términos de desempeño los valores obtenidos durante la fase de filtrado son relativamente bajos comparados con estudios previos aunque internamente similares; lo primero puede deberse al uso deficiente de las técnicas de preprocesamiento de datos. Mientras que en la fase de agrupamiento los índices de evaluación pintan resultados muy dispares.

3.4.1 Resultados de la evaluación de los algoritmos para el filtrado y recomendaciones

Dados los valores reunidos en la Tabla 3.3 se realizó la prueba de hipótesis para dos muestras entre todos los algoritmos y formas de representación de los datos analizadas. Para cada par analizado de algoritmos A1 y A2 se determinó la región crítica para determinar si existía una diferencia significativa entre las medias de los valores que obtuvieron en la medida-F y el coeficiente de Cohen's Kappa.

Cada una de las comparaciones realizadas quedó dentro de la región crítica con un 95 % de confianza; con lo que se puede concluir que en el ámbito de la clasificacción de opiniones de usuarios de acuerdo a la utilidad potencial de la información que brindan para el proceso de elicitación de requisitos de software; no existen diferencias significativas entre los algoritmos y formas de representación de los datos más populares.

Otra conclusión curiosa que se extrapola de los datos es que la mayoría de los algoritmos tienen valores relativamente buenos en la precisión mientras que todos presentan valores bajos en el resto de los indicadores. Esto sugiere que la precisión por sí sola no es una medida suficientemente confiable para determinar la calidad de un clasificador; en cambio, la medida-F resultó ser una medida mucho más representativa de la utilidad del modelo para el problema de clasificación dado.

Con el fin de mejorar el desempeño de los modelos se recomienda mejorar la fase de preprocesamiento de los datos al incluir en la lista de palabras vacías términos propios del dominio, realizar la corrección ortográfica de las opiniones de usuario y la substitución de sinónimos. Esto reduciría considerablemente la dimensionalidad de las instancias (opiniones), al transformar los datos al modelo de bolsa de palabras o al de TF-IDF. Adicionalmente se puede aplicar algoritmos como Latent Dirichlet Allocation para reducir aún más las dimensiones y mejorar el desempeño de las preicciones.

3.4.2 Resultados de la evaluación de los algoritmos para el agrupamiento y recomendaciones

Un primer vistazo a los valores recogidos en la tabla 3.4 permite comprobar la disparidad en los valores obtenidos entre los algoritmos testeados de acuerdo a todos los índices. Pero especialmente llama la atención la marcada diferencia interna para las dos formas de representación de los datos en el algoritmos BBSCAN. Esto se debe a que la correcta calibración de los hiperparámetros de este algoritmo para el agrupamiento de datos con muchas dimensiones es una tarea muy complicada que fácilmente

recae en el tanteo. En cambio, los resultados del algoritmo k-means presentaron una mayor consistencia en los valores obtenidos debido a la sencillez de su calibración.

El índice de Dunn para el algoritmo DBSCAN alcanzó valores muy buenos e incluso superiores a los de k-means; especialmente sobre el modelo de representación de datos TF-IDF, sin embargo este valor es engañoso dado el alto número de opiniones clasificadas como ruido.

Para mejorar los valores obtenidos sería conveniente mejorar el pre-procesamiento de los datos y aplicar técnicas que permitan una gran reducción de la dimensionalidad como Latent Dirichlet Allocation. Esto, junto con una concienzuda calibración de los hiperparámetros de los modelos de aprendizaje automático contribuirán a mejorar el desempeño de los algoritmos.

3.5 Problemas de alcance

Los resultados obtenidos por la presente investigación abarcan un conjunto limitado de algoritmos y formas de representación de los datos. No se recomienda extender las conclusiones alcanzadas tras la evaluación a otras formas de representación de datos o algoritmos alternativos.

No se realizaron ajustes en los parámetros de los modelos empleados en la fase de filtrado; acto que puede modificar los resultados obtenidos; sin embargo requiere una profunda comprensión de los algoritmos empleados. En [PERE20], Pérez et al. sugiere emplear algoritmos genéticos para calibrar los hiperparámetros de los métodos.

Otra posible limitación consiste en el empleo del procesamiento de lenguaje natural solo hasta el nivel léxico. Si bien esto agiliza el procesamiento de los datos; emn los niveles superiores se incrementa considerablemente la cantidad y calidad de información obtenida; lo que puede mejorar considerablemente el desempeño de los clasificadores.

Finalmente, los datos de prueba se limitan a reseñas de usuarios de algunas aplicaciones más populres en la *App-Store*, por lo que los resultados no pueden extenderse a otras fuentes de datos dinámicos o, incluso, podrían ser insuficiente para cubrir el proceso de elicitación sobre reseñas de usuarios en aplicaciones de menor popularidad.

3.6 Conclusiones parciales

- Los algoritmos de aprendizaje automático supervisado y los no supervisados requieren diferentes métodos de evaluación.
- No existen diferencias notables entre la aplicación de los principales algoritmos de clasificación para el filtrado de opiniones en dos clases disjuntas.
- K-Means es un algoritmo efectivo a la hora de agrupar las opiniones en diferentes clases de potenciales requisitos.
- La etapa de preprocesamiento de los datos puede ser vital para el desempeño de los algoritmos de aprendizaje automático.
- El redimensionamiento de los datos puede ser una etapa muy importante en el proceso.

CONCLUSIONES GENERALES

La elicitación automática de requisitos de software es aún un área de investigación en constante expansión y práctica. El empleo de datos dinámicos antropogénicos, generalmente inutilizables en otras subesferas de la elicitación de requisitos por su enorme volúmen y constante actualización; es el camino a seguir para integrar las opiniones de los usuarios en el proceso de ingeniería de software de forma óptima y continua.

Existen varios trabajos que han explorado este acercamiento con buenos resultados lo que conllevó a la presente investigación a replicar los mismos utilizando una selección propia de algoritmos a evaluar; no obteniendo tan buenos resultados.

De las limitaciones del empleo del aprendizaje automático para el proceso de clasificación y agrupamiento de opiniones de usuarios que se exploró en este documento se desprende la necesidad de mejorar la manipulación previa de los datos convirtiendo el preprocesamiento de los mismos en el objetivo principal de futuras investigaciones.

Se recomienda para investigaciones posteriores explorar técnicas de Procesamiento de Lenguaje Natural que analicen el texto de las opiniones hasta el nivel semántico y el empleo de algoritmos como Latent Semantic Allocation para la reducción de la dimensionalidad de los datos.

Se concluye, dados los resultados de la investigación, que el desarrollo de un método que aproveche las técnicas de aprendizaje automático para procesar datos dinámicos antropogéncos y automatizar parcial o totalmente el proceso de elicitación de requisitos de software contribuye al mejoramiento de dicho proceso en la producción de software; puesto que presentan un desempeño comparable al de un analista humano y requieren un tiempo mucho menor para procesar grandes volúmenes de información. No obstante los resultados fueron inferiores a los esperados; para lo cual ya se han realizado varias propuestas en pos de su mejoramiento.

BIBLIOGRAFÍA

[ABAD17] Abad, Z. S. H., Sims, S. D., Cheema, A., Nasir, M. B., & Harisinghani, P. **(2017, September). Learn more, pay less! lessons learned from applying the wizard-of-oz technique for exploring mobile app requirements**. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW) (pp. 132-138). IEEE.
https://ieeexplore.ieee.org/document/8054841

[ALI19] Ali, N., Hwang, S., & Hong, J. E. **(2019). Your opinions let us know: mining social network sites to evolve software product lines**. KSII Transactions on Internet and Information Systems (TIIS), 13(8), 4191-4211.
http://itiis.org/digital-library/manuscript/2469

[ALTI21] Altinok, D. **(2021). Mastering spaCy: An end-to-end practical guide to implementing NLP applications using the Python ecosystem**. Packt Publishing Ltd.
https://spacy.io/

[CARV21] de Carvalho, E. A., Gomes, J. O., Jatobá, A., da Silva, M. F., & de Carvalho, P. V. R. **(2021). Employing resilience engineering in eliciting software requirements for complex systems: experiments with the functional resonance analysis method (FRAM)**. Cognition, Technology & Work, 23(1), 65-83.

[CHEN14] Chen, N., Lin, J., HOI, S. C., Xiao, X., & Zhang, B. **(2014). AR-Miner: Mining informative reviews for developers from mobile app marketplace**.(2014). ICSE.

[CHRI92] Christel, M. G., & Kang, K. C. **(1992). Issues in requirements elicitation**. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

[DIAS20] Dias Canedo, E., & Cordeiro Mendes, B. **(2020). Software Requirements Classification Using Machine Learning Algorithms**. Entropy, 22(9), 1057.

[FIRM16] Firmani, D., Mecella, M., Scannapieco, M., & Batini, C. **(2016). On the meaningfulness of "big data quality"(invited paper)**. Data Sci Eng. 2016; 1 (1): 6–20.

[GEA11] de Gea, J. M. C., Nicolás, J., Alemán, J. L. F., Toval, A., Ebert, C., & Vizcaíno, A. **(2011). Requirements engineering tools**. IEEE software, 28(4), 86-91.

[GOOD90] Goodrich, V., & Olfman, L. **(1990, January). An experimental evaluation of task and methodology variables for requirements definition phase success**. In Twenty-Third Annual Hawaii International Conference on System Sciences (Vol. 4, pp. 201-209). IEEE Computer Society.

[IEEE83] Institute of Electrical and Electronics Engineers. **(1984). IEEE guide to software requirements specifications: approved September 22, 1983, IEEE Standards Board; approved July 20, 1984, American National Standards Institute**. Inst. of Electrical and Electronics Engineers.

[IEEE90] IEEE Standards Coordinating Committee. **(1990). IEEE standard glossary of software engineering terminology (IEEE Std 610.12-1990)**. Los Alamitos. CA: IEEE Computer Society, 169, 132.

[KRAM16] Kramer, O. **(2016). Scikit-learn**. In Machine learning for evolution strategies (pp. 45-53). Springer, Cham.
https://scikit-learn.org/stable/index.html

[LIDD01] Liddy, E. D. **(2001). Natural language processing**.

[LIM11] Lim, S. L., & Finkelstein, A. **(2011). StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation**. IEEE transactions on software engineering, 38(3), 707-735.

[LIM21] Lim, S., Henriksson, A., & Zdravkovic, J. **(2021). Data-Driven Requirements Elicitation: A Systematic Literature Review**. SN Computer Science, 2(1), 1-35.

[MAAL16] Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. **(2016). On the automatic classification of app reviews**. Requirements Engineering, 21(3), 311-331.

[MADD00] Madden, A. D. **(2000, November). A definition of information**. In Aslib Proceedings. MCB UP Ltd.

[MCKI11] McKinney, W. **(2011). pandas: a foundational Python library for data analysis and statistics**. Python for high performance and scientific computing, 14(9), 1-9.
https://pandas.pydata.org/

[METH13] Meth, H., Brhel, M., & Maedche, A. **(2013). The state of the art in automated requirements elicitation**. Information and Software Technology, 55(10), 1695-1709.

[MILI21] Milián-Núñez, Vladimir. **(2021). Método para el análisis de opiniones de usuarios como apoyo al proceso de extracción de requisitos de software**. Universidad de las Ciencias Informáticas, La Habana, Cuba.

[MORA19] Morales-Ramirez, I., Kifetew, F. M., & Perini, A. **(2019). Speech-acts based analysis for requirements discovery from online discussions**. Information Systems, 86, 94-112. https://www.sciencedirect.com/science/article/abs/pii/S0306437917306087?via%3Dihub

[PERE14] Perera, K. J. A., Kuruppu, K. A. I., Gamage, M. P., Jayakody, J. A. P. B., Gunasekara, K. S. G. S., & Kodagoda, G. N. **(2014). A step towards a Natural Language Programming Tool (NLPT)**. https://www.nltk.org/

[PERE20] Pérez-Verdejo, J. M., Mezura-Montes, E., Sánchez-García, Á. J., & Ocharán-Hernández, J. O. **(2020). Calibración de hiperparámetros para la clasificación de requisitos de software mediante evolución diferencial**. Res. Comput. Sci., 149(8), 241-251.

[PYTH17] Python, M. Ö. O. **(2017). Python 3.7**. https://www.python.org/downloads/release/python-370/

[REND11] Rendón, E., Abundez, I., Arizmendi, A., & Quiroz, E. M. **(2011). Internal versus external cluster validation indexes**. International Journal of computers and communications, 5(1), 27-34.

[ROJO21] Rojo, G., Klee, C., & Muñoz-Basols, J. **(2021). Introducción a la lingüística de corpus en español**. Routledge.

[RAE21] Rojo, G., & Española, R. A. **Investigaciones fraseológicas y corpus textuales**. Estudios en homenaje a Alfredo Matus Olivier. 2021.

[SANT06] Santos, D. **(2006). What is natural language? Differences compared to artificial languages, and consequences for natural language processing**.

[SRLN18] Srinivasa-Desikan, B. (2018). **Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras**. Packt Publishing Ltd.

[TAKA15] Takahashi, H., Nakagawa, H., & Tsuchiya, T. **(2015). Towards Automatic Requirements Elicitation from Feedback Comments: Extracting Requirements Topics Using LDA**. In SEKE (pp. 489-494). https://ksiresearchorg.ipage.com/seke/seke15paper/seke15paper_103.pdf

[TORR18] Torres, T. V. **(2018). First Contact with Deep Learning: Practical Introduction with Keras**. Kindle Direct Publishing.

[VILL16] Villarroel, L., Bavota, G., Russo, B., Oliveto, R., & Di Penta, M. **(2016, May). Release planning of mobile apps based on user reviews**. In 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE) (pp. 14-24). IEEE.

[WAN15] Wan, T. A. N. G., Jun, H. U., Hui ZHANG, P. W., & Hua, H. E. **(2015). Kappa coefficient: a popular measure of rater agreement**. Shanghai archives of psychiatry, 27(1), 62.

[WINO72] Winograd, T. **(1972). Understanding natural language**. Cognitive psychology, 3(1), 1-191.

[ZOWG05] Zowghi, D., & Coulin, C. **(2005). Requirements elicitation: A survey of techniques, approaches, and tools**. In Engineering and managing software requirements (pp. 19-46). Springer, Berlin, Heidelberg.

ANEXOS

Anexo 1: Creciente número de estudios sobre la elicitación automática de requisitos de software basada en datos dinámicos en los últimos años.

ll